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.

Maľované krížovky (16 bodov, backtracking)

V každom dobrom krížovkárskom časopise sa zvykne okrem klasických krížoviek a sudoku nachádzať aj nejaká tá maľovaná krížovka. Kvôli zjednodušeniu implementácie zadania budeme uvažovať zjednodušenú verziu maľovaných krížoviek. O čo vlastne ide? Krížovka je mriežka s rozmermi M (počet riadkov) x N (počet stĺpcov). Pri každom riadku a každom stĺpci je uvedený počet zafarbených políčok v danom riadku, resp. stĺpci.

Príklad krížovky s rozmermi 5 x 7 a jej možného vyplnenia:

	5	2	3	0	5	2	2
6							
3							
4							
2							
4							

	5	2	3	0	5	2	2
6	■	■	■		■	■	■
3	■		■		■		
4	■	■	■		■		
2	■				■		
4	■				■	■	■

Vytvorte program, ktorý z textového súboru načíta rozmery maľovanej krížovky (čísla M, N) a ďalších M+N čísel určujúcich počty zafarbených políčok v jednotlivých stĺpcoch, resp. riadkoch. Program následne nájde aspoň jedno také vyplnenie krížovky (ak existuje), ktoré spĺňa podmienky správneho vyplnenia krížovky.

Rada: Tabuľku s rozmermi M x N možno reprezentovať ako pole s M*N prvkami. Tabuľku 5 x 7 možno uložiť v 35 prvkovom poli.

Tranzitívny uzáver relácie (13 bodov, grafové algoritmy)

Pojem relácie patrí k prvým pojmom, s ktorými sa študent UPJŠ stretáva na matematických predmetoch. Pripomeňme si pár definícií:

- 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 $R \subseteq R^*$.

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 pre zadaný prvok x_i vypočíta veľkosť množiny $\{y \in X | (x_i, y) \in R^*\}$.

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 $i\ j$.

PAZ1b a algebra (15 bodov, dynamické programovanie)

Spomínate si na násobenie matíc? Ak A je matica typu $r_A \times s_A$ (r_A riadkov a s_A stĺpcov) a B je matica typu $r_B \times s_B$ tak na to, aby sme ich mohli vynásobiť musí platiť, že $s_A = r_B$. Výsledkom násobenia je matica C typu $r_A \times s_B$. Na to, aby sme vypočítali maticu C, potrebujeme celkom $r_A \cdot s_A \cdot s_B$ operácií násobenia prvkov matíc (použijúc naivný školský algoritmus). Teda ak násobíme maticu typu 5×3 a maticu typu 3×8, výsledkom je matica typu 5×8 a na jej výpočet naivným (školským) algoritmom potrebujeme $5 \cdot 3 \cdot 8 = 120$ operácií násobenia prvkov matíc. Z algebry viete, že operácia násobenia matíc je asociatívna, t.j. nezáleží na ozátvorkovaní. A to ide celkom šikovne využiť. (Pripomíname, že násobenie matíc nie je komutatívne.)

Zoberme si 3 matice: A_1 typu 5×4, A_2 typu 4×6 a A_3 typu 6×2. Výsledkom súčinu $A_1A_2A_3$ je matica typu 5×2. Ak by sme matice násobili ako $(A_1A_2)A_3$, na výpočet A_1A_2 potrebujeme celkom $120\ (= 5 \cdot 4 \cdot 6)$ násobení. Potom na vynásobenie matíc (A_1A_2) a A_3 potrebujeme celkom $60\ (= 5 \cdot 6 \cdot 2)$ násobení. Celkom teda máme $120 + 60 = 180$ násobení. Ak by sme súčin $A_1A_2A_3$ počítali ako $A_1(A_2A_3)$, postačí nám už len $88\ (= 4 \cdot 6 \cdot 2 + 5 \cdot 4 \cdot 2)$ násobení. Najprv vypočítame A_2A_3 a potom súčin $A_1(A_2A_3)$.

Vidíme, že na ozátvorkovaní záleží. Problém je jasný: Nájsť také ozátvorkovanie súčinu matíc, aby sme výslednú maticu vypočítali s čo najmenším počtom operácií násobenia. Vytvorte program bežiaci v polynomiálnom čase, ktorý pre zadanú postupnosť matíc $A_1, A_2, \ldots, A_n$ nájde minimálny počet operácií násobenia potrebných na výpočet ich súčinu $A_1A_2 \ldots A_n$. Keďže za sebou idúce matice musia byť kompatibilné, celý vstup vieme popísať pomocou $n + 1$ čísel $r_1, r_2, \ldots, r_n, r_{n+1}$ pričom matica A_i je typu $r_i \times r_{i+1}$.

