Zadání projektu do předmětu **Algoritmy II**
zimní semestr 2024/2025
prezenční a kombinované studium

Historie modifikací

30. října 2024 První verze

Obsah

Deadline

Vypracované řešení projektu je nutno odevzdat do

- prezenční forma studia – **8. prosince 2024 23:59** a
- kombinovaná forma studia – **15. prosince 2024 23:59**.

Obecné pokyny

- Každý student či studentka má přiděleno jedno zadání. Rozdělení zadání mezi studenty je součástí tohoto zadání.
- S případnými dotazy kontaktujte svého cvičícího (studenti prezenční formy studia) či tutora (studenti kombinované formy studia).
- Termíny obhajob budou vypsány v systému Edison.
- Deadline je konečný a nebude dále posunován. Na projekty odevzdané po tomto datu nebude brán zřetel. Projekt lze pochopitelně odevzdat a domluvit si termín obhajoby i dříve.
- Každý cvičící (tutor) Vám sdělí, jakým způsobem budete projekt odevzdávat – pomocí git repozitáře, emailem, uložením na sdílené úložiště, pomocí ftp a tak podobně. Obecně platí, že se odevzdávají soubory se zdrojovým kódem, hlavičkové soubory, soubory s projektem atd., jinak řečeno vše, co je potřebné pro bezproblémovou kompilaci odevzdaného projektu.
- Součástí zdrojových kódů Vašeho programu bude programátorská dokumentace ve formě dokumentačních komentářů, zpracovatelných programem Doxygen, viz `www.doxygen.org`. Vygenerovanou dokumentaci není nutné odevzdávat. Postačuje, pokud Vámi odevzdaný archiv bude obsahovat konfigurační soubor `doxyfile`, případné adresáře pro vygenerovanou dokumentaci, vkládané obrázky atd.
- Ačkoliv se zadání mohou jevit na první pohled složitá, nepropadejte panice. Vyřešení kteréhokoliv zadání by nemělo zkušenému programátorovi zabrat více než „jeden večer“. Studentům prvního a druhé ročníku to zabere asi více času, ale v žádném případě by řešení nemělo trvat „desítky a desítky“ člověkohodin práce. Stejně tak co se týče délky vytvořeného kódu. Pokud jste už napsali „tisíce“ řádků kódu a stále není konec v dohledu, je to špatně. Takový zdrojový kód rovnou zahoďte. Klíčem k řešení je tužka, papír a hlava. Zkuste si řešení nejdříve promyslet, kreslit si u toho různá schémata, náčrtky datových struktur, volání funkcí a tak dále. Projděte si literaturu. A až se dostaví „aha moment“ tak začněte psát kód.

Hodnocení projektů

Kritéria hodnocení řešení projektů jsou tato:

1. **Správnost řešení** Správnost řešení je podmínka nezbytná. Aplikace, která nebude poskytovat správné výsledky, bude hodnocena automaticky 0 body bez ohledu na další kritéria. Ke každému zadání jsou k dispozici testovací data a výsledky, lze si tedy ověřit správnost řešení.
2. **Volba vhodných datových struktur** Toto kritérium hodnotí, v závislosti na konkrétním zadání, volbu vhodné datové struktury a algoritmů pro manipulaci se zvolenou datovou strukturou.
3. **Dekompozice problému na menší celky**
    - V předmětu Algoritmy I je požadována procedurální dekompozice, čili dekompozice řešení do funkcí. Využití objektově orientovaného programování je v tomto předmětu dobrovolné.
    - V předmětu Algoritmy II je požadován objektový návrh řešení a tomu odpovídající implementace.

4. **Způsob implementace** Toto kritérium hodnotí oddělení deklarace a definice funkcí nebo tříd do h a cpp souborů, využívání konstant místo přímo zapsaných hodnot, využívání parametrů funkcí a výsledků funkcí místo využívání side efektu založeném na globálních proměnných. Dále sem patří i úroveň zápisu zdrojového kódu, například odsazování vnořených konstrukcí, vhodné pojmenování proměnných, funkcí, tříd, dodržování zvolené konvence pojmenování[1] proměnných, funkcí a tříd, dodržování zvolené konvence psaní bloků kódu[2] a tak dále.

5. **Efektivita implementovaného algoritmu** Smyslem tohoto kritéria není nutit vás k implementaci nejlepšího známého algoritmu tím nejlepším možným způsobem. Tímto kritériem si vyučující ponechávají prostor pro případné snížení bodového hodnocení za použití algoritmu zcela nevhodného, nesmyslného, zmateného. *Příklad*: Součástí řešení projektu je třídění pole či vektoru s n prvky. Pokud použijete algoritmus se složitostí $O(n \log_2 n)$ je vše v pořádku. Pokud použijete některý z jednodušších algoritmů se složitostí $O(n^2)$, asi vás vyučující při obhajobě upozorní, že to nebyla dobrá volba, ale stále je to v pořádku. Za zcela nesmyslný algoritmus je v tomto případě považován algoritmus se složitostí větší než $O(n^2)$, protože takový algoritmus z podstaty věci dělá zbytečnou práci.

6. **Dokumentace k projektu** Ke každé funkci, třídě, atributu třídy musí existovat alespoň krátký dokumentační komentář ve formátu zpracovatelném programem Doxygen. Pro zápis dokumentačních komentářů lze využít libovolný z formátů, které Doxygen podporuje. Vygenerovanou dokumentaci není nutné odevzdávat, ale je nutné odevzdat konfigurační soubor `doxyfile`, aby bylo možné dokumentaci bez problémů vygenerovat.

