UNIFEI - UNIVERSIDADE FEDERAL DE ITAJUBÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

SIN110 - ALGORITMOS E GRAFOS

RESOLUÇÃO DOS EXERCÍCIOS E02 DO DIA 28/08/2015

ITAJUBÁ

2015

Exercícios E02 – 28/08/15

Aluna: Karen Dantas **Número de matrícula:** 31243

1) Enunciado: Considere a seguinte função g, definida no conjunto dos números naturais, da seguinte forma:

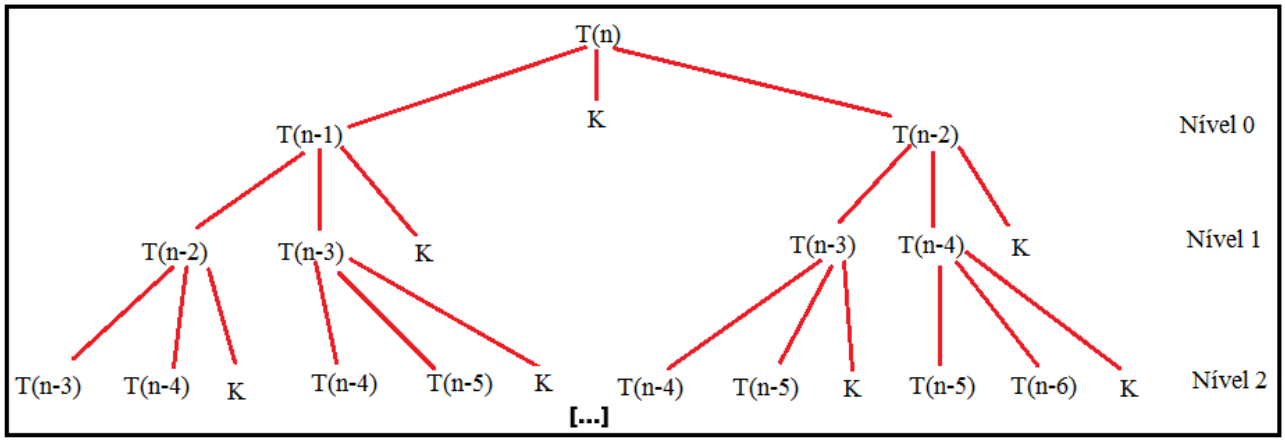
g(0) = 0, g(1) = 1 e,
g(n) = 5g(n-1) – 6g(n-2), para n ≥ 2.

a) Escreva um algoritmo que computa g;

```
Calcula_g (n)
1    se (n=1 ou n=0)
2          então devolve n
3          senão devolve 5*Calcula_g(n-1) – 6*Calcula_g(n-2)
```

b) Escreva e resolva uma recorrência para o tempo T(n) consumido pela função g(n), determine seu comportamento assintótico.

Expandindo g(n) e já utilizando a notação T(n), temos:



Com a análise da árvore, obtemos as seguintes soma de níveis:

Nível	Soma do nível
0	$1K = 2^0 . K$
1	$2K = 2^1 . K$
2	$4K = 2^2 . K$
...	
n-1	$2^{n-1} . K$
n	$2^n . K$

Sendo K uma constante, temos:
$T(n) = 2^0 . K + 2^1 . K + ... + 2^n . K$
$T(n) = [(2^0 . K + 2^n . K) . n]/2$
$T(n) = (K + 2^n . K) . n/2$

Comportamento Assintótico: $O(2^n)$

2) <u>Enunciado:</u> Considere o seguinte algoritmo recursivo para calcular o máximo de um vetor A[e..d]:

<u>Algoritmo</u>

```
Max(A, e, d)
1       se e = d
2           então devolve A[e]
3           senão x<- ⌊(e+d)/2⌋
4               a<- Max(A, e, x)
5               b<- Max(A, x+1, d)
6               se a > b
7                   então devolve a
8                   senão devolve b
```

Seja C(n) o número de vezes que a comparação da linha 6 é executada em uma chamada de **Max(A, e, d)**, onde **n=d – e + 1**. Escreva uma recorrência que define **C(n)** e, determine sua ordem de crescimento. Justifique sua resposta.

