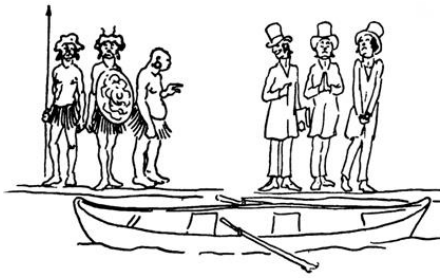
Záverečný test
praktická časť

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.

Prievoznícke úlohy (18 bodov, grafové algoritmy)



Traja misionári a traja kanibali sa stretli na jednom brehu rieky. Na brehu bola malá loďka, na ktorú sa zmestia maximálne dvaja. Všetci by sa chceli prepraviť na druhý breh, ale na žiadnom brehu nesmie nikdy zostať prevaha kanibalov nad misionármi, inak by mohlo dôjsť k tragédii.

Túto úlohu ide riešiť logickým uvažovaním, ale aj využitím grafových algoritmov. Nás zaujíma druhý spôsob. Všimnite si, že po každom presune loďky medzi brehmi vieme umiestnenie misionárov i kanibalov popísať trojicou čísel (M, K, B). Číslo M je počet misionárov na prvom brehu (na druhom brehu ich je teda $3 - M$) a K je počet kanibalov na prvom brehu (na druhom ich je $3 - K$). Číslo B sa rovná 1, ak sa loďka nachádza pri prvom brehu a 2, ak sa nachádza pri druhom brehu. Východiskovú situáciu (stav) vieme popísať trojicou $(3, 3, 1)$. Našim cieľom je dosiahnuť stav $(0, 0, 2)$.

Každý presun ľudí loďkou znamená, že sa mení stav. Ak sa zo stavu $(3, 2, 1)$ presunie jeden misionár na druhý breh, dostaneme sa do stavu $(3 - 1, 2, 2)$, teda $(2, 2, 2)$. Podobne, zo stavu $(2, 2, 2)$, kedy je loďka pri druhom brehu, na ktorom je jeden misionár a jeden kanibal, môžeme prejsť preplavením jedného misionára a jedného kanibala loďkou do stavu $(3, 3, 1)$.

Samozrejme nie všetky stavy sú pre prievozníckú úlohu vyhovujúce. Pre vyhovujúce stavy (M, K, B) musí platiť: $M = 0 \vee M = 3 \vee (M \geq K \wedge 3 - M \geq 3 - K)$.

Uvažujme graf, ktorého vrcholová množina je tvorená vyhovujúcimi stavmi prievozníckej úlohy a hrana je medzi vrcholmi práve vtedy, ak sa z jedného stavu vieme dostať do druhého jedným preplavením loďky medzi brehmi. Upozorňujeme, že pri preplavení loďky medzi brehmi v nej musí sedieť aspoň jeden človek, no nie viac ako dvaja ľudia. Všimnime si, že cesta v tomto grafe zodpovedá nejakej bezpečnej postupnosti preplavení osôb loďkou medzi brehmi.

Vytvorte program, ktorý pre zadaný východiskový stav popísaný trojicou čísel vyššie uvedeným spôsobom nájde minimálny počet preplavení potrebný na prechod do cieľového stavu $(0, 0, 2)$, resp. vráti, že taká postupnosť preplavení neexistuje. Ak taká postupnosť existuje, program nech vráti plán, ktorý preplavenie misionárov a kanibalov z východiskového stavu zrealizuje s minimálnym počtom preplavení.

Hodnotenie:

- 7b za minimálny počet preplavení,
- 5b za plán preplavení,
- 6b za za riešenie s ľubovoľne zadaným počtom misionárov a kanibalov (teda uvažujeme úlohu s X misionármi, X kanibalmi a dvojmiestnou loďkou; v základnej verzii je X = 3).

Hra Logic (14 bodov, backtracking)

Cieľom logickej hry Logic je odhaliť skrytý, tajný kód čo s najmenším počtom pokusov za pomoci súperovej nápovedy. Tajný kód zostaví hráč A z piatich farebných kolíkov, hráč B sa tajný kód pokúša uhádnuť svojou farebnou kombináciou. Ak uhádne farbu aj pozíciu jednotlivých kolíkov tajného kódu, hráč A umiestni do tohto poľa toľko červených informačných kolíkov, koľko ich súper uhádol. Ak uhádol iba farbu a nie pozíciu, hráč A umiestni do informačného poľa iba žltý kolík. Hráč B má na uhádnutie tajného kódu maximálne 10 pokusov. Po skončení hry sa pozície vymenia a vyhráva ten, komu sa podarilo odhaliť tajný kód s menším počtom kolíkov. Farebné kolíky sú jednej z ôsmich farieb – kvôli jednoduchosti ich budeme číslovať 0..7.

