Zadání příkladů pro semestrální práci 2019

Cílem seminární práce je aplikace teoretických znalostí z přednášky na konkrétní úlohy. Podstatu algoritmu totiž člověk nejlépe pochopí pokud jej sám aplikuje na řešení konkrétního problému.

1 Termíny

- **20.3.** končí možnost odevzdání podle loňského zadání pro opakující
- **8.3.** zveřejnění zadání pro 2021
- **15.3.** server `http://alva.nti.tul.cz:5000/` aktualizován pro nové zadání
- do **19.4.** zaslání rozboru úlohy (textová část)
- do **24.5.** odevzdání první funkční verze programu

2 Pravidla pro vypracování práce

- Studenti vytvoří řešitelské týmy po jednom nebo dvou lidech. Každý tým vypracuje řešení jedné úlohy dle vlastního výběru. Týmy nechť mi mailem nahlásí své složení a úlohu, kterou chtějí řešit. Aktuální stav budu průběžně zveřejňovat v dokumentu: přehled týmů

  Jednu úlohu může řešit maximálně 10 lidí v týmech celkem.

- Týmy vypracují textový rozbor problému. V rozboru formulujte úloha pomocí grafu (co jsou vrcholy, co hrany, ohodnocení a pod.) a popište použití vhodného grafového algoritmu pro danou úlohu. Dále popište jaké datové struktury použijete. Určete časovou a paměťovou složitost použitého algoritmu v nejhorším případě. Textovou část ve formátu PDF mi zašlete na e-mail: jan.brezina@tul.cz

- Týmy implementují vybraný algoritmus. Pro implementaci můžete použít jeden z jazyků: C, C++, C#, Java, Python, Pascal (viz. server) Program by měl být strukturovaný s výstižnými jmény identifikátorů, s okomentovanými funkcemi a hlavními datovými strukturami. Program má skutečně mít časovou a paměťovou složitost uvedenou v textovém rozboru. Program čte vstup ze standardního vstupu a zapisuje výstup na standardní výstup.

- Tým otestuje implementaci na testovacím serveru. Webové rozhraní váš program přeloží, spustí na testovací sadě vstupů a porovná s referenčními výsledky. Pokud jste se svou prací spokojeni, nebo potřebujete jinou zpětnou vazbu, vyberte "REQUEST CODE REVIEW". Kód dále zkontroluji a přes seerver okomentuji, nebo potvrdím jeho přijetí.

- Předchozí body plní tým jako celek, nicméně každý člen týmu musí rozumět jak teoretické části, tak programu a práce mu bude uznána, pokud v rámci zkoušky dokáže zodpovědět dotazy ohledně práce.

3 Zadání úloh

3.1 Železnice `kód: BIGLOKO1`

Železniční společnost VELKÁ MAŠINA plánuje propojit železnicí n měst, mezi nimiž zatím nikde železnice nevede. Byla vypracována studie, ve které bylo ohodnocena výstavba železnice mezi některými dvojicemi měst. Pro některé dvojice měst bylo vypracováno více variant, celkem bylo uvažováno m variant spojení mezi dvojicí některých měst. Studie také určila míru zátěže životního prostředí a obyvatelstva pro každou variantu a to stupni 1-5. Společnost VELKÁ MAŠINA potřebuje program, který pro zadané výsledky studie navrhne železniční síť, která propojí všechna města a zároveň bude mít nejmenší zátěž životního prostředí. V případě více takových sítí bude vybrána síť nejlevnější. Formulujte úlohu jako grafový problém hledání minimální kostry a navrhněte a implementujte algoritmus pro jeho řešení s časovou složitostí $O(m \log n)$ nebo lepší.

Vstupem je textový soubor, kde na prvním řádku jsou čísla n a m. Na dalších m řádcích jsou vždy data jedné varianty, tj. čtyři čísla. První dvě udávají indexy koncových měst úseku, tady jsou to čísla z množiny $\{0, \ldots, n-1\}$, další číslo udává cenu vybudování úseku (v miliónech korun) opět celé číslo a poslední číslo udává míru zátěže životního prostředí z množiny $1, 2, 3, 4, 5$, pět je nejhorší.

Na výstupu bude $n-1$ řádků s indexy variant železničních úseků (číslovaných od 0 do $m-1$) použitých výslednou síť. Jejich pořadí je dáno následovně:

- Uvažujte výslednou kostru jako zakořeněný strom v městě 0. Představte si že kostra je z korálků (vrcholy) a provázků stejné délky (hrany) a pověsíte jí za město 0.
- Pro každé další město j existuje v tomto zakořeněném stromu jediný předek $\pi[j]$. Od každého korálku vede nahoru jediný provázek.
- Na pozici $j-1$ výstupu bude index (0 až $m-1$) varianty použité pro spojení z j do předka $\pi[j]$, tj. index hrany $(j, \pi[j])$.

