Algoritmos e Estruturas de Dados II

Exercícios dos Quizzes até a 1ª Prova

Questões extraídas dos *quizzes* da disciplina elaborados pela Prof.ª Eveline Alonso
Resoluções elaboradas pelo aluno Luca Ferrari Azalim

---

1. O comando condicional if-else possibilita a escolha de um grupo de ações a serem executadas quando determinadas condições de entrada são ou não satisfeitas. O trecho de código abaixo contém uma estrutura condicional.

```
if (n < a + 3 || n > b + 4 || n > c + 1){
    l+= 5;
} else {
    l+= 2; k+=3; m+=7; x += 8;
}

if (n > b + 4){
    l+= 2; k+=3; m+=7; x += 8;
} else {
    l+= 5;
}
```

Considerando o código acima, marque a opção que apresenta o melhor e pior caso, respectivamente, para o número de adições.

a) 5 e 9
b) 4 e 9
c) 5 e 12
d) 6 e 12
e) 4 e 12

Resolução:

n < a + 3	n > b + 4	n > c + 1	Total de Adições
V	V	V	7
V	V	F	7
V	F	V	4
V	F	F	4

F	V	V	8
F	V	F	8
F	F	V	9
F	F	F	9

---

2. O trecho de código abaixo contém o comando *for*.

```
for (int i = 0; i <= n; i+=2){
   b *= 3;
}
```

Considerando o código acima, assinale a opção que apresenta o número de vezes que realizamos a operação de multiplicação.

a) $piso(\frac{n}{2})$

b) $teto(\frac{n}{2} - 1)$

c) $teto(\frac{n}{2})$

d) $piso(\frac{n}{2} + 1)$

e) $teto(\frac{n}{2} + 1)$

Resolução:

Valor de n	**Total de Multiplicações**
4	3
8	5
3	2
5	3

---

3. O trecho de código abaixo contém o comando *for*.

```
for (int i = n; i >= 0; i--) {
```

```
    for (int j = 2; j < (n - 7); j++) {
        b *= 5;
    }
}
```

Considerando o código acima, assinale a opção que apresenta o número de vezes que realizamos a operação de multiplicação.

a) $n(n-2)$
b) $n(n-9)$
c) $n(n-7)$
d) $(n+1)(n-9)$
e) $n(n-1)(n-7)$

---

4. O trecho de código abaixo realiza operações lógicas dentro de uma estrutura condicional.

```
if (n < a + 3 && n > b + 4 && n > c + 1){
    l+= 5;
} else {
    l+= 2; k+=3; m+=7; x += 8;
}

if (n <= b + 4){
    l+= 2; k+=3; m+=7; x += 8;
} else {
    l+= 5;
}
```

a) 6 e 12
b) 4 e 10
c) 6 e 11
d) 5 e 9
e) 4 e 9

Resolução:

n < a + 3	**n > b + 4**	**n > c + 1**	**n <= b + 4 ou NOT(n > b + 4)**	**Total de Adições**
V	V	V	F	6

V	V	F	F	9
V	F	V	V	11
V	F	F	V	11
F	V	V	F	7
F	V	F	F	7
F	F	V	V	10
F	F	F	V	10

---

5. Indique o número de multiplicações que o código abaixo realiza:

```
for (int i = n + 1; i > 0; i /= 2) {
    a *= 2;
}
```

a) $teto(log(n))$
b) $piso(log(n + 1)) + 1$
c) $piso(log(n)) + 1$
d) $piso(log(n))$

Resolução:

Valor de n	Valor de i	Total de Multiplicações
8	9, 4, 2, 1	4
16	17, 8, 4, 2, 1	5
7	8, 4, 2, 1	4
17	18, 9, 4, 2, 1	5

---

6. Indique o número de multiplicações que o código abaixo realiza:

```
for (int i = 1; i < n; i *= 2) {
    a *= 2;
```

```
}
```

a) $2 \times (piso(log(n)) + 1)$
b) $2 \times (piso(log(n)))$
c) $2 \times (teto(log(n)))$
d) $2 \times (piso(log(n + 1) + 1)$

Resolução:

Valor de n	Valor de i	Total de Multiplicações
8	1, 2, 4	6
16	1, 2, 4, 8	8
7	1, 2, 4	6
17	1, 2, 4, 8, 16	10

---

7. Indique o número de multiplicações que o código abaixo realiza:

```
for (int i = 1; i <= n; i *= 2) {
    a *= 2;
}
```

a) $2 \times (piso(log(n)) + 1)$
b) $2 \times piso(log(n))$
c) $2 \times teto(log(n))$
d) $2 \times (piso(log(n + 1)) + 1)$

Resolução:

