Examen - Algoritmi și Structuri de Date
Restanță - Seria 14

11 Iunie 2017

Instrucțiuni

În primul rând, vă rog să vă scrieți numele și grupa pe foaia de examen. Timpul de lucru este de 2 ore. Nu aveți voie să aveți asupra dumneavoastră decât instrumentul de scris și foile pe care vi le vor oferi instructorii.

Dacă vom găsi asupra dumneavoastră telefoane mobile, laptopuri, tablete, fițuici sau alte materiale ce conțin informații ajutătoare, veți fi scoși din sala de examinare și raportați. Dacă aveți întrebări, ridicați mâna și unul dintre instructori va veni la dumneavoastră în cel mai scurt timp. Scrieți răspunsurile în spațiul indicat. Ultima foaie capsată poate fi folosită ca ciornă. Dacă mai aveți nevoie de hârtie, adresați-vă unui instructor.

1. Exerciții de nota 5 - (5 puncte)

1.1 - 1 punct (0,5 puncte pe exercițiu)

Exprimați funcțiile următoare în notația Θ:

(a) $n^4 - \frac{n^4}{2} + 10000 \cdot n + 10$

Se observa ca n^4 este termenul dominant al functiei, ca urmare intuim ca vom avea complexitatea de $\Theta(n^4)$. Ca sa demonstram dam o valoare pentru c a.i. $\lim_{n\to\infty} \frac{cn^4}{f(x)} \in \mathbb{R}_+$, de exemplu 2.

(b) $\ln \ln n + \ln n$

Termenul dominant este $\ln n$, ca urmare complexitatea este $\Theta(\ln n)$. Se poate demonstra anolog subpunctului a), doar ca va trebui sa folosim L'Hospital de 2 ori pentru a ajunge la faptul ca limita da 1.

1.2 - 1 punct

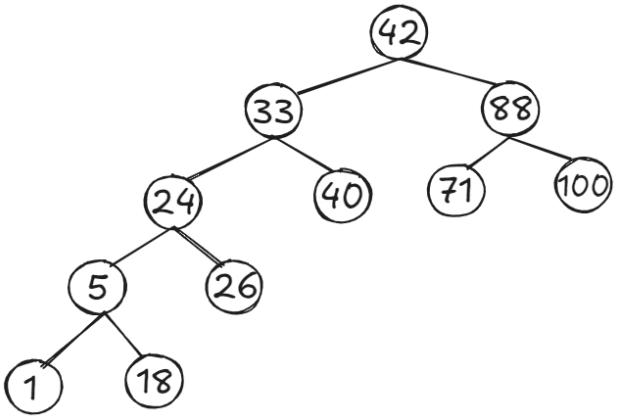
Care este numărul minim și numărul maxim de noduri într-un heap de înălțime h?

Un heap este un arbore binar complet $\rightarrow$ fiecare nivel contine numarul maxim de noduri, iar ultimul nivel poate fi incomplet, fiind completat de la stanga la dreapta. La nivelul h sunt 2^h noduri $\rightarrow$ pe primele $h-1$ nivele sunt $2^0 + 2^1 + \cdots + 2^{h-1} = 2^h - 1$ noduri. Pe ultimul nivel ar putea fi un singur nod, sau ar putea fi un nivel complet $\rightarrow$ numarul **minim** de noduri este $2^h - 1 + 1 = 2^h$, iar numarul **maxim** de noduri este $2^h - 1 + 2^h = 2^{h+1} - 1$.

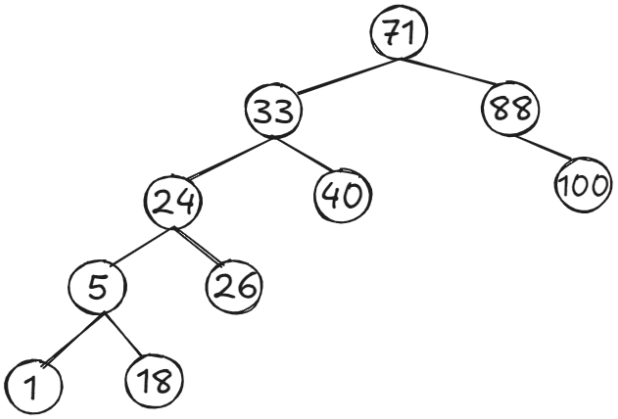
1.3 - 1 punct (0,5 puncte pe exercițiu)

Se dau următoarele chei citite pe rând de la consolă:
`42, 33, 88, 71, 24, 5, 18, 1, 40, 100, 26`

(a) Să se deseneze arborele binar de căutare rezultat din inserarea lor pe rând, în ordinea citirii (doar arborele final, fără pași intermediari).



