Facultatea de Matematică şi Informatică
Algoritmi şi Structuri de Date – Laborator
Anul I, semestrul I, an universitar 2018/2019

Tema 9
4 decembrie 2018

Probleme obligatorii

Termen de predare : Laboratorul din săptămâna 12 (17-21 decembrie 2018)

(2 p) **1.** Sa se implementeze algoritmul *randomized quick-sort* (alegerea pivotului se va face aleator).

(3 p) **2.** Să se scrie algoritmul pentru sortarea unui şir de numere folosind metoda Heapsort. Structura de Heap va fi implementată ca un arbore binar **într-una** din cele două forme care urmează :

a) max - Heap – arbore binar în care fiecare nod are cheia mai mare decât oricare dintre fiii săi.
b) min - Heap – arbore binar în care fiecare nod are cheia mai mică decât oricare dintre fiii săi.

Scrieţi funcţii pentru crearea heap-ului şi pentru decapitarea lui.

(3 p) **3.** Să se implementeze o coadă cu priorităţi folosindu-se un heap (T. Cormen, C. Leiserson, R. Rivest, C. Stein – *Introduction to algorithms*, 3rd edition, capitolul 6.5, pag.162). Elementele cozii vor avea doua câmpuri: prioritate şi cheie. Vor exista urmatoarele operatii:

- `insert(q, x)` care insereaza nodul x in coada q;
- `maximum(q)` care intoarce elementul de prioritate maxima din coada q;
- `extract_max(q)` care intoarce elementul de prioritate maxima din q, eliminându-l din coadă.

Probleme suplimentare

Termen de predare : Laboratorul din săptămâna 12 (17-21 decembrie 2018)

(2 p) **4**. Să se implementeze algoritmul *Shell-Sort* folosind ca tablou de incrementi unul dintre şirurile propuse în materialul ajutător alăturat.

(2 p) **5.** Sa se optimizeze procedura de *sortare rapidă*, folosind următoarea tehnică: subşirurile de dimensiune ≤ 11 elemente se sortează cu inserţia directă (Insertion Sort).

(4 p) **6. Roata**

Una dintre atracţiile celebrului parc de distracţii Prater din Viena este Marea Roată Vieneză. Din ea se poate admira priveliştea întregii Viene.

Roata are *n* cabine, numerotate de la 1 la *n* în sens orar şi dispuse simetric pe circumferinţa roţii. Îmbarcarea clienţilor se face în cabina în care roata este tangentă cu solul, iar rotirea începe cu cabina 1 aflată în poziţia de îmbarcare şi se face în sens antiorar. Un client plăteşte pentru o rotire 1 EUR şi poate cumpăra un număr oarecare de rotiri.

Cei *p* clienţi care doresc utilizarea roţii trebuie să respecte următoarea procedură: clientul cu numărul de ordine *i* îşi cumpără un bilet pe care sunt înscrise numărul său de ordine şi numărul de rotiri c_i, $1 \le i \le p$, apoi se aşează la rând. Când în poziţia de îmbarcare este o cabină liberă sau se eliberează o cabină, roata se opreşte şi urcă următorul clientul. Un client coboară după ce se efectuează numărul de rotiri înscris pe bilet.

Cerinţă

Să se scrie un program care, cunoscând numărul *n* de cabine al roţii, numărul *p* de clienţi, precum şi numărul de rotiri cumpărate de fiecare client, c_i, $1 \le i \le p$, să calculeze:

- suma totală încasată de administratorul roţii de la clienţi;
- ordinea în care coboară clienţii din roată;
- numărul cabinei din care coboară ultimul client.

Date de intrare

Fişierul de intrare *roata.in* conţine pe primul rând numărul natural *n*, pe al doilea rând numărul natural *p* iar pe al treilea rând numerele naturale c_i, $1 \le i \le p$, separate printr-un spaţiu, cu semnificaţiile de mai sus.

Date de ieşire

Fişierul de ieşire *roata.out* va conţine pe prima linie suma totală încasată, pe a doua linie numerele de ordine ale clienţilor, în ordinea coborârii, separate printr-un spaţiu, iar pe a treia linie numărul cabinei din care va coborî ultimul client.

Restricţii

- $2 \le n \le 360$
- $1 \le p \le 100\,000$
- $1 \le c_i \le 100\,000$

Exemplu

roata.in	roata.out	Explicaţie
4 7 6 4 1 5 2 8 3	29 3 5 2 4 1 7 6 3	Roata are n = 4 cabine şi numărul de clienţi este p = 7. Primul client cumpără 6 rotiri, al doilea 4 rotiri , ... , iar al şaptelea client cumpără 3 rotiri. Suma totală încasată este de 29 EUR. După ce primii 4 clienţi se urcă în roată

