Záverečný test
praktická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Doplňujúce zdrojové kódy sú na stránke predmetu PAZ1b. Funkčnosť každého riešenia musí byť preukázaná spustením na testovacom vstupe – nespustiteľné riešenia neumožňujú zisk príslušných bodov.

Najdlhšia cesta v grafe (14 + 3 bodov, backtracking)

Na predmete PAZ1b sme sa stretli s viacerými algoritmami na nájdenie najkratšej, resp. najlacnejšej cesty v grafe (Bellman-Fordov, Dijsktrov, či Floyd-Warshalov algoritmus). Všetky tieto algoritmy fungovali v kvadratickom prípadne kubickom čase, t.j. relatívne rýchlo. Teória zložitosti v kombinácii s teóriou grafov ale dokázali, že pre opačný problém – nájdenie najdlhšej cesty v grafe – to už neplatí. V praxi to znamená, že úlohu je možné vyriešiť len spôsobmi so zložitosťou porovnateľnou skúmaniu všetkých ciest v grafe. Dokonca ak sa zameriame len na neohodnotené grafy, tak nájdenie najdlhšej (v zmysle počtu hrán) cesty v grafe je rovnako ťažký problém.

Úloha: Pre zadaný neohodnotený orientovaný graf (môže obsahovať aj cykly) popísaný v textovom súbore (formát vstupu si môžete zvoliť) nájdite najdlhšiu cestu v grafe. Dĺžka cesty je meraná počtom hrán.

Dodatočné 3 body sú za riešenie, ktoré nebude generovať postupnosti vrcholov, ktoré nie sú cestami.

Zoznamový indexOf (7 bodov, spájané zoznamy + stringológia)

Pripomeňme si najprv metódu `indexOf` triedy `String`. Táto metóda vráti najľavejšiu pozíciu v reťazci, na ktorej začína parametrom zadaný podreťazec, resp. -1, ak reťazec neobsahuje zadaný reťazec ako podreťazec.

Uvažujme triedu `SpajanyZoznam` z prednášky o spájaných zoznamoch. Do triedy `SpajanyZoznam` pridajte metódu `indexOf`, ktorá vráti najľavejšiu pozíciu (index) v zozname, na ktorom začína parametrom zadaný zoznam z ako podzoznam, resp. -1 ak zoznam neobsahuje zoznam z ako podzoznam.

```
public int indexOf(SpajanyZoznam z)
```

Príklad 1: Predpokladajme, že spájaný zoznam obsahuje hodnoty `[8, 9, 5, 5, 1, 4]`. Potom volanie metódy `indexOf([5, 5, 1])` vráti hodnotu 2.
Príklad 2: Predpokladajme, že spájaný zoznam obsahuje hodnoty `[8, 9, 5, 5, 1, 4]`. Potom volanie metódy `indexOf([5, 3, 1])` vráti hodnotu -1.

Efektívnosť implementácie: Ak n je dĺžka spájaného zoznamu a m je dĺžka parametrom zadaného zoznamu `z`, potom časová zložitosť metódy `indexOf` musí byť $O(m.n)$ a pamäťová $O(1)$.

Tranzitívny uzáver relácie (15 bodov, grafové algoritmy)

Určite ste natrafili v minuloročných zadaniach na úlohu o tranzitívnom uzávere relácie, za ktorou sa skrývalo prehľadávanie grafov. Pripomeňme si definície:

- Nech X je množina, potom reláciou nazývame ľubovoľnú množinu R takú, že $R \subseteq X \times X$.
- Povieme, že relácia R je tranzitívna, ak $\forall x, y, z \in X, (x, y) \in R \wedge (y, z) \in R \Longrightarrow (x, z) \in R$
- Tranzitívnym uzáverom relácie R nazveme najmenšiu takú tranzitívnu reláciu R^*, že $R \subseteq R^*$.
- Povieme, že relácia R je symetrická, ak $\forall x, y \in X, (x, y) \in R \Longrightarrow (y, x) \in R$

Uvažujme symetrickú reláciu R. Relácia R môže byť dosť veľká (obsahovať veľa dvojíc). Ako však vyzerá najmenšia taká pomnožina R_{min} množiny R ($R_{min} \subseteq R$), že $R^*_{min} = R^*$ a R^*_{min} je symetrická relácia? Inými slovami chceme z relácie R vyhodiť čo najviac dvojíc tak, aby tranzitívny uzáver vytvorenej symetrickej relácie (podrelácie) bol rovnaký ako tranzitívny uzáver pôvodnej relácie R.

Nech prvky n-prvkovej množiny X sú $x_0, x_1, x_2, \ldots, x_{n-1}$. To, že $(x_i, x_j) \in R$, vieme reprezentovať dvojicou čísel i, j. Vytvorte program, ktorý z textového súboru načíta prvky relácie R a nájde prvky relácie R_{min}.
Formát súboru si môžete zvoliť taký, ako vám vyhovuje. Odporúčaný formát je: v prvom riadku uviesť počet prvkov množiny X (t.j. n) a potom v ďalších riadkoch uviesť prvky relácie R – v každom riadku jeden prvok ako dvojicu čísel, t.j. prvok $(x_i, x_j) \in R$ zapísať ako čísla $i\ j$.

Noemova archa (10+6+8 bodov, dynamické programovanie)

Istotne poznáte príbeh o Noemovi, ktorý postavil archu a nalodil na ňu z každého živočíšneho druhu jeden pár. Archa mala po svojom obvode v niekoľkých poschodiach umiestnené miestnosti – „kajuty“ pre zvieratá. Ale nalodiť zvieratá na archu to nie je len tak. Rôzne zvieratá majú rôzne hmotnosti. Ak by Noe nepostupoval pri umiestňovaní zvierat do „kajút“ archy uvážene, ľahko by sa mohlo stať, že sa archa preváži a prevráti sa na jeden bok. Keďže pred Noemom bola dlhá plavba po rozbúrenom mori, bolo nevyhnutné nájsť také umiestnenie zvierat do „kajút“, aby súčet váh zvierat na ľavoboku sa od súčtu váh zvierat na pravoboku čo najmenej líšil. Noe si teda vytvoril zoznam zvierat a pre každý pár si do tohto zoznamu poznačil aj váhu tohto páru (pár bol samozrejme umiestnený v rovnakej „kajute“). Váha každého páru bola uvedená v celých kilogramoch. Zvieratá, ktoré vážili menej ako kilogram, Noe „neriešil“, keďže na ich váhe až tak pri prevažovaní nezáležalo (ale také hrochy, slony, kone, ... tam tú váhu nemôžeme ignorovať).

Úloha: Vytvorte program, ktorý načíta Noemov zoznam a nájde (a vypíše) najoptimálnejšie rozdelenie zvierať do kajút na ľavoboku a pravoboku.

Noemov zoznam (druh, váha páru v kilogramoch):

```
Slony 24000
Kone 900
Hrochy 3500
```

Hodnotenie:

- 10 bodov za nájdenie optimálneho súčtu váh na ľavoboku a na pravoboku
- 6 bodov za nájdenie optimálneho riešenia s pamäťovou zložitosťou memoizačnej tabuľky závislou od počtu živočíšnych druhov
- 8 bodov za nájdenie optimálneho riešenia s pamäťovou zložitosťou memoizačnej tabuľky nezávislou od počtu živočíšnych druhov