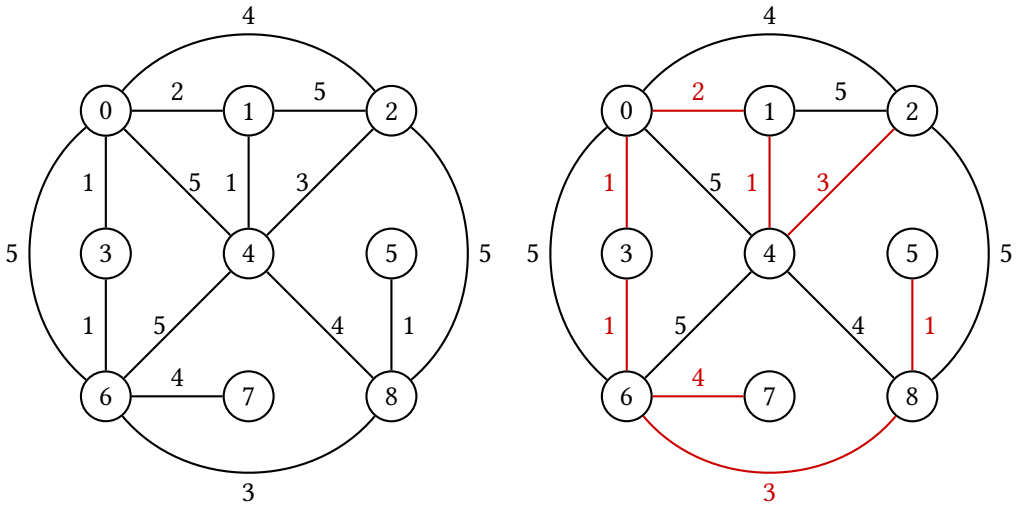
   K programu Doxygen existuje rozsáhlá dokumentace volně dostupná na URL `https://www.doxygen.nl/manual/index.html`. Pro dokumentaci řešení semestrálního projektu je vhodné si prostudovat následující části dokumentace:

   - „Getting started“, `https://www.doxygen.nl/manual/starting.html`,
   - „Documenting the code“, `https://www.doxygen.nl/manual/docblocks.html`
   - „Doxygen usage“, `https://www.doxygen.nl/manual/doxygen_usage.html`
   - „Doxywizard usage“, `https://www.doxygen.nl/manual/doxywizard_usage.html`

     Program Doxywizard slouží k pohodlnému vygenerování konfiguračního souboru `doxyfile`, který tak není nutné vytvářet ručně.

7. **Citace zdrojů** Žádný program nevzniká úplně z ničeho, ani řešení semestrálních projektů. K řešení projektů můžete používat odbornou literaturu, učebnice, příklady zdrojových kódů z ostatních předmětů, internetové zdroje. V tom případě je nutné uvést, že „Tento algoritmus jsem převzal z…“, „Tuto část kódu jsem převzal z…“. Tyto případné citace musí umístit do dokumentačních komentářů. Asi není nutné citovat Levitinovu knihu, že jsem si tam „přečetl něco o průchodu grafem do šířky“ nebo, že „toto jsme řešili na cvičení“. Ostatní zdroje by se však citovat měly.

[1] Typicky camelCase, PascalCase, méně vhodná je už například maďarská notace.
[2] Typicky – složená levá závorka za příkazem nebo na novém řádku.



(a) Ukázkový graf G

(b) minimální kostra K_0

Obrázek 1: Ukázkový graf a jeho minimální kostra

1 k-tá minimální kostra grafu

Problém

V tomto zadání budete hledat specifické kostry neorientovaného váženého grafu G. Nalezení minimální kostry K grafu G je dobře známý problém, jehož řešení není nijak komplikované. Vaším úkolem ale bude najít posloupnost minimálních koster grafu $K_0 = K, K_1, K_2, \ldots, K_m$, kde K_1 je druhá minimální kostra grafu, K_2 je třetí minimální kostra grafu a tak dále, až K_m je maximální kostra grafu. Označme $w(K_i)$ součet vah všech hran v kostře K_i, potom pro hodnoty $w(K_0), \ldots, w(K_m)$ musí platit

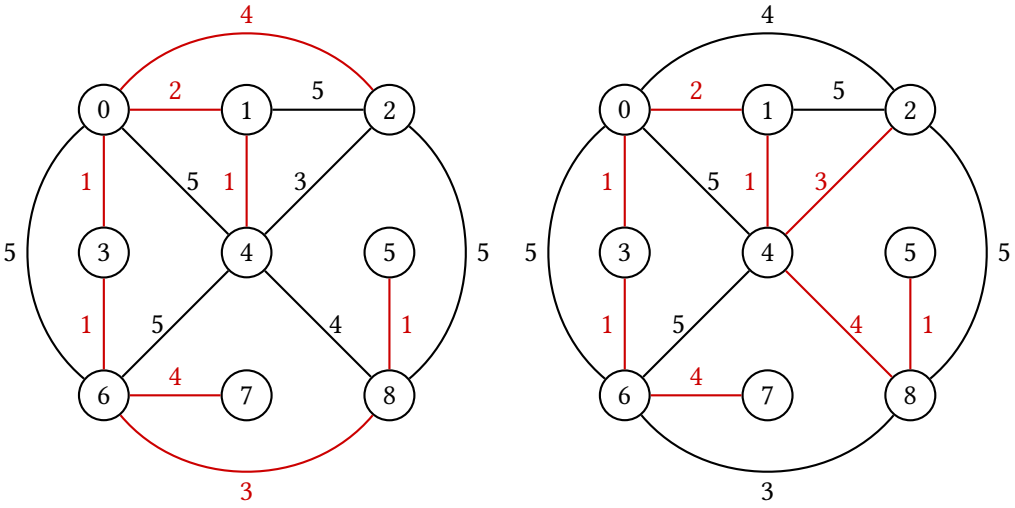
$$w(K_0) \leq w(K_1) \leq \cdots \leq w(K_m).$$

