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.

Popadané písmená (12 bodov, backtracking, stringológia)

Spoločnosť PAZTruck dostala objednávku na prevoz reťazcov znakov. Jednotlivé písmená prenášaných reťazcov boli prilepené na dlhých doskách, každá bola uložená v samostatnej krabici. Pri prevoze sa však stalo to, že sa na doskách jedno alebo viac písmen odlepilo a spadlo. Pri nakládke sa trebárs naložil reťazec „Programovanie“ a pri vykládke ostal na doske reťazec „Pr_gra_ova_ie“ a na zemi písmená „m“, „o“ a „n“ (podčiarkovník označuje pozíciu na doske, z ktorej sa odlepilo písmeno). Aby zákazník nereklamoval poškodenie prepravovaného tovaru, musia sa teraz odlepené písmená prilepiť. Ako však zistiť, ktoré písmeno kam patrí? Našťastie na každej krabici je napísaný md5 hash reťazca, ktorý v nej bol uložený. Vytvorte program, ktorý pre zadaný reťazec s odlepenými písmenami, nájdené písmená a hash pôvodného reťazca určí, ako nalepiť jednotlivé nájdené písmená tak, aby výsledný reťazec mal požadovaný hash.

Formát vstupu: Každý riadok vstupu obsahuje 3 medzerou oddelené reťazce. Prvý reťazec obsahuje reťazec na doske, druhý reťazec odlepené písmená a tretí reťazec hash pôvodného reťazca.

Príklad:

```
Pr_gra_ova_ie mon 5465b1fe6a114adc8ee7b6001dfc3abc
A_go_itmiz_c_a aril d2207c7e8b36b61bed4ad38d5f5e3aaa
```

Metóda na výpočet md5 hashu zadaného reťazca:

```
public static String md5(String s) {
	try {
		MessageDigest m = MessageDigest.getInstance("MD5");
		m.reset();
		m.update(s.getBytes());
		byte[] digest = m.digest();
		BigInteger bigInt = new BigInteger(1, digest);
		return bigInt.toString(16);
	} catch (Exception e) {
		return null;
	}
}
```

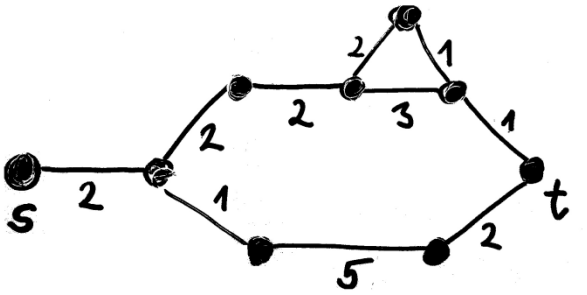
Minimálna triangulácia konvexného mnohouholníka (18 bodov, dynamické programovanie)

Na internete sme našli návod, ako nájsť minimálnu trianguláciu konvexného mnohouholníka (v prílohe zadania). Aplikovaním dynamického programovania implementujte riešenie tohto problému na základe tohto návodu.

Cesty, cesty koľko vás je? (15 bodov, grafové algoritmy)

Nájsť najkratšiu cestu v grafe je už pre každého, kto zažil PAZko, nudná vec. Jednoducho sa nasadí nejaký z grafových algoritmov a potom sa spätným prechodom zrekonštruuje najkratšia cesta. Celkom zaujímavá je však iná úloha. Ak si zoberieme 2 vrcholy grafu, koľko rôznych najkratších ciest medzi nimi existuje?
Vytvorte program, ktorý pre zadaný ohodnotený graf (načítaný z textového súboru) a zadané dva vrcholy grafu vráti počet rôznych najkratších ciest vedúcich medzi týmito vrcholmi. Predpokladajte, že každá hrana grafu má kladné ohodnotenie (>0). Je na vašom rozhodnutí, či budete uvažovať orientované alebo neorientované grafy.

Príklad: Medzi s a t existujú 3 rôzne najkratšie cesty. Pridanie hrany s cenou väčšou ako 9 medzi t a susedom s by počet týchto ciest nezmenilo, no pridanie hrany s cenou menšou ako 8 by počet rôznych najkratších ciest zmenilo na 1.



