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.

Písmenká sú „in“

Popadané písmená (10 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 škatuli. 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 škatuli 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;
	}
}
```

Slovný rebrík (14 bodov, grafové algoritmy)

Slovný rebrík je jednoduchá optimalizačná hra pre jedného hráča. Na začiatku hry sa zvolí začiatočné slovo, koncové slovo a hráč ma k dispozícii slovník so slovami (zoznam slov). Začiatočné slovo, koncové slovo a každé slovo v slovníku má rovnaký počet písmen. Začiatočné a koncové slovo nemusia byť v slovníku. Hra začína so začiatočným slovom. V každom kroku hry môže hráč zmeniť jedno písmeno v slove. Musí však platiť, že slovo po zmene písmena je alebo koncové slovo alebo niektoré slovo zo slovníka. Cieľom hráča je nájsť čo najkratšiu postupnosť krokov (zmien písmen), ktorá zmení začiatočné slovo na koncové slovo.

Úloha: Implementujte program, ktorý z textového súboru načíta slovník (zoznam slov, v každom riadku je jedno slovo) a pre zadané začiatočné a koncové slovo nájde najkratšiu postupnosť krokov (zmien písmen), po ktorej sa zo začiatočného slova vyrobí koncové slovo. Ak taká postupnosť krokov neexistuje, program nech o tom vypíše správu.

Príklad:
Slovník: `["hot","dot","dog","lot","log"]`
Najkratšia zmena z `"hit"` na `"cog"`:
`["hit","hot","dot","dog","cog"]` prípadne `["hit","hot","lot","log","cog"]`

Hodnotenie:
- 8b za riešenie, ktoré v polynomiálnom čase vráti počet krokov najkratšej postupnosti
- +6b za riešenie, ktoré v polynomiálnom čase vráti postupnosť krokov (slov) najkratšej postupnosti

Žolíkové znaky (16 bodov, napr. dynamické programovanie)

Pri používaní príkazového riadku v Linuxe ste sa možno stretli s tzv. „žolíkovnými znakmi“ (wildcard characters). Tieto znaky slúžia ako zástupné symboly za jeden alebo niekoľko (aj nula) znakov.

Uvažujme malé písmená anglickej abecedy (a-z) a žolíkové znaky * (hviezdička) a ? (otáznik). Znak ? nahrádza (zastupuje) ľubovoľné jedno písmeno. Znak * nahrádza ľubovoľnú postupnosť znakov - aj prázdnu. Povieme, že zadaný reťazec zodpovedá vzoru, ak každý žolíkový znak vo vzore vieme nahradiť takou postupnosťou znakov, že dostaneme zadaný reťazec. Resp. že znaky v zadanom reťazci vieme nahradiť žolíkovými znakmi tak, že dostaneme zadaný vzor.

Príklad:
- vzoru "p?o*m*" vyhovujú napríklad reťazce "programovanie", "program", "pjotrami"
- vzoru "*p?c?tac*" vyhovujú napríklad reťazce "mikropocitac", "pocitac", "hyperpocitacovy"

Úloha: Vytvorte metódu, ktorá pre zadané slovo a vzor vráti, či slovo zodpovedá (vyhovuje) vzoru. Pri riešení nie je dovolené použiť regulárne výrazy ani inú obdobnú techniku dostupnú v Jave.

Rada: Označme si $R[i,j]$ pravdivostnú hodnotu, ktorá hovorí, či prvých i znakov reťazca zodpovedá (vyhovuje) prvým j znakom vzoru.

Hodnotenie:
- 8b za riešenie v polynomiálnom čase
- +8b za metódu, ktorá pre každý znak vzorky vráti počet prislúchajúcich znakov v reťazci.
  Príklad: pre vzorku "p?o*m*" a reťazec "programovanie" vráti pole [1, 1, 1, 3, 1, 6].

Žiarovky (13 bodov, grafové algoritmy?)

Máme N žiaroviek, číslovaných 1, …, N. Každá žiarovka je pripojená na nulový vodič. Zároveň je pri každej žiarovke svorkovnica na pripojenie fázového vodiča. Do svorkovnice môžeme pripojiť aj viacero káblov. Okrem toho máme špeciálnu svorkovnicu označenú 0, na ktorej je zdroj napätia. V každom kroku môžem káblom prepojiť nejaké dve svorkovnice. Žiarovka zasvieti, ak sa na prislúchajúcej svorkovnici objaví napätie. Pripomeňme, že ak na niektorú zo svorkovníc privedieme napätie (ľudovo fázu), objaví sa toto napätie aj na všetkých svorkovniciach prepojených s touto svorkovnicou.

Úloha: Na vstupe (napr. v textovom súbore) máme zoznam dvojíc popisujúci postupné prepájanie svorkovníc káblami. Každá dvojica zodpovedá jednému kroku, kedy sa prepájajú káblom príslušné svorkovnice. Vytvorte program, ktorý vypočíta, po ktorom kroku zasvietia všetky žiarovky.

Príklad:
N=5
`1 3` – prepájam káblom svorkovnice 1 a 3, žiadna žiarovka nesvieti
`0 3` - prepájam káblom svorkovnice 0 a 3, napätie je na svorke 3 (priamo z 0) a 1 (cez svorku 3), svietia žiarovky 1 a 3
`3 2` - prepájam káblom svorkovnice 3 a 2, napätie z 3 sa dostane na 2 a svietia žiarovky 1, 2, 3

Hodnotenie: 6b za riešenie v polynomiálnom čase + 7b za riešenie v čase $O(n^2)$.

Návraty k midtermu I (5 bodov)

Vieme, že platí toto tvrdenie: **„***V čase $\underline{O(n.logn)}$ môžeme zistiť, koľko hodnôt sa v n-prvkovom poli vyskytuje práve raz.*“

Naprogramujte metódu, ktorá takýto výpočet v uvedenom (aj priemernom) čase realizuje.

```
public int pocetJedinecnych(int[] p)
```

Návraty k midtermu II (5 bodov)

Vieme, že platí toto tvrdenie: **„***Máme 2 binárne vyhľadávacie stromy, pričom každý obsahuje n hodnôt. V čase $\underline{O(n)}$ vieme zistiť, či tieto binárne vyhľadávacie stromy obsahujú rovnaké prvky.*“

Do triedy `BVS` z prednášky pridajte metódu, ktorá vráti, či iný binárny vyhľadávací strom obsahuje rovnaké hodnoty (počty hodnôt môžu byť oproti tvrdeniu aj rôzne).

```
public boolean maRovnakeHodnoty(BVS inyStrom)
```