Valor de n	Valor de i	Total de Multiplicações
8	1, 2, 4, 8	8
16	1, 2, 4, 8, 16	10
7	1, 2, 4	6
17	1, 2, 4, 8, 16	10

---

8. Indique o número de multiplicações que o código abaixo realiza:

```
for (int i = n; i > 1; i /= 2) {
    a *= 2;
}
```

a) $piso(log(n + 1)) + 1$
b) $piso(log(n)) + 1$
c) $teto(log(n))$
d) $piso(log(n))$

Resolução:

Valor de n	Valor de i	Total de Multiplicações
8	8, 4, 2	3
16	16, 8, 4, 2	4
7	7, 3	2
17	17, 8, 4, 2	4

---

9. Considere a soma: 4 + 25 + 64 + 121

Qual expressão resulta na soma acima?

a) $\sum(i^2 + 4), para\ 0 \leq i \leq 3$

b) $\sum (3i + 2)^2, para\ 0 \leq i \leq 3$

c) $\sum (2i + 4)^2, para\ 0 \leq i \leq 3$

d) $\sum(i^2 + i + 4), para\ 0 \leq i \leq 3$

e) $\sum(i^2 + 2i + 4), para\ 0 \leq i \leq 3$

---

10. Indique se a igualdade abaixo é verdadeira ou falsa:

$$\sum(3 + t) = 75 + \sum t, \; para \; 8 \leq t \leq 32$$

a) Falso
b) Verdadeiro

Resolução:

Passo 1: Aplicar a propriedade de associatividade.

$$\sum_{8}^{32} 3 + \sum_{8}^{32} t = 75 + \sum_{8}^{32} t$$

Passo 2: Passar o $\sum_{8}^{32} t$ para o outro lado com sinal inverso.

$$\sum_{8}^{32} 3 = 75 + \sum_{8}^{32} t - \sum_{8}^{32} t$$

Passo 3: Resolver a subtração do lado direito.

$$\sum_{8}^{32} 3 = 75$$

Passo 4: Resolver o somatório do lado esquerdo.

$$(32 - 8 + 1) \times 3 = 75$$
$$25 \times 3 = 75$$
$$75 = 75$$

---

11. Indique se a igualdade abaixo é verdadeira ou falsa:

$$\sum k^p = (\sum k)^p \; , \; para \; 0 \leq k \leq 12$$

a) Falso
b) Verdadeiro

Resolução:

Passo 1: Substituir p por um número qualquer

$$\sum_{0}^{12} k^2 = (\sum_{0}^{12} k)^2$$

Passo 2: Troque o valor máximo dos somatórios por um número menor.

$$\sum_{0}^{3} k^2 = (\sum_{0}^{3} k)^2$$

Passo 3: Resolva os somatórios e perceba que não se trata de uma igualdade.

$$0^2 + 1^2 + 2^2 + 3^2 = 0 + 1 + 2 + 3$$
$$0 + 1 + 4 + 9 = 0 + 1 + 2 + 3$$
$$14 = 6$$

---

12. Dado o somatório:

$$\sum(2i + x),\ para\ 1 \leq i \leq 3$$

Escolha uma opção:

a) $6 + 3x$
b) $6 + x$
c) $2 + 3x$
d) 12
e) $12 + 3x$

Resolução:

Passo 1: Aplicar a propriedade da associatividade.

$$\sum_{1}^{3}(2i) + \sum_{1}^{3}(x)$$

Passo 2: Aplicar a propriedade da distributividade.

$$2 \times \sum_{1}^{3}(i) + \sum_{1}^{3}(x)$$

Passo 3: Resolver o primeiro somatório utilizando a fórmula $\sum_{0}^{n} i = \frac{n(n+1)}{2}$, que é a fórmula que resulta na soma dos termos de uma progressão aritmética de razão 1.

$$\sum_{1}^{3}(i) = \frac{3(3+1)}{2} + i_0 = 6$$

Passo 4: Resolver o segundo somatório multiplicando x pelo número de iterações do somatório.

$$\sum_{1}^{3}(x) = 3x$$

Passo 5: Substituir os somatórios pelas resoluções e simplificar.

$$2 \times \sum_{1}^{3}(i) + \sum_{1}^{3}(x) = 2 \times 6 + 3x$$
$$2 \times \sum_{1}^{3}(i) + \sum_{1}^{3}(x) = 12 + 3x$$

---

13. Um ponto importante a ser considerado no projeto de um algoritmo é a eficiência da solução proposta. Essa eficiência tradicionalmente relaciona o tamanho da entrada com o tempo de execução ou espaço de memória necessários para a execução do algoritmo. Sabendo que a função de complexidade de tempo de um algoritmo é $f(n) = 3n^2 + 5n - 4$, analise as afirmações abaixo sobre a ordem de complexidade desse algoritmo:

I. $O(n^3)$
II. $O(n^2)$
III. $O(n^n)$

É correto o que se afirma em:

a) II e III, apenas.
b) I, apenas.
c) I, II e III.
d) I e II, apenas.

