Záverečný test
teoretická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	

1/3	2/2	3/2	4/2.5	5/2	6/1.5	7/1.5	8/2.5	9/1.5	10/1.5	11/1.5	Σ21.5

1. (3b) Metóda `finalterm` dostane ako parameter g maticu susednosti jednoduchého ohodnoteného neorientovaného grafu a v poli s, ktorého dĺžka je rovná počtu vrcholov grafu, informáciu, ktoré vrcholy grafu patria do nejakej množiny S (`s[i] == true` $\Leftrightarrow i \in S$). Neprítomnosť hrany je v matici g kódovaná hodnotou `Double.POSITIVE_INFINITY`.

```
public static double finalterm(double[][] g, boolean[] s) {
      double result = 0;
      for (int i = 0; i < g.length; i++)
            for (int j = 0; j < g.length; j++)
                  if (s[i] && s[j] && (g[i][j] != Double.POSITIVE_INFINITY))
                        result = result + g[i][j];

      return result;
}
```

Nájdite taký ohodnotený súvislý neorientovaný graf s 8 vrcholmi a takú 4-prvkovú množinu vrcholov S, že metóda `finalterm` vráti číslo 24. Graf zakreslite a uveďte prvky množiny S.

2. (2b) Uvažujme Floyd-Warshallow algoritmus na hľadanie najkratších ciest v grafe. Pre ktoré priradenia premenných i, j, k symbolom vo for-cykloch bude algoritmus fungovať korektne, t.j. korektne sa naplní obsah matice d?

(a) len $\Delta{=}k, \blacktriangledown{=}i, \odot{=}j$
(b) $\Delta{=}k, \blacktriangledown{=}i, \odot{=}j$ a $\Delta{=}k, \blacktriangledown{=}j, \odot{=}i$
(c) $\Delta{=}k, \blacktriangledown{=}i, \odot{=}j$ $\Delta{=}k, \blacktriangledown{=}j, \odot{=}i$ $\Delta{=}i, \blacktriangledown{=}j, \odot{=}k$
(d) $\Delta{=}k, \blacktriangledown{=}i, \odot{=}j$ $\Delta{=}i, \blacktriangledown{=}j, \odot{=}k$ $\Delta{=}j, \blacktriangledown{=}k, \odot{=}i$
(e) len $\Delta{=}k, \blacktriangledown{=}i, \odot{=}j$ a $\Delta{=}i, \blacktriangledown{=}j, \odot{=}k$

```
for (int Δ = 0; Δ < n; Δ++)
  for (int ▼ = 0; ▼ < n; ▼++)
    for (int ☺ = 0; ☺ < n; ☺++)
      if (d[i, k] + d[k, j] < d[i, j])
        d[i, j] = d[i, k] + d[k, j];
```

3. (2b) Základná myšlienka Bellman-Fordovho algoritmu je v orientovanom grafe s n vrcholmi $n-1$ krát postupne vykonať relaxáciu všetkých orientovaných hrán grafu. Vykonávať všetkých n „fáz“ relaxácií všetkých hrán grafu nie je potrebné. Výpočet môžeme ukončiť ihneď po vykonaní takej „fázy“ relaxácií všetkých hrán grafu, v ktorej sa pre žiaden vrchol neznížil odhad ohodnotenia najkratšej cesty zo štartovacieho vrcholu do tohto vrcholu. Pre nejaké vami zvolené n, $n > 6$, nájdite taký n-vrcholový ohodnotený graf a taký „štartovací“ vrchol, že Bellman-Fordov algoritmus bude skutočne potrebovať všetkých $n-1$ „fáz“ relaxácií na nájdenie najkratších ciest z tohto „štartovacieho“ vrcholu.

4. (2.5b) Nech S je množina intervalov $S = \{I_1, I_2, I_3, \dots, I_n\}$, pričom $I_k = \langle a_k, b_k \rangle$. Chceme vybrať najväčšiu (počtom prvkov – intervalov) takú podmnožinu Q množiny S, $Q \subseteq S$, že pre ľubovoľné 2 prvky $I_i, I_k \in Q$ platí $I_i \cap I_k = \emptyset$. Teda žiadne 2 intervaly z Q nemajú spoločný prienik. Uvažujme nasledovný greedy algoritmus:

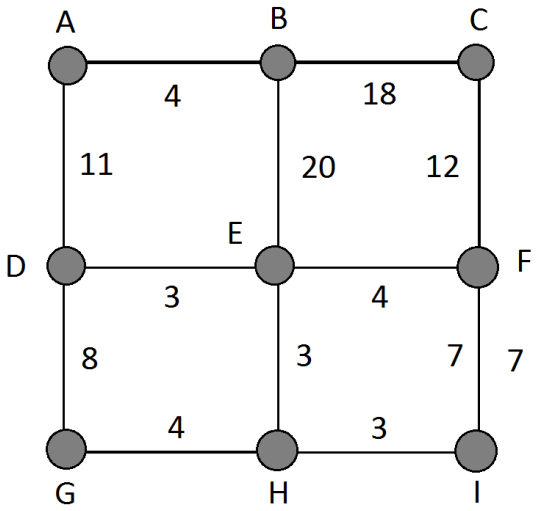
   $Q \leftarrow \emptyset$;
   kým (S nie je prázdna) {
      vyber taký interval $I \in S$, ktorý má najmenší možný začiatok spomedzi intervalov v S;
      presuň I z množiny S do Q, t.j. $Q \leftarrow Q \cup \{I\}$ a $S \leftarrow S \setminus \{I\}$;
      odstráň z množiny S všetky intervaly, ktoré majú neprázdny prienik s I;
   }

   Dokážte, že algoritmus vráti korektný výsledok alebo nájdite kontrapríklad.

5. (2b) Kľúčovým prvkom Knuth-Morris-Prattovho algoritmu je skip matica, ktorá pre aktuálnu pozíciu vo vzorke a písmeno na vstupe rozhoduje, o koľko pozícií sa má posunúť začiatok vzorky vo vstupe (narozdiel od naivného algoritmu, kde sa pri nezhode posúva začiatok vzorky vždy o 1 pozíciu doprava). Vyplňte skip maticu pre zadanú vzorku a množinu písmen vzorky. Stĺpce odpovedajú aktuálnej pozícii vo vzorke a riadky písmenu, ktoré je na vstupe.

	A	A	A	B	A
A		0			
B					
iné písmeno			3		

V úlohách 6 a 7 budeme uvažovať nasledujúci <u>neorientovaný ohodnotený</u> graf:



6. (1.5b) Uvažujme prehľadávanie do šírky, v ktorom susedné vrcholy navštevujeme v abecednom poradí. Zapíšte takú postupnosť vrcholov grafu, ktorá zodpovedá poradiu, v akom budú jednotlivé vrcholy navštívené pri prehľadávaní do šírky so štartovacím vrcholom D:

   D, _____

7. (1.5b) Zapíšte postupnosť hrán grafu, v akej budú pridané jednotlivé hrany grafu do minimálnej kostry pri aplikovaní Primovho algoritmu s iniciálnym vrcholom A:

   _____

8. (2.5b) Nech $e = (u, v)$ je prvá hrana, ktorú zaradí Primov algoritmus s iniciálnym vrcholom u do minimálnej kostry. Potom minimálna kostra, ktorú nájde Primov algoritmus s iniciálnym vrcholom u, a minimálna kostra, ktorú nájde Primov algoritmus s iniciálnym vrcholom v, sú rovnaké. Tvrdenie dokážte alebo vyvráťte (nájdite kontrapríklad).

9. (1.5b) Pri hľadaní najkratších ohodnotených ciest v grafe je základnou operáciou relaxácia hrany. Nech $d[x]$ je aktuálny odhad dĺžky najkratšej ohodnotenej cesty (resp. sledu) zo štartovacieho vrcholu do vrcholu x a nech $p[x]$ je predchodca vrcholu x v nejakom slede s dĺžkou $d[x]$ zo štartovacieho vrcholu do vrcholu x. Ďalej nech $g[i][j]$ je ohodnotenie orientovanej hrany z vrcholu i do vrcholu j. Napíšte „zdrojový kód“ **operácie relaxácie** orientovanej hrany z vrcholu i do vrcholu j spolu s aktualizáciou obsahu poľa p:

10. (1.5b) Nech p je pole prirodzených čísel dĺžky n indexované od 1 po n. Označme $D[S][i]$ logickú hodnotu (true/false) vyjadrujúcu, či existuje taká podmnožina z prvých i čísel poľa p $(p[1], p[2], \cdots, p[i])$, ktorej súčet je presne S (S je nezáporné celé číslo). Vyjadrite hodnotu $D[S][i]$ na základe hodnôt $D[?][j]$, kde $j < i$, a obsahu poľa p, t.j. na základe rekurzívneho vzorca „pre dynamické programovanie“. Očakáva sa matematické vyjadrenie vzorca pre $D[S][i]$, nie zdrojový kód!

Triviálny prípad (pre i = 0):
$D[S][0] \Leftrightarrow S = 0$

Všeobecný prípad (pre $0 < i \leq n$):

$D[S][i] \Leftrightarrow$ _____

11. (1.5b) Aký bude obsah poľa v, ak pole p obsahuje hodnoty uvedené v tabuľke?

	0	1	2	3	4	5	6	7
p	5	4	8	7	9	3	10	5
v								

```
int[] v = new int[p.length];
Stack<Integer> z = new Stack<Integer>();
for (int i = 0; i < p.length; i++) {
      while ((!z.isEmpty()) && (p[z.peek()] >= p[i]))
            z.pop();

      if (z.isEmpty())
            v[i] = -1;
      else
            v[i] = z.peek();

      z.push(i);
}
```

peek – vráti, aká je hodnota na vrchu zásobníka.