Záverečný test
teoretická časť

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Píšte prosím čitateľne!

Hodnotenie, vyplní opravujúci:

Meno a priezvisko:	Skupina PAZ:	

1/2	2/1.5	3/7	4/2	5/1.5	6/1	7/2.5	8/2.5	9/2.5	∑22.5

1. (2b) 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 = Double.POSITIVE_INFINITY;
      for (int i = 0; i < g.length; i++)
            for (int j = 0; j < i; j++)
                  if (s[i] && !s[j] && (g[i][j] != Double.POSITIVE_INFINITY))
                        result = Math.min(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 22. Graf zakreslite a uveďte prvky množiny S.

2. (1.5b) Uvažujme Karp-Rabinov algoritmus, ktorý pri vyhľadávaní všetkých výskytov vzorky vo vstupe využíva haše vzorky a príslušných podreťazcov vstupu. Nájdite takú vzorku a taký vstup, aby Karp-Rabinov algoritmus vykonal najväčší možný počet porovnaní znakov nezávisle od použitej hašovacej funkcie (=postupu, ako sa vyráta haš pre reťazec).

Vstup (15 znakov):

Vzorka (5 znakov):

3. (4b) Označte pravdivosť tvrdení (A – pravda, N – nepravda, +0.5b za správnu odpoveď, -0.5b za nesprávnu odpoveď, 0b za žiadnu odpoveď):

☐ 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.

☐ (A) Memoizácia (pamätanie si a znovupoužitie výsledkov podvýpočtov) umožňuje znížiť časovú zložitosť každého rekurzívneho výpočtu.

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

☐ (B) Uvažujme neorientovaný ohodnotený graf G. Vytvorme nový graf H z grafu G tak, že ohodnotenie každej hrany zvýšime o 10. Potom kostra (=množina vybraných hrán, nie cena kostry) nájdená Kruskalovým algoritmom v grafe G je totožná s kostrou nájdenou týmto algoritmom v grafe H.

☐ (C) Ak graf obsahuje nejaké 2 hrany s rovnakým ohodnotením, potom má viac než len jednu najlacnejšiu kostru.

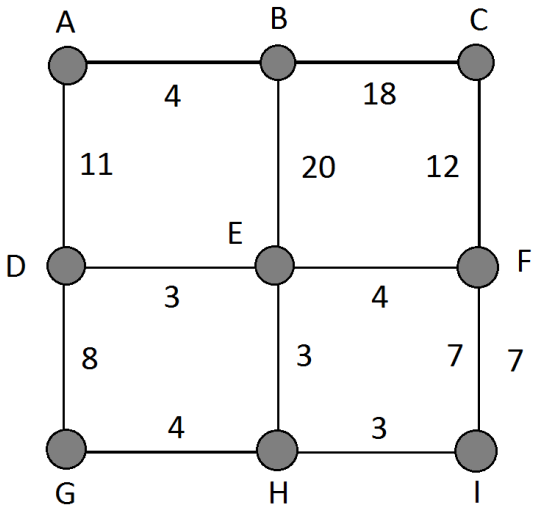
☐ (D) Nech T je najlacnejšia kostra neorientovaného ohodnoteného grafu. Potom pridanie mimokostrovej hrany (t.j. hrany grafu, ktorá nie je v T) do kostry T v nej vytvorí cyklus (kružnicu). Zároveň platí, že táto mimokostrová hrana má cenu, ktorá je väčšia alebo rovnaká ako cena ktorejkoľvek inej hrany v tomto cykle.

☐ 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 naivný algoritmu na nájdenie všetkých výskytov vzorky v texte za každých okolností vykoná aspoň $n.m/2$ porovnaní znakov.

Zdôvodnite svoju odpoveď ku ktorémukoľvek tvrdeniu označenému písmenom A-D (3b). Odôvodňované tvrdenie označte zakrúžkovaním písmena.

