

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

- **8.3.** končí možnost odevzdání podle loňského zadání pro opakující
- **12.3.** zveřejnění zadání pro 2019
- **9. 4.** server <https://flowdb.nti.tul.cz/secure> aktualizován pro nové zadání
- do **23.4.** 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 nechtě mi mailem nahlásí své složení a úlohu, kterou chtějí řešit. Aktuální stav budu průběžně zveřejňovat na adrese:

https://docs.google.com/spreadsheets/d/17Zw4NxqiF--IbPwC9d_FMQeccwS9oCbKWx75qsR_z4o/edit?usp=sharing

Jednu úlohu může řešit maximálně 10 lidí v týmech celkem.

- Týmy vypracují textový rozbor problému. V rozboru formulujte úlohu 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. Informujte mě e-mailem o úspěšném odevzdání nebo nezvladatelných problémech s odevzdáním. Každý kód dále kontroluji osobně a e-mailem potvrdím jeho přijetí, nebo případné výhrady.
- 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 $v_i < 100$, $i \in 1, \dots, n$ 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 algoritmu 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 Web stránky

kód: WEBISL

Danou množinu webových stránek chceme rozdělit na co největší disjunktní podmnožiny $1, \dots, k$ tak, že v každé podmnožině se lze proklikat z libovolné stránky na libovolnou jinou.

Formulujte problém jako grafovou úlohu. Co budou vrcholy, co hrany? O jaký typ grafu se jedná? Navrhněte vhodný algoritmus a určete složitost vaší implementace vzhledem k počtu stránek.

Vstup je textový soubor, kde na prvním řádku je počet stránek n a počet odkazů m , na dalších m řádcích jsou jednotlivé odkazy jako dvojice index stránky, ze které odkaz odchází, a index stránky, na kterou vede. Výstup má n řádek, na řádce i je číslo komponenty, ke které patří stránka i . Číslo komponenty je minimální index stránky v komponentě.

Příklad vstupu:

```
3 3
1 2
2 1
1 2
```

Očekávaný výstup:

```
0
1
1
```

3.3 Železnice

kód: BIGLOK01

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 navrhnete 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, tedy jsou to čísla z množiny $\{0, \dots, n-1\}$, 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 sítí. 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 $S = \{0, \dots, n_S - 1\}$,
- množina předmětů $P = \{0, \dots, n_P - 1\}$,
- množina učitelů $U = \{0, \dots, n_U - 1\}$,
- množina výukových bloků $B = \{0, \dots, n_B - 1\}$.

Každý student $i \in S$ má definovanou 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 barvicích 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 Segmentace

kód: SEGMENT

Pro daný obrázek $n \times n$ pixelů ve stupních šedi $0 \dots 255$ je třeba odlišit pixely vyobrazeného objektu O od kontrastního pozadí B (problém segmentace obrazu). Přitom není známo zda jde o tmavý objekt na světlém pozadí nebo naopak, ani není znám nějaký odstup intenzit objektu a pozadí. Předpokládáme však, že maximální rozdíl intenzit sousední pixelů a a b v rámci objektu ($a, b \in O$) i v rámci pozadí ($a, b \in B$) je alespoň o 2 stupně šedi menší než minimální rozdíl intenzit sousedních pixelů z nichž jeden je z objektu a druhý z pozadí ($a \in O$, $b \in B$). Předpokládáme, že po obvodu obrázku jsou pouze pixely patřící k pozadí a detekovaný objekt nemá díry.

Navrhněte a popište algoritmus založený na hledání minimální kostry, kde ohodnocení hran je dáno rozdílem intenzit sousedících pixelů. Formulujte problém jako grafovou úlohu. Co budou vrcholy, co hrany? Určete složitost vaší implementace vzhledem k počtu pixelů obrázku.

Vstup je textový soubor, kde na prvním řádku je velikost obrázku n a na dalších n řádcích je vždy n čísel udávajících intenzity v obrázku. Intenzity jsou čísla od 0 do 255. Výstup je ve stejném formátu jako vstup, jen místo intenzit je uvedena 0 pro pixel pozadí a 1 pro pixel objektu.

Příklad vstupu:

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

Očekávaný výstup:

```

0 0 0 0 0
0 0 1 1 0
0 1 1 1 0
0 1 1 0 0
0 0 0 0 0

```

3.6 Dodávky elektřiny

kód: DELIVERY

Model elektrické rozvodné sítě je tvořen elektrárnami $0, \dots, n_e - 1$, koncovými přípojkami $n_e, \dots, n_e + n_s - 1$ a rozvodnami $n_e + n_s, \dots, n_e + n_s + n_r - 1$. Tyto objekty jsou propojeny sítí elektrických vodičů n_v . Jeden vodič vždy spojuje právě dva objekty 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. Za účelem výstavby nového výrobního závodu bylo vytipováno N míst, každé s několika nezávislými elektrickými přípojkami. Popište a implementujte algoritmus, který pro každou lokalitu určí maximální dostupný elektrický příkon.

Popis vstupu: Na prvním řádku vstupu jsou čtyři přirozená čísla udávající: počet elektráren n_e , počet koncových přípojek n_s , počet rozvodů n_r a počet vodičů n_v . Na dalších n_e řádcích jsou výkony elektráren. Na dalších n_v řádcích je specifikace vodičů. Na jednom řádku jsou tři přirozená čísla udávající postupně: index prvního, index druhého propojeného objektu 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 koncových přípojek jedné lokality (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 sítě. Použijte BFS pro hledání zlepšující cesty, nebo algoritmus s lepší složitostí.

Příklad vstupu:

```

2 2 1 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

```