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.

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.

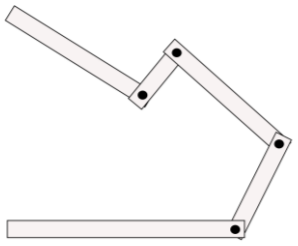
Návraty k midtermu (6 bodov)

Vieme, že platí toto tvrdenie: **„***V čase $O(n.logn)$ môžeme zistiť, počet rôznych hodnôt v n-prvkovom poli.*“

Naprogramujte metódu, ktorá v uvedenom čase vypočíta počet rôznych hodnôt uložených v poli. Metóda nesmie modifikovať referencované pole `p`. Akceptovaná pamäťová zložitosť je $O(n)$. Nie je dovolené používať triedy z JCF (napr. implementácie rozhrania `Set`).

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

Skladací meter (12 bodov, backtracking)



Firma *PAZomer* sa rozhodla vyrábať netradičné skladacie metre. Na rozdiel od tých klasických sa tieto metre neskladajú z dielov rovnakej dĺžky, ale každý diel môže mať inú dĺžku. Obchod sa rozbehol a prišlo mnoho objednávok. Problém však vznikol pri balení. Keďže je žiaduce, aby balenie bolo čo najmenšie, vznikla otázka: na akú najmenšiu dĺžku je možné poskladať meter (meradlo)? Vytvorte program, ktorý vyrieši túto otázku pre zadanú postupnosť dĺžok jednotlivých dielov skladacieho metra a vráti poskladanie, v ktorom sa dosahuje táto najmenšia dĺžka. Vstup načítajte z textového súboru.

Príklad: Ak sa meter skladá z dielov s dĺžkami 15 10 12, tak najmenšie možné poskladanie má dĺžku 17.

Rada: Ak sa meter skladá z n dielov, tak obsahuje $n-1$ kĺbov. Každé poskladanie metra je určené zalomením/nezalomením jednotlivých kĺbov.

Textový editor (12+5 bodov, grafové algoritmy)

Obzvlášť linuxáci obľubujú konzolu – možno aj preto sa tak stránia myšiek a pri všetkom si vystačia s klávesnicou. Pri práci s textom používajú v termináloch jednoduché textové editory. Všimli ste si, ako vlastne funguje presun kurzora klávesnicou len s použitím kurzorových šípok nahor, nadol, vľavo a vpravo (odporúčame spustiť Notepad)? Sumarizácia pozorovaní:

- ak je v riadku l znakov, potom sa kurzor môže nachádzať na jednej z $l+1$ pozícii (číslovaných od 0, pričom pozícia 0 je pred prvým znakom a l je pozícia za posledným znakom),
- každú pozíciu kurzora môžeme charakterizovať ako dvojicu (r, c), kde r je číslo riadka (číslujeme od 0) a c je pozícia kurzora v rámci tohto riadka,
- ak stlačíte šípku vľavo a kurzor nie je na začiatku súboru, t.j. na pozícii $(0, 0)$, tak sa kurzor presunie o jeden znak vľavo, resp. na po koniec predošlého riadka, ak sa pred stlačením nachádzal na začiatku riadka,
- ak stlačíte šípku vpravo a kurzor nie je na konci súboru, tak sa kurzor presunie o jeden znak vpravo, resp. na začiatok ďalšieho riadka, ak sa pred stlačením nachádzal na konci riadka,
- ak stlačíte šípku nahor a kurzor nie je v prvom riadku, tak sa kurzor presunie na rovnakú pozíciu v riadku vyššie – ak má riadok vyššie menej znakov ako je pozícia kurzora, kurzor sa presunie na koniec tohto riadka,
- ak stlačíte šípku nadol a kurzor nie je v poslednom riadku, tak sa kurzor presunie na rovnakú pozíciu v riadku nižšie – ak má tento riadok menej znakov ako je pozícia kurzora, kurzor sa presunie na koniec tohto riadka.

Uvažujme súbor s n riadkami, pričom v riadku i je `l[i]` znakov (vstupom môže byť n-prvkové pole). Pre zadanú začiatočnú pozíciu kurzora (r_s, c_s) a koncovú pozíciu kurzora (r_k, c_k) vypočítajte minimálny počet stlačení uvažovaných štyroch kurzorových šípok, ktoré sú potrebné, aby sme kurzor presunuli zo začiatočnej pozície do cieľovej pozície.

Rada: Každú pozíciu kurzora môžete reprezentovať ako vrchol grafu. Medzi vrcholmi grafu je hrana práve vtedy, keď sa medzi prislúchajúcimi pozíciami dá presunúť stlačením jednej z kurzorových šípok. Všimnite si, že v takomto grafe má každý vrchol stupeň nanajvýš 4.

Hodnotenie: 12 bodov za vrátenie minimálneho počtu stlačení kláves, 5 bodov za vrátenie toho, aké klávesy treba stláčať, aby sa dosiahlo vypočítané minimum.

Stromovač (10 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(int uroven) {
		for (int i = 0; i < uroven; i++) {
			System.out.print('*');
		}
		System.out.println(meno);
		for (Osoba dieta : deti) {
			dieta.vypisStrom(uroven + 1);
		}
	}
}
```

Ak zavoláme metódu `vypisStrom(0)` na koreni stromu, dostaneme textový zápis obsahu tohto stromu. Je to teda spôsob ako nejaký všeobecný strom zakódovať.

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(0)` dostaneme výpis totožný s obsahom tohto textového súboru. Môžete predpokladať, že súbor obsahuje korektný vstup.

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