(b) Să se extragă din arborele construit cheia cu valoarea 42, ilustrându-se arborele rezultat.



1.4 - 1 punct (0,5 puncte pe exercițiu)

Să se scrie heap-urile (ansamblele) rezultate prin inserția pe rând a următoarelor chei. Apoi, să se extragă rădăcina din heap-urile rezultate.

(a) min-heap: `43, 18, 81, 28, 22, 33, 30`

Figura 1: a) Insert 43 and 18

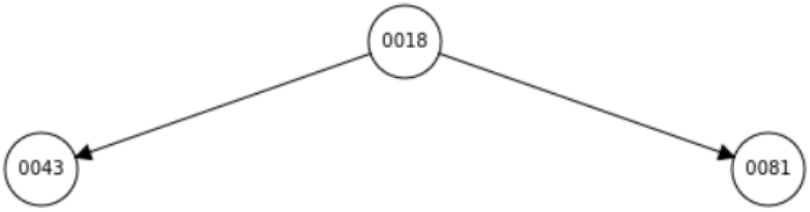


Figura 2: a) Insert 81

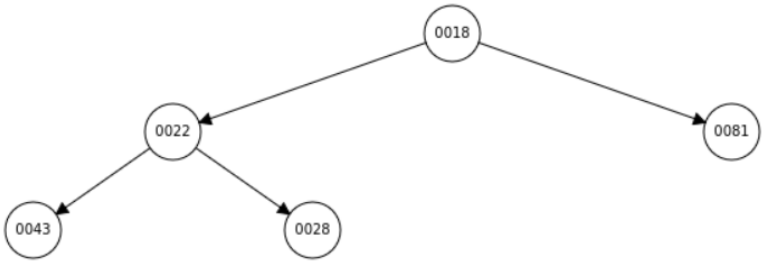


Figura 3: a) Insert 28 and 22

