3. sada domácich úloh

Termín: **utorok 20. 5. 2025 23:59**

Úloha 1

Koľko existuje neklesajúcich postupností dĺžky 200 pozostávajúcich z prvkov množiny $\{1, \ldots, 100\}$, ktoré každé z čísel $1, 2, \ldots, 100$ obsahujú najviac 30-krát? Výsledok môžete uviesť v tvare jednej sumy. Vaše tvrdenie dokážte.

Nech M je množina všetkých neklesajúcich 200-prvkových postupností prvkov množiny $\{1, \ldots, 100\}$ a nech pre $i \in \{1, 2, \ldots, 100\}$ je M_i množina všetkých postupností z M, ktoré obsahujú aspoň 31-krát číslo i. Množinu hľadaných postupností vieme vyjadriť ako $M - (M_1 \cup M_2 \cup \cdots \cup M_{100})$. Počet jej prvkov vieme teda vyjadriť pomocou princípu inklúzie a exklúzie, presnejšie jeho dôsledku na výpočet komplementu zjednotenia množín.

Začneme určením mohutnosti

$$|M_{i_1} \cap M_{i_2} \cap \cdots \cap M_{i_k}|.$$

Ide o množinu všetkých postupností z M, ktoré obsahujú každé z čísel $i_1, i_2, \ldots, i_k$ aspoň 31-krát. Z každej takejto postupnosti vieme odstrániť povinných 31 výskytov každého z k predpísaných čísel. Ostane nám tak neklesajúca postupnosť dĺžky $200 - 31k$ čísel z množiny $\{1, \ldots, 100\}$. Tú zas vieme reprezentovať ako postupnosť $200-31k$ guľôčok a 99 paličiek. Paličky rozdelia guľôčky na 100 úsekov a počet guľôčok v j-tom úseku bude počet čísel j v postupnosti. Takáto postupnosť guľôčok a paličiek je určená výberom 99 miest z celkových $299 - 31k$ miest, na ktorých budú paličky, a na to máme

$$\binom{299 - 31k}{99}$$

možností.

Podľa princípu inklúzie a exklúzie máme túto hodnotu sčítať pre všetky výbery k indexov $i_1, i_2, \ldots, i_k$. Keďže mohutnosť vypočítaného prieniku nezávisí od výberu konkrétnych indexov, tak len $\binom{100}{k}$-krát sčítame toto číslo, čím máme

$$S_k = \binom{100}{k}\binom{299 - 31k}{99}.$$

Na záver tak dostávame

$$|M - (M_1 \cup M_2 \cup \cdots \cup M_{100})| = \sum_{k=0}^{100} (-1)^k \binom{100}{k}\binom{299 - 31k}{99}.$$

Výsledok ešte možno zjednodušiť, keď si uvedomíme, že pre $k \geq 7$ je $299-31k < 99$, teda dostávame nulové kombinačné číslo. Tieto členy tak možno zo sumy vyhodiť, čím dostaneme

$$\sum_{k=0}^{6} (-1)^k \binom{100}{k}\binom{299 - 31k}{99}.$$

Nech $f\colon \mathbb{N}^+ \to \mathbb{R}$ je zobrazenie s predpisom $f(n) = \dfrac{1}{6\sqrt{n}} + \dfrac{10}{n}$. Rozhodnite, či platí

a) $f(n) = \Theta(\frac{1}{n})$, b) $f(n) = \Theta(\frac{1}{\sqrt{n}})$, c) $f(n) = o(1)$.

d) Nájdite také konštanty $a, b \in \mathbb{R}$, pre ktoré platí $f(n) \sim an^b$.

Vaše tvrdenia poriadne dokážte (v d) nemusíte dokazovať, že iné konštanty neexistujú). Pri dôkazoch vychádzajte len z definícií, bez odvolávania sa tvrdenia o asymptotických odhadoch.

Keďže všetky funkcie v zadaní sú zjavne kladné, absolútne hodnoty nebudeme písať.

Časť a) sporom s ohraničenosťou Ukážeme, že tvrdenie neplatí, konkrétne, že neplatí $f(n) = O(1/n)$. Pre spor predpokladajme opak, teda, že existujú $c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$ také, že pre všetky $n \geq n_0$ platí

$$\begin{aligned} \frac{1}{6\sqrt{n}} + \frac{10}{n} &\leq c \cdot \frac{1}{n}, \qquad | \cdot 6n \\ \sqrt{n} + 60 &\leq 6c, \\ \sqrt{n} &\leq 6c - 60, \\ n &\leq (6c-10)^2. \end{aligned}$$

Dostali sme, že všetky prirodzené čísla počnúc nejakým n_0 sú ohraničené konštantnou $(6c-10)^2$, čo je spor.

