Universidade Federal do Maranhão
Departamento de Informática

Estruturas de Dados II (DEIN0083) 2024.2
Curso de Ciência da Computação
1ª avaliação

Prof. João Dallyson Sousa de Almeida **Data**: 16/12/2024

Aluno: João Guilherme Rapose Labato **Matrícula**: [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]

Regras durante a prova:

- 2)É vetada: a consulta a material de apoio, conversa com colega e a utilização de dispositivos eletrônicos. A não observância de algum dos itens acima acarretará a anulação da prova.

I. (2.0pt) Responda às seguintes questões sobre algoritmos de ordenação que discutimos em sala de aula.
a) Qual é a complexidade de inserir um novo item em uma lista encadeada ordenada em ordem decrescente, de modo que a lista permaneça ordenada? Justifique sua resposta.
b) Temos um sistema executando a ordenação com o QuickSort e descobrimos que ele está sendo concluído mais devagar do que o esperado. O que podemos concluir sobre a entrada fornecida ao sistema?
c) Forneça um vetor de 6 números inteiros que obtenha o melhor caso de tempo de execução para o Insertion Sort.
d) O ShellSort é um método de ordenação estável? Justifique e exemplifique a sua resposta.

II. (1.0pt) Considere os seguintes algoritmos recursivos que resolvem o mesmo problema em uma entrada de tamanho n:
Algoritmo 1: Divide o problema em 3 partes de tamanho n/16 e gasta um tempo adicional de $\sqrt{n}$
Algoritmo 2: Divide o problema em 5 partes de tamanho n/3 e gasta um tempo adicional n
Apresente a complexidade dos algoritmos 1 e 2. Descreva a solução e mostre a ordem de complexidade.

III. (2.0pt) Modifique o algoritmo do ShellSort (Figura abaixo), trocando a ordenação original das sub-listas pelo método do Select Sort. Descreva a solução, apresente a nova versão e discuta a complexidade computacional da nova versão em relação à versão original.

```
int h, temp, j;

for (h = 1 ; h < V.length; h = (2*h)+1);

while (h > 0) {
    h = (h-1)/2;

    for (int i = h; i < V.length; i++) {
        temp = V[i];
        j = i;
        while (V[j-h] > temp) {
            V[j] = V[j-h];
            j = j - h;
            if (j < h) break;
        }
        V[j] = temp;
    }
}
return V;
```

IV. (3.0pt) Considere a seguinte sequência de chaves do vetor [5, 1, 0, 2, 9, 3]. OBS: Todos os itens a seguir devem considerar o vetor original.
a) Mostre o vetor resultante após a segunda iteração do método "Particiona" do algoritmo do QuickSort para ordenar em ordem crescente. Utilize o elemento da **esquerda como pivô**.
b) Mostre o passo a passo da construção do Heap e o vetor resultante após 3 execuções completas do HeapSort para ordenar em ordem **decrescente**. O Heap deve ser construído usando o vetor de entrada.
c) Mostre o passo a passo da ordenação utilizando o MergeSort. Mostre o número de comparações e trocas realizadas.

V. (2.0pt) Apresente o custo de execução e a análise assintótica do algoritmo abaixo. Os algoritmos A e B pertencem a quais classes de complexidade assintótica? Justifique sua resposta.

(A)

```
int aux1 = 0;
int aux2 = 0;

for (int i=1; i < M; i*=3)
    for (int j=0; j < N; j++)
        for (int k = 0; k < N; k = k++)
                aux1++;
print(aux1);
```

(B)

```
int aux1 = 0;
int aux2 = 0;

for (int i=M; i > 0; i=i/3)
    for (int j=0; j < M; j++)
        aux1++;
print(aux1);

for (int i = 1 ; i < M; i*=2)
    aux2++;
print(aux2);
```

Aluno: João Guilherme Lopes Lobato

2.0

Questão 01

a) Devido ao fato de estarmos trabalhando com uma lista encadeada, o melhor algoritmo para fazer a inserção será $O(n)$, já que não podemos indexar a lista em tempo constante para aplicar uma busca binária, portanto nossa complexidade fica $T(n) = n+1 = O(n)$ (Pior caso: insere no final ($O(n)$), Melhor caso: insere no começo ($\Omega(1)$))

b) Podemos concluir que o vetor já está ordenado na ordem desejada, já que esse é o cenário que leva ao pior caso do QuickSort ($O(n^2)$)

c) $V = \{1, 1, 1, 1, 1, 1\}$

d) Não, pois o ShellSort é um algoritmo "descendente" do InsertionSort que não faz mudança direta a fim de garantir a estabilidade. Como exemplo temos o vetor $v = \{3, 2, 2, 3\}$. Aplicando $h=2$ e $h=1$, temos:

- $h=2$:

  $\{3, ②, [2], 1\} \Rightarrow$ Sublistas: $\{3, [2]\}$ $\{②, 1\} \Rightarrow \{[2], 3\}, \{1, ②\}$

  $\Rightarrow$ Resultado: $\{[2], ②, 3, 1\}$

- $h=1$:

  $\{[2], ②, 3, 1\} \Rightarrow \{[2], ②, 3, 1\} \Rightarrow \{[2], ②, 3, 1\} \Rightarrow \{[2], ②, 1, 3\}$

  $\Rightarrow \{[2], 1, ②, 3\} \Rightarrow \{1, [2], ②, 3\}$ (fim do sort)

Logo, notamos que a ordem relativa de [2] e ② foi trocada, tornando-o instável.

__/__/__

1,0

Questão 02

Podemos definir as complexidades dos algoritmos como as recorrências:

$T_1(n) = 3T_1(n/16) + \sqrt{n}$

$T_2(n) = 5T_2(n/3) + n$

Como ambos tem o segundo termo pertencente a $\Theta(n^d)$, podemos aplicar o teorema mestre:

- P/ T_1: $a = 3$, $b = 16$, $d = 1/2 \Rightarrow b^d = 4 > a$

$\Rightarrow T_1(n) \in \Theta(n^{\frac{1}{2}}) \in \Theta(\sqrt{n})$

- P/ T_2: $a = 5$, $b = 3$, $d = 1 \Rightarrow b^d = 3 < a$

$\Rightarrow T_2(n) \in \Theta(n^{\log_3 5})$

1,5

Questão 03

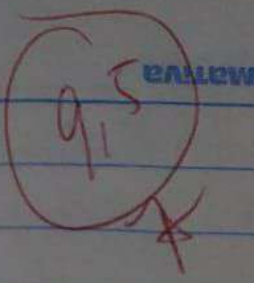
```
int[] SelectionShellSort(int[] v){
int n = v.lenght, h, min, temp;
for(h=1; h<n; h=(2*h)+1);
while(h>0){
  h=(h-1)/2;
  for(int i=0; i<n; i+=h){
    min=i;
    for(int j=i+h; j<n; j+=h)
      if(v[j]<v[min])
        min=j;
    temp = v[i];
    v[i] = v[min];
    v[min]=temp;
  }
  return v;
}
```

falta um loop p/ controlar a qtd de sublistas

alternativa

João Guilherme Ropero Lobato

Questão 04



$v = \{S, 1, 0, 2, 9, 3\}$

a) Pivô: S

$\{S, 1, 0, 2, 9, 3\} \Rightarrow \{S, 1, 0, 2, 9, 3\} \Rightarrow \{S, 1, 0, 2, 9, 3\}$

$\Rightarrow \{S, 1, 0, 2, 9, 