Resolução:

A ordem de complexidade da função $f(n) = 3n^2 + 5n - 4$ é $O(n^2)$ e, consequentemente, O de qualquer função maior que $f(n) = n^2$.

---

14. Leia as afirmativas a seguir, considerando que $f1(n)$ e $f2(n)$ são funções positivas.

1. Se $f2(n)$ é $O(f1(n))$, um algoritmo de função de complexidade de tempo $f1(n)$ possui ordem de complexidade $f2(n)$.

2. Se $f2(n)$ é $O(f1(n))$, então $f1(n)$ é um limite superior para $f2(n)$.

3. Se a função $f2(n) = 5 \times log(n) + 11$, então a função $f2(n)$ é $O(log(n))$.

4. Se $f2(n) = n^2$ e $f1(n) = (n + 2)^2$, temos que $f2(n)$ é $O(f1(n))$ e $f1(n)$ é $O(f2(n))$.

5. Se $f2(n) = 2 \times n + 1$ e $f1(n) = 2 \times n$, temos que $f2(n) = O(f1(n))$.

Assinale a alternativa que apresenta somente afirmativas corretas:

a) 1, 2, 4 e 5
b) 1, 3, 4 e 5
c) 2, 3, 4 e 5
d) 2, 3 e 5
e) 2, 3 e 4

---

15. Considere o código abaixo, implementado utilizando-se a linguagem de programação Java:

```
public static boolean saoDisjuntos(int[] A, int[] B) {
    for (int i = 0; i < A.length; i++)
        for (int j = 0; j < B.length; j++)
            if (A[i] == B[j])
                    return false;
    return true;
}
```

Considere ainda que cada um dos vetores “A” e “B” apresenta n elementos.

A ordem de complexidade desse algoritmo é:

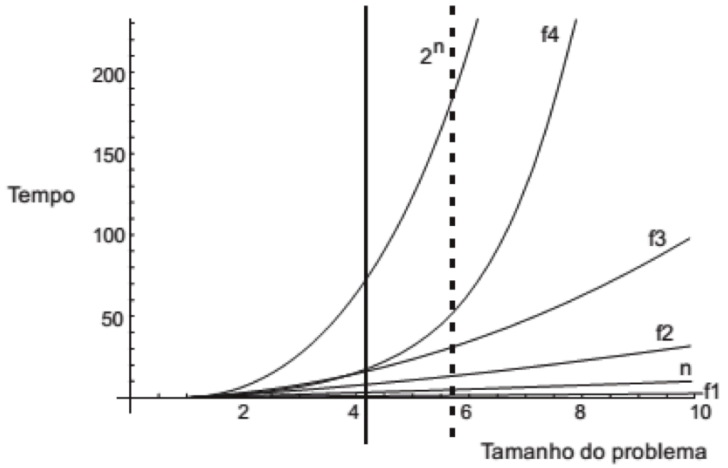
a) $O(log(n))$
b) $O(n)$
c) $O(n \times log(n))$
d) $O(n^2)$
e) $O(2 \times n)$

16. Considere dois vetores ordenados. Assuma ainda que cada um desses vetores apresente n elementos. Precisa-se formar então um terceiro vetor, correspondente à junção desses dois vetores ordenados mencionados anteriormente. Dessa forma, esse terceiro vetor apresentará $2n$ elementos, que também serão armazenados de forma ordenada. Pode-se afirmar que a ordem de complexidade desse processo será:

a) $\Theta(log(n))$
b) $\Theta(n)$
c) $\Theta(n^2)$
d) $\Theta(n \times log(n))$
e) $\Theta(1)$

---

17. Abaixo temos um gráfico que evidencia a relação de domínio assintótico entre diversas funções de complexidade de algoritmos.



Analisando o gráfico apresentado, podemos afirmar que:

a) O limite superior do comportamento do custo de uma função de complexidade $f(n)$ é n^2 quando n aproxima-se de $2n$.
b) A notação O permite comparar os limites superiores dos comportamentos de algoritmos. Assim,

podemos afirmar que um programa $O(f4)$ é sempre melhor que um algoritmo $O(f3)$.

c) A ordem de complexidade de f3 corresponde a $O(log(n))$.

d) f3 e f4 apresentam desempenho melhor que 2n.

e) A complexidade de f1 corresponde a $\Theta(n^2)$.