Časť a) priamo Ukážeme, že tvrdenie neplatí, na čom nám stačí ukázať, že $f(n) \neq O(1/n)$. Uvažujme ľubovoľné $c \in \mathbb{R}^+$ a $n_0 \in \mathbb{N}$. Zvoľme $n = \max(36c^2 + 1, n_0)$. Potom platí $n \geq n_0$ a:

$$\begin{aligned} n &> 36c^2 \\ &\Downarrow \\ \sqrt{n} &> 6c \\ &\Downarrow \\ \sqrt{n} &> 6c - 6 \\ &\Downarrow \\ \sqrt{n} + 6 &> 6c \qquad | : 6n \\ &\Downarrow \\ \frac{1}{6\sqrt{n}} + \frac{10}{n} &> c \cdot \frac{1}{n} \end{aligned}$$

Časť b) Dokážte, že $f(n) = \Theta(1/\sqrt{n})$.

Pre každé $n \geq 1$ platí

$$f(n) = \frac{1}{6\sqrt{n}} + \frac{10}{n} \leq \frac{1}{6\sqrt{n}} + \frac{10}{\sqrt{n}} = \frac{61}{10} \cdot \frac{1}{\sqrt{n}},$$

teda $f(n) = O(1/\sqrt{n})$.

Pre každé $n \geq 1$ platí

$$f(n) = \frac{1}{6\sqrt{n}} + \frac{10}{n} \leq \frac{1}{6\sqrt{n}} \geq \frac{1}{6\sqrt{n}} = \frac{1}{6} \cdot \frac{1}{\sqrt{n}},$$

teda $f(n) = \Omega(1/\sqrt{n})$.

Časť c) Vypočítajme limitu

$$\lim_{n\to\infty} \frac{f(n)}{1} = \lim_{n\to\infty} \left(\frac{1}{6\sqrt{n}} + \frac{10}{n} \right) = 0 + 0 = 0,$$

teda $f(n) = o(1)$.

Časť d) Nech $a = \frac{1}{6}$ a $b = -\frac{1}{2}$. Potom

$$\lim_{n\to\infty} \frac{f(n)}{\frac{1}{6}n^{-\frac{1}{2}}} = \lim_{n\to\infty} \left(\frac{1}{6\sqrt{n}} + \frac{10}{n} \right) \cdot 6 \cdot \sqrt{n} = \lim_{n\to\infty} \left(1 + \frac{60}{\sqrt{n}} \right) = 1 + 0 = 1,$$

teda $f(n) = \frac{1}{6}n^{-\frac{1}{2}}$.

Úloha 1. *(3 body)* Nech G je súvislý 3-regulárny graf. Dokážte, že graf G má takú kostru T, že každá kružnica grafu G má s kostrou T aspoň jednu spoločnú hranu.

Bonus. *(cca 2 body)* Napíšte algoritmus, ktorý načíta zo štandardného vstupu graf a vypíše na štandardný výstup jeho kostru s požadovanou vlastnosťou.

Formát vstupu. V prvom riadku sa nachádza kladné celé číslo n predstavujúce počet vrcholov grafu, ktoré číslujeme $0, 1, \ldots, n-1$. Nasleduje n riadkov, ktoré postupne predstavujú susedov vrcholov $0, 1, \ldots, n-1$. Každý z týchto n riadkov obsahuje tri medzerou oddelené čísla vrcholov, s ktorými je príslušný vrchol spojený.

Formát výstupu. Na výstup vypíšte hrany kostry, každú do samostatného riadku ako dve medzerou oddelené čísla, ktoré predstavujú čísla vrcholov, ktoré spája.

Hodnotenie. V prvom rade je dôležitá správnosť vášho programu, ktorá musí byť zdôvodnená (komentármi v kóde alebo v pdf s riešeniami ostatných úloh). Dĺžka takéhoto zdôvodnenia silno závisí od zvoleného algoritmu. Existujú programy, ktorým stačí veľmi krátke zdôvodnenie. V druhom rade je dôležitá efektívnosť programu. Veľmi efektívne programy môžu byť ocenené aj viac ako 2 bodmi.

