



Prova
De 26/10/2020 até 28/10/2020
Valor: 40 pontos

Nome: Guilherme Henrique da Cunha Júnior

Instruções

1. A prova é individual;
2. A prova deve ser resolvida a próprio punho, em outras palavras, deve ser resolvida à mão;
3. O aluno deve entregar a resolução da prova de forma digitalizada, seja por meio de scanner ou fotografia;
4. Resoluções de prova ilegíveis não serão corrigidas;
5. A resolução da prova deve ser entregue por meio do sistema Google Sala de Aula.

1. Considere o vetor A formado pelas letras do seu nome completo em maiúsculo ignorando letras repetidas, acentos e espaço em branco.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
G	U	I	L	H	E	R	M	N	Q	D	O	A	F	S						

Por exemplo, o nome TARCISIO ALMEIDA, resulta no vetor:

1	2	3	4	5	6	7	8	9	10	11
T	A	R	C	I	S	O	L	M	E	D

Exiba o passo-a-passo da execução dos algoritmos a seguir:

- a) Ordenação do vetor A por meio do algoritmo de seleção. Mostre como fica o vetor a cada iteração. (1,5 pontos)
- b) Ordenação do vetor A por meio do algoritmo de inserção. Mostre como fica o vetor a cada iteração. (1,5 pontos)
- c) Ordenação do vetor A por meio do algoritmo Shellsort. Mostre como fica o vetor a cada iteração, destacando quais foram os saltos considerados. (4 pontos)
- d) Ordenação do vetor A por meio do algoritmo Quicksort. Mostre como fica o vetor a cada chamada recursiva, destacando as partições e os pivôs. (12 pontos)
- e) Ordenação do vetor A por meio do algoritmo Heapsort. Mostre como fica a *heap* ou a árvore que a represente a cada iteração. (4,5 pontos)
- f) Considerando que B seja o vetor A ordenado, faça uma pesquisa binária para encontrar o elemento A[1]. Mostre as comparações realizadas até encontrar a letra pesquisada. (4,5 pontos)

2. O algoritmo a seguir recebe um vetor v de números inteiros e rearranja esse vetor de tal forma que seus elementos, ao final, estejam ordenados de forma crescente.

```
01 void ordena(int *v, int n)
02 {
03     int i, j, chave;
04     for(i = 1; i < n; i++)
05     {
06         chave = v[i];
07         j = i - 1;
08         while(j >= 0 && v[j] < chave)
09         {
10             v[j+1] = v[j];
11             j = j - 1;
12         }
13         v[j+1] = chave;
14     }
15 }
```

Ordenação por inserção

* v = vetor

n = tamanho do vetor

Considerando que nesse algoritmo há erros de lógica que deve ser corrigidos para que os elementos sejam ordenados de forma crescente, informe quais são os ajustes necessários. (4 pontos)

3. Considere a função recursiva F a seguir, que em sua execução chama a função G:

```
1 void F(int n) {
2     if(n > 0) {
3         for(int i = 0; i < n; i++) {
4             G(i);
5         }
6         F(n/2);
7     }
8 }
```

Informe qual é a função de complexidade para a função F, considerando que a operação relevante é a chamada à função G. (4 pontos)

4. Faça uma análise geral dos métodos de ordenação e pesquisa. Apresente argumentos de quando é positivo ou negativo utilizar um determinado método. (4 pontos)

1. a)

$$\text{Interacão 1} \Rightarrow [A, U, I, L, H, E, R, M, N, Q, J, O, G, F, S]$$

$$\text{Interacão 2} \Rightarrow [A, D, I, L, H, E, R, M, N, Q, U, O, G, F, S]$$

$$\text{Interacão 3} \Rightarrow [A, D, E, L, M, I, R, M, N, Q, U, O, G, F, S]$$

$$\text{Interacão 4} \Rightarrow [A, D, E, F, M, I, R, M, N, Q, U, O, G, L, S]$$

$$\text{Interacão 5} \Rightarrow [A, D, E, F, G, I, R, M, N, Q, U, O, M, L, S]$$

$$\text{Interacão 6} \Rightarrow [A, D, E, F, G, H, R, M, N, Q, U, O, I, L, S]$$

$$\text{Interacão 7} \Rightarrow [A, D, E, F, G, H, I, M, N, Q, U, O, I, L, S]$$

$$\text{Interacão 8} \Rightarrow [A, D, E, F, G, H, I, L, M, N, Q, U, O, R, N, S]$$

$$\text{Interacão 9} \Rightarrow [A, D, E, F, G, H, I, L, M, N, O, U, R, Q, S]$$

$$\text{Interacão 10} \Rightarrow [A, D, E, F, G, H, I, L, M, N, O, U, R, Q, S]$$