Para a contagem de cada linha, a qual é demonstrada na tabela abaixo, foi considerado que o último elemento do vetor era o maior, por isso, a linha 7 é executada nenhuma vez.

Algoritmo	Contagem
Max(A, e, d)	-----
1 se e = d	n+(n-1)
2 então devolve A[e]	n
3 senão x<- ⌊(e+d)/2⌋	(n-1)
4 a<- Max(A, e, x)	(n-1)
5 b<- Max(A, x+1, d)	(n-1)
6 se a > b	(n-1)
7 então devolve a	0
8 senão devolve b	(n-1)

Observa-se que a linha 6 é executada n-1 vezes, para vetores com n>1, pois, se for igual a 1, a execução do código termina na linha 2.

Para realmente ter certeza do resultado, fiz uma série de testes no computador. O resultado pode ser visto na tabela abaixo:

Tamanho do Vetor	Contagem da linha 6
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
15	14
20	19

Como pode-se observar na tabela acima, para variados tamanhos de vetores a linha 6 é

executada n-1 vezes. Logo, o C(n), em relação a linha 6, é:

$$C(1) = 0 \text{ , para } n=1$$
$$C(n) = n-1 \text{ , para } n > 1$$

Ordem de crescimento: O(n)

---

3) Enunciado: No método de classificação por inserção, uma otimização possível seria uma pesquisa mais rápida do local de inserção de uma chave através de busca binária. Escreva uma versão recursiva do algoritmo de ordenação por inserção implementando esta otimização considerando que o algoritmo deverá colocar em ordem crescente um vetor A[e..d]. Determine e compare a eficiência dos dois algoritmos para um melhor e para um pior caso.

- Inserção - iterativo

Pior caso:

Algoritmo	Contagem
Inserção (A, n)	----
1 para j <- 2 até n faça n	n
2 x <- A[j]	n-1
3 i <- j – 1	n-1
4 enquanto i>0 e A[i]>x faça	2+3+4+...+ n = (n+2).(n-1)/2
5 A[i+1] <- A[i]	1+2+3+...+ n-1 = (n).(n-1)/2
6 i <- i – 1	1+2+3+...+ n-1 = (n).(n-1)/2
7 A[i+1] <- x	n-1

$F(n) = n + 3(n-1) + 2[(n).(n-1)/2] + (n+2).(n-1)/2$
$F(n) = n + 3n -3 + n^2 – n + (n^2 – n + 2n -2) / 2$
$F(n) = 3n – 3 + n^2 + (n^2 + n -2) / 2$
$F(n) = (6n – 6 + 2n^2 + n^2 + n -2) / 2$
$F(n) = (3n^2 + 7n - 8)/2$
$F(n) = 3n^2/2 + 7n/2 – 4$

Comportamento Assintótico: $O(n^2)$

Melhor caso:

Algoritmo	Contagem
Inserção (A, n)	----
1 para j <- 2 até n faça n	n
2 x <- A[j]	n-1
3 i <- j – 1	n-1
4 enquanto i>0 e A[i]>x faça	1+1+1+...+ 1 = n-1
5 A[i+1] <- A[i]	0
6 i <- i - 1	0
7 A[i+1] <- x	n-1

$F(n) = n + 4(n-1)$
$F(n) = n + 4n - 4$
$F(n) = 5n – 4$
Comportamento Assintótico: $\Omega(n)$

- Inserção - recursivo

Pior caso:

Algoritmo	Contagem
Inserção_rec (A, e, d)	----
1 se e < d	1
2 então Inserção_Rec(A,e,d-1)	F(n-1)
3 x <- A[d]	1
4 i <- d-1	1
5 enquanto i ≥ e E A[i] > x faça	n
6 A[i+1] <- A[i]	n-1
7 i <- i-1	n-1
8 A[i+1] ← x	1

A partir do cálculo realizado na tabela acima, obtemos que:

$F(n) = 4.1 + 2.(n-1) + n + F(n-1)$

$F(n) = 4 + 2n – 2 + n + F(n-1)$

$F(n) = F(n-1) + 3n + 2$

Sabemos que em todas as operações temos ao menos uma execução, logo:

$F(1) = 1$, para n=1

$F(n) = F(n-1) + 3n + 2$, para n > 2

Por recorrência, temos:

$F(n) = F(n-1) + 3n + 2$

$F(n) = F(n-2) + [3(n-1) + 2] + [3n + 2]$

$F(n) = F(n-3) + [3(n-2) + 2] + [3(n-1) + 2] + [3n + 2]$

[...]

$F(n) = F(1) + [3(2) + 2] + [3(3) + 2] + ... + [3(n-1) + 2] + [3n + 2]$

$F(n) = 1 + 3[2 + 3+ ... + (n-1) + n] + 2(n-1)$

$F(n) = 1 + \{3[(n+2)(n-1)]\}/2 + 2n – 2$

$F(n) = 2n -1 + \{3[(n^2 – n + 2n - 2)]\}/2$

$F(n) = 2n -1 + \{3[(n^2 + n - 2)]\}/2$

$F(n) = 2n -1 + (3n^2 + 3n – 6)/2$

$F(n) = (4n -2 + 3n^2 + 3n – 6)/2$

$F(n) = (3n^2 + 7n – 8)/2$

$F(n) = 3n^2/2 + 7n/2 – 4$

Comportamento Assintótico: $O(n^2)$

Melhor caso:

Algoritmo	Contagem
Inserção_rec (A, e, d)	----
1 se e < d	1
2 então Inserção_Rec(A,e,d-1)	F(n-1)
3 x <- A[d]	1
4 i <- d-1	1
5 enquanto i ≥ e E A[i] > x faça	n
6 A[i+1] <- A[i]	0
7 i <- i-1	0
8 A[i+1] ← x	1

A partir do cálculo realizado na tabela acima, obtemos que:

$F(n) = 4.1 + n + F(n-1)$

$F(n) = F(n-1) + n + 4$

Sabemos que em todas as operações temos ao menos uma execução, logo:

$F(1) = 1$, para $n=1$

$F(n) = F(n-1) + n + 4$, para $n > 2$

Por recorrência, temos:

$F(n) = F(n-1) + n + 4$

$F(n) = F(n-2) + [(n-1) + 4] + [n + 4]$

$F(n) = F(n-3) + [(n-2) + 4] + [(n-1) + 4] + [n + 4]$

[...]

$F(n) = F(1) + [2 + 4] + [3 + 4] + ... + [(n-1) + 4] + [n + 4]$

$F(n) = 1 + [2 + 3+ ... + (n-1) + n] + 4(n-1)$

$F(n) = 1 + [(n+2)(n-1)]/2 + 4n – 4$

$F(n) = 4n - 3 + (n^2 – n + 2n - 2)/2$

$F(n) = 4n - 3 + (n^2 + n - 2)/2$

$F(n) = (8n - 6 + n^2 + n - 2)/2$

$F(n) = (n^2 + 9n – 8)/2$

$F(n) = n^2/2 + 9n/2 – 4$

Comportamento Assintótico: $\Omega(n^2)$

Comparação:

Através dos cálculos realizados acima pode-se concluir que, no pior caso, o algoritmo iterativo e o recursivo tiveram o mesmo desempenho que foi $O(n^2)$.

Já no melhor caso, o iterativo tem melhor desempenho, pois, tem complexidade $O(n)$ e o recursivo tem complexidade $O(n^2)$.