Poznámky ke státnicím

Diskrétní modely a algoritmy, léto 2026

Filip Úradník

11. června 2026

Tento text slouží k mé přípravě k magisterským státnicím z Diskrétních modelů a algoritmů na MFF v létě 2025/2026. Obsahuje výcuc mých poznámek z předmětů a má sloužit jako osnova a body o kterých chci vědět/mluvit. Pro detaily viz moje poznámky na `https://furadnik.github.io/notes`. Na stejné adrese je i aktuální verze tohoto dokumentu, spolu s gitovým repozitářem. Pull requesty jsou vítány.

V textu se celkem volně mixuje čeština a angličtina, za to se omlouvám; cca půlka mých poznámek z předmětů je v angličtině a nechtělo se mi to překládat...

Další zdroje:

- `https://github.com/Pheeck/mff-statnice-grafove-algoritmy`
- `https://tury.codeberg.page/work/#notes`
- `https://kam.mff.cuni.cz/~fiala/tw.pdf`

Obsah

I Úvod do složitosti a vyčíslitelnosti **3**

1 Výpočetní modely **4**
1.1 Turingův stroj . . . 4
1.2 RAM . . . 5
1.3 Rozhodnutelnost . . . 6
1.4 Vyčíslitelnost . . . 7
1.5 Převoditelnost a úplnost . . . 7

2 Základní třídy složitosti **10**

3 Aproximační algoritmy a schémata **15**
3.1 Greedy . . . 15
3.2 Zaokrouhlování a dynamické programování . . . 16
3.3 LP . . . 17
3.4 Náhodnost . . . 17
3.5 Semidefinitní programování . . . 18
3.6 Neaproximovatelnost . . . 19

II Datové struktury **20**

4 Vyhledávací stromy **21**
4.1 (a, b)-stromy . . . 21
4.2 Splay stromy . . . 22
4.3 BB[α] stromy . . . 23

5 Haldy **24**
5.1 Binomiální haldy . . . 24
5.2 Líná a Fibonacciho halda . . . 25

6 Hešování **26**
6.1 Hešovací funkce . . . 27
6.2 Otevřená adresace . . . 28
6.3 Vektory a řetězce . . . 29
6.4 Bloomův filtr . . . 30

III Kombinatorika a teorie grafů **31**

7 Barevnost grafů a její varianty **32**
7.1 Kritické grafy . . . 32
7.2 List Coloring . . . 33
7.3 Degree Choosability . . . 35
7.4 Nowhere-Zero Flows . . . 35
7.5 Fractional Coloring . . . 36

7.6 Věty dokazované metodou náboje . . . 38

8 Grafové minory **39**
8.1 Drawings . . . 40
8.2 Characterization by forbidden substructures . . . 40
8.3 Clique-sums . . . 41

9 Stromová šířka **42**
9.1 Dynamické programování . . . 43
9.2 Courcelle's Theorem . . . 44
9.3 Series-Parallel Graphs . . . 45
9.4 Chordal Graphs . . . 45
9.5 Brambles . . . 46

10 Geometrické reprezentace grafů **48**
10.1 Chordal graphs . . . 48
10.2 Filament Graphs . . . 50
10.3 Maximum Independent Set in IFA . . . 51
10.4 Recognizing CHOR . . . 52

11 Algebraické vlastnosti grafů **53**

12 Teorie párování **55**

13 Ramseyova teorie **56**
13.1 Halesova–Jewettova věta . . . 57

14 Szemerédiho lemma o regularitě **58**
14.1 Finding Subgraphs . . . 60
14.2 Arithmetic Progressions in Dense Sets . . . 61

15 Množinové systémy **63**

IV Polyedrální optimalizace **65**

16 Teorie mnohostěnů **66**
16.1 Afinní prostory . . . 66
16.2 Konvexní množiny . . . 67
16.3 Mnohostěny . . . 68
16.4 Důkaz Minkowskiho–Weylovy věty . . . 69

17 Problém obchodního cestujícího **71**

18 Speciální matice **73**
18.1 Totálně unimodulární matice . . . 73
18.2 Representation of Matroids . . . 74
18.2.1 Binary Matroids . . . 75
18.2.2 Ternary Matroids . . . 75
18.2.3 Regular Matroids . . . 75

19 Celočíselnost **76**

20 Párování a toky v sítích **77**

21 Teorie matroidů **78**
21.1 Basic Operations . . . 79
21.2 Connectivity . . . 80
21.3 Greedy Algorithm . . . 81

21.4 Matroid Intersection 81

22 Elipsoidová metoda **83**

V Grafové algoritmy 85

23 Nejkratší cesty, tranzitivní uzávěr **86**
23.1 Point-to-point shortest paths 88
23.2 All-pairs shortest paths 89

24 Toky v sítích **92**

25 Řezy **95**

26 Párování **97**
26.1 Perfektní párování 98
26.2 Perfektní párování minimální ceny 99

27 Minimální kostry **101**
27.1 Vektory na RAMu 105
27.2 Verifikace minimální kostry 105
27.3 Minimální kostra randomizovaně lineárně 107
27.4 Verifikace koster na PM 107

28 Testování rovinnosti grafů a kreslení do roviny **109**

29 Union-find **112**
29.1 Unions known in advance 112

30 Link-cut stromy **114**

31 Dynamické komponenty souvislosti **116**
31.1 Forests 116
31.2 Fast ET-sequence operations 117
31.3 Non-Forests 117

32 Společní předchůdci ve stromech **119**
32.1 Pointer Machine 120

Okruh I

Úvod do složitosti a vyčíslitelnosti

Otázka 1

Výpočetní modely

Budeme mluvit o dvou výpočetních modelech: Turingově stroji a Random-access machine. V obou případech se jedná spíše o *rodiny modelů*, každý autor používá trochu jinou definici. Všechny definice se na sebe více méně dají převádět pouze s minimálními problémy (např, za cenu mírného zpomalení).

Principem je v obou případech snaha o *formalizování výpočtu* pro účely kvantifikace problémů z hlediska složitosti, nebo splnitelnosti.

1.1 Turingův stroj

Definice 1.1 (Turingův stroj, TM). *Turingův stroj* je $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$, kde

- Q je konečná množina *stavů*,
- Σ je konečná neprázdná množina *vstupních symbolů*,
- $\Gamma \supset \Sigma$ je konečná množina *symbolů pro pásku*, $Q \cap \Gamma = \emptyset$,
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ je (částečná) *přechodová funkce*,
- $q_0 \in Q$ je *počáteční stav*,
- $B \in \Gamma \setminus \Sigma$ je *defaultní symbol prázdné buňky*,
- $F \subseteq Q$ je množina *koncových (přijímajících) stavů*.

Definice 1.2 (Přijímání). TM M

(1) *přijímá* $w \in \Sigma^* \equiv M$ skončí v přijímajícím stavu,

(2) *odmítá* $w \in \Sigma^* \equiv M$ skončí v nepřijímajícím stavu.

Jazyk přijímaných slov je značen $L(M)$. $M(w)\downarrow$ značí že výpočet skončí, $M(w)\uparrow$ že ne.

Definice 1.3 (Rozhodování). Turingův stroj M *rozhoduje* jazyk $L \equiv L(M) = L$ a na všech vstupech se M zastaví.

Fakt 1.4 (Churchova-Turingova teze). Ke každému intuitivnímu algoritmu existuje ekvivalentní TM.

Jedna obvyklá varianta TM je *vícepáskový TM*, který má k pásek.

Definice 1.5 (Vícepáskový TM). Vícepáskový TM má k pásek, z toho jedna vstupní (na té je vstup), a druhá výstupní (z té se případně čte výstup).

Věta 1.6. Ke každému k-páskovému TM existuje jednopáskový TM přijímající stejný jazyk.

Důkaz. Reprezentujeme k pásek jako $2k$-tici, na sudých pozicích budeme ukládat pozici hlavy a na lichých samotný obsah pásek. Tím výrazně nafoukneme velikost páskové abecedy, ale to nám nevadí. Také na konec a začátek popsané části pásky přidáme zarážky.

Při samotném výpočtu pak postupujeme následovně:

(1) Projdeme celou popsanou část, zapamatujeme si symbol pod každou hlavou (ve stavu).

(2) Provedeme krok k-páskového Turingova stroje.

(3) Projdeme opět popsanou část a přesuneme hlavy.

■

1.2 RAM

Definice 1.7 (RAM). RAM se skládá z

(1) neomezené paměti složené z *registrů* $r_1, r_2, \ldots$ přirozených čísel – na začátku nuly,

(2) výpočetní jednotky, která podporuje operace

- načíst konstantu,
- součet, odečtení registrů,
- kopírování, kopírování s nepřímou adresací,
- podmíněný skok,
- čtení vstupu, zápis výstupu.

Definice 1.8 (Rozhodnutelnost RAM). RAM R čte slovo $w \in \Sigma^*$ jako posloupnost čísel – indexů v Σ.

(1) R *přijme* $w \in \Sigma^* \equiv R(w)\downarrow$ a první znak na výstupu je 1,

(2) R *odmítne* $w \in \Sigma^* \equiv R(w)\downarrow$ a první znak na výstupu není 1, nebo je výstup prázdný.

(3) $L(R)$ je jazyk *přijímaných slov*,

(4) $L \subseteq \Sigma^*$ je *částečně rozhodnutelný* $\equiv (\exists R)\, L = L(R)$,

(5) $L \subseteq \Sigma^*$ je *rozhodnutelný* $\equiv$ je částečně rozhodnutelný RAMem který skončí na každém vstupu.

Definice 1.9 (Počítání funkce). RAM R *počítá funkci* $f \equiv$

$$(\forall x) \quad (x \in \operatorname{dom} f \leftrightarrow R(x)\downarrow) \wedge (R(x)\downarrow \rightarrow R(x) = f(x))\,.$$

Poznámka 1.10. Pokud f je řetězcová funkce, pak výstupem jsou indexy do abecedy zakončeny nulou.

Věta 1.11. Ke každému TM existuje ekvivalentní RAM, a naopak.

Důkaz (TM na RAM). V tomto směru je hlavní problém že nemáme oboustranně nekonečnou paměť. To vyřešíme tak, že rozdělíme buňky podle zbytku adres modulo 3 na: nezáporné buňky (od počáteční), záporné buňky a pracovní paměť. V pracovní paměti pak ještě můžeme mít uloženou přechodovou funkci pokud ji nechceme mít zadrátovanou do programu (což bychom ale mohli). Kroky TM lze očividně provést. ■

Důkaz (RAM na TM). Pořídíme si TM s dostatečným množství pásek. Na jedné pásce si budeme držet paměť zakódovanou jako: `## adresa # hodnota ##`. Takto jsme schopni získat z paměti libovolnou hodnotu, i do paměti zapsat. Na zbytku pásek jsme schopni triviálně implementovat všechny operace RAMu. ■

1.3 Rozhodnutelnost

DEFINICE 1.12 (ROZHODNUTELNOST). Jazyk L je

(1) *částečně rozhodnutelný (rekurzivně spočetný)* $\equiv$

$$(\exists M) \quad L = L(M)\,,$$

(2) *rozhodnutelný (rekurzivní)* $\equiv$ je částečně rozhodnutelný TM M který skončí na každém vstupu.

DEFINICE 1.13 (KÓDOVÁNÍ). $\langle X \rangle$ značí *binární řetězec, kód objektu* X.

VĚTA 1.14. Pro každý TM M existuje $\langle M \rangle$.

Důkaz. Můžeme předpokládat že abeceda je očíslovaná, že Σ je prefix Γ a že B je první znak mimo Σ. V kódu si tedy stačí pamatovat tyto údaje a přechodovou funkci. To je triviální v binární soustavě, až na oddělovače. Proto si představujeme že oddělovač je speciální znak `#` a finální kód pak dostaneme nahrazením `0` za `00`, `1` za `01`, `#` za `11`. ■

LEMMA 1.15. Částečně rozhodnutelných jazyků je spočetně mnoho.

Důkaz. Pro každý částečně rozhodnutelný jazyk existuje TM, a TM jsou dle předchozí věty očíslované $\mathbb{N}$. ■

VĚTA 1.16. Existují jazyky, které nejsou částečně rozhodnutelné.

Důkaz. Dle lemmatu 1.15 je částečně rozhodnutelných jazyků spočetně mnoho, ale všech jazyků je nespočetně mnoho, protože je jich tolik jako bin. posloupností. ■

DEFINICE 1.17 (UNIVERZÁLNÍ TM). *Univerzální TM* je $\mathcal{U}$, tž

$$(\forall M)(\forall w) \qquad M(w)\downarrow \leftrightarrow \mathcal{U}(\langle M, w\rangle)\downarrow \quad \wedge \quad M(w) \leftrightarrow \mathcal{U}(\langle M, w\rangle)\,.$$

Dále definujeme $L_u := L(\mathcal{U})$.

VĚTA 1.18. Diagonální jazyk $L_d = \{\langle M\rangle\,;\, \langle M\rangle \notin L(M)\}$ není částečně rozhodnutelný.

Důkaz. Pro spor, nechť M rozhoduje L_d. Pak buď $\langle M\rangle \in L_d$ nebo $\langle M\rangle \notin L_d$, obojí vede ke sporu. ■

VĚTA 1.19. L_u je částečně rozhodnutelný, ale není rozhodnutelný.

Důkaz. Je částečně rozhodnutelný protože může prostě simulovat. Není rozhodnutelný, protože kdyby byl, pak by rozhodoval diagonální jazyk. ■

VĚTA 1.20 (VLASTNOSTI ČR). Nechť jazyky L_1, L_2 jsou (částečně) rozhodnutelné. Pak $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 \cdot L_2$ i L_1^* jsou (částečně) rozhodnutelné.

Důkaz. Nechť M_1 a M_2 (částečně) rozhodují L_1 a L_2. Nechť $w \in \Sigma^*$.

(1) Přijmeme pokud oba přijali.

(2) Přijmeme pokud alespoň jeden přijal. Spustíme výpočty paralelně (střídavě), takže pokud alespoň jeden přijímá, zjistíme to v konečném čase.

(3) Zkusíme střídavě všechna místa kde se slovo dá rozpůlit.

(4) Zkusíme střídavě všechna místa kde se slovo dá rozdělit na k částí (pro $k \in \{1, \dots |w|\}$).

■

VĚTA 1.21 (POSTOVA). L je rozhodnutelný, právě když L i $\overline{L}$ jsou částečně rozhodnutelné.

Důkaz. Pokud L je rozhodnutelný, rozhodnutí $\overline{L}$ je prostě negace.

Pokud L a $\overline{L}$ jsou částečně rozhodnutelné, pak můžeme střídavě spustit oba TM, a jeden určitě v konečném čase skončí. To je naše odpověď. ■

DŮSLEDEK 1.22. Rozhodnutelné jazyky jsou uzavřené na doplňek, částečně rozhodnutelné ne.

VĚTA 1.23 (RICE). Nechť $\mathcal{C}$ je třída částečně rozhodnutelných jazyků, a $L := \{\langle M\rangle ; L(M) \in \mathcal{C}\}$. Pak L je rozhodnutelný, právě když $\mathcal{C}$ je *triviální*, tj. buď prázdná nebo obsahuje všechny částečně rozhodnutelné jazyky.

Důkaz. Pokud je $\mathcal{C}$ triviální, věta platí. Pro spor, nechť $\mathcal{C}$ není triviální, ale existuje $M_{\mathcal{C}}$ který rozhoduje L. Využijeme $M_{\mathcal{C}}$ pro rozhodování L_u.

Nechť $\emptyset \notin \mathcal{C}$ (jinak vezmeme doplněk). Nechť $L' \in \mathcal{C}$ je libovolný jazyk částečně rozhodnutelný pomocí M'.

Definujeme nyní nový Turingův stroj M'' který potom využijeme pro rozhodování $\langle M, x\rangle \in L_u$. Vstup tohoto Turingova stroje označme y.

(1) Pustíme $M(x)$, pokud odmítne, odmítneme.

(2) Pustíme $M'(y)$, pokud přijme, přijmeme, jinak odmítneme.

Pokud $\langle M, x\rangle \in L_u$, pak M'' spustí pro každý vstup $M'(y)$, tedy výsledný jazyk je L' a tedy $\langle M''\rangle \in L$. Pokud $\langle M, x\rangle \notin L_u$, pak M'' nikdy nespustí $M'(y)$, tedy výsledný jazyk je $\emptyset$ a tedy $\langle M''\rangle \notin L$. ■

POZNÁMKA 1.24. Ekvialentně i pro *funkce*.

1.4 Vyčíslitelnost

DEFINICE 1.25. Funkce f je *Turingovsky vyčíslitelná* $\equiv$ existuje TM, který ji počítá, tj. existuje M, tž.

(1) $x \in \operatorname{dom} f \leftrightarrow M(x)\downarrow$,

(2) $(\forall x \in \operatorname{dom} f)(f(x) = M(x))$.

DEFINICE 1.26 (TOTÁLNÍ FUNKCE). Funkce f je *totální* $\equiv \operatorname{dom} f = \Sigma^*$.

VĚTA 1.27. $L \subseteq \Sigma^*$ je rozhodnutelný, právě když jeho charakteristická funkce $\chi_L(x) := [x \in L]$ je vyčíslitelná.

DŮSLEDEK 1.28. Existuje nevyčíslitelná funkce.

1.5 Převoditelnost a úplnost

DEFINICE 1.29 (TURINGOVSKÁ PŘEVODITELNOST). Nechť $A, B \subseteq \Sigma^*$. Pak A je *převoditelný na* B, $A \leq_T B$ $\equiv$ existuje M, tž.

(1) M rozhoduje A,

(2) M se může ptát na $y \in B$.

DŮSLEDEK 1.30.

$$(\forall A \subseteq \Sigma^*) \quad A \leq_{\mathrm{T}} \overline{A}.$$

TVRZENÍ 1.31. Nechť $A \leq_{\mathrm{T}} B$. Pak B je rozhodnutelný implikuje A je rozhodnutelný.

DEFINICE 1.32 (HALITNG PROBLEM). Halting problem rozhoduje, zda se M na vstupu x zastaví. Odpovídá jazyku

$$L_h = \{\langle M, x\rangle \,;\, M(x)\downarrow\}\,.$$

LEMMA 1.33.

$$L_u \leq_\mathrm{T} L_h.$$

Důkaz. Nechť $\langle M, x\rangle$ je na vstupu. Sestrojíme M' tak, aby odsimuloval M, pokud přijme, přijal, a pokud odmítne, zacyklil se. Pak $\langle M, x\rangle \in L_u$ právě když $\langle M', x\rangle \in L_h$. ■

DŮSLEDEK 1.34. L_h je částečně rozhodnutelný, ale není rozhodnutelný.

DEFINICE 1.35 (m-PŘEVODITELNOST). Jazyk A je *m-převoditelný na B*, $A \leq_m B \equiv$ existuje totální vyčíslitelná f, tž.

$$(\forall x) \quad x \in A \leftrightarrow f(x) \in B.$$

POZOROVÁNÍ 1.36. $\leq_m$ je kvaziuspořádání.

POZOROVÁNÍ 1.37.

$$L_u \leq_m L_h.$$

TVRZENÍ 1.38. Nechť $A \leq_m B$. Pak B je (částečně) rozhodnutelný implikuje A je (částečně) rozhodnutelný.

Důkaz. Nechť M (částečně) rozhoduje B. Nechť M' je TM který pro vstup x

(1) sestrojí $y = f(x)$,

(2) spustí $M(y)$.

Toto (částečně) rozhoduje A. ■

DEFINICE 1.39 (PROBLÉM NEPRÁZDNÉHO JAZYKA). Problém neprázdného jazyka má na vstupu M, a rozhoduje, zda $L(M) \neq \emptyset$. Jeho jazyk označíme

$$L_n := \{\langle M\rangle \,;\, L(M) \neq \emptyset\}\,.$$

TVRZENÍ 1.40.

$$L_u \leq_m L_n.$$

Důkaz. Chceme vědět, zda $\langle M, x\rangle \in L_u$. Sestrojíme M' který ignoruje svůj vstup a spustí $M(x)$. Pak $L(M')$ jsou buď všechny řetězce nebo žádný. To zjistí L_n. ■

DŮSLEDEK 1.41. L_n je částečně rozhodnutelný, ale není rozhodnutelný.

DEFINICE 1.42 (ÚPLNOST). Jazyk L je *m-úplný* $\equiv$

(1) L je částečně rozhodnutelný,

(2) pro A částečně rozhodnutelný platí $A \leq_m L$.

TVRZENÍ 1.43. Pokud A je m-úplný, B je částečně rozhodnutelný a $A \leq_m B$, pak B je m-úplný.

Důkaz. Z tranzitivity kvaziuspořádání. ■

VĚTA 1.44. L_u je m-úplný.

Důkaz. Částečně rozhodnutelný je, protože můžeme simulovat.

Pro libovolný jiný jazyk existuje stroj M který ho částečně rozhoduje. Pak stačí definovat $f(x) = \langle M, x \rangle$, a toto je m-převod. ■

Důsledek 1.45. L_h a L_n jsou m-úplné.

Otázka 2

Základní třídy složitosti

Definice 2.1 (Složitost na TM). Nechť M je Turingův stroj. Nechť $f : \mathbb{N} \to \mathbb{N}$ je funkce definovaná pro každý vstup. Pak

(1) *M pracuje v čase $f(n)$* $\equiv$ výpočet M na každém vstupu x o velikosti $|x| = n$ skončí po $\leq f(n)$ krocích,

(2) *M pracuje v prostoru $f(n)$* $\equiv$ výpočet M na každém vstupu x o velikosti $|x| = n$ skončí s využitím $\leq f(n)$ buněk.

Definice 2.2 (Třídy jazyků). Pro $f : \mathbb{N} \to \mathbb{N}$ definujeme třídy

(1) $\mathsf{TIME}(f)$, třída jazyků přijímaných TM v čase $\mathcal{O}(f(n))$,

(2) $\mathsf{SPACE}(f)$, třída jazyků přijímaných TM o prostoru $\mathcal{O}(f(n))$,

Pozorování 2.3. Pro libovolnou $f : \mathbb{N} \to \mathbb{N}$ je

$$\mathsf{TIME}(f) \subseteq \mathsf{SPACE}(f)\,.$$

Definice 2.4 (Deterministické třídy složitosti). Definujeme

(1) $\mathsf{P} := \bigcup_{k\in\mathbb{N}} \mathsf{TIME}\left(n^k\right)$,

(2) $\mathsf{PSPACE} := \bigcup_{k\in\mathbb{N}} \mathsf{SPACE}\left(n^k\right)$,

(3) $\mathsf{EXPTIME} := \bigcup_{k\in\mathbb{N}} \mathsf{TIME}\left(2^{n^k}\right)$,

Fakt 2.5 (Churchova-Turingova teze, složitost). Reálné výpočetní modely lze simulovat Turingovými stroji s pouze polynomiálním zpomalením/nárůstem prostoru.

Důsledek 2.6. Třída P nezávisí na zvoleném výpočetním modelu.

Definice 2.7 (Složitost na RAMu). Nechť R je RAM a $\ell(n)$ určuje cenu uložení čísla n v registru. Pak definujeme složitost operace jako součet $\ell(n)$ pro všechna ne-konstantní čísla n vyskytující se v operaci.

Poznámka 2.8. Většinou se uvažuje buď $\ell(n) = 1$, nebo $\ell(n) = \max\{1, \log_2 n\}$.

Definice 2.9. RAM R *pracuje v čase $f(n)$* $\equiv$ výpočet R na každém vstupu x o velikosti $n = \ell(x)$ skončí se součtem cen operací $\leq f(n)$.

Fakt 2.10 (Cook, Reckhow). Nechť L je rozhodnutelný RAMem R v čase $f(n)$. Pak je rozhodnutelný vícepáskovým TS, který pracuje v čase $F(n)$ pro

$$F \in \begin{cases} \mathcal{O}\left(f^2\right) & \ell(n) = \max\{1, \log_2 n\}\,, \\ \mathcal{O}\left(f^3\right) & \ell(n) = 1. \end{cases}$$

FAKT 2.11. Je-li L rozhodnutelný vícepáskovým TS v $f(n)$, pak je rozhodnutelný jednopáskovým TS v $\mathcal{O}(f^2(n))$.

DŮSLEDEK 2.12. P je třída jazyků rozhodnutelných RAMem v poly čase.

DEFINICE 2.13 (VERIFIKÁTOR). *Verifikátor* jazyka L je algoritmus $V(x, y)$, tž.

$$L = \{x \in \Sigma^*;\ (\exists y \in \Sigma^*)(V(x, y))\}\,.$$

POZNÁMKA 2.14. Časovou složitost verifikátoru měříme vůči $|x|$.

DEFINICE 2.15 (NP). NP je třída jazyků, které mají verifikátor pracující v poly $|x|$.

DEFINICE 2.16 (ČASOVÁ SLOŽITOST NTS). Nedeterministický TS T pracuje v

(1) *čase* $f(n) \equiv$ každý výpočet T nad x délky n skončí po $\leq f(n)$ krocích,

(2) *prostoru* $f(n) \equiv$ každý výpočet T nad x délky n skončí a využije $\leq f(n)$ buněk.

DEFINICE 2.17 (NEDETERMINISTICKÉ TŘÍDY JAZYKŮ). Pro $f : \mathbb{N} \to \mathbb{N}$ definujeme třídy

(1) $\mathsf{NTIME}(f)$, třída jazyků přijímaných nedeterministickým TM v čase $\mathcal{O}(f(n))$,

(2) $\mathsf{NSPACE}(f)$, třída jazyků přijímaných nedeterministickým TM o prostoru $\mathcal{O}(f(n))$,

POZOROVÁNÍ 2.18. Pro každou f platí

$$\mathsf{TIME}(f) \subseteq \mathsf{NTIME}(f)\,,$$
$$\mathsf{SPACE}(f) \subseteq \mathsf{NSPACE}(f)\,.$$

VĚTA 2.19.

$$\mathsf{NP} = \bigcup_{k \in \mathbb{N}} \mathsf{NTIME}\left(n^k\right).$$

Důkaz ($\subseteq$). Nechť $L \in \mathsf{NP}$. Existuje tedy certifikát y délky $p(|x|)$ pro každé x. Definujeme nedeterministický M který nejprve vygeneruje certifikát, a pak ho zkontroluje. ■

Důkaz ($\supseteq$). Nechť L je v čase $p(|x|)$ řešitelný NTS M. To znamená že existuje výpočet, tj. posloupnost konfigurací, které přijímají. Každá konfigurace je navíc dlouhá $\mathcal{O}(p(|x|))$, a celkem je jich $p(|x|)$. Toto dává certifikát. ■

VĚTA 2.20. Pro každou f platí

$$\mathsf{NTIME}(f) \subseteq \mathsf{SPACE}(f)\,.$$

Důkaz. Odsimulujeme NTS M jako deterministický. Na jedné pásce si budeme udržovat $\mathcal{O}(f(|x|))$ znaků, které určují jaký výpočet zrovna děláme. Na dalších $\mathcal{O}(1)$ budeme simulovat výpočty postupně. Každý výpočet trvá $\mathcal{O}(f(|x|))$, tedy nepřekročí prostor $\mathcal{O}(f)$. ■

DŮSLEDEK 2.21.

$$\mathsf{NP} \subseteq \mathsf{PSPACE}.$$

DEFINICE 2.22 (MODEL S MENŠÍM NEŽ LINEÁRNÍM PROSTOREM). Vícepáskový TS M pracuje v prostoru $f(n) \equiv$ na *pracovních páskách* zabere nejvýše $f(n)$ buněk, na vstupní pásku nelze zapisovat, a na výstupní pásce se hlava pohybuje pouze doprava.

DEFINICE 2.23 (LOGARITMICKÉ TŘÍDY). Definujeme $\mathsf{L} := \mathsf{SPACE}(\log_2 n)$ a $\mathsf{NL} := \mathsf{NSPACE}(\log_2 n)$.

Definice 2.24 (NPSPACE). Definujeme

$$\mathsf{NPSPACE} := \bigcup_{k\in\mathbb{N}} \mathsf{NSPACE}(n^k)\,.$$

Definice 2.25 (Graf konfigurací). Graf konfigurací NTS M se vstupem x je $G_{M,x}$, tž.

(1) $V(G_{M,x})$ je množina všech možných konfigurací M se vstupem x,

(2) $E(G_{M,x}) := \{\langle C_1, C_2\rangle\,;\, C_2 \in \delta(C_1)\}$.

Lemma 2.26. Pro každý TM M pracující v prostoru $f \geq \log_2$ existuje $c \in \mathbb{N}$, že pro každé x je

$$n(G_{M,x}) \leq 2^{cf(|x|)}.$$

Důkaz. Máme $\Sigma^{f(|x|)}$ možností jak vypadá páska, $|Q|$ stavů, $f(|x|)$ pozic hlavy, a konečně ještě nejvýše $|x|$ pozic hlavy na vstupní pásce. ■

Věta 2.27. Pro každou funkci $f \geq \log_2$ platí

$$(\forall L \in \mathsf{NSPACE}(f))(\exists c \in \mathbb{N}) \qquad L \in \mathsf{TIME}\big(2^{cf}\big)\,.$$

Důkaz. Graf konfigurací dokážeme procházet v $\mathrm{poly}(n(G_{M,x}))$. Bound plyne z předchozího lemmatu. ■

Důsledek 2.28. Pro $f \geq \log_2$ a g že $f \in o(g)$ je

$$\mathsf{NSPACE}(f) \subseteq \mathsf{TIME}(2^g)\,.$$

Důsledek 2.29.

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{NPSPACE} \subseteq \mathsf{EXPTIME}.$$

Věta 2.30 (Savič). Nechť $f \geq \log_2$. Pak

$$\mathsf{NSPACE}(f) \subseteq \mathsf{SPACE}\big(f^2\big)\,.$$

Důkaz. Procházíme graf konfigurací, aniž bychom ho měli celý v paměti (má $2^{\mathcal{O}(f(|x|))}$, to je moc). Předpokládejme nejprve (nerealisticky), že známe $n = c \cdot f(|x|)$ z lemmatu 2.26. Pak postupujeme následovně (začínáme s $k = n$):

(1) Procházíme všechny konfigurace K postupně (každá je velikosti maximálně n).

(2) Pro každou rekurzivně zjistíme, zda se ze startovní konfigurace dostaneme do K a z K do přijímající konfigurace za 2^{k-1} konfigurací.

(3) Pokud $k = 0$, prostě zkontrolujeme zda se start a konec rovnají, nebo mezi nimi vede hrana v grafu konfigurací.

Pokud neznáme n, začínáme s $n = 1$ a zvětšujeme dokud existuje konfigurace s $n + 1$ políčky do které se pomocí předchozího rekurzivního algoritmu dostaneme.

Evidentně toto správně vyhodnotí jazyk původního nedeterministického stroje. Hloubka rekurze je $\mathcal{O}(f(|x|))$ a v každé úrovni rekurze potřebujeme nejvýše $\mathcal{O}(f(|x|))$ hodnot, tedy jsme v $\mathsf{SPACE}\big(f^2\big)$. ■

Důsledek 2.31.

$$\mathsf{PSPACE} = \mathsf{NPSPACE}.$$

Definice 2.32 (Prostorová konstruovatelnost). Funkce $f : \mathbb{N} \to \mathbb{N}$ je *prostorově konstruovatelná* $\equiv$ $f \geq \log$, a funkce, která 1^n přemění na $f(n)$ v binární reprezentaci, je vyčíslitelná v prostoru $\mathcal{O}(f(n))$.

VĚTA 2.33. Pro každou prostorově konstruovatelnou f existuje $A \subseteq \Sigma^*$ rozhodnutelný v prostoru $\mathcal{O}(f)$, ale ne $o(f)$.

Důkaz. Použijeme diagonalizaci. Konstruujeme N pracující v prostoru $\mathcal{O}(f)$. Budeme chtít aby pro každý M pracující v $o(f)$ prostoru se na každém vstupu tvaru $\langle M\rangle 10^*$ lišil $L(M)$ a $L(N)$. Na vstupu x budeme provádět následující

(1) Pokud $x \neq \langle M\rangle 10^*$, odmítneme.

(2) Spočítáme $f(|x|)$ (z prostorové konstruovatelnosti).

(3) Vyznačíme $f(|x|)$ políček na pásce v obou směrech.

(4) Simulujeme $M(x)$, pokud jdeme mimo vyznačené pole, nebo počet kroků přesáhne $2^{f(|x|)}$, odmítneme.

(5) Znegujeme výsledek M.

Toto zvládneme v prostoru $\mathcal{O}(f)$. Pokud $L \in \mathsf{SPACE}(o(f))$, existuje N pracující v prostoru $g \in o(f)$ rozhodující L. Tedy existuje n, tž. $cg(n) < f(n)$, kde c je z lemmatu 2.26. Pro vstup $\langle M\rangle 10^n$ je tedy určitě počet políček dostatečný, a pokud v kroku (4) zamítneme, tak se N zacyklil (což je spor s tím že rozhoduje L). Tedy dostaneme se do posledního kroku kde se zachováme opačně. ■

DŮSLEDEK 2.34. Pro každou prostorově konstruovatelnou f platí

$$\mathsf{SPACE}(o(f)) \subset \mathsf{SPACE}(f)\,.$$

DŮSLEDEK 2.35.

$$\mathsf{NL} \subset \mathsf{PSPACE} \subset \mathsf{EXPSPACE}.$$

DEFINICE 2.36 (ČASOVÁ KONSTRUOVATELNOST). Funkce $f : \mathbb{N} \to \mathbb{N} \in \Omega(n \log n)$ je *časově konstruovatelná* $\equiv$ funkce, která 1^n přemění na $f(n)$ v binární reprezentaci, je vyčíslitelná v čase $\mathcal{O}(f(n))$.

VĚTA 2.37. Pro každou časově konstruovatelnou f existuje $A \subseteq \Sigma^*$ rozhodnutelný v čase $\mathcal{O}(f)$, ale ne $o\left(\frac{f}{\log f}\right)$.

Důkaz. Použijeme diagonalizaci analogicky jako u prostorové konstruovatelnosti. Konstruujeme N pracující v čase $\mathcal{O}(f)$. Budeme chtít aby pro každý M pracující v $o(f/\log_2(f))$ čase se na každém vstupu tvaru $\langle M\rangle 10^*$ lišil $L(M)$ a $L(N)$. Na vstupu x budeme provádět následující

(1) Pokud $x \neq \langle M\rangle 10^*$, odmítneme.

(2) Spočítáme $f(|x|)$ (z časové konstruovatelnosti).

(3) Sestrojíme počítadlo inicializované $f(|x|)\,/\log_2 f(|x|)$, které budeme brát všude s sebou, v každém kroku (tj. instrukci, ne odsimulovaného kroku M) snížíme jeho hodnotu o 1, pokud dojde do nuly, odmítneme.

(4) Simulujeme $M(x)$, znegujeme jeho výsledek.

Toto zvládneme v čase $\mathcal{O}(f)$ – jeden krok simulace trvá $\mathcal{O}(1)$ a práce s počítadlem trvá $\mathcal{O}(\log f(|x|))$.

Pokud $L \in \mathsf{TIME}(o(f/\log f))$, existuje N pracující v čase $g \in o(f/\log f)$ rozhodující L. Tedy existuje n, tž. $cg(n) < f(n)\,/\log f(n)$, kde c je konstanta z overheadu simulace. Pro vstup $\langle M\rangle 10^n$ je tedy určitě $f(n)\,/\log_2 f(n)$ kroků dostatečných, a na tomto vstupu se tedy jazyky liší. ■

DŮSLEDEK 2.38. Pro každou časově konstruovatelnou f platí

$$\mathsf{TIME}\left(o\left(\frac{f}{\log f}\right)\right) \subset \mathsf{TIME}(f)\,.$$

DŮSLEDEK 2.39.

$$\mathsf{P} \subset \mathsf{EXPTIME}.$$

Důsledek 2.40. Alespoň jedna z inkluzí

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE} \subseteq \mathsf{EXPTIME},$$

a alespoň jedna z inkluzí

$$\mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE}$$

jsou ostré.

Věta 2.41. SAT je NP-úplný.

Věta 2.42. Nezávislá-Množina je NP-úplná.

Otázka 3

Aproximační algoritmy a schémata

Definice 3.1 (Optimalizační problém). *Optimalizační problém* je $\langle \mathcal{I}, \mathcal{F}, f, g \rangle$, kde

(1) $\mathcal{I}$ je množina *vstupů*,

(2) $\mathcal{F}(I)$ je množina *přípustných řešení* pro vstup I,

(3) $f : \mathcal{F}(\mathcal{I}) \to \mathbb{R}$ je účelová funkce, a

(4) g je buď minimalizace nebo maximalizace, určující „směr".

Definice 3.2 (Aproximační algoritmus). *α-aproximační algoritmus* pro optimalizační problém je polynomiální algoritmus, který produkuje řešení v rámci multiplikativního faktoru α od optima.