Ukázkový příklad

Ukázkový graf G a jeho minimální kostra K_0 je zobrazena na obrázku 1, součet vah hran minimální kostry je $w(K_0) = 16$. Druhé minimální kostra má dvě varianty, viz obrázek 2. V první variantě, viz obrázek 2a, byla hrana $(2, 4)$ s vahou 3 nahrazena hranou $(0, 2)$ s vahou 4. Ve druhé variantě, viz obrázek 2b, byla hrana $(6, 8)$ s vahou 3 nahrazena hranou $(4, 8)$ s vahou 4. V obou případech je $w(K_1) = w(K_0) - 3 + 4 = 17$. Třetí minimální kostra je už jen jedna, viz obrázek 3, a vznikne nahrazením zbývající hrany $(6, 8)$ resp. $(2, 4)$ s vahou 3 hranou $(4, 8)$ resp. $(0, 2)$. Výsledný součet vah je $w(K_1) = w(K_1) - 3 + 4 = 18$. Maximální kostra grafu je zobrazena na obrázku 4, kde $w(K_m) = 30$. Maximální kostra existuje v několika variantách, které se liší napojením vrcholů 3 a 4 ke zbávající části kostry.

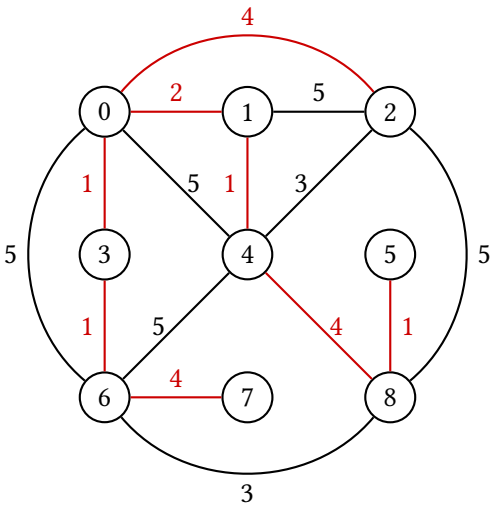
Poznámky

- Vstupem do vašeho programu bude textový soubor obsahující matici sousednosti grafu. Každý vrchol grafu je označen, celým číslem i, kde $0 \leq i < n$. Graf bude uložen v textovém souboru v následujícím formátu. Na prvním řádku bude číslo n. Na dalších n řádcích bude uložena vždy n-tice celých čísel, větších než nula, představujících váhy hran k sousedům i-tého vrcholu. Pokud je váha hrany rovna 0, hrana neexistuje. Graf z obrázku 1 tak bude uložen ve tvaru:

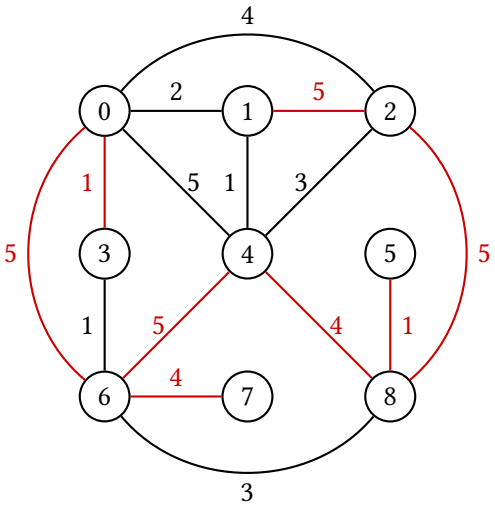


(a) první varianta (b) druhá varianta

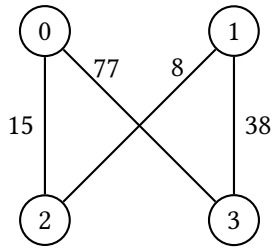
Obrázek 2: Dvě varianty druhé minimální kostry grafu G



Obrázek 3: Třetí minimální kostra grafu G



Obrázek 4: Maximální kostra grafu G, jedna z variant



Obrázek 5: Ukázková logistická síť

```
9
0 2 4 1 5 0 5 0 0
2 0 5 0 1 0 0 0 0
4 5 0 0 3 0 0 0 5
1 0 0 0 0 0 1 0 0
5 1 3 0 0 0 5 0 4
0 0 0 0 0 0 0 0 1
5 0 0 1 5 0 0 4 3
0 0 0 0 0 0 4 0 0
0 0 5 0 4 1 3 0 0
```

- Nalezené minimální kostry grafu vypište ve vhodné formě na standardní výstup.
- Jak asi víte, algoritmy oba žravé algoritmy vytváří minimální kostru grafu tak, že do ní postupně přidávají hrany s nejmenšími váhami tak, aby propojily všechny vrcholy grafu a současně nevytvořily cyklus. Druhá minimální kostra grafu je tedy taková kostra grafu, kterou by algoritmus vytvořil, kdyby se jednou „spletl“ a přidal nesprávnou hranu. Tím by vznikla kostra grafu, která bude mít větší součet vah než ta skutečně minimální. Ale současně se musí algoritmus „splést“ tak, aby nárůst součtu vah hran ve druhé minimální kostře byl co možná nejmenší.