Příklad vstupu:

```
4 7
0 1 20  5
0 1 40  2
1 2 5   3
1 3 11  2
1 3 20  1
0 2 42  2
2 3 30  4
```

Očekávaný výstup:

```
1
5
4
```

3.2 Bludiště

kód: MINOS

Krétský král Mínos plánuje stavět bludiště pro nevlastního krvelačného syna Minotaura. Bludiště má mít pouze jedno patro, bude kutáno v podzemí z jediného vchodu, chodby bludiště budou pravoúhlé s celočíselnými délkami. Král chce, aby v bludišti existovala mezi každou dvojicí křižovatek právě jedna cesta. Při kopání je však skála různě tvrdá (tvrdost 0 až 2^{16}) a stavitel chce aby se dělníci co nejméně nadřeli, přitom tvrdost skály je známá jen na stěnách již vykopaných chodeb. Chceme tedy bludiště, které má celkovou tvrdost vykopané horniny co nejmenší a pokud takových existuje více, chceme aby se při vynášení vykopané horniny urazila co nejmenší vzdélenost (součet vzdáleností všech křižovatek od vstupu).

Napište program na generování bludiště na čtvercové síti $n \times m$ bodů. Chodby mohou vést pouze po hranách čtvercové sítě. Kopání bludiště probíhá vždy pouze po jedné hraně a v okamžiku dosažení jejího koncového vrcholu se ohodnotí tvrdost dosud neohodnocených hran, které z něj vycházejí a to v pořadí (východ-0, západ-1, sever-2, jih-3). Tvrdost α_i se generujte náhodně pomocí kongruenčního generátoru:

$$\alpha_{i+1} = \alpha_i * 1664525 + 1013904223 \bmod M$$

hodnota α_0, která je součástí vstupu, se použije pro ohodnocení první hrany. Hodnota M je 2^{16}. Vstup do bludiště uvažujte vždy v levém horním rohu z levé strany.

Vstupem programu je jeden řádek se třemi celými čisly n, m, α_0. Výstupem bude pro každou křižovatku (bod sítě) je kód typu křižovatky. Z jedné křižovatky existuje nebo neexistuje chodba do každého ze čtyř směrů (východ-0, západ-1, sever-2, jih-3), kód křižovatky je tedy 4-bitové číslo s uvedeným významem jednotlivých bitů. Jedna křižovatka je tedy jedna cifra v 16-kové soustavě (např. křižovatka sever, západ jih je reprezentována cifrou $2 + 4 + 8 = 14 = E$). Bludiště vypište na n řádků po m znacích.

Příklad vstupu:

```
3 3 3000
```

Výsledné bludiště, zaokrouhlené ceny hran v tisících:

```
X -  3 - X -  1 - X
|
56       64       65
|
X - 59 - X - 17 - X
                  |
61       58       46
                  |
X - 31 - X -  7 - X
```

Výstup:

```
932
53A
136
```

3.3 Dodávky elektřiny

kód: DELIVERY

Model elektrické rozvodné sítě je tvořen elektrárnami $0, \ldots n_e - 1$ a rozvodnami $n_e, \ldots, n_e + n_r - 1$. Tyto *přípojné body* jsou propojeny sítí čítající n_v vodičů. Jeden vodič vždy spojuje právě dva přípojné body a má definovaný maximální výkon, který jím může být přenášen (v obou směrech stejný). Žádné dvě elektrárny nejsou přímo propojeny a pro každou je znám její výkon. Elektrárny jsou napojeny libovolně na rozvodny. Rozvodny mohou být propojovány libovolně mezi sebou.

Za účelem výstavby nového výrobního závodu bylo vytipováno N míst. Pro každé místo $i = 1, \ldots, N$ bylo navrženo $0 < k_i \leq 3$ nezávislých elektrických přípojek napojených na přípojné body $r_{i,1}, \ldots r_{i,k_i}$. Navrhněte a implementujte algoritmus, který pro každou lokalitu určí maximální dostupný elektrický příkon.