$$\text{Interacão 11} \Rightarrow [A, D, E, F, G, H, I, L, M, N, O, Q, R, U, S]$$

$$\text{Interacão 12} \Rightarrow [A, D, E, F, G, H, I, L, M, N, O, Q, R, U, S]$$

$$b) 1 \Rightarrow [G]$$

$$2 \Rightarrow [G, U]$$

$$3 \Rightarrow [G, I, U]$$

4 $\Rightarrow [G, I, L, U]$

5 $\Rightarrow [G, H, I, L, U]$

6 $\Rightarrow [E, G, H, I, L, U]$

7 $\Rightarrow [E, G, H, I, L, R, U]$

8 $\Rightarrow [E, G, H, I, L, M, R, U]$

9 $\Rightarrow [E, G, H, I, L, M, N, R, U]$

10 $\Rightarrow [E, G, H, I, L, M, N, Q, R, U]$

11 $\Rightarrow [D, E, G, H, I, L, M, N, Q, R, U]$

12 $\Rightarrow [D, E, G, H, I, L, M, N, O, Q, R, U]$

13 $\Rightarrow [A, D, E, G, H, I, L, M, N, O, Q, R, U]$

14 $\Rightarrow [A, D, E, F, G, H, I, L, M, N, O, Q, R, U]$

15 $\Rightarrow [A, D, E, F, G, H, I, L, M, N, O, Q, R, S, U]$

c) $h = 13$ (dolto)

[G, U, I, L, H, E, R, M, N, Q, D, O, A, F, S]

A;

F, R

[F, U, I, L, H, E, R, M, N, Q, D, O, A, F, S]

$h = 6$

$[A, U, I, L, H, E, F, M, N, Q, D, O, R, G, S]$

$H = 3$

$[A, U, I, F, H, E, L, M, N, Q, D, O, R, G, S]$

$H = 3$

$[A, D, E, F, G, H, I, L, M, N, O, Q, R, S, U]$

a) $\Rightarrow [G, U, I, L, H, E, R, M, N, Q, D, O, A, F, S]$

$P(\text{pivot}) = M \quad E(\text{esquerda}) = G \quad D(\text{direita}) = S$

quicksort (vetor, E, P) \rightarrow chamadas recursivas

quicksort (vetor, P+1, D)

$\Rightarrow [G, F, I, L, H, E, A, D, N, Q, M, O, R, U, S]$

$P = A \quad E = G \quad D = D$

quicksort (vetor, E, P)

quicksort (vetor, P+1, D)

3 $[G, F, I, D, H, E, A, L, N, Q, M, O, R, U, S]$

$P = F \quad E = G \quad D = A$

quicksort (vetor, E, P)

quicksort (vetor, P+1, D)

4 $[A, D, I, F, H, E, G, L, N, Q, M, O, R, U, S]$

$P = A \quad E = A \quad D = D$

quicksort (vetor, E, P)

quicksort (vetor, P+1, D)

5 $[A, D, I, F, H, E, G, L, N, Q, M, O, R, U, S]$

$P = H \quad E = I \quad D = G$

quicksort (vetor, E, P)

quicksort (vetor, P+1, D)

b $[A, O, G, F, E, H, I, L, N, Q, M, O, R, U, S]$

$P = F \quad E = F \quad D = G$

quicksort (vetor, E, P)

quicksort (vetor, P+1, D)

7 [A, D, E, F, G, H, I, L, N, Q, M, O, R, U, S]

$$P = E \quad E = E \quad D = F$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

8 [A, D, E, F, G, H, I, L, N, Q, M, O, R, U, S]

$$P = H \quad E = H \quad D = I$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

9 [A, D, E, F, G, H, I, L, N, Q, M, O, R, U, S]

$$P = M \quad E = N \quad D = S$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

10 [A, D, E, F, G, H, I, L, N, O, M, Q, R, U, S]

$$P = O \quad E = N \quad D = M$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

11 [A, D, E, F, G, H, I, L, N, M, O, Q, R, U, S]

$$P = N \quad E = N \quad D = M$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

12 [A, D, E, F, G, H, I, L, M, N, O, Q, R, U, S]

$$P = R \quad E = Q \quad D = S$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

13 [A, D, E, F, G, H, I, L, M, N, O, Q, R, U, S]

$$P = Q \quad E = Q \quad D = R$$

quicksort (vector, E, P)

quicksort (vector, P+1, D)

14 [A, D, E, F, G, H, I, L, M, N, O, Q, R, U, S]

