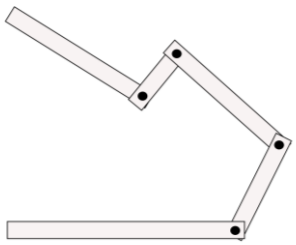
Záverečný test
praktická časť

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.

Skladací meter (10 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 žiadúce, 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. Zamyslite sa, ako na základe zalomenia vypočítať potrebnú dĺžku balenia.

Žiarovky (11 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: 4 b za riešenie v polynomiálnom čase + 7b za riešenie v čase $O(n^2)$.

Návraty k midtermu (5 bodov)

Vieme, že platí toto tvrdenie: **„***Máme 2 usporiadané (neklesajúce) postupnosti (polia)* A *veľkosti* n *a* 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* A *aj v* 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

Sústava zlomkových rovníc (15 bodov, grafové algoritmy)

Uvažujme sústavu zlomkových rovníc tvaru $x/y = r$, kde čitateľ (x) aj menovateľ (y) sú názvy premenných, t.j. reťazce bez medzier, a r je konkrétne číslo s desatinnou čiarkou. Na základe týchto rovníc vyriešte zadanú zlomkovú rovnicu. Výsledkom nech je `Double.NaN`, ak rovnicu nie je možné vyriešiť na základe informácii, ktoré máme.

Príklad:

- Sústava zlomkových rovníc: $a\ /\ b\ =\ 2.0, b\ /\ c\ =\ 3.0$
- Rovnice na riešenie: $a\ /\ c\ =?, b\ /\ a\ =?, d\ /\ a\ =?$
- Riešenia: $a\ /\ c\ =\ 6, b\ /\ a\ =\ 0.5, d\ /\ a\ =\ Double.NaN$

Vstup: Sústavu rovníc je možné popísať textovým súborom, v ktorom každý riadok predstavuje jednu rovnicu. Jeden riadok sa skladá z 3 medzerami oddelených častí, ktoré postupne reprezentujú meno premennej v čitateli, meno premennej v menovateli a hodnotu podielu. Príklad súboru:

```
a b 2.0
b c 3.0
```

Iný príklad súboru:

```
cit men 3.0
a men 2.0
cit a 1.5
```

Rovnice na riešenie môžete načítať zo súboru alebo z konzoly.

Rada: Všimnite si, že (1) ak poznáme hodnotu a/b, ľahko si vieme vypočítať hodnotu výrazu b/a, (2) ak poznáme hodnotu a/b a hodnotu b/c, potom poznáme aj hodnotu a/c pretože $a/b * b/c = a/c$. Druhú implikáciu možno zovšeobecniť: Ak poznáme $a/b, b/c, c/d$, potom poznáme aj $a/d = a/b * b/c * c/d$.

Môžete predpokladať, že žiadna z premenných nie je 0.

Nocľah (6 + 7 bodov, dynamické programovanie)

Juraj sa cez prázdniny chystá na dlhú túru (putovanie) naprieč jedným kanadským národným parkom. Cez deň putuje a obdivuje krásy prírody, živí sa lesnými plodmi a pije vodu z prameňov. V noci zloží hlavu v niektorej z chát, ktoré sú pozdĺž jeho trasy (ostávať na noc v prírode je dosť nebezpečné). Za nocľah sa platí a každá chata má inú cenu za prenocovanie. Chaty sú popri trase umiestnené tak, aby vzdialenosť medzi nimi bolo možné prejsť za pol dňa. Kým času má Juraj počas prázdnin dosť, s peniazmi na prenocovanie to až také optimistické nie je.

Popri trase je umiestnených N chát, pričom poznáme ceny za prenocovanie v nich. Označme si C[i] cenu za prenocovanie v i-tej chate. Keďže vzdialenosť medzi chatami je pol dňa, ak Juraj prenocuje v i-tej chate, potom ďalšiu noc môže stráviť v chate i+1 alebo i+2. Zostavte pre Juraja plán cesty, t.j. plán, v ktorých chatách má prenocovať, aby ho túra vyšla čo najlacnejšie.

Hodnotenie: 6 bodov za vypočítanie minimálnej celkovej ceny za prenocovania, 7 bodov za plán cesty, pri ktorom sa táto minimálna celková cena dosahuje.

Rada: Označme si R[i] minimálnu celkovú cenu, ktorú je potrebné zaplatiť za prenocovania na trase po i-tu chatu, pričom sme v i-tej chate prenocovali.

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.

Stromový iterátor (20 bodov)

Do triedy BVS z prednášky o binárnych vyhľadávacích stromoch pridajte triedu `StromovyIterator`, ktorá umožní iterovať hodnoty uložené v binárnom vyhľadávacom strome a implementuje rozhranie `Iterator`. Hodnoty nech sú iterované v in-order postupnosti.

```
public static class StromovyIterator implements Iterator<Integer> {
  private Uzol koren;

  public StromovyIterator(BVS bvs) {
    koren = bvs.koren;
  }

  @Override
  public boolean hasNext() {
    return false;
  }

  @Override
  public Integer next() {
    return null;
  }
}
```

Uvedený testovací kód by mal vypísať postupnost inorder prechodu stromom – teda rastúcu postupnosť hodnôt:

```
BVS strom = new BVS();
// naplnenie stromu …
StromovyIterator it = new StromovyIterator(strom);
while (it.hasNext()) {
  System.out.println(it.next());
}
```

Hodnotenie: Očakáva sa implementácia s pamäťovou zložitosťou $O(h)$, kde h je výška stromu, pričom celková časová zložitosť iterovania je dokopy $O(n)$, kde n je počet hodnôt uložených v strome.

Rada: Iterátor je v každom okamihu na nejakom uzle stromu pričom začíname na najľavejšom liste stromu. Operácia `next` presunie „ukazovateľ“ (referenciu na uzol, v ktorom sa nachádzame) na ďalší uzol v inorder postupnosti. Aký je to uzol? Ako vyzerá cesta k nemu? Keďže uzly stromu uchovávajú len referencie na deti, je potrebné si nejako „bokom” pamätať informáciu, kto je rodičom istých z uzlov stromu (metafora: „istiace lano z koreňa“).