Definice 3.3 (PTAS). *Polynomiální aproximační schéma* pro optimalizační problém je rodina algoritmů $\{A_\varepsilon\}_\varepsilon$ taková, že pro každé $\varepsilon > 0$ je A_ε $(1+\varepsilon)$-aproximací (pro minimalizaci, $(1-\varepsilon)$ pro maximalizaci).

Definice 3.4 (FPTAS). *Plně polynomiální aproximační schéma* pro optimalizační problém je PTAS, kde každý A_ε je navíc polynomiální v $\frac{1}{\varepsilon}$.

3.1 Greedy

Prvním způsobem jak navrhovat aproximační algoritmy je konstruovat řešení hladově tak, aby vždy maximalizovalo zlepšení. Příkladem je Vertex-Cover.

Příklad 3.5 (Vertex-Cover). Problém *Vertex-Cover* je, pro daný G najít $W \subseteq V$, tž.

$$\min |W| \,; \qquad (\forall e \in E)(e \cap W \neq \emptyset) \,.$$

Algoritmus 3.6 (Greedy Vertex-Cover).
Vstup: Graf G.
Výstup: Aproximační řešení Vertex-Cover.
1: Najdeme libovolné maximální párování M.
2: $W \leftarrow \bigcup M$.

Věta 3.7. Algoritmus 3.6 je 2-aproximace pro Vertex-Cover.

Důkaz. Každá hrana je pokryta z maximality párování. Naopak, každá hrana párování musí být pokryta, tj. alespoň půlka $|W|$ musí být přítomna v optimu. ■

Příklad 3.8 (Metric-TSP). V problému Metric-TSP máme danou množinu vrcholů V a metriku $d : V \times V \to \mathbb{R} \cup \{\infty\}$ a hledáme nejkratší okruh přes všechny vrcholy.

ALGORITMUS 3.9 (GREEDY METRIC-TSP).
Vstup: Instance METRIC-TSP, tj. úplný graf G.
Výstup: Aproximace řešení.

1: Najdeme T minimální kostru G.
2: $V' \leftarrow$ vrcholy jež mají v T lichý stupeň.
3: Najdeme minimální perfektní párování M v $G[V']$.
4: Najdeme Eulerovský tah v $T \cup M$.
5: Vrátíme tah zkrácený tak aby každý vrchol navštívil max. jednou.

VĚTA 3.10. Algoritmus 3.9 je 3/2-aproximace METRIC-TSP.

Důkaz. Minimální kostra má určitě váhu $w(T) \leq \mathsf{opt}$. Zároveň, pokud máme $V' \subseteq V$ sudé velikosti, uvážíme, jak V' prochází optimální řešení (použijeme zkratky), tak dostaneme řešení TSP na $G[V']$, které se dá rozložit na dvě párování, tj. to z nich s menší váhou má určitě nejvýše $\mathsf{opt}/2$, tj. i minimální.

Zkratky které používáme v posledním kroku to můžou jen zlepšit, tedy

$$w \leq w(T) + w(M) \leq \mathsf{opt} + \mathsf{opt}/2 = \tfrac{3}{2}\mathsf{opt}.$$

■

3.2 Zaokrouhlování a dynamické programování

PROBLÉM 3.11 (KNAPSACK). Instance problému *KNAPSACK* je $\langle \langle s_i, v_i \rangle_{i \in I}, B \rangle$ a naším cílem je najít $S \subseteq I$ takovou, že $\sum_{i \in S} v_i$ je maximální a $\sum_{i \in S} s_i \leq B$.

ALGORITMUS 3.12 (DP PRO KNAPSACK).
Vstup: Instance problému KNAPSACK s n předměty.
Výstup: Optimální řešení.

1: $A \leftarrow \{\langle 0, 0 \rangle\}$
2: Pro $j \in \{2, \ldots, n\}$:
3: $\quad A' \leftarrow A$
4: $\quad$ Pro $\langle s, w \rangle \in A'$:
5: $\quad\quad$ Pokud $s + s_j \leq B$:
6: $\quad\quad\quad A \leftarrow A \cup \{\langle s + s_j, w + v_j \rangle\}$
7: $\quad$ Odstraníme z A dominované dvojice.
8: Vrátíme $\max_{\langle s,w \rangle \in A} w$.

VĚTA 3.13. Algoritmus 3.12 počítá optimální hodnotu problému KNAPSACK.

Důkaz. Očividné z konstrukce algoritmu. ■

DŮSLEDEK 3.14. KNAPSACK je pseudo-polynomiální.

ALGORITMUS 3.15 (FPTAS PRO KNAPSACK).
Vstup: Instance KNAPSACK spolu s $\varepsilon > 0$.
Výstup: $(1 - \varepsilon)$-aproximace KNAPSACK.

1: $M \leftarrow \max_i v_i$
2: $\mu \leftarrow \varepsilon M / n$
3: $(\forall i)(v'_i \leftarrow \lfloor v_i / \mu \rfloor)$
4: Spustíme algoritmus 3.12 s hodnotami v'_i, před návratem převedeme hodnoty zpět.

VĚTA 3.16. Algoritmus 3.15 je FPTAS pro KNAPSACK.

Důkaz. Nejprve dokážeme, že hodnota w množiny $A \subseteq [n]$ která vyjde z algoritmu je $(1-\varepsilon)$-aproximace. Nechť $O \subseteq [n]$ je optimum s hodnotou opt. Nechť $i \in [n]$. Pak $v_i - v_i'\mu \leq \mu$. Z toho

$$w \geq \mu \cdot \sum_{i \in A} v_i' \geq \mu \cdot \sum_{i \in O} v_i' \geq \sum_{i \in O} (v_i - \mu) \geq \mathsf{opt} - \mu \cdot n = \mathsf{opt} - \varepsilon M \geq (1-\varepsilon)\,\mathsf{opt}.$$

Algoritmus je navíc polynomiální v n a $\frac{1}{\varepsilon}$, protože je nejvýše n/ε hladin, a tedy po každém kroku hlavní smyčky je tolik maximální velikost A. ■

3.3 LP

Značení 3.17. Značíme

(1) opt *skutečné optimum* instance problému,

(2) opt^* optimum konkrétní *lineární relaxace* problému.

Pozorování 3.18. Pro minimalizační problémy platí

$$\mathsf{opt}(I) \geq \mathsf{opt}^*(I)\,.$$

Definice 3.19 (Integrality gap). Nechť P je minimalizační problém. Definujeme *Integrality gap* jako

$$\gamma(P) := \sup_I \frac{\mathsf{opt}(I)}{\mathsf{opt}^*(I)}.$$

Pro maximalizační problémy ho definujeme jako

$$\gamma(P) := \inf_I \frac{\mathsf{opt}(I)}{\mathsf{opt}^*(I)}.$$

Příklad 3.20 (Vertex-Cover). *Vertex-Cover* je, pro daný graf G, najít

$$\min |C|\,; \qquad C \subseteq V,\ (\forall e \in E)(e \cap C \neq \emptyset)\,.$$

Příklad 3.21 (Relaxace Vertex-Cover). *Relaxací* Vertex-Cover je

$$\min \sum_{v \in V} x_v; \qquad x_v \in [0,1]\,,\ (\forall uv \in E)(x_u + x_v \geq 1)\,.$$

Tvrzení 3.22. Zaokrouhlením optimálního řešení lineární relaxace získáme 2-aproximaci.

Důkaz. Když $x_u + x_v \geq 1$, pak alespoň jeden je alespoň $1/2$, tedy každá podmínka byla splněna. Naopak, když zaokrouhlíme, zvětšíme tak každé x_u nejvýše dvakrát. ■

3.4 Náhodnost

Fakt 3.23. Pokud existuje ρ-aproximace SAT pro $\rho > \frac{7}{8}$, pak $\mathsf{P} = \mathsf{NP}$.

Tvrzení 3.24. Nechť I je instance 3-SAT. Náhodným ohodnocením proměnných získáme v očekávání $\frac{7}{8}$-aproximaci $\mathsf{opt}(I)$.

Důkaz. Pravděpodobnost, že libovolná klauzule je nesplněná je $1 - 2^{-3}$. To dává z linearity střední hodnoty $\frac{7}{8}$-část klauzulí splněných, což je i bound na optimum. ■

Definice 3.25. Náhodné veličiny $\langle X_i \rangle_i$ jsou *3-wise nezávislé* $\equiv$

$$(\forall i,j,k) \qquad X_i, X_j, X_k \text{ jsou nezávislé.}$$

Tvrzení 3.26. Pokud je X 3-wise nezávislé na $\{0,1\}^n$, pak střední podíl splněných klauzulí je $\frac{7}{8}$.

Důkaz. Lze použít stejná technika jako na zcela nezávisle náhodné ohodnocení. ■

Tvrzení 3.27. Nechť $y_0, y_1, y_2 \in \{0,1\}$ jsou iid. Pak pro

$$g(x) = y_0 + y_1 x + y_2 x^2 \bmod 2,$$

jsou veličiny $\langle g(x) \rangle_{x \in [n]}$ 3-wise nezávislé.

Důkaz. Pravděpodobnost že $g(x_1) = a$, $g(x_2) = b$ a $g(x_3) = c$ je soustava 3 rovnic o 3 neznámých, tedy existuje jedna trojice y_0, y_1, y_2. Ta má pravděpodobnost 2^{-3}. ■

Důsledek 3.28. Existuje deterministická $\frac{7}{8}$-aproximace Max-SATu.

3.5 Semidefinitní programování

Značení 3.29. V kontextu Max-Cutu značíme

$$\alpha_0 := \frac{2}{\pi} \cdot \min_{0 \le \theta \le \frac{\pi}{2}} \frac{\theta}{1 - \cos\theta} \approx 0.878...$$

Definice 3.30 (SDP maximálního řezu). Nechť G je graf. Definujeme SDP program Max-Cutu jako

$$\max \sum_{ij \in E} \frac{1 - X_{i,j}}{2}; \qquad X \in \mathbb{R}^{n \times n}, X \succcurlyeq 0, (\forall i \in V)(X_{i,i} = 1)\,.$$

Toto SDP je relaxací Max-Cutu.

Definice 3.31 (Zaokrouhlení nadrovinou). Nechť $u \in \{x \in \mathbb{R}^n;\ \|x\| = 1\}$ a $p \in_R \{x \in \mathbb{R}^n;\ \|x\| = 1\}$. Pak definujeme

$$z_i := \begin{cases} 1 & \text{pokud } p^\top u \ge 0, \\ -1 & \text{jinak.} \end{cases}$$

Věta 3.32. Existuje polynomiální α_0-aproximace Max-Cut.

Důkaz. Nechť $X = V^\top V$ je optimální řešení SDP (resp. ε-optimální pro dostatečně malé ε). Nechť $z_1, \dots, z_n$ je zaokrouhlení nadrovinou. Pak

$$\begin{aligned} \mathbb{E}[\text{velikost řezu}] &= \sum_{uv \in E} \Pr[z_u \text{ a } z_v \text{ jsou zaokrouhleny jinak}] = \sum_{uv \in E} \frac{\arccos\left(v_u^\top v_v\right)}{\pi} \\ &= \sum_{uv \in E} \frac{1 - v_u^\top v_v}{2} \cdot \frac{2}{1 - v_u^\top v_v} \cdot \frac{\arccos\left(v_u^\top v_v\right)}{\pi} \ge \sum_{uv \in E} \frac{1 - v_u^\top v_v}{2} \min_{\theta \in [0, \frac{\pi}{2}]} \frac{2}{\pi} \cdot \frac{\theta}{1 - \cos\theta} \\ &= \alpha_0 \cdot \mathsf{opt}^*. \end{aligned}$$

■

Fakt 3.33. Neexistuje α-aproximace Max-Cut pro $\alpha > \frac{16}{17}$, pokud $\mathsf{P} \neq \mathsf{NP}$.

Předpokládá se, že neexistuje α-aproximace Max-Cut pro $\alpha > \alpha_0$, pokud $\mathsf{P} \neq \mathsf{NP}$.

3.6 Neaproximovatelnost

VĚTA 3.34. Nechť $\varepsilon > 1$ takové, že existuje polynomiální ε-aproximace Obchodního cestujícího v obecném grafu. Pak $\mathsf{P} = \mathsf{NP}$.

Důkaz. Použijeme aproximaci pro nalezení Hamiltonovské kružnice. Definujeme $d : V \times V \to \mathbb{R}^+$ jako

$$d(u,v) = \begin{cases} 1 & \text{pro } uv \in E, \\ |V| \cdot \varepsilon + 1 & \text{jinak.} \end{cases}$$

Pokud existuje Hamiltonovská kružnice, vrátí naše aproximace lepší řešení než $|V| \cdot \varepsilon + 1$, jinak vrátí alespoň $|V| \cdot (\varepsilon + 1)$. Tím pádem aproximace rozhoduje tento NP-úplný problém. ■

Okruh II

Datové struktury

Otázka 4

Vyhledávací stromy

4.1 (a, b)-stromy

Definice 4.1 ((a,b)-strom). Nechť $a \geq 2, b \geq 2a-1$. Pak (a,b)-*strom* je vícecestný vyhledávací strom, kde

(A1) data jsou uložena pouze ve vnitřních vrcholech,

(A2) každý vrchol má a až b synů, s výjimkou kořene, který má 2 až b synů,

(A3) všechny listy jsou ve stejné hloubce.

Lemma 4.2. Hloubka (a,b)-stromu s k klíči je $\mathcal{O}(\log_a n)$ a $\Omega(\log_b n)$.

Důkaz. Chceme maximalizovat hloubku. Tedy předpokládejme, že každý vrchol obsahuje minimální počet vrcholů, tedy $a-1$ (a dva v kořeni). Tedy počet prvků je

$$n \geq \sum_{i=0}^{h} 2 \cdot a^i \geq \sum_{i=0}^{h} a^i \geq a^h.$$

Tedy $h \in \mathcal{O}(\log_a n)$. Podobně z druhé strany pro b. ■

Věta 4.3. Operace Insert, Delete trvají $\mathcal{O}\left(\log n \frac{b}{\log a}\right)$ a Find trvá $\mathcal{O}\left(\log n \frac{\log b}{\log a}\right)$.

Důkaz. Faktor $\mathcal{O}\left(\frac{\log n}{\log a}\right) = \mathcal{O}(\log_a n)$ je z hloubky, a faktor b nebo $\log b$ je čas ztrávený v každém vrcholu. U úprav musíme kontrolovat boundy, a případně měnit prvky nebo slučovat, což trvá až $\mathcal{O}(b)$, kdežto při hledání nás zdržuje jen to že musíme najít kam dál, což je $\mathcal{O}(\log b)$. ■

Věta 4.4. Posloupnost m operací Insert ve zpočátku prázdném (a,b)-stromě změní $\mathcal{O}(m)$ vrcholů.

Důkaz. Každý Insert mění jeden vrchol vložením klíče, a další vrcholy pouze štěpením. Na konci má ale strom $\mathcal{O}(m)$ vrcholů, tedy mohlo se stát nejvýše $\mathcal{O}(m)$ štěpení. ■

Důsledek 4.5. (a,b)-strom lze ze setříděné posloupnosti vybudovat v $\mathcal{O}(n)$ (pokud si pamatujeme nejpravější vrchol).

Věta 4.6. Posloupnost m operací Insert a Delete ve zpočátku prázdném $(a,2a)$-stromě změní $\mathcal{O}(m)$ vrcholů.

Důkaz. Najdeme funkci $f(k)$ vyjadřující potenciál vrcholu s k klíči, a použijeme

$$\Phi = \sum_{\substack{v \text{ vrchol} \\ \text{ne kořen}}} f(\#\text{ klíčů ve } v)\,.$$

Potřebujeme, aby

(1) $|f(i) - f(i+1)| \leq c$ (tj. vložení/odebrání klíče je $\mathcal{O}(1)$),

(2) $f(2a) \geq f(a) + f(a-1) + c + 1$ (tj. štěpení je zadarmo),

(3) $f(a-2) + f(a-1) \geq f(2a-2) + c + 1$ (tj. slučování je zadarmo).

Funguje např.

$$f(k) = \begin{cases} 4 & \text{pro } k = 2a, \\ 2 & \text{pro } k \in \{2a-1,\, a-2\}, \\ 1 & \text{pro } k = a-1, \\ 0 & \text{jinak.} \end{cases}$$

■

4.2 Splay stromy

Definice 4.7 (Splay strom). *Splay strom* je binární vyhledávací strom.

Značení 4.8.

(1) $s(v)$ je *počet vrcholů v podstromě* $T(v)$,

(2) $r(v) := \log_2 s(v)$ je *rank vrcholu* v,

(3) $\Phi(T) := \sum_{v \in V(T)} r(v)$.

Pozorování 4.9. Φ je potenciál.

Definice 4.10 (Operace na Splay stromech). Na Splay stromu T umíme

- Splay, vyrotování pomocí zig, zig-zig, zig-zag,
- Insert, Delete, Find, pomocí jednoho Splay.

Lemma 4.11.

$$2\log(\alpha + \beta) - 2 \leq \log\alpha + \log\beta.$$

Lemma 4.12. Amortizovaná cena Splay je nejvýš $3\,(r'(v) - r(v)) + 1$, kde v je splayovaný vrchol.

Důkaz. Dokážeme, že pro každou rotaci platí

$$A \leq 3\,(r_{i-1} - r_i) + 1,$$

kde člen $+1$ je přítomný pouze u rotace „zig“. Například u „zig-zag“ cesty ywx máme

$$A = 2 + r'(x) - r(x) + r'(y) - r(y) + r'(z) - r(z) \leq 3\,(r'(x) - r(x))\,,$$

kde jediné co jsme používali je monotonie $s(v)$ pro podstromy a lemma 4.11 na disjunktní podstromy, tj. $r'(w) + r'(z)$. Dvojka v ceně je protože skutečná cena jsou 2 rotace.

U operace „zig-zig“ využíváme toho, že

$$r(x) + r'(z) \leq 2r'(x) - 2.$$

■

Věta 4.13. Vykonání m operací Splay na T o n vrcholech má složitost $\mathcal{O}((m+n)\log n)$.

Důkaz. Počáteční potenciál je nejvýš $\mathcal{O}(n \log n)$ a každá operace má $\mathcal{O}(\log n)$ amortizovanou složitost dle lemmatu 4.12. ■

Věta 4.14. Vykonání m operací Insert, Delete, Find na ze začátku prázdném stromě, kdy n je největší dosažený počet vrcholů, má složitost $\mathcal{O}(m \log n)$.

Důkaz. V tomto případě začínáme s potenciálem 0. ■

Fakt 4.15 (Statická optimalita). Nechť $a_1, \ldots, a_n \in X$ je posloupnost přístupů ke všem prvkům v X. Nechť T je statický strom, c_T čas přístupů ke všem prvkům dohromady. Pak časová složitost přístupu Splay stromu je $\mathcal{O}(c_T)$.

Fakt 4.16. Přístup ke všem prvkům v rostoucím pořadí trvá $\mathcal{O}(n)$.

Fakt 4.17 (WS bound). Nechť w_i je počet Findů od posledního hledání a_i. Pak Splay strom hledá v $\mathcal{O}(m \log n + m + \sum_i \log(1 + w_i))$.

4.3 BB[α] stromy

Značení 4.18. Budeme pracovat s binárním vyhledávacím stromem T. Navíc označíme $s(v)$ *počet vrcholů v podstromě* $T(v)$, pro $v \in V(T)$.

Definice 4.19. Nechť $\alpha \in \left(\frac{1}{2}, 1\right)$. Vrchol $v \in V(T)$ je *α-vyvážený* $\equiv$ pro každého jeho syna w

$$s(w) \leq \alpha \cdot s(v)\,.$$

Strom T je *α-vyvážený* $\equiv$ každý jeho vrchol je α-vyvážený.

Věta 4.20. Nechť α je konstantní. Pak α-vyvážený strom s n vrcholy má hloubku $\mathcal{O}(\log n)$.

Důkaz. Počet vrcholů v podstromu klesá exponenciálně s hloubkou (jako $\alpha^i n$ pro hloubku i). To znamená, že maximální hloubka bude řádově $\Theta(\log_\alpha n)$. ■

Definice 4.21. BB[α] strom je BVS, jehož Insert a Delete navíc kontrolují α-vyváženost, a pokud přestane platit, zavolají rebuild od nejvyššího vrcholu, kde přestala α-vyváženost platit.

Věta 4.22. BB[α] stromy mají Insert i Delete amortizovanou složitost $\mathcal{O}(\log n)$.

Důkaz. Zadefinujeme si potenciál

$$\Phi = \sum_{v \text{ vrchol}} \varphi(v)\,,$$

kde $\varphi(v)$ definujeme jako

$$\varphi(v) = \begin{cases} |s(\ell(v)) - s(r(v))| & \text{pokud toto je alespoň 2,} \\ 0 & \text{jinak.} \end{cases}$$

Takový potenciál má tu vlastnost že je zvětšen o $\mathcal{O}(\log n)$ při přidání vrcholu (protože hloubka je $\mathcal{O}(\log n)$), a navíc klesne o $\mathcal{O}(s(v))$ při přestavění. Nově postavený podstrom má totiž všechny $\varphi(v)$ nulové. To je přesně co potřebujeme. ■

Otázka 5

Haldy

Budeme předpokládat, že známe jednoduché haldy, zbytek odvodíme:

	Binární	Pole	k-regulární	Binomiální	Líná binomiální	Fibonacciho
Insert	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}\left(\frac{\log n}{\log k}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
FindMin	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
ExtractMin	$\mathcal{O}(\log n)$	$\mathcal{O}(n)$	$\mathcal{O}\left(\frac{k\cdot\log n}{\log k}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$
Decrease	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}\left(\frac{\log n}{\log k}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
Increase	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}\left(\frac{k\cdot\log n}{\log k}\right)$	$\mathcal{O}(\log n)$	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$
Build	$\mathcal{O}(n)$	$\mathcal{O}(1)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Merge	–	–	–	$\mathcal{O}(\log n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Věta 5.1. Vybudování binární haldy v $\mathcal{O}(n)$ lze provést postupným budováním zespoda.

Důkaz. Nejprve umístíme prvky do haldy libovolně, pak budeme směrem zespoda „bublat“ prvky směrem dolů dokud nebude uspořádání správně. Na hladině i nám to zabere $\mathcal{O}(h-i+1)$, kde h je celková hloubka. Navíc v hloubce i je 2^i prvků. Dostáváme že celkem to zabere

$$\sum_{i=0}^{h-1} 2^i\,(h-i+1) = \sum_{j=1}^{h} 2^{h-j}\,(j+1) = \mathcal{O}(n) + n\sum_{j=1}^{h}\frac{j}{2^j} = \mathcal{O}(n)\,.$$

■

5.1 Binomiální haldy

Definice 5.2 (Binomiální strom). *Binomiální strom řádu k* je zakořeněný strom B_k, kde

(1) B_0 je jen kořen,

(2) B_k je strom jehož kořen má k dětí, což jsou po řadě stromy $B_0, \ldots, B_k$.

Pozorování 5.3. B_k je kostra hyperkrychle $\{0,1\}^k$.

Pozorování 5.4. Počet vrcholů na i-té vrstvě je $\binom{k}{i}$.

Pozorování 5.5. Strom B_k jsou dvě kopie B_{k-1} s vrcholy spojenými hranou.

Pozorování 5.6.

(1) Počet vrcholů B_k je 2^k.

(2) Počet hladin B_k je $k+1$.

(3) Kořen B_k má k dětí, ostatní méně.

Definice 5.7 (Binomiální halda). *Binomiální halda* je les binomiálních stromů setříděných podle řádu. V každém vrcholu je prvek, a prvky jsou setříděny tak, že syn je větší nebo roven otci.

Pozorování 5.8. Struktura binomiální haldy je jednoznačně určena počtem prvků.

5.2 Líná a Fibonacciho halda

Definice 5.9 (Líná BH). Dovolíme víc stromů stejného řádu, konsolidujeme strukturu před každým ExtractMinem.

Lemma 5.10. Amortizovaná cena Merge a Insert je $\mathcal{O}(1)$, zbytek stále $\mathcal{O}(\log n)$.

Důkaz. Definujeme

$$\Phi = \#\text{ stromů haldy.}$$

Insert zvýší potenciál o 1 a Merge jej nezmění (počítáme Φ obou hald najednou). Jediná další změna je ExtractMin, který dělá konsolidaci. Rozdělíme konsolidaci na *kroky*: v každém kroku vezmeme dva stromy, sloučíme je do jednoho, a přidáme na správné místo. Krok tedy trvá $\mathcal{O}(1)$ a sníží potenciál o 1, tedy amortizovaná cena kroku je 0.

Zbytek času konsolidace je jen procházení všech ranků a kontrolování že tam nejsou další stromy, což je $\mathcal{O}(\log n)$ celkem. Celkem tedy ExtractMin má stále $\mathcal{O}(\log n)$. ■

Definice 5.11 (Fibonacciho halda). Při Decrease usekneme podstrom abychom nemuseli až nahoru. Pokud by takto vrchol přišel o více než jedno dítě, usekneme ho taky (definujeme vrcholům „barvu“, černému vrcholu nelze sebrat dítě aniž by se z něj stal následně také kořen).

Lemma 5.12.

$$F_0 + F_1 + \ldots + F_k = F_{k+2} - 1.$$

Lemma 5.13. Ve Fibonacciho haldě má vrchol řádu k ve svém podstromu alespoň F_{k+2} vrcholů.

Důkaz. Indukcí dle k. Pro $k = 0$ je $F_2 = 1$, tedy lemma platí. Obecně, nechť má vrchol v syny $v_1, \ldots, v_k$. Když jsme přidávali vrchol v_k, pak všechny ostatní už měl, tedy vrchol v_k měl řád alespoň $k-1$, a předchozí měli $0, \ldots, k-1$. Mezi tím se mohly všechny řády snížit o 1, tedy teď jsou alespoň

$$0, 0, 1, 2, 3, \ldots, k-2.$$

Z indukčního předpokladu je počet vrcholů v nich

$$F_2, F_2, F_3, F_4, \ldots, F_k.$$

K tomu můžeme přidat $F_0 = 0$ a navíc $F_2 = F_1$. Navíc máme ještě jeden vrchol v, tedy dohromady máme

$$1 + F_0 + F_1 + F_2 + \ldots + F_k = F_{k+2}.$$

■

Věta 5.14. Fibonacciho halda má všechno amortizovaně $\mathcal{O}(1)$ až na ExtractMin, který je $\mathcal{O}(\log n)$.

Důkaz. Přidáme počet černých vrcholů do potenciálu:

$$\Phi = (\#\text{ stromů}) + 2 \cdot (\#\text{ černých vrcholů}).$$

Když odtrháváme, vzroste tím počet stromů o 1, ale zároveň uděláme z černého vrcholu bílý, takže kaskádované odtrhávání černých vrcholů je zadarmo. Poslední vrchol zůstane černý, to zvýší potenciál o 2. Zbytek funguje stále stejně. Využíváme toho, že $F_k \sim \exp(k)$, tedy stále máme logaritmické ranky stromů. ■

Otázka 6

Hešování

Značení 6.1. Označíme

(1) $\mathcal{U}$ *universum*, obvykle čísla $[u]$,

(2) $\mathcal{B} := [m]$ *přihrádky*,

(3) $X \subseteq \mathcal{U}$ množinu *vybraných prvků* velikosti n.

Definice 6.2. Systém funkcí $\mathcal{H}$ je *c-universální* $\equiv$

$$(\forall x, y \in \mathcal{U}, x \neq y) \quad \Pr_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{c}{m}.$$

Lemma 6.3. Nechť $\mathcal{H}$ je c-universální a $y \in \mathcal{U} \setminus X$. Pak

$$\mathop{\mathbb{E}}_{h \in \mathcal{H}}[\#x \in X;\, h(x) = h(y)] \leq \frac{cn}{m}.$$

Věta 6.4. Průměrná složitost Find, Insert, Delete v hešovací tabulce je $\mathcal{O}\left(\frac{cn}{m}\right)$.

Definice 6.5 ((k,c)-nezávislost). Systém funkcí $\mathcal{H}$ je (k,c)-*nezávislý* $\equiv$

$$(\forall x_1, \ldots, x_k \in \mathcal{U}, x_i \neq x_j)(\forall b_1, \ldots, b_k \in \mathcal{B}) \quad \Pr_{h \in \mathcal{H}}[(\forall i)(h(x_i) = b_i)] \leq \frac{c}{m^k}.$$

Značení 6.6. Pro systém funkcí $\mathcal{H}$ označíme $\mathcal{H} \bmod k := \{h \bmod k;\, h \in \mathcal{H}\}$.

Lemma 6.7. Nechť $\mathcal{H}$ je (k,c)-nezávislý systém z $\mathcal{U}$ do $\mathcal{B}$ a $n \in \mathbb{N}$ tž. $m \geq 2kn$. Pak $\mathcal{H} \bmod n$ je $2c$-univerzální a $(k, 2c)$-nezávislý z $\mathcal{U}$ do $[n]$.

Důkaz. Nechť $\mathcal{H}' = \mathcal{H} \bmod n$.

Pro univerzalitu, nechť $x, y \in \mathcal{U}$ jsou různé. Pak

$$\Pr_{h' \sim \mathcal{H}}[h'(x) = h'(y)] = \Pr_{h \sim \mathcal{H}}[h(x) \equiv h(y) \bmod n] \leq \sum_{u \equiv v} \Pr_{h \sim \mathcal{H}}[h(x) = u,\, h(y) = v]\,.$$

Ale počet kongruentních dvojic $\langle u, v \rangle$ je nejvýš $m \cdot \left\lceil \frac{m}{n} \right\rceil \leq m \cdot \frac{m+n-1}{n} \leq \frac{2m^2}{n}$. Dostáváme

$$\sum_{u \equiv v} \Pr_{h \sim \mathcal{H}}[h(x) = u,\, h(y) = v] \leq \sum_{u \equiv v} \frac{c}{m^2} \leq \frac{2m^2}{n} \cdot \frac{c}{m^2} = \frac{2c}{n}.$$

Pro nezávislost, nechť $x_1, \dots, x_k \in \mathcal{U}$ a $v_1, \dots, v_k \in [n]$. Pak

$$\Pr_{h' \sim \mathcal{H}'}[(\forall i)(h'(x_i) = v_i)] = \sum_{u_i \equiv v_i} \Pr_{h \sim \mathcal{H}}[(\forall i)(h(x_i) = u_i)] \leq \sum_{u_i \equiv v_i} \frac{c}{m^k} \leq \left\lceil \frac{m}{n} \right\rceil^k \cdot \frac{c}{m^k} \leq \left(\frac{m+n-1}{n} \right)^k \cdot \frac{c}{m^k}$$
$$= \left(\frac{m+n-1}{m} \right)^k \cdot \frac{c}{n^k} \leq \frac{2c}{n^k},$$

kde poslední nerovnost využívá $e^x \geq 1 + x$. ■

6.1 Hešovací funkce

DEFINICE 6.8. Nechť $\mathcal{U} = [p]$ pro $p \in \mathbb{P}$. Definujeme množinu *lineárních hešovacích funkcí* jako

$$\mathcal{L} := \{h_{a,b};\, a, b \in \mathcal{U}\}, \qquad h_{a,b} : x \mapsto ax + b \bmod p.$$

VĚTA 6.9. Systém $\mathcal{L} \bmod m$ (pro $m \leq p$) je 2-universální, $(2,4)$-nezávislý, a navíc, pokud $p \geq 4m$, pak je $(2,2)$-nezávislý.

Důkaz. $\mathcal{L}$ je prostě uniformně náhodná permutace na prvcích. Ptát se na to kam se zahešuje x je tedy ekvivalentní ptát se na uniformně náhodný prvek $r \in [p]$.

(1) 2-univerzalita:

$$\Pr_{r,s}[r \equiv s \bmod m] = \frac{1}{p^2} \cdot p \cdot \left\lceil \frac{p}{m} \right\rceil \leq \frac{1}{p^2} \cdot p \cdot \frac{p+m-1}{m} \leq \frac{2}{m}.$$

(2) $(2,4)$-nezávislost:

$$\Pr_{r,s}[r \equiv y_1,\, s \equiv y_2 \bmod m] = \Pr_r[r \equiv y_1 \bmod m] \Pr_s[s \equiv y_2 \bmod m] = \left(\frac{1}{p} \cdot \left\lceil \frac{p}{m} \right\rceil \right)^2 \leq \frac{4}{m^2}.$$

(3) $\mathcal{L}$ je triviálně $(2,1)$-nezávislý. Z lemmatu 6.7 je tedy $\mathcal{L} \bmod m$ $(2,2)$-nezávislý pro $p \geq 4m$.

■

DEFINICE 6.10. Definujeme $\mathcal{P}_k$ systém z $\mathbb{Z}_p$ do $\mathbb{Z}_p$ jako

$$\mathcal{P}_k := \left\{h_t;\, t \in \mathbb{Z}_p^k\right\}, \qquad h_t : x \mapsto \sum_{i=0}^{k-1} x^i t_i.$$

VĚTA 6.11. $\mathcal{P}_k$ je $(k,1)$-nezávislý, a tedy $\mathcal{P}_k \bmod m$, když $p \geq 2km$, je $(k,2)$-nezávislý.

Důkaz. Polynom stupně k je jednoznačně určen k body. Proto pro $y_1, \dots, y_k \in \mathbb{Z}_p$ existuje právě jeden polynom, který dává $p(x_i) = y_i$. To je to co chce $(k,1)$-nezávislost. ■

POZNÁMKA 6.12. Výběr funkce stojí $\Theta(k)$.

DEFINICE 6.13 (MULTIPLY-SHIFT). Nechť $w \in \mathbb{N}$ a $\ell \leq w$. *Multiply-shift* jsou funkce $\mathcal{M}$ z $\mathcal{U} = [2^w]$ do $\mathcal{B} = \left[2^\ell\right]$ definované jako

$$\mathcal{M} := \{\mu_a;\, a \in [2^w] \text{ je liché}\},$$

kde $\mu_a(x)$ vezme *horních ℓ bitů ze spodních w bitů z ax*.

FAKT 6.14. $\mathcal{M}$ je 2-universální.

DEFINICE 6.15 (TABELAČNÍ HEŠOVÁNÍ). Nechť $k, t, \ell \in \mathbb{N}$. Tabelační hešovací funkce jsou

$$\mathcal{T} := \left\{ \tau_{T_1,\dots,T_t};\, T_i \in \left[2^\ell\right]^{2^k} \right\},$$

kde $\tau_{T_1,\dots,T_t}(x)$ rozdělí x bitově na t složek délky k, každou zaindexuje do příslušné T_i, a výsledky XORuje.

POZOROVÁNÍ 6.16. Položek v tabulkách je $t \cdot 2^k$, výběr funkce trvá $\Theta\big(t \cdot 2^k\big)$, vyhodnocení $\Theta(t)$.

TVRZENÍ 6.17. Tabelace pro $t \geq 2$ je 3-nezávislá (ale ne 4-nezávislá).

DEFINICE 6.18 (KUKAČČÍ HEŠOVÁNÍ). *Kukaččí hešování* udržuje dvě tabulky a dvě funkce h_1, h_2. Platí

(1) každá přihrádka obsahuje nejvýše jeden prvek,

(2) pokud struktura obsahuje x, pak je v $h_1(x)$ nebo $h_2(x)$,

(3) při INSERTu v případě obsazení políček vyhodíme prvek a zkusíme ho zahešovat druhou funkcí; vzdáme to po $\lceil 6 \log m \rceil$ pokusech a přehešujeme.

FAKT 6.19. Nechť $\mathcal{H}$ je $\lceil 6 \log n \rceil$ nezávislý (nebo tabelace), $\varepsilon > 0$ a $m \geq (2+\varepsilon)\, n$. Pak očekávaná amortizovaná složitost INSERTu je $\mathcal{O}(1)$.

6.2 Otevřená adresace

DEFINICE 6.20 (PŘÍSTUPOVÁ POSLOUPNOST). *Přístupová posloupnost* je $f : \mathcal{U} \times \mathcal{B} \to \mathcal{B}$, tž. $(\forall x)\, f(x, -)$ je permutace na $\mathcal{B}$.

DEFINICE 6.21 (OTEVŘENÁ ADRESACE). Otevřená adresace používá přístupovou posloupnost f, a

(1) FIND hledá podle f dokud nenajde nebo nenarazí na prázdnou buňku,

(2) INSERT přidá do první prázdné buňky dle f,

(3) DELETE dělá pomníčky.

DEFINICE 6.22 (LINEÁRNÍ PŘIDÁVÁNÍ). *Lineární přidávání* používá

$$f(x, i) := (h(x) + i) \bmod m,$$

pro nějakou $h \in \mathcal{H}$.

POZNÁMKA 6.23. Narozdíl od hešování se seznamy v buňkách se u otevřené adresace rychle zhoršuje časová složitost když je tabulka moc plná.

POZNÁMKA 6.24. Lineární přidávání je dobré pro cache – sekvenční přístup.

ZNAČENÍ 6.25. Označíme jako

(1) „běh R“ maximální souvislý úsek v tabulce,

(2) „blok B velikosti 2^t“ posloupnost buněk $\{c2^t, \dots, (c+1)\, 2^t - 1\}$ pro nějaké $c \in \mathbb{N}$.

