

Trabalho 1

1. Para cada par de funções $f(n)$ e $g(n)$ na tabela a seguir, indique se $f(n)$ pertence a $O(g(n))$, $\Omega(g(n))$ ou $\Theta(g(n))$. Considere que $k \geq 1$, $\epsilon > 0$ e $c > 1$. Justifique sua resposta.

	$f(n)$	$g(n)$	$f(n) = O(g(n))?$	$f(n) = \Omega(g(n))?$	$f(n) = \Theta(g(n))?$
a)	$\lg^k n$	n^ϵ		X	
b)	n^k	c^n	X		
c)	2^n	$2^{n/2}$		X	
d)	$n^{\lg c}$	$c^{\lg n}$			X

Solução:

Usando limite para analisar qual complexidade $f(n)$ corresponde em $g(n)$ aplicamos: $L = \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)}$

$$\text{a) } \lim_{n \rightarrow \infty} \frac{\lg^k n}{n^\epsilon} = \lim_{n \rightarrow \infty} n^{-\epsilon} \lg^k n = \lim_{n \rightarrow \infty} n^{-\epsilon} \cdot \lim_{n \rightarrow \infty} \lg^k n = \infty \Rightarrow f(n) = \Omega(g(n))$$

$$\text{b) } \lim_{n \rightarrow \infty} \frac{n^k}{c^n} = \lim_{n \rightarrow \infty} \frac{\log_n n^k}{\log_n c^n} = \lim_{n \rightarrow \infty} \frac{k}{\log_n c^n}$$

$$\text{Como } \lim_{n \rightarrow \infty} \log_n c^n = \infty$$

$$\lim_{n \rightarrow \infty} \frac{k}{\log_n c^n} = 0 \Rightarrow f(n) = O(g(n))$$

$$\text{c) } \lim_{n \rightarrow \infty} \frac{2^n}{2^{n/2}} = \lim_{n \rightarrow \infty} 2^n \cdot 2^{-\frac{n}{2}} = \lim_{n \rightarrow \infty} 2^{\frac{n}{2}} = \infty \Rightarrow f(n) = \Omega(g(n))$$

$$\text{d) } \lim_{n \rightarrow \infty} \frac{n^{\lg c}}{c^{\lg n}} = \lim_{n \rightarrow \infty} \frac{n^{\lg c}}{n^{\lg c}} = \lim_{n \rightarrow \infty} 1 = 1 \Rightarrow f(n) = \Theta(g(n))$$

2. Para cada item a seguir, assinale **Verdadeiro** ou **Falso**. **Justifique** sua resposta usando as definições de notação assintótica.

- (a) ☒ V ☐ F Se $f(n) = \log_{16} n$ então $f(n) = \Theta(\lg n)$?
 (b) ☐ V ☒ F $2^{n+a} = \Theta(2^{2n})$? Onde $a \in \mathbb{N}$ é uma constante.
 (c) ☒ V ☐ F $\frac{n^2}{4} - 3n - 16 = \Omega(n^2)$?
 (d) ☒ V ☐ F $7n^2 + 13n = O(n^2)$.

Solução:

$$\text{a) } \lim_{n \rightarrow \infty} \frac{\log_{16} n}{\lg n} = \frac{1}{4} \rightarrow f(n) = \Theta(g(n))$$

$$\text{b) } \lim_{n \rightarrow \infty} \frac{2^{n+a}}{2^{2n}} = 0 \Rightarrow f(n) = 2^{n+a} \in O(2^{2n})$$

$$\text{c) } \lim_{n \rightarrow \infty} \frac{\frac{n^2}{4} - 3n - 16}{n^2} = \frac{1}{4} \Rightarrow f(n) = \frac{n^2}{4} - 3n - 16 \in \Theta(n^2) = O(n^2) \wedge \Omega(n^2) \Rightarrow f(n) \in \Omega(n^2)$$

$$\text{d) } \lim_{n \rightarrow \infty} \frac{7n^2 + 13n}{n^2} = 7 \rightarrow f(n) = 7n^2 + 13n \in \Theta(n^2) = O(n^2) \wedge \Omega(n^2) \Rightarrow f(n) \in O(n^2)$$

3. Mostre usando as definições de notação assintótica:

- (a) $\frac{n}{2} \lg(\frac{n}{2}) = \Omega(n \lg n)$.
 (b) $n^2 + 1000n = O(n^2)$.
 (c) $2^{n+1} = \Theta(2^n)$.

