

Universidade Federal de Ouro Preto
PCC104 - Projeto e Análise de Algoritmos
Semanas 1 e 2

Prof. Rodrigo Silva

August 31, 2023

1 Leitura Recomendada

- Capítulo 2 - *Introduction to the Design and Analysis of Algorithms (3rd Edition)* - Anany Levitin

2 Questões

1. Ao que se referem os termos *Complexidade de Tempo* e *Complexidade de Espaço*?
2. Defina os termos, eficiência de melhor caso, caso médio e pior caso.
3. Qual, ou quais os problemas de utilizar unidades de tempo, por exemplo, segundos para analisar o tempo de execução de algoritmos? Qual a estratégia mais adequada para esta tarefa?
4. Para cada uma das seguintes funções, indique quanto o valor da função aumenta se o tamanho do argumento aumentar 4 vezes.
 - (a) $\log_2 n$
 - (b) \sqrt{n}
 - (c) n
 - (d) n^2
 - (e) n^3
 - (f) 2^n
5. Eliminação Gaussiana é um algoritmo clássico para resolver um sistema de n equações lineares com n variáveis. O método requer aproximadamente $\frac{1}{3}n^3$ multiplicações que é a operação básica do algoritmo.
 - (a) Quantas vezes mais devagar você espera que a resolução de um sistema de 1000 equações seja em relação a um sistema de 500.
 - (b) Você está considerando comprar 1000 vezes mais rápido do que o seu atual. Por qual fator o novo computador irá aumentar o tamanho dos sistemas resolvíveis no antigo dada a mesma quantidade de tempo?
6. Descreva as notações O , Ω e Θ .
7. Prove o seguinte teorema:

TEOREMA: Se $t_1(n) \in O(g_1(n))$ e $t_2(n) \in O(g_2(n))$ então $t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$

(OBS: Afirmações análogas são verdadeiras para Ω e Θ .)
8. Utilize limites para comparar as seguintes ordens de crescimento:

- (a) $\frac{1}{2}n(n-1)$ e n^2
 (b) $\log_2 n$ e \sqrt{n}
 (c) $n!$ e 2^n
9. Utilize as definições informais de O , Ω e Θ para determinar quais das afirmações abaixo são verdadeiras e quais são falsas.
- (a) $n(n+1)/2 \in O(n^3)$
 (b) $n(n+1)/2 \in \Theta(n^3)$
 (c) $n(n+1)/2 \in O(n^2)$
 (d) $n(n+1)/2 \in \Omega(n)$
10. Prove que todo polinômio de grau k , $p(n) = a_k n^k + a_{k-1} n^{k-1} + a_{k-2} n^{k-2} \dots + a_0$ com $a_i > 0$, pertence a $\Theta(n^k)$. (*Dica? Você pode provar esta afirmação utilizando limites.*)
11. Prove que funções exponenciais, a^n , têm diferentes ordens de crescimento para diferentes valores da base $a > 0$. (*Análise o limite, $\lim_{n \rightarrow \infty} \frac{a_1^n}{a_2^n}$.*)
12. Considere os três algoritmos abaixo:

ALGORITHM *Mystery(n)*
 //Input: A nonnegative integer n
 $S \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
 $S \leftarrow S + i * i$
return S

Figure 1: Algoritmo 1

ALGORITHM *Secret(A[0..n-1])*
 //Input: An array $A[0..n-1]$ of n real numbers
 $minval \leftarrow A[0]$; $maxval \leftarrow A[0]$
for $i \leftarrow 1$ **to** $n-1$ **do**
 if $A[i] < minval$
 $minval \leftarrow A[i]$
 if $A[i] > maxval$
 $maxval \leftarrow A[i]$
return $maxval - minval$

Figure 2: Algoritmo 2

ALGORITHM *Enigma(A[0..n-1, 0..n-1])*
 //Input: A matrix $A[0..n-1, 0..n-1]$ of real numbers
for $i \leftarrow 0$ **to** $n-2$ **do**
 for $j \leftarrow i+1$ **to** $n-1$ **do**
 if $A[i, j] \neq A[j, i]$
 return false
return true

Figure 3: Algoritmo 3

Para cada algoritmo responda:

- (a) O que este algoritmo computa?
 - (b) Qual a operação básica deste algoritmo?
 - (c) Quantas vezes esta operação básica é executada?
 - (d) Qual a classe deste algoritmo em relação à eficiência?
 - (e) Sugira alguma melhoria ou um novo algoritmo melhor e indique a classe desta sugestão. Se você não conseguir, tente provar que, de fato, a melhoria não pode ser feita.
13. Resolva as seguintes relações de recorrência:
- (a) $x(n) = x(n-1) + 5$ para $n > 1$, $x(1) = 0$
 - (b) $x(n) = 3x(n-1)$ para $n > 1$, $x(1) = 4$
 - (c) $x(n) = x(n-1) + n$ para $n > 0$, $x(0) = 0$
 - (d) $x(n) = x(n/2) + n$ para $n > 1$, $x(1) = 1$ (resolver para $n = 2^k$)
 - (e) $x(n) = x(n/3) + 1$ para $n > 1$, $x(1) = 1$ (resolver para $n = 3^k$)
14. Projete um algoritmo para computar 2^n para qualquer inteiro não negativo, n , baseado na fórmula $2^n = 2^{n-1} + 2^{n-1}$.
- (a) Apresente a relação de recorrência para o número de adições feitas pelo algoritmo e resolva a relação.
 - (b) Desenhe a árvore de chamadas recursivas para este algoritmo e conte o número de chamadas feita pelo algoritmo.
 - (c) Este é um bom algoritmo para resolver este problema.
15. Considere o seguinte algoritmo recursivo.
- ```

ALGORITHM Riddle($A[0..n-1]$)
 //Input: An array $A[0..n-1]$ of real numbers
 if $n = 1$ return $A[0]$
 else $temp \leftarrow Riddle(A[0..n-2])$
 if $temp \leq A[n-1]$ return $temp$
 else return $A[n-1]$

```
- (a) O que este algoritmo faz?
  - (b) Apresente a relação de recorrência para a operação básica do algoritmo, resolva a relação.
16. Considere o algoritmo abaixo que verifica se um grafo, definido por sua matriz de adjacência é completo.
- Qual é classe de eficiência deste algoritmo no pior caso?

**ALGORITHM** *GraphComplete*( $A[0..n-1, 0..n-1]$ )

//Input: Adjacency matrix  $A[0..n-1, 0..n-1]$  of an undirected graph  $G$   
 //Output: 1 (true) if  $G$  is complete and 0 (false) otherwise

**if**  $n = 1$  **return** 1 //one-vertex graph is complete by definition  
**else**  
   **if not** *GraphComplete*( $A[0..n-2, 0..n-2]$ ) **return** 0  
   **else for**  $j \leftarrow 0$  **to**  $n-2$  **do**  
     **if**  $A[n-1, j] = 0$  **return** 0  
   **return** 1