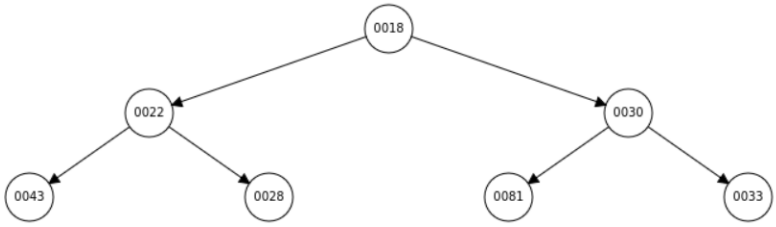


Figura 4: Insert 33 and 30

(b) max-heap: `17, 45, 2, 88, 12, 4, 28`

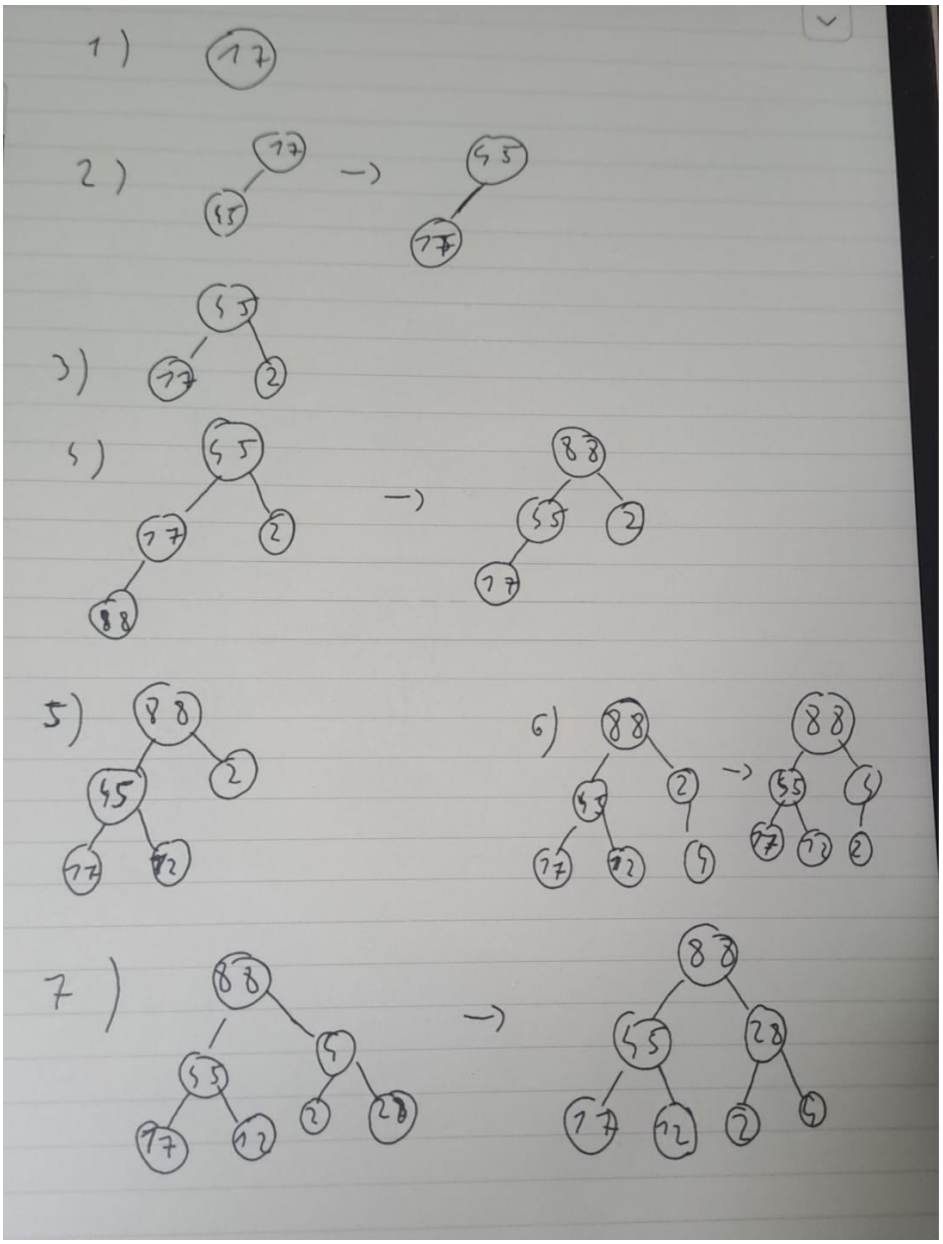


Figura 5: Max Heap steps

1.5 - 1 punct

Care este codul (arborele) Huffman optim pentru următoarele frecvențe:
`a:1, b:2, c:7, d:4, e:5, f:8, g:13, h:3`?

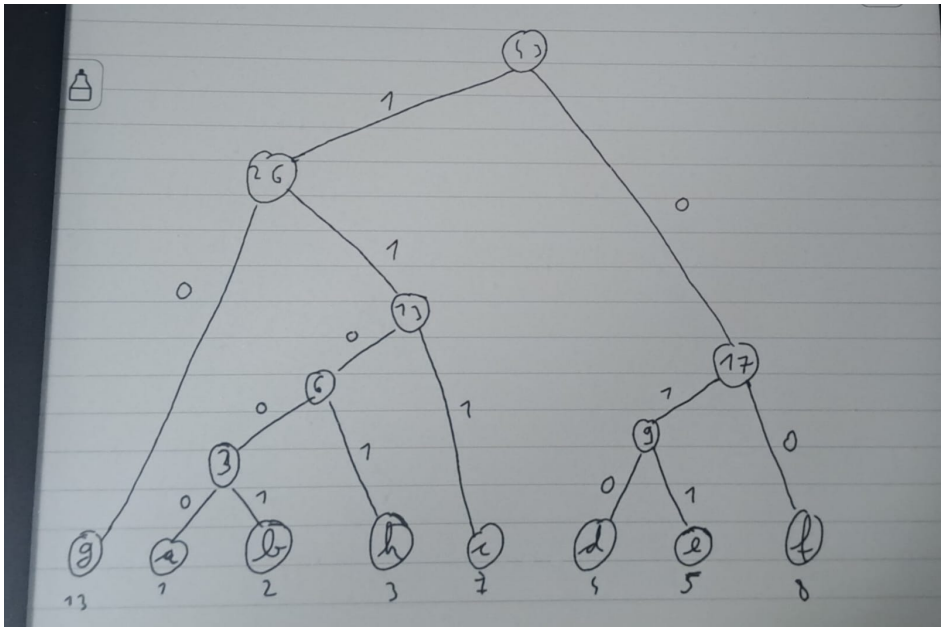


Figura 6: Huffman Tree

2. Exerciții cu demonstrații - (3 puncte)

2.1 - 1 punct

Rezolvați recurența:
$T(n) = T(n/100) + T(99n/100) + n$ și demonstrați că soluția găsită este corectă.