2 Minimální cena přepravy zboží

Problém

V tomto zadání máte za úkol optimalizovat cenu přepravy zboží v logistické síti mezi městy, která je reprezentována jako neorientovaný vážený graf. Města představují vrcholy grafu, cesty jsou reprezentovány neorientovanými váženými hranami, kde váha představuje cenu přepravy zboží mezi těmito městy. Vaším úkolem je nalézt trasu s minimální cenou přepravy zboží z města A do města B a to:

- v původní logistické síti a
- v pozměněné logistické síti – na jednu libovolnou hranu grafu můžete aplikovat 50% slevu na cenu přepravy.

Ukázkový příklad

Uvažujme logistickou síť jejíž graf je na obrázku 5. Pro přepravu zboží z města $A = 0$ do města $B = 3$ je minimální cena přepravy beze slevy rovna 61 po trase 0-2-1-3, se slevou je pak minimální cena přepray 38.5 a trasa vede přímo z města 0 do města 3.

Poznámky

- Vstupem do vašeho programu bude textový soubor obsahující specifikaci grafu logistické sítě. Každý vrchol grafu je označen celým číslem i, kde $0 \leq i < n$. Graf bude uložen v textovém souboru v následujícím formátu. Na prvním řádku bude číslo n, na dalších n řádcích bude uložena matice sousednosti

$$\begin{pmatrix} w_{0,0} & \cdots & w_{0,n-1} \\ \vdots & & \vdots \\ w_{n-1,0} & \cdots & w_{n-1,n-1} \end{pmatrix},$$

  kde pro všechny prvky $w_{i,j}$ platí:

  - $w_{i,j} = 0$ pro $i = j$, tj. cena přepravy z města do sebe sama je nulová (nejsou zde žádné náklady),
  - $w_{i,j} = -1$, pokud hrana mezi městy i a j neexistuje a
  - $w_{i,j} > 0$ udává cenu přepravy po hraně z města i do města j.

  Za maticí sousednosti bude na samostatných řádcích uložena dvojice kladných celých čísel A a B, $0 \leq A, B < n$, které označují počáteční a koncové město přepravy zboží. Zadání z ukázkového příkladu bude tedy uloženo takto:

```
4
0 -1 15 77
-1 0 8 38
15 8 0 -1
77 38 -1 0
0
3
```

- Přestože jsou váhy hran ve vstupní souboru celočíselné, nemusí být vždy sudé. Jak je ostatně i vidět z ukázkového přkladu.
- Na standardní výstup vypište cenu přepravy beze slevy a se slevou. Dále na standardní výstup vypište i města tvořící obě trasy (beze slevy a se slevou). Pokud trasa mezi městy A a B neexistuje, vypište na standardní výstup -1.

3 Násobení posloupnosti matic

V tomto zadání máte zadánu posloupnost $n + 1$ přirozených čísel $D = d_0, d_1, \ldots, d_n$, kde všechna čísla $d_i > 0$. Dvojice čísel z posloupnosti D budou, v našem zadání, představovat rozměry matic $A_0, A_1, \ldots, A_{n-1}$ a to tak, že matice A_i má rozměry $d_i \times d_{i+1}$. Dále předpokládejme, že chceme vypočítat součin všech matic

$$M = A_0 \cdot A_1 \cdot A_2 \cdots A_{n-1}.$$

Přímočarý postup výpočtu součinu znamená, že vynásobíme matice A_0 a A_1. Tento mezivýsledek vynásobíme maticí A_2 a tak dále, dokud nevynásobíme všechny matice. Z lineární algebry ale víme, že násobení matic je asociativní operace. Do součinu si můžeme doplnit pomyslné závorky a díky tomu násobit matice v jiném pořadí než v jakém jsou v součinu uvedeny. A je asi zřejmé, že pro různá uzávorkování provedeme různý počet aritmetických operací.

Vaším úkolem je implementovat algoritmus, který najde optimální pořadí násobení matic v tomto součinu. Optimálním pořadím rozumíme takové pořadí násobení matic, při kterém se provede nejmenší možný počet operací násobení prvků matic, tedy čísel. Předpokládejte, že násobení dvou matic probíhá pomocí algoritmu hrubou silou.

Matice	Rozměry
A_0	7×5
A_1	5×3
A_2	3×6
A_3	6×5

(a) rozměry ukázkových matic

Uzávorkování	Počet násobení
$((A_0 \cdot A_1) \cdot A_2) \cdot A_3$	441
$(A_0 \cdot A_1) \cdot (A_2 \cdot A_3)$	300
$(A_0 \cdot (A_1 \cdot A_2)) \cdot A_3$	510
$A_0 \cdot ((A_1 \cdot A_2) \cdot A_3)$	415
$A_0 \cdot (A_1 \cdot (A_2 \cdot A_3))$	340

(b) počty násobení pro různá uzávorkování

Tabulka 1: Násobení matic – ukázkové zadání

Příklad

V ukázce máme dáno $n = 4$ a posloupnost $D = 7, 5, 3, 6, 5$. Matice $A_0, \ldots A_3$ budou mít rozměry uvedené v tabulce 1a. Ukažme si postup výpočtu součinu matic pro dvě možná uzávorkování:

- Nejprve součin vypočítáme pomocí „přímočarého“ postupu a tomu odpovídá uzávorkování

  $$((A_0 \cdot A_1) \cdot A_2) \cdot A_3.$$

  Postup násobení je následující:

Součin matic	Rozměry násobených matic	Počet násobení
$B_0 = A_0 \cdot A_1$	$(7 \times 5) \cdot (5 \times 3) = (7 \times 3)$	$7 \times 5 \times 3 = 105$
$B_1 = B_0 \cdot A_2$	$(7 \times 3) \cdot (3 \times 6) = (7 \times 6)$	$7 \times 3 \times 6 = 126$
$M = B_1 \cdot A_3$	$(7 \times 6) \cdot (6 \times 5) = (7 \times 5)$	$7 \times 3 \times 6 = 126$
Celkem násobení		441

- Dále si ukážeme výpočet součinu

  $$(A_0 \cdot A_1) \cdot (A_2 \cdot A_3)$$

  Postup násobení je v tomto případě následující:

Součin matic	Rozměry násobených matic	Počet násobení
$B_0 = A_0 \cdot A_1$	$(7 \times 5) \cdot (5 \times 3) = (7 \times 3)$	$7 \times 5 \times 3 = 105$
$B_1 = A_2 \cdot A_3$	$(3 \times 6) \cdot (6 \times 5) = (3 \times 5)$	$3 \times 6 \times 5 = 90$
$M = B_0 \cdot B_1$	$(7 \times 3) \cdot (3 \times 5) = (7 \times 5)$	$7 \times 3 \times 5 = 105$
Celkem násobení		300

Počty násobení pro všechna možná uzávorkování jsou uvedeny v tabulce 1b. Z tabulky je zřejmé, že optimální pořadí násobení matic je dáno uzávorkováním $(A_0 \cdot A_1) \cdot (A_2 \cdot A_3)$.

Poznámky

- V tomto zadání jde opravdu jen o určení *počtu násobení* prvků matic. Matice samotné není nutné nijak zadávat, načítat ze souboru, generovat atd. Stejně tak není nutné matice doopravdy násobit.

- Vstupem do vašeho programu bude textový soubor, který na každém řádku obsahuje jednu posloupnost čísel D. Čísla jsou na řádku oddělena mezerami. Řádek s ukázkovým zadáním bude mít podobu

  ```
  7 5 3 6 5
  ```

- Na standardní výstup vypište nalezené optimální uzávorkování součinu matic a počet násobení prvků matic odpovídající nalezenému uzávorkování. Pokud existuje několik uzávorkování se shodným počtem násobení, stačí vypsat jen jedno uzávorkování, tj. stačí vypsat jedno řešení.

4 Cesty jeskynním systémem

Problém

V tomto zadání budete hledat cesty v jeskynním labyrintu. Labyrint se skládá z jeskyní spojených navzájem chodbami. Jeskyně mohou být dvou typů *velké* a *malé*. Velké jeskyně jsou označeny textovými řetězci složenými z velkých písmen, malé jeskyně pak řetězci z malých písmen. Vaším úkolem je na základě mapy labyrintu, kterou dostanete na vstupu, najít všechny možné cesty daným labyrintem, které

1. začínají vždy v jeskyni `start` a končí vždy v jeskyni `end`, a současně
2. navštíví malé jeskyně nejvýše jednou.

Na standardní výstup vypište počet nalezených cest.

Ukázkový příklad

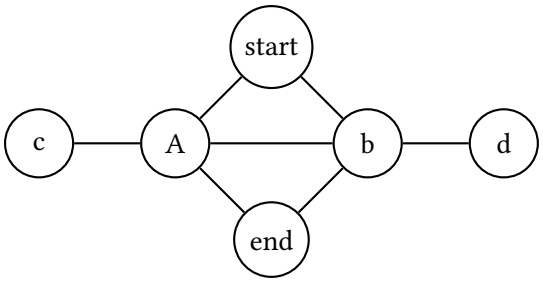
Uvažujme jeskynní systém zadaný sadou spojovacích chodeb:

```
start-A
start-b
A-c
A-b
b-d
A-end
b-end
```

Graf, který odpovídá tomuto zadání je možné vidět na obrázku 6. Všech cest, které splňují zadané podmínky, je 10 a vypadají takto:

```
start,A,b,A,c,A,end
start,A,b,A,end
start,A,b,end
start,A,c,A,b,A,end
start,A,c,A,b,end
start,A,c,A,end
start,A,end
start,b,A,c,A,end
start,b,A,end
start,b,end
```

Můžete si všimnout, že `d` není navštívené nikdy, protože vejít do `d` znamená vejít dvakrát do `b`, což není povoleno.



Obrázek 6: Ukázkový systém jeskyní

Poznámky

- Vstup je soubor, který na každém řádku obsahuje dvojici názvů jeskynních sálů ve formě `jeskyne1-jeskyne2`. Vstup bude vždy obsahovat jeskyně `start` a `end`.
- Jeskyně `start` a `end` jsou malé.
- Velké jeskyně nemají žádné omezení v počtu návštěv.
- Výstupem programu bude jedno číslo, počet cest. Cesty samotné není nutné vypisovat, ale není to ani zakázáno.

5 Bipartitní graf

Problém

V tomto zadání máte implementovat metodu `IsBipartite`, která pro daný graf G bude vracet **true**, pokud je graf G bipartitní. V opačném případě bude vracet **false**. Co je to bipartitní graf se můžete dočíst v literatuře k předmětu nebo například na Wikipedii, viz `https://en.wikipedia.org/wiki/Bipartite_graph`[3]. Předpokládejte, že graf G má n vrcholů, které jsou očíslovány $0, 1, \ldots, n-1$. Graf G bude zadán maticí sousednosti, tj. čtvercovou maticí A o rozměrech $n \times n$, kde prvek matice je $a_{ij} = 1$, pokud existuje hrana z vrcholu i do vrcholu j, jinak je $a_{ij} = 0$.

