Záverečný test
teoretická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	

1/2	2/2	3/2	4/3	5/2	6/2	7/1.5	8/1.5	9/2	10/1.5	11/4	Σ23.5

1. (2b) Priemerom súvislého neorientovaného grafu nazveme také najmenšie prirodzené číslo d, že medzi každými dvoma vrcholmi grafu existuje cesta dĺžky nanajvýš d. Bellman-Fordov algoritmus je založený na tom, že sa zrealizuje $n-1$ iterácií, pričom v každej z nich sa zrelaxujú všetky hrany grafu. Dokážte alebo vyvráťte: V grafe s priemerom d budú po d iteráciách algoritmu všetky vrcholy vybavené (t.j. nie je potrebných všetkých $n-1$ iterácií).

2. (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.

	X	Y	X	Y	Y
X			0		
Y					
iné písmeno			3		

3. (2b) Pole P dĺžky n obsahuje permutáciu čísel $0,\dots,n-1$ (t.j. každé číslo z tohto rozsahu sa v poli P nachádza práve raz). Navrhnite a zapíšte kód, ktorý v čase $O(n)$ vytvorí také pole R dĺžky n, že platí: $R[x] < R[y]$ práve vtedy, ak sa hodnota x v poli P nachádza na menšom indexe ako hodnota y ($x, y \in \{0,\dots,n-1\}$).

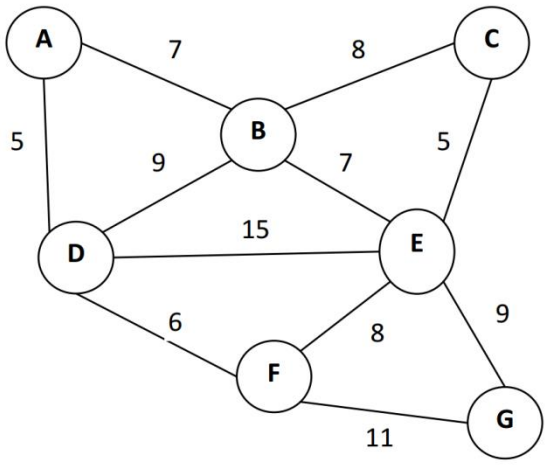
4. (3b) Nech `g` je matica susednosti ohodnoteného orientovaného grafu, v ktorom je neprítomnosť hrany reprezentovaná hodnotou `Double.POSITIVE_INFINITY` Uvažujme Floyd-Warshallov algoritmus nižšie, ktorý v matici `d` pre každú dvojicu vrcholov i a j vypočíta dĺžku najlacnejšej cesty z vrcholu i do vrcholu j. Doplňte do tohto algoritmu príslušné príkazy, ktoré zabezpečia, že sa počas výpočtu vypočíta matica `p`, ktorá pre každú dvojicu vrcholov i a j bude uchovávať priameho predchodcu vrcholu j na nejakej najlacnejšej ceste z i do j (predposledný vrchol na nejakej najlacnejšej ceste z i do j) alebo hodnotu -1, ak cesta z i do j neexistuje.

```
double[][] g = ...;
int n = g.length;
double[][] d = new double[n][n];
int[][] p = new int[n][n];
for (int i=0; i<n; i++) {
      for (int j=0; j < n; j++) {
            d[i][j] = g[i][j];



      }
}

for (int k=0; k < n; k++) {
      for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                  if (d[i][k] + d[k][j] < d[i][j]) {
                        d[i][j] = d[i][k] + d[k][j];



                  }
            }
      }
}
```

5. (2b) Nech `p` je matica vytvorená algoritmom z predošlej úlohy. Napíšte kód metódy `pocetHranCesty`, ktorá pre zadané vrcholy vráti počet hrán nájdenej najlacnejšej cesty z u do v (resp. -1, ak takej cesty niet).

```
public int pocetHranCesty(int u, int v, int[][] p) {









}
```

6. (2b) Uvažujme graf nižšie. Určte množinu hrán, ktoré vyberie Kruskalov algoritmus na nájdenie najlacnejšej kostry. Hrany zapíšte v poradí, v akom ich tento algoritmus pridá do konštruovanej kostry.



