

Tema 9
5 decembrie 2017

Probleme obligatorii

Termen de predare : Laboratorul din săptămâna 13 (11 ianuarie 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 șir 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.

Scrieți funcții pentru crearea heap-ului și pentru decapitarea lui.

(3 p) **3.** Sa se implementeze o coadă cu priorități folosindu-se un heap (Cormen, capitolul 7.5). Elementele cozii vor avea doua câmpuri: prioritate și cheie. Vor exista următoarele operații:

- `insert(q, x)` care inserează nodul x în coada q;
- `maximum(q)` care întoarce elementul de prioritate maxima din coada q;
- `extract_max(q)` care întoarce elementul de prioritate maxima din q, eliminându-l din coadă.

Probleme suplimentare

Termen de predare : Laboratorul din săptămâna 13 (11 ianuarie 2018)

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

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

(4 p) 6. Roata

Una dintre atracțiile celebrului parc de distracții Prater din Viena este Marea Roată Veneză. Din ea se poate admira priveliștea întregii Viene.

Roata are n cabine, numerotate de la 1 la n în sens orar și dispuse simetric pe circumferința roții. Îmbarcarea clienților se face în cabina în care roata este tangentă cu solul, iar rotirea începe cu cabina 1 aflată în poziția de îmbarcare și se face în sens antiorar. Un client plătește pentru o rotire 1 EUR și poate cumpăra un număr oarecare de rotiri.

Cei p clienți care doresc utilizarea roții trebuie să respecte următoarea procedură: clientul cu numărul de ordine i își cumpără un bilet pe care sunt înscrise numărul său de ordine și numărul de rotiri c_i , $1 \leq i \leq p$, apoi se așează la rând. Când în poziția de îmbarcare este o cabină liberă sau se eliberează o cabină, roata se oprește și urcă următorul clientul. Un client coboară după ce se efectuează numărul de rotiri înscris pe bilet.

Cerință

Să se scrie un program care, cunoscând numărul n de cabine al roții, numărul p de clienți, precum și numărul de rotiri cumpărate de fiecare client, c_i , $1 \leq i \leq p$, să calculeze:

- suma totală încasată de administratorul roții de la clienți;
- ordinea în care coboară clienții din roată;
- numărul cabinei din care coboară ultimul client.

Date de intrare

Fișierul de intrare `roata.in` conține pe primul rând numărul natural n , pe al doilea rând numărul natural p iar pe al treilea rând numerele naturale c_i , $1 \leq i \leq p$, separate printr-un spațiu, cu semnificațiile de mai sus.

Date de ieșire

Fișierul de ieșire `roata.out` va conține pe prima linie suma totală încasată, pe a doua linie numerele de ordine ale clienților, în ordinea coborârii, separate printr-un spațiu, iar pe a treia linie numărul cabinei din care va coborî ultimul client.

Restricții

- $2 \leq n \leq 360$
- $1 \leq p \leq 100\,000$
- $1 \leq c_i \leq 100\,000$
- pentru rezolvarea primei cerințe se acordă 20% din punctaj, iar pentru celelalte două cerințe se acordă câte 40% din punctaj fiecare.

Exemplu

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

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

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 primește un pui și pleacă acasă.
2. **Sosire:** la coadă se mai așează o persoană. Noii veniți sunt numerotați în continuare: $N + 1$, $N + 2$ ș.a.m.d.
3. **Îmbrâncire(x):** persoana numărul x face rost de o relație și se îmbrâncește până pe prima poziție a cozii. Dacă persoana era deja prima, nu se schimbă nimic.

Se dă o listă de K operații. Să se spună care este configurația finală a cozii. Se garantează că în niciun moment lungimea cozii nu va depăși N (oamenii se descurajează dacă văd o coadă prea lungă și nu se mai așează). Se garantează că operațiile de servire și îmbrâncire nu se vor efectua pe o coadă goală.

Date de intrare

Fișierul de intrare lacoada.in conține pe prima linie numerele N și K. Pe următoarele K linii se vor găsi operațiile, numerotate ca mai sus, într-una din formele

1

2

3 x

Se garantează că x este numărul unei persoane din coadă.

Date de ieșire

În fișierul de ieșire lacoada.out se va tipări pe prima linie lungimea cozii la sfârșitul operațiilor. Pe a doua linie se vor tipări, în ordine, numerele persoanelor de la coadă, începând cu prima.

Restricții

- $1 \leq N \leq 60.000$
- $1 \leq K \leq 1.000.000$

Exemplu

lacoada.in	lacoada.out	Explicație
6 6	5	5 se îmbrâncește, coada devine 5 1 2 3 4 6
3 5	3 1 2 4 6	5 este servit, coada devine 1 2 3 4 6
1		3 se îmbrâncește, coada devine 3 1 2 4 6
3 3		7 sosește, coada devine 3 1 2 4 6 7
2		7 se îmbrâncește, coada devine 7 3 1 2 4 6
3 7		7 este servit, coada devine 3 1 2 4 6
1		

Autor: Cătălin Frâncu

Probleme facultative

Termen de predare : Laboratorul din săptămâna 13 (11 ianuarie 2018)

(5 ps) 1. Spunem ca o tabla de sah de $2^k \times 2^k$ patrate este defecta, daca unul din cele 2^{2k} 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 $m \times m$ se poate reprezenta printr-o matrice $\text{Tabla}[m][m]$, unde $\text{Tabla}[i][j]$ indica numarul trominoului cu care este acoperit patratul $(i; j)$.

(b) Functia recursiva ce construiește solutia poate fi de forma: $\text{Acopera}(\text{rt}, \text{ct}, \text{rd}, \text{cd}, \text{latura})$, 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.

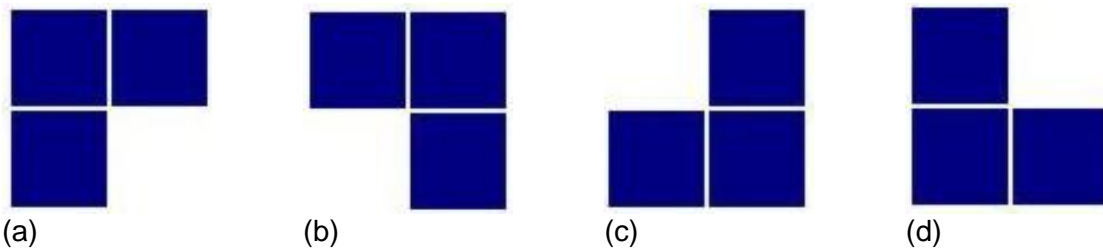


Figura 1. Tromino-uri

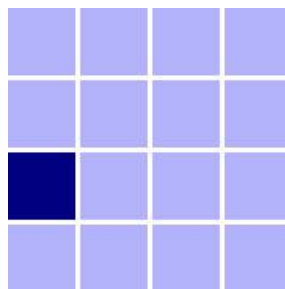


Figura 2. O tablă de șah defectă de dimensiuni $2^2 \times 2^2$