Solução:

Escreva a solução aqui.

4. Expresse as seguintes funções em termos da notação Θ .

- (a) $2n + 3 \log^{100} n$.
- (b) $7n^3 + 1000n \log n + 3n$.
- (c) $3n^{1.5} + (\sqrt{n})^3 \log n$.
- (d) $2^n + 100^n + n!$.

Solução:

Escreva a solução aqui.

5. É comum usar a notação $f(n) \prec g(n)$ para denotar que $f(n) \in o(g(n))$. Use esta notação para expressar a hierarquia de classes de complexidade das seguintes funções: \sqrt{n} , 2^{n^2} , n , $\lg n$, 1 , $\lg \lg n$, $n!$, n^2 , $n^{3/4}$, 2^n , $n \lg n$.

Solução:

Escreva a solução aqui.

6. Sejam $f(n)$ e $g(n)$ funções positivas. Informe se a afirmação é verdadeira ou falsa e justifique.

- (a) $f(n) = O(g(n))$ implica $g(n) = O(f(n))$.
- (b) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$
- (c) $f(n) = O(g(n))$ implica $g(n) = \Omega(f(n))$
- (d) $f(n) = \Theta(f(n/2))$

Solução:

Escreva a solução aqui.

7. Mostre uma função $f(n)$ tal que $f(n) \notin \Omega(f(n+1))$.

Solução:

Escreva a solução aqui.

8. Mostre que $\sum_{i=1}^n \lg i = \Theta(n \lg n)$.

Solução:

Escreva a solução aqui.

9. Mostre que $n! = O(2^{n^2})$.

Solução:

Escreva a solução aqui.

10. Seja $p(n) = \sum_{i=0}^k a_i n^i$ (polinômio em n de grau k), onde k é um inteiro não-negativo, a_i é uma constante e $a_k > 0$, mostre que $p(n) = \Theta(n^k)$.

Solução:

Usando limite, devemos mostrar que: $L = \lim_{n \rightarrow \infty} \frac{\sum_{i=0}^k a_i n^i}{n^k}, 0 < L < \infty$

$$\begin{aligned}
 L &= \lim_{n \rightarrow \infty} \frac{\sum_{i=0}^k a_i n^i}{n^k} \\
 &= \lim_{n \rightarrow \infty} \frac{a_0 n^0 + a_1 n^1 + a_2 n^2 + \dots + a_k n^k}{n^k} \\
 &= \lim_{n \rightarrow \infty} \frac{a_0 n^0}{n^k} + \lim_{n \rightarrow \infty} \frac{a_1 n^1}{n^k} + \lim_{n \rightarrow \infty} \frac{a_2 n^2}{n^k} + \dots + \lim_{n \rightarrow \infty} \frac{a_k n^k}{n^k} \\
 &= 0 + 0 + 0 + a_k \\
 &= a_k.
 \end{aligned}$$

11. Papai Noel resolveu antecipar seu presente de Natal. Sabendo que você foi um(a) bom(a) menino(a), ele escreveu um algoritmo para que você analise e ganhe uns pontinhos na prova de PAA. Sua tarefa é simples. Dado um inteiro n como entrada, expresse por meio de notação assintótica a quantidade de “Ho!”s que será impressa pelo algoritmo (use a notação Θ).

FELIZ-NATAL(n)

```

1  $i \leftarrow 1$ 
2 while  $i \leq n$  do
3   | for  $j \leftarrow i$  to  $2i - 1$  do
4   |   | print “Ho!”;
5   |  $i \leftarrow 2i$ 
```

Solução:

Escreva a solução aqui.

12. Dado um inteiro n (assuma que $n = 2^k$, tal que k é um número inteiro positivo) e $expr$ (que corresponde a uma expressão a ser impressa), informe, por meio de notação assintótica, a quantidade de mensagens que o algoritmo a seguir irá imprimir. Dê sua resposta em função de n .