Jeden pokus hráča B môžeme popísať sedmicou čísel (napr. v riadku). Konkrétne prvých 5 čísel sú čísla farieb, ktoré hráč hádal (v rozsahu 0..7). Šieste číslo je počet červených informačných kolíkov (0..5) a siedme číslo je počet žltých kolíkov (0..5), ktoré umiestnil hráč A do informačného poľa.

Vytvorte program, ktorý prečíta riadky (zo súboru alebo z konzoly) s popisom jednotlivých pokusov hráča B. Program následne nájde všetky možné tajné kódy, ktoré mohol hráč A vytvoriť a zároveň zodpovedajú počtom farebných kolíkov v informačnom poli pri jednotlivých pokusoch hráča B.

Rada: V najjednoduchšej verzii stačí vygenerovať všetky tajné kódy a pre každý overiť, či vyhovuje výsledkom jednotlivých pokusov.

Noemova archa (10+6+8 bodov, dynamické programovanie)

Istotne poznáte príbeh o Noemovi, ktorý postavil archu a nalodil na ňu z každého živočíšneho druhu jeden pár. Archa mala po svojom obvode v niekoľkých poschodiach umiestnené miestnosti – „kajuty“ pre zvieratá. Ale nalodiť zvieratá na archu to nie je len tak. Rôzne zvieratá majú rôzne hmotnosti. Ak by Noe nepostupoval pri umiestňovaní zvierat do „kajút“ archy uvážene, ľahko by sa mohlo stať, že sa archa preváži a prevráti sa na jeden bok. Keďže pred Noemom bola dlhá plavba po rozbúrenom mori, bolo nevyhnutné nájsť také umiestnenie zvierat do „kajút“, aby súčet váh zvierat na ľavoboku sa od súčtu váh zvierat na pravoboku čo najmenej líšil. Noe si teda vytvoril zoznam zvierat a pre každý pár si do tohto zoznamu poznačil aj váhu tohto páru (pár bol samozrejme umiestnený v rovnakej „kajute“). Váha každého páru bola uvedená v celých kilogramoch. Zvieratá, ktoré vážili menej ako kilogram, Noe „neriešil“, keďže na ich váhe až tak pri prevažovaní nezáležalo (ale také hrochy, slony, kone, ... tam tú váhu nemôžeme ignorovať).

Úloha: Vytvorte program, ktorý načíta Noemov zoznam a nájde (a vypíše) najoptimálnejšie rozdelenie zvierať do kajút na ľavoboku a pravoboku.

Noemov zoznam (druh, váha páru v kilogramoch):

```
Slony 24000
Kone 900
Hrochy 3500
```

Hodnotenie:

- 10 bodov za nájdenie optimálneho súčtu váh na ľavoboku a na pravoboku
- 6 bodov za nájdenie optimálneho riešenia s pamäťovou zložitosťou memoizačnej tabuľky závislou od počtu živočíšnych druhov
- 8 bodov za nájdenie optimálneho riešenia s pamäťovou zložitosťou memoizačnej tabuľky nezávislou od počtu živočíšnych druhov

Stromovač (10 bodov, stromy)

Uvažujme triedu `Uzol` z prednášky o binárnych stromoch:

```
public class Uzol {
	private int hodnota;
	private Uzol lavy;
	private Uzol pravy;

	public Uzol(int hodnota, Uzol lavy, Uzol pravy) {
		this.hodnota = hodnota;
		this.lavy = lavy;
		this.pravy = pravy;
	}

	public void vypisStrom(String cesta) {
		System.out.println(hodnota + " " + cesta);

		if (lavy != null) {
			lavy.vypisStrom(cesta + "L");
		}

		if (pravy != null) {
			pravy.vypisStrom(cesta + "P");
		}
	}

	public int getHodnota() {
		return hodnota;
	}

	public void setHodnota(int hodnota) {
		this.hodnota = hodnota;
	}
}
```

Ak zavoláme metódu `vypisStrom("")` na koreni stromu, dostaneme výpis všetkých uzlov v tomto strome. V tomto zápise je každý uzol charakterizovaný svojou hodnotou a navigačnou cestou (L-pokračuj v ľavom synovi, P-pokračuj v pravom synovi) k nemu od koreňa stromu.

Do triedy `Uzol` pridajte statickú 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 Uzol vytvorStrom(File suborSPopisomStromu)
```

Návraty k midtermu (5 bodov)

Vieme, že platí toto tvrdenie: **„***V čase* $O(n.logn)$ *môžeme zistiť, ktorá hodnota sa v n-prvkovom poli vyskytuje najväčší počet krát.*“

Naprogramujte metódu, ktorá v uvedenom (aj priemernom) čase nájde tú hodnotu v poli, ktorá sa v ňom vyskytuje najväčší počet krát. Ak je takých hodnôt viac, metóda nech vráti ľubovoľnú z nich. Metóda nesmie modifikovať referencované pole p. Akceptovaná pamäťová zložitosť je $O(n)$.

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