Zadání příkladů pro semestrální práci 2019

Cílem seminární práce je aplikace teoretických znalostí z přednášky na konkrétní úlohy. Podstatu algoritmu totiž člověk nejlépe pochopí až pokud jej sám implementuje, nebo modifikuje při řešení konkrétního problému.

1 Termíny

- **20.3.** končí možnost odevzdání podle loňského zadání pro opakující
- **17.3.** zveřejnění zadání pro 2020
- **27.3.** server `https://flowdb.nti.tul.cz/secure` aktualizován pro nové zadání
- do **21.4.** zaslání rozboru úlohy (textová část)
- do **22.5.** 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 Přelévání nádob `MINSPILL`

Máte k dispozici n nádob s celočíselnými kladnými objemy $v_i < 100$, $i \in 1, \ldots n$ v litrech. Největší nádoba o objemu V je plná a cílem je odměřit daný počet litrů $w \leq V$ v co nejkratším čase, přičemž přelévání probíhá vždy rychlostí 1 litr za sekundu. Chceme získat postup optimálního přelévání pro všechny možné cílové objemy $w \leq V$. Jedno přelití končí buď plnou cílovou nádobou nebo prázdnou vylévanou nádobou a nedochází ke ztrátám.

Formulujte problém jako grafovou úlohu. Co budou vrcholy, co hrany? O jaký typ grafu se jedná? Navrhněte vhodné datové struktury umožňující běh Dijkstrova algortimu pro graf, který není dán na vstupu, ale vzniká dynamicky. Napište program používající vhodnou implementaci Dijkstrova algoritmu pro nalezení optimálních přelévacích postupů. Pokuste se určit složitost vaší implementace v nejhorším případě:

1. vzhledem k počtu vrcholů grafu
2. vzhledem k počtu nádob.

Vstup je textový soubor, kde na prvním řádku je počet nádob, na dalších n řádcích jsou objemy nádob. Na výstupu bude V řádků. Na jednom výstupním řádku je postupně: cílový objem w, doba přelití (počet přelitých litrů) a počet přelití. Pokud objem w nelze naměřit, vypíše se pouze w následované koncem řádku.

Příklad vstupu:

```
3
2
4
6
```

Očekávaný výstup:

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

3.2 IDOS

`kód IDOS`

Uvažujte jednoduchý systém MHD s n zastávkami a jízdními řády, které se každý den opakují. Jednotlivé spoje jsou dány počáteční stanicí, cílovou stanicí, časem odjezdu (v minutách od začátku dne) a dobou cesty v minutách. Navrhněte algoritmus a napište program, který pro zadaný čas, stratovní a cílovou stanici nalezne nejrychlejší spojení. Předpokládejte, že žadné spoje ani dotazy na spojení nejdou přes půlnoc.

Na prvním řádku vstupu je počet stanic n a počet spojů m. Na dalších m řádkcích je pro každý spoj startovní a cílová stanice, čas odjezdu a doba cesty. Na řádku $m + 2$ je počet testovacích dotazů N a na dalších N řádcích jsou data pro jednotlivé dotazy. Jeden dotaz se skládá ze tří čísel: číslo startovní stanice, číslo cílové stanice a čas (v minutách od začátku dne) od kdy se má spojení hledat.

Na výstupu bude N řádků, pro každý dotaz jeden. Výsledek jednoho dotazu je posloupnost použitých spojů. Jeden spoj je dán dvojicí: (číslo stanice, čas odjezdu). Dvojice jsou uzavřeny v závorce a odděleny mezerou. Po použitých spojích následuje ještě jedna dvojice obsahující index cílové stanice a čas příjezdu do ní. Pokud pro daný dotaz spojení neexistuje, vypíše se prázdná závorka '()'.

Příklad vstupu:

```
4 6
0 1 10 30
0 1 70 30
0 2 130 10
1 2 60 30
1 3 100 60
3 1 30 20
2
0 2 0
0 3 20
```

Očekávaný výstup:

```
(0 10) (1 60) (2 90)
(0 70) (1 100) (3 160)
```

3.3 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. Pro každé další město j existuje jediný předek k. Na pozici $j-1$ v seznamu bude index 0 až $m-1$ varianty použité pro spojení z j do předka k.

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
```

3.4 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
```

3.5 Sudoku

kód: SUDOGOB

Ve vaší knize se Sudoku problémy řádil šotek, přidal nebo vymazal některá čísla ze zadání. Máte napsat program, který zjistí kolik řešení pro dané zadání existuje a pokud existuje právě jedno, tak ho vypíše.

Problém řešení sudoku formulujte jako grafovou úlohu (barvení grafu). Navrhněte a implementujte algoritmus, který pro zadané sudoku 9x9 nalezne jeho řešení. Použijte některý aproximativní algoritmus a doplňte ho o backtracking v případě, že by aproximativní řešení vyžadovalo více než 9 barev.

Na prvním řádku vstupu je počet zadání N k testování. Následuje N tabulek oddělených prázdným řádkem. Jedna tabulka je 9 řádků po devíti cifrách oddělených mezerou. Cifra 0 označuje prázné pole. Výstupem je pro každou vstupní tabulku počet řešení, pokud má úloha jediné řešení, pak po počtu řešení následuje výpis unikátního řešení ve stejném formátu jako je vstup. Pokud úloha nemá řešení vypíše se počet řešení 0.

Příklad vstupu:

```
3
3 0 6 0 0 2 5 0 0
0 0 0 0 3 8 0 0 0
7 0 8 0 1 6 0 9 0
0 0 7 0 0 3 8 6 0
8 2 0 0 7 0 0 4 5
0 6 3 1 0 0 9 0 0
0 7 0 3 5 0 6 0 2
0 0 0 8 2 0 0 0 0
0 0 2 9 0 0 7 0 4

3 0 6 0 0 2 5 0 3
0 0 0 0 3 8 0 0 0
7 0 8 0 1 6 0 9 0
0 0 7 0 0 3 8 6 0
8 2 0 0 7 0 0 4 5
0 6 3 1 0 0 9 0 0
0 7 0 3 5 0 6 0 2
0 0 0 8 2 0 0 0 0
0 0 2 9 0 0 7 0 4

3 0 6 0 0 2 0 0 0
0 0 0 0 3 8 0 0 0
7 0 8 0 1 6 0 9 0
0 0 7 0 0 3 8 6 0
8 2 0 0 7 0 0 4 5
0 6 3 1 0 0 9 0 0
0 7 0 3 5 0 6 0 2
0 0 0 8 2 0 0 0 0
0 0 2 9 0 0 7 0 4
```

Výstup:

```
1
3 1 6 4 9 2 5 7 8
2 9 5 7 3 8 4 1 6
7 4 8 5 1 6 2 9 3
9 5 7 2 4 3 8 6 1
8 2 1 6 7 9 3 4 5
4 6 3 1 8 5 9 2 7
1 7 9 3 5 4 6 8 2
6 3 4 8 2 7 1 5 9
5 8 2 9 6 1 7 3 4
0
2
```

3.6 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
```