Ukázkový příklad

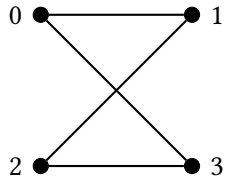
Příklad bipartitního grafu je uveden na obrázcích 7 a 9. Naopak příklad grafu, který není bipartitní je uveden na obrázku 8.

Poznámky

- Každý vrchol grafu je označen, celým číslem i, kde $0 \leq i < n$. Graf bude uložen v textovém souboru v následujícím formátu. Na prvním řádku bude číslo n. Na každém řádku bude uložena n-tice číslic 0 a 1, která představuje jeden řádek matice sousednosti. Ukázkový graf z obrázku 7 bude uložen takto:

```
4
0101
1010
0101
1010
```

[3] Doporučuji si projít celý wiki článek v angličtině – najdete tam mnohem více informací než v českém článku.

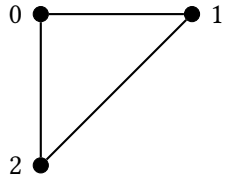


(a) Vizualizace grafu

	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0

(b) Matice sousednosti

Obrázek 7: Bipartitní graf a jeho matice sousednosti



(a) Vizualizace grafu

	0	1	2
0	0	1	1
1	1	0	1
2	1	1	0

(b) Matice sousednosti

Obrázek 8: Cyklický graf

Graf z obrázku 8 bude uložen jako

```
3
011
101
110
```

A konečně graf z obrázku 9 bude uložen jako

```
9
000001000
001010010
010000000
000001000
010000000
100100000
000000001
010000000
000000100
```

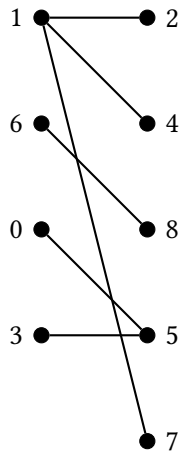
6 Optimalizace rozvrhu úloh

Problém

V tomto zadání budete implementovat algoritmus pro optimální rozvrhování úloh. Máte zadáno n úloh, kde každá úloha i je charakterizována třemi parametry: časem začátku s_i, časem konce e_i a ziskem p_i, který získáme při dokončení úlohy. Dvě úlohy se překrývají, pokud existuje časový okamžik, kdy jsou obě úlohy aktivní. Vaším úkolem je najít takovou podmnožinu nepřekrývajících se úloh, která maximalizuje celkový zisk.

Ukázkový příklad

Předpokládejme, že máme následující úlohy:



(a) Vizualizace grafu

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

(b) Matice sousednosti

Obrázek 9: Bipartitní graf a jeho matice sousednosti

Úloha	Začátek (s_i)	Konec (e_i)	Zisk (p_i)
1	1	4	5
2	2	4	6
3	3	5	5
4	4	6	7

Pro tuto sadu úloh je optimální řešení vybrat úlohy 2 a 4, což dává celkový zisk 13. Všimněte si, že nemůžeme vybrat například úlohy 1 a 2 současně, protože se překrývají v čase.

Poznámky

- Vstupem do vašeho programu bude textový soubor obsahující na prvním řádku počet úloh n. Následuje n řádků, kde každý řádek obsahuje tři celá čísla oddělená mezerou – čas začátku s_i, čas konce e_i a zisk p_i pro danou úlohu. Například:

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

- Pro každou úlohu platí $s_i < e_i$ a všechny hodnoty jsou kladná celá čísla.
- Na standardní výstup vypište maximální možný zisk, kterého lze dosáhnout výběrem nepřekrývajících se úloh.
- Volitelně můžete vypsat i seznam vybraných úloh, které dávají optimální řešení.
- Pro kontrolu správnosti implementace použijte přiložené testovací případy.
- Pro řešení můžete zvolit buď přístup pomocí dynamického programování, nebo využít hladový algoritmus. Obě metody mají své výhody.

Rozdělení zadání mezi studenty

Prezenční forma studia

	Login	Příjmení jméno	Zadaný projekt
1	ADA0306	Adámek Šimon	1
2	ALW0011	Al Wohaishi Gamal	2
3	AND0153	Andrys Matyáš	3
4	ANL0009	Anlauf Dominik	4
5	ASC0002	Ascher Adam	5
6	BAC0092	Backa Jakub	6
7	BAL0312	Baliga Matěj	1
8	BAR0804	Barvík Lukáš	2
9	BEC0072	Becher Adam	3
10	BED0134	Bednik Fedor	4
11	BED0178	Bednárek David	5
12	BER0307	Berezovský Peter	6
13	BES0061	Besta Vojtěch	1
14	BOC0082	Bochkov Artem	2
15	BOH0162	Bohunovský Richard	3
16	BRE0197	Brežák Ota	4
17	BRO0142	Broskov Maxim	5
18	BUC0126	Bučková Eliška	6
19	BUC0130	Buček Martin	1
20	BUR0289	Burget Antonín	2
21	CAR0091	Carnevale Dennis	3
22	CAR0106	Čarnota Dominik	4
23	CER0588	Černošek Jakub	5
24	CER0589	Černý Filip	6
25	CER0590	Černý Tomáš	1
26	CES0022	Česlárová Eliška	2
27	CES0023	Češka Jakub	3
28	CRO0009	Cron Matyáš	4
29	DAV0082	Davydova Nadezhda	5