Návod: Označme si $M[i,j]$ minimálny počet operácií násobenia na výpočet súčinu matíc $A_iA_{i+1} \ldots A_j$. Zjavne $M[i,i] = 0$. Podobne $M[i,i+1] = r_i \cdot r_{i+1} \cdot r_{i+2}$. Zoberme si teraz $M[i,j]$ také, že $j - i > 1$. Keďže operácia súčinu matíc je binárna, súčin $A_iA_{i+1} \ldots A_j$ musel vzniknúť ako súčin $(A_iA_{i+1} \ldots A_k)(A_{k+1}A_{k+2} \ldots A_j)$ pre nejaké k. Ak chceme minimalizovať počet operácií násobenia pri rozdelení súčinu za k-tou maticou, dostaneme, že počet potrebných operácií násobenia je $M[i,k] + M[k+1,j] + r_i \cdot r_{k+1} \cdot r_{j+1}$. Keďže nevieme, ktoré k je „najlepšie“, tak, ako je to v dynamickom programovaní zvykom, vyskúšame všetky k a vyberieme to „najlepšie“. Preto:

$$M[i,j] = \min\{M[i,k] + M[k+1,j] + r_i \cdot r_{k+1} \cdot r_{j+1} | i \leq k < j\}$$

Porovnávanie zoznamov (5 bodov, spájané zoznamy)

Na spájaný zoznam celých čísel sa môžeme pozerať ako na vektor celých čísel. Ak teda zoznam obsahuje hodnoty `[8, 5, 2, 5, 1, 4]`, je to akoby vektor dĺžky 6. S číselnými vektormi môžeme robiť kadejaké operácie. Jedna zo zaujímavých operácií je lexikografické porovnávanie vektorov. Je to úplne rovnaký spôsob porovnávania, aký používame na usporadúvanie slov: Najprv rozhodujeme podľa prvých písmen slova. Ak sú rovnaké, rozhodujú druhé písmena slova, atď. Pri číselných vektoroch sa nerozhodujeme podľa písmen, ale podľa jednotlivých zložiek vektora. Pri porovnávaní vektorov `[8, 5, 2, 5, 1, 4]` a `[8, 5, 1, 100, 1, 40]` by bol postup na zistenie, ktorý z vektorov je lexikografický menší, takýto:

- prvé čísla oboch vektorov sú rovnaké (nevieme rozhodnúť),
- druhé čísla oboch vektorov sú rovnaké (nevieme rozhodnúť),
- tretie čísla oboch vektorov sú rôzne – v prvom vektore je 2, v druhom vektore je 1. Keďže 2 > 1, druhý vektor je lexikograficky menší.

V prípade, že vektory (zoznamy) nemajú rovnakú dĺžku, chýbajúce zložky považujeme za 0.

Uvažujme triedu `SpajanyZoznam` z prednášky o spájaných zoznamoch. Do triedy `SpajanyZoznam` pridajte metódu `compareTo`, ktorá vráti:

- záporné číslo, ak tento spájaný zoznam predstavuje lexikograficky menší vektor ako zoznam `z`,
- 0, ak tento spájaný zoznam predstavuje rovnaký vektor ako zoznam `z`,
- kladné číslo, ak tento spájaný zoznam predstavuje lexikograficky väčší vektor ako zoznam `z`.

```
public double compareTo(SpajanyZoznam z)
```

Podpole so súčtom (3 body)

Vytvorte statickú metódu *podpoleSoSuctom*, ktorá vráti, či pole `p` obsahuje súvislú podpostupnosť (podpole), ktorej súčtom je zadaná hodnota `sucet`. Pozor, hodnoty v poli môžu byť aj záporné.

```
public static boolean podpoleSoSuctom(int[] p, int sucet)
```

Príklad: Pole `[3, 4, 5, 1, 5, 2, 3, 7, 3, 2, 8]` obsahuje súvislé podpole so súčtom 13.

Poznámka: Akceptované je každé riešenie bežiace v polynomiálnom čase vzhľadom na dĺžku poľa `p`.