Popis vstupu: Na prvním řádku vstupu jsou tři přirozená čísla udávající: počet elektráren n_e, počet rozvoden n_r a počet vodičů n_v. Na dalších n_e řádcích jsou výkony elekráren. Na dalších n_v řádcích je specifikace vodičů. Na jednom řádku jsou vždy tři přirozená čísla udávající postupně: index prvního, index druhého propojeného přípojného bodu a maximální přenášený výkon. Výkony elektráren a kapacity vodičů jsou přirozená čísla menší než 1000. Následuje počet lokalit N na samostatném řádku. Na každém z následujících N řádcích je seznam přípojných bodů pro navržené elektrické přípojky (maximálně 3). Na výstupu programu bude N řádků, na každém maximální dostupný výkon pro odpovídající lokalitu.

Formulujte úlohu grafově jako úlohu pro tak, aby šla pro jednu lokalitu řešit jako problém sítě s jedním zdrojem a jedním cílem. Napište program pro nalezení optimálního toku sítí. Použijte Edmons-Karpův algoritmus (BFS pro hledání zlepšující cesty), nebo algoritmus s lepší složitostí.

Příklad vstupu:

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

```
3
2 3
```

Očekávaný výstup:

```
4
7
8
```

3.4 Nejspolehlivější cesta

kód: RELCONN

Máme rozlehlou počítačovou síť, která je realizována rádiovým spojením. Rádiové spojení může být nejrůznějším způsobem rušeno a tudíž není moc spolehlivé. Síť tvoří uzly a jejich rádiová spojení. Uzly jsou očíslované 0 až $n-1$ a jedno spojení je dáno: číslem i vysílajícího uzlu, číslem j přijímajícího uzlu a pravděpodobností $0 < p_{i,j} < 1$ přijmutí správného paketu. Spojení mezi dvěma uzly sítě je vždy symetrické. Najděte v zadané síti nejspolehlivější cestu z vrcholu s do vrcholu t. Nejspolehlivější cesta je ta s nejmenší pravděpodobností chyby.

Na prvním řádku vstupu je počet uzlů n a celkový počet spojení mezi nimi, m. Na dalších m řádkcích je pro každé spojení číslo vysílajícího uzlu, číslo přijímajícího uzlu (číslováno od 0) a pravděpodobnost p (např. 0.8931). Na řádku $m + 2$ je počet testovacích dotazů N a na dalších N řádcích jsou data pro jednotlivé dotazy. Jeden dotaz je dvojice uzlů, mezi kterými chceme najít cestu, první je index počátečního a druhý index koncového uzlu.

Na výstupu bude N řádků, pro každý dotaz jeden. Výsledek jednoho dotazu je posloupnost indexů uzlů podél nalezené nejspolehlivější cesty včetně počátečního a koncového ze zadání dotazu. V případě, že žádná cesta (byť s minimální spolehlivostí) pro dotaz neexistuje, vypíše se pouze index počátečního uzlu ze zadání dotazu.

Příklad vstupu:

```
3 3
0 1 0.8
0 2 0.5
1 2 0.7
1
0 2
```

Očekávaný výstup:

```
0 1 2
```

3.5 Rozvrh

kód: SCHED

Řešte zjednodušený problém sestavování rozvrhu. Jsou dány:

- množina studentů $S = \{0, \ldots, n_S - 1\}$,
- množina předmětů $P = \{0, \ldots, n_P - 1\}$,
- množina učitelů $U = \{0, \ldots, n_U - 1\}$,
- množina výukových bloků $B = \{0, \ldots, n_B - 1\}$.

Každý student $i \in S$ má definovánu množinu zapsaných předmětů $p_i \subset P$. Každý předmět $j \in P$ má definovaného učitele $u_j \in U$. Předpokládáme neomezenou zásobu učeben. Cílem je přiřadit předmětům vyučovací bloky tak, aby předměty které mají průnik studentů nebo učitelů neměly stejný blok. Úkoly:

1. Formulujte úlohu jako problém barvení grafu. Co jsou vrcholy, hrany, barvy?
2. Vyberte si jeden z heuristických barvících algoritmů a popište jeho průběh při aplikaci na tento konkrétní problém.
3. Zvojený algoritmus implementujte.

Vstupem je textový soubor. Na prvním řádku jsou čísla n_S, n_P, n_U, n_B. Na dalších n_S řádcích je pro každého studenta i seznam čísel jeho předmětů p_i. Na dalších n_P řádcích je pro každý předmět j číslo jeho učitele u_j. Výstup programu má n_P řádků na řádku j je pro předmět j číslo jeho bloku. Bloky jsou očíslovány tak, aby při ponechání pouze řádků s prvním výskytem bloku tvořily uspořádanou posloupnost.

Příklad vstupu:

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

Očekávaný výstup:

```
0
1
0
2
1
2
```