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
30	DON0040	Dona Václav	6
31	DRE0070	Dressler Jiří	1
32	DRN0013	Drnovský Daniel	2
33	DUR0150	Duraj Tomáš	3
34	DZI0037	Dzifčák Radoslav	4
35	FED0041	Fedorová Adela	5
36	FED0042	Fedorov Dmytro	6
37	FER0151	Feri Kristian	1
38	FIL0196	Filák Jan	2
39	FOJ0143	Fojtíková Lenka	3
40	FOJ0144	Fojtů Radim	4
41	GAF0006	Gafu Madi	5
42	GAL0195	Gála Petr	6
43	GEN0022	Genda Aurel	1
44	GLO0064	Glogar Josef	2
45	HAN0390	Hanke Matěj	3
46	HAV0306	Havelka Václav	4
47	HAV0323	Havlíček Jiří	5
48	HAV0327	Havelka Jakub	6
49	HEC0083	Heczko Vojtěch	1
50	HER0218	Heralt Michal	2
51	HIK0009	Hikl Radek	3
52	HLU0035	Hlubinová Petra	4
53	HLU0048	Hlůšková Natálie	5
54	HOR0590	Horák Jan	6
55	HOT0023	Hotárek Tomáš	1
56	HUD0153	Hudymač Jakub	2
57	CHA0246	Chaloupka Zbyněk	3
58	CHO0275	Chodura Marek	4
59	CHR0188	Chrenko Daniel	5
60	CHY0092	Chyla Petr	6

pokračování na další stránce...

	Login	Příjmení jméno	Zadaný projekt
61	ILY0005	Ilyk Robin	1
62	IND0040	Indrák Jakub	2
63	IRZ0006	Irzhytskyi Dmytro	3
64	JAK0128	Jakubec Adam	4
65	JAN0894	Jančařík Lukáš	5
66	JAN0949	Janča Vojtěch	6
67	JAR0183	Jaroš František	1
68	JEZ0097	Ježek Rostislav	2
69	JEZ0098	Ježek Jiří	3
70	JIR0123	Jiříček Vítek	4
71	JOS0038	Josefus Lukáš	5
72	JUC0040	Juchelka Jakub	6
73	JUK0013	Jukl Vojtěch	1
74	JUR0510	Jurda Tomáš	2
75	KAD0209	Kadlečík Martin	3
76	KAL0339	Kalabiha Myroslav	4
77	KAN0275	Kántor Michal	5
78	KAN0291	Kaňok Marek	6
79	KEL0060	Keleshidi Kirill	1
80	KEM0039	Kemel Václav	2
81	KLE0207	Klečková Hana	3
82	KLI0275	Kliš Daniel	4
83	KLI0293	Klimčenko Jiří	5
84	KOB0078	Kobliha Karel	6
85	KOC0383	Kocik Marek	1
86	KOD0034	Kodrla Daniel	2
87	KOL0603	Kolečkář Tomáš	3
88	KON0500	Kontrik Dominik	4
89	KON0501	Koňařík Ondřej	5
90	KON0503	Konečný Pavel	6
91	KOS0409	Kostelenec Adam	1

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
92	KOV0393	Koval Nazar	2
93	KOV0404	Kováč Marcel	3
94	KOV0406	Kovařčík Jan	4
95	KOZ0345	Kozintsev Roman	5
96	KOZ0366	Kozar Michal	6
97	KRA0677	Kraváriková Glória	1
98	KRA0716	Králík Václav	2
99	KRA0718	Krásný Daniel	3
100	KRI0375	Křižák Michal	4
101	KUB0712	Kubík Josef	5
102	KUB0765	Kubeš Jaroslav	6
103	KUB0770	Kubečka Jiří	1
104	KUC0389	Kucheriavyi Vsevolod	2
105	KUD0134	Kudarov Akzhol	3
106	KUT0082	Kutáč Lukáš	4
107	LAL0017	Lalošák Kryštof	5
108	LAN0229	Laník Aleš	6
109	LEP0042	Lepík Jonatan	1
110	LIA0017	Bednářová Snizhana	2
111	LUT0017	Lutkov Mykola	3
112	LUZ0033	Luzar Jan	4
113	LYZ0003	Lyžbická Jana	5
114	MAC0547	Macura Jan	6
115	MAC0646	Macháček Jakub	1
116	MAR0960	Mariánek Jan	2
117	MAR0962	Markvart Patrik	3
118	MAR0963	Martynková Adéla	4
119	MAR0964	Marszalek Pavel	5
120	MAT0508	Matůš Filip	6
121	MAZ0126	Mazúr Kristián	1
122	MED0061	Medvetskyi Denys	2

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
123	MEL0104	Melnikov Evgenii	3
124	MER0126	Merta Jaroslav Matěj	4
125	MIC0428	Michalský Miroslav	5
126	MIC0480	Michenka Pavel	6
127	MIC0526	Mičánek David	1
128	MIK0542	Mikoláš Adam	2
129	MIL0100	Milian Ondřej	3
130	MIL0101	Milián Dominik	4
131	MOK0060	Mokrousov Ihor	5
132	MOR0257	Moravec Hynek	6
133	MOT0101	Motyka Stanislav Ondřej	1
134	MUC0075	Mucha Tomáš	2
135	MUK0009	Mukha Stanislav	3
136	MUZ0041	Mužný Ondřej	4
137	MYN0035	Mynář Pavel	5
138	NEM0278	Němec Jan	6
139	NES0064	Neskoromna Diana	1
140	NEV0081	Nevoral Rostislav	2
141	NOG0046	Nogová Nikol	3
142	OCH0024	Ochvat Tomáš	4
143	OND0300	Ondrůšek Jakub	5
144	OSC0015	Oščádal Ivo	6
145	OTI0033	Otipková Magdaléna	1
146	OZD0017	Ozdinec Martin	2
147	PAJ0029	Pajonk Sebastian	3
148	PAP0113	Papavasilevská Silvie	4
149	PAT0099	Patzián Jakub, Bc.	5
150	PAT0112	Patola Dominik	6
151	PAZ0043	Pazderskyi Maksym	1
152	PET0469	Peter Tomáš	2
153	PET0471	Petrov Jakub	3