Formát vstupu a výstupu je potrebné striktne dodržať! Teda nevypisujte ani veci „Zadajte n:“, „Odpoveď je“, nenačítavajte vstup zo súboru a podobné veci.

Príklad vstupu:

```
8
1 3 4
0 2 5
1 3 6
2 0 7
0 7 5
1 4 6
2 5 7
3 6 4
```

Príklad výstupu:

```
0 1
0 3
0 4
3 2
3 7
7 6
6 5
```

Riešenie cez upravovanie kostry

Keďže graf G je súvislý, tak isto nejakú kostru má. Nech T je kostra grafu G taká, že počet kružníc grafu G, ktoré neobsahujú žiadnu hranu kostry T, je najmenší možný. Pre spor predpokladajme, že tento počet takýchto *zlých* kružníc je nenulový. Nech C je jedna taká kružnica s vrcholmi

$v_1, v_2, \ldots, v_k$ spojenými v tomto poradí. Každý z vrcholov v_i (pre $1 \leq i \leq k$) má dve hrany v kružnici C, preto jeho zvyšná hrana – označme si ju e_i – musí byť obsiahnutá v kostre T.

Nech T' je podgraf grafu G, ktoré vznikne z kostry T pridaním hrany v_1v_2 a odobraním hrany e_1. Podgraf T' je zjavne súvislý. Každá kružnica vytvorená pridaním hrany v_1v_2 musí ďalej viesť cez hranu e, teda podgraf T' je aj acyklický. Preto T' je kostra grafu G. Kostra T' obsahuje aspoň jednu hranu kružnice C. Navyše tvrdíme, že ľubovoľná kružnica K grafu G, ktorá má aspoň jednu hranu spoločnú s kostrou T, má spoločnú hranu aj s kostrou T'. Ak by to tak nebolo, tak kružnica K musela mať s kostrou T spoločnú iba hranu e_1. To je totiž jediná hrana T, ktorá sa nenachádza v kostre T'. Žiadna z hrán $e_2, \ldots, e_n$ však nepatrí do kružnice K. Tým pádom kružnica K nemá ako opustiť vrcholy kružnice C, čo je spor.

Dostali sme tak, že každá kružnica grafu G, ktorá má spoločnú hranu s kostru T, má spoločnú hranu aj s kostrou T'. Navyše, s kostrou T' má spoločnú hranu aj kružnica K. Tým pádom sme našli kostru T' grafu G, ktorá má menej zlých kružníc ako kostra T. To je spor s predpokladom, že kostra T mala najmenej zlých kružníc. Preto kostra T je hľadaná kostra grafu G, ktorá nemá žiadne zlé kružnice.

Program sledujúci toto riešenie

Na základe týchto myšlienok vieme napísať aj program pre bonus. Stručne naznačíme, že ako:

- Načítame vstup.
- Nájdeme nejakú kostru T grafu G – na to môžeme použiť štandardné algoritmy ako DFS, BFS, nejaké ich modifikácie alebo si môžeme napísať aj vlastný podľa dôkazu z prednášky (veta, že každý súvislý graf má kostru).
- Vo while cykle opakujeme:
  1. Zistíme, či $G - T$ obsahuje kružnicu, ak nie, tak končíme a vypíšeme T.
  2. Ak áno, tak v kostre T vymeníme dve hrany podľa dôkazu vyššie.

Takto napísaný program priamo demonštruje naše opísané riešenie – takémuto dôkazu cez extremálny princíp totiž zodpovedá while cyklus, v ktorom tvoríme kostry s menším a menším počtom kružníc, dokým to ide. Takéto opakované hľadanie kružnice však nie je úplne efektívne.

Program možno zefektívniť na základe zopár pozorovaní. Keďže každý vrchol má v kostre T stupeň aspoň 1, tak v $G - T$ má stupeň najviac 2. To znamená, že graf $G - T$ je zjednotenie disjunktných ciest a kružníc. Keď teda nájdeme v grafe $G - T$ kružnicu a zbavíme sa jej upravením kostry T, tak môžeme pokračovať v hľadaní kružníc ďalej.

Pokiaľ máte záujem vidieť celý program, tak mi napíšte, spolu aj s tým, či máte preferenciu na jazyk (C++ alebo Python).