PROG($n, expr$)

```

1 while  $n \geq 1$  do
2   | for  $j \leftarrow 1$  to  $n$  do
3   |   | print  $expr$ 
4   |  $n \leftarrow n/2$ 
```

Solução:

A cada repetição do While a linha 3 executará $\frac{n}{2}$ a cada iteração. O While executa $\lg n$ vezes, então a linha 3

será executada $\sum_{k=1}^{\lg n} \frac{n}{2^k}$. Encontraremos a fórmula fechada para esse somatório:

$$\begin{aligned}
 \sum_{k=1}^{\lg n} \frac{n}{2^k} &= n \sum_{k=1}^{\lg n} \frac{1}{2^k} \\
 &= n \frac{\frac{1}{2} \left(\frac{1}{2}^{\lg n} - \frac{1}{2}^{-1} \right)}{\frac{1}{2} - 1} \\
 &= n \frac{\frac{1}{2} (n^{\lg \frac{1}{2}} - 2)}{-\frac{1}{2}} \\
 &= n \frac{\frac{1}{2} (n^{\lg 1 - \lg 2} - 2)}{-\frac{1}{2}} \\
 &= -n(n^{\lg 1 - \lg 2} - 2) \\
 &= 2n - 1
 \end{aligned}$$

\therefore O algoritmo irá imprimir a mensagem $2n - 1$ vezes uma entrada n e $T(n) = \Theta(n)$.

13. Seja *count* o número total de iterações feitas pelo algoritmo a seguir para uma entrada n . Informe, usando notação assintótica, o valor de *count* em função de n .

COUNT(n)

```

1  $count \leftarrow 0$ 
2 for  $i \leftarrow 1$  to  $n$  do
3   | for  $j \leftarrow 1$  to  $\lfloor n/i \rfloor$  do
4   |   |  $count \leftarrow count + 1$ 
5 return  $count$ 
```

Solução:

Escreva a solução aqui.

14. Seja *count* o número total de iterações feitas pelo algoritmo a seguir para uma entrada n (considere que $n = 2^{2^k}$, para algum inteiro positivo k). Informe, usando notação assintótica, o valor de *count* em função de n .

```

COUNT( $n$ )
1  $count \leftarrow 0$ 
2 for  $i \leftarrow 1$  to  $n$  do
3   |   for  $j \leftarrow 2; j \leq n; j \leftarrow j^2$  do
4   |   |    $count \leftarrow count + 1$ 
5 return  $count$ 

```

Solução:

Escreva a solução aqui.

15. Dado um inteiro $n \geq 0$ como entrada, muitos afirmam que o algoritmo a seguir é capaz de medir o desespero na prova de PAA. Outros afirmam que o algoritmo mede a alegria. Para o professor, não interessa o que o algoritmo mede. O objetivo desta questão é avaliar se o aluno é capaz de encontrar uma fórmula fechada que representa o valor final de x em função do valor de entrada n . Em outras palavras, uma função que representa quantas vezes a linha 5 será executada.

```

PROG( $n$ )
1  $x \leftarrow 0$ 
2 for  $i \leftarrow 1$  to  $n$  do
3   |   for  $j \leftarrow i + 1$  to  $n$  do
4   |   |   for  $k \leftarrow 1$  to  $j - i$  do
5   |   |   |    $x \leftarrow x + 1$ 

```

Solução:

$$\begin{aligned}
\sum_{k=1}^{n-1} \frac{(n-k+1)(n-k)}{2} &= \sum_{k=1}^{n-1} \frac{(n-k)^2 + (n-k)}{2} \\
&= \sum_{k=1}^{n-1} \frac{(n-k)^2}{2} + \sum_{k=1}^{n-1} \frac{n}{2} - \sum_{k=1}^{n-1} \frac{k}{2} \\
&= \frac{1}{2} \left[\sum_{k=1}^{n-1} k^2 \right] + \frac{1}{2} \left[n(n-1) - \frac{n(n-1)}{2} \right] \\
&= \frac{1}{2} \left[\sum_{k=1}^{n-1} k^2 \right] + \frac{n(n-1)}{4} \\
&= \frac{1}{2} \left[\sum_{k=1}^{n-1} k^2 \right] + \frac{n^2 - n}{4}
\end{aligned}$$

