LISTA 2 - ANALISE

Patrícia Moy.o

August 2023

Exercício 1c
$T(n) = 7T(\frac{n}{3}) + Cn^2$ para todo $n \geq 2$, $S = \{3^k : k \in N\}$

T(n) = 7T(n/3) + Cn^2

$= 7(7\mathrm{T}(\mathrm{n}/3^2) + Cn^2/3) + Cn^2 = 7^2T(n/3^2) + 7/3 \cdot Cn^2 + Cn^2$
$= 7^2(7T(n/3^3) + Cn^2/3^2) + 7/3 \cdot Cn^2 + Cn^2 = 7^3T(n/3^3) + (\frac{7}{3})^2Cn^2 + \frac{7}{3} \cdot Cn^2 + Cn^2$
...
$= 7^kT(n/3^k) + \sum_{i=0}^{k-1}(\frac{7}{3})^kCn^2$
= para k = $\log_3 n = 7^{log_3 n} \cdot T(1) + n^2 \cdot \sum_{i=0}^{log_3 n-1}(\frac{7}{3})^k$
Sabendo que a soma da PG é dada por a$_1\frac{r^{n-1}-1}{r-1}, temos:$
$7^{log_3 n} + 1 * \frac{\frac{7^{log_3 n}}{3}-1}{\frac{7}{3}-1} = \frac{3}{4} \cdot (\frac{7}{3})^{log_3 n} - 3$
$= 7^{log_3 n} + 3 * n^2(\frac{1}{4} * (\frac{7}{3})^{log_3 n} - 1)$
$= 7^{log_3 n} + 3 * n^2(\frac{1}{4} * \frac{7^{log_3 n}}{n} - 1)$
$= 7^{log_3 n} + 3 * n^2(\frac{1}{4} * \frac{(3^{log_3 7})^{log_3 n}}{n} - 1)$
$= 7^{log_3 n} + 3 * n^2(\frac{1}{4} * \frac{n^{log_3 7}}{n} - 1)$
$= 7^{log_3 n} + \frac{3}{4}n * n^{log_3 7} - 3n => \theta(n^{(calcular(log_3 7+1))})$
$= 7^{log_3 n} + \frac{3}{4}n * n^{log_3 7} - 3n => \theta(n^{1.7712437491614221})$

Fórmula fechada sem recursão: $7^{log_3 n} + \frac{3}{4}n * n^{log_3 7} - 3n^2$

Exercício 1d
$T(n) = 2T(\frac{n}{2}) + Cn^3$ para todo $n \geq 2$, $S = \{2^k : k \in N\}$
$T(n) = 2T(n/2) + Cn^3$ para todo $n \geq 2$, $S = \{2^k : k \in N\}$

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

$= 2(2\mathrm{T}(\mathrm{n}/2^2) + Cn^3/2) + Cn^3 = 2^2(T(n/2^2) + 2C \cdot n^3$
$= 2^2(2T(n/2^3) + Cn^3/2^2) + 2C \cdot n^3 = 2^3(T(n/2^3) + 3Cn^3$
$= 2^3(2T(n/2^4) + Cn^3/2^3) + 2C \cdot n^3 = 2^4T(n/2^4) + 4C \cdot n^3$
$= 2^kT(n/2^k) + kn) = nlgn + n = \theta(nlgn)$

Fórmula fechada sem recursão: $n \lg n + n$

Exercício 3
3. Considere a sequência de vetores $A_k[1 \ldots 2^k]$, $A_{k-1}[1 \ldots 2^{k-1}]$, $\ldots$, $A_1[1 \ldots 2^1]$, e $A_0[1 \ldots 2^0]$. Suponha que cada um dos vetores é crescente. Queremos reunir, por meio de sucessivas operações de intercalação (*merge*), o conteúdo dos vetores $A_0, \ldots, A_k$ em um único vetor crescente $B[1 \ldots n]$, onde $n = 2k + 1 - 1$. Escreva um algoritmo que faça isso em $O(n)$ unidades de tempo. Use como subrotina o *Intercala* visto em aula.

Algorithm 1 mesclaConta(vetor, esq, q, dir)

```
ladoEsq ← vetor[esqatéq]
ladoDir ← vetor[q + 1atédir]
i ← 0
j ← 0
inversoes ← 0
k ← esq
while i < tamanhodeladoEsqej < tamanhodeladoDir do
    if ladoEsq[i] ≤ ladoDir[j] then
        vetor[k] ← ladoEsq[i]
        i ← i + 1
    else
        vetor[k] ← ladoDir[j]
        j ← j + 1
        inversoes ← inversoes + (q − esq + 1) − i
    end if
    k ← k + 1
end while
while i < tamanhodeladoEsq do
    vetor[k] ← ladoEsq[i]
    i ← i + 1
    k ← k + 1
end while
while j < tamanhodeladoDir do
    vetor[k] ← ladoDir[j]
    j ← j + 1
    k ← k + 1
end while
retorne inversoes
```

Algorithm 2 contadorDeInversoes(vetor, esq, dir)

```
inversoes ← 0
if esq < dir then
    q ← (esq + dir)/2
    inversoes ← inversoes + contadorDeInversoes(vetor, esq, q)
    inversoes ← inversoes + contadorDeInversoes(vetor, q + 1, dir)
    inversoes ← inversoes + mesclaConta(vetor, esq, q, dir)
end if
retorne inversoes
```

```
vetor ← [1, 5, 4, 3, 2, 6]
inversoes ← contadorDeInversoes(vetor, 0, tamanho de vetor - 1)
escreva "O vetor possui", inversoes, "inversões."
```

.