Riešenie cez prehľadávanie do hĺbky (DFS)

Viacerým sa podarilo zistiť, že správnu kostru nám nájde DFS. Pre programovaciu časť to znamená, žeby mohol stačiť takýto program.

```
n = int(input())
G = [list(map(int, input().split())) for _ in range(n)]

def dfs(G, v, visited, tree_edges):
    for neigh in G[v]:
```

```
        if not visited[neigh]:
            tree_edges.append((v, neigh))
            visited[neigh] = True
            dfs(G, neigh, visited, tree_edges)

start = 0
visited = [False] * n
tree_edges = []
visited[start] = True
dfs(G, start, visited, tree_edges)

for e in tree_edges:
    print(*e)
```

Ako však dokázať, že takýto algoritmus je správny? Dôležitým uvedomením je, že nestačí nám len taká hocijaká kostra. Napr. ak prehľadáme graf K_4 do šírky, tak nám ostane kružnica dĺžky 3 mimo kostry. Preto by mal náš dôkaz využiť isté vlastnosti prehľadávania do hĺbky. Ďalším problémom je, že prehľadávanie do hĺbky máme opísané len algoritmom, čo nie je veľmi korektný spôsob z pohľadu matematických formalizmov. Ukážeme si, ako takéto riešenie spísať bez potreby odvolávať sa na algoritmus.

Definície pojmov

Prehľadávaní grafov existuje veľa a sú rôzne spôsoby, ako ich vieme vyjadriť. Jeden z formálne jednoduchých spôsobov je definovať *prehľadávanie grafu* ako *zakorenenú kostru*, teda usporiadanú dvojicu (T, r), kde T je kostra G a r je ľubovoľný vrchol grafu G nazývaný *koreň*. Jediné, čo to znamená, je, že sme jeden z vrcholov kostry prehlásili za špeciálny.

V zakorenenom strome (T, r) (teda aj kostre) vieme hovoriť o tom, že nejaký vrchol je pod iným: *vrchol u je pod vrcholom v* ak (jediná) u-r-cesta v strome T prechádza vrcholom v. (Záujemcovia si môžu rozmyslieť, že po pridaní reflexívnosti ide o usporiadanie množiny $V(G)$, vo väčšine stromov neúplné.)

Prehľadávanie grafu $G = (V, E)$ do hĺbky definujeme ako kostru T grafu G zakorenenú vo vrchole r takú, že pre každú hranu $uv \in E(G)$ je vrchol u pod vrcholom v alebo naopak.

Dôkaz

Veta 1. *Každý súvislý graf G obsahuje pre každý vrchol $r \in V(G)$ kostru T zakorenenú vo vrchole v, ktorá je prehľadávaním do hĺbky.*

Dôkaz. Tvrdenie dokážeme matematickou indukciou vzhľadom na počet vrcholov grafu.

Nech n a predpokladajme, že tvrdenie platí pre všetky súvislé grafy s menej ako n vrcholmi. Uvažujme n-vrcholový graf G a jeho vrchol r. Po odstránení vrchola r dostaneme $k \geq 1$ komponentov súvislosti $G_1, G_2, \ldots, G_k$. Keďže graf G je súvislý, tak v každom komponente G_i existuje vrchol r_i, ktorý susedí s vrcholom r. Každý komponent G_i má menej ako n vrcholov, preto podľa indukčného predpokladu má prehľadávanie do hĺbky (T_i, r_i). Nech T je podgraf G, ktorý vznikne zjednotením podgrafov $T_1, T_2, \ldots, T_k$ a pridaním hrán $rr_1, rr_2, \ldots, rr_k$. Dokážeme, že podgraf T je prehľadávaním do hĺbky:

- T je súvislý: Z každého jeho vrcholu v zjavne existuje cesta do vrcholu r, z čoho vyplýva súvislosť.

- T je acyklický: V jeho podgrafoch T_i sa nenachádza kružnica. Každé z pridaných hrán rr_i je mostom, preto neleží na kružnici.
- Je to prehľadávaním do hĺbky: Uvažujme ľubovoľnú hranu $uv \in E(G)$. Ak jeden z jej vrcholov je koreň, BUNV $u = r$, tak vrchol v je pod u. Ďalej uvažujme, že $u \neq r \neq v$. Vtedy u aj v musia byť vrcholy rovnakého komponentu G_i, keďže medzi komponentmi súvislosti nemôže viesť hrana. Potom z indukčného predpokladu dostávame, že jeden z vrcholov hrany uv je pod druhým.

