SD – Seminar 4 / Analiza eficientei timp a algoritmilor

27.10.2020

1. Sa se arate ca

de aratat!

adevarat pt c =1 >0, n0 =1 🡪

de aratat!

adevarat pt c =1/2 >0, n0 =1 🡪

* 1. 🡪 TEMA

1. Care dintre urmatoarele afirmatii sunt adevarate? Demonstrati!

Fals Fals Adev.

T(n) = n(n+1)(2n+1)/6 = n^3 / 3 + n^2 / 2 + n/6

PP RA ca T(n) = O(n^2) 🡪 Exista c>0 , no >=1 a.i. orice n >= n0 : T(n) <= c n^2 🡪

! Dar limita este infinit. ?!

* T(n) ≠ O(n^2) 🡪 T(n) ≠

1. Se considera urmatorul algoritm, unde x este un tablou de numere intregi

procedure alg1(k, x[0..n])

begin

if k == 1 then for i <- 1 to n do print x[i]

else for i <- 1 to k do {

swap(x[i], x[k])

alg1(k-1, x)

swap(x[i], x[k])

}

end

Stabiliti si demonstrati clasa de complexitate a algoritmului de mai sus pentru apelul alg1(n, x).

1. Scrieti un algoritm pentru a rezolva problema turnurilor din Hanoi. Stabiliti si demonstrati clasa de complexitate a algoritmului propus.

Input: n >= 1 numarul de discuri, ta, tb, tc cele trei turnuri cu discurile pe ta de mutat pe tb.

Output: secventa de mutari (conform descrierii pb)

procedure hanoi(int n, char ta, char tb, char tc)

begin

if n==1 then print ta, ‘-‘, tb

else {

hanoi(n-1, ta, tc, tb)

print ta, ‘-‘, tb

hanoi(n-1, tc, tb, ta)

}

end

T(n) = numarul total de mutari a celot n discuri

T(1) = 1

T(n) = 1 + 2 T(n-1)

Iterarea inversa:

T(n) = 2 T(n-1) + 1 | 2^0

T(n-1) = 2 T(n-2) + 1 | 2^1

| 2^2

…

T(2) = 2 T(1) + 1 | 2^(n-2)

T(1) = 1 | 2^(n-1)

T(n) = =

1. Sa se rezolve urmatoarele recurente folosind arbori de recursie.
2. T(n) =

T(n) = n + 3 ( n/2 + 3 ( n/4 + 3 ( n/8 + 3 T(n/16 ) ) )

= n + n 3/2 + n (3/2)^2 + n (3/2)^3 + 3^4 T(n/(2^4))

PP ca n = 2^k k= log\_2 n