P = U

E = U

D = S

quicksort (vetor, t, p)

quicksort (vetor, p+1, d)

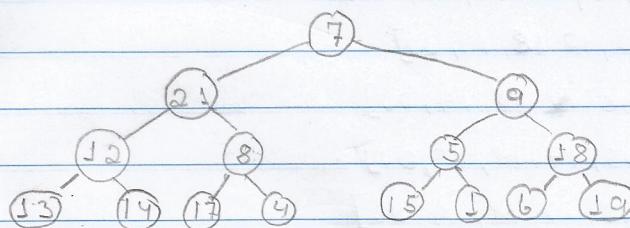
15 [A, O, E, F, G, H, I, L, M, N, O, Q, R, S, U]

a) Para facilitar o desenvolvimento, vou atribuir um número para cada letra.

[G, U, I, L, H, E, R, M, N, Q, D, O, A, F, S]

[7, 23, 9, 12, 8, 5, 18, 13, 14, 17, 4, 15, 1, 6, 19]

2. i+1 2. i+2



1. [7, 23, 9, 12, 8, 5, 18, 13, 14, 17, 4, 15, 1, 6, 19]

2. [7, 23, 9, 12, 8, 5, 19, 13, 14, 17, 4, 15, 1, 6, 18]

3. [7, 23, 9, 12, 8, 15, 19, 13, 14, 17, 4, 5, 1, 6, 18]

4. [7, 23, 9, 12, 17, 15, 19, 13, 14, 8, 4, 5, 1, 6, 18]

5. [7, 23, 9, 14, 17, 15, 19, 13, 12, 8, 4, 5, 1, 6, 18]

6. [7, 23, 19, 14, 17, 15, 9, 13, 12, 8, 4, 5, 1, 6, 18]

7. [7, 23, 19, 14, 17, 15, 18, 13, 12, 8, 4, 5, 1, 6, 9]

8. [23, 7, 19, 14, 17, 15, 18, 13, 12, 8, 4, 5, 1, 6, 9]

9. [23, 17, 19, 14, 7, 15, 18, 13, 12, 8, 4, 5, 1, 6, 9]

10. [23, 17, 19, 14, 8, 15, 18, 13, 12, 7, 4, 5, 1, 6, 9]

11. [19, 17, 9, 14, 8, 15, 18, 13, 12, 7, 4, 5, 1, 6, 23]

12. [19, 17, 18, 14, 8, 15, 9, 13, 12, 7, 4, 5, 1, 6, 23]

13. [18, 17, 6, 14, 8, 15, 9, 13, 12, 7, 4, 5, 1, 19, 23]

14. [18, 17, 15, 14, 8, 6, 9, 13, 12, 7, 4, 5, 1, 19, 23]

15. [12, 3, 15, 14, 8, 6, 9, 13, 12, 7, 4, 5, 18, 19, 25]

16. [17, 14, 15, 3, 8, 6, 9, 13, 12, 7, 4, 5, 18, 19, 25]

17. [17, 14, 15, 13, 2, 6, 9, 1, 10, 7, 4, 5, 18, 19, 25]

18. [15, 14, 5, 13, 2, 6, 9, 1, 12, 7, 4, 17, 18, 19, 25]

19. [15, 14, 9, 13, 8, 6, 5, 1, 10, 7, 4, 17, 4, 18, 19, 25]

20. [14, 4, 9, 13, 2, 6, 5, 1, 12, 7, 15, 17, 18, 19, 25]

21. [14, 13, 9, 9, 8, 6, 5, 1, 10, 7, 15, 17, 18, 19, 25]

22. [14, 13, 9, 12, 8, 6, 5, 1, 4, 7, 15, 17, 18, 19, 25]

23. [13, 7, 9, 12, 2, 6, 5, 1, 4, 14, 15, 17, 18, 19, 25]

24. [13, 12, 9, 7, 8, 6, 5, 1, 4, 14, 15, 17, 18, 19, 25]

25. [12, 4, 9, 7, 8, 6, 5, 1, 13, 14, 15, 17, 18, 19, 25]

26. [12, 8, 9, 7, 6, 5, 1, 13, 14, 15, 17, 18, 19, 25]

27. [9, 8, 1, 7, 4, 6, 5, 12, 13, 14, 15, 17, 18, 19, 25]

28. [9, 8, 6, 7, 4, 1, 5, 12, 13, 14, 15, 17, 18, 19, 25]

29. [8, 5, 6, 7, 4, 1, 9, 10, 13, 14, 15, 17, 18, 19, 25]

30. [8, 5, 6, 7, 4, 1, 9, 12, 13, 14, 15, 17, 18, 19, 25]

