Facultatea de Matematică şi Informatică
Algoritmi şi Structuri de Date – Laborator
Anul II, semestrul I, an universitar 2019/2020
Seria: 25

Tema 6
17 decembrie 2019

Probleme suplimentare

Termen de predare : Laboratorul din săptămâna 14 (13-17 ianuarie 2020)

(5 p) **1.** Scrieţi un algoritm care să construiască un arbore Huffman pentru un alfabet cu ponderi dat, arbore reprezentat în aşa fel încât să poată fi folosit atât la codificare, cât şi la decodificare. Scrieţi proceduri care fac, la cerere, codificarea şi decodificarea

Algoritm de construcţie a arborelui Huffman
Pas 1. Iniţializare :
- fiecare caracter reprezintă un arbore format dintr-un singur nod;
- organizăm caracterele ca un min-heap, în funcţie de frecvenţele de apariţie;

Pas 2. Se repetă de n-1 ori :
- extrage succesiv X şi Y, două elemente din heap
- unifică arborii X şi Y :
  - crează Z un nou nod ce va fi rădăcina arborelui
  - Z^.st := X
  - Z^.dr := Y
  - Z^.frecv := X^.frecv+Y^.frecv
- inserează Z în heap;

Pas 3. Singurul nod rămas în heap este rădăcina arborelui Huffman. Se generează codurile caracterelor, parcurgând arborele Huffman.

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

(5 p) **3. 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

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

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

(5 p) **4. Bitone**

O secvenţă de numere întregi se numeşte **bitonă** dacă este crescătoare la început, iar apoi descrescătoare. Mai precis, o secvenţă a_1, a_2, ..., a_n este bitonă dacă:

- este o secvenţă nedescrescătoare: $a_1 \le a_2 \le ... \le a_n$ **sau**
- este o secvenţă necrescătoare: $a_1 \ge a_2 \ge ... \ge a_n$ **sau**
- exista un indice i pentru care $a_1 \le a_2 \le ... \le a_i \ge a_{i+1} \ge ... \ge a_n$

Cerinţă

Dată o secvenţă de numere întregi a_1, a_2, ..., a_n şi nişte întrebări de forma (i, j) să se răspundă pentru fiecare întrebare dacă subsecvenţa a_i, a_{i+1}, ..., a_j este bitonă.

Date de intrare

Fişierul de intrare *bitone.in* conţine pe prima linie numărul de numere din secvenţă, n. Pe a doua linie conţine cele n numere ale secvenţei, separate de spaţii. Pe a treia linie se află numărul de întrebări q. Pe următoarele q linii se vor găsi cîte două numere i j separate prin spaţiu, reprezentînd întrebările la care se cere răspuns.

Date de ieşire

Fişierul de ieşire *bitone.out* va conţine o singură linie cu q caractere 0 sau 1, fără spaţii între ele, caractere ce reprezintă răspunsurile la întrebări. Pentru fiecare întrebare veţi răspunde 1dacă subsecvenţa este bitonă, sau 0 în caz contrar.

Restricţii

- $1 \le n \le 1.000.000$
- $-2.000.000.000 \le a_i \le 2.000.000.000$
- $1 \le q \le 1.000.000$
- $1 \le i \le j \le n$

Exemple

bitone.in	**bitone.out**	**Explicaţie**
10 10 19 19 18 18 21 21 11 11 13 6 9 10 6 10 4 8 8 10 1 7 3 3	101101	subsecvenţele (6, 10) şi (1, 7) nu sînt bitone. Toate celelalte sînt.

15 10 11 13 13 6 8 8 8 4 4 5 9 0 2 2 10 2 10 9 13 1 3 7 14 4 7 1 9 3 10 4 11 13 13 9 9	0110000011	subsecvenţele (9, 13) (1, 3) (13, 13) şi (9, 9) sînt bitone. Toate celelalte nu.

Cerc informatică Vianu

(5 p) **5. La coadă**

La BIG au băgat pui[1]. 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.

[1] Spre norocul vostru, nu este nevoie să ştiţi ce este un BIG sau cine „au băgat" pui.

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 14 (13-17 ianuarie 2020)

(5 ps) **6.** 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:

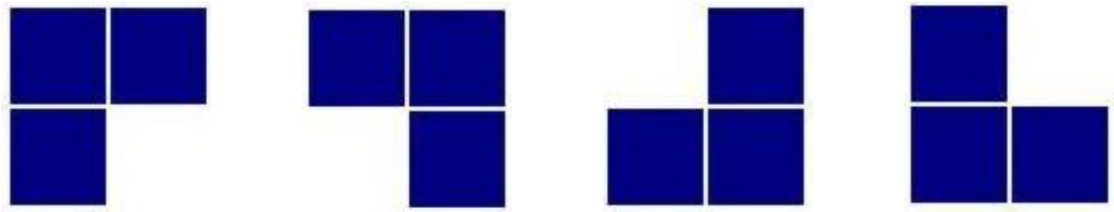
(a) o acoperire a unei table m x m se poate reprezenta printr-o matrice `Tabla[m][m]`, unde `Tabla[i][j]` indica numarul trominoului cu care este acoperit patratul `(i;j)`.

(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

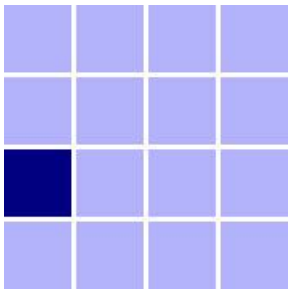


Figura 2. O tablă de şah defectă de dimensiuni $2^2 \times 2^2$