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 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;
    }
}
```

Návraty k midtermu (6 bodov)

Vieme, že platí toto tvrdenie: **„***Máme 2 usporiadané (neklesajúce) postupnosti (polia)* $\mathcal{A}$ *veľkosti* n *a* $\mathcal{B}$ *veľkosti* m*. Potom v čase* $O(n+m)$ *možno zistiť počet rôznych hodnôt s tou vlastnosťou, že sa nachádzajú v oboch postupnostiach, t.j. aj v* $\mathcal{A}$ *aj v* $\mathcal{B}$*.*“

Naprogramujte metódu, ktorá v uvedenom čase zistí počet rôznych hodnôt, ktoré sa nachádzajú v oboch poliach. Metóda nesmie modifikovať referencované polia. Akceptovaná pamäťová zložitosť je $O(1)$.

```
public int velkostPrieniku(int[] a, int[] b)
```

Príklad: $a = [1, 1, 2, 5, 5, 5, 9, 17, 20]$, $b = [1, 4, 5, 5, 8, 9, 9, 21, 21, 21]$ -> 3

Cézarove légie (18 bodov, dynamické programovanie)

Gaius Július Cézar pred každou bitkou premyslene zostavoval prvú líniu vojakov. Jej sila v mnohom určovala priebeh bojov. Prvú líniu Cézarovej armády tvorilo n_1 pešiakov a n_2 jazdcov. Aby na každom mieste línie bola dostatočná diverzita, línia musela byť zostavená tak, že žiadnych viac ako k_1 pešiakov a žiadnych viac ako k_2 jazdcov nestálo v rade pri sebe. Koľko rôznych prvých línií spĺňajúcich uvedenú podmienku môže Cézar zostaviť? Vytvorte program, ktorý tento počet zistí.

Poznámka: Pešiakov navzájom považujeme za nerozlíšiteľných. Rovnako sú navzájom nerozlíšiteľní jazdci.

Príklady (1 označuje pešiakov, 2 jazdcov):

- $n_1 = 2, n_2 = 1, k_1 = 1, k_2 = 10$: výsledok je 1, keďže existuje len jedno prípustné zostavenie prvej línie: 121
- $n_1 = 2, n_2 = 3, k_1 = 1, k_2 = 2$: výsledok je 5, keďže existujú tieto prípustné zostavenia prvej línie: 12122, 12212, 21212, 21221, 22121
- $n_1 = 2, n_2 = 4, k_1 = 1, k_2 = 1$: výsledok je 0

Rada: Označme si `P[i][j]` počet takých rôznych rozostavení i pešiakov a j jazdcov, v ktorých je posledný vojak v rade pešiak. Podobne si označme `R[i][j]` počet takých rôznych rozostavení i pešiakov a j jazdcov, v ktorých je posledný vojak v rade jazdec. Zoberme si teraz nejaké jedno prípustné rozostavenie $i + j$ vojakov končiace pešiakom. Nech x je počet pešiakov za sebou v rade na konci tohto rozostavenia (radu). Keďže rozostavenie bolo prípustné, zjavne $1 \leq x \leq k_1$. Ak týchto x pešiakov z konca rozstavenia (radu) dáme preč, dostaneme nejaké jedno prípustné rozostavenie $i + j - x$ vojakov končiace jazdcom. Pozorovanie: Každému prípustnému rozostaveniu $i + j$ vojakov končiacemu práve x pešiakmi zodpovedá práve jedno prípustné rozostavenie $i + j - x$ vojakov končiace jazdcom a naopak (t.j. máme bijekciu a mohutnosti množín sú rovnaké). Analogická situácia platí pre nejaké rozostavenie $i + j$ vojakov končiace jazdcom.

Bipartitné grafy (12 bodov, grafy)

Zaujímavou skupinou grafov (aj z pohľadu praxe) sú takzvané bipartitné grafy. Aké to sú grafy? Graf G nazveme bipartitným, ak množinu jeho vrcholov $V(G)$ môžeme rozdeliť na dve disjunktné podmnožiny X a Y, t.j. $X \cap Y = \emptyset, X \cup Y = V(G)$, s takou vlastnosťou, že pre každú hranu grafu platí, že jeden jej koniec je v množine X a druhý v množine Y.

Úloha: Vytvorte program, ktorý pre zadaný neorientovaný (potenciálne aj nesúvislý) graf overí, či je tento graf bipartitný. Graf načítajte z textového súboru, formát si zvoľte podľa vlastného uváženia.

Rada: Nech G je súvislý bipartitný graf. Ak si zoberieme ľubovoľný vrchol $s \in X$, tak jeho susedia (ak nejakých má) sú z množiny Y. Toto pozorovanie ide rozšíriť. Ak si zoberieme ľubovoľný vrchol $s \in X$, potom vrcholy, ktorých vzdialenosť od s je párna, sú v množine X a vrcholy, ktorých vzdialenosť od s je nepárna, sú v množine Y. Čo ale s hocijakým súvislým grafom? Nuž ak by takýto graf mal byť bipartitný, potom po rozdelení vrcholov do množín X a Y podľa vzdialenosti od zvoleného vrcholu by malo platiť, že v grafe niet hrany, ktorá by mala oba konce buď v X alebo Y.

Stromovač (8 + 6 bodov, stromy)

Uvažujme triedu `Osoba` z prednášky o stromoch:

```
public class Osoba {
	private String meno;
	private List<Osoba> deti = new ArrayList<Osoba>();

	public Osoba(String meno) {
		this.meno = meno;
	}

	public void pridajDieta(Osoba dieta) {
		deti.add(dieta);
	}

	public void vypisStrom(String cestaKRodicovi) {
		String cesta = cestaKRodicovi + meno;
		System.out.println(cesta);
		for (Osoba dieta : deti) {
			dieta.vypisStrom(cesta + "/");
		}
	}

	public void vypisStrom2(String cestaKRodicovi) {
		String cesta = cestaKRodicovi + meno;
		if (deti.size() == 0) {
			System.out.println(cesta);
		}
		for (Osoba dieta : deti) {
			dieta.vypisStrom2(cesta + "/");
		}
	}

}
```

Ak zavoláme metódu `vypisStrom("")` na koreni stromu, v každom riadku sa vypíše cesta k nejakému uzlu tohto stromu.

Do triedy `Osoba` pridajte metódu `vytvorStrom`, ktorá vráti referenciu na koreň novovytvoreného stromu podľa zadaného textového súboru. Vytvorený strom má mať tú vlastnosť, že ak na jeho koreni zavoláme metódu `vypisStrom("")`, dostaneme výpis totožný s obsahom tohto textového súboru. Môžete predpokladať, že súbor obsahuje korektný vstup.

```
public static Osoba vytvorStrom(File suborSPopisomStromu)
```

Hodnotenie: +6 bodov, za verziu s výstupom vytvoreným volaním `vypisStrom2("")` na koreni stromu.