□

Nech (T, v) je prehľadávanie grafu G do hĺbky zakorenené v ľubovoľnom jeho vrchole r. Pre spor predpokladajme, že graf G obsahuje kružnicu C, ktorá nemá s kostrou T spoločnú hranu. Kružnica musí obsahovať aspoň tri hrany, z ktorých najviac dve môžu byť incidentné s koreňom r. Preto kružnica C obsahuje hranu e, ktorá spája dva listy kostry T rôzne od jej koreňa r. Avšak list nemôže byť pod iným listom, ktorý nie je koreň. To je v spore s vetou 1.

Poznámky

Posledný odsek uvedeného riešenia sa dá brať ako správne riešenie s chýbajúcimi detailami a v nebonusovej časti som za riešenia takéhoto typu nestrhal veľa bodov. Je však potrebné, aby tam bol ošetrený samostatne koreň, nakoľko z neho môžu viesť dve mimokostrové hrany do listov. Miesto pojmov *byť pod iným vrcholom* možno hovoriť aj o tom, že niektorý vrchol *bol navštívený skôr* ako iný. Takýto jazyk je bežný v oblasti programovania. Tiež takéto riešenie by obstálo na algoritmických predmetoch, kde sa na exaktné matematické vyjadrovanie až tak nedbá.

Iný spôsob, ako definovať prehľadávanie grafu $G = (V, E)$ je $t = (v_0, v_1, \ldots, v_n)$ jeho vrcholov taká, že pre každé celé číslo i, $1 \leq i < n$ je vrchol v_i spojený s práve jedným z vrcholom z $v_0, v_1, \ldots, v_{i-1}$. S týmto ste sa mohli stretnúť v cvičeniach 12, v úlohe 17. Toto zoradenie (alebo aj číslovanie) vrcholov zodpovedá poradiu, v ktorom sme vrcholy pri prehľadávaní objavili. Vlastnosť DFS možno formulovať aj pomocou takýchto postupností. Postupnosť t je vlastne bijekcia $\{0, 1, \ldots, |V(G)| - 1\} \to V(G)\}$ a možno sa na ňu pozerať aj opačným smerom – ako očíslovanie vrcholov. Ich čísla predstavujú poradie, v ktorom boli navštívené. Na týchto hodnotách sa zakladajú aj pokročilejšie algoritmy využívajúce prehľadávanie do hĺbky alebo iné. O nich sa naučíte na predmete *Tvorba efektívnych algoritmov*.

Ponaučenia na záver

S úlohou takéhoto typu sa môžete stretnúť aj na iných, informatických predmetoch. Pekne zapadá do predmetu *Tvorba efektívnych algoritmov*, na ktorom sa budete učiť aj o vlastnostiach prehľadávania do hĺbky a jeho využitiu na efektívne riešenie rôznych problémov. Na skúškach z takýchto predmetoch je bežné, že je potrebné zdôvodniť správnosť programu. Dokonca je to často aj kľúčová časť, niekedy ani vôbec nepíšte program a skúšku robíte na papier. Viacerí z Vás v komentároch pomerne podrobne vysvetľovali, ČO program robí. No práve pri riešení tejto úlohy cez DFS bolo dôležité vysvetliť, PREČO to program robí. Prečo práve DFS a prečo nám to zaručí správny výsledok.

Načo sú takéto vysvetlenia, či dôkazy dobré? V programovaní poznáme situácie, kedy zistíme, že náš program nefunguje, ako má. Napr. že dostaneme WA od testovača, neskôr to však môže byť problém vo firme. Existujú dva hlavné spôsoby, ako tomu predísť:

1. Dôkaz správnosti – teda to, čo sme od Vás vyžadovali a pekne súvisel s riešením úlohy 3.

2. Testovanie – teda spustiť program na viacerých vstupoch a overiť si, či dal správny výsledok. Testovanie tejto úlohy nebolo zrovna najľahšie – vytvoriť dostatočne pestré vstupy nie je ľahká úloha. No mohli ste si aspoň overiť funkčnosť na príklade zo zadania – dostal som zopár programov, ktoré na vzorovom vstupe vypísali nesprávnu kostru.

Matematické predmety ako je aj UKTG, slúžia na to, aby ste sa naučili zamýšľať sa nad programami tým prvým spôsobom – vedieť odargumentovať, že môj program vo všeobecnosti funguje (a nie len na konkrétnych príkladoch, kde som si to overil). Práve preto, som Vám nedával k tejto úlohe testovať, ktorý by Vám rovno povedal, či je riešenie správne alebo nie. V štúdiu a hlavne v praxi je dôležité, aby ste to vedeli zhodnotiť sami.