		şi se efectuează o rotire completă, primul care coboară este clientul al 3-lea şi imediat se urcă clientul al 5-lea. După încă 2 rotiri, clientul al 5-lea coboară şi se urcă clientul al 6-lea. După încă o rotire coboară clientul al 2-lea şi se urcă al 7-lea client. Ultimii 4 clienţi coboară în ordinea 4, 1, 7, 6. Cabina din care coboară ultimul client este cabina cu numărul 3

OJI 2012 - clasa a 9-a

(3 p) 7. La coadă

La BIG au băgat pui. Instantaneu s-a format o coadă de N persoane, numerotate în ordine de la 1 la N. La coadă se pot întâmpla următoarele lucruri:

1. Servire: prima persoană de la coadă primeşte un pui şi pleacă acasă.
2. Sosire: la coadă se mai aşează o persoană. Noii veniţi sunt numerotaţi în continuare: N + 1, N + 2 ş.a.m.d.
3. Îmbrâncire(x): persoana numărul x face rost de o relaţie şi se îmbrânceşte până pe prima poziţie a cozii. Dacă persoana era deja prima, nu se schimbă nimic.

Se dă o listă de K operaţii. Să se spună care este configuraţia finală a cozii. Se garantează că în niciun moment lungimea cozii nu va depăşi N (oamenii se descurajează dacă văd o coadă prea lungă şi nu se mai aşează). Se garantează că operaţiile de servire şi îmbrâncire nu se vor efectua pe o coadă goală.

Date de intrare

Fişierul de intrare lacoada.in conţine pe prima linie numerele N şi K. Pe următoarele K linii se vor găsi operaţiile, numerotate ca mai sus, într-una din formele

```
1
2
3 x
```

Se garantează că x este numărul unei persoane din coadă.

Date de ieşire

În fişierul de ieşire lacoada.out se va tipări pe prima linie lungimea cozii la sfârşitul operaţiilor. Pe a doua linie se vor tipări, în ordine, numerele persoanelor de la coadă, începând cu prima.

Restricţii

- 1 ≤ N ≤ 60.000
- 1 ≤ K ≤ 1.000.000

Exemplu

lacoada.in	lacoada.out	Explicaţie
6 6 3 5 1 3 3 2 3 7 1	5 3 1 2 4 6	5 se îmbrânceşte, coada devine 5 1 2 3 4 6 5 este servit, coada devine 1 2 3 4 6 3 se îmbrânceşte, coada devine 3 1 2 4 6 7 soseşte, coada devine 3 1 2 4 6 7 7 se îmbrânceşte, coada devine 7 3 1 2 4 6 7 este servit, coada devine 3 1 2 4 6

Autor: Cătălin Frâncu

Probleme facultative

Termen de predare : Laboratorul din săptămâna 11 (10-15 decembrie 2018)

(5 ps) **1.** Spunem ca o tabla de sah de 2^k x 2^k patrate este defecta, daca unul din cele 2^{2^k} patrate lipseste. Problema va cere sa acoperiti o astfel de tabla cu tromino-uri (Figura 1), astfel incat oricare doua tromino-uri nu se suprapun, ele nu acopera patratul lipsa, dar acopera toate celelalte patrate. Sugestii de implementare:

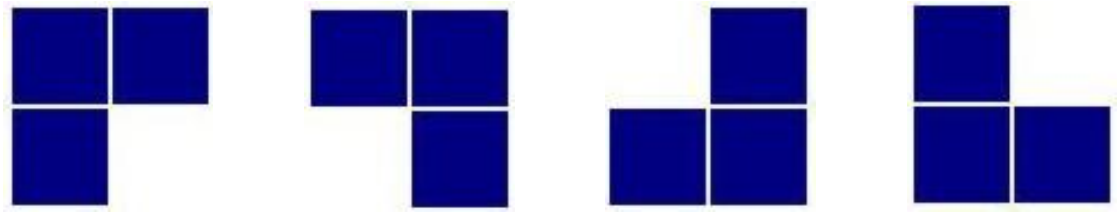
(a) o acoperire a unei table m x m se poate reprezenta printr-o matrice `Tabla[m][m]`, unde `Tabla[i][j]` indica numarul trominoului cu care este acoperit patratul `(i; j)`.

(b) Functia recursiva ce construieste solutia poate fi de forma: `Acopera(rt,ct,rd, cd,latura)`, unde :

i. `rt, ct` reprezinta randul si coloana patratului din coltul stanga sus al portiunii patratice de tabla ce trebuie acoperita;

ii. `rd, cd` reprezinta randul si coloana patratului lipsa;

iii. latura reprezinta latura portiunii patratice de tabla ce trebuie acoperita.



(a) (b) (c) (d)

Figura 1. Tromino-uri

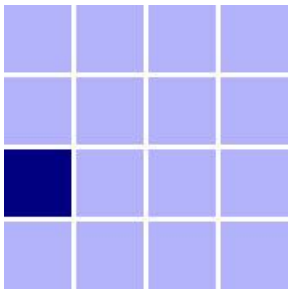


Figura 2. O tablă de şah defectă de dimensiuni $2^2 \times 2^2$