Resolução da MINI-PROVA 2025.1
CIN0009 - Algoritmos e Estruturas de Dados
Questão 1:
a) Array inicial: [5, 4, 1, 0, 6, 2, 3, 7]

Início da Partição #1 | Índices: 0 até 7 | Pivô = 5 (v[0])
Iteração 1
i anda para Direita: v[1] = 4 < pivô = 5
i anda para Direita: v[2] = 1 < pivô = 5
i anda para Direita: v[3] = 0 < pivô = 5
i parou em v[4] = 6 (>= pivô = 5)
j anda para Esquerda: v[7] = 7 > pivô = 5
j parou em v[6] = 3 (<= pivô = 5)
Como i = 4 < j = 6
Trocamos v[4] = 6 com v[6] = 3
Após troca: [5, 4, 1, 0, 3, 2, 6, 7]
Iteração 1
i anda para Direita: v[5] = 2 < pivô = 5
i parou em v[6] = 6 (>= pivô = 5)
j parou em v[5] = 2 (<= pivô = 5)
i (6) >= j (5) → Saímos do loop da partição #1
Trocando pivô v[0] = 5 com v[5] = 2
Finalizando partição #1 → Pivô vai para a posição 5
Estado atual do vetor: [2, 4, 1, 0, 3, 5, 6, 7]

Início da Partição #2 | Índices: 0 até 4 | Pivô = 2 (v[0])
Iteração 1
i parou em v[1] = 4 (>= pivô = 2)
j anda para Esquerda: v[4] = 3 > pivô = 2
j parou em v[3] = 0 (<= pivô = 2)
Como i = 1 < j = 3
Trocamos v[1] = 4 com v[3] = 0
Após troca: [2, 0, 1, 4, 3, 5, 6, 7]
Iteração 1

i anda para Direita: v[2] = 1 < pivô = 2
i parou em v[3] = 4 (>= pivô = 2)
j parou em v[2] = 1 (<= pivô = 2)
i (3) >= j (2) → Saímos do loop da partição #2
Trocando pivô v[0] = 2 com v[2] = 1
Finalizando partição #2 → Pivô vai para a posição 2
Estado atual do vetor: [1, 0, 2, 4, 3, 5, 6, 7]

Início da Partição #3 | Índices: 0 até 1 | Pivô = 1 (v[0])
Iteração 1
i anda para Direita: v[1] = 0 < pivô = 1
j parou em v[1] = 0 (<= pivô = 1)
i (2) >= j (1) → Saímos do loop da partição #3
Trocando pivô v[0] = 1 com v[1] = 0
Finalizando partição #3 → Pivô vai para a posição 1
Estado atual do vetor: [0, 1, 2, 4, 3, 5, 6, 7]

Início da Partição #4 | Índices: 3 até 4 | Pivô = 4 (v[3])
Iteração 1
i anda para Direita: v[4] = 3 < pivô = 4
j parou em v[4] = 3 (<= pivô = 4)
i (5) >= j (4) → Saímos do loop da partição #4
Trocando pivô v[3] = 4 com v[4] = 3
Finalizando partição #4 → Pivô vai para a posição 4
Estado atual do vetor: [0, 1, 2, 3, 4, 5, 6, 7]

Início da Partição #5 | Índices: 6 até 7 | Pivô = 6 (v[6])
Iteração 1
i parou em v[7] = 7 (>= pivô = 6)
j anda para Esquerda: v[7] = 7 > pivô = 6
j parou em v[6] = 6 (<= pivô = 6)
i (7) >= j (6) → Saímos do loop da partição #5
Trocando pivô v[6] = 6 com v[6] = 6
Finalizando partição #5 → Pivô vai para a posição 6

Estado atual do vetor: [0, 1, 2, 3, 4, 5, 6, 7]

Array final ordenado: [0, 1, 2, 3, 4, 5, 6, 7]

b)