Como $\sum_{k=1}^{n-1} k^2$ é a soma das $n-1$ primeiras linha do triângulo de pascal, podemos concluir

$$\begin{aligned}
\sum_{k=1}^{n-1} \frac{(n-k+1)(n-k)}{2} &= \frac{1}{2} \left[\frac{n(n-1)(2n-1)}{6} \right] + \frac{n^2 + n}{4} \\
&= \frac{(n^2 - n)(2n-1)}{2} + \frac{n^2 - n}{4} \\
&= \frac{2n^3 - 3n^2 + n}{12} + \frac{3n^2 - 3n}{12} \\
&= \frac{2n^3 - 2n}{12} \\
&= \frac{12}{n^3 - n} \\
&= \frac{12}{6}
\end{aligned}$$

16. Resolva as seguintes recorrências (use o método da substituição):

- (a) $T(n) = 8T(n/2) + \Theta(n^2)$
- (b) $T(n) = 7T(n/2) + \Theta(n^2)$
- (c) $T(n) = T(n/4) + 1$
- (d) $T(n) = 2T(n/2) + n \lg n$

Solução:

a) $T(n) = 8T(n/2) + \Theta(n^2)$

Supondo que $T(n) \in O(n^3 - n^2)$ e substituindo em $T(n)$:

$$\begin{aligned} T(n) &= 8T(n/2) + \Theta(n^2) \\ &= 8c\left(\frac{n^3}{8} - \frac{n^2}{4}\right) + \Theta(n^2) \\ &= c.n^3 - 2.n^2 + d.n^2 \\ &= c.n^3 - n^2(2c - d) \end{aligned}$$

$$p/c \geq \frac{d}{2} \Rightarrow T(n) = O(n^3)$$

b) $T(n) = 7T(n/2) + \Theta(n^2)$

Supondo que $T(n) \in O(n^{\lg 7} - n^2)$ e substituindo em $T(n)$:

$$\begin{aligned} T(n) &= 7T(n/2) + \Theta(n^2) \\ &= 7c\left(\left(\frac{n}{2}\right)^{\lg 7} - \frac{n^2}{4}\right) + \Theta(n^2) \\ &= 7c\left(7^{\lg \frac{n}{2}} - \frac{n^2}{4}\right) + \Theta(n^2) \\ &= c7^{\lg n} - \frac{7n^2}{4} + \Theta(n^2) \\ &= c7^{\lg n} - n^2\left(\frac{7}{4} - d\right) \end{aligned}$$

$$p/c \geq \frac{7}{4} \Rightarrow T(n) = O(n^{\lg 7})$$

c) $T(n) = T(n/4) + 1$

Supondo que $T(n) \in O(\lg n)$ e substituindo em $T(n)$:

$$\begin{aligned} T(n) &= T(n) = T(n/4) + 1 \\ &= c\left(\frac{1}{4} \lg n + 1\right) \\ &= c\left(\frac{1}{4} \lg n + 1\right) \\ &= \frac{1}{4}c \lg n + 1 \end{aligned}$$

$$p/c > 4 \Rightarrow T(n) = O(n^{\lg n})$$

d) $T(n) = 2T(n/2) + n \lg n$

Supondo que $T(n) \in O(n \lg^2 n)$ e substituindo em $T(n)$:

$$\begin{aligned} T(n) &= T(n) = 2T(n/2) + n \lg n \\ &= 2c\left(\frac{n}{2} \lg^2 \frac{n}{2}\right) + n \lg n \\ &= cn \lg^2 n + n \lg n \end{aligned}$$

$$p/c > 0 \Rightarrow T(n) = O(n \lg^2 n)$$

17. Use árvore de recorrência para estimar um limite superior para as seguintes recorrências. Assuma que $T(n)$ é uma constante para $n \leq 2$. Depois comprove usando o método de substituição.

(a) $T(n/2) + T(n/4) + T(n/8) + n$

(b) $2T(n/4) + \sqrt{n}$

Solução:

Escreva a solução aqui.

18. Utilize o método de árvore de recursão para supor um limite assintótico superior para a recorrência $T(n) = 3T(n-1) + 1$. Depois verifique pelo método de substituição que este limite está correto.

Solução:

Escreva a solução aqui.

19. Utilize o método de árvore de recursão para supor um limite assintótico superior para a recorrência $T(n) = 2T(n/2) + n \lg n$. Depois verifique pelo método de substituição que este limite está correto.