FAKT 6.26. Nechť $m \geq (1+\varepsilon)\, n$. Pak

(1) pokud h je úplně náhodná, $\mathbb{E}[\text{čas FINDu}] \in \mathcal{O}\big(\frac{1}{\varepsilon^2}\big)$,

(2) pokud $\mathcal{H}$ je 5-nezávislý, $\mathbb{E}[\text{čas FINDu}] \in \mathcal{O}\big(\frac{1}{\varepsilon^{13/6}}\big)$,

(3) existuje $\mathcal{H}$ 4-nezávislý, tž. $\mathbb{E}[\text{čas FINDu}] \in \Omega(\log n)$,

(4) existuje $\mathcal{H}$ 2-nezávislý, tž. $\mathbb{E}[\text{čas FINDu}] \in \Omega(\sqrt{n})$,

(5) pro tabelaci je $\mathbb{E}[\text{čas FINDu}] \in \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$.

FAKT 6.27 (ČERNOVOVA NEROVNOST). Nechť $X_1, \dots, X_n \in \{0,1\}$ jsou nezávislé náhodné veličiny, $X = \sum_i X_i$, $\mu := \mathbb{E}\, X$ a $c > 1$. Pak

$$\Pr[X > c\mu] < \left(\frac{e^{c-1}}{c^c}\right)^{\mu}.$$

DEFINICE 6.28. Blok B velikosti 2^t je *kritický* $\equiv$ bylo do něj zahešováno (ne nutně uloženo) $\geq \frac{2}{3}2^t$ prvků.

LEMMA 6.29. Pro blok B velikosti 2^t platí

$$\Pr[B \text{ kritický}] \leq q^{2^t}, \qquad q = \left(\frac{e}{4}\right)^{1/3} < 1.$$

LEMMA 6.30. Pro R běh délky $\geq 2^{t+2}$ a $B_0, \dots, B_3$ první 4 bloky v R platí, že alespoň jeden z B_i je kritický.

LEMMA 6.31. Nechť $x \in \mathcal{U}$, R běh obsahující $h(x)$, $|R| \in \left[2^{t+2}, 2^{t+3}\right)$. Nechť B_i je blok velikosti 2^t obsahující $h(x)$. Pak alespoň jeden z bloků $B_{i-8}, \dots, B_{i+3}$ je kritický.

VĚTA 6.32. Existuje c, že pro všechna $m = 2^k$ a $n \leq \frac{m}{3}$ a h plně náhodné je

$$\mathbb{E}[\text{počet buněk navštívených při Findu}] \leq c.$$

6.3 Vektory a řetězce

DEFINICE 6.33 (SKALÁRNÍ SOUČIN). Definujeme $\mathcal{S}$ systém z $\mathcal{U} = \mathbb{Z}_p^d$ do $\mathbb{Z}_p$, kde $p \in \mathbb{P}$, jako

$$\mathcal{S} := \left\{h_a;\, a \in \mathbb{Z}_p^d\right\}, \qquad h_a : x \mapsto x^\top a.$$

TVRZENÍ 6.34. $\mathcal{S}$ je 1-universální.

LEMMA 6.35. Nechť $\mathcal{F}$ je c-universální z $\mathcal{U}$ do $[r]$, $\mathcal{G}$ je $(2,d)$-nezávislý z $[r]$ do $[m]$. Pak (multimnožina) $\mathcal{H} = \mathcal{G} \circ \mathcal{F} := \{g \circ f;\, f \in \mathcal{F}, g \in \mathcal{G}\}$ je $(2,c')$-nezávislý, kde

$$c' = \left(\frac{cm}{r} + 1\right) d.$$

VĚTA 6.36. $\mathcal{L} \circ \mathcal{S}$ je $(2,8)$-nezávislý, a pokud $p \geq 4m$, je $(2,2.5)$-nezávislý, s volbou i vyhodnocením v $\Theta(d)$.

DEFINICE 6.37 (POLYNOMY). Definujeme $\mathcal{P}_*$ systém z $\mathbb{Z}_p^*$ do $\mathbb{Z}_p$, kde $p \in \mathbb{P}$, jako

$$\mathcal{P}_* := \{r_a;\, a \in \mathbb{Z}_p\}, \qquad r_a : x \mapsto \sum_i x_i a^i.$$

TVRZENÍ 6.38. $\mathcal{P}_*$ je d-universální, kde d je maximální délka řetězce (nebo dimenze pokud hešujeme vektory), s volbou v čase $\Theta(1)$ a vyhodnocením v $\Theta(d)$.

VĚTA 6.39. Nechť $p \geq 4dm$. Pak $\mathcal{L} \circ \mathcal{P}_*$ je $(2,2.5)$-nezávislý.

6.4 Bloomův filtr

DEFINICE 6.40 (BLOOMŮV FILTR). *Bloomův filtr* je datová struktura parametrizovaná rodinou $\mathcal{H}$. Drží si bitové pole A velikosti m, inicializované nulami, a $h \in_R \mathcal{H}$. Má operace

(1) INSERT, která nastaví $A[h(x)] \leftarrow 1$.

(2) FIND, která pro x odpoví *ano*, pokud $A[h(x)] = 1$.

POZOROVÁNÍ 6.41. Pokud Bloomův filtr odpoví ne, je to pravda.

TVRZENÍ 6.42. Pokud $\mathcal{H}$ je 1-universální, $x_1, \ldots, x_n \in \mathcal{U}$ jsou různé, vložené do Bloomova filtru. Nechť $y \in \mathcal{U}$ je různé od všech x_i. Pak

$$\Pr[\text{FIND}(y) = \text{ano}] \leq \frac{n}{m}.$$

DEFINICE 6.43 (VÍCEPÁSMOVÝ FILTR). *Vícepásmový filtr* je datová struktura parametrizovaná $k, m, h_1, \ldots, h_k$, kde h_i jsou nezávislé hešovací funkce. Udržuje si bitová pole $A_1, \ldots, A_k$, inicializované nulami, a má operace

(1) INSERT, která nastaví $A_i[h_i(x)] \leftarrow 1$ pro všechna i.

(2) FIND, která pro x odpoví *ano*, pokud $(\forall i)(A_i[h_i(x)] = 1)$.

VĚTA 6.44. Pro $k := \left\lceil \log_2 \frac{1}{\varepsilon} \right\rceil$ a $m := 2n$ je $\Pr[\text{false positive}] \leq \varepsilon$.

POZNÁMKA 6.45. Nechť $\mathcal{H}$ jsou plně náhodné. Pak toto nastavení parametrů je až na konstantu optimální vůči využité paměti $m \cdot k$.

Okruh III

Kombinatorika a teorie grafů

Otázka 7

Barevnost grafů a její varianty

Značení 7.1. Pro graf G definujeme

(1) $\chi(G)$ *chromatické číslo* G,

(2) $\delta(G)$ minimální stupeň, $\overline{\delta}(G)$ průměrný stupeň,

(3) $n(G), m(G), f(G)$ počet vrcholů, hran, stěn (pokud definováno), někdy jen prostě n, m.

Definice 7.2. *Obvod* (girth) grafu G je délka nejkratšího cyklu v G.

Definice 7.3. Graf G je *d-degenerovaný* $\equiv$

$$(\forall H \subseteq G) \qquad \delta(H) \leq d.$$

Definice 7.4. Graf G je *perfektní* $\equiv$

$$\omega(G) = \chi(G).$$

7.1 Kritické grafy

Definice 7.5. Graf G je *$(c+1)$-kritický* $\equiv$

$$\chi(G) = c+1 \qquad \wedge \qquad (\forall H \subset G)\ \chi(H) \leq c.$$

Pozorování 7.6. C_{2n+1} is 3-critical and K_n is n critical for any $n \geq 1$.

Pozorování 7.7. For each $(c+1)$-critical graph G,

$$\delta(G) \geq c.$$

Definice 7.8 (Genus). The *genus* of a surface Σ is

$$g(\Sigma) = \begin{cases} g & \Sigma \cong \Sigma_g, \\ 2g & \Sigma \cong \Pi_g. \end{cases}$$

Fakt 7.9 (Euler's fle). For G drawable on Σ, it holds

$$m(G) \leq n(G) + f(G) + g(\Sigma) - 2.$$

Lemma 7.10. Let Σ be a surface of sufficiently large genus g. Let G be drawable of Σ, also sufficiently large. Then

$$\overline{\delta}(G) \leq 6 + \frac{6g-12}{n}.$$

Důkaz. Algebraická úprava Eulerovy formule s tím že $2m \geq 3f$. ■

VĚTA 7.11. Let $c \geq 7$. Let G be $(c+1)$-critical, drawable on Σ. Then

$$n(G) \leq \frac{6g(\Sigma) - 12}{c - 6}.$$

Důkaz. Kombinace lemmatu 7.10 s pozorováním 7.7. ■

DŮSLEDEK 7.12. Let $c \geq 7$. For each Σ, there are finitely many $(c+1)$-critical graphs drawable on Σ.

FAKT 7.13.

(1) For triangle-free graphs, there are only finitely $(c+1)$-critical graphs drawable on Σ for $c \geq 4$.

(2) For girth 6, there are only finitely $(c+1)$-critical graphs drawable on Σ for $c \geq 3$.

(3) Other combinations are not bounded, but at least polynomial (except for girth 3, and 4 colors, which is open).

7.2 List Coloring

DEFINICE 7.14 (LIST ASSIGNMENT). Let G be a graph. A *list assignment* is a function $L : V \to \mathcal{P}(\mathbb{N})$. An *$L$-coloring* is then a proper coloring φ, such that

$$(\forall v) \qquad \varphi(v) \in L(v)\,.$$

DEFINICE 7.15 (LIST CHROMATIC NUMBER). A *list chromatic number* of G is $\chi_\ell(G) := \min k$, such that every list assignment L where each list is of size at least k has an L-coloring.

POZOROVÁNÍ 7.16.

$$\chi_\ell(G) \geq \chi(G)\,.$$

POZOROVÁNÍ 7.17. If G is d-degenerate, then

$$\chi_\ell(G) \leq d + 1.$$

POZOROVÁNÍ 7.18.

$$\chi_\ell(C_n) = \chi(C_n)\,.$$

POZOROVÁNÍ 7.19.

$$\chi_\ell(K_{n,n^n}) > n.$$

FAKT 7.20. If G is planar, then

$$\chi_\ell(G) \leq 5,$$

and this is optimal.

LEMMA 7.21. Let $p(x) \neq 0$ be a polynomial in $x_1, \ldots, x_n$. Let d_i be the degree of x_i in $p(x)$. Then for any $S_1, \ldots, S_i \subseteq \mathbb{C}$, such that $|S_i| > d_i$ for all i, there exists $s \in \prod_i S_i$, such that

$$p(s) \neq 0.$$

DEFINICE 7.22 (GRAPH POLYNOMIAL). Let G be a graph, $\overline{G}$ its orientation. Then the *graph polynomial* of $\overline{G}$ is

$$P_{\overline{G}}(x) := \prod_{uv \in E(\overline{G})} (x_v - x_u)\,.$$

LEMMA 7.23.

$$P_{\overline{G}}(c) \neq 0 \leftrightarrow c_1, \ldots, c_n \text{ is a proper coloring of } G.$$

VĚTA 7.24 (ALON-TARSI METHOD). Let G be a graph, $\overline{G}$ its orientation. Let $d \in \mathbb{N}^n$ and L a list assignment for G, such that $|L(v_i)| > d_i$ for all i. Then if the coefficient of $x_1^{d_1} \cdot \ldots \cdot x_n^{d_n}$ in $P_{\overline{G}}$ is non-zero, then G is L-colorable.

Důkaz. BÚNO je každé $L(v_i)$ velikosti $d_i + 1$. Definujeme polynom

$$p_i(x) = \prod_{c \in L(v_i)} (c - x)\,.$$

Pak platí $p_i(c) = 0$ pro všechna $c \in L(v_i)$. Definujme ještě polynom $q_i(x) = x^{d_i+1} - p_i(x)$. Tento polynom má stupeň nejvýše $d_i + 1$, a pokud dosadíme $c \in L(v_i)$, dostaneme c^{d_i+1}.

Definujme P tak že opakovaně substituujeme $x_i^{d_i+1}$ za $q_i(x_i)$. Tento výsledný P má tedy pro každé x_i stupeň nejvýše d_i. Navíc, pro $c_1, \ldots, c_n \in L(v_1), \ldots, L(v_n)$ je

$$P(c_1, \ldots, c_n) = P_{\overline{G}}(c_1, \ldots, c_n)\,.$$

Navíc, každý monom v $P_{\overline{G}}$ má stejný součet stupňů, to jest $m(G)$, a substituce vytváří jen monomy menšího součtu stupňů, tedy stupeň u $x_1^{d_1} \cdot \ldots \cdot x_n^{d_n}$ je stejný v P i v $P_{\overline{G}}$. Dostáváme, že $P \neq 0$, a dle lemmatu 7.21 existuje ohodnocení c, pro které $P(c) \neq 0$. ■

DEFINICE 7.25. Let G be a graph, $\overline{G}$ its orientation. Let $\overline{G}'$ be another orientation, differing from $\overline{G}$ in p edges. The *sign* of $\overline{G}'$ wrt $\overline{G}$ is $\operatorname{sgn}_{\overline{G}} \overline{G}' := (-1)^p$.

TVRZENÍ 7.26. Let G be a graph, $\overline{G}$ its orientation. Let $\mathbb{O}_G(d)$ be the set of orientations of G, where v_i has in-degree d_i. Then the absolute value of the coefficient of $x_1^{d_1} \cdot \ldots \cdot x_n^{d_n}$ in $P_{\overline{G}}$ is

$$\left| \sum_{\overline{G}' \in \mathbb{O}_G(d)} \operatorname{sgn}_{\overline{G}} \overline{G}' \right|.$$

Důkaz. Když roznásobíme definici $P_{\overline{G}}$, každý člen odpovídá tomu že jsme z jedné hrany zvolili jeden konec. Každý člen tedy odpovídá orientaci kdy vybraný vrchol je koncový.

Abychom dostali $\pm x_1^{d_1} \cdot \ldots \cdot x_n^{d_n}$, musíme vybrat x_i přesně d_i-krát. Orientace má tedy vstupní vrcholy $d_1, \ldots, d_n$. Mínus je na každé hraně, která neodpovídá $\overline{G}$. Z toho plyne tvrzení. ■

DŮSLEDEK 7.27. Let G be a graph, $\overline{G}$ its orientation, where v_i has in-degree d_i. Let $\mathcal{E}$ be the set of subsets of edges forming an Eulerian subgraph. Then the absolute value of the coefficient of $x_1^{d_1} \cdot \ldots \cdot x_n^{d_n}$ in $P_{\overline{G}}$ is

$$\left| \sum_{X \in \mathcal{E}} (-1)^{|X|} \right|.$$

Důkaz. Stupně musí být zachovány. ■

DŮSLEDEK 7.28. Let G be a *bipartite* graph, $\overline{G}$ its orientation, where v_i has in-degree d_i. Let L be a list assignment with $L(v_i) > d_i$. Then G is L-colorable.

Důkaz. V bipartitním grafu je každý eulerovský podgraf sudé velikosti. ■

7.3 Degree Choosability

DEFINICE 7.29. A graph G is *degree-choosable* $\equiv$ G is L-colorable for all L satisfying

$$(\forall v \in V) \qquad |L(v)| \geq \deg v.$$

DEFINICE 7.30 (GALLAI TREE). A G is a *Gallai tree* $\equiv$ it has a decomposition into 2-connected blocks, where

(1) each block is either a clique or an odd cycle,

(2) the blocks are connected via vertices, forming a tree.

FAKT 7.31 (BROOKS). For each connected $G \ncong K_k, C_{2k+1}$,

$$\chi(G) \leq \Delta(G)\,.$$

VĚTA 7.32. A connected graph is not degree-choosable, iff it is a Gallai tree.

Důkaz. Pokud je to Gallai tree, není degree-choosable, protože jinak bychom dostali spor s Brooksovou větou při vhodně zvoleném L.

Naopak, pokud není degree-choosable, použijeme indukci dle n. Pro 2-souvislý graf to platí z Brooksovy věty. Pokud G není 2-souvislý, nechť B je blok 2-souvislosti. Chceme dokázat, že B je klika nebo lichý cyklus.

Vezmeme jiný blok B' (G není 2-souvislý, tj. má alespoň dva bloky), z něj vybereme nějaký vrchol (ne-řezový), tomu předurčíme barvu a odstraníme ji z listů jeho sousedů. Vznikne nový graf G', který má stále B jako blok, a není obarvitelný upraveným L. Z IP plyne že B je klika nebo lichý cyklus. ■

7.4 Nowhere-Zero Flows

DEFINICE 7.33 (DUAL GRAPH). A *dual graph* of a planar (multi)graph $G = \langle V, E\rangle$ is a (multi)graph $G^* = \langle V^*, E^*\rangle$, where

(1) V^* are the faces of a drawing of G, and

(2) E^* are taken from the edges of the original graph, connecting the two faces they connect when G is drawn on the plane.

ZNAČENÍ 7.34. By $\mathcal{A}$, I will denote an arbitrary finite Abelian group.

ZNAČENÍ 7.35. For a multiset of nondirected edges E, I will denote by $\hat{E}$ the multiset of both possible directions of all edges in E.

DEFINICE 7.36. An $\mathcal{A}$-flow in G is $f : \hat{E} \to \mathcal{A}$, such that

$$f(ab) = -f(ba)\,, \qquad (\forall ab \in E)$$
$$\sum_{e \in \delta^-(v)} f(e) = 0.$$

DEFINICE 7.37. An $\mathcal{A}$-flow is *nowhere-zero* $\equiv$

$$(\forall e \in E) \qquad f(e) \neq 0.$$

POZOROVÁNÍ 7.38. Let f be an $\mathcal{A}$-flow in G. Let $X \subseteq V$. Then

$$\sum_{e \in \delta^-(X)} f(e) = 0.$$

DEFINICE 7.39. An $\mathcal{A}$-coloring of G is $\varphi : V \to \mathcal{A}$, such that

$$(\forall uv \in E) \qquad \varphi(u) \neq \varphi(v)\,.$$

VĚTA 7.40. G has an $\mathcal{A}$-coloring, iff G^* has a nowhere-zero $\mathcal{A}$-flow.

Důkaz. Nechť φ je $\mathcal{A}$-obarvení. Definujeme $f(uv) = \varphi(u) - \varphi(v)$. Když vezmeme $\delta^-(v)$, to určuje nějakou kružnici v původním grafu, tedy na ní je součet nulový.

Naopak, pokud máme f nikde-nulový, nechť $v \in V$ je libovolný, a $a \in \mathcal{A}$ taky. Definujeme pro $w \in V$ $\varphi(v) = a + \sum_{e \in P_{v,w}} f(e)$. Pak

$$\varphi(u) - \varphi(w) = a + \sum_{e \in P_{v,u}} f(e) - a - \sum_{e \in P_{v,w}} f(e) = \sum_{e \in P_{u,v,w}} f(e) = -f(uv)\,,$$

tedy je to ne-nula. ■

POZOROVÁNÍ 7.41. G has a nowhere-zero $\mathbb{Z}_2$-flow, iff it is Eulerian.

POZOROVÁNÍ 7.42. Let G be 3-regular. Then

(1) G has a nowhere-zero $\mathbb{Z}_3$-flow, iff it is bipartite,

(2) G has a nowhere-zero $\mathbb{Z}_2^2$-flow, iff it is 3-edge-colorable.

POZOROVÁNÍ 7.43. A planar graph G is a triangulation, iff G^* is 3-regular and 2-edge-connected.

DŮSLEDEK 7.44. A planar triangulation is 3-colorable, iff it is Eulerian.

DŮSLEDEK 7.45. The 4-color theorem is equivalent to the fact that every 3-regular 2-edge-connected planar graph is 3-edge-colorable.

FAKT 7.46. The actual $\mathcal{A}$ does not matter, only its size.

7.5 Fractional Coloring

DEFINICE 7.47 (FRACTIONAL COLORING). A *fractional coloring* of G is a function $\varphi : V \to \mathcal{P}(\mathbb{R})$, such that

(A1) for each $v \in V$, $\varphi(v)$ is measurable,

(A2) for each $v \in V$, $\mu(\varphi(v)) = 1$,

(A3) for each $uv \in E$, $\varphi(v) \cap \varphi(u) = \emptyset$.

DEFINICE 7.48.

$$\mu(\varphi) = \mu\left(\bigcup_{v \in V} \varphi(v)\right).$$

DEFINICE 7.49 (FRACTIONAL CHROMATIC NUMBER). The *fractional chromatic number* of G is

$$\chi_{\mathrm{f}}(G) = \inf\{\mu(\varphi)\,;\, \varphi \text{ is a fractional coloring}\}\,.$$

POZOROVÁNÍ 7.50.

$$\chi_{\mathrm{f}}(G) \leq \chi(G)\,.$$

DEFINICE 7.51. For $r \in \mathbb{R}$, define

$$A(r) := \{v \in V;\, r \in \varphi(v)\}\,.$$

Pozorování 7.52. $A(r)$ are independent sets.

Definice 7.53. Let I be an independent set, and $A^{-1}(I) = \{r \in \mathbb{R};\ A(r) = I\}$. Then define

$$x(I) := \mu\big(A^{-1}(I)\big)\,.$$

Tvrzení 7.54 (Primar).

$$\begin{aligned} x(I) &\geq 0, && (\forall I \in \mathcal{I}(G)) \\ \sum_{v \in I} x(I) &= 1, && (\forall v \in V) \\ \min \mu(\varphi) &= \sum_{I \in \mathcal{I}(G)} x(I)\,. \end{aligned}$$

Důsledek 7.55 (Dual).

$$\begin{aligned} &\max \sum_{v \in V} y_v, \\ \sum_{v \in I} y_v &\leq 1, && (\forall I \in \mathcal{I}(G)) \\ y_v &\geq 0. && (\forall v \in V) \end{aligned}$$

Pozorování 7.56.

$$\chi(G) = 2 \quad \to \quad \chi_{\mathrm{f}}(G) = 2.$$

Definice 7.57 (Kneser graph). Let $a, b \in \mathbb{N}$ and $a \geq b$. Define $K_{a:b} := \langle V, E\rangle$, such that

$$V = \binom{[a]}{b}, \qquad E = \{xy;\ x \cap y = \emptyset\}\,.$$

Pozorování 7.58.

$$\chi_{\mathrm{f}}(K_{a:b}) \leq \frac{a}{b}.$$

Definice 7.59.

$$\mathcal{S}_d = \left\{x \in \mathbb{R}^{d+1};\ \|x\|_2 = 1\right\}.$$

Fakt 7.60 (Borsuk–Ulam). Nechť $X_1, \ldots, X_{d+1}$ jsou podmnožiny $\mathcal{S}_d$, každá z nich buď otevřená nebo uzavřená, a $X_1 \cup \ldots, \cup X_{d+1} = \mathcal{S}_d$. Pak existuje i a $x \in \mathcal{S}_d$, tž. $x, -x \in A_i$.

Lemma 7.61.

$$\chi(K_{a:b}) \geq a - 2b + 2.$$

Důkaz. Nechť $\varphi : \binom{[a]}{b} \to [k]$ je obarvení pro $k = a - 2b + 1$. Nechť $x_1, \ldots, x_a$ jsou body na $\mathcal{S}_k$ v obecné poloze, tj. žádná $(k+1)$-tice neleží na nadrovině procházející středem.

Pro barvu $c \in [k]$ a $p \in \mathcal{S}_k$ definujeme množinu $S_{p,c} \in \binom{[a]}{b}$ tak, že

(1) $\varphi(S_{p,c}) = c$, a

(2) $\{x_i;\ i \in S_{p,c}\}$ leží na otevřené polokouli se středem p.

Definujeme $A_c = \{p \in \mathcal{S}_k;\ S_{p,c} \text{ existuje}\}$. Dodefinujme $A_{k+1} = \mathcal{S}_k \setminus \bigcup_c A_c$. Určitě A_c jsou otevřené pro $c \in [k]$ a A_{k+1} je uzavřená. Dohromady dávají $\mathcal{S}_k$. Z věty 7.60 existuje $c \in [k+1]$ a $p \in \mathcal{S}_k$, pro které $p, -p \in A_c$.

Pokud $c \in [k]$, existují stejně obarvené $S_{p,c}$ a $S_{-p,c}$, jejichž body ale leží na disjunktních polokoulích, tedy jsou disjunktní. To ale znamená, že jsou v grafu spojené hranou, což je spor.

Pokud je vybraná barva $k+1 = a - 2b + 2$, pak každá z polokoulí se středy $p, -p$ obsahuje nejvýše $b-1$ bodů. To znamená že zbytek jich je přesně v polovině mezi nimi. Zbylých je ale

$$a - 2\,(b-1) = k+1.$$

To je spor s obecnou polohou bodů. ■

Věta 7.62.

$$(\forall \varepsilon > 0)(\forall k)(\exists G) \qquad \chi_f(G) \leq 2 + \varepsilon, \qquad \chi(G) \geq k.$$

7.6 Věty dokazované metodou náboje

Důsledek 7.63 (Grötzsch). Every planar triangle-free graph is 3-colorable.

Věta 7.64 (4 barvy). Every planar graph is 4-colorable.

Věta 7.65 (4 barvy, ekviv.). Every planar 2-edge-connected 3-regular graph is 3-edge-colorable.

Poznámka 7.66. There is a correspondence between the counterexamples, as they are duals of each other, so we can flip as we please.

Lemma 7.67. Pro counterexample věty o 4 barvách platí

$$\delta(G^*) \geq 5.$$

Důsledek 7.68. G neobsahuje 4-cyklus.

Otázka 8

Grafové minory

Definice 8.1 (Minors). Let G, H be graphs.

(1) H is a *minor* of G, $H \preccurlyeq_{\mathrm{m}} G \equiv$ we can get H from G by deleting vertices and edges, and contracting edges.

(2) H is an *induced minor* of G, $H \preccurlyeq_{\mathrm{i}} G \equiv$ we can get H from G by deleting vertices, and contracting edges.

(3) H is a *topological minor* of G, $H \preccurlyeq_{\mathrm{t}} G \equiv$ we can get H from G by deleting vertices, edges, and contracting edges with at least one endpoint of degree 2.

Značení 8.2. Let G, H be graphs. We denote

(1) $H \sqsubseteq G \equiv H$ is an *induced subgraph* of G,

(2) $H \subseteq G \equiv H$ is a *subgraph* of G,

We further use $\preccurlyeq_*$ as a general inequality, can be any of the above.

Pozorování 8.3. $H \preccurlyeq_* G$, iff it can be obtained from some $H' \subseteq G$ (or induced) by contraction (or top. contraction).

Lemma 8.4. $H \preccurlyeq_{\mathrm{m}} G$, iff there is a $f : V_H \to \mathcal{P}(V_G)$, such that

(1) for all $v \in V_H$ the image $f(v) \neq \emptyset$ and $f(v)$ induces a connected graph,

(2) for all $u \neq v \in V_H$, the intersection $f(v) \cap f(u) = \emptyset$, and

(3) for all $uv \in E_H$, the union $f(v) \cup f(u)$ induces a connected graph.

Lemma 8.5. $H \preccurlyeq_{\mathrm{i}} G$, iff there is a $f : V_H \to \mathcal{P}(V_G)$, such that

(1) for all $v \in V_H$ the image $f(v) \neq \emptyset$ and $f(v)$ induces a connected graph,

(2) for all $u \neq v \in V_H$, the intersection $f(v) \cap f(u) = \emptyset$, and

(3) the pair $uv \in E_H$, *iff* the union $f(v) \cup f(u)$ induces a connected graph.

Lemma 8.6. For each H of $\Delta(H) \leq 3$, and for all G,

$$H \preccurlyeq_{\mathrm{m}} G \leftrightarrow H \preccurlyeq_{\mathrm{t}} G.$$

8.1 Drawings

DEFINICE 8.7. Let H and G be graphs with drawings φ_H, φ_G. We say that the drawings themselves are minors, $\varphi_H \preccurlyeq_\mathrm{m} \varphi_G$, if $H \preccurlyeq_\mathrm{m} G$ and if we contract φ_G to H, we get a drawing isomorphic to φ_H.

POZNÁMKA 8.8. In this section, we keep multiedges and loops.

POZNÁMKA 8.9. We assume that all graphs in this section have a fixed drawing.

DEFINICE 8.10 (DUAL). The dual of G is $G^* := \langle F, E^* \rangle$, where in E^*, for each original edge $e \in E$, we have an edge e^* connecting the two faces incident to e.

POZOROVÁNÍ 8.11. If G is connected, then $G^{**} \cong G$.

POZOROVÁNÍ 8.12. If G and H are connected, then

$$H^* \preccurlyeq_\mathrm{m} G^* \leftrightarrow H \preccurlyeq_\mathrm{m} G.$$

DEFINICE 8.13. The graph G is *outerplanar* $\equiv$ it has a drawing where all vertices are incident to the outer face.

POZOROVÁNÍ 8.14. G is outerplanar $\leftrightarrow K_4, K_{2,3} \not\preccurlyeq_\mathrm{m} G \leftrightarrow$ its dual is a tree + one vertex connecting to all others.

8.2 Characterization by forbidden substructures

VĚTA 8.15. Graph G is bipartite, iff it has no odd cycle.

VĚTA 8.16. Graph G is planar, iff it has no subdivision of $K_5, K_{3,3}$.

VĚTA 8.17. G is perfect, iff it doesn't contain odd holes or odd antiholes.

DEFINICE 8.18. A class of graphs $\mathcal{G}$ is $\preccurlyeq_*$*-closed* $\equiv$

$$(\forall G \in \mathcal{G})(\forall H) \quad H \preccurlyeq_* G \to H \in \mathcal{G}.$$

POZOROVÁNÍ 8.19. If $\mathcal{G}$ is minor-closed, it is induced-minor-closed.

DEFINICE 8.20 (FORBIDDEN CLASS). Let $\mathcal{F}$ be a class of graphs. We define the class of graphs not containing $\mathcal{F}$ as

$$\mathrm{Forb}_{\preccurlyeq_*} \mathcal{F} := \{G;\ (\forall F \in \mathcal{F})(F \not\preccurlyeq_* G)\}.$$

DEFINICE 8.21 (OBSTRUCTION). A graph F is a $\preccurlyeq_*$*-obstruction* for $\mathcal{G} \equiv$

$$F \notin \mathcal{G} \wedge (\forall F' \preccurlyeq_* F)(F \neq F' \to F' \in \mathcal{G}).$$

DEFINICE 8.22. A partial order $\leq$ is *locally finite* $\equiv$

$$(\forall G) \quad V(G) \prec \mathbb{N} \to \{H;\ H \leq G\} \prec \mathbb{N}.$$

LEMMA 8.23. Let $\mathcal{G}$ be a class of graphs. Let $\leq$ be locally finite. Then the following are equivalent:

(1) $\mathcal{G}$ is $\leq$-closed.

(2) $\mathcal{G} = \mathrm{Forb}_{\leq} \mathcal{F}$ for some $\mathcal{F}$.

(3) $\mathcal{G} = \mathrm{Forb}_{\leq}(\mathrm{Obst}_{\leq} \mathcal{G})$.

POZNÁMKA 8.24. For $\mathcal{F}$ from (2), the smallest possible choice is $\mathrm{Obst}_{\leq} \mathcal{G}$.

8.3 Clique-sums

DEFINICE 8.25 (CLIQUE-SUM). Let G_1, G_2 be two graphs sharing a clique K_k. Let f be a bijection between the vertices of the clique in G_1 and in G_2. A *k-clique-sum* of G_1, G_2 via f is a graph G, such that G is received by connecting G_1, G_2 in the clique vertices, as mapped via f, and then removing some edges from the clique.

POZNÁMKA 8.26. For $k = 0$, the clique-sum is a disjoint union.

LEMMA 8.27. If G is a clique-sum of G_1, G_2 and $K_k \preccurlyeq_\mathrm{m} G$, then $K_k \preccurlyeq_\mathrm{m} G_1$ or $K_k \preccurlyeq_\mathrm{m} G_2$.

VĚTA 8.28. Let H be 3-connected, and $H \not\preccurlyeq_\mathrm{m} G$. Then there are graphs $G_1, \ldots, G_t$, such that

(1) $(\forall i)(H \not\preccurlyeq_\mathrm{m} G_i)$,

(2) $(\forall i)$ G_i is 3-connected,

(3) G can be obtained from $G_1, \ldots, G_t$ via (≤ 2)-clique-sums.

Důkaz. Indukcí dle vrcholů. Pokud je G 3-souvislý, je tvrzení triviální. Nechť $S \subseteq V$ je nejmenší řez, $|S| \leq 2$.

Rozdělíme G na podgrafy G_1, G_2 dle řezu S, aby $S = V_1 \cap V_2$. Přidáním hran do S se neudělá H-minor, protože jinak by byl H-minor už v původním G. Indukcí získáme tyto G_1 a G_2 pomocí menších 3-souvislých grafů a pak použijeme clique-sum přes S. ■

POZOROVÁNÍ 8.29. $G \not\preccurlyeq_\mathrm{m} K_2$, iff $E(G) = \emptyset$, that is, G can be obtained from copies of K_1 via 0-clique-sums.

POZOROVÁNÍ 8.30. $G \not\preccurlyeq_\mathrm{m} K_3$, iff G is a forest, that is, G can be obtained from copies of K_1, K_2 via (≤ 1)-clique-sums.

LEMMA 8.31. If G is 3-connected, and $|V| \geq 4$, then $K_4 \preccurlyeq_\mathrm{m} G$.

Důkaz. Vezmeme nejkratší cyklus v G. Pak vezmeme v mimo, a z Mengerovy věty najdeme vrcholově disjunktní cesty z v do cyklu. To je podrozdělení K_4. ■

VĚTA 8.32. $K_4 \not\preccurlyeq_\mathrm{m} G$ iff G can be obtained from copies of K_1, K_2, K_3 via (≤ 2)-clique-sums.

Důkaz. G není 3-souvislý dle lemma 8.31. Podle věty 8.28 existují 3-souvislé grafy ze kterých lze udělat G. To musí být K_1, K_2, K_3. ■

FAKT 8.33 (WAGNER). $K_5 \not\preccurlyeq_\mathrm{m} G$ iff G can be obtained from planar graphs and the Wagner graph, which is C_8 with connected opposite vertices, via (≤ 3)-clique-sums.

DŮSLEDEK 8.34. Let $k \leq 5$. Then

$$K_k \not\preccurlyeq_\mathrm{m} G \leftrightarrow \chi(G) < k.$$

DOMNĚNKA 8.34.1 (HADWIGER).

$$(\forall k) \quad K_k \not\preccurlyeq_\mathrm{m} G \rightarrow \chi(G) < k.$$

Otázka 9

Stromová šířka

Definice 9.1 (Tree decomposition). A *tree decomposition* of a graph G is the tuple (T, β), where

(1) T is a tree, and

(2) $\beta : V(T) \to \mathcal{P}(V(G))$ is the *bag function*, assigning a bag to each vertex with the following properties:

$$(\forall v \in V(G))(\exists n) \quad v \in \beta(n)\,, \tag{9.1.1}$$
$$(\forall uv \in E(G))(\exists n) \quad u, v \in \beta(n)\,, \tag{9.1.2}$$
$$(\forall v \in V(G)) \quad T\left[\{n;\, v \in \beta(n)\}\right] \text{ is connected.} \tag{9.1.3}$$

Pozorování 9.2. If $S \subseteq V(G)$ induces a clique in G, then for each (T, β) of G,

$$(\exists n) \quad S \subseteq \beta(n)\,.$$

Pozorování 9.3. Let $H \subseteq G$ be connected and (T, β) be a tree decomposition of G. Then

$$\{x \in V(T)\,;\, V(H) \cap \beta(x) \neq \emptyset\}$$

induces a connected subtree of T.

Definice 9.4 (Width). The *width* of the bag decomposition (T, β) is

$$w(\beta) := \max_{n \in V(T)} |\beta(n)| - 1.$$

Pozorování 9.5. Let (T, β) be a tree decomposition of G. Then

$$\delta(G) \leq w(\beta)\,.$$

Definice 9.6 (Tree width). The *tree width* of a graph G is

$$\mathrm{tw}(G) := \min_{(T,\beta)} w(\beta)\,.$$

Pozorování 9.7.

$$\delta(G) \leq \mathrm{tw}(G)\,.$$

Pozorování 9.8.

$$H \preccurlyeq_{\mathrm{m}} G \to \mathrm{tw}(H) \leq \mathrm{tw}(G)\,.$$

Pozorování 9.9. $\mathrm{tw}(G) \leq t$, iff G can be obtained by clique-sums from graphs on at most $t + 1$ vertices.

9.1 Dynamické programování

Stromová šířka lze využít na to abychom vyřešili spoustu kombinatorických problémů jednoduše pomocí DP. Například barvení, vrcholové pokrytí, nezávislá množina. Hodí se mít tzv. pěknou dekompozici.

