Projeto e Análise de Algoritmos

June 2, 2019

Prova 01 de 2016.2

2 Escreva um algoritmo recursivo (isto é, o algoritmo precisa chamar a si mesmo) que receba um número inteiro como entrada e imprima Θ ($n^{log_4 11}$ log n) asteriscos. Para justificar a complexidade você pode utilizar o teorema mestre.

Θ ($n^{log_4 11}$ log n)
T(n) = 11T(n/4) + $n^{log_4 11}$
a=11 b=4 c= $log_4 11$

Asterisco(n):

1 se $n > 1$ então :
2 para i de 1 até 11 faça:
3 Asterisco(n/4)
4 para j de 1 até $n^{log_4 11}$ faça:
5 print "*"

3 Sejam X[1..n] e Y[1...n] dois vetores ordenados. Esceva um algoritmo Θ (log n) para encontrar a mediana de todos os 2n elementos nos vetores X e Y. Prove esta complexidade.

Mediana(X,x, x', Y, y, y')

m_x = [(x+x') / 2]
m_y = [(y+y') / 2]
se X[m_x] = Y[m_y]
  retorne X[m_y]
se X[m_x] < Y[m_y]
  Mediana (X, m_x, x', Y, y, m_y)
  senão
   Mediana (X, x, m_x, Y, m_y, y')

Complexidade (Θ (log n))

T(n) = T(n/2) + 1
T(n/2) = T(n/ 2^2)
...
T(n/ 2^{k-1}) = T(n/ 2^k) + 1
T(n/ 2^k) = Θ (1)
(k = $log_2 n$)
T(n) = Θ (1) + k*1
T(n) = Θ (Log n)

4 Escreva um algoritmo recursivo com tempo Θ (n^2 log n) que recebe uma matriz quadrada M[n,n] com n linhas e n colunas, onde n é uma potência de 2, e ordena os elementos de M de modo que M[x_1][y_1] ¡= M[x_2][y_2]. Justifique.

Algoritmo(M, p, r):

para i de 1 até n faça:
  MergeSort(M[i], 1, r)
  Mergelinhas(M, n, p, r)

Mergelinhas(M, n, p, r)

```
se p < r então:
    q = (p + r) /2
    Mergelinhas(M, n, p, q)
    Mergelinhas(M, n, q+1, r)
Intercala(M, n, p, q, r)
```

Intercala(M, n, p, q, r):

```
Criar matriz L[1,...,q- p+2][n] e uma R[1,..., r- q+1][n]
para i de 1 até q - p+1 faça:
    para j de 1 até n faça:
      L[i][j] = M[p + i -1][j]
L[i + 1][j] = ∞
para i de 1 até r - q faça:
    para j de 1 até n faça:
      R[i][j] = M[q + i][j]
R[i+1][1] = ∞
i_1 , i_2, j_1, j_2 = 1
para k de p até r faça:
    para l de 1 até n faça:
      se L[ i_1][ j_1] <= R[ i_2][ j_2] então:
        M[k][l] = L[ i_1][ j_1]
         j_1 = j_1 + 1
        se j_1 > n então:
           j_1 = 1
         i_1 = i_1 + 1
      senão:
        M[k][l] = R[ i_2][ j_2]
         j_2 = j_2 + 1
        se j_2 > n então:
           j_2 = 1
           i_2 = i_2 +1
```

5 Explique os algoritmos Merge-Sort e Counting-Sort. Não precisa escrever código em sua explicação.

MergeSort
MergeSort é um algoritmo de divisão e conquista com complexidade Θ (n log n). Funciona recebendo um vetor como entrada e o ordena dividindo o problema em 2, chamando recursivamente o MergeSort para cada metade do vetor. Quando chegar ao caso base, executa o algoritmo intercala no primeiro e no segundo vetor, comparando o primeiro elemento de cada vetor e retorna o pedaço do vetor ordenado até concluir a recursão.

CountingSort
CountingSort é um algoritmo de contagem com complexidade Θ (n). Funciona criando um contador para cada dígito entre 1 e k. Percorre o vetor e contabiliza quantas vezes aparece. Depois adiciona ao valor contador o valor do contador anterior. Em seguida, percorre o vetor de n a 1, para manter a estabilidade e verifica qual a poisção correta de acordo com o valor do contador correspondente ao dígito.