Solução:

Escreva a solução aqui.

20. Utilize o método de árvore de recursão para supor um limite assintótico superior para a recorrência $T(n) = T(n/3) + T(2n/3) + \Theta(n)$. Depois verifique pelo método de substituição que este limite está correto.

Solução:

Escreva a solução aqui.

21. A recorrência $T(n) = 7T(n/2) + n^2$ descreve o tempo de execução de um algoritmo A . Um algoritmo alternativo A' tem um tempo de execução $T'(n) = aT'(n/4) + n^2$. Qual é o maior inteiro a que faz com que A' seja assintoticamente mais rápido que A ?

Solução:

Escreva a solução aqui.

22. Use o método mestre para resolver as seguintes recorrências:

- (a) $T(n) = 3T(n/2) + n \lg n$
- (b) $T(n) = 3T(n/2) + n^2$
- (c) $T(n) = 4T(n/2) + n^2$
- (d) $T(n) = 4T(n/2) + n^2 \sqrt{n}$
- (e) $T(n) = 5T(n/5) + n$
- (f) $T(n) = 6T(n/3) + n^2$
- (g) $T(n) = 9T(n/2) + n^3$

Solução:

a) $T(n) = 3T(n/2) + n \lg n$

$\log_b a = \lg 3 \simeq 1.58$ e $f(n) = n \lg n \Rightarrow$ Caso 1 do Método Mestre

$f(n) \in O(n^{\lg 3 - \epsilon})$, com $\epsilon = 0.08 \Rightarrow f(n) \in O(n^{\frac{3}{2}}) \Rightarrow T(n) = \Theta(n^{\lg 3})$

b) $T(n) = 3T(n/2) + n^2$

$\log_b a = \lg 3 \simeq 1.58$ e $f(n) = n^2 \Rightarrow$ Caso 3 do Método Mestre

$f(n) \in O(n^{\lg 3 + \epsilon})$, com $\epsilon = 0.42 \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^2)$

c) $T(n) = 4T(n/2) + n^2$

$\log_b a = 2$ e $f(n) = n^2 \Rightarrow$ Caso 2 do Método Mestre

$f(n) \in \Theta(n^2) \Rightarrow T(n) = \Theta(n^2 \lg n)$

d) $T(n) = 4T(n/2) + n^2 \sqrt{n}$

$\log_b a = 2$ e $f(n) = n^2 \cdot \sqrt{n} = n^{\frac{5}{2}} \Rightarrow$ Caso 3 do Método Mestre

$f(n) \in \Theta(n^{2+\epsilon})$, com $\epsilon = 0.5 \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^2 \cdot \sqrt{n})$

e) $T(n) = 5T(n/5) + n$

$\log_b a = 1$ e $f(n) = n \Rightarrow$ Caso 2 do Método Mestre

$f(n) \in \Theta(n) \Rightarrow T(n) = \Theta(n \lg n)$

f) $T(n) = 6T(n/3) + n^2$

$\log_b a = \log_3 6 \simeq 1.63$ e $f(n) = n^2 \Rightarrow$ Caso 3 do Método Mestre

$f(n) \in \Omega(n^{\log_3 6 + \epsilon})$, com $\epsilon = 0.33 \Rightarrow T(n) = \Theta(f(n)) = \Theta(n^2)$

g) $T(n) = 9T(n/2) + n^3$

$\log_b a = \lg 9 \simeq 3.16$ e $f(n) = n^3 \Rightarrow$ Caso 1 do Método Mestre

$f(n) \in O(n^{\lg 9 - \epsilon})$, com $\epsilon = 0.16 \Rightarrow f(n) \in O(n^3) \Rightarrow T(n) = \Theta(n^{\lg 9})$

23. Dadas as recorrências dos algoritmos A e B , determine a complexidade de cada um deles e compare-os (informe se A é assintoticamente mais rápido que B , B é assintoticamente mais rápido que A , ou ambos possuem a mesma complexidade assintótica).

- $T_A(n) = 27T_A(n/3) + n$
- $T_B(n) = 4T_B(n/2) + n^3$

Solução:

Escreva a solução aqui.

24. Os três algoritmos a seguir resolvem um problema de tamanho n por meio da técnica de divisão e conquista. Analise a complexidade de cada um deles e informe qual algoritmo é assintoticamente mais eficiente.