7. (1.5b) Uvažujme graf z predošlej úlohy. Ak prehľadávanie do hĺbky je inicializované vo vrchole B a susedné vrcholy „spracúvavame“ vždy v abecednom poradí, zapíšte postupnosť vrcholov grafu, v akej budú navštívené:

_____

8. (1.5b) Uvažujme neorientovaný ohodnotený graf s n vrcholmi a m hranami. Kruskalov algoritmus na nájdenie najlacnejšej kostry sa skladá z troch algoritmických komponentov:
   - usporiadanie hrán v čase $O(m.log\ m)$,
   - overenie, či konce hrany ležia v dvoch rôznych množinách vrcholov v čase $T_F(n)$,
   - spojenie (zjednotenie) dvoch množín vrcholov v čase $T_U(n)$.

   Vyjadrite časovú zložitosť Kruskalovho algoritmu ako funkciu závislú od n, m, $T_F(n)$ a $T_U(n)$.

9. (2b) Kniha cvikov obsahuje zoznam rôznych fitness cvikov. O každom cviku vieme, koľko kalórii sa pri ňom spáli a tiež aké je trvanie cviku v minútach. Fitness trenér chce zostaviť M-minútový tréning (tréningový plán), pričom každý cvik chce počas tréningu realizovať nanajvýš raz a v plnej dĺžke. Zároveň chce, aby sa počas tohto tréningu spálilo čo najviac kalórii. Rozhodol sa použiť takýto algoritmus:
   - Usporiadaj cviky podľa pomeru spálených kalórii a dĺžky cviku v minútach pričom chceme, aby tieto pomery tvorili nerastúcu postupnosť. T.j. prvý cvik je cvik s najväčším počtom spálených kalórii za minútu. Potom do výberu postupne uvažuj cviky podľa vytvoreného usporiadania. Cvik sa pridá do výberu práve vtedy, keď jeho trvanie je kratšie alebo rovné ako M mínus súčet trvaní už vybraných cvikov. Teda cvik pridávame, ak sa časovo celý zmestí do zostávajúceho času po vykonaní už vybraných cvikov.

   Vedie tento algoritmus k optimálnemu riešeniu? Zdôvodnite alebo nájdite kontrapríklad.

10. (1.5b) Nájdite orientovaný graf, ktorý má práve 4 rôzne topologické usporiadania vrcholov.

11. (4b) Označte pravdivosť tvrdení (A – pravda, N – nepravda, +0.5b za správnu odpoveď, -0.5b za nesprávnu odpoveď, 0b za žiadnu odpoveď):

☐ Uvažujme prehľadávanie súvislého grafu algoritmom do šírky resp. do hĺbky. Hrany, ktorými sme nejaký vrchol navštívili po prvý ráz tvoria kostru. Kostra vytvorená prehľadávaním do šírky má vždy menší počet hrán ako kostra vytvorená prehľadávaním do hĺbky.

☐ Primov algoritmus na nájdenie najlacnejšej kostry funguje korektne aj v prípade, že ohodnotenia hrán sú záporné.

☐ V každom ohodnotenom n-vrcholovom grafe s nezápornými cenami a s $n > 10$ existuje medzi každými 2 vrcholmi nanajvýš $2.n^2$ rôznych najlacnejších ciest.

☐ Časová zložitosť Floyd-Warshallovho algoritmu je $O(n^3)$, kde n je počet vrcholov grafu a nezáleží od počtu hrán grafu.

☐ Ak n je veľkosť vstupu a m je veľkosť vzorky, potom nájdenie všetkých výskytov vzorky vo vstupnom texte je možné realizovať Karp-Rabinovým algoritmom (založený na hašovaní) s časovou zložitosťou $O(n + m + c.m)$, kde c je počet výskytov vzorky vo vstupnom texte.

☐ Algoritmus na topologické usporiadanie vrcholov orientovaného grafu môžeme priamočiaro (bez úprav) použiť aj na zistenie prítomnosti orientovaného cyklu v grafe.

☐ Dijkstrov algoritmus vyberá z množiny potenciálne nevybavených vrcholov taký vrchol, ktorého horné ohraničenie ceny doň vedúcej najlacnejšej cesty je minimálne. Vybraný vrchol je prehlásený za vybavený a žiadna ďalšia neskoršia relaxácia akejkoľvek hrany v grafe už nezníži toho horné ohraničenie.

☐ Uvažujme Dijkstrov algoritmus na nájdenie najlacnejších ciest. V okamihu, keď pri relaxácií hrán vychádzajúcich z vybraného vrcholu nedôjde k zmenám horných ohraničení jeho susedov, algoritmus môžeme ukončiť, pričom platí, že všetky zatiaľ nevybrané vrcholy už sú vybavené, t.j. ich horné ohraničenia sa rovnajú cenám najlacnejších doň vedúcich ciest.