Rada: Predpokladajme, že pre každý vrchol v grafu $G = (V, E)$ poznáme $\delta(v)$ - cenu najkratšej cesty zo zadaného vrcholu s do vrcholu v. Nech $c(u, v)$ označuje cenu hrany $(u, v) \in E$. Označme si ako $PC(v)$ počet rôznych najkratších ciest z vrcholu s do vrcholu v. Zjavne $PC(s) = 1$. Uvažujme vrchol $v \neq s$. Ak je vrchol v dosiahnuteľný z s, potom existuje aspoň jeden jeho susedný vrchol w, že w je predchodca v na nejakej najkratšej ceste z s do v. Nech W je množina všetkých takýchto susedov vrcholu v, t.j. $W = \{w \in V, (w, v) \in E \wedge \delta(w) + c(w, v) = \delta(v)\}$. Potom $PC(v) = \sum_{w \in W} PC(w)$.

Extrakcia podzoznamov (7 bodov, spájané zoznamy)

Uvažujme triedu `SpajanyZoznam` z prednášky o spájaných zoznamoch. Do triedy `SpajanyZoznam` pridajte metódu `exkrahujPodzoznam`. Táto metóda nech zo spájaného zoznamu odstráni uzly podzoznamu, ktorý začína na indexe `odIdx` a jeho dĺžka je `dlzka`. Volanie metódy zároveň vráti referenciu na novovytvorený objekt triedy `SpajanyZoznam`, ktorý bude obsahovať tie uzly, ktoré boli v odstránenom podzozname. V prípade, že extrahovanie podzoznamu zadanej dĺžky nie je možné, extrahujte čo najviac uzlov v zozname počnúc zadaným indexom (vid príklady).

```
public SpajanyZoznam exkrahujPodzoznam(int odIdx, int dlzka)
```

Príklady: Uvažujme zoznam s hodnotami `[3, 5, 1, 8, 10, 4, 9]`.

- `exkrahujPodzoznam(2, 4)`: `[3, 5, 9]`, vráti `[1, 8, 10, 4]`
- `exkrahujPodzoznam(0, 3)`: `[8, 10, 4, 9]`, vráti `[3, 5, 1]`
- `exkrahujPodzoznam(4, 5)`: `[3, 5, 1, 8]`, vráti `[10, 4, 9]`
- `exkrahujPodzoznam(20, 5)`: `[3, 5, 1, 8, 10, 4, 9]`, vráti `[]`

Požiadavky na implementáciu:

- Metóda má byť implementovaná efektívne, t.j. je dovolené len modifikovať referencie na existujúce uzly. Volanie `new Uzol(…)` sa vo vykonávaní tejto metódy nesmie uskutočniť.
- Pamäťová zložitosť $O(1)$, časová zložitosť $O(n)$ pri n-prvkovom zozname.

Miešačka (10 bodov, greedy algoritmy)

Jožko si cez leto našiel brigádu na stavbe. Jeho úloha bola jednoduchá – v miešačke robiť betón. Jožko je však veľký lenivec a tak sa chce v práci čo najmenej nadrieť. Na jeho prácu však dozerá vedúci, ktorý ho chodí pravidelne kontrolovať. Keďže Jožko nechce prísť o prácu, nemôže si dovoliť, aby ho vedúci načapal ako nepracuje. Po pár dňoch Jožko vypozoroval, že vedúci ho chodí kontrolovať stále v rovnaké časy. Vytvoril si teda textový súbor, do ktorého si zapísal časy dňa (napr. ako počet minút od začiatku šichty), kedy ho vedúci príde skontrolovať. Pri betóne sa ale nedá pracovať tak, že keď vedúci odíde, prestane sa pracovať. Betón by v miešačke zatvrdol. Ak teda Jožko začne plniť miešačku, musí ju celú dokončiť a betón vysypať. Prvá miešačka trvá Jožkovi presne m (parameter programu) minút, druhá o 3 minúty dlhšie ako prvá, tretia o 3 minúty dlhšie ako druhá, atď. (s každou miešačkou mu síl ubúda). Vytvorte program, ktorý na základe plánu kontrol vypočíta pre Jožka, kedy má začať robiť miešačky, aby vždy, keď ho príde skontrolovať vedúci, pracoval.