- Algoritmo A resolve problemas dividindo-os em cinco subproblemas de tamanho $n/2$, recursivamente resolve cada subproblema e combina suas soluções em tempo linear para obter uma solução do problema original.
- Algoritmo B resolve problemas dividindo-os em dois subproblemas de tamanho $n - 1$, recursivamente resolve cada um dos subproblemas e combina as soluções em tempo constante para obter a solução do problema original.
- Algoritmo C resolve problemas dividindo-os em nove subproblemas de tamanho $n/3$, recursivamente resolve cada subproblema e combina suas soluções em tempo $O(n^2)$ para obter uma solução do problema original.

Solução:

Escreva a solução aqui.

25. Os três algoritmos a seguir computam corretamente x^n para $x > 0$ e $n \geq 0$. Mostre que os três algoritmos estão corretos e analise a complexidade assintótica de cada um deles (em função de n) e informe qual deles é mais eficiente.

POWER1(x, n)

```
1 resp ← 1
2 i ← 0
3 while i < n do
4   | resp ← resp * x
5   | i ← i + 1
6 return resp
```

POWER2(x, n)

```
1 if n = 0 then
2   | return 1
3 else
4   | return POWER2(x, n - 1) * x
```

POWER3(x, n)

```
1 if n = 0 then
2   | return 1
3 else if (n mod 2) = 0 then
4   | aux ← POWER3(x, n/2)
5   | return aux * aux
6 else
7   | return POWER3(x, n - 1) * x
```

Solução:

Escreva a solução aqui.

26. Considere o algoritmo HEAPSORT descrito a seguir. Mostre que o algoritmo está correto usando o seguinte invariante de laço: “No começo de cada iteração do laço **for** das linhas 2–5, o subvetor $A[1 \dots i]$ contém os i menores elementos de $A[1 \dots n]$, e o subvetor $A[i+1 \dots n]$ contém os $n - i$ maiores elementos de $A[1 \dots n]$ em ordem.

HEAPSORT(A)

```
1 BUILD-MAX-HEAP(A)
2 for i ← A.length to 2 do
3   | SWAP(A[1], A[i])
4   | A.heap-size ← A.heap-size - 1
5   | MAX-HEAPIFY(A, 1)
```

Solução:

Escreva a solução aqui.

27. Analise a complexidade dos seguintes algoritmos:

$F(n)$

```
(a) 1 if n = 1 then
    2 | return 1
    3 else
    4 | return F(n - 1) + F(n - 1)
```

Solução:

$$\begin{aligned} 2T(n-1) + \Theta(1) &= \sum_{k=0}^{\lg n - 1} 2^k + \Theta(1) &= \frac{2^{n+1-1}-1}{2-1} + \Theta(1) \\ &= 2^n - 1 + \Theta(1) \\ &= O(2^n) \end{aligned}$$

BUSCA($A[], key, min, max$)

```
1 if max < min then
2   | return -1
3 mid ← min + ((max - min)/2)
(b) 4 if A[mid] > key then
5   | return BUSCA(A, key, min, mid - 1)
6 else if A[mid] < key then
7   | return BUSCA(A, key, mid + 1, max)
8 else
9   | return mid
```

Solução:

$$\begin{aligned} T\left(\frac{n}{2}\right) + \Theta(1) &= \sum_{k=0}^{\lg n - 1} 1 + \Theta(1) &= \lg n - 1 + \Theta(1) \\ &= O(\lg n) \end{aligned}$$

RECURSIVE(n)

```
1 if n > 1 then
(c) 2   | for i ← 1 to n3 do
3     | Faz algo (custo Θ(1))
4     | RECURSIVE(n/3)
```

Solução:

$$\begin{aligned} T\left(\frac{n}{3}\right) + \Theta(n^3) &= n^3 \sum_{k=0}^{n-1} \frac{1}{27} + \Theta(n^3) &= n^3 \frac{1}{1-\frac{1}{27}} + \Theta(n^3) \\ &= n^3 \frac{26}{27} + \Theta(n^3) \\ &= O(n^3) \end{aligned}$$