Definice 9.10 (Pěkná dekompozice). Dekompozice $\langle T, \beta \rangle$ je *pěkná* $\equiv$ vrcholy $t \in V_T$ jsou jednoho ze čtyř druhů:

(1) *list node* bez dětí,

(2) *forget node* s jedním dítětem t' a
$$\beta(t) = \beta(t') \setminus \{x\},$$

(3) *introduced node* s jedním dítětem t' a
$$\beta(t) = \beta(t') \cup \{x\},$$

(4) *join node* s dvěma dětmi t_1, t_2 a
$$\beta(t) = \beta(t_1) = \beta(t_2).$$

Věta 9.11. Každý graf má pěknou dekompozici optimální šířky s $\mathcal{O}(n)$ nodes, kterou lze získat v $2^{\mathcal{O}(k)} \cdot n$.

Algoritmus 9.12 (Barvení).
Vstup: G, pěkná dekompozice $\langle T, \beta \rangle$, počet barev c.
Výstup: Existuje obarvení grafu G pomocí c barev?

Vyřešíme pomocí DP procedur popsaných níže, dostaneme řešení B.
Pokud $B \neq \emptyset$, vrátíme Ano, jinak Ne.

Leaf(t):
    Vrátíme všechna možná obarvení $\beta(t)$ c barvami.
Forget(t, t'):
    Vrátíme řešení pro t' bez zapomenutého vrcholu x.
Introduce(t, t'):
    $B' \leftarrow$ Vyřešíme pro t'.
    $x \leftarrow$ Introduced vrchol.
    $B \leftarrow \emptyset$
    Pro $\varphi \in B'$:
        Pro $i \in [c]$:
            Pokud rozšíření φ o $\varphi(x) = i$ je validní obarvení, přidáme jej do B.
    Vrátíme B.
Join(t, t_1, t_2):
    $B_1 \leftarrow$ Řešení pro t_1.
    $B_2 \leftarrow$ Řešení pro t_2.
    Vrátíme $B_1 \cap B_2$.

Algoritmus 9.13 (Nezávislá množina).
Vstup: G s váhami $w : V \to \mathbb{R}$, pěkná dekompozice $\langle T, \beta \rangle$.
Výstup: Maximální váha nezávislé množiny v G.

1: Leaf(t):
2: Vrátíme všechny možné nezávislé množiny spolu s jejich váhou.
3: Forget(t, t'):
4: Vrátíme řešení pro t' s odstraněným x, při konfliktu preferujeme to s větší váhou.
5: Introduce(t, t'):
6: $B' \leftarrow$ Vyřešíme pro t'.
7: $x \leftarrow$ Introduced vrchol.
8: $B \leftarrow B'$
9: Pro $(I, k) \in B'$:
10: Pokud $I \cup \{x\}$ je nezávislá, přidáme $I \cup \{x\}$ do B s váhou $k + w(x)$.
11: Vrátíme B.
12: Join(t, t_1, t_2):
13: $B_1 \leftarrow$ Řešení pro t_1.
14: $B_2 \leftarrow$ Řešení pro t_2.
15: Vrátíme kombinace nez. množin s váhami $k_1 + k_2 - \sum_{v \in I} \beta(v)$.

9.2 Courcelle's Theorem

Definice 9.14 (MSOL). A formula is in *monadic second order logic*, if it contains

(1) variables $v_1, v_2, \ldots$ for vertices,
(2) variables $e_1, e_2, \ldots$ for edges,
(3) variables $V_1, V_2, \ldots$ for sets of vertices,
(4) variables $E_1, E_2, \ldots$ for sets of edges,
(5) equality $=$ for vertices and edges,
(6) incidence $i(v_i, e_j) \equiv$ vertex v_i is incident edge e_j,
(7) set membership for v_i, V_j and e_i, E_j,
(8) logical operators $\vee, \neg$,
(9) the existence quantifier on any variable,
(10) anything else expressible via these.

Definice 9.15 (Variable assignment). Let G be a graph. A *variable assignment* is σ, such that

$$\begin{aligned} &(\forall i) \quad \sigma(v_i) \in V(G)\,, \\ &(\forall i) \quad \sigma(e_i) \in E(G)\,, \\ &(\forall i) \quad \sigma(V_i) \subseteq V(G)\,, \\ &(\forall i) \quad \sigma(E_i) \subseteq E(G)\,. \end{aligned}$$

Definice 9.16 (Satisfying assignment). Let G be a graph. An assignment σ of G *satisfies* φ in MSOL, $(G, \sigma) \vDash \varphi \equiv$

(1) $\varphi \equiv \alpha = \beta$ and $\sigma(\alpha) = \sigma(\beta)$,
(2) $\varphi \equiv i(v_i, e_j)$ and $\sigma(v_i)$ is incident $\sigma(e_j)$,
(3) $\varphi \equiv a \in A$ and $\sigma(a) \in \sigma(A)$,
(4) $\varphi \equiv \varphi_1 \vee \varphi_2$ and $(G, \sigma) \vDash \varphi_1 \vee (G, \sigma) \vDash \varphi_2$,
(5) $\varphi \equiv \neg\varphi_1$ and $(G, \sigma) \nvDash \varphi_1$, or

(6) $\varphi \equiv (\exists\alpha)\,\varphi_1$ and there exists an extension of σ to σ' by assigning the value $\sigma'(\alpha)$, such that $(G, \sigma') \vDash \varphi_1$.

Definice 9.17 (Satisfying a formula). Let φ be a MSOL formula without free variables. Let G be a graph. Then G *satisfies* φ, $G \vDash \varphi$, if $(G, \emptyset) \vDash \varphi$.

Fakt 9.18 (Courcelle). Let G be a graph of $\mathrm{tw}(G) \leq k$, φ be a MSOL formula without free variables. Then there exists an algorithm for deciding $G \vDash \varphi$, running in $\mathcal{O}_{k,|\varphi|}(n+m)$.

9.3 Series-Parallel Graphs

Definice 9.19. A graph G is *series-parallel* $\equiv$ there are vertices u, v labeled *north* and *south*, such that

(1) K_2 is series-parallel,

(2) for G_1, G_2 series-parallel, connecting them in series—identifying south of one and north of the other, removing the label—yields a series-parallel graph,

(3) for G_1, G_2 series-parallel, connecting them in parallel—identifying their souths and norths—yields a series-parallel graph.

Věta 9.20. A graph G has treewidth at most 2, iff it is a subgraph of a series-parallel graph.

Věta 9.21. A graph G is a subgraph of a series-parallel graph, iff it has no K_4 minor.

9.4 Chordal Graphs

Definice 9.22. Graph is chordal, if it has no induced cycle of length ≥ 4.

Definice 9.23 (PES). A *perfect elimination scheme* is $v_1, \ldots v_n$, such that v_i's neighborhood is a clique in $G[v_1, \ldots, v_i]$.

Lemma 9.24. G is chordal, iff it has a PES.

Tvrzení 9.25. G is chordal, iff it has a tree decomposition, where each node induces a clique.

Důsledek 9.26. Treewidth of a chordal graph is its clique number.

Definice 9.27 (k-tree). A k-*tree* is

(1) K_{k+1}, or

(2) if G is a k-tree and $V' \subseteq V$ of size k induces a clique, then adding a vertex adjacent to V' is a k-tree.

Tvrzení 9.28. G has treewidth at most k, iff it is a subgraph of a k-tree.

Důsledek 9.29.

$$\mathrm{tw}(G) + 1 = \min\{\omega(G')\,;\, G' \supseteq G, G' \text{ chordal}\}.$$

9.5 Brambles

DEFINICE 9.30. *Bramble* je $\mathcal{B} \subseteq \mathcal{P}(V)$, tž.

(1) $(\forall B \in \mathcal{B})(G[B]$ je souvislý),

(2) $(\forall B, B' \in \mathcal{B})(G[B \cap B']$ je souvislý).

DEFINICE 9.31. Množina $W \subseteq V$ *drží* $\mathcal{B} \equiv$

$$(\forall B \in \mathcal{B})(B \cap W \neq \emptyset)\,.$$

DEFINICE 9.32. *Řád* bramblu $\mathcal{B}$ je

$$r(\mathcal{B}) = \min\{|W|\,;\, W \text{ drží } \mathcal{B}\}.$$

LEMMA 9.33. Nechť $\mathcal{B}$ je bramble a (T, β) je stromová dekompozice. Pak existuje $t \in V_T$, pro které $\beta(t)$ drží $\mathcal{B}$.

Důkaz. Standardní důkaz označením hran stromu. Nechť lemma neplatí. Pro každý $t \in V_T$ vezmeme $B \in \mathcal{B}$ který není pokrytý $\beta(t)$. Toto B je souvislé, tedy tvoří souvislý podstrom. Nechť u je soused t směrem k tomu podstromu. Označme hranu tu tímto směrem.

Protože existuje $|V_T| - 1$ hran, existuje alespoň jedna označená dvakrát. Odpovídající B, B' tedy neindukují souvislý podgraf, spor. ■

LEMMA 9.34. Let G have no $(k+1)$-order bramble. Then for each $\mathcal{B}$, there is a decomposition (T, β), such that for each $t \in V_T$ where $|\beta(t)| \geq k+1$,

(1) t is a leaf of T, and

(2) $\beta(t)$ doesn't hold $\mathcal{B}$.

Důkaz. Indukce dle počtu množin držících $\mathcal{B}$ velikosti nejvýše k. Předpokládejme, že pro $\mathcal{B}$ je lemma pravdivé pro všechny ostatní brambly, které jsou drženy méně množinami velikosti nejvýše k. BÚNO je $|V| \geq k+1$.

Nechť W je minimální držící $\mathcal{B}$, tj. velikosti nejvýše k. Označme $G_1, \ldots, G_\ell$ komponenty grafu $G - W$ s W přidanou zpět (i s incidentními hranami). Označme taky $A_i = V(G_i) - W$. Dokážeme teď následující:

Pro každé i existuje (T_i, β_i) dekompozice G_i, pro kterou

(1) W je $\beta(t)$ pro nějaké t,

(2) každý t s $|\beta(t)| \geq k+1$ je list a nedrží $\mathcal{B}$.

Definujme $\mathcal{B}_i = \mathcal{B} \cup \{A_i\}$. Rozlišíme dva případy:

(1) Pokud $\mathcal{B}_i$ není bramble, můžeme definovat T na dvou vrcholech, jeden s bagem W a druhý s $A_i \cup N(A_i)$. Protože $\mathcal{B}_i$ není bramble, druhá z nich nedrží $\mathcal{B}$ a máme dokázán celý indukční krok.

(2) Jinak, $\mathcal{B}$ je pokrýván množinou W, která ale nepokrývá $\mathcal{B}_i$. Můžeme tedy použít indukční předpoklad a získáme (T_i, β). Pokud to je validní řešení i pro $\mathcal{B}$, jsme hotovi s důkazem celého lemmatu. Jinak existuje $t \in V_{T_i}$ velikosti $k+1$, disjunktní s A_i a pokrývající $\mathcal{B}$.

   Všimněme si, že každý S separátor W a $\beta(t)$ musí držet $\mathcal{B}$. Z toho vyplývá, že $|S| \geq k+1$, a tedy, dle Mengera, existují $P_1, \ldots, P_{|W|}$ vrcholově-disjunktní cesty z W do $\beta(t)$.

   Nyní upravíme T_i na T_i', což bude stromová dekompozice dokazující pomocné tvrzení. Úpravy budou následující:

   (1) Pro všechna $u \in T_i$ omezíme $\beta(u)$ na $A_i \cup W$.

   (2) Pro všechna $w \in W$ vložíme w do $\beta(u)$ na cestě v T_i mezi t a nějakým nodem obsahujícím w.

Vrchol w jsme, kromě t, přidali jen do vnitřních vrcholů, takže listy (mimo W) jsou stále podmnožiny původních. Navíc, každý node do kterého jsme přidali w obsahoval nějaký $v \in P_i$, který jsme odstranili, takže velikosti vnitřních nodů nevzrostly.

Pak každý t' s $|\beta(t')| > k$ má $\beta(t') \subset W \cup A_i$. Protože $|W| = k$, obsahuje $\beta(t')$ alespoň jeden prvek A_i. Z IP ale t' nedrží $\mathcal{B}_i$, tedy nedrží ani $\mathcal{B}$.

Tím jsme pomocné tvrzení dokázali. Teď už stačí zkombinovat T_i do T přes node W a máme dokázané lemma. ■

Věta 9.35 (Bramble Theorem). A graph G has a bramble of order $k+1$, iff its treewidth is at most k. In other words,

$$\max r(\mathcal{B}) = \mathrm{tw}(G)\,.$$

Důkaz. Aplikujeme lemma 9.34 na $\mathcal{B} = \{V\}$. Každá množina drží $\mathcal{B}$, a proto odpovídající stromová dekompozice má šířku k. ■

Fakt 9.36. A graph G is planar, iff the class of graphs not containing G as a minor has bounded treewidth.

Otázka 10

Geometrické reprezentace grafů

Definice 10.1 (Intersection graph). Let $\mathcal{A}$ be a finite family of sets. An *intersection graph* of $\mathcal{A}$ is

$$\mathrm{IG}(\mathcal{A}) := \langle \mathcal{A}, \{ab;\, a, b \in \mathcal{A}, a \neq b, a \cap b \neq \emptyset\}\rangle\,.$$

Definice 10.2 (Family of IG). Let $\mathcal{M}$ be a family of (mostly geometric) objects. Then

$$\mathcal{IG}(\mathcal{M}) := \{\mathrm{IG}(\mathcal{A})\,;\, \mathcal{A} \subseteq \mathcal{M}, |\mathcal{A}| \in \mathbb{N}\}\,.$$

Definice 10.3 (Interval graphs). The *interval graphs* is a family defined as

$$\mathsf{INT} := \mathcal{IG}(\{(a, b)\,;\, a, b \in \mathbb{R}\})\,.$$

Tvrzení 10.4. INT are perfect.

10.1 Chordal graphs

Definice 10.5 (Chordal graph). A graph is *chordal* $\equiv$

$$(\forall k \geq 4) \qquad C_k \not\leq G.$$

Lemma 10.6. Each inclusion-wise minimal vertex cut in a chordal graph induces a clique.

Definice 10.7 (Simplicial vertex). A vertex is *simplicial* $\equiv$ its neighborhood induces a clique.

Lemma 10.8. A chordal graph is either a clique, or has two non-adjacent simplicial vertices.

Definice 10.9 (PES). A perfect elimination scheme is an ordering of V to $v_1, \ldots, v_n$, such that

$$(\forall i) \qquad v_i \text{ is simplicial in } G[\{v_1, \ldots, v_i\}].$$

Definice 10.10. A *clique-tree* decomposition of G is $T = \langle \mathcal{Q}, E\rangle$, such that

(1) $\mathcal{Q} = \{Q;\, Q \text{ is an inclusion-wise maximal clique in } G\}$,

(2) T is a tree,

(3) $(\forall v \in V)(T[\{Q \in \mathcal{Q};\, v \in Q\}]$ is connected).

Věta 10.11. The following are equivalent:

(1) G is chordal,

(2) G has a PES,

(3) G has a clique-tree decomposition,

(4) $G \cong$ IG(subtrees in a finite tree).

Pozorování 10.12 (Helly property). For a collection of intervals $\langle I_i \rangle_i$, it holds

$$(\forall i \neq j)(I_i \cap I_j \neq \emptyset) \to \bigcap_i I_i \neq \emptyset.$$

Důsledek 10.13. For a max clique Q of an interval graph G, it holds

$$\bigcap_{v \in Q} I_v \neq \emptyset.$$

Definice 10.14 (Clique-path decomposition). A *clique-path* decomposition of G is $P = \langle \mathcal{Q}, E \rangle$, such that

(1) $\mathcal{Q} = \{Q;\ Q \text{ is an inclusion-wise maximal clique in } G\}$,

(2) P is a path,

(3) $(\forall v \in V)(P[\{Q \in \mathcal{Q};\ v \in Q\}]$ is connected).

Věta 10.15. The following are equivalent:

(1) $G \in \mathsf{INT}$,

(2) G has a clique-path decomposition,

(3) $G \cong$ IG(subpaths of a finite path).

Definice 10.16. A *comparability graph* (CO) on $P = \langle X, < \rangle$ is G_P, such that

$$V(G_P) = X, \quad E(G_P) = \{xy;\ x, y \in X, x < y \vee y < x\}.$$

Pozorování 10.17. $G \in \mathsf{CO} \leftrightarrow$ there is a transitive orientation G', that is

$$(\forall uv, vw \in E') \qquad uw \in E'.$$

Definice 10.18. Let $A = [0,1] \times \{0\}$ and $B = [0,1] \times \{1\}$. We define

(1) permutation graphs $\mathsf{PER} = \mathcal{IG}$(straight lines from A to B),

(2) function graphs $\mathsf{FUN} = \mathcal{IG}$(curves from A to B).

Definice 10.19 (co-class). Let $\mathcal{M}$ be a family of graphs. Then

$$\text{co-}\mathcal{M} := \left\{\overline{G};\ G \in \mathcal{M}\right\}.$$

Věta 10.20.

(1) $\mathsf{FUN} = \text{co-}\mathsf{CO}$,

(2) $\mathsf{PER} = \text{co-}\mathsf{CO} \cap \mathsf{CO}$,

(3) $\mathsf{INT} = \mathsf{CHOR} \cap \text{co-}\mathsf{CO}$.

Důkaz.

(1) Nechť $G \in \mathsf{FUN}$, kde všechny funkce jsou mezi dvěma vodorovnými přímkami. Pak pokud se dvě funkce neprotínají, je jedna nad druhou. To udává pořadí.

Nechť $G \in$ co-CO určené pomocí $\leq$. Pak $\leq = \leq_1 \cap \leq_2 \ldots \cap \leq_r$, kde všechny $\leq_i$ jsou lineární. Stačí je dát vedle sebe, propojit, a dostaneme $G \in \mathsf{FUN}$.

(2) Nejprve ukážeme PER = co-PER. Stačí prohodit permutaci. Teď, určitě $\mathsf{PER} \subseteq \mathsf{FUN} =$ co-CO. Zároveň $\mathsf{PER} =$ co-$\mathsf{PER} \subseteq$ co- (co-CO) = CO.

Pokud $\leq_1$ je uspořádání G a $\leq_2$ je usp. jeho doplňku, je $\leq = \leq_1 \cup \leq_2^{-1}$ uspořádání celého K_n. Toto určuje permutaci, která dává přesně G.

(3) INT $\subseteq$ CHOR je jasné. INT $\subseteq$ co-CO je taky jasné: stačí zadefinovat $\leq$ pokud je jeden interval zcela nad druhým. Nechť nyní G je chordální s $\leq$ transitivním nad jeho ne-hranami. Definujeme nad klikami $\mathcal{Q}$ uspořádání

$$Q_1 < Q_2 \leftrightarrow (\exists u \in Q_1)(\exists v \in Q_2)(u \leq v)\,.$$

Toto je lineární uspořádání. To určuje clique-path decompozici, a tedy $G \in$ INT.

■

10.2 Filament Graphs

Definice 10.21 (Interval filament graphs). $G \in \mathsf{IFA} \equiv$ there is a representation $R = \{\langle f_u, I_u\rangle\,;\, u \in V\}$, where

(1) I_u is an interval on the line $\mathbb{R}$,

(2) $f_u : [0,1] \to I_u \times \mathbb{R}_0^+$ is a curve, $f(0) = \inf I_u$ and $f(1) = \sup I_u$,

(3) $uv \in G \leftrightarrow f_u \cap f_v \neq \emptyset$.

Definice 10.22. We further define

(1) CA $= \mathcal{IG}$(arcs on a circle),

(2) CIRC $= \mathcal{IG}$(chords of a circle),

(3) PC $= \mathcal{IG}$(polygons with all vertices on a circle).

Tvrzení 10.23.

$$\mathsf{CHOR} \subseteq \mathsf{PC}.$$

Definice 10.24 (Alternating representation). An *alternating representation* of G is $w \in V(G)^* \equiv$

$$(\forall u, v \in V) \qquad uv \in E \leftrightarrow u, v \textit{ alternate}, \text{ that is, } uvuv \text{ is a subword of } w.$$

Tvrzení 10.25. G has an alternating representation $\leftrightarrow G \in$ PC.

Důkaz. Alternující reprezentace je určena polohou vrcholů polygonu na kružnici. ■

Tvrzení 10.26.

$$\mathsf{IFA} \supseteq \mathsf{INT}, \mathsf{CHOR}, \mathsf{FUN}, \mathsf{CIRC}, \mathsf{CA}, \mathsf{PC}.$$

Důkaz. INT a CHOR jsou FUN, CIRC a CA jsou PC. PC se dají převést na IFA pomocí alternující reprezentace, a FUN se dají převést pomocí toho, že konce funkcí lineárně rozšíříme tak, aby se dostaly zpět na $y = 0$ bez dalšího křížení. ■

Definice 10.27 (Mixing property). A graph has the *mixing property* via $E_1 \dot\cup E_2 = E \equiv$ there is a transitive orientation of E_2, such that

$$(\forall u, v, w \in V) \qquad uv \in E_2 \wedge vw \in E_1 \to uw \in E_1.$$

Definice 10.28 (Mixed class). Let $\mathcal{A}$ be a class of graphs. Then mixed($\mathcal{A}$) is a class of graphs, such that $G \in \text{mixed}(\mathcal{A}) \equiv$ there is a partition of $E = E_1 \dot\cup E_2$ which satisfies

(1) $\langle V, E_1\rangle \in \mathcal{A}$, and

(2) G has the mixing property via $E_1 \dot\cup E_2$.

Věta 10.29.

$$\text{co-IFA} = \text{mixed}(\text{co-INT})\,.$$

Důkaz „$\subseteq$“. Nechť $G \in$ IFA. Tento graf má ne-hrany ze dvou důvodů: buď se intervaly nekříží, nebo se nekříží funkce. První typ tvoří graf z co-INT. Druhý typ jsou intervaly vnořené do sebe, tedy můžeme definovat $u \le v \equiv I_u \subseteq I_v$. Toto tvoří orientaci E_2, a konečně, graf má mixing property. ■

Důkaz „$\supseteq$“. Nechť $G \in$ co-mixed(co-INT). Jeho ne-hrany jsou tedy $E_1 \dot\cup E_2$ kde E_2 má tranzitivní orientaci takovou, že $E_1 \dot\cup E_2$ má mixing property, a E_1 je co-INT. Nechť I_u pro $u \in V$ jsou odpovídající intervaly.

Pokud $I_u \cap I_v = \emptyset$, pak $uv \in E_1$. Jinak buď můžou být vnořené, nebo se nějak částečně překrývat. Platí následující: *Existuje intervalová reprezentace taková, že pro $uv \in \overline{E}_2$ platí, že $I_u \subseteq I_v$.*

Toto se dokazuje sporem: vezme se reprezentace s min. počtem intervalů, pro které toto neplatí, a dojde se ke sporu.

Z tohoto vyplývá, že všechny intervaly, které se protínají, určují hrany. Nakreslíme nad intervaly půlkruhy. To udělá všechny hrany pro protínající-se intervaly. Ještě ale můžou být hrany z intervalů které jsou vnořené. Vybereme pro $I_u \subseteq I_v$ nějaký vnitřní bod $\ell \in I_u$, a „provlékneme“ půlkruh přímkou $x = \ell$ tak, aby protínal půlkruh v, a všechny ostatní půlkruhy které nemáme protínat nadzvedneme. Z mixing property platí že tím nevytvoříme žádná protínání která nemáme. ■

10.3 Maximum Independent Set in IFA

Definice 10.30 (Max-Weight-Independent-Set). Let G be a graph and $w : V \to \mathbb{N}$. The goal in Max-Weight-Independent-Set is to find an independent set of maximum weight.

Definice 10.31 (Max-Weight-Clique). Let G be a graph and $w : V \to \mathbb{N}$. The goal in Max-Weight-Clique is to find a clique of maximum weight.

Tvrzení 10.32. Max-Weight-Clique is polynomial on INT.

Důkaz. Je lineárně mnoho klik, zkusíme všechny. ■

Tvrzení 10.33. Max-Weight-Independent-Set is polynomial on INT.

Důkaz. Dynamickým programováním: procházíme intervaly zleva doprava, držíme minima přes konce. ■

Tvrzení 10.34. Max-Weight-Clique is polynomial on CO a co-INT.

Důkaz. CO dynamickým programováním: udržujeme si pro každý vrchol maximální kliku končící v něm. Pro co-INT převodem na Max-Weight-Independent-Set v doplňku. ■

Tady ještě platí obecná věta o mixed-třídě, ale to je hrozně technický důkaz který se opravdu nechci učit.

10.4 Recognizing CHOR

ALGORITMUS 10.35 (LEXBFS).
Vstup: $G = \langle V, E \rangle$.
Výstup: Ordering σ on V.

$T \leftarrow \emptyset$
$(\forall v)(w(v) \leftarrow \{\infty\})$
For $i \in \{1, \ldots, n\}$:
    $u \in \arg\min_{x \in V \setminus T} w(x)$, ordered lexicographically (sets sorted from smallest to largest).
    $\sigma(i) \leftarrow u$
    $T \leftarrow T \cup \{u\}$
    For $x \in N(u), x \notin T$:
        $w(x) \leftarrow w(x) \cup \{i\}$

VĚTA 10.36. If $G \in$ CHOR, then LEXBFS outputs a PES.

Důkaz. Nechť $G \in$ CHOR. Nechť σ seřadí vrcholy $v_1, \ldots, v_n$, a toto není PES. Máme tedy $i < j < k$, tž. $v_i v_j \notin E$, ale $v_i v_k, v_j v_k \in E$. Protože v_j jsme vybrali před v_k a $i \in w(v_k)$, existuje nějaké $i > i' \in w(v_j)$.

Takto pokračujeme, a pokud předpokládáme minimalitu rozdílu $k - i$, nemůžeme přestat, protože bychom tím uzavřeli velkou kružnici. Dostali jsme spor s konečností grafu. ■

POZNÁMKA 10.37. σ can start with an arbitrary vertex.

ALGORITMUS 10.38 (TESTPES).
Vstup: $G = \langle V, E \rangle$, ordering σ.
Výstup: Is σ a PES?

$(\forall v \in V)(A(v) \leftarrow \emptyset)$
For $i \in \{n, \ldots, 2\}$:
    $v \leftarrow \sigma(i)$
    $X \leftarrow N(v) \cap \{\sigma(j)\,;\, j < i\}$
    If $X \neq \emptyset$:
        $u \in \arg\max_{u \in X} \sigma^{-1}(u)$
        $A(u) \leftarrow A(u) \cup (X \setminus \{u\})$
    If $A(v) \setminus N(v) \neq \emptyset$:
        Output NO.
Output YES.

VĚTA 10.39. TESTPES is correct and runs in $\mathcal{O}(m)$.

Důkaz. Pokud to je PES, je to určitě ok.

Pokud to není PES, nechť uv je nějaká hrana která chybí, tj. $u, v \in N(w)$, ale $uv \notin E$ a w je po u, v. Toto očividně zjistíme, každý vrchol v sousedství v nějakém kroku musí zkontrolovat že má všechny potřebné sousedy. ■

DŮSLEDEK 10.40. CHOR is recognizable on $\mathcal{O}(m)$.

Otázka 11

Algebraické vlastnosti grafů

Dá se mluvit o Alon-Tarsiho metodě, viz sekce 7.2, věta 7.24. Kdyžtak i o Tutteho polynomu.

Definice 11.1 (Multigraf). *Multigraf* je $G = (V, E)$, kde

(1) V je neprázdná množina *vrcholů*,

(2) E je *multimnožina*, $E \subseteq \binom{V}{2} \cup V$.

Značení 11.2. $k(G)$ je počet komponent grafu G.

Definice 11.3 (Hodnost). *Hodnost* multimnožiny E je

$$r(E) := |V| - k(G)\,.$$

Poznámka 11.4. Hodnost je velikost největší $F \subseteq E$ bez cyklů, tž. $k(G) = k(F)$.

Definice 11.5 (Nulita). *Nulita* je

$$n(E) := |E| - r(E)\,.$$

Poznámka 11.6. Nulita je velikost největší $F \subseteq E$, tž. $k(G) = k(G - F)$.

Definice 11.7 (Tutteův polynom). *Tutteův polynom* multigrafu G je

$$T_G(x, y) := \sum_{F \subseteq E} (x-1)^{r(E)-r(F)} (y-1)^{n(F)}\,, \qquad 0^0 \equiv 1.$$

Tvrzení 11.8. Nechť G je souvislý. Pak $T_G(1, 1)$ je roven počtu koster G.

Tvrzení 11.9. Nechť $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$ jsou multigrafy, a $|V_1 \cap V_2| \leq 1$ a $|E_1 \cap E_2| = 0$. Pak $G = (V_1 \cup V_2, E_1 \cup E_2)$ je multigraf, a

$$T_G(x, y) = T_{G_1}(x, y) \cdot T_{G_2}(x, y)\,.$$

Věta 11.10. Nechť G je multigraf. Pak Tutteův polynom G je jednoznačně určen jako

$$T_G(x, y) = \begin{cases} 1 & \text{pro } E = \emptyset, \\ xT_{G-e}(x, y) & \text{pro } e \in E \text{ most}, \\ yT_{G-e}(x, y) & \text{pro } e \in E \text{ smyčku}, \\ T_{G-e}(x, y) + T_{G/e}(x, y) & \text{jinak.} \end{cases}$$

DEFINICE 11.11 (CHROMATICKÝ POLYNOM). Nechť G je multigraf. *Chromatický polynom* G je $\mathrm{ch}_G : \mathbb{N} \to \mathbb{N}$, kde $\mathrm{ch}_G(b)$ je počet dobrých obarvení G b barvami.

VĚTA 11.12. Pro každý multigraf G platí

$$\mathrm{ch}_G(b) = (-1)^{|V|+k(G)}\, b^{k(G)} T_G(1-b, 0)\,.$$

Otázka 12

Teorie párování

Viz otázka 26.

Otázka 13

Ramseyova teorie

Hlavně budu mluvit o Regularity Lemma a jeho aplikacích, viz otázka 14. Budu ale umět i toto.

Definice 13.1 (Obarvení). *n-obarvení* je libovolné zobrazení $c : E \to [n]$.

Tvrzení 13.2 (Dirichletův princip). Nechť $n_1, \ldots, n_r \in \mathbb{N}, r \in \mathbb{N}$. Obarvíme-li prvky X r barvami a je-li $|X| \geq D(n_1, \ldots, n_r, r) = 1 + \sum_{i=1}^{r} (n_i - 1)$, pak existuje n_i takové, že X obsahuje n_i prvků i-té barvy.

Definice 13.3 (Ramsey). Nechť $k, \ell \in \mathbb{N}$. Definujeme $R(k, \ell)$ jako nejmenší $n \in \mathbb{N}$, že pro každé 2-obarvení K_n existuje K_k jedné barvy, nebo K_ℓ druhé barvy, jako podgraf K_n.

Věta 13.4 (Ramsey).

$$(\forall k, \ell \in \mathbb{N}) \qquad R(k, \ell) \leq \binom{k + \ell - 2}{\ell - 1} = \binom{k + \ell - 2}{k - 1}.$$

Důkaz. Indukcí dle k, ℓ. Pro k nebo ℓ rovno 1 je to jednoduché. Jinak, vezmeme $n = R(k-1, \ell) + R(k, \ell - 1) = n_1 + n_2$. To je

$$\leq \binom{k + \ell - 3}{k - 2} + \binom{k + \ell - 3}{\ell - 2} \leq \binom{k + \ell - 2}{\ell - 1}.$$

Vezmeme libovolný vrchol. Z Dirichletova principu existuje alespoň n_1 sousedů spojených jednou barvou nebo n_2 sousedů spojených druhou. Z IP v této podmnožině najdeme správně barevnou množinu. ■

Definice 13.5 (Ramsey pro obecné p). *Ramseyovo číslo* $R_p(n_1, \ldots, n_r)$ je nejmenší N, tž.

$$\forall X : |X| \geq N, \forall \chi : \binom{X}{p} \to [r]\ \exists i \in [r], \exists Y \subseteq X : |Y| = n_i \wedge \forall y \in \binom{Y}{p} : \chi(y) = i.$$

Věta 13.6 (Ramsey pro obecné p). Nechť $p, r, n_1, \ldots, n_r \in \mathbb{N}$. Pak $R_p(n_1, \ldots, n_r) \in \mathbb{N}$.

Důkaz. Indukcí dle $p, r, n_1, \ldots, n_r$. Pro $p = 1$ je to Dirichletův princip, pro nějaké $n_i = 1$ je řešení prostě jeden vrchol.

Vezmeme libovolné $v \in X$. Obarvíme každé $e \in \binom{X - v}{p-1}$ barvou $\chi(e + v)$. Z IP pro dost velké X máme $j \in [r]$ a dost velké $X' \subseteq X$ obarvené barvou j. Tedy n_j můžeme snížit o 1 a tentokrát na všech p-ticích a X' zavoláme znovu IP. Hotovo. ■

Věta 13.7 (Nekonečný Ramsey). $\forall p, r \in \mathbb{N}, \forall \chi : \binom{\mathbb{N}}{p} \to [r]\ \exists i \in [r], \exists A \subseteq \mathbb{N}$ nekonečná, tž.

$$\forall y \in \binom{A}{p} : \chi(y) = i.$$

Důkaz. Indukcí. Pro $p = 1$ je to triviální.

Nechť $\chi : \binom{\mathbb{N}}{p} \to [r]$ je obarvení. Najdeme posloupnost $A_1 \supseteq A_2 \supseteq \dots$ nekonečných množin následovně. Nechť $A_1 = \mathbb{N}$. Nyní, zvolme $v_i = \min A_i$. Definujme $\chi_i(e) = \chi(e + v_i)$ pro $e \in \binom{A_i}{p-1}$. Nyní použijeme indukční předpoklad na $A_i - v_i$, a ten nám dá nekonečnou A_{i+1} obarvenou stejně dle χ_i.

Takto máme posloupnost $v_1, v_2, \dots$, a barvy $b_1, \dots$, kde b_i je barva vyskytující se nekonečně-krát v A_i. Nějaká $b \in \{b_i\}_i$ se vyskytuje nekonečně-krát. Zvolíme $A = \{v_i;\, b_i = b\}$. ■

13.1 Halesova–Jewettova věta

Stručně popíšu Halesovu–Jewettovu větu, byť jsem ji na Kombinatorice a grafech 3 neměl. Docela dobře ale sedí na Ramseyovu teorii, tak je asi fajn umět alespoň znění.

Značení 13.8. Budeme používat $a \in \mathbb{N}$ pro *velikost* a $d \in \mathbb{N}$ pro *dimenzi* krychle $[a]^d$.

Definice 13.9. *Šablona* je funkce $\lambda \in ([a] \cup \{\star\})^d$, kde $\star$ je speciální prvek, tž. existuje i, že $\lambda_i = \star$. Na šablonu lze nahlížet také jako na *funkci* $\lambda : [a] \to [a]^d$, která pro $n \in a$ nahradí všechny výskyty $\star$ za n.

Definice 13.10 (Kombinatorická přímka). *Kombinatorická přímka* je

$$\{\lambda(n)\,;\, n \in [a]\}\,,$$

kde λ je libovolná šablona.

Věta 13.11 (Hales–Jewett). Nechť $a, r \in \mathbb{N}$. Pak existuje $d \in \mathbb{N}$ takové, že každé $\chi : [a]^d \to [r]$ obsahuje jednobarevnou přímku.

Věta se dokazuje indukcí přes a. Pro $a = 1$ je to triviální, a v indukčním kroku se ukáže že když vezmu dostatečně šíleně velikou dimenzi, skutečně budu moct nějakou jednobarevnou přímku z $[a-1]^d$ rozšířit na $[a]^d$.

Otázka 14

Szemerédiho lemma o regularitě

Značení 14.1. Let G be a graph, $A \dot\cup B \subseteq V(G)$. In this chapter, we denote

(1) $e(A,B) := |E(A,B)|$,

(2) $d(A,B) := \frac{e(A,B)}{|A||B|}$ the *density* of edges between A and B,

(3) $\mathcal{G}(n,p)$ the *distribution of graphs* on n vertices with each edge present independently with probability p.

Fakt 14.2 (Černovova nerovnost). Nechť $p \in [0,1]$ a $X_1, \dots, X_n \sim \mathrm{Ber}(p)$. Nechť $X = \sum_{i=1}^n X_i$. Pak

$$\Pr[|X - pn| > t] \leq 2\exp\left(-\frac{t^2}{2\,(pn + t/3)}\right).$$

Definice 14.3 (Regular pairs). Let G be a graph. Let $\delta, \varepsilon > 0$. We say $A \dot\cup B \subseteq V$ form a $\langle\delta,\varepsilon\rangle$-*regular pair* $\equiv$

$$(\forall A' \subseteq A)(\forall B' \subseteq B) \qquad (|A'| \geq \delta\,|A| \wedge |B'| \geq \delta\,|B|) \to |d(A',B') - d(A,B)| \leq \varepsilon.$$

The pair is ε-*regular* $\equiv$ it is $\langle\varepsilon,\varepsilon\rangle$-regular.