Array inicial: [5, 4, 1, 0, 6, 2, 3, 7]
Fase 1: Divisão Recursiva
O array é dividido ao meio até que restem apenas arrays de um elemento.
Divisão #1: [5, 4, 1, 0, 6, 2, 3, 7]
Metade Esquerda: [5, 4, 1, 0]
Metade Direita: [6, 2, 3, 7]
Divisão #2: [5, 4, 1, 0]
Metade Esquerda: [5, 4]
Metade Direita: [1, 0]
Divisão #3: [6, 2, 3, 7]
Metade Esquerda: [6, 2]
Metade Direita: [3, 7]
Divisão #4: [5, 4] → Resulta em [5] e [4]
Divisão #5: [1, 0] → Resulta em [1] e [0]
Divisão #6: [6, 2] → Resulta em [6] e [2]
Divisão #7: [3, 7] → Resulta em [3] e [7]
Ao final da divisão, temos os sub-arrays: [5] [4] [1] [0] [6] [2] [3] [7]
Fase 2: Combinação Ordenada (Merge)
Agora, os sub-arrays são combinados de volta, de forma ordenada.
Início do Merge #1 | Combinando [5] e [4]
Comparamos 5 e 4. O 4 é menor.
Pegamos o 5 restante.
Resultado do Merge #1: [4, 5]
Início do Merge #2 | Combinando [1] e [0]
Comparamos 1 e 0. O 0 é menor.
Pegamos o 1 restante.
Resultado do Merge #2: [0, 1]
Início do Merge #3 | Combinando [6] e [2]
Comparamos 6 e 2. O 2 é menor.
Pegamos o 6 restante.
Resultado do Merge #3: [2, 6]
Início do Merge #4 | Combinando [3] e [7]
Comparamos 3 e 7. O 3 é menor.
Pegamos o 7 restante.
Resultado do Merge #4: [3, 7]
Estado atual dos sub-arrays: [4, 5], [0, 1], [2, 6], [3, 7]

Início do Merge #5 | Combinando [4, 5] e [0, 1]
Array Esquerda: [4, 5] | Array Direita: [0, 1]
Comparamos 4 e 0. O 0 é menor. Array auxiliar: [0]

Comparamos 4 e 1. O 1 é menor. Array auxiliar: [0, 1]
Array da Direita está vazio. Copiamos o resto da Esquerda (4, 5).
Resultado do Merge #5: [0, 1, 4, 5]
Início do Merge #6 | Combinando [2, 6] e [3, 7]
Array Esquerda: [2, 6] | Array Direita: [3, 7]
Comparamos 2 e 3. O 2 é menor. Array auxiliar: [2]
Comparamos 6 e 3. O 3 é menor. Array auxiliar: [2, 3]
Comparamos 6 e 7. O 6 é menor. Array auxiliar: [2, 3, 6]
Array da Esquerda está vazio. Copiamos o resto da Direita (7).
Resultado do Merge #6: [2, 3, 6, 7]
Estado antes do merge final: [0, 1, 4, 5] e [2, 3, 6, 7]
Início do Merge #7 (Final) | Combinando [0, 1, 4, 5] e [2, 3, 6, 7]
Array Esquerda: [0, 1, 4, 5] | Array Direita: [2, 3, 6, 7]
Comparamos 0 e 2. O 0 é menor. Auxiliar: [0]
Comparamos 1 e 2. O 1 é menor. Auxiliar: [0, 1]
Comparamos 4 e 2. O 2 é menor. Auxiliar: [0, 1, 2]
Comparamos 4 e 3. O 3 é menor. Auxiliar: [0, 1, 2, 3]
Comparamos 4 e 6. O 4 é menor. Auxiliar: [0, 1, 2, 3, 4]
Comparamos 5 e 6. O 5 é menor. Auxiliar: [0, 1, 2, 3, 4, 5]
Array da Esquerda está vazio. Copiamos o resto da Direita (6, 7).
Resultado do Merge #7: [0, 1, 2, 3, 4, 5, 6, 7]
Array final ordenado: [0, 1, 2, 3, 4, 5, 6, 7]

c)

Vetor ordenado V: [0, 1, 2, 3, 4, 5, 6, 7]
Valor a ser buscado (x): 4

1ª Comparação:
O espaço de busca inicial é o vetor inteiro (índices 0 a 7).
Calculamos o índice do meio: (0 + 7) / 2 = 3 (Pegamos o Piso).
O elemento comparado é o V[3], que vale **3**.
Comparamos x = 4 com o valor 3.
Como 4 > 3, sabemos que o valor, se existir, está na metade direita do vetor.
Eliminamos a parte esquerda.
nosso vetor estaria: [4, 5, 6, 7]

2ª Comparação
O novo espaço de busca vai dos índices 4 a 7.
Calculamos o novo índice do meio: (4 + 7) / 2 = 5 (Pegamos o Piso).
O elemento comparado é o V[5], que vale **5**.
Comparamos x = 4 com o valor 5.
Como 4 < 5, sabemos que o valor, se existir, está na metade esquerda deste novo espaço.
nosso vetor estaria: [4]

3ª Comparação O novo espaço de busca vai dos índices 4 a 4.
Calculamos o novo índice do meio: (4 + 4) / 2 = 4.
O elemento comparado é o V[4], que vale **4**.
Comparamos x = 4 com o valor 4.
Como 4 == 4, o elemento foi encontrado e a busca termina.

R: Os elementos de V comparados com x = 4 foram 3, 5 e 4.

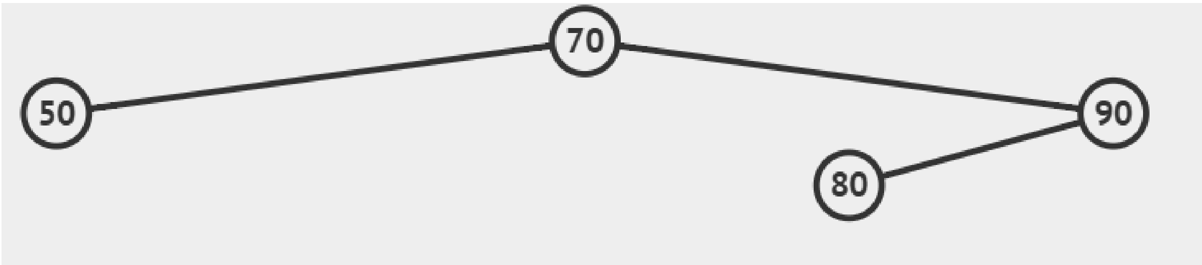
Questão 2:

Respostas:

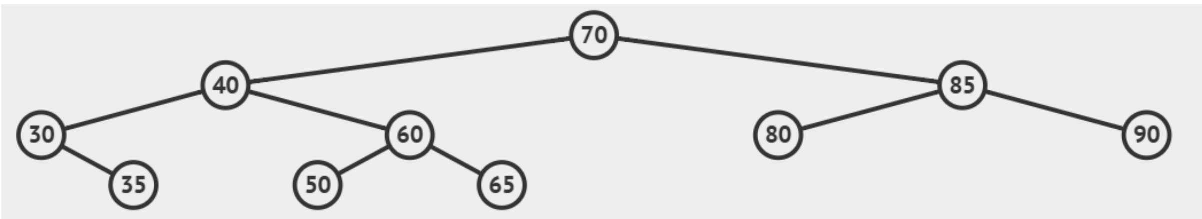
1. [F]Tendo $f(x) = x^4 - 4x + 5$ e $g(x) = 4x + 100$, temos $x^2 = O(f(x))$ e $x^3 = \Omega\ (g(x))$.
2. [V]Tendo $f(x) = x + 156$ e $g(x) = x^5 + 3$, temos $x/100 = O(f(x))$ e $2\text{^}x = \Omega\ (g(x))$.
3. [V]Tendo $f(x) = x^8 + 8$ e $g(x) = 3x^8$, temos $x^8 = O(f(x))$ e $x^8 = \Omega\ (g(x))$.
4. [F]Tendo $f(x) = x^{100} - 2x - 3$ e $g(x) = x^{21} + 300$, temos $x^{99} = O(f(x))$ e $x^{16} + 750= \Omega\ (g(x))$.
5. [F]Tendo $f(x) = x^{37} + 28x - 20$ e $g(x) = x^{25} - 3x$, temos $x = O(f(x))$ e $x = \Omega\ (g(x))$.

Questão 3:

Após a inserção do 80:



Após a inserção do 35:



Questão 4:

(a)

Pilha:

10
8
6
4
2

Fila:

1
3
5
7
9

(b)

Lista:

1	10	3	8	5	6	7	4	9	2

Vetor:

2	9	4	7	6	5	8	3	10	1

(c)

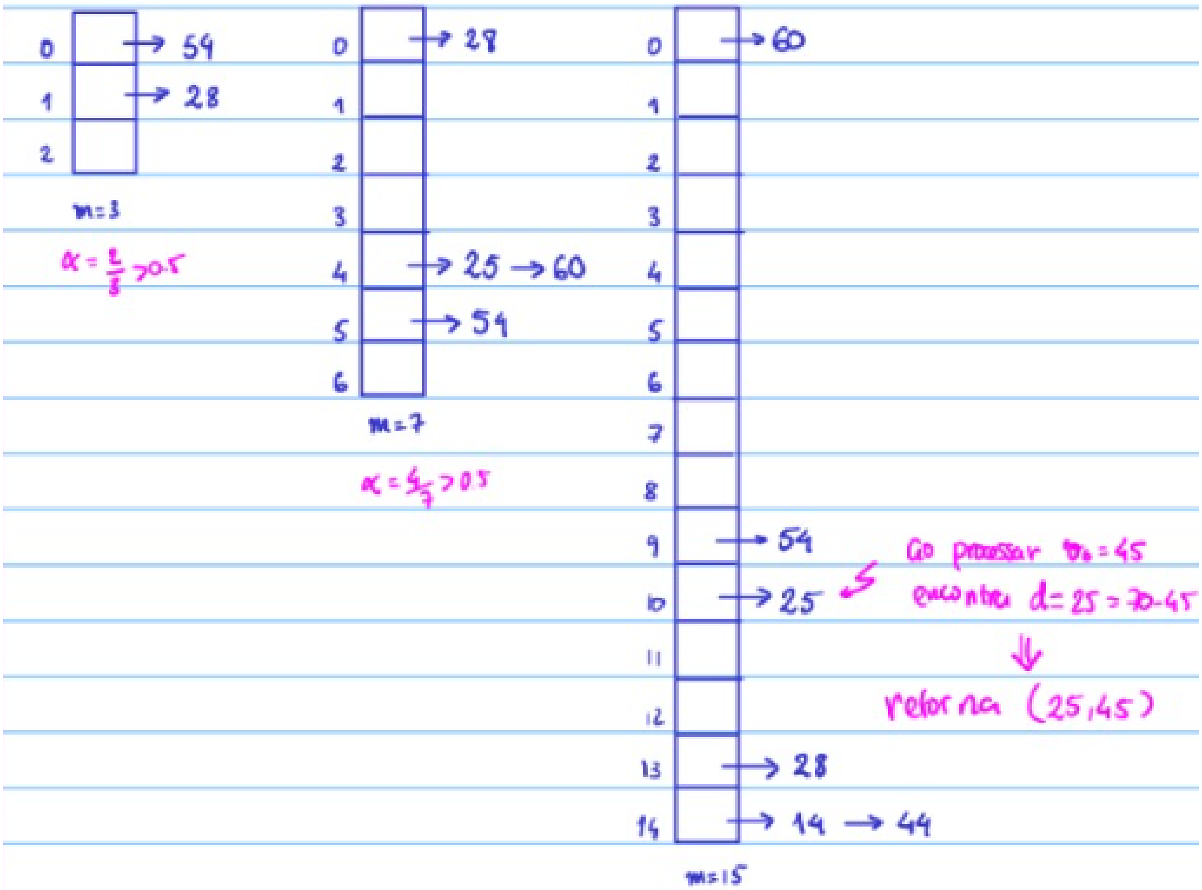
- 2, 9, 4, 7, 6, 5, 8, 3, 10, 1 ; $6 > 1$. ok

- 2, 9, 4, 7, 6, 5, 8, 3, 10, 1 ; $7 < 10$
- 2, 9, 4, 10, 6, 5, 8, 3, 7, 1 ; $9 < 10$
- 2, 10, 4, 9, 6, 5, 8, 3, 7, 1 ; $2 < 10$
- 10, 2, 4, 9, 6, 5, 8, 3, 7, 1 ;

- 10, 2, 4, 9, 6, 5, 8, 3, 7, 1 ; $4 < 8$
- 10, 2, 8, 9, 6, 5, 4, 3, 7, 1 ;

- 10, 2, 8, 9, 6, 5, 4, 3, 7, 1 ; $2 < 9$
- 10, 9, 8, 2, 6, 5, 4, 3, 7, 1 ; $2 < 7$
- 10, 9, 8, 7, 6, 5, 4, 3, 2, 1 ;

Questão 5:



m=3
α = 2/3 > 0.5
m=7
α = 4/7 > 0.5
m=15
Ao processar = 45
encontra d = 25 = 70-45
retorna (25,45)