28. Assinale Verdadeiro (V) ou Falso (F). **Justifique**

- (a) ☒ V ☐ F O limite assintótico inferior para algoritmos de ordenação baseados em comparação é $\Omega(n \lg n)$. Um algoritmo de ordenação por comparação que faz $2T(n/2) + \Theta(1)$ comparações no pior caso com certeza não efetua corretamente a ordenação para algumas instâncias.
- (b) ☐ V ☒ F Suponha que iremos gerar n números aleatórios no intervalo $[0 \dots n^2]$. Considerando base decimal (isto é, um número x possui $\lfloor \log_{10} x \rfloor + 1$ dígitos na base 10), é correto afirmar que o algoritmo RADIX SORT faz a ordenação destes n números em tempo $O(n)$.
- (c) ☐ V ☒ F Visto que o limite assintótico inferior para algoritmos de ordenação baseados em comparação é $\Omega(n \lg n)$, não seria possível o desenvolvimento de um algoritmo de ordenação correto com complexidade de tempo $O(n\sqrt{n})$ no pior caso.
- (d) ☐ V ☒ F Suponha que iremos gerar n números aleatórios no intervalo $[0 \dots n^2]$. É correto afirmar que o algoritmo COUNTING SORT faz a ordenação destes n números em tempo $O(n)$.
- (e) ☒ V ☐ F Suponha que iremos gerar n números aleatórios no intervalo $[0 \dots n^2]$. É correto afirmar que o algoritmo MERGESORT faz a ordenação destes n números em tempo $O(n \lg n)$.
- (f) ☐ V ☒ F Não é possível construir um heap máximo com n elementos em tempo $O(n)$. Pois para inserir um elemento no heap temos custo $O(\lg n)$ e, como temos n elementos a serem inseridos, o custo total seria pelo menos $O(n \lg n)$.
- (g) ☒ V ☐ F Em um heap binário, metade dos elementos do vetor são folhas. Se aplicarmos o procedimento MAXHEAPFY para cada elemento da metade até o primeiro, então ao fim teremos um Heap Máximo.
- (h) ☒ V ☐ F É correto afirmar que: no melhor caso, o algoritmo INSERTION SORT é mais eficiente que os algoritmos MERGESORT e HEAPSORT.

Solução:

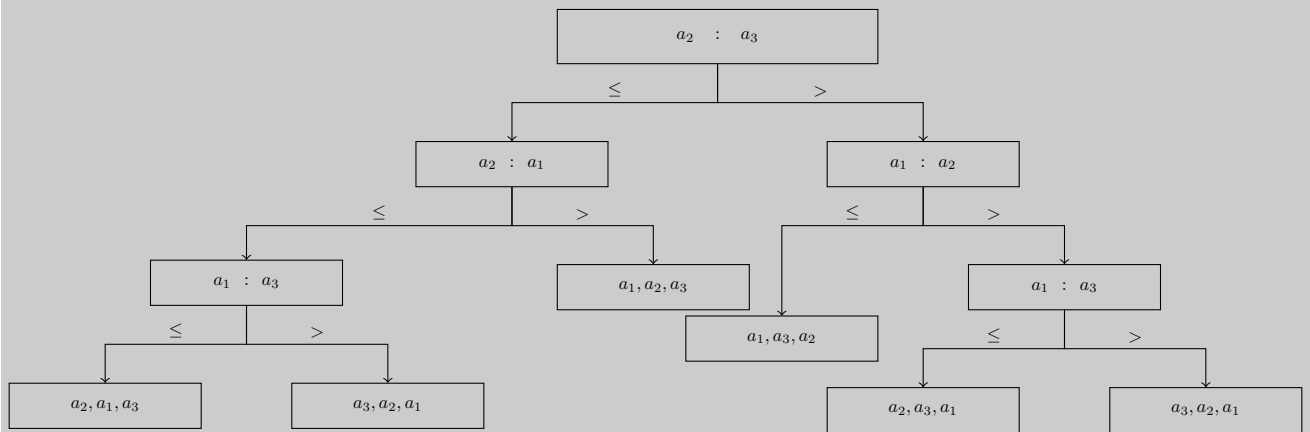
Escreva a solução aqui.

29. Use o modelo de árvore de decisão para representar as comparações efetuadas pelo algoritmo INSERTION-SORT para uma instância de entrada com quatro elementos.

Solução:

Escreva a solução aqui.