V úlohách 4-6 budeme uvažovať nasledujúci neorientovaný ohodnotený graf:



4. (2b) Zapíšte postupnosť hrán grafu, v akej budú pridané jednotlivé hrany grafu do minimálnej kostry pri aplikovaní Primovho algoritmu so štartovacím vrcholom A:

_____

5. (1.5b) Ak prehľadávanie do hĺbky je inicializované vo vrchole E a susedné vrcholy „spracúvame“ vždy v abecednom poradí, zapíšte postupnosť vrcholov grafu, v akej budú navštívené:

_____

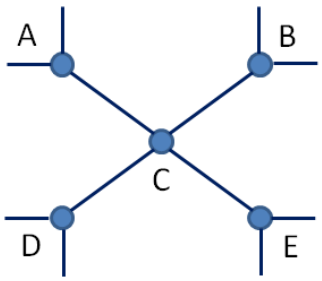
6. (1b) Uvažujme Dijkstrov algoritmus s iniciálnym vrcholom I. Prvý vrchol označený ako vybavený je vrchol I. Ktorý vrchol grafu bude označený ako vybavený v poradí ako druhý a ktorý ako tretí?

Druhý:_____ Tretí: _____

7. (2.5b) Po skončení prehľadávania do šírky súvislého neorientovaného grafu sme dostali takéto pole uchovávajúce pre každý vrchol grafu jeho vzdialenosť od iniciálneho vrcholu prehľadávania:

A	B	C	D	E	.....
5	2	4	3	3	

Váš kolega tvrdí, že vzhľadom na to, ako vyzerá malá časť grafu s vrcholmi A-E, toto nemôže byť korektné riešenie. Zdôvodnite, že toto riešenie nemôže byť korektné, alebo nájdite taký graf (=doplňte do znázornenej časti grafu nové vrcholy a hrany) a označte iniciálny vrchol prehľadávania tak, aby hodnoty v tabuľke boli korektným výsledkom. Vo vytvorenom grafe hrany incidujúce s vrcholmi A-E môžete pridať len ako rozšírenia znázornených výbežkov (teda vrcholy A, B, D a E musia mať stupeň 3 a vrchol C stupeň 4).



8. (2.5b) Uvažujme súvislý ohodnotený orientovaný graf, ktorého vrcholy sú označené číslami 0 až *N - 1*, kde *N* je počet vrcholov v grafe. V tomto grafe sme si zvolili štartovací vrchol `s` a aplikovali sme algoritmus na nájdenie najkratších (najlacnejších) ciest z vrcholu `s` do všetkých ostatných vrcholov grafu. Získali sme taktiež aj jednorozmerné pole `p`, pričom platí, že `p[s] = -1` a pre všetky ostatné vrcholy `p[i]` určuje predchodcu vrcholu `i` na nejakej najkratšej ceste z `s` do `i`. Napíšte metódu `cesta`, ktorá zrekonštruuje najkratšiu cestu zo štartovacieho vrcholu `s` do zvoleného vrcholu `t`. Môžete predpokladať, že `t` je rôzne od `s`. Poradie vrcholov v zrekonštruovanej ceste môže byť buď od štartovacieho vrcholu `s` po koncový vrchol `t` alebo aj opačne od `t` po `s` (uveďte aké).

```
public static List<Integer> cesta(int[] p, int t) {
     List<Integer> vysledok = new ArrayList<Integer>();




     return vysledok;
}
```

9. (2.5b) Pred výťahom s maximálnou nosnosťou M, stojí n osôb s váhami $m_1, m_2, \ldots, m_n$. Cieľom je všetky osoby prepraviť výťahom. Uvažujme takýto greedy algoritmus:
   1. Osoby usporiadame do radu podľa hmotnosti. Prvá osoba v rade má najmenšiu hmotnosť, posledná najväčšiu hmotnosť.
   2. Kým kapacita výťahu nie je prekročená, osoby nastupujú do výťahu podľa poradia v rade (najľahšia čakajúca osoba nastupuje prvá, atď.).
   3. Výťah odíde.
   4. Ak pred výťahom ešte niekto stojí, privolá sa výťah a ide sa na krok 2.

   Vedie ale tento greedy algoritmus k riešeniu s minimálnym počtom jázd výťahu? Vyargumentujte korektnosť algoritmu alebo nájdite kontrapríklad, že tento algoritmus nevedie k optimálnemu riešeniu.