Lemma 14.4. Let $0 \geq p \geq 1$ and $\varepsilon > 0$. Then there exists an $n_0 \in \mathbb{N}$, such that

$$(\forall n)(\forall G \sim \mathcal{G}(n,p))(\forall A \dot\cup B \subseteq V(G)\,, |A| = |B| \geq n_0) \quad \Pr[\langle A,B\rangle \text{ form an } \varepsilon\text{-regular pair}] \geq 1-\varepsilon.$$

Důkaz. Nechť $A, B \subseteq V$ velikosti N. Pro jednu dvojici $A' \subseteq A$ a $B' \subseteq B$ velikosti alespoň εN odhadneme Černovovou nerovností pravděpodobnost, že se liší od předpokládané $|A'| \cdot |B'| \cdot p$, jako

$$\Pr\left[|d(A',B') - p| > \frac{\varepsilon}{2}\right] = \Pr\left[|e(A',B') - p\,|A'|\,|B'|| > \frac{\varepsilon\,|A'|\,|B'|}{2}\right] \leq 2\exp\left(-\frac{\left(\frac{\varepsilon|A'||B'|}{2}\right)^2}{2\left(p\,|A'|\,|B'| + \frac{\varepsilon|A'||B'|}{6}\right)}\right)$$

$$= 2\exp\left(-\frac{\varepsilon^2\,|A'|\,|B'|}{8\left(p+\frac{\varepsilon}{6}\right)}\right) \leq 2\exp\left(-\frac{\varepsilon^2 N^2}{8\left(p+\frac{\varepsilon}{6}\right)}\right).$$

Celkem je nejvýše 4^N párů A', B', a tedy pravděpodobnost, že existuje nějaký takový který se liší od předpokládané hustoty je nejvýše

$$4^N 2\exp\left(-\frac{\varepsilon^2 N^2}{8\left(p+\frac{\varepsilon}{6}\right)}\right) = \exp\left(N\ln 4 + \ln 2 - \frac{\varepsilon^2 N^2}{8\left(p+\frac{\varepsilon}{6}\right)}\right),$$

což je dominováno N^2 pro dost velké N, a tedy vyjde nejvýše $\varepsilon/2$. S pravděpodobností alespoň $1 - 2\frac{\varepsilon}{2}$ se tedy jedná o ε-regulární pár. ∎

LEMMA 14.5. Let G be a graph and $\langle A, B\rangle$ be an $\langle\delta, \varepsilon\rangle$-regular pair. Let $B' \subseteq B$ of size at least $\delta |B|$. Then

$$|\{v;\, \deg_{B'} v \geq (d(A,B) + \varepsilon)\, |B'|\}| < \delta\, |A|\,, \qquad |\{v;\, \deg_{B'} v \leq (d(A,B) - \varepsilon)\, |B'|\}| < \delta\, |A|\,.$$

Důkaz. Kdyby toto neplatilo, pak taková množina A' by byla dost velká na to, aby se na ni vztahovala podmínka $\langle\varepsilon, \delta\rangle$-regulárního páru, kterou by ale neprošla. ■

LEMMA 14.6. Let G be a graph and $\langle A, B\rangle$ be an $\langle\delta, \varepsilon\rangle$-regular pair, such that $|A| = |B| = n$ and $d(A,B) \geq \varepsilon + \delta$. Then the number of 4-cycles $v_1v_2v_3v_4$, such that $v_1, v_3 \in A, v_2, v_4 \in B$, is at least

$$\left((1-\delta)^2 (d(A,B) - \varepsilon)^4 - \frac{2}{n}\right) n^4.$$

Důkaz. Vybereme $v_1 \in A$ tak, aby měl alespoň $(d(A,B) - \varepsilon)\, |B| \geq \delta n$ sousedů v B. Takových v_1 je $(1-\delta)\, n$. Označíme $B_2 \subseteq B$ ty vrcholy, které mají jako souseda v_1. Pak $|B_2| \geq (d(A,B) - \varepsilon)\, n$.

Dále, nechť $A_3 \subseteq A$ jsou ty vrcholy které mají alespoň $(d(A,B) - \varepsilon)\, |B_2|$ sousedů v B_2. Pak protože $|B_2| \geq (d(A,B) - \varepsilon)\, n \geq \delta n$, je $|A_3| \geq \delta n$.

Konečně, spočítejme počet voleb v_1, v_2, v_3, v_4, které jsou spojené hranami. Nejprve vybereme v_1 a v_3 z A_1 a A_3. Na to máme $(1-\delta)^2 n^2$ možností. Pak ještě vybereme dva společné sousedy v_1, v_3, na což máme $(d(A,B) - \varepsilon)^2 n$ možností.

To je celkem

$$(1-\delta)^2 (d(A,B) - \varepsilon)^4 n^4.$$

Ještě odečteme neplatné cykly, kterých je ale nejvýše $2n^3$. To jsme chtěli dokázat. ■

LEMMA 14.7. Let G be a graph and $\langle A, B\rangle$, $\langle B, C\rangle$ and $\langle A, C\rangle$ be $\langle\delta, \varepsilon\rangle$-regular pairs, such that $|A| = |B| = |C| = n$, $\delta, \varepsilon \leq \frac{1}{2}$, and $d(A,B), d(B,C), d(A,C) \geq \varepsilon + \delta$. Then the number of triangles $v_1v_2v_3$, such that $v_1 \in A, v_2 \in B, v_3 \in C$, is at least

$$(1-2\delta)\, (d(A,B) - \varepsilon)\, (d(B,C) - \varepsilon)\, (d(A,C) - \varepsilon)\, n^3.$$

Důkaz. Nechť A_B jsou vrcholy z A které mají méně než $(d(A,B) - \varepsilon)\, n$ sousedů v B, podobně A_C. Budeme vybírat $A' = A \setminus (A_C \cup A_B)$. Určitě $|A'| \geq (1-2\delta)\, n$ z lemma 14.5.

Nechť nyní B' a C' jsou sousedi nějakého $v_1 \in A'$. Pak $|B'|, |C'| \geq \delta n$ (z předpokladů), a tedy $d(B', C') \geq d(B,C) - \varepsilon$. Máme tedy $(d(A,B) - \varepsilon)\, (d(A,C) - \varepsilon)\, n^2$ možností pro v_2 a v_3, a z toho $(d(B,C) - \varepsilon)$ mají mezi sebou hranu. Dohromady je to tedy

$$(1-2\delta)\, (d(A,B) - \varepsilon)\, (d(A,C) - \varepsilon)\, (d(B,C) - \varepsilon)\, n^3,$$

což jsme chtěli dokázat. ■

DEFINICE 14.8 (ε-REGULAR PARTITION). A partition $V_0 \dot\cup V_1 \dots V_r$ of $V(G)$ is *ε-regular* $\equiv$

(A1) $|V_0| \leq \varepsilon\, |V(G)|$,

(A2) $(\forall 1 \leq i \leq j \leq r)(|V_i| = |V_j|)$,

(A3) for all but at most εr^2 values of $1 \leq i < j \leq r$, the pair $\langle V_i, V_j\rangle$ is ε-regular.

We say that r is the *order* of the partition.

VĚTA 14.9 (REGULARITY LEMMA). Let $\varepsilon > 0, m_0 \in \mathbb{N}$. Then there exists $M \geq m_0 \in \mathbb{N}$, such that if G has at least m_0 vertices, then it has an ε-regular partition of order $m_0 \leq r \leq M$.

Idea důkazu. Začneme s nějakým rozkladem $\mathcal{P}$ splňující podmínky (A1) a (A2). Zadefinujeme *kvalitu* rozkladu jako

$$q(A,B) = |A| \cdot |B| \cdot d^2(A,B)\,, \qquad q(\mathcal{P}) = \sum_{i<j} q(P_i,P_j)\,.$$

Tato kvalita se nezhorší podrozdělením $\mathcal{P}$.

Konkrétně, pro A, B množiny v $\mathcal{P}$ a A', B' protipříklady proti tomu že A, B je ε-regulární, platí že když rozdělíme A na A' a $A \setminus A'$ a podobně B, vzroste kvalita o alespoň $\varepsilon^4 n$. Pak už jen stačí ukázat že toto zlepšení kvality pro dost velké n umíme

(1) zajistit dalším rozdělením aby všechny části měly stejný počet prvků,

(2) zajistit že tímto vzroste kvalita dostatečně, a díky tomu

(3) ukázat, že počet singletonů vzroste na nejvýše εn.

■

Věta 14.10 (Triangle removal lemma). $(\forall 1 \geq \alpha > 0)(\exists \beta > 0)(\exists n_0 \in \mathbb{N})(\forall G, |V(G)| \geq n_0)$

(1) G contains at least βn^3 triangles, or

(2) $(\exists X \subseteq E(G))\ |X| \leq \alpha n^2$ and $G - X$ contains no triangles.

Důkaz. Nechť $V_0, \ldots, V_r$ jsou z aplikace Regularity lemma pro vhodné ε a m_0. Označme $k = |V_i| \in \left[(1-\varepsilon)\,\frac{m}{n}, \frac{m}{n}\right]$. Rozdělme E na

(1) E_0 incidentní V_0: těch je εn^2,

(2) E_1 vedoucí uvnitř V_i: těch je $mk^2 \leq \frac{n^2}{m_0}$,

(3) E_2 vedoucí mezi V_i, V_j pro $d(V_i,V_j) \leq 2\varepsilon$: těch je $m^2 2\varepsilon k^2 \leq 2\varepsilon n^2$,

(4) E_3 vedoucí mezi V_i, V_j pro V_i, V_j ne-ε-regulární: těch je $\varepsilon m^2 k^2 \leq \varepsilon n^2$,

(5) E_4 zbytek.

Pokud $m_0 = \frac{1}{\varepsilon}$, je celkem $E' = E_0 \cup E_1 \cup E_2 \cup E_3$ velikosti nejvýše $5\varepsilon n^2$. Označme $\varepsilon = \frac{\alpha}{5}$. Pak všechny tyto hrany můžeme odstranit.

Pokud $G - E'$ obsahuje trojúhelník, znamená to, že existují V_i, V_j, V_k o hustotě alespoň 2ε. Dle lemma 14.7 je tedy v G alespoň

$$(1-2\varepsilon)\,(d(A,B)-\varepsilon)^3 \left(\frac{(1-\varepsilon)\,n}{M}\right)^3 \geq (1-2\varepsilon)\,\varepsilon^3 \left(\frac{(1-\varepsilon)\,n}{M}\right)^3 \geq \beta n^3$$

trojúhelníků pro vhodné β, jak vyžaduje tvrzení. ■

14.1 Finding Subgraphs

Fakt 14.11 (Turán). Každý graf s více než $\frac{k-2}{k-1} \cdot \frac{n^2}{2}$ hranami obsahuje K_k jako podgraf.

Lemma 14.12. Let $k \in \mathbb{N}_+$, H be a k-colorable graph. Let $d \in \mathbb{R}_+$. Then there exists $d > \varepsilon > 0$ and $n_1 \in \mathbb{N}$, such that for each graph G, and for each $V_1 \dot\cup \ldots \dot\cup V_k \subseteq V(G)$, $|V_1| = |V_2| \ldots |V_k| \geq n_1$, if for each i, j $\langle V_i, V_j\rangle$ is an ε-regular pair with $d(V_i,V_j) \geq d$, then $H \subseteq G$.

Důkaz. Mějme rozklad dle předpokladu a nechť φ je k-obarvení H. Nechť $x_1, \ldots, x_t$ jsou vrcholy H. Zvolíme vrcholy $v_1, \ldots, v_t$ grafu G tak, že

(1) $v_i \in V_{\varphi(x_i)}$,

(2) pokud $\varphi(x_i) \neq \varphi(x_j)$, pak $v_i v_j \in E$.

To už implikuje lemma. Volíme v_i postupně.

Nechť máme $v_1, \dots, v_p$, a nechť $C_i = \{v \in V_i;\ (\forall j)(vv_j \in E)\}$. Udržujeme, že

$$|C_i| \geq (d-\varepsilon)^p\, n,$$

pro všechna i.

Při volení v_{i+1} využijeme lemmatu 14.6 na všechny $j \neq \varphi(x_{i+1})$. Toto pro $(d-\varepsilon)^p \geq 2\varepsilon$ a dost velké n_1 platí. ■

Věta 14.13 (Erdös-Stone). Let H be a graph with chromatic number $k \geq 2$. Then

$$(\forall \alpha > 0)(\exists n_0)(\forall G, |V(G)| \geq n_0) \qquad |E(G)| \geq \left(1 - \tfrac{1}{k-1} + \alpha\right) \tfrac{|V(G)|^2}{2} \ \to\ H \subseteq G.$$

Důkaz. Použijeme lemma 14.12 s vhodným $d > 0$. Dostaneme ε a n_1. Pro vhodné m_0 dostaneme z Regularity lemmatu $V_0, \dots, V_m$. Označme $k = |V_i| \in \left[(1-\varepsilon)\,\tfrac{m}{n}, \tfrac{m}{n}\right]$. Opět rozdělíme:

(1) E_0 incidentní V_0: těch je εn^2,

(2) E_1 vedoucí uvnitř V_i: těch je $rk^2 \leq \tfrac{n^2}{m_0} \leq \varepsilon n^2$ pro $m_0 = 1/\varepsilon$,

(3) E_2 vedoucí mezi V_i, V_j pro $d(V_i, V_j) \leq d$: těch je $m^2 d k^2 \leq d n^2$,

(4) E_3 vedoucí mezi V_i, V_j pro V_i, V_j ne-ε-regulární: těch je $\varepsilon m^2 k^2 \leq \varepsilon n^2$,

(5) E_4 zbytek.

Dohromady je $E' = E_0 \cup \ldots \cup E_3$ velikosti nejvýše $(3\varepsilon + d)\, n^2 < 4dn^2 = \alpha \tfrac{n^2}{2}$ pro $d = \tfrac{\alpha}{8}$. Díky tomu má $G - E'$ $\left(1 - \tfrac{1}{k-1}\right) \tfrac{n^2}{2}$ hran. Označme G' graf s vrcholy odpovídajícím $V_1, \dots, V_m$ a hranami právě když $d(V_i, V_j) \neq 0$ v $G - E'$. Dostáváme

$$|E(G - E')| \leq s^2\, |E(G')| \leq \frac{n^2}{m^2}\, |E(G')|\,.$$

Úpravou tohoto výrazu dostaneme že i $|E(G')| \geq \left(1 - \tfrac{1}{k-1}\right) \tfrac{m^2}{2}$. Z Turánovy věty 14.11 dostáváme že v G' je K_k. Lemma 14.12 tedy říká, že $H \subseteq G$. ■

14.2 Arithmetic Progressions in Dense Sets

Lemma 14.14.

$$(\forall \delta > 0)(\exists n_0)(\forall n \geq n_0)\Big(\forall A \subseteq [n]^2, |A| > \delta n^2\Big)(\exists x, y \in [n])(\exists d \in \mathbb{N}_+) \qquad \langle x, y\rangle, \langle x, y+d\rangle, \langle x+d, y\rangle \in A.$$

Důkaz. Zkonstruujeme graf G s partitami R, S, T indexované $[2n]$. Hrany definujeme

(1) $r_x s_y \in E$ pro $(x, y) \in A$,

(2) $r_x t_z \in E$ pro $(x, z-x) \in A$,

(3) $s_y t_z \in E$ pro $(z-y, y) \in A$.

Pokud najdeme trojúhelník $r_x s_y t_z$, pak můžeme definovat $d = z - x - y$. Toto dokazuje lemma, pokud $z \neq x + y$. Takových trojic je ale jen $(2n)^2$.

Podle věty 14.10 buď obsahuje graf trojúhelníků dost (pro vhodné n), nebo obsahuje X velikosti $\mathcal{O}(n^2)$ která odstraní všechny trojúhelníky. No jo, ale G obsahuje alespoň $|A|$ „degenerovaných“ trojúhelníků, které jsou hranově disjunktní. To znamená, že pro vhodné α X nemůže pokrýt všechny trojúhelníky. To dokazuje lemma. ■

Věta 14.15 (Roth).

$$(\forall \gamma > 0)(\exists n_0)(\forall n \geq n_0)(\forall A \subseteq [n], |A| > \gamma n)(\exists x, d \in \mathbb{N}_+) \qquad x, x+d, x+2d \in A.$$

Důkaz. Definujeme $A = \{\langle x, y\rangle \,;\, x - y \in B\}$. Použijeme předchozí lemma pro vhodné konstanty a vyjde to. ■

Fakt 14.16 (Szemerédi).

$$(\forall \gamma > 0)(\forall k)(\exists n_0)(\forall n \geq n_0)(\forall A \subseteq [n]\,, |A| > \gamma n)(\exists x, d \in \mathbb{N}_+) \qquad x, x + d, \ldots, x + kd \in A.$$

Poznámka 14.17. The proof for a general k requires a non-straight-forward generalization of the Regularity Lemma to Hypergraphs.

Domněnka 14.17.1. Any $B \subseteq \mathbb{N}$ such that

$$\sum_{n \in B} \frac{1}{n} = \infty$$

contains arbitrarily long arithmetic progressions.

Otázka 15

Množinové systémy

Definice 15.1 (Množinový systém). *Množinový systém* je dvojice $(X, \mathcal{P})$, kde

- X je množina (bodů),
- $\mathcal{P} \subseteq \mathcal{P}(X)$ je množina „přímek“.

Definice 15.2 (KPR). *Konečná projektivní rovina* je množinový systém $(X, \mathcal{P})$, kde

(A1) $\forall x, y \in X\ \exists! P \in \mathcal{P} : x, y \in P$,

(A2) $\forall P, Q \in \mathcal{P}\ \exists! x \in X : P \cap Q = \{x\}$,

(A3) $\exists C \subseteq X : |C| = 4, \forall P \in \mathcal{P} : |C \cap P| \leq 2$.

Protože je přímka dvěma body jednoznačně určena, můžeme zavést značení $\overline{xy} \equiv$ přímka z $\mathcal{P}$ určena body $x, y \in X$.

Lemma 15.3. Pro každé dvě přímky $P, Q \in \mathcal{P}$ existuje $x \in X$, že $x \notin P$ ani $x \notin Q$.

Důkaz. Vychází hned z axiomu (A3). Buď $C \not\subseteq P \cup Q$, nebo vezmeme „ortogonální“ dvě přímky a jejich průsečík neleží ani na P ani Q. ■

Věta 15.4 (Velikost přímek). Nechť $(X, \mathcal{P})$ je KPR. Pak platí

$$(\forall P, Q \in \mathcal{P}) \qquad |P| = |Q|\,.$$

Důkaz. Nechť $x \notin P \cup Q$. Zkonstruujeme $\varphi : P \to Q$, které mapuje y na průsečík $\overline{xy} \cap Q$. Toto je prosté zobrazení dle axiomu (A2), a tedy $|P| \leq |Q|$. Obráceně identicky. ■

Definice 15.5 (Řád KPR). *Řád konečné projektivní roviny* je $n \in \mathbb{N} : \forall P \in \mathcal{P} : n = |P| - 1$.

Věta 15.6 (Vlastnosti řádu).

(1) $\forall x \in X : |\{P;\ P \in \mathcal{P} \wedge x \in P\}| = n + 1$

(2) $|X| = n^2 + n + 1$

(3) $|\mathcal{P}| = n^2 + n + 1$

Důkaz.

(1) Pro $x \in X$ existuje přímka P která x neobsahuje. Každý bod $y \in P$ určuje přímku $\overline{xy}$. Naopak, každá přímka procházející x musí protínat v jednom bodě P.

(2) Nechť $P \in \mathcal{P}$ a $x \notin P$. Každá přímka $\overline{yx}$ pro $y \in P$ obsahuje n dalších bodů, tedy bodů je alespoň $n^2 + n + 1$.

Naopak, každý $z \in X \setminus \{x\}$ určuje $\overline{zx}$ která protíná P v jednom bodě y, a je tedy identická s $\overline{xy}$. Tedy každý takový z byl započítán.

(3) plyne z věty 15.8.

■

Definice 15.7 (Duál). *Duální množinový systém* k $(X, \mathcal{P})$ je množinový systém $(\mathcal{P}, \mathcal{S})$, kde

$$\mathcal{S} = \{\{S \in \mathcal{P};\, x \in S\};\, x \in X\}$$

Věta 15.8. Duálem KPR řádu n je KPR řádu n.

Důkaz. Axiomy (A1) a (A2) jsou si navzájem duální. Axiom (A3) je duální sobě sama, a to pro $C = \{a, b, c, d\}$ jako

$$C^* = \left\{\overline{ab}, \overline{bc}, \overline{cd}, \overline{da}\right\}.$$

■

Věta 15.9 (Existence KPR). Pokud existuje algebraické těleso s n prvky, pak existuje KPR řádu n.

Důkaz. Nechť $\mathbb{F}$ je těleso. Zadefinujeme relaci $\sim$ nad $\mathbb{F}^3 \setminus \{\langle 0, 0, 0\rangle\}$ jako

$$\langle a, b, c\rangle \sim \langle \alpha a, \alpha b, \alpha c\rangle$$

pro všechna $\alpha \neq 0$.

Jako přímky označíme třídy ekvivalence $\sim$. Těch je

$$\frac{n^3 - 1}{n - 1} = n^2 + n + 1.$$

Ověříme axiomy. (A1) platí protože matice dvou prvků má hodnost 2, a tedy jádro má dimenzi 1. (A2) platí protože se jedná o soustavu dvou rovnic o dvou neznámých. Pro (A3) stačí vzít množinu

$$C = \{\langle 0, 0, 1\rangle, \langle 0, 1, 0\rangle, \langle 1, 0, 0\rangle, \langle 1, 1, 1\rangle\}.$$

■

Domněnka 15.9.1. KPR řádu n existuje, právě když existuje algebraické těleso s n prvky.

Dál mluvit o matroidech, viz otázka 21.

Okruh IV

Polyedrální optimalizace

Otázka 16

Teorie mnohostěnů

16.1 Afinní prostory

Definice 16.1 (Afinní prostor). Množina $L \subseteq \mathbb{R}^n$ je *afinní prostor* $\equiv L = x + V$ pro $x \in \mathbb{R}^n$ a V vektorový prostor.

Definice 16.2 (Dimenze afinního prostoru). *Dimenze* prostoru $L = x + V$ je $\dim L := \dim V$. Zvlášť definujeme $\dim \emptyset := -1$.

Definice 16.3 (Afinní obal). *Afinní obal* množiny X je $\operatorname{aff} X := \bigcap \{L;\ X \subseteq L \wedge L \text{ je afinní prostor}\}$.

Definice 16.4 (Afinní kombinace). *Afinní kombinace* bodů z X je

$$\sum_i x_i \alpha_i;\ \sum_i \alpha_i = 1 \wedge \forall i : x_i \in X.$$

Věta 16.5. Afinní obal množiny X je *afinním prostorem* a je roven množině afinních kombinací bodů z X, tj

$$\operatorname{aff} X = \left\{ \sum_i x_i \alpha_i;\ \sum_i \alpha_i = 1 \wedge \alpha_i \geq 0 \wedge \forall i : x_i \in X \right\}.$$

Definice 16.6 (Afinní nezávislost). Body $x_1, \ldots, x_k$ jsou *afinně nezávislé* $\equiv \forall \alpha_1, \ldots, \alpha_k \in \mathbb{R}$:

$$\left(\sum_i \alpha_i x_i = 0 \wedge \sum_i \alpha_i = 0 \right) \rightarrow (\forall i)(\alpha_i = 0)\,.$$

Pozorování 16.7. $x_0, \ldots, x_k$ jsou afinně nezávislé $\leftrightarrow (x_1 - x_0), (x_2 - x_0), \ldots (x_k - x_0)$ jsou lineárně nezávislé.

Definice 16.8 (Dimenze množiny). *Dimenze* množiny X je $\dim X := \dim \operatorname{aff} X$.

Pozorování 16.9. $\dim X = k \leftrightarrow$ maximální počet afinně nezávislých bodů v X je roven $k + 1$.

16.2 Konvexní množiny

Definice 16.10 (Konvexní množina). Množina X je *konvexní množina* $\equiv$

$$(\forall x, y \in X)(\forall t \in [0,1])(tx + (1-t)\,y \in X)\,.$$

Definice 16.11 (Konvexní obal). $\operatorname{conv} X$ je *konvexní obal* množiny $X \equiv$

$$\operatorname{conv} X := \bigcap \{C;\ X \subseteq C \wedge C \text{ je konvexní}\}\,.$$

Tvrzení 16.12. Nechť $Y \subseteq \operatorname{conv} X$. Pak $\operatorname{conv} Y \subseteq \operatorname{conv} X$.

Tvrzení 16.13. Nechť C je konvexní a $X \subseteq C$. Pak $\operatorname{conv} X \subseteq C$.

Tvrzení 16.14. Nechť $(\forall x \in X)\big(a^\top x \leq b\big)$. Pak platí

(1) $(\forall y \in \operatorname{conv} X)\big(a^\top y \leq b\big)$,

(2) $(\forall x \in X)\big(a^\top < b\big) \to (\forall y \in \operatorname{conv} X)\big(a^\top y < b\big)$,

(3) $(\forall y = \sum_i \alpha_i x_i \in \operatorname{conv} X)\big((\exists i)\big(a^\top x_i < b \wedge \alpha_i \neq 0\big) \to a^\top y < b\big)$.

Věta 16.15. Nechť $X \subseteq \mathbb{R}^d$. Pak

$$\operatorname{conv} X = \left\{ \sum_{i=1}^{k} \alpha_i x_i;\ k \in \mathbb{N},\ x_i \in X,\ \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Důkaz. Pro „$\supseteq$“ si uvědomme, že každá konvexní kombinace musí být v každé konvexní množině z definice $\operatorname{conv} X$. Pro „$\subseteq$“ si všimneme, že množina všech konvexních kombinacích je konvexní, a tedy je jedna z množin přes které se dělá průnik v definici $\operatorname{conv} X$. ∎

Věta 16.16 (Carathéodory). Nechť $X \subseteq \mathbb{R}^n$, $\dim X = d \geq 0$. Pak

$$\operatorname{conv} X = \left\{ \sum_{i=0}^{d} \alpha_i x_i;\ (\forall i \in [d])(x_i \in X \wedge \alpha_i \geq 0) \wedge \sum_{i} \alpha_i = 1 \right\}.$$

Důkaz. Směr „$\supseteq$“ plyne např. z předchozí věty. Druhý směr dokážeme sporem. Nechť $y \in \operatorname{conv} X$ a $y = \sum_{i=0}^{k} \alpha_i x_i$. Nechť k je minimální, a předpokládejme že $k > d$. Tedy $x_0, \ldots, x_k$ jsou afinně závislé, a tedy lze vyjádřit

$$\sum_{i=0}^{k} \beta_i x_i = 0,$$

kde $\sum_{i=0}^{k} \beta_i = 0$. Najdeme γ maximální takové, že

$$(\forall i) \qquad \gamma\beta_i + \alpha_i \geq 0.$$

Takové určitě existuje. Pak ale

$$\sum_{i=1}^{k} (\gamma\beta_i + \alpha_i)\, x_i = y$$

je konvexní kombinace, a z maximality γ je alespoň jeden koeficient roven nule. Dostáváme spor s minimalitou k. ∎

16.3 Mnohostěny

Definice 16.17 (Nadrovina). Nechť $a \in \mathbb{R}^n, b \in \mathbb{R}$. *Nadrovina* v $\mathbb{R}^n$ je

$$\left\{x \in \mathbb{R}^n;\, a^\top x = b\right\}.$$

Definice 16.18 (Poloprostor). Nechť $a \in \mathbb{R}^n, b \in \mathbb{R}$. *Poloprostor* v $\mathbb{R}^n$ je

$$\left\{x \in \mathbb{R}^n;\, a^\top x \leq b\right\}.$$

Definice 16.19 (Konvexní mnohostěn). *Konvexní mnohostěn*, polyhedr, je průnik konečně mnoha poloprostorů.

Definice 16.20 (Omezený konvexní mnohostěn). *Omezený konvexní mnohostěn*, polytop, je konvexní mnohostěn, který je omezený.

Věta 16.21 (O oddělování). Nechť C, D jsou konvexní, uzavřené, disjunktní mnohostěny. Nechť C je navíc omezená. Pak

$$(\exists b \in \mathbb{R})(\exists a \in \mathbb{R}^n)\big(C \subseteq \left\{x \in \mathbb{R}^n;\, a^\top x < b\right\} \wedge D \subseteq \left\{x \in \mathbb{R}^n;\, a^\top x > b\right\}\big).$$

Věta 16.22 (Minkowski-Weyl). Nechť $X \subseteq \mathbb{R}^n$. Pak X je konvexní omezený mnohostěn $\iff$

$$(\exists V \subseteq \mathbb{R}^n, V \text{ konečná})(X = \operatorname{conv} V).$$

Definice 16.23 (Tečná nadrovina). Nechť $P \subseteq \mathbb{R}^n$ je konvexní mnohostěn. *Tečná nadrovina* mnohostěnu P je $R = \{x;\, ax = b\}$ pro $a \in \mathbb{R}^n, b \in \mathbb{R}$, tž.

$$(\forall x \in P)(ax \leq b) \wedge (\exists x \in P)(ax = b).$$

Definice 16.24 (Stěna mnohostěnu). Nechť $P \subseteq \mathbb{R}^n$ je konvexní mnohostěn. Jeho *stěna*[1] je $P \cap R$ pro R jeho tečnou nadrovinu, nebo P, nebo $\emptyset$. *Vlastní stěna* je stěna, která není P ani $\emptyset$.

Definice 16.25 (Vrchol). *Vrchol* mnohostěnu P je stěna dimenze 0.

Definice 16.26 (Hrana). *Hrana* mnohostěnu P je stěna dimenze 1.

Definice 16.27 (Faseta). *Faseta* mnohostěnu P je stěna dimenze $\dim P - 1$.

Věta 16.28. Nechť $P \subseteq \mathbb{R}^n$ je konvexní mnohostěn, V jeho vrcholy, $V_{\text{ext}} := \{x \in \mathbb{R}^n;\, x \notin \operatorname{conv}(P \setminus \{x\})\}$. Pak $V = V_{\text{ext}}$.

Věta 16.29. Průnik dvou stěn P je stěnou P.

Věta 16.30 (Stěna stěny je stěna). Nechť $F \subseteq P$ je stěna P, $E \subseteq F$. Pak E je stěna $F \iff E$ je stěna P.

Definice 16.31 (Minimální mnohostěn). Nechť $P \subseteq \mathbb{R}^n$ je konvexní mnohostěn daný podmínkami $A'x = b'$ a $Ax \leq b$. Toto je *minimální popis* $P \equiv$ beze změny P nelze vynechat jednu rovnici/nerovnici, popř. změnit nerovnici na rovnici.

Tvrzení 16.32. Nechť $P = \{x;\, A'x = b'\} \neq \emptyset$, pro $A' \in \mathbb{R}^{m \times n}$. P je minimální $\iff m = \operatorname{rank} A'$.

Lemma 16.33. Nechť $\emptyset \neq P \subseteq \mathbb{R}^n$ je konvexní mnohostěn daný podmínkami $A'x = b'$ a $Ax \leq b$. Nechť $z \in P$ a $Az < b$. Pak $\dim P = n - \operatorname{rank} A'$. Navíc, je-li popis minimální, pak takové z existuje.

[1] Anglicky *face*.

Věta 16.34. Nechť $\emptyset \neq P \subseteq \mathbb{R}^n$ je konvexní mnohostěn daný minimálním popisem $A'x = b'$ a $Ax \leq b$. Pak v minimálním popisu P odpovídají vzájemně jednoznačně nerovnice fasetám P. Navíc, každá vlastní stěna je stěnou nějaké fasety.

Důsledek 16.35. Je-li $\dim P = n$, pak je minimální popis jednoznačný až na škálování.

Důsledek 16.36. Každá stěna P je průnikem faset.

16.4 Důkaz Minkowskiho–Weylovy věty

Definice 16.37. Let $P \subseteq \mathbb{R}^d$. An inequality $c^\top x \leq d$ is *valid* in $P \equiv$

$$(\forall y \in P)\big(c^\top y \leq d\big)\,.$$

Definice 16.38. A *face* of $P \subseteq \mathbb{R}^d$ is $F \subseteq P$, such that there is a valid inequality $c^\top x \leq d$, for which

$$F = \big\{x \in P;\, c^\top x = d\big\}\,.$$

- We call P and $\emptyset$ *improper* faces.
- A face F is a *facet* $\equiv \dim F = \dim P - 1$.

Tvrzení 16.39. Let $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$ and $P = \big\{x \in \mathbb{R}^d;\, Ax \leq b\big\}$. Then $\emptyset \neq F \subseteq P$ is a face of P, iff there is $I \subseteq [m]$, such that

$$F = \{x \in P;\, A_I x = b_I\}\,.$$

Důsledek 16.40. Face of a face is a face.

Definice 16.41. An *extreme point* (or *vertex*) is a face of dimension 0.

Definice 16.42. The set $P \subseteq \mathbb{R}^d$ is *pointed* $\equiv$ it does not contain a line, meaning for all $x, y \in \mathbb{R}^d$ and $y \neq 0$, there is a t, such that

$$x + ty \notin P.$$

Tvrzení 16.43. Let $P \subseteq \mathbb{R}^d$ be a closed, convex set and let $v \notin P$. Then there exists a separating hyperplane, i.e., α, β, such that

$$\alpha^\top v < \beta, \qquad (\forall x \in P)\big(\alpha^\top x > \beta\big)\,.$$

Tvrzení 16.44. A closed, convex set $P \subseteq \mathbb{R}^d$ is pointed, iff it has an extreme point.

Pozorování 16.45. The set of extreme points of a polyhedron P is finite.

Lemma 16.46. A bounded $\mathcal{H}$-polyhedron is equal to the convex hull of its extreme points.

Definice 16.47 (Polar dual). The *polar dual* of P is $P^\Delta = \big\{y \in \mathbb{R}^d;\, y^\top x \leq 1, x \in P\big\}$.

Tvrzení 16.48. Let $P \subseteq \mathbb{R}^d$ be closed, convex, centered. Then

$$\big(P^\Delta\big)^\Delta = P.$$

Tvrzení 16.49. Let $P \subseteq \mathbb{R}^d$ be a centered $\mathcal{V}$-polyhedron. Then P^Δ is an $\mathcal{H}$-polyhedron.

Tvrzení 16.50. Let $P \subseteq \mathbb{R}^d$ be a centered $\mathcal{H}$-polyhedron. Then P^Δ is a $\mathcal{V}$-polyhedron.

Lemma 16.51. Let $P \subseteq \mathbb{R}^d$ be a convex set containing the line $\{v + tw;\, t \in \mathbb{R}\}$. Define $P' = \big\{x \in P;\, w^\top x = w^\top v\big\}$. Then

(1) if P is an $\mathcal{H}$-polyhedron, then P' is an $\mathcal{H}$-polyhedron, and

(2) if P' is a $\mathcal{V}$-polyhedron, then P is a $\mathcal{V}$-polyhedron.

Definice 16.52 (Homogenized cone). Let $P \subseteq \mathbb{R}^d$. Define the set of rays of P to be the set $R(P) = \{v \in \mathbb{R}^d;\ (\forall t \geq 0)(tv \in P)\}$. Define the *homogenized cone* of P as

$$\operatorname{homog}(P) := \operatorname{cone}\{\binom{x}{1};\ x \in P\} + \operatorname{cone}\{\binom{v}{0};\ v \in R(P)\} \in \mathbb{R}^{d+1}.$$

Lemma 16.53. If P is an $\mathcal{H}$-polyhedron, then $\operatorname{homog}(P)$ is an $\mathcal{H}$-cone. If $\operatorname{homog}(P)$ is a $\mathcal{V}$-cone, then P is a $\mathcal{V}$-polyhedron.

Lemma 16.54. The set P is pointed, iff $\operatorname{homog}(P)$ is pointed.

Lemma 16.55. Let $P \subseteq \mathbb{R}^d$ be a pointed $\mathcal{H}$-cone or $\mathcal{V}$-cone. Then there exists a $c \in \mathbb{R}^d$, such that $c^\top x > 0$ for all non-zero $x \in P$.

Lemma 16.56. Let $P \subseteq \mathbb{R}^d$ be pointed and centered and let $c \in \mathbb{R}^d$, such that $c^\top x > 0$ for all non-zero $x \in P$. Define $P' = \{x \in P;\ c^\top x = 1\}$. Then

(1) if P is an $\mathcal{H}$-cone then P' is an $\mathcal{H}$-polytope,

(2) if P' is a $\mathcal{V}$-polytope then P is a $\mathcal{V}$-cone.

Věta 16.57 (Minkowski-Weyl).

(1) P is a $\mathcal{V}$-polytope, iff P is an $\mathcal{H}$-polytope.

(2) P is a $\mathcal{V}$-polyhedron, iff P is an $\mathcal{H}$-polyhedron.

(3) P is a $\mathcal{V}$-cone, iff P is an $\mathcal{H}$-cone.

