**SD – Seminar 9 – SORTARE**

**15.12.2020**

1. Dati un exemplu de secventa de intrare pentru care algoritmul de sortare prin interclasare

are complexitatea timp O(n log n), iar algoritmul de sortare prin insertie are complexitatea O(n).

-o secventa deja sortata.

2. Fie a[] un vector cu n + m elemente. Primele n elemente sunt sortate, iar ultimele m

elemente nu sunt sortate.

(a) Care este complexitatea timp a algoritmului de sortare prin insertie utilizat pentru a

sorta a[], in cazul cel mai nefavorabil?

(b) Sugerati o metoda de sortare si precizati complexitatea timp a acesteia ın cazul cel

mai nefavorabil pentru urmatoarele situatii:

a) m = O(1) b) m = O(log n) c) m = O(n).

3. Propuneti un algoritm de complexitate O(n log k) pentru a sorta un vector de

dimensiune n ın care fiecare element se afla la o distanta de cel mult k de pozitia finala in vectorul sortat.

exemplu:

Output: tablou sortat: n = 8 , a = (2, 5, 8, 12, 21, 43, 67, 75)

k=2

tablou de intrare: a = ( 8, 2, 12, 5, 43, 21, 75, 67)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| intial | 8 | 2 | 12 | 5 | 43 | 21 | 75 | 67 |
| sortat | 2 | 5 | 8 | 12 | 21 | 43 | 67 | 75 |

I Solutie naiva: nu tinem cont de proprietatea tabloului de intrare 🡪 O(n log n)

II construim um minHeap cu primle k+1elemente din tabloul de intrare

copiem radacina minHeap-ului in tabloul sortat

eliminam din minHeap si inseram un nou element + refacem minHeap

===========================

Fie tabloul (5, 13, 2, 25, 7, 17, 20, 8, 4)

Aplicati algoritmul HeapSort pe vectorul de mai sus:

procedure scufundaAlTlea (n , a[], t ) // pentru un minHeap

begin

heap <- false

j <- t

fiu <- 2\*j+1

while (fiu < n and not heap) do {

if (fiu + 1 < n) then if a[fiu] > a[fiu +1] then fiu <- fiu+1

if a[j] > a[fiu] then {swap(a[j], a[fiu]); j <- fiu}

else heap <- true

}

end

procedure construiesteMinHeap(n , a[0..n-1])

begin

for j <- n/2 – 1 downto 0 do scufundaAlTlea(n, a, j)

end

procedure sortare( n , k, a[0..n-1] )

begin

for i <- 0 to k do c[i] <- a[i]

construiesteMinHeap(k+1, c)

for i <- 0 to n – k - 2 do {

a[i] <- c[0]

c[0] <- a[k+i+1]

scufundaAlTlea(k+1, c, 0)

}

// TEMA: de sortat elementele ramase in minHeap-ul de dimensiune k+1

end