2.2 - 1 punct

Este adevărat că $f(n) + g(n) = \Theta(\max\{f(n), g(n)\})$? Demonstrați.

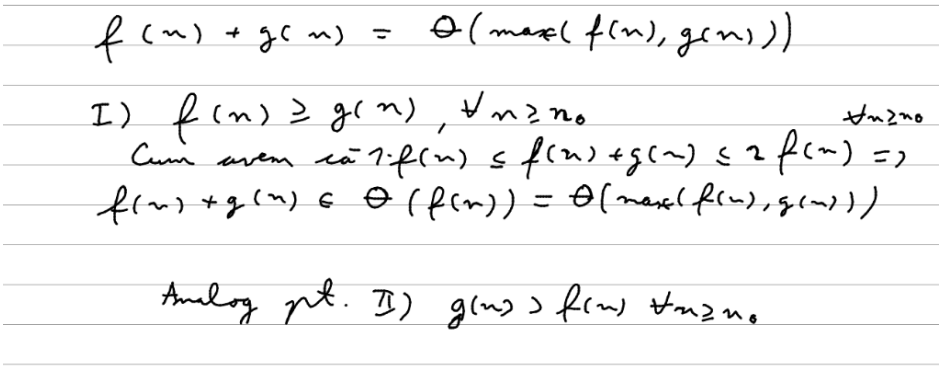


Figura 7: Exercitiul 2.2

2.3 - 1 punct

Demonstrați că orice algoritm care construiește un arbore binar de căutare cu n numere rulează în timp $\Omega(n)$.

Rezolvare: Stim ca algoritmul de construire a arborelui binar de cautare trebuie sa treaca prin fiecare element cel putin o data, iar apoi, in functie de tipul de input si ce arbore dorim sa construim, vom avea nevoie de $\Omega(nlogn)$ (daca sortam sirul dat si apoi folosim divide et impera pentru a forma un arbore balansat) sau $\Omega(n)$ (daca sirul este deja sortat). Deci, in cazul cel mai favorabil, tot vom fi in $\Omega(n)$.

3. Exerciții cu algoritmi - (3,5 puncte)

3.1 - 1 punct

Cum se poate implementa o coadă folosind un heap? Dar o stivă?

Rezolvare: Pentru a implementa o coada folosind un heap vom stoca in heap elementele in forma (idx, val) unde idx este indexul elementului cu valoarea val care se va incrementa de fiecare data cand adaugam un nou element in heap (nu este indexul in ordinea sortarii). Facem acest lucru pentru ca sortarea elementelor heap-ului sa se faca dupa acest index. Pentru a reprezenta o stiva, aplicam acelasi principiu doar ca vom folosi un max-heap.

3.2 - 1 punct

Se dau n numere între 0 și k. Descrieți un algoritm care preprocesează input-ul în timp $O(n+k)$ și răspunde în $O(1)$ la întrebări de forma:
„Se citesc două numere $0 \leq a, b \leq k$. Câte din cele n numere date ca input se găsesc în intervalul $[a \dots b]$?"

Rezolvare: Se va folosi un vector de frecvente f de lungime k, astfel incat f_i reprezinta numarul de elemente egale cu i din sirul dat. Aceasta preprocesare este in $O(n)$. Dupa aceea, vom folosi tehnica de sume partiale pentru a afla numarul de numere din sir mai mici sau egale decat i in f_i iterand prin sirul f cu i de la 1 la n si calculand noul f cu formula $f_i = f_i + f_{i-1}$. Aceasta preprocesare are complexitatea $O(k)$. Astfel, pentru a afla raspunsul la o interogare de tipul (a, b) cu formula $f_b - f_{a-1}$.

3.3 - 1,5 puncte

Cum putem sorta n numere în intervalul $0 \dots n^3 - 1$ în timp $O(n)$?

Solutie: Se va folosi radix sort: se va descrie pe scurt algoritmul (vezi Tutoriat 1) si se va demonstra ca $f(n) = 3n$ este in $O(n)$