Otázka 17

Problém obchodního cestujícího

Definice 17.1 (Problém čínského posťáka). Nechť $\ell : V \to \mathbb{Q}^+$. *Problém čínského poštáka* Čínský-Pošťák je najít minimální uzavřený sled procházející všechny hrany alespoň jednou.

Definice 17.2 (Problém obchodního cestujícího). Nechť $\ell : V \to \mathbb{Q}^+$. *Problém obchodního cestujícího* Obchodní-Cestující je najít minimální uzavřený sled procházející všechny vrcholy alespoň jednou.

Fakt 17.3. Obchodní-Cestující je NP-úplný.

Věta 17.4. Nechť G má všechny stupně sudé. Pak je řešením Čínský-Pošťák Eulerovský tah, který je nalezitelný v polynomiálním čase.

Tvrzení 17.5. Pokud existují vrcholy s lichým stupněm, pak je délka řešení ostře větší než $\ell(E)$.

Definice 17.6 (T-join). Nechť $T \subseteq V$. Pak $E' \subseteq E$ je *T-join* $\equiv$

$$\deg_{E'} v = 2k + 1 \leftrightarrow v \in T.$$

Věta 17.7. Nechť E' je množina hran minimálního průchodu Čínský-Pošťák, které se projdou více než 1. Pak

(1) $(\forall e \in E')$ se projde přesně dvakrát,

(2) E' je minimální T-join pro $T = \{v;\ \deg_G v = 2k + 1\}$.

Důsledek 17.8. Pokud najdeme minimální T-join, vyřešíme Čínský-Pošťák.

Algoritmus 17.9 (Min T-join).
Vstup: G graf, $\ell : V \to \mathbb{Q}^+$.
Výstup: E' minimální T-joint.

1: $H \leftarrow \left(T, \binom{T}{2}\right)$
2: Pro $u, v \in T$, definujeme $w(uv)$ jako délku nejkratší cesty mezi u, v v G.
3: $M \leftarrow$ perfektní párování v H minimální délky dle w.
4: $E' \leftarrow \{e \in E;\ e$ je v konkrétní nejkratší cestě mezi nějakými vrcholy v $T\}$

Věta 17.10. Výstup algoritmu je minimální T-join.

Fakt 17.11. Když G je rovinný, tak generující funkce T-joinů je polynomiálně spočetná.

Poznámka 17.12. Pro Obchodní-Cestující předpokládáme úplný graf, a trojúhelníkovou nerovnost.

Algoritmus 17.13 (Christofidesova heuristika).
Vstup: G úplný, ℓ splňující trojúhelníkovou nerovnost.
Výstup: Aproximační řešení Obchodní-Cestující.

1: $T \leftarrow$ minimální kostra v G.
2: $M \leftarrow$ minimální perfektní párování nad vrcholy lichého stupně v T.
3: $J \leftarrow T \dot\cup M$ (případně s multihranami).
4: Dokud $(\exists u)(\deg u > 2)$:
5: Nahradíme jednu cestu procházející u „zkratkou“ přímo mezi dvěma vrcholy.

Věta 17.14. Christofidesova heuristika dá řešení nejvýše $\frac{3}{2}$-krát delší než optimum.

Poznámka 17.15. Experimentálně ukázáno, že to bývá jen cca 1.09-krát delší.

Otázka 18

Speciální matice

18.1 Totálně unimodulární matice

Definice 18.1 (Totální unimodularita). Nechť $A \in \mathbb{R}^{m \times n}$. A je *totálně unimodulární* $\equiv (\forall I \subseteq [m])(\forall J \subseteq [n])$

$$\det A^{IJ} \in \{0, 1, -1\}\,.$$

Důsledek 18.2. Každá totálně unimodulární matice je z $\{0, -1, 1\}^{m \times n}$.

Věta 18.3. Nechť $A \in \mathbb{R}^{m \times n}$ je totálně unimodulární. Nechť $b \in \mathbb{Z}^m$. Pak oba polyhedry $\{x;\ Ax = b \wedge x \geq 0\}$ a $\{x;\ Ax \geq b \wedge x \geq 0\}$ mají všechny vrcholy celočíselné.

Důsledek 18.4. Pokud má odpovídající lineární program pro libovolné c optimum, pak má i celočíselné optimum.

Tvrzení 18.5. Totální unimodularita se nezmění

- transpozicí,
- permutací řádků/sloupců,
- vynásobením řádku/sloupce -1,
- přidáním/odstraněním řádku/sloupce s nejvýš 1 jedničkou, nulami jinde.

Věta 18.6. Nechť $A \in \{0, 1, -1\}^{m \times n}$ má v každém sloupci nejvýše dva nenulové prvky, a pokud jsou dva, pak si nejsou rovny. Pak A je totálně unimodulární.

Věta 18.7. Nechť A má v každém sloupci ≤ 2 nenuly. Pak A je totálně unimodulární, právě když vynásobením některých řádků -1 lze dostat tvar z předchozí věty.

Věta 18.8. Matice incidence orientovaných grafů jsou totálně unimodulární.

Věta 18.9. Matice incidence neorientovaných grafů jsou totálně unimodulární, právě když graf je bipartitní.

Důsledek 18.10. TU nám může pomoci s větou o celočíselnosti toků a duality max. toku a min. řezu.

18.2 Representation of Matroids

ZNAČENÍ 18.11. In this section, we denote by $\mathbb{F}$ a *field*, with characteristic $\mathrm{char}(\mathbb{F})$.

DEFINICE 18.12 (REPRESENTABILITY). A matroid $\mathcal{M}$ is

(1) $\mathbb{F}$-*representable* $\equiv$ there is a matrix A over $\mathbb{F}$, such that

$$\mathcal{M}(A) = \mathcal{M}.$$

(2) *representable* $\equiv$ there is a field $\mathbb{F}$, such that $\mathcal{M}$ is $\mathbb{F}$-representable,

(3) *regular* $\equiv$ it is representable for all $\mathbb{F}$.

LEMMA 18.13 (RELAXATION). Let $X \subseteq E$ be a circuit and a hyperplane. Then $\mathcal{B}' = \mathcal{B} \cup \{X\}$ are bases of a matroid with circuits

$$\mathcal{C}' = (\mathcal{C} - X) \cup \{X + e;\, e \in E \setminus X\}.$$

DEFINICE 18.14 (FUNDAMENTAL CIRCUIT). Let $B \in \mathcal{B}$. The *fundamental circuit of* $e \in E \setminus B$ *with respect to* B is the unique circuit $C_e^B \in \mathcal{C}$, such that

$$C_e^B \subseteq B + e.$$

DEFINICE 18.15 (FC INCIDENCE MATRIX). Let $B \in \mathcal{B}$ and let $L = E \setminus B$. Define the *fundamental circuit incidence matrix* of B as $D^{\#} \in \mathbb{F}^{B \times L}$, defined as

$$D^{\#}_{b,e} = \begin{cases} 1 & \text{if } b \in C_e^B, \\ 0 & \text{otherwise.} \end{cases}$$

DEFINICE 18.16 (FANO, NON-FANO). The *Fano matroid* F_7 is the representation of the Fano plane, and the *non-Fano* matroid is F_7^-, defined as the relaxation of F_7 in an arbitrary circuit/hyperplane.

VĚTA 18.17.

(1) F_7 is representable over $\mathbb{F}$, iff $\mathrm{char}(\mathbb{F}) = 2$.

(2) F_7^- is representable over $\mathbb{F}$, iff $\mathrm{char}(\mathbb{F}) \neq 2$.

DŮSLEDEK 18.18. $F_7 \oplus F_7^-$ is not representable.

TVRZENÍ 18.19. The class of matroids representable over $\mathbb{F}$ is closed under dual, deletion, contraction, minor.

TVRZENÍ 18.20. $U_{2,n}$ is $\mathbb{F}$-representable, iff $|\mathbb{F}| \geq n - 1$.

DEFINICE 18.21 (EXCLUDED MINOR). The matroid $\mathcal{M}$ is an *excluded minor* of a class of matroids $\mathbb{M} \equiv$ it is minimal with respect to taking minors, such that $\mathcal{M} \notin \mathbb{M}$ and for all e, $\mathcal{M} \setminus e, \mathcal{M} / e \in \mathbb{M}$.

DŮSLEDEK 18.22. For all $p \in \mathbb{P}$, the matroid $U_{2,p+2}$ is an excluded minor for $\mathbb{Z}_p$-representability.

18.2.1 Binary Matroids

DEFINICE 18.23. Let $k \geq 3$, $E = \{x_1, \ldots, x_k, y_1, \ldots, y_k\}$ and define

$$\mathcal{C}_1 = \{\{x_i, x_j, y_i, y_j\}\,;\, i \neq j\}\,, \qquad \mathcal{C}_2 = \{C \subseteq E;\, 2 \setminus |C \cap \{y_1, \ldots, y_k\}|\}\,.$$

Finally, let $\mathcal{M}$ be a matroid such that $\mathcal{C}(\mathcal{M}) = \mathcal{C}_1 \cup \mathcal{C}_2$.

POZOROVÁNÍ 18.24. $\mathcal{M}$ is binary.

POZOROVÁNÍ 18.25. Each $X \in \mathcal{C}$ is both a circuit and a hyperplane.

DEFINICE 18.26. Define $\mathcal{M}_X$ to be the relaxation of $\mathcal{M}$ with $X \in \mathcal{C}$.

POZOROVÁNÍ 18.27. $\mathcal{M}_X$ is not binary.

DŮSLEDEK 18.28. To know whether or not $\mathcal{M}$ is binary, we must check $\Omega\big(2^{k-1}\big)$ subsets.

VĚTA 18.29. $\mathcal{M}$ is not binary, iff it has a $U_{2,4}$-minor.

18.2.2 Ternary Matroids

TVRZENÍ 18.30. The forbidden minors of ternary matroids are $U_{2,5}, U_{3,5}, F_7, F_7^*$.

18.2.3 Regular Matroids

DEFINICE 18.31 (TU). A matrix $A \in \mathbb{R}^{m\times n}$ is *totally unimodular* $\equiv$ for each square sub-matrix A', it holds

$$\det A' \in \{0, \pm 1\}\,.$$

DEFINICE 18.32. A matroid $\mathcal{M}$ is *unimodular* $\equiv$ it has a representation $\mathcal{M} = \mathcal{M}(A)$ where A is TU.

LEMMA 18.33. If A is totally unimodular and B is obtained from A via pivoting, then B is also totally unimodular.

DŮSLEDEK 18.34. The class of unimodular matroids is closed under dual and minors.

LEMMA 18.35. If $\mathcal{M}$ is binary and $\mathbb{F}$-representable for some $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq 2$ with representation $A \in \mathbb{F}^{r\times n}$, such that $A \in \{\pm 1, 0\}^{r\times n}$, and $B \in \mathbb{F}^{r\times n}$ is obtained by pivoting (over $\mathbb{F}$), then

$$B \in \{\pm 1, 0\}^{r\times n}\,.$$

VĚTA 18.36. The following are equivalent.

(1) $\mathcal{M}$ is unimodular,

(2) $\mathcal{M}$ is regular,

(3) $\mathcal{M}$ is binary and representable over $\mathbb{F}$ with $\mathrm{char}(\mathbb{F}) \neq 2$.

Otázka 19

Celočíselnost

Definice 19.1 (Platný řez). Nechť $P = \{x \in \mathbb{R}^n;\ x \geq 0 \wedge Ax \geq b\}$. Nechť $Z = P \cap \mathbb{Z}^n$. Nechť $\alpha x \leq \beta$ platí $\forall x \in P$. Pak $\lfloor \alpha \rfloor x \leq \lfloor \beta \rfloor$ je *platný řez*.

Pozorování 19.2. Nechť $\alpha x \leq \beta$ je platný řez. Pak

$$(\forall x \in Z)(\alpha x \leq \beta)\,.$$

Lemma 19.3. Mějme $\alpha \in \mathbb{Z}^n, \beta \in \mathbb{Z}$. Pak $\alpha x \leq \beta$ je platný řez $\leftrightarrow$

$$(\exists y \geq 0)\big(\exists u \in \mathbb{Z}^n : u \leq \lfloor A^\top y \rfloor\big)\big(\alpha = u^\top \wedge \beta = \lfloor b^\top y \rfloor\big)\,.$$

Věta 19.4 (Universalita metody řezu). Nechť $P = \{x \in \mathbb{R}^n;\ x \geq 0 \wedge Ax \geq b\}$. Nechť $Z = P \cap \mathbb{Z}^n$. Nechť $\alpha x \leq \beta$ je platná v Z. Pak $\alpha x \leq \beta$ lze odvodit ze soustvy $Ax \geq b$ postupným přidáváním platných řezů.

Otázka 20

Párování a toky v sítích

Párování je pokryto otázkou 26 a toky v sítích jsou pokryty primárně otázkou 24.

Otázka 21

Teorie matroidů

Definice 21.1 (Matroid). A *matroid* is $\mathcal{M} = \langle E, \mathcal{I} \rangle$, where

(1) E is a finite, non-empty set,

(2) $\mathcal{I} \subseteq \mathcal{P}(E)$ is the set of *independent sets*, satisfying

$$\emptyset \in \mathcal{I}, \tag{21.1.1}$$

$$I' \in \mathcal{I}, \qquad (\forall I \in \mathcal{I})(\forall I' \subseteq I) \tag{21.1.2}$$

$$(\exists e \in J)\ I + e \in \mathcal{I}, \qquad (\forall I, J \in \mathcal{I}, |I| < |J|) \tag{21.1.3}$$

Definice 21.2. Let $\mathcal{M}$ be a matroid. The *circuits* of $\mathcal{M}$ is the set

$$\mathcal{C} = \{S \subseteq E;\ S \notin \mathcal{I} \wedge (\forall e \in E)(S - e \in \mathcal{I})\}.$$

Lemma 21.3. Let $\mathcal{M}$ be a matroid and $\mathcal{C}$ its circuits. Then

$$\emptyset \notin \mathcal{C}, \tag{21.3.1}$$

$$C, D \in \mathcal{C}, C \subseteq D \ \rightarrow\ C = D, \tag{21.3.2}$$

$$C, D \in \mathcal{C}, C \cap D \neq \emptyset, e \in C \cap D, C \neq D \ \rightarrow\ (\exists E \in \mathcal{C})\ E \subseteq (C \cup D) - e. \tag{21.3.3}$$

Lemma 21.4. If a set $\mathcal{C}$ satisfies conditions (21.3.1) to (21.3.3), then there exists a matroid $\mathcal{M}$ such that $\mathcal{C}$ are exactly its circuits.

Definice 21.5. Let $\mathcal{M}$ be a matroid. The *bases* of $\mathcal{M}$ are

$$\mathcal{B} = \{B \subseteq E;\ B \in \mathcal{I} \wedge (\forall e \in E)(B + e \notin \mathcal{I})\}.$$

Lemma 21.6. Let $B, B' \in \mathcal{B}$. Then

$$|B| = |B'|.$$

Lemma 21.7. Let $\mathcal{M}$ be a matroid and $\mathcal{B}$ its bases. Then

$$\mathcal{B} \neq \emptyset, \tag{21.7.1}$$

$$B, B' \in \mathcal{B}, e \in B \setminus B' \ \rightarrow\ (\exists f \in B' \setminus B)(B - e + f \in \mathcal{B}). \tag{21.7.2}$$

Lemma 21.8. If a set $\mathcal{B}$ satisfies conditions (21.7.1) and (21.7.2), then there exists a matroid $\mathcal{M}$ such that $\mathcal{B}$ are exactly its bases.

Definice 21.9. Let $\mathcal{M}$ be a matroid. Define its *rank function* $r : \mathcal{P}(E) \rightarrow \mathbb{N}$, such that

$$(\forall X \subseteq E) \qquad r(X) = \max_{I \subseteq X, I \in \mathcal{I}} |I|.$$

LEMMA 21.10. Let $\mathcal{M}$ be a matroid and r its rank function. Then

$$|X| \geq r(X) \geq 0, \qquad (\forall X \subseteq E)\,, \tag{21.10.1}$$
$$r(X) \geq r(Y)\,, \qquad (\forall X \subseteq E)(\forall Y \subseteq X)\,, \tag{21.10.2}$$
$$r(X) + r(Y) \geq r(X \cup Y) + r(X \cap Y)\,, \qquad (\forall X, Y \subseteq E)\,. \tag{21.10.3}$$

LEMMA 21.11. If a function r satisfies conditions (21.10.1) to (21.10.3) and $X, Y \subseteq E$, then

$$(\forall e \in Y)(r(X + e) = r(X)) \to r(X) = r(X \cup Y)\,.$$

LEMMA 21.12. If a function r satisfies conditions (21.10.1) to (21.10.3), then there exists a matroid $\mathcal{M}$ such that r is its rank function.

21.1 Basic Operations

DEFINICE 21.13 (SUM). Let $\mathcal{M}_1, \mathcal{M}_2$ be matroids, such that $E_1 \cap E_2 = \emptyset$. We define $\mathcal{M}_1 \oplus \mathcal{M}_2 = \mathcal{M}$, such that

$$E = E_1 \cup E_2, \qquad \mathcal{I} = \{X;\, X \cap E_1 \in \mathcal{I}_1, X \cap E_2 \in \mathcal{I}_2\}\,.$$

POZOROVÁNÍ 21.14. This is a matroid.

POZOROVÁNÍ 21.15.

$$\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2.$$

POZOROVÁNÍ 21.16.

$$r(X) = r_1(X \cap E_1) + r_2(X \cap E_2)\,.$$

PŘÍKLAD 21.17. Let $\mathbb{F}$ be a field. Let $A \in \mathbb{F}^{m \times n}$. Define $\mathcal{M}(A) = \mathcal{M}$, where $E = \{A_{*i};\, i \in [n]\}$ and $\mathcal{I}$ are independent vectors.

Without loss of generality, we can assume that $A = (I_r \mid D)$, which we call the *standard representation*.

PŘÍKLAD 21.18. The *uniform matroid* $U_{n,r}$ is defined on $E = [n]$, and

$$\mathcal{I} = \{X \subseteq [n]\,;\, |X| \leq r\}\,.$$

LEMMA 21.19. An alternate version of condition (21.7.2) is

$$B, B' \in \mathcal{B}, f \in B' \setminus B \;\to\; (\exists e \in B \setminus B')(B - e + f \in \mathcal{B})\,. \tag{21.19.1}$$

DEFINICE 21.20 (DUAL). The *dual matroid* to $\mathcal{M}$ is $\mathcal{M}^*$, such that

$$\mathcal{B}^* = \{E \setminus B;\, B \in \mathcal{B}\}\,.$$

POZOROVÁNÍ 21.21.

$$U^*_{n,r} = U_{n,n-r},$$
$$\mathcal{M}(I_r \mid D) = \mathcal{M}\big(D^\top \mid I_{n-r}\big)\,.$$

POZNÁMKA 21.22. For terms of the dual, we often use co-(term), so for example co-circuit, co-basis, etc.

TVRZENÍ 21.23. Let $X \subseteq E$. Then

$$r^*(X) = |X| - r(E) + r(E \setminus X)\,.$$

DEFINICE 21.24. A set $H \subseteq X$ is a *hyperplane* $\equiv$ H is maximal in inclusion such that $r(H) < r(E)$.

LEMMA 21.25. A set $H \subseteq X$ is a hyperplane, iff $E \setminus H \in \mathcal{C}^*$.

LEMMA 21.26. Let $C \in \mathcal{C}$ and $C^* \in \mathcal{C}^*$. Then

$$|C \cap C^*| \neq 1.$$

POZOROVÁNÍ 21.27.

$$(\mathcal{M}_1 \oplus \mathcal{M}_2)^* = \mathcal{M}_1^* \oplus \mathcal{M}_2^*.$$

DEFINICE 21.28. Let $T \subseteq E$. We define a *deletion* $\mathcal{M} \setminus T$ as a matroid $\langle E \setminus T, \{I \setminus T;\, I \in \mathcal{I}\}\rangle$.

POZOROVÁNÍ 21.29.

$$r_{\mathcal{M}\setminus T}(X) = r(X)\,.$$

DEFINICE 21.30. Let $T \subseteq E$. We define a *contraction* $\mathcal{M} \,/\, T$ as $(\mathcal{M}^* \setminus T)^*$.

TVRZENÍ 21.31.

$$r_{\mathcal{M}/T}(X) = r(X \cup T) - r(T)\,.$$

TVRZENÍ 21.32. Let $T \subseteq E$ and B_T be a maximal independent subset of T. Then

$$\mathcal{I}_{\mathcal{M}/T} = \{I \subseteq E \setminus T;\, I \cup B_T \in \mathcal{I}\}\,.$$

TVRZENÍ 21.33. Let $T_1, T_2 \subseteq E$ be disjoint. Then

(1) $\mathcal{M} \setminus T_1 \setminus T_2 = \mathcal{M} \setminus T_2 \setminus T_1$,

(2) $\mathcal{M} \,/\, T_1 \,/\, T_2 = \mathcal{M} \,/\, T_2 \,/\, T_1$,

(3) $\mathcal{M} \setminus T_1 \,/\, T_2 = \mathcal{M} \,/\, T_2 \setminus T_1$.

DEFINICE 21.34. The matroid $\mathcal{N}$ is a *minor* of $\mathcal{M} \equiv$ there exist $T_1, T_2 \subseteq E$ disjoint, such that

$$\mathcal{N} = \mathcal{M} \setminus T_1 \,/\, T_2.$$

TVRZENÍ 21.35. Let $C' \subseteq E \setminus T$. Then $C' \in \mathcal{C}_{\mathcal{M}/T}$, iff it is a non-empty inclusion-wise minimal member of $\{C \setminus T;\, C \in \mathcal{C}\}$.

21.2 Connectivity

DEFINICE 21.36. Let γ be the following relation on E:

$$\langle e, f\rangle \in \gamma \equiv e = f \vee (\exists C \in \mathcal{C})(e, f \in C)\,.$$

LEMMA 21.37.

$$(\forall C, D \in \mathcal{C})(\forall e \in C \cap D)(\forall f \in C \setminus D)(\exists G \in \mathcal{C})(G \subseteq (C \cup D) - e \wedge f \in G)\,.$$

VĚTA 21.38. The relation γ is an equivalence.

DEFINICE 21.39. *Components of connectivity* of a matroid $\mathcal{M}$ are the equivalence classes of γ. The matroid $\mathcal{M}$ is *connected* $\equiv$ it has one component of connectivity.

POZNÁMKA 21.40. In graphic matroids, this corresponds to 2-connectivity.

DEFINICE 21.41. $X \subseteq E$ is a *separator* $\equiv X$ is the union of a subset of components of connectivity.

POZOROVÁNÍ 21.42. X is a separator, iff

$$(\forall C \in \mathcal{C}) \qquad C \subseteq X \vee C \cap X = \emptyset.$$

TVRZENÍ 21.43. X is a separator, iff

$$r(X) + r\left(\overline{X}\right) = r(E)\,.$$

TVRZENÍ 21.44. Let X be a separator. Then for all $F \subseteq E$,

$$r(F) = r(F \cap X) + r(F \setminus X)\,.$$

TVRZENÍ 21.45. $\mathcal{M} \setminus X = \mathcal{M} \,/\, X$, iff X is a separator.

TVRZENÍ 21.46. X is a separator, iff

$$r(X) + r^*(X) = |X|\,.$$

DŮSLEDEK 21.47. X is a separator in $\mathcal{M}$, iff it is a separator in $\mathcal{M}^*$.

21.3 Greedy Algorithm

ALGORITMUS 21.48 (GREEDY).
Vstup: Matroid $\mathcal{M}$ and $w : E \to \mathbb{R}_0^+$.
Výstup: $B \in \mathcal{I}$ with maximal $w(B)$.

1: Sort by weights in descending order: $e_1, \ldots, e_n$.
2: $A \leftarrow \emptyset$.
3: For $i \in \{1, \ldots, n\}$:
4: If $A + e_i \in \mathcal{I}$, $A \leftarrow A + e_i$.

ZNAČENÍ 21.49. We assume that the matroid is given by an independence oracle, taking time τ to answer us.

VĚTA 21.50. The Greedy algorithm works in $\mathcal{O}(n \cdot \tau + n \log n)$, and it correctly finds the solution.

VĚTA 21.51. If $\langle E, \mathcal{I} \rangle$ is an arbitrary set system for which $\emptyset \in \mathcal{I}$, and the Greedy algorithm works for all w, then $\langle E, \mathcal{I} \rangle$ is a matroid.

21.4 Matroid Intersection

VĚTA 21.52 (MATROID INTERSECTION THEOREM). Let $\mathcal{M}_1, \mathcal{M}_2$ be matroids on the same E. Then

$$\max_{I \in \mathcal{I}_1 \cap \mathcal{I}_2} |I| = \min_{E_1 \cup E_2 = E} r_1(E_1) + r_2(E_2)\,.$$

TVRZENÍ 21.53. MIT can be implemented in $\mathcal{O}\left(r^2 n \tau\right)$, where $r = \max\{r(\mathcal{M}_1)\,, r(\mathcal{M}_2)\}$.

VĚTA 21.54. A matroid $\mathcal{M}$ contains k disjoint bases, iff

$$(\forall S \subseteq E) \qquad |E \setminus S| \geq k \cdot (r(E) - r(S))\,.$$

DŮSLEDEK 21.55. $\mathcal{M}$ can be covered by k independent sets, iff

$$(\forall S \subseteq E) \quad |S| \leq k \cdot r(S)\,.$$

Věta 21.56 (Nash-Williams). The graph G has k disjoint spanning trees, iff each partition $V_1 \dot\cup \ldots \dot\cup V_\ell = V$ has number of edges between distinct V_i at least

$$k \cdot (\ell - 1)\,.$$

Definice 21.57. Let $G = \langle V, E\rangle$. Define matroid $\mathcal{M}^+(G)$ such that

$$I \in \mathcal{I}^+ \leftrightarrow I \text{ has at most 1 circuit.}$$

Pozorování 21.58. $\mathcal{M}^+$ is a matroid with rank

$$r^+(X) = \begin{cases} r(X) & \text{if } X \in \mathcal{I}, \\ r(X)+1 & \text{otherwise.} \end{cases}$$

Věta 21.59. There is an algorithm for maximum average degree of a graph.

Otázka 22

Elipsoidová metoda

Algoritmus 22.1 (Ellipsoid Method).
Vstup: A bounded polytope P, either empty, or of full dimension.
Výstup: A point $x \in P$, if it exists.

1: $E \leftarrow E(z_0, C_0)$. ▷ Choose z_0, C_0 by previous lemmas.
2: $k \leftarrow 0$.
3: While $z_k \notin P$:
4: $\quad E_{k+1} \leftarrow E(z_{k+1}, C_{k+1})$, such that $P \subseteq E_{k+1}$ and $\operatorname{vol}(E)_{k+1} \leq q \cdot \operatorname{vol}(E)_k$.
5: $\quad k \leftarrow k+1$.
6: $\quad$ If k is too large, terminate.
7: Return z_k.

Lemma 22.2. Let $E = E(I, 0)$ and $H = \{x;\, x_1 \leq 0\}$. Let

$$Z = \begin{pmatrix} \frac{1}{p^2} & 0 & \dots & 0 \\ 0 & \frac{1}{q^2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{q^2} \end{pmatrix}, \qquad z = \begin{pmatrix} t \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad t = -\frac{1}{n+1},\ p^2 = (1+t)^2,\ q^2 = \frac{(1+t)^2}{1+2t}.$$

Then for $E' = E(Z, z)$, it holds that $E' \supseteq E \cap H$, and

$$\frac{\operatorname{vol}(E)'}{\operatorname{vol}(E)} \leq \exp\left(-\frac{1}{2n+2}\right) < 1.$$

Lemma 22.3. Let $E = E(A, a)$, $H = \left\{x;\, c^\top x \leq 0\right\}$. Let

$$f = a - \frac{1}{n+1}\frac{Ac}{\sqrt{c^\top Ac}}, \qquad C = \frac{n^2}{n^2-1}\left(A - \frac{2}{n+1} \cdot \frac{Ac(Ac)^\top}{c^\top Ac}\right).$$

Then for $E' = E(C, f)$, it holds that $E' \supseteq E \cap H$, and

$$\frac{\operatorname{vol}(E)'}{\operatorname{vol}(E)} \leq \exp\left(-\frac{1}{2n+2}\right) < 1.$$

Lemma 22.4. If P has full dimension, then

$$\operatorname{vol}(P) \geq 2^{-(n+1)\langle C\rangle + n^3}.$$

Důsledek 22.5. If P has full dimension and is non-empty, then the Ellipsoid method finds an $x \in P$ in

$$k = 2\,(n+1)\left(2\,(n+1)\,\langle C\rangle + n\,\langle d\rangle - n^3\right).$$

Tvrzení 22.6. If P is unbounded, we can simply add $2n$ constraints of type

$$-2^{\langle C|d\rangle} \leq x \leq 2^{\langle C|d\rangle}.$$

Lemma 22.7 (Farkas). Exactly one of the following has a solution:

(1) $A^\top y = 0, b^\top y < 0, y \geq 0$,

(2) $Ax \leq b$.

Lemma 22.8. The system $Ax \leq b$ has a solution, iff $Ax \leq b + \varepsilon$ for

$$\varepsilon = \frac{1}{2m \cdot 2^{\langle A|b\rangle - n^2}}.$$

Věta 22.9. Linear programming is poly solvable (except for the square-root).

Okruh V

Grafové algoritmy

Otázka 23

Nejkratší cesty, tranzitivní uzávěr

Značení 23.1.

(1) G is a digraph,

(2) $\ell : E \to \mathbb{R}$,

(3) $d : V^2 \to \mathbb{R}$, such that

$$d(u, v) := \min\{\ell(W)\,;\, W \in \mathcal{W}(uv)\}.$$

Tvrzení 23.2 (Negative cycles). If we define d using walks, then for G with negative cycles, d can be undefined (approaching $-\infty$). On the other hand, if we define d using paths, then on G with negative cycles, the triangle inequality does not hold.

Tvrzení 23.3. On G without negative cycles, there is a shortest walk which is also a path.

Tvrzení 23.4. Suppose G doesn't have negative cycles. Then

$$(\forall u, v)\ \ d(u, v) \leq d(u, w) + d(w, v)\,.$$

Pozorování 23.5. A prefix of a shortest path is a shortest path.

Definice 23.6 (Shortest path tree). Then a *shortest path tree* T_u is an oriented tree rooted in u, such that for each $v \in V$, the uv-path in T_u is a shortest uv-path in G.

Lemma 23.7. If G doesn't have negative cycles, then T_u exists $(\forall u)$.

Algoritmus 23.8 (Relaxation scheme).
Vstup: G, source $z \in V$.
Výstup: h, the distance of all vertices from s.

```
1:  (∀v ∈ V)(s(v) ← UNSEEN).
2:  (∀v ∈ V)(h(v) ← ∞)
3:  s(z) ← OPEN
4:  h(z) ← 0
5:  While (∃v)(s(v) = OPEN):
6:      Choose v, such that s(v) = OPEN.        ▷ Freedom in choice of v, affects time complexity.
7:      For uv ∈ E:                             ▷ Relaxation of vertex v.
8:          If h(u) > h(v) + d(uv):
9:              h(u) ← h(v) + d(uv)
10:             s(u) ← OPEN
```

Věta 23.9 (Properties of Relaxation scheme). Let G be a graph, $s \in V$, and h, s from the Relaxation scheme. Then $(\forall v \in V)$

(1) $h(v) \neq \infty \to (\exists W \in \mathcal{W}(sv))(\ell(W) = h(v))$,

(2) $(h(v) \neq \infty \wedge \textsc{Nnc}(G)) \to (\exists P \in \mathcal{P}(sv))(\ell(P) = h(v))$,

(3) $\textsc{Nnc}(G) \to$ Relaxation scheme terminates,

(4) if Relaxation scheme terminated, then v is reachable from $s \leftrightarrow s(v) = \textsc{closed} \leftrightarrow h(v) < \infty$,

(5) if Relaxation scheme terminated, then $d(s, v) = h(v)$.

Algoritmus 23.10 (Bellman–Ford–Moore). The Bellman–Ford–Moore algorithm is an implementation of Relaxation scheme with *queue*.

Definice 23.11 (Epoch). An *epoch* of Bellman–Ford–Moore is E_i, such that

(1) E_0 ends by closing s,

(2) E_{i+1} ends by closing vertices opened in E_i.

Lemma 23.12. Each E_i runs in $\mathcal{O}(m)$.

Lemma 23.13. If $\textsc{Nnc}(G)$, then at the end of E_i,

$$(\forall v \in V)(\forall W \in \mathcal{W}(sv))(|W| \leq i \to h(v) \leq \ell(W))\,.$$

Věta 23.14. BFM runs in $\mathcal{O}(mn)$.

Algoritmus 23.15 (Dijkstra). The Dijkstra algorithm is an implementation of Relaxation scheme with a *heap-like* structure.

Věta 23.16. If $\ell \geq 0$, then Dijkstra closes vertices in the order of distance from s, each at most once.

Věta 23.17 (Dijkstra Complexity). Dijkstra uses

$$\mathcal{O}(n) \times \textsc{Insert}, \quad \mathcal{O}(m) \times \textsc{Decrease}, \quad \mathcal{O}(n) \times \textsc{ExtractMin}.$$

Důsledek 23.18.

- With a binary heap, Dijkstra runs in $\mathcal{O}(m \log n)$.
- With a Fibonacci heap, Dijkstra runs in $\mathcal{O}(m + n \log n)$.
- With a $\frac{m}{n}$-ary heap, Dijkstra runs in $\mathcal{O}\left(m \frac{\log n}{\log \frac{m}{n}}\right)$.

Věta 23.19 (Array of buckets). For $\operatorname{rng} \ell \subseteq [L]$, using an *array of buckets*, Dijkstra runs in $\mathcal{O}(m + Ln)$.

Poznámka 23.20. All open buckets will be at most L far, so we can index mod L, reaching space complexity $\mathcal{O}(L + n)$.

Definice 23.21 (Buckets with trees). The *buckets with trees* DS holds

(1) n binary trees of size L (of which all but ≤ 2 are empty),

(2) in the i-th tree at position k the open vertices with $h(v) = iL + k$.

Důsledek 23.22. All operations of Dijkstra are $\mathcal{O}(\log L)$, reaching $\mathcal{O}(m \log L)$ time complexity.

Věta 23.23 (Dinitz trick). For non-integer values and $\delta > 0$, such that

$$(\forall e)(\ell(e) \geq \delta)\,,$$

we can close all v, such that

$$h(v) < \min_{u \text{ open}} h(u) + \delta.$$

However, this isn't monotonous (it destroyes the „closes vertices in order of distance" property of Dijkstra).

Důsledek 23.24. In the heap, we can replace

$$h(v) \to \left\lfloor \frac{h(v)}{\delta} \right\rfloor,$$

and this is monotonous.

Tvrzení 23.25. Adding Δ to edges doesn't preserve the shortest paths.

Definice 23.26 (potential). A *potential* is a function $\varphi : V \to \mathbb{R}$.

Definice 23.27 (reduced edge length). A *reduced edge length with respect to* φ is

$$\ell_\varphi(uv) := \ell(uv) + \varphi(u) - \varphi(v)\,.$$

Důsledek 23.28. For φ, the shortest paths are preserved, as

$$(\forall P \in \mathcal{P}(uv))(\ell_\varphi(P) = \ell(P) + \varphi(u) - \varphi(v))\,.$$

Definice 23.29 (Feasible potential). A potential φ is *feasible* $\equiv$

$$(\forall uv \in E)(\ell_\varphi(uv) \geq 0)\,.$$

Algoritmus 23.30 (Finding a feasible potential).
Vstup: Graph G with no negative cycles.
Výstup: A feasible potential φ.

1: Add vertex s, connected to all other vertices with $\ell = 0$.
2: Run BFM from s.
3: $(\forall v \in V)(\varphi(v) \leftarrow d(s, v))$

Důsledek 23.31. The above algorithm is correct, hence a feasible potential always exists for a graph with no negative cycles.

23.1 Point-to-point shortest paths

Poznámka 23.32. PtPSP is asymptotically same as SSSP.

Tvrzení 23.33. When looking for PtPSP, we can stop after closing the target vertex.

Tvrzení 23.34 (Bi-directional PtP). Suppose $\ell > 0$. We can run the algorithm from the source and from the target at the same time. Afer a vertex is closed by both, the path consists of some vertices opened by either (doesn't need to pass through the connecting vertex!).

Algoritmus 23.35 (A^*). A^* is an augmentation of Dijkstra, where we use a heuristic function $\psi : V \to \mathbb{R}$.

Věta 23.36. A^* with $\ell, \psi \cong$ Dijkstra with $\ell_{-\psi}$.

Důsledek 23.37. A^* is correct in terms of a case of Relaxation scheme.

Důsledek 23.38. A^* has properties similar to Dijkstra, iff ψ is a feasible potential.

23.2 All-pairs shortest paths

Značení 23.39.

(1) L is the *length matrix*,

(2) D is the *distance matrix*,

(3) A is the *adjacency matrix*,

(4) R is the *reachability matrix*.