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
154	PET0472	Petera Jaroslav	4
155	PIA0016	Piatovol Roman	5
156	POK0124	Pokluda Ondřej	6
157	POL0541	Polakovič Jakub	1
158	POL0561	Polach Lukáš	2
159	PRO0387	Procházka Jan	3
160	PRO0388	Prorok Vojtěch	4
161	PUK0024	Pukalík Radim	5
162	QUI0020	Quilla Marek Alejandro	6
163	RAB0046	Rabiyev Zhamshed	1
164	REC0030	Rechbergová Eva	2
165	REH0127	Řeha Kryštof	3
166	REP0058	Řepa Lukáš	4
167	RUS0122	Rusek Leoš	5
168	RYC0095	Rychlý Štěpán	6
169	SAJ0054	Sajdok Adam	1
170	SAL0185	Salga Jan	2
171	SEN0097	Šenkeřík Lukáš	3
172	SEV0181	Ševeček Jiří	4
173	SHA0060	Shatalov Roman	5
174	SHA0062	Shatalov Dmitrii	6
175	SHV0005	Shvets Bohdan	1
176	SIH0006	Sihelský František	2
177	SIK0224	Šikula Marek	3
178	SIN0153	Sinkovicz Marian	4
179	SKA0216	Skalík Radek	5
180	SKO0219	Skok Boris	6
181	SKO0236	Škor Matúš	1
182	SKU0126	Škuta Martin	2
183	SLA0373	Slaný Lukáš, Bc.	3
184	SLI0133	Slivka Jan	4

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
185	SLO0177	Slováček Jan	5
186	SNE0038	Sněhota Matěj	6
187	SOL0147	Solovei Vladyslav	1
188	SRO0051	Srostlík Ondřej	2
189	STA0677	Stacha Robin	3
190	STA0678	Staškovan Adam	4
191	STE0585	Štefek Mikuláš	5
192	STU0196	Štůrala Jakub	6
193	SUR0101	Šurman Adam	1
194	SUS0112	Šustr Stanislav	2
195	SVR0053	Svrček Matyáš	3
196	TAN0081	Tancoš Dušan	4
197	TAT0027	Tatarka Andrej	5
198	TIS0028	Tischlinger Ferdinand	6
199	TOB0049	Tobola David	1
200	TOM0429	Tománek Tomáš	2
201	TOM0483	Tomica Martin	3
202	TOS0029	Toška Adam	4
203	TRA0117	Trávníček Richard	5
204	TRA0162	Trabalík Matej	6
205	TRA0163	Tran Phat Dai	1
206	TRA0164	Tran Vuong Dai	2
207	TYL0046	Tyl Marek	3
208	TYM0011	Tymoshenko Bohdan	4
209	URB0295	Urban Dominik	5
210	URB0296	Urbančík Jan	6
211	URU0014	Uruba Štěpán	1
212	VAC0289	Vachník Patrik	2
213	VAL0503	Valošek Pavel	3
214	VAV0268	Vavrečková Veronika	4
215	VES0141	Veselá Michaela	5

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
216	VOD0071	Vodák Filip	6
217	VOJ0129	Vojtěch Jakub	1
218	VOJ0152	Vojtek Lukáš	2
219	VOJ0154	Vojtík Peter	3
220	VRA0163	Vrána Adam	4
221	VRA0175	Vrážel Vojtěch	5
222	VYM0055	Vymazal Zdeněk	6
223	WRU0008	Wrubel Lukáš	1
224	YAT0008	Yatsenko Denys	2
225	ZAJ0138	Zajac Martin	3
226	ZAK0105	Zakharko Volodymyr	4
227	ZAM0070	Zámorský Ondřej	5
228	ZAS0020	Zástřešek Vojtěch	6
229	ZHA0068	Zhaksylyk Nurzhan	1
230	ZIN0023	Žingor Radek	2
231	ZIZ0050	Žižka Vladislav Stephen	3
232	ZUB0052	Zůbek Ondřej	4

Kombinovaná forma studia

	Login	Příjmení jméno	Zadaný projekt
1	BAK0046	Bakoš Šimon, Bc.	1
2	BAR741	Bartoň Stanislav	2
3	BIT0050	Bittmannová Tereza, Ing.	3
4	FIC0024	Ficek Richard	4
5	HOR0588	Horčička Michal, Ing.	5
6	HRO0104	Strafella Jana, Bc.	6
7	HRT031	Hrtus Vít	1
8	JOP0015	Jopčík Dávid	2
9	KOZ0344	Kožušník Lukáš	3
10	KRU0219	Krupík Michal	4

pokračování na další stránce…

	Login	Příjmení jméno	Zadaný projekt
11	KUC0415	Kučera Peter, Bc.	5
12	LAB0040	Labuda Martin	6
13	MAT0509	Matýsek Jindřich	1
14	MUS0160	Musil Juraj, Ing. Ph.D.	2
15	POL0511	Poláš Přemysl, Ing.	3
16	REH0140	Řehák Tomáš, Bc.	4
17	RIC0105	Richter David	5
18	SOL0139	Šoltys Daniel	6
19	SZC0033	Szczepanik Marek	1
20	TAB0048	Tabara Alexandr	2
21	TRO0090	Trombik Daniel	3
22	URV0007	Urválek Lukáš	4
23	VLC028	Vlček Adam	5