30. Use o modelo de árvore de decisão para representar as comparações efetuadas pelo algoritmo MERGESORT para uma instância de entrada com três elementos.

Solução:

31. Use o modelo de árvore de decisão para representar as comparações efetuadas pelo algoritmo QUICKSORT para uma instância de entrada com três elementos.

Solução:

Escreva a solução aqui.

32. Dado um vetor de inteiros distintos e ordenados em ordem crescente $A = \{a_1, \dots, a_n\}$:
- Descreva um algoritmo que determina se existe um índice i tal que $a_i = i$ em tempo $O(\lg n)$. Por exemplo, em $\{-10, -3, 3, 5, 7\}$, $a_3 = 3$. Em $\{2, 3, 4, 5, 6, 7\}$ não existe tal i . Argumente que seu algoritmo está correto.

Solução:

Escreva a solução aqui.

- Explique por que sua solução leva tempo $O(\lg n)$.

Solução:

Escreva a solução aqui.

33. Descreva um algoritmo baseado no paradigma de Divisão e Conquista que encontra o mínimo de um conjunto de n números. Assuma que os elementos estão em um vetor $A = [1 \dots n]$. Mostre que o algoritmo está correto e analise sua complexidade.

Solução:

```

MINA(A[ ], inicio, fim)
1 if inicio = fim then
2   return A[inicio]
3 else
4   meio ← inicio + ((fim - inicio)/2)
5   left = MINA(A[ ], inicio, meio)
6   right = MINA(A[ ], meio + 1, fim)
7   if left ≥ right then
8     return y
9   else
10    return x
  
```

$$\begin{aligned}
2T(n-1) + \Theta(1) &= \sum_{k=0}^{\lg n - 1} 2^k + \Theta(1) = \frac{2^{n+1}-1}{2-1} + \Theta(1) \\
&= 2^{n+1} - 1 + \Theta(1) \\
&= O(2^{n+1})
\end{aligned}$$

34. Descreva um algoritmo baseado no paradigma de Divisão e Conquista que encontra o segundo maior elemento de um conjunto de n números. Assuma que os elementos estão em um vetor $A = [1 \dots n]$. Mostre que o algoritmo está correto e analise sua complexidade.

Solução:

Escreva a solução aqui.

35. Descreva um algoritmo que faz uso do procedimento PARTITION para encontrar o k -ésimo menor elemento. Isto é, o algoritmo recebe como entrada um vetor $A[1 \dots n]$ e um valor $1 \leq k \leq n$ e devolve qual seria este k -ésimo menor elemento. Por exemplo, se $k = 1$ o algoritmo deveria devolver o mínimo do vetor; se $k = n$ o algoritmo deveria devolver o máximo; para um $k = 3$ devolveria o terceiro menor elemento. Analise seu algoritmo no pior e no melhor caso.

Solução:

Escreva a solução aqui.

36. Assuma que você possui k vetores ordenados, cada um com n elementos, e você precisa combiná-los em um único vetor ordenado com kn elementos.
- (a) Usando o procedimento MERGE, faça a intercalação do primeiro vetor com o segundo, então intercale o terceiro, depois o quarto e assim por diante. Qual a complexidade de tempo deste algoritmo, em função de k e n ?

Solução:

Escreva a solução aqui.

- (b) Apresente uma solução mais eficiente para este problema, por meio da técnica de Divisão e Conquista. Qual a complexidade de tempo de sua solução, em função de k e n ?

Solução:

Escreva a solução aqui.

37. Dado um vetor de números inteiros $A[1 \dots n]$, determine quais elementos do vetor são únicos. Apresente um algoritmo eficiente. Faça uma análise de complexidade.

Solução:

Escreva a solução aqui.

38. Descreva um algoritmo de tempo $\Theta(n \lg n)$ que, dado um conjunto S de n números inteiros e outro número x , determine se existe dois elementos em S cuja soma é exatamente x .

Solução:

Escreva a solução aqui.

39. Problema da moeda falsa. Dado um conjunto de n moedas, $n-1$ delas verdadeiras (com mesmo peso) e uma falsa (mais leve), descreva um algoritmo eficiente (com tempo $o(n)$) para encontrar a moeda falsa.

Solução:

Escreva a solução aqui.