Poznámka 23.40. BFS runs in $\mathcal{O}(mn)$, Dijkstra in $\mathcal{O}(mn + n^2 \log n)$. This is good if $m \propto n$, but not if $m \propto n^2$.

Algoritmus 23.41 (Floyd-Warshall).
Vstup: G.
Výstup: D.

```
1: D^0 ← L
2: For k ∈ [n]:
3:     For i, j ∈ [n]:
4:         D^k_{i,j} ← min(D^{k-1}_{i,j}, D^{k-1}_{i,k} + D^{k-1}_{k,j})
```

Věta 23.42. Floyd-Warshall runs in $\mathcal{O}(n^3)$. It uses auxiliary space $\mathcal{O}(1)$.

Definice 23.43 (Bunch). A *bunch* is a set of walks with the same source and target. The *type* of a bunch is (s, t) for the source and target.

Definice 23.44 (Walk algebra). A *walk algebra* is $(B, e_1, \ldots, e_m, \varepsilon_1, \ldots, \varepsilon_n, \cup, \cdot, ^*)$, where

(1) B is the set of all *bunches*,

(2) $(\forall i \in [m])$ $e_i \in E$ is the bunch containing only the edge e_i,

(3) $(\forall i \in [n])$ $\varepsilon_i \in V$ is the bunch containing only the walk from v_i to v_i of length 0,

(4) $\cup$ is the *union* of bunches,

(5) $\cdot$ is the *concatenation* of bunches,

(6) * is the *star* of a bunch, defined as

$$X^* := \varepsilon \cup X \cup (X \cdot X) \cup \ldots$$

We further define

(1) $A^1 := A$,

(2) $(\forall k > 1)$ $A^k := A \cdot A^{k-1}$,

(3) $A^0 := \varepsilon$.

Tvrzení 23.45. Floyd-Warshall uses only operations on walk algebras.

Tvrzení 23.46. Walk algebra can be implemented using circuits in $\mathcal{O}(n^3)$ space.

Poznámka 23.47. We say that matrix multiplication can be done in $\mathcal{O}(n^\omega)$, for some $\omega \in (2, 3)$.

Tvrzení 23.48.

$$A_{i,j} = [ij \in E]\,.$$

Tvrzení 23.49.

$$A^k_{i,j} = |\{w \in \mathcal{W}(ij)\,;\, |w| = k\}|\,.$$

Tvrzení 23.50.

$$(A + I_n)^k_{i,j} = |\{w \in \mathcal{W}(ij)\,;\, |w| \leq k\}|\,.$$

Tvrzení 23.51 (Reachability).

$$R_{i,j} = \left[(A + I_n)^n_{i,j} \neq 0\right].$$

Důsledek 23.52. R can be computed in $\mathcal{O}(n^\omega \log n)$.

Poznámka 23.53. The numbers can get exponentially large, so we should after each squaring replace the non-zero numbers with ones.

Definice 23.54 ($(\oplus,\otimes)$-matrix product). Then the $(\oplus,\otimes)$-*matrix product* of A and B is

$$C := \left(\bigoplus_{k=1}^{n} A_{i,k} \otimes B_{k,j}\right)_{i,j}.$$

Důsledek 23.55. $(\cup, \cdot)$ on walks is walk algebra.

Důsledek 23.56. $(\vee, \wedge)$ on A is reachability.

Důsledek 23.57. $(\min, +)$ on L is Floyd-Warshall.

Poznámka 23.58. The best algorithms don't work on $(\min, +)$, because there is no inverse of min. The best time complexity is $\mathcal{O}\left(\frac{n^3}{\log n}\right)$.

Tyto přístupy se dají použít i na různé příbuzné problémy, jako je „nejspolehlivější cesta“, nebo cesta s nejmenším omezením na výšku, atd. To samé platí i o následujícím algoritmu.

Značení 23.59 (Divide and Conquer). Suppose $n = 2^i$ for some i. We will denote

(1) $P = (A_{i,j})^{\frac{n}{2},\frac{n}{2}}_{i,j=1,1}$.

(2) $Q = (A_{i,j})^{\frac{n}{2},n}_{i,j=1,\frac{n}{2}+1}$.

(3) $R = (A_{i,j})^{n,\frac{n}{2}}_{i,j=\frac{n}{2}+1,1}$.

(4) $S = (A_{i,j})^{n,n}_{i,j=\frac{n}{2}+1,\frac{n}{2}+1}$.

Důsledek 23.60.

$$A = \begin{pmatrix} P & Q \\ R & S \end{pmatrix}.$$

Algoritmus 23.61 (Divide and Conquer).
Vstup: A.
Výstup: A^*.
1: $I \leftarrow (P \vee QS^*R)^*$
2: $J \leftarrow IQS^*$
3: $K \leftarrow S^*RI$
4: $L \leftarrow S^* \vee S^*RIQS^*$
5: $A^* \leftarrow \left(\begin{smallmatrix} I & J \\ K & L \end{smallmatrix}\right)$

Věta 23.62. $A^* = R$, and can be computed in $\mathcal{O}(n^\omega)$.

Definice 23.63 (Graph square). $G^2 = (V', E')$, such that

(1) $V' = V$,

(2) $E' = \{uv;\ (\exists W \in \mathcal{W}(uv)_G)(|W| \leq 2)\}$.

Důsledek 23.64. G^2 can be computed in $\mathcal{O}(n^\omega)$ using one $(\vee, \wedge)$.

Tvrzení 23.65. Let G be undirected, $\operatorname{rng} \ell = \{0, 1\}$. Let d' be the distances from s in G^2. Then

$$d'(v) = \left\lceil \frac{d(v)}{2} \right\rceil, \qquad d(v) = \begin{cases} 2d'(v) - 1 & \operatorname{avg}_{vw \in E} d'(w) < d'(v) \\ 2d'(v) & \operatorname{avg}_{vw \in E} d'(w) \geq d'(v) \end{cases}$$

Poznámka 23.66.

$$\operatorname*{avg}_{vw \in E} d'(w) = \frac{(D'A)_{s,v}}{\deg v}.$$

Algoritmus 23.67 (Seidel).
Vstup: G, undirected, with lengths of edges equal to one, one component.
Výstup: A.

1: If $G = K_n$, return $(1)_{i,j} - I_n$.
2: $G' \leftarrow G^2$
3: $D' \leftarrow \textsc{Seidel}(G')$
4: $F \leftarrow \frac{D'A}{\deg v}$
5:

$$D_{i,j} \leftarrow \begin{cases} 2D'_{i,j} - 1 & F_{i,j} < D'_{i,j} \\ 2D'_{i,j} & F_{i,j} \geq D'_{i,j} \end{cases}$$

Věta 23.68. Seidel runs in $\mathcal{O}(n^\omega \log n)$.

Dohromady tedy umíme následující APSP:

(1) Reachability v $\mathcal{O}(n^\omega)$ pomocí DaC.

(2) Unit-length neorientované vzdálenosti v $\mathcal{O}(n^\omega \log n)$ pomocí Siedelova algoritmu.

(3) Obecné vzdálenosti v $\mathcal{O}\big(n^2 \log n + mn\big)$ pomocí Dijkstry plus Bellman-Forda (pokud NNC).

Otázka 24

Toky v sítích

Definice 24.1 (Network). A *network* is $\langle G, s, t, c\rangle$, where

(1) $G = \langle V, E\rangle$ is a *digraph*, WLOG symmetric,

(2) $s \neq t \in V$ are *source* and *target* vertices,

(3) $c : E \to \mathbb{R}_0^+$ is a *capacity function*.

Definice 24.2 (flow). A function $f : E \to \mathbb{R}_0^+$ is a *flow* $\equiv$

(1) $(\forall e)\ 0 \leq f(e) \leq c(e)$,

(2) $(\forall v \neq s, t)\ f(\delta^+(v)) - f(\delta^-(v)) = 0$,

Značení 24.3. Let f be a flow.

- $f^+(v) := f(\delta^+(v))$,
- $f^-(v) := f(\delta^-(v))$,
- $f^\Delta(v) := f^+(v) - f^-(v)$,
- $|f| := f^\Delta(t)$.

Lemma 24.4.

$$|f| = -f^\Delta(s)\,.$$

Definice 24.5 (Residual capacity).

$$r(uv) := (c(uv) - f(uv)) + f(vu)\,.$$

Algoritmus 24.6 (Ford-Fulkerson).
Vstup: a network $\langle G, s, t, c\rangle$.
Výstup: a flow f.

```
1: f ← 0
2: While (∃P ∈ P(st))(∀e ∈ P)(r(e) > 0):
3:     ε ← min_{e∈P} r(e)
4:     For uv ∈ P:
5:         δ ← min(c(uv) − f(uv), ε)
6:         f(uv) ← f(uv) + δ
7:         f(vu) ← ε − δ
```

Věta 24.7. If $\operatorname{rng} c \subseteq \mathbb{Q}$, then Ford-Fulkerson terminates.

Značení 24.8. For $A, B \subseteq V$,

$$E(A,B) := E \cap A \times B.$$

Definice 24.9 (Elemental cut). $C \subseteq E$ is an *elemental cut* $\equiv$

$$(\exists A \subseteq V)\big(s \in A \wedge t \notin A \wedge C = E\big(A, \overline{A}\big)\big)\,.$$

Značení 24.10. For $A, B \subseteq V$, $g : E \to X$,

$$g(A,B) := g(E(A,B))\,.$$

Věta 24.11. Let $C = E\big(A, \overline{A}\big)$ be an elemental cut. Then

$$f^{\Delta}(C) = |f|\,.$$

Důsledek 24.12.

$$(\forall A \subseteq V, s \in A)\ \ |f| \leq c\big(A, \overline{A}\big)\,.$$

Věta 24.13 (Correctness of FF). For $\operatorname{rng} c \subseteq \mathbb{Q}$, Ford-Fulkerson computes a maximum flow.

Věta 24.14. For $\operatorname{rng} c \subseteq \mathbb{N}$, Ford-Fulkerson computes a maximum flow f, such that $\operatorname{rng} f \subseteq \mathbb{N}$.

Věta 24.15. If $c \leq L$, then Ford-Fulkerson terminates in $\mathcal{O}(Lmn)$.

Poznámka 24.16. For $\operatorname{rng} c \subseteq \mathbb{R}$, Ford-Fulkerson needn't terminate. It does terminate if you choose the minimum augmenting path, and it runs in $\mathcal{O}\big(m^2 n\big)$.

Definice 24.17 (Pure flow). For a flow f, its *pure flow* is

$$f^*(uv) := f(uv) - f(vu)\,.$$

Důsledek 24.18.

$$f^{\Delta}(v) := \sum_{e \in \delta^+(v)} f^*(e)$$

Lemma 24.19. For any function f^* satisfying:

(1) $f^*(uv) = -f^*(vu)$,

(2) $f^*(e) \leq c(e)$,

(3) $(\forall v \neq s, t)\ \ f^{\Delta}(v) = 0$,

there exists a corresponding flow f, for which f^* is a pure flow.

Definice 24.20 (Residual network). A *residual network* of network $N = \langle G, s, t, c\rangle$ and flow f is

$$\mathcal{R}(N,f) := \langle G, s, t, r\rangle\,.$$

Lemma 24.21. For a flow f in N, and g in $\mathcal{R}(N,f)$, there exists a flow h, such that

$$|h| = |f| + |g|\,,$$

further, h is constructable in $\mathcal{O}(m)$.

Definice 24.22 (Blocking flow). A flow f is *blocking* $\equiv$

$$(\forall P \in \mathcal{P}(st))(\exists e \in P)(f(e) = c(e))\,.$$

ALGORITMUS 24.23 (DINIC).
Vstup: N.
Výstup: f, maximal flow.

$f \leftarrow 0$
Loop:
  Construct a *Layered network*:
    $R \leftarrow \mathcal{R}(N, f)$
    Remove $e : r(e) = 0$.
    Keep only those vertices and edges included on some shortest st paths. ▷ Via Queue
  If $\mathcal{P}(st) = \emptyset$, stop.
  Get a blocking flow g in the layered network:
    $g \leftarrow 0$
    While $\exists P \in \mathcal{P}(st)$:
      $\varepsilon \leftarrow \min_{e \in P}(r(e) - g(e))$
      Increase $g(P)$ by ε
      Remove $\{e \in P;\, g(P) = r(P)\}$
      Cleanup dead ends.
  $f \leftarrow f + g$ ▷ We need to properly handle the translation between R and N.

LEMMA 24.24. Between phases, the number of layers increases.

DŮSLEDEK 24.25. DINIC finishes in $\mathcal{O}\big(n^2 m\big)$.

POZOROVÁNÍ 24.26. In each phase, the number of remaining phases is upper bounded by the size of the flow

$$|f_*| - |f| = |g| \leq r(\text{some cut})\,,$$

where g is some flow in $\mathcal{R}(N, f)$.

VĚTA 24.27. If $c \in \{0, 1\}$, then DINIC runs in $\mathcal{O}(m\sqrt{m})$.

VĚTA 24.28. If $c \in \{0, 1\}$ and $\min\big(\deg^i, \deg^o\big) \leq 1$, then DINIC runs in $\mathcal{O}(m\sqrt{n})$.

VĚTA 24.29. If $c \in \{0, 1\}$ and there are no parallel lines (i.e. G is not a multigraph), then DINIC runs in $\mathcal{O}\Big(mn^{\frac{2}{3}}\Big)$.

VĚTA 24.30. If $\operatorname{rng} c \subseteq \mathbb{Z}$, then DINIC runs in $\mathcal{O}(\Delta f \cdot n + mn)$.

VĚTA 24.31. If $c \leq C \in \mathbb{R}$, then DINIC runs in $\mathcal{O}\big(Cn^2 + mn\big)$.

DEFINICE 24.32 (SCALED CAPACITIES). For a network N, where $\operatorname{rng} c \subseteq \{0, \ldots, C\}$, we define

(1) $k := \lfloor \log_2 C \rfloor$,
(2) $c_i(e) := \lfloor \frac{c(e)}{2^{k-i}} \rfloor$ (i most important bits of c),
(3) $N_i := \langle G, s, t, c_i \rangle$.

TVRZENÍ 24.33. For a flow f_i in N_i, $2f_i$ is a valid flow in N_{i+1}.

TVRZENÍ 24.34.

$$|f_{i+1}| \leq 2\,|f_i| + m.$$

VĚTA 24.35. When computing maximal f_{i+1} from maximal f_i, it takes $\mathcal{O}(mn)$ by Theorem 24.30 and Proposition 24.34, so computing the whole flow incrementally takes $\mathcal{O}(mn \log C)$.

Otázka 25

Řezy

Definice 25.1 (Edge-connectivity). Let G be an undirected graph.

(1) G is *k-edge-connected* $\equiv (\forall F \subseteq E, |F| < k)$ $G - F$ is connected.

(2) $F \subseteq E$ is a *cut* $\equiv G - F$ is disconnected.

(3) The *edge-connectivity* of G is $\kappa(G) :=$ the size of the minimum cut.

(4) F is an *st-cut* $\equiv G - F$ has no st-path.

(5) (my notation) $\kappa(G, s, t) :=$ the size of the smallest st-cut.

Tvrzení 25.2. The minimum st-cut is an elementary cut.

Věta 25.3 (Menger). $\kappa(G, s, t)$ equals the number of edge-disjoint st-paths, and the conversion from the maximal st-flow to the paths can be done in $\mathcal{O}(m)$.

Tvrzení 25.4. The minimum st-cut can be found using flows in $\mathcal{O}\left(n^{\frac{2}{3}}m\right)$.

Věta 25.5 (Global Menger). $\kappa(G)$ equals the maximum k, such that for each s, t, there is a system of k edge-disjoint st-paths.

Věta 25.6. The minimum cut in an undirected graph can be found by

(1) trying all choices of s, t, in $\mathcal{O}(n^{\frac{8}{3}}m)$,

(2) fixing s, trying all t, in $\mathcal{O}(n^{\frac{5}{3}}m)$.

Definice 25.7 (Vertex-connectivity). Let G be an undirected graph.

(1) $W \subset V$ is a *cut*, or *separator* $\equiv G - W$ is disconnected.

(2) The *vertex-connectivity* of G is $\lambda(G) :=$ the size of the minimum separator, or $n - 1$ for K_n.

(3) G is *k-vertex-connected* $\equiv \lambda(G) \geq k$.

(4) $W \subseteq (V \setminus \{s, t\})$ is an *st-separator* $\equiv G - W$ has no st-path.

(5) (my notation) $\lambda(G, s, t) :=$ the size of the smallest st-separator.

Definice 25.8 (Internally vertex-disjoint paths). A set of paths is *internally vertex-disjoint* $\equiv$ it is vertex-disjoint except for the endpoints.

Tvrzení 25.9. $\lambda(G, s, t)$ is the number of internally vertex-disjoint st-paths, and the conversion from the maximal st-flow to the paths can be done in $\mathcal{O}(m)$.

Tvrzení 25.10. The minimum st-separator can be found in $\mathcal{O}\left(n^{\frac{1}{2}}m\right)$.

Věta 25.11.

(1) The globally-minimum separator can be found by gridsearch in $\mathcal{O}\left(n^{\frac{5}{2}}m\right)$.

(2) Actually, it can be found in $\mathcal{O}\left(\lambda(G)\,n^{\frac{3}{2}}m\right)$, as we can stop after trying $\lambda(G)$ sources.

Algoritmus 25.12 (Contraction).
Vstup: G with n vertices, ℓ.
Výstup: G' with ℓ vertices.

```
1: G' ← G
2: While |V(G')| > ℓ:
3:     e ∈_R E(G')
4:     G' ← G' / e
```

Tvrzení 25.13. If C is a cut in $G \,/\, e$, then C is a cut in G.

Tvrzení 25.14. If C is a cut in G and $e \notin C$, then C is a cut in $G \,/\, e$.

Důsledek 25.15. If e is not in a minimum cut, then $\kappa(G) = \kappa(G \,/\, e)$.

Věta 25.16.

$$\Pr[\kappa(G) = \kappa(G')] \geq \frac{\ell\,(\ell-1)}{n\,(n-1)}.$$

Věta 25.17. Contraction can be done in $\mathcal{O}(n^2)$ using the adjacency matrix.

Tvrzení 25.18. $\ell = 2 \to \Pr[\text{found cut is minimum}] \approx \frac{c}{n^2}$. This can be increased by iterating k-times.

Důsledek 25.19. $\Pr[\text{wrong output}] \leq e^{-\frac{ck}{n^2}}$.

- $k \approx n^2 \to$ constant,
- $k \approx n^3 \to e^{-c'n}$,
- $k \approx n^2 \log n \to \frac{1}{\operatorname{poly} n}$, also called *with high probability*.

Důsledek 25.20. A minimum cut can be found in $\mathcal{O}(n^4 \log n)$ with high probability.

Tvrzení 25.21. For $\ell = \lceil \frac{n}{\sqrt{2}} + 1 \rceil$, $\Pr[\text{minimum cut survives contraction}] \geq \frac{1}{2}$.

Algoritmus 25.22 (Karger-Stein).
Vstup: G.
Výstup: $\kappa(G)$.

```
1: If n ≤ 7: find the minimum cut by brute force.
2: ℓ ← ⌈n/√2 + 1⌉
3: C_1 ← Karger-Stein(Contraction(G, ℓ))
4: C_2 ← Karger-Stein(Contraction(G, ℓ))
5: Return the minimum of C_1 and C_2.
```

Věta 25.23. Iterated Karger-Stein runs in $\mathcal{O}(n^2 k \log n)$, and

$$\Pr[\text{fail}] \approx e^{-\frac{ck}{\log n}}.$$

Důsledek 25.24. The minimum cut can be found with high probability in $\mathcal{O}(n^2 \log^3 n)$.

Otázka 26

Párování

Definice 26.1 (párování). Nechť G je graf. Pak $M \subseteq E$ je *párování* $\equiv$

$$(\forall v \in V) \quad |\{e \in M;\, v \in e\}| \leq 1.$$

Definice 26.2 (největší párování). Nechť $M \subseteq E$ je párování v G. M je

(1) *maximální* (k inkluzi) $\equiv$ po přidání libovolné další hrany M přestane být párování,

(2) *největší* $\equiv$ ze všech párování má maximální velikost.

Definice 26.3 (volný vrchol). Nechť $M \subseteq E$ je párování. Pak vrchol v je *volný* $\equiv$

$$(\forall e \in M)(v \notin e)\,,$$

jinak je *vázaný*.

Definice 26.4 (VSC). Nechť P je cesta v G. Nechť $M \subseteq E$ je párování. Pak P je vůči M *střídavá* $\equiv$ střídají se na ní hrany z M a hrany z $E \setminus M$. Navíc, pokud konce nejsou v M, říkáme, že P je *volná střídavá*.

Věta 26.5. Párování $M \subseteq E$ je největší, právě tehdy když G neobsahuje žádnou VSC vůči M.

Definice 26.6 (kytka). Nechť $M \subseteq E$ je párování. Pak *kytka* je podgraf G, který se skládá z

(1) *květu*, což je lichý střídavý cyklus,

(2) *stonku*, což je sudá střídavá cesta, napojená na květ přes ne-volnou hranu, a jeho konec je volný vrchol.

Definice 26.7 (kontrakce). Nechť $M \subseteq E$ je párování. Nechť C je květ nějaké kytky z G vůči M. Pak $G \,/\, C$ je graf, ve kterém je květ zkontrahován na jeden vrchol, a multihrany odstraněny.

Věta 26.8. Graf G obsahuje VSC vůči párování M, právě tehdy když pro každý květ C obsahuje graf $G \,/\, C$ VSC vůči $M \,/\, C$.

ALGORITMUS 26.9 (EDMONDSŮV KYTIČKOVÝ ALGORITMUS).
Vstup: Graf G, párování M.
Výstup: Lepší párování M', pokud takové existuje, jinak M.

```
S ← volné vrcholy vůči M.                    ▷ Volné vrcholy budou v nulté hladině.
Z ← ∅                                        ▷ Zpracované vrcholy.
Pro v ∈ S:
    Pokud ∃w : vw ∈ M:                       ▷ Nezajímají nás volné hrany z liché do sudé úrovně.
        S ← S \ {v}
        Z ← Z ∪ {v}
        v ← w
    Pokud existuje vu ∈ E, kde u je v sudé hladině jiného stromu:
        Provedeme přepárování střídavé cesty.
        Vrátíme nové párování.
    Pokud existuje vu ∈ E, kde u je v sudé hladině stejného stromu:
        Provedeme kontrakci nalezeného květu.
        M' ← EDMONDSŮV KYTIČKOVÝ ALGORITMUS(G / C, M / C)
        Převedeme M' na párování v G a vrátíme ho.
    Pro u ∈ V : uv ∈ E ∧ u ∉ Z ∪ S:
        S ← S ∪ {u}
    S ← S \ {v}
    Z ← Z ∪ {v}
Vrátíme původní párování M.
```

TVRZENÍ 26.10. EDMONDSŮV KYTIČKOVÝ ALGORITMUS vždy pracuje jen s vrcholy na sudých úrovních.

VĚTA 26.11. EDMONDSŮV KYTIČKOVÝ ALGORITMUS nalezne lepší párování, pokud existuje. Běží v $\mathcal{O}(n(m+n))$.

26.1 Perfektní párování

DEFINICE 26.12 (PERFEKTNÍ PÁROVÁNÍ). Párování M je *perfektní* $\equiv |M| = \frac{n}{2}$.

DEFINICE 26.13 (ODD). Nechť G je graf. Pak $\mathrm{odd}(G)$ je počet lichých komponent v G.

LEMMA 26.14. Po přidání hrany do G se $\mathrm{odd}(G)$ nezvětší.

VĚTA 26.15 (TUTTE). Graf G má perfektní párování, právě když platí Tutteova podmínka, tj.

$$(\forall S \subseteq V) \qquad \mathrm{odd}(G - S) \leq |S|\,.$$

Důkaz. Pokud G má perfektní párování, podmínka platí. Druhou implikaci dokážeme indukcí přes počet ne-hran.

Pro K_n implikuje Tutteova podmínka že počet vrcholů je sudý, takže perfektní párování existuje. Jinak označíme $S = \{v \in V; \delta(v) = V \setminus v\}$. Pokud jsou všechny komponenty $G - S$ kliky, je tvrzení jednoduché. Pokud ne, najdeme třešničku xuy v $G - S$, a vrchol v nespojený s u (takový existuje, protože $u \notin S$).

Definujeme $G_1 = G + xy$ a $G_2 = G + uv$. Tím jsme nerozbili Tutteovu podmínku, vezmeme párování z IP, a upravíme ho na párování G. ■

DEFINICE 26.16 (SOUVISLOST). Graf G je *hranově k-souvislý* $\equiv$ každý hranový řez má velikost alespoň k.

DEFINICE 26.17 (REGULÁRNÍ GRAF). Graf G je *k-regulární* $\equiv$

$$(\forall v \in V)(\deg v = k)\,.$$

TVRZENÍ 26.18. Každý hranově 2-souvislý 3-regulární graf má perfektní párování.

Věta 26.19 (Hall). Pro G bipartitní platí, že má párování velikosti A, právě když platí Hallova podmínka:

$$(\forall S \subseteq A) \qquad |S| \leq |\delta(S)|\,.$$

Důkaz. Hallova podmínka je určitě nutná pro existenci párování vel. A.

Jinak, přidáme z a s a najdeme maximální tok. Jeho velikost je rovna velikosti největšího párování. Nechť $R \subseteq E'$ je odpovídající min. řez. Označme A', B' vrcholy z A, B incidentní R. Označme $J = A \setminus A'$.

Z tohoto J vedou hrany jen do B, tedy $\delta(J) \subseteq B$. Proto

$$|R| = |A'| + |B'| \geq |A'| + |\delta(J)| \geq |A|\,,$$

kde poslední nerovnost používá Hallovu podmínku. ■

Ekvivalentně lze Hallovu větu říct pro systém různých reprezentantů.

Důsledek 26.20. Nechť G je bipartitní s partitami A, B, $E \neq 0$ a platí

$$(\forall x \in A)(\forall y \in B) \qquad \deg(x) \geq \deg(y)\,.$$

Pak G má párování velikosti $|A|$.

26.2 Perfektní párování minimální ceny

Nyní rozšíříme ideu Kytičkového algoritmu tak, abychom našli perfektní párování minimální ceny (pokud takové existuje).

Problém 26.21 (Perfektní párování minimální ceny).
Vstup: Graf G s cenami $c : E \to \mathbb{R}_0^+$.
Výstup: Perfektní párování M, pro které $c(M)$ je minimální.

Použijeme lineární programování. Klíčové bude přidat podmínky pro *liché řezy*, viz následující pozorování.

Pozorování 26.22. Nechť M je perfektní a $S \subseteq V$ je lichá. Pak

$$|M \cap \delta(S)| \geq 1.$$

Můžeme tedy definovat následující lineární program pro PPMC.

$$\begin{aligned} &\min c^\top x \\ &x(\delta(v)) = 1, && (\forall v \in V)\,, \\ &x(\delta(S)) \geq 1, && (\forall S \subseteq V, |S| \not\equiv 2)\,, \\ &x \geq 0. \end{aligned}$$

Jeho duální program je

$$\begin{aligned} &\max \sum_{v \in V} y_v + \sum_{S \subseteq V} Y_S \\ &y_u + y_v + \sum_{uv \in \delta(S)} Y_S \leq c_{uv}, && (\forall uv \in E)\,, \\ &y \in \mathbb{R}^V, \\ &Y \geq 0. \end{aligned}$$

Definice 26.23 (Redukovaná cena). Nechť y, Y jsou duální proměnné. Pak *redukovaná cena* je

$$\overline{c}_{uv} = c_{uv} - y_u - y_v - \sum_{uv \in \delta(S)} Y_S.$$

Algoritmus 26.24 (Úprava KA).

(1) Udržujeme si duální řešení y, Y, na začátku samé nuly.

(2) Při kontrakci interpretujeme novou y_z jako Y_C kde C je kontrahovaný cyklus. Toto z nazýváme *pseudovrchol*.

(3) Používáme hrany pouze v $E_= = \{e \in E;\, \overline{c}_e = 0\}$.

(4) Pokud nelze aplikovat žádný z kroků, definujeme ε tak, že pro $v \in T$ na liché hladině změníme $y_v \leftarrow y_v - \varepsilon$ a na sudé $y_v \leftarrow y_v + \varepsilon$, kde

$$\varepsilon = \min \begin{cases} \overline{c}_{uv} & \text{pro } u \text{ na liché hladině a } v \notin T, \\ \overline{c}_{uv}/2 & \text{pro } u, v \text{ na liché hladině}, \\ y_z & \text{pro } z \text{ pseudovrchol na liché hladině.} \end{cases}$$

Pokud je toto $\varepsilon = \infty$, vrátíme „perf. párování neexistuje“.

(5) Při kontrakci změním c_{uv} pro $u \in C$ a $v \notin C$ na $c_{uv} \leftarrow c_{uv} - y_u$.

(6) Pokud $y_z = 0$ a z je pseudovrchol na liché úrovni, dovolíme de-kontrakci z.

Tvrzení 26.25. Toto se zastaví, a pokud je výsledek „perf. párování neexistuje“, je to pravda.

Důkaz. Párování se stále zvětšuje. Když de-kontrahujeme vrchol, je na jiné úrovni než byl když jsme ho kontrahovali, takže počet de/kontrakcí je $\mathcal{O}(n^2)$. ■

Věta 26.26. Tento algoritmus správně hledá perf. párování minimální váhy.

Důkaz. Řešení, které na konci vrátíme, splňuje podmínky komplementarity s duálním řešení které budujeme v průběhu algoritmu. Navíc duální řešení je přípustné po celou dobu běhu algoritmu. Pokud je vrácené řešení perf. párování, je tedy optimální. ■

Otázka 27

Minimální kostry

Značení 27.1. Let G be an *undirected, connected* graph. We will denote

(1) a *spanning tree* $T \subseteq G \equiv V(T) = V(G)$, and T is a tree,

(2) *weights* $w : E \to \mathbb{R}$, WSLOG distinct, i.e. $e \neq f \to w(e) \neq w(f)$,

(3) for a tree T and $u, v \in V$, $T[u, v]$ as the unique path from u to v in T,

(4) for $e = uv \in E \setminus T$, $T[e] := T[u, v]$.

Definice 27.2 (minimum spanning tree). A tree T is a *minimum spanning tree* $\equiv w(T)$ is minimum.

Definice 27.3 (minimum spanning forest). If G is not connected, then a *minimum spanning forest* is a union of minimum spanning trees for all components of G.

Definice 27.4 (T-light edge). $e \in E \setminus T$ is *T-light* $\equiv$

$$(\exists f \in T[e])(w(f) > w(e)) .$$

Věta 27.5. T is a minimum spanning tree $\leftrightarrow$ there are no T-light edges in G.

Důkaz. Implikace „$\to$“ je triviální. Druhá implikace plyne až z věty 27.9. ■

Důsledek 27.6. The weights just need to give a linear ordering on edges, no need for algebra on them.

Algoritmus 27.7 (Jarník).
Vstup: A connected, undirected graph G, with unique weights w.
Výstup: T, a minimum spanning tree in G.

1: $T \leftarrow (\{v_0\}, \emptyset)$
2: While $V(T) \neq V(G)$:
3: $\quad uv \leftarrow \arg\min\{w(uv) ; u \in T \wedge v \notin T \wedge uv \in E\}$
4: $\quad V(T) \leftarrow V(T) \cup \{v\}$
5: $\quad E(T) \leftarrow E(T) \cup \{uv\}$.

Lemma 27.8 (Cut). Let C be an elemental cut in G, $e = \arg\min_{e \in C} w(e)$. Let T be a minimum spanning tree. Then $e \in T$.

Důkaz. Nechť $T \not\ni e$. Přidání e tedy vytvoří cyklus. Tento cyklus musí procházet C ještě v (alespoň) jednom místě. Odstraněním druhé hrany vznikne lehčí T. ■

Věta 27.9. Jarník gives a minimum spanning tree.

Důkaz. V každém kroku Jarníka tvoří už pokryté vrcholy elementární řez, a Jarník vybírá jeho nejlehčí hranu. ■

Důsledek 27.10. For a given G and w, the minimum spanning tree is unique.

Věta 27.11. Jarník runs in $\mathcal{O}(mn)$ trivially, $\mathcal{O}(m \log n)$ with a heap.

Důkaz. V každé iteraci zvětšíme kostru o 1, tedy iterací je maximálně $\mathcal{O}(n)$. Drahý je krok 3, který probírá až $\mathcal{O}(m)$ hran.

Haldu můžeme udržovat na hranách. Pak přidání vrcholu do kostry způsobí update u nejvýše $\deg(v)$ hran. To se dohromady za celý algoritmus sečte na $\mathcal{O}(m \log n)$. ■

Algoritmus 27.12 (Red-Blue).
Vstup: A connected, undirected graph G.
Výstup: A coloring c of edges to Blue and Red.

1: For $e \in E$: $c(e) \leftarrow$ None.
2: Repeat while possible:
3: Choose either
4: e lightest in an elementary cut, which is not Blue, and color it Blue, or
5: e heaviest on a cycle, which is not Red, and color it Red.

Lemma 27.13 (Red). Let T be a minimum spanning tree.

$$c(e) = \text{Red} \rightarrow e \notin T.$$

Důkaz. Nejtěžší hrana na cyklu může být odstraněna z kostry a nahrazena některou jinou hranou. ■

Lemma 27.14 (Blue). Let T be a minimum spanning tree.

$$c(e) = \text{Blue} \rightarrow e \in T.$$

Důkaz. Toto je lemma 27.8. ■

Důsledek 27.15. Edges do not change color between Red and Blue.

Důsledek 27.16. Red-Blue stops.

Lemma 27.17 (Rainbow). When Red-Blue stops, all edges are colored.

Důkaz. Z průběhu Jarníka je vidět že všechny modré hrany najdeme. Pak ale každá ne-modrá hrana je nejtěžší na tom cyklu určeném incidentními modrými hranami. ■

Tvrzení 27.18. All minimum spanning tree algorithms are variants of Red-Blue.

Algoritmus 27.19 (Borůvka).
Vstup: A connected, undirected graph G, with unique weights w.
Výstup: T, the minimum spanning tree in G.

1: $T_0 \leftarrow$ a forest, where each vertex is a tree with no edges.
2: $i \leftarrow 0$
3: While T_i has more than one component:
4: $L \leftarrow$ Each component of T_i chooses the lightest neighbouring edge e.
5: $T_{i+1} \leftarrow T_i \cup L$
6: $i \leftarrow i + 1$

Věta 27.20. Borůvka gives a minimum spanning tree, and runs in $\mathcal{O}(m \log n)$.

Algoritmus 27.21 (Kruskal).
Vstup: A connected, undirected graph G, with unique weights w.
Výstup: T, the minimum spanning tree in G.

1: Sort E by weight:
$$w(e_1) < w(e_2) < \cdots < w(e_m)\,.$$
2: $T \leftarrow (\emptyset, \emptyset)$
3: For $i \in [m]$:
4: If $T \cup \{e_i\}$ has no cycles:
5: $T \leftarrow T \cup \{e_i\}$

Věta 27.22. Kruskal gives a minimum spanning tree, and runs in $\mathcal{O}(mn)$.

Věta 27.23. Using Union-Find, Kruskal runs in $\mathcal{O}(m \log n)$.

Poznámka 27.24. There are implementations of Kruskal (excluding sorting) with $\mathcal{O}(m \log^* n)$, or even $\mathcal{O}(m \cdot \alpha(m, n))$.

Algoritmus 27.25 (Contractive Borůvka).
Vstup: A connected, undirected graph G, with unique weights w.
Výstup: T, the minimum spanning tree in G.

1: $T \leftarrow$ a forest, where each vertex is a tree with no edges.
2: While $|V(G)| \geq 1$:
3: $L \leftarrow$ Each component of T chooses the lightest neighbouring edge e.
4: $T \leftarrow T \cup L$
5: Contract all edges in L.
6: Filter out loops and parallel edges. ▷ By bucket-sort in $\mathcal{O}(m)$.

Pozorování 27.26. The number of edges decreases exponentially as
$$n_i \leq \frac{n}{2^i}.$$

Věta 27.27 (Time complexity of Contractive Borůvka). The Contractive Borůvka algorithm runs in

(1) $\mathcal{O}(m \log n)$ on any graph,
(2) $\mathcal{O}(n^2)$ on any graph,
(3) $\mathcal{O}(n)$ on planar graphs,
(4) $\mathcal{O}(n)$ if G is from a non-trivial minor-closed class.

Definice 27.28 (minor). Let G be a graph Then H is a *minor of* G, $H \prec_m G \equiv G$ turns into H using only vertex/edge deletions and edge contractions.

Definice 27.29 (minor-closed). Let $\mathcal{C}$ be a class of graphs. Then $\mathcal{C}$ is *minor-closed* $\equiv$
$$(\forall G \in \mathcal{C})(\forall H \prec_{\mathrm{m}} G)(H \in \mathcal{C})\,.$$

Příklad 27.30 (Minor-closed classes). The following are minor-closed: the class of all graphs, the empty class, the class of all forests, the class of all planar graphs.

Definice 27.31 (density). Let G be a graph. Then the *density* of G is
$$\varrho(G) := \frac{m}{n}.$$
The density of a graph class $\mathcal{C}$ is
$$\varrho(\mathcal{C}) := \sup_{G \in \mathcal{C}} \varrho(G)\,.$$

Věta 27.32. Let $\mathcal{C}$ be non-trivial, minor-closed. Then $\varrho(\mathcal{C}) \in \mathrm{Fin}$.

Definice 27.33 (Forb). Let $\mathcal{C}$ be class of graphs. Then we define

$$\mathrm{Forb}_{\preccurlyeq_*}(\mathcal{C}) := \{G;\ (\nexists H \in \mathcal{C})(H \prec_{\mathrm{m}} G)\}.$$

Důsledek 27.34.

$$(\forall \mathcal{C})\ \mathrm{Forb}_{\preccurlyeq_*}(\mathcal{C}) \text{ is minor-closed.}$$

Tvrzení 27.35. Let $\mathcal{C}$ be minor-closed. Then $\mathcal{C} = \mathrm{Forb}_{\preccurlyeq_*}(\overline{\mathcal{C}})$.

Fakt 27.36 (Mader).

$$(\forall k)(\exists h)(\forall G)\left(\operatorname*{avg}_{v \in V(G)} \deg v > h \to G \text{ contains a subdivision of } K_k\right).$$

Fakt 27.37. Let G be a graph, such that the average degree of G is $\Omega\left(k\sqrt{\log k}\right)$. Then $K_k \prec_{\mathrm{m}} G$.

Algoritmus 27.38 (Jarník ála Dijkstra). We keep the vertices in a heap when running Jarník, with the weight equal to the actual least-weighted edge connected to the vertex from the tree.

Důsledek 27.39. With Fibonacci heaps, Jarník then runs in $\mathcal{O}(n \log n + m)$.

Důsledek 27.40. If $\varrho(G) \in \Omega(\log n)$, then Jarník runs in $\mathcal{O}(m)$.

Algoritmus 27.41 (Fredman–Tarjan).
Vstup: Graph G.
Výstup: The minimum spanning tree T.

```
m ← |E(G)|
T ← ∅
While E(G) ≠ ∅:
    n ← |V(G)|
    k ← 2^⌈2m/n⌉
    F ← ∅
    While (∃v ∈ V)(v ∉ F):
        Run Jarník(G, v), stop when:
            the size of the heap ≥ k, or
            the heap is empty, or
            the algorithm connected a vertex to its tree which was already in F.
        Put the new found tree into F.
    Contract G based on the edges of F.
    T ← T ∪ F
```

Lemma 27.42. Each phase runs in $\mathcal{O}(m)$.

Důsledek 27.43. If a phase is not final, then every tree is incident (including internal edges) to at least k edges.

Důsledek 27.44. The number of trees is at most $\frac{2m}{k}$.

Důsledek 27.45.

(1) $n_{i+1} \leq \frac{2m_i}{k_i}$,

(2) $k_{i+1} \geq 2^{k_i}$,

(3) $k_1 \geq 2$,

(4) $k_i \geq 2^{2^{\cdots^2}}$ i-times.

Důsledek 27.46. The number of phases is $\mathcal{O}(\log^* n)$.

Důsledek 27.47. Fredman–Tarjan runs in $\mathcal{O}(m \log^* n)$.

Definice 27.48. Let $m, n \in \mathbb{N}$. Then

$$\beta(m,n) := \min\{i;\ \log^{(i)} n \leq \frac{m}{n}\}.$$

Důsledek 27.49. Fredman–Tarjan runs in $\mathcal{O}(m \cdot \beta(m,n))$.

Tvrzení 27.50.

$$(\forall i)\Big(\varrho(G) \geq \log^{(i)} n \to \text{Fredman–Tarjan} \in \mathcal{O}(im)\Big)\,.$$

Poznámka 27.51. We currently know

(1) a randomised approach with $\mathbb{E}\,\mathcal{O}(m)$,

(2) non-randomised $\mathcal{O}(m)$ for integers small enough that arithmetic operations are constant,

(3) $\mathcal{O}(m)$ on sorted edges,

(4) $\mathcal{O}(m \cdot \alpha(m,n))$ in general (inverse Ackermann function),

(5) also, $\mathcal{O}$(optimal), but we don't know what optimal is.

27.1 Vektory na RAMu

Definice 27.52 (Vektor v jednom slově). Nechť $d, b \in \mathbb{N}$, tž. $d\,(b+1) \leq w$. Pak vektor d b-bitových čísel je

$$x = \sum_{i=0}^{d-1} x_i 2^{(b+1)i}.$$

Tvrzení 27.53. Umíme následující operace v $\mathcal{O}(1)$: získat x_i, zapsat x_i, získat $\sum_i x_i$ (pokud se vejde do b), porovnávat (získat vektor 1/0 zda $x_i < y_i$).

Tvrzení 27.54. Z toho lze sestrojit Rank (počet x_i menších než α), minimum ze dvou vektorů.

27.2 Verifikace minimální kostry

Značení 27.55. Zafixujeme konkrétní kostru T. Také označíme jako $T[u,v]$ cestu v T mezi vrcholy u a v. Předpokládáme také ohodnocení hran $w : E \to \mathbb{R}^+$, tž. každá hrana má různou hodnotu.

Definice 27.56. Hrana $e \in E \setminus E(T)$ je *T-lehká* $\equiv$

$$(\exists f \in T[e])(w(f) > w(e))\,.$$

Věta 27.57. Kostra T je minimální, právě když neexistuje T-lehká hrana.

Problém 27.58 (Cestové maximum).
Vstup: Strom T, váhy w a dotazy $Q = \{\langle u,v\rangle\}_{u,v}$.
Výstup: Pro každý dotaz $u, v \in Q$ maximum na cestě $T[u,v]$.

Definice 27.59 (Borůvkův strom). Definujeme B_T, jež má

(1) jako listy vrcholy V,

(2) jako vnitřní vrcholy na i-té hladině komponenty Borůvkova algoritmu v i-té fázi.

Pozorování 27.60.

(1) Hloubka B_T je $\mathcal{O}(\log n)$.

(2) Každý list je na stejné hladině.

(3) Vybudovat B_T zvládneme v $\mathcal{O}(n)$.

Věta 27.61. Maximum na cestě $T[u,v]$ je stejné jako maximum na cestě $B_T[u,v]$.

Algoritmus 27.62 (Komlós).
Vstup: Strom T s váhami w, dotazy Q.
Výstup: Odpovědi na dotazy Q.

1: Označme pro stromovou hranu $e = uv$, kde v je níž:

$$Q_e = \{q \in Q;\, e \in T[q]\}\,.$$
$$T_e = \{\text{vršky cest z } Q_e, \text{ každý max. jednou, setříděné sestupně}\}\,.$$
$$P_e[i] = \max\{w(f)\,;\, f \in T[T_e[i]\,, v]\}\,.$$

2: FindPeaks(vrchol u, hrana p po které jsme přišli, T_p, P_p):
3: Uzavřeme každý $q \in Q_e$ se spodkem v u.
4: Pro v dítě u (hranou e):
5: Sestrojíme T_e z T_p (smažeme cesty končící v u, přidáme ty co začínají v u).
6: Sestrojíme P_e z P_p.
7: Prvky lehčí než e přepíšeme na $w(e)$.
8: FindPeaks(v, e, P_e, T_e).

Pozorování 27.63. Krok 7 je jediný kdy porovnáváme hrany mezi sebou, a to vždy binárním vyhledáváním pouze $\mathcal{O}(\log h)$-krát, kde h je hloubka stromu.

Lemma 27.64. Počet porovnání je $\mathcal{O}(n+q)$.

Nyní odvodíme lineární časovou složitost. Budeme využívat vektory ze sekce 27.1. Řekneme že *slot* je $w/3$-bitové číslo. Výhoda je, že pokud má funkce vstupy velikosti slotu, lze v $\mathcal{O}(n)$ předpočítat.

Reprezentace T_e. Identifikujeme vrcholy na cestě do současného v pomocí *hloubky*. To je identifikátor velikosti $\mathcal{O}(\log\log n)$. T_e pak můžeme uložit jako bitovou masku, která má na pozici identifikátoru v právě když $v \in T_e$.

Reprezentace P_e. Pro P_e budeme mít dvě různé reprezentace podle toho, kolik bude zrovna vrcholů v T_e: *malý* a *velký* list. Malý list je uložen jako vektor v jednom slotu, hodnoty jsou uloženy po řadě – pokud je vrchol z hora i-tý který je v T_e, bude na indexu i. Velké listy jsou pole vektorů, podle toho kolik jich potřebujeme. Hodnoty jsou v nich ale uloženy indexované přímo id vrcholu.

Lemma 27.65. V čase $\mathcal{O}(n+q)$ lze spočítat hloubky i T_e všech vrcholů a hran.

Lemma 27.66. T_e i P_e lze indexovat v čase $\mathcal{O}(1)$. P_e lze v tomto čase indexovat vrcholem, hloubkou, i pořadím z aktivních.

Věta 27.67. Kolmósův algoritmus pracuje v $\mathcal{O}(n+q)$ na RAMu.

27.3 Minimální kostra randomizovaně lineárně

LEMMA 27.68. Pokud je e T-těžká pro libovolnou kostru T, pak není v minimální kostře.

DEFINICE 27.69. Rozšiřme pojem „F-těžké“ hrany na les F jako *uzavírá cyklus a na něm je nejtěžší.*

LEMMA 27.70 (KARGEROVO VZORKOVACÍ). Nechť $H \subseteq G$ je náhodný podgraf G, kde každá hrana je přítomna s pravděpodobností p. Nechť F je minimální kostra (les) H. Pak

$$\mathbb{E}[\text{počet } e \in E \text{ které nejsou } F\text{-těžké}] \leq \frac{n}{p}.$$

ALGORITMUS 27.71 (KKT).
Vstup: Graf G.
Výstup: Minimální kostra (les) G.

1: Smažeme izolované vrcholy.
2: Pokud $G = \emptyset$, vrátíme $\emptyset$.
3: Provedeme 2 kroky kontrahujícího Borůvky na G, dostaneme zkontrahované hrany B.
4: Vzorkujeme $H \subseteq G$ s $p = \frac{1}{2}$.
5: $F \leftarrow \mathrm{KKT}(H)$.
6: $G' \leftarrow G - (F\text{-těžké hrany})$.
7: $R \leftarrow \mathrm{KKT}(G')$.
8: Vrátíme $R \cup B$.

LEMMA 27.72. Algoritmus běží ve worst-case $\mathcal{O}\big(\min\{n^2, m \log n\}\big)$.

VĚTA 27.73. Algoritmus běží v průměrně $\mathcal{O}(n + m)$.

27.4 Verifikace koster na PM

FAKT 27.74 (UNION-FIND). Problém UNION-FIND lze řešit v $\mathcal{O}(m \cdot \alpha(m, n) + n)$ na PM, kde m je počet dotazů a n je počet prvků.

DŮSLEDEK 27.75. Upravený problém „BAREVNÝ-UF“, kde je b barevných vrcholů, a každého UNIONu se účastní alespoň jeden barevný prvek, je možné dělat v

$$\mathcal{O}(m \cdot \alpha(m, b) + n)$$

DEFINICE 27.76 (MAKRO-MIKRO DEKOMPOZICE). Nechť $g \in \mathbb{N}$ a T je strom. Vrchol v označíme jako *kořen μ-stromu* $\equiv |T(v)| \leq g$ a pokud p je jeho rodič, pak $|T(p)| > g$. M-strom je pak všechno co není pod kořenem nějakého μ-stromu.

POZOROVÁNÍ 27.77. Pod každým listem M-stromu leží alespoň jeden μ-strom. Nějaký μ-strom ale může ležet i pod vnitřním vrcholem.

POZOROVÁNÍ 27.78. Počet listů M-stromu je $\leq \frac{n}{g}$.

TVRZENÍ 27.79. Mikro-makro dekompozici lze spočítat v $\mathcal{O}(n)$ na PM.

VĚTA 27.80. Problém LCA lze vyřešit na PM v $\mathcal{O}(n + q)$.

DEFINICE 27.81 (LINK-EVAL). Datová struktura pro LINK-EVAL si udržuje les s hranovými ohodnoceními, a podporuje operace

(1) LINK(u, v, c): u je kořen nějakého stromu, v je vrchol nějakého jiného, a c je ohodnocení nové uv-hrany,

(2) EVAL(v): v je vrchol, a odpovědí je maximum na cestě z v do kořene.

FAKT 27.82. LINK-EVAL lze pomocí UF implementovat v $\mathcal{O}(m \cdot \alpha(m, b) + n)$, kde m je počet dotazů a n je počet prvků.

VĚTA 27.83. Verifikace koster lze implementovat na PM v $\mathcal{O}(n + q)$.

Otázka 28

Testování rovinnosti grafů a kreslení do roviny

Fakt 28.1 (Kuratowski). G není rovinný, právě když obsahuje podrozdělení K_5 nebo $K_{3,3}$ jako podgraf, právě když platí $G \preccurlyeq_{\mathrm{m}} K_5$ nebo $G \preccurlyeq_{\mathrm{m}} K_{3,3}$.

Poznámka 28.2. Kdykoliv budeme v této kapitole mluvit o 2-souvislosti, myslíme *vrcholovou* 2-souvislost.

V této kapitole vybudujeme algoritmus pro kreslení grafů v $\mathcal{O}(n)$. Začneme definicí *bloků*, které budou reprezentovat nějakou část grafu na kterou se budeme moci dívat více méně jako na celek.

Definice 28.3. Nechť $e, f \in E$. Označíme $e \sim f$, pokud $e = f$ nebo e a f leží na společné kružnici.

Pozorování 28.4. Relace $\sim$ je ekvivalence.

Pozorování 28.5. G je 2-souvislý, právě když $\sim$ má jednu ekvivalenční třídu.

Definice 28.6. Ekvivalenčním třídám $\sim$ budeme říkat *bloky*.

Pozorování 28.7. Bloky jsou spojeny přes artikulace do stromové struktury. Každý blok je buď most, nebo 2-souvislý podgraf.

Hledání bloků. Prvním krokem naší konstrukce bude efektivní hledání toho jaké z již nakreslených vrcholů patří do kterých bloků. K tomu nám poslouží DFS.

Značení 28.8. Označme T nějaký fixní strom DFS. Pro každý vrchol $v \in V$ dále označme T_v podstrom pod v.

Tvrzení 28.9. Nechť uv je stromová hrana T. Označme $w_1, \ldots, w_k$ syny v. Pak

$$uv \sim vw_i \leftrightarrow \text{existuje zpětná hrana z } T_{w_i} \text{ do/nad } u.$$

Dále taky $vw_i \sim vw_j$, právě když $uv \sim vw_i \sim vw_j$.

Pozorování 28.10. Nechť B je libovolný blok. Pak $T \cap B$ je souvislý.

Lemma 28.11. Při výpočtu DFS lze v $\mathcal{O}(n)$ také počítat následující kvantity:

(1) $\textsc{Enter}(v) :=$ čas prvního vstupu do v.

(2) $\textsc{Ancestor}(v) := \min\{\textsc{Enter}(x)\,;\, vx \text{ je zpětná hrana, kde } x \text{ je nad } v\}$.

(3) $\textsc{LowPoint}(v) := \min\{\textsc{Ancestor}(x)\,;\, x \in V(T_v)\}$.

Pozorování 28.12. $vw_i \sim uv$, právě když $\textsc{LowPoint}(v) < \textsc{Enter}(v)$.

Algoritmus, který vybudujeme, bude hrany kreslit v pořadí klesajících Enterů. Při kreslení vrcholu v přidá všechny hrany vw_i jako kružnice délky 2, aby se pak dalo lépe analyzovat jejich obcházení.

Pozorování 28.13. Nenakreslená část je vždy souvislá.

Důsledek 28.14. Pokud otočíme stěny aby šla nenakreslená část nakreslit do vnější stěny, musí i všechny externí vrcholy ležet ve vnější stěně.

Definice 28.15 (Externí hrana). V bodě kdy kreslíme v je hrana e *externí*, pokud

$$|e \cap V(T_v)| = 1.$$

Definice 28.16 (Externí vrchol). Vrchol $v \in V$ je *externí* $\equiv$ je incidentní externí hraně, nebo je to artikulace incidentní bloku obsahující jiný externí vrchol.

Budeme využívat toho, že bloky lze otáčet. Tím si pomůžeme kdyby hrozilo že si nějaký externí vrchol zavřeme do ne-stěny.

Poznámka 28.17. Technický detail: Budeme si udržovat kopii každé artikulace pro každý její blok zvlášť.

Pozorování 28.18. Nechť $v \in V$ je právě dokreslený vrchol. Pak vrchol $w \in V$ je externí, právě když $\textsc{Ancestor}(w) < \textsc{Enter}(v)$, nebo nějaký syn s vrcholu w má $\textsc{LowPoint}(s) < \textsc{Enter}(v)$.

Abychom uměli rychle poznat externí vrcholy, budeme potřebovat strukturu BlockList.

Definice 28.19 (BlockList). Pro $w \in V$ je $\textsc{BlockList}(w)$ seznam pod-bloků, reprezentovaný kopiemi artikulací, jimiž jsou připojeny. BlockList je seřazen vzestupně dle LowPointů syna dané artikulace patřícího do daného bloku.

Tvrzení 28.20. BlockListy lze vybudovat a udržovat dohromady v $\mathcal{O}(n)$. Pomocí nich lze rozhodnout zda w je externí vrchol v $\mathcal{O}(1)$.

Poznámka 28.21 (Reprezentace bloku). Blok budeme reprezentovat pomocí vrcholů na jeho hranici: každý vrchol bude mít odkazy na své dva sousedy. V artikulaci přes kterou je blok připojen k nadřazenému bloku si budeme pamatovat *flip bit*, který určuje orientaci, tj. jestli je blok vůči pořadí překlopený. Na konci algoritmu tyto flip bity ještě propagujeme do všech vrcholů aby bylo všem vrcholům jasné jak jsou orientovány.

Abychom se nezdržovali s bloky ve kterých nemusíme zrovna nic dělat, zavedeme definici *živé* části grafu.

Definice 28.22. Vrchol u je *živý* $\equiv$ vede z něj zpětná hrana do právě nakresleného v, nebo je v jeho bloku nějaký jiný živý vrchol. Pokud vrchol není živý ani externí, říkáme mu *pasivní*.

Poznámka 28.23 (Nastavování živých vrcholů). Z v se podíváme na všechny zpětné hrany. Vrcholy kam vedou označíme za živé, dále pak jejich artikulace atd.

Poznámka 28.24 (Hledání artikulace současného bloku). Artikulaci hledáme tak, že jdeme současně oběma směry, pak asymptoticky v minimu času ji najdeme. Zároveň si ukládáme zkratky z těch vrcholů co jsme navštívili abychom hledání nemuseli opakovat.

Lemma 28.25. Pro každý nakreslený v trvá hledání živých vrcholů $\mathcal{O}(k + \ell)$, kde k je počet zpětných hran a ℓ je počet hran které po nakreslení zpětných zmizí z vnější stěny.

Při kreslení budeme využívat následující pravidla:

(1) V živém vrcholu kreslíme hrany v pořadí:
- nejprve zpětné hrany do v,
- podřízené živé interní bloky,
- podřízené živé externí bloky.

(2) Při příchodu do nového bloku vybíráme směr následovně:
- preferujeme směr k živému internímu vrcholu,
- pokud to je jiný směr než doposud, překlopíme blok.

Poznámka 28.26 (Implementace pravidel). Kdykoliv bychom při vybírání procházeli úsek pasivních vrcholů, opět si pořizujeme zkratky.

Algoritmus 28.27 (Kreslení do roviny).
Vstup: Graf G.
Výstup: Nakreslení grafu.

1: Pokud má G více než $3n - 6$ hran, rovnou končíme.
2: Prohledáme G do hloubky a počítáme Enter, Ancestor, LowPoint.
3: Vytvoříme BlockList každému vrcholu přihrádkovým tříděním.
4: Procházíme v v pořadí klesajícího Enteru:
5: Nakreslíme v se strom. hranami jako 2-cykly.
6: Označíme živý podgraf.
7: Pro každého syna v obcházíme jeho živý podgraf a kreslíme zpětné hrany do v.
8: Pokud nějaká hrana nebyla nakreslena, končíme, graf je nerovinný.
9: Propagujeme flip-bity do všech vrcholů.

Lemma 28.28. Pokud algoritmus nenakreslil nějakou hranu, graf není rovinný.

Věta 28.29. Algoritmus běží v $\mathcal{O}(n)$ a pokud je graf rovinný, vydá rovinné nakreslení, jinak ohlásí nerovinnost.

Otázka 29

Union-find

Definice 29.1 (Union-Find). Union-Find is a semi-dynamic datastructure defining a Union operation of two components and a Find operation, which returns whether two elements are in the same component.

Algoritmus 29.2 (Union-Find by Shrubs). We build a forest of trees, where each vertex has a pointer to its parent. Then Union is just changing the parent of one tree to the root of the other, and Find is just following the pointers to the root and comparing them.

Algoritmus 29.3 (Union by Rank). When doing Union, connect the shallower tree to the deeper one.

Důsledek 29.4. A tree of depth k has 2^k vertices. Both Union and Find then run in $\mathcal{O}(\log n)$.

Algoritmus 29.5. Whenever going up the tree, set the parent of all visited vertices to be the root.

Důsledek 29.6. In combination with UbR, this yields $\mathcal{O}(\alpha(n)) \subset \mathcal{O}(\log^* n)$.

29.1 Unions known in advance

From now, we assume that we know the unions in advance. We want to have an efficient Union-Find datastructure.

Poznámka 29.7. Another way to look at the problem is that we start with already connected sets, and we are deleting edges.

Definice 29.8 (Frederickson's partition). Let T be a tree, whose each vertex has deg ≤ 3. Let $k \in \mathbb{N}$. Let $V_1 \dot\cup \cdots \dot\cup V_c = V(T)$. Then we will denote

- a *cluster* C_i as the graph induced by V_i,
- $|C_i| := |V(C_i)|$,
- an edge of T is *outer* $\equiv$ it connects two clusters, *inner* otherwise,
- the external degree of a cluster, $\mathrm{ed}(C_i)$ as the number of neighbouring external edges,
- the clusters of degree 0 are *isolated*, of degree 1 are *leaves*, of degree 2 are *paths*, and of degree 3 are *branching*.

Then $V_1 \dot\cup \cdots \dot\cup V_n$ is a *Frederickson's k-partition* $\equiv$

(1) $(\forall i)\, C_i$ is connected,

(2) $(\forall i)(\mathrm{ed}(C_i) \leq 3)$,

(3) $(\forall i)(|C_i| \leq k \wedge (\mathrm{ed}(C_i) = 3 \to |C_i| = 1))$,

(4) No cluster can be joined together.

Tvrzení 29.9. Let C and D be neighbouring clusters. Then C and D can be joined together, if and only if $|C| + |D| \leq k$ and one of the following holds:

(1) neither of C and D are branching,

(2) one is branching, and the other is a leaf.

Věta 29.10 (Frederickson). Let $V_1 \dot\cup \cdots \dot\cup V_c = V(T)$ be a Frederickson's k-partition. Then $c \in \mathcal{O}\left(\frac{n}{k}\right)$.

Věta 29.11 (Frederickson). A Frederickson's partition can be found in $\mathcal{O}(n)$.

Algoritmus 29.12 (UF using Frederickson's partition).
Vstup: A tree T.
Výstup: DS for Union-Find.

1: For each vertex v of degree > 3:
2: Replace v by a path, such that each vertex on the path has degree ≤ 3.
3: Find a Frederickson's $(\log n)$-partition.
4: Initialize a bit-wise representation of each cluster.
5: Initialize the coloring structure on the cluster tree.

Otázka 30

Link-cut stromy

Definice 30.1. Link-Cut strom udržuje pro každou hranu zda je *tlustá*, nebo *tenká* tak, že do každého vrcholu vede ≤ 1 tlustá hrana.

Definice 30.2 (Expose). Operace Expose přenastaví tlusté a tenké hrany tak, aby z v do rootu vedla tlustá cesta.

Definice 30.3 (Rozhraní pro cesty).

- Prev, Next, First, Last,
- Cut, Link, Reverse,
- Cost, PathMin,
- SetCost, PathUpdate,

Algoritmus 30.4 (Expose).
Vstup: Vrchol v.
Výstup: Aktualizovaná DS.

1: $u \leftarrow \text{Last}(v)$.
2: $r \leftarrow \text{LightParent}(v)$.
3: $q \leftarrow \text{Pred}(r)$.
4: Pokud $q \neq \emptyset$:
5: $\quad \text{Cut}(q)$.
6: $\quad \text{LightParent}(q) \leftarrow r$.
7: $\text{Link}(u, r)$.
8: $\text{LightParent}(u) \leftarrow \emptyset$.

Věta 30.5. S Operací Expose umíme operace

(1) Parent(v): rodič vrcholu v.
(2) Cut(v): odstranění hrany.
(3) Link(u, v): přidání hrany (u je kořen jiného stromu).
(4) Evert(v): udělat z v kořen.
(5) Cost(v): váha v.
(6) PathMin(v): minimum na cestě z v do kořene.
(7) SetCost(v): nastaví váhu vrcholu.
(8) PathUpdate(v, δ): upraví váhu $+\delta$ na cestě z v do kořene.

Reprezentace cest. Cesty budeme reprezentovat jako *Splay stromy*, které jako vnitřní vrcholy mají vrcholy cesty. Kdykoliv sáhneme na vrchol, vySplayujeme ho do kořene. PathUpdate a Reverse budeme dělat líně.

Tvrzení 30.6. Taková reprezentace je korektní a každá operace s cestou odpovídá $\mathcal{O}(1)$ operacím Splay a mají amortizovanou cenu Splaye.

Definice 30.7 (Společný strom). *Společný strom* je složen ze Splay stromů cest, jejichž kořeny jsou speciálními „tenkými“ hranami napojené na ty vrcholy, kam vede skutečná tenká hrana cesty ve stromě.

Algoritmus 30.8 (Expose).
Vstup: Společný strom, vrchol v.
Výstup: Upravený společný strom s Exposeovaným v.

1: Najdeme všechny přechody po lehkých hranách na cestě z v do kořene.
2: Provedeme Splay uvnitř tlustých stromů u každého jeho prvního navštíveného vrcholu.
3: Změníme tlusté hrany na tenké a obráceně.
4: Provedeme Splay(v) v novém těžkém stromě.

Věta 30.9. Takto definovaná operace Expose s $s(v)$ jako počtem *všech* potomků (včetně těch přes lehké hrany) má amortizovanou cenu $\mathcal{O}(\log n)$.

Věta 30.10. Dinic s Link-Cut stromem na hledání blokujících toků pracuje v $\mathcal{O}(mn \log n)$.

Otázka 31

Dynamické komponenty souvislosti

Definice 31.1 (dynamic connectivity). A datastructure for *dynamic connectivity* has:

(1) Insert for inserting an edge,

(2) Delete for deleting an edge,

(3) Connected for query if two vertices are connected.

Definice 31.2 (ET-sequence). Let G be a graph. An *ET-sequence* of G is a sequence of vertices given by the order they are encountered by DFS from an arbitrary root.

Poznámka 31.3. ET-sequence for G isn't unique.

Poznámka 31.4. For each $v \in V$, we define a *primary occurance* of v in an ET-sequence, to which we point from other places.

Pozorování 31.5. Let s be an ET-sequence. Then

$$|s| = 2n - 1 \in \Theta(n)\,.$$

31.1 Forests

From now, we'll assume that G is a forest.

ALGORITMUS 31.6 (FOREST DYNAMIC CONNECTIVITY).
Vstup: Forest F.
Výstup: DS for dynamic connectivity.

1: For $T \in F$:
2: Choose a root $r_T \in V(T)$ arbitrarily.
3: $s_T \leftarrow$ an ET-sequence from r_T.
4: CUT(u, v): ▷ WLOG u is a parent of v.
5: Disassemble s to
$$s = \alpha \cdot u \cdot v \cdot \beta \cdot v \cdot u \cdot \gamma.$$
6: Return two trees: $\alpha \cdot u \cdot \gamma$ and $v \cdot \beta \cdot v$.
7: CHANGE ROOT(u):
8: Disassemble s to
$$s = r \cdot \alpha \cdot u \cdot \beta \cdot u \cdot \gamma \cdot r, \qquad u \notin \alpha, u \notin \gamma.$$
9: $s \leftarrow u \cdot \beta \cdot u \cdot \gamma \cdot r \cdot \alpha \cdot u$
10: $r \leftarrow u$
11: LINK(u, v):
12: CHANGE ROOT(u)
13: CHANGE ROOT(v)
14: Disassemble s_u and s_v to
$$s_u = u \cdot \alpha \cdot u, \qquad s_v = v \cdot \beta \cdot v.$$
15: $s \leftarrow u \cdot \alpha \cdot u \cdot v \cdot \beta \cdot v \cdot u$

POZNÁMKA 31.7. When doing operations, we need to mantain the primary occurances.

31.2 Fast ET-sequence operations

DEFINICE 31.8 (ET-TREE). Let s be an ET-sequence. Then an *ET-tree* is an (a, b)-tree, where

(1) The leaves hold s,
(2) The inner vertices hold edge traversals.

POZNÁMKA 31.9. Usually, $b = 2a$.

POZNÁMKA 31.10. An ET-tree has operations INSERT, DELETE, JOIN, SPLIT, all in $\mathcal{O}(a \log_a n)$.

VĚTA 31.11. CUT, LINK in FOREST DYNAMIC CONNECTIVITY can be translated into $\mathcal{O}(1) \times$ SPLIT, JOIN.

31.3 Non-Forests

POZNÁMKA 31.12 (IDEA). We want to keep the spanning trees and non-tree edges.

POZNÁMKA 31.13. When deleting a tree edge, we need to find a replacement non-tree edge.

DEFINICE 31.14 (LEVEL STRUCTURE). Let G be a graph. We'll define

(1) for an edge $e \in E$ a *level* $\ell(e) \in \{0, \dots, L\}$, and
(2) for F a spanning forest of G, $F_i := \{e \in E(F)\,;\, \ell(e) \geq i\}$,

such that the following invariants hold:

(I1) F is a maximal spanning forest with respect to ℓ,
(I2) $(\forall i)(\forall T \in F_i)\big(|V(T)| \leq \frac{n}{2^i}\big)$.

Algoritmus 31.15 (General dynamic connectivity).
Vstup: A graph G.
Výstup: A DS for dynamic connectivity.

1: $(\forall e)(\ell(e) \leftarrow 0)$
2: $F \leftarrow$ a spanning forest of G.
3: $(\forall i)$ initialize dynamic connectivity DS for F_i.
4: Insert(e):
5: $\ell(e) \leftarrow 0$
6: If both ends in the same component, insert a non-tree edge.
7: Otherwise, insert a tree edge.
8: Delete(uv):
9: If uv is a non-tree edge, delete it and stop.
10: $i \leftarrow \ell(uv)$
11: While $i \geq 0$:
12: $T \leftarrow$ tree in F_i containing uv; $T_1, T_2 \leftarrow$ the trees after removing uv, such that $|V(T_1)| \leq |V(T_2)|$.
13: $(\forall e \in E(T_1), \ell(e) = i)(\ell(e) \leftarrow i + 1)$
14: For $e \in \delta(T_1)$, such that $\ell(e) = i$:
15: If e goes to T_2, insert e as a tree edge and finish.
16: Otherwise, $\ell(e) \leftarrow \ell(e) + 1$
17: $i \leftarrow i - 1$

Věta 31.16. Over the lifetime of the datastructure described above, both Invariants (I1) and (I2) hold.

Tvrzení 31.17 (Implementation). When using the ET-trees to store the DS, we need to keep track of:

(1) tree size: each node holds the number of leaves in its subtree,
(2) non-tree edges: stored at the incident vertices (their primary occurance); internal nodes keep the number of non-tree edges in subtree,
(3) tree edges: similar to non-tree edges.

Věta 31.18. For a constant, we have

(1) Insert, Delete in $\mathcal{O}\left(\log^2 n\right)$ amortised,
(2) Find in $\mathcal{O}(\log n)$ worst case.

Věta 31.19. When we set $a = \log n$ in F_0, we have

(1) Insert, Delete in $\mathcal{O}\left(\log^2 n\right)$ amortised,
(2) Find in $\mathcal{O}\left(\frac{\log n}{\log \log n}\right)$ worst case.

Otázka 32

Společní předchůdci ve stromech

Definice 32.1 (Lca). Let T be a rooted tree. Then the Lowest common ancestor problem is

$$\textsc{Lca}(x, y) := v, \text{ the deepest common ancestor of both } x, y.$$

Algoritmus 32.2. Trivially, we can walk twice up the tree, yielding $\mathcal{O}(n)$.

Definice 32.3 (Rmq). Let $x_1, \dots, x_n$ be a sequence. Then the Range minimal query problem is

$$\textsc{Rmq}(i, j) := \arg\min\{x_k;\ i \le k \le j\}.$$

Tvrzení 32.4. Lca $\to$ Rmq in $\mathcal{O}(n)$.

Tvrzení 32.5 (Rmq ± 1). When converting Lca to Rmq, we actually get a special case Rmq ± 1, where

$$(\forall i)(|a_i - a_{i+1}| = 1)\,.$$

Algoritmus 32.6 (Solving Rmq). Let $x_1, \dots, x_n$ be a sequence. Then solving of Rmq can be done by

(1) *precomputation* $\equiv$ build $\mathcal{O}(n^2)$, query $\mathcal{O}(1)$,

(2) *range trees* $\equiv$ build $\mathcal{O}(n)$, query $\mathcal{O}(\log n)$,

(3) $\forall k$ *pre-compute all ranges of size* 2^k $\equiv$ build $\mathcal{O}(n \log n)$, query $\mathcal{O}(1)$.

Algoritmus 32.7 (solve Rmq ± 1 via Decomposition).
Vstup: $a_1, \dots, a_n$, an instance of Rmq ± 1.
Výstup: A structure with a Query for Rmq ± 1.

1: $b_1, \dots, b_m \leftarrow$ Split $a_1, \dots, a_n$ into blocks of size $b = \frac{1}{2}\log n$.
2: Record the *type* of each block: the sequence of $a_i - a_{i-1}$ for each element. ▷ There are $\le \sqrt{n}$ types.
3: Build the *quadratic* structure for each block type. ▷ $\mathcal{O}\left(\sqrt{n}\log^2 n\right) = \mathcal{O}(n)$
4: Create a sequence $c_1, \dots, c_m$, such that c_i is the minimum of the block b_i.
5: Build the *logarithmic* structure for $c_1, \dots, c_m$. ▷ $\mathcal{O}\left(\frac{n}{b}\log\frac{n}{b}\right) = \mathcal{O}(n)$
6: Query(i, j): ▷ $\mathcal{O}(1)$
7: Get the minimum of the corresponding blocks.
8: Get the minimum of the corners.
9: Return the minimum of them.

Definice 32.8 (Cartesian tree). Let $a_1, \dots, a_n$ be a sequence. Let $j := \arg\min_i a_i$. Then a *Cartesian tree* is a tree, whose root is a_j, the left subtree is the Cartesian tree of $a_1, \dots, a_{j-1}$ and the right subtree is the Cartesian tree of $a_{j+1}, \dots, a_n$.

Tvrzení 32.9. A Cartesian tree can be constructed in $\mathcal{O}(n)$.

Tvrzení 32.10. Rmq $\to$ Lca in $\mathcal{O}(n)$.

32.1 Pointer Machine

FAKT 32.11 (UNION-FIND). Problém UNION-FIND lze řešit v $\mathcal{O}(m \cdot \alpha(m,n) + n)$ na PM, kde m je počet dotazů a n je počet prvků.

DŮSLEDEK 32.12. Upravený problém „BAREVNÝ-UF“, kde je b barevných vrcholů, a každého UNIONu se účastní alespoň jeden barevný prvek, je možné dělat v

$$\mathcal{O}(m \cdot \alpha(m,b) + n)$$

DEFINICE 32.13 (MAKRO-MIKRO DEKOMPOZICE). Nechť $g \in \mathbb{N}$ a T je strom. Vrchol v označíme jako *kořen* μ*-stromu* $\equiv |T(v)| \leq g$ a pokud p je jeho rodič, pak $|T(p)| > g$. M-strom je pak všechno co není pod kořenem nějakého μ-stromu.

POZOROVÁNÍ 32.14. Pod každým listem M-stromu leží alespoň jeden μ-strom. Nějaký μ-strom ale může ležet i pod vnitřním vrcholem.

POZOROVÁNÍ 32.15. Počet listů M-stromu je $\leq \frac{n}{g}$.

TVRZENÍ 32.16. Mikro-makro dekompozici lze spočítat v $\mathcal{O}(n)$ na PM.

VĚTA 32.17. Problém LCA lze vyřešit na PM v $\mathcal{O}(n+q)$.