31. [8, 7, 6, 5, 4, 1, 9, 12, 13, 14, 15, 17, 18, 19, 25]

32. [7, 1, 6, 5, 4, 8, 9, 12, 13, 14, 15, 17, 18, 19, 25]

33. [7, 5, 6, 3, 4, 8, 9, 12, 13, 14, 15, 17, 18, 19, 25]

34. [6, 5, 4, 1, 7, 8, 9, 12, 13, 14, 15, 17, 18, 19, 25]

35. [5, 1, 4, 6, 7, 8, 9, 12, 13, 14, 15, 17, 18, 19, 25]

36. [1, 4, 5, 6, 7, 2, 9, 12, 13, 14, 15, 17, 18, 19, 25]

[A, D, E, F, G, H, I, L, M, N, O, Q, R, S, U]

4) Para facilitar o desenvolvimento, vou fazer a mesma atribuição numérica da questão anterior.

$$A = [7, 2, 1, 9, 10, 8, 5, 18, 13, 14, 17, 4, 15, 1, 6, 19]$$

$$B = [1, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 17, 18, 19, 25]$$

$$A[1] = 2$$

Número a ser encontrada = 21 (variável m)

Vetor de 15 posições, inicio (i) = 0, final (F) = 14 = meio(m) = $(i+F)/2$

Vetor = (variável v)

$\Rightarrow \rightarrow$ é igual à

$v[m] == m \Rightarrow$ Não

$v[m] > m \Rightarrow$ Não

$v[m] < m \Rightarrow$ sim $\Rightarrow i = m + 1$

$$m = (i+F)/2$$

$v[m] == m \Rightarrow$ Não

$v[m] > m \Rightarrow$ Não

$v[m] < m \Rightarrow$ sim $\Rightarrow i = m + 1$

$$m = (i+F)/2$$

$v[m] == m \Rightarrow$ Não

$v[m] > m \Rightarrow$ Não

$v[m] < m \Rightarrow$ sim $\Rightarrow i = m + 1$

$$m = (i+F)/2$$

$v[m] == m \Rightarrow$ sim \Rightarrow break

2. void ordena (int v[], int m) {

 int j, chave;

 for (int i = 1; i < m; i++) {

 chave = v[i];

 j = i - 1;

 while (j >= 0 && v[j] > chave) {

 v[j+1] = v[j];

 j = j - 1;

}

 v[j+1] = chave;

y

z

3. $F(x) = O(n^2)$

4. Ordenação por Seleção:

É um algoritmo de ordenação baseado em se passar sempre o menor valor do vetor para a segunda posição, e assim é feito sucessivamente com os $n-1$ elementos restantes, até os últimos 2 elementos.

Vantagens = . Fácil implementação

- Não necessita de um vetor auxiliar
- Ocupa menos memória, pois não utiliza um vetor auxiliar
- É um dos mais rápidos em vetores de tamanhos pequenos

Desvantagens = . Ele é um dos mais lentos para vetores de tamanhos grandes

- Não é estável
- Sempre faz $(n^2-n)/2$ comparações, independentemente do vetor estar ordenado ou não

• Ordenação por Inserção:

É um algoritmo de ordenação que, dado uma estrutura contém uma matriz final com os elementos de cada vez, uma inserção por vez.

Vantagens = . É comum a ser utilizado quando o arquivo está "quase" ordenado.

- É um bom método quando se desejam adicionar poucos elementos em um arquivo já ordenado, pois seu custo é linear
- É estável

Desvantagem = . Alto custo de movimentação.

• Shell Sort

É o algoritmo mais eficiente de classificações dentre os de complexidade quadrática. Ordena parcialmente os dados por meio de sucessivos saltos e trocas para depois ordenar por inserção.

Vantagens = • Implementação Fácil

- Eficiente

Desvantagens = • Instável

• Quick Sort

Estratégia da divisão e conquista.

Vantagens = • Extremamente eficiente

Desvantagens = • Difícil Implementação

- Importância do pivô para evitar o pior caso
- Instável

• Heap Sort

É um melhoramento da ordenação por seleção, utiliza a estrutura de dados heap para ordenar um vetor de dados.

Vantagens = • Eficiente

- Previsível

Desvantagens = . Implementação difícil
. Ineficiente

• Busca Linear

Pesquisa de maneira seqüencial em vetores ou listas, comparando elementos por elemento

Vantagem

Vantagem = . Fácil implementação

Desvantagem = . Lento para vetores grandes.

Pesquisa Binária

é um algoritmo de busca em vetores que segue o paradigma da divisão e conquista.

Vantagens = . Eficiente

. Fácil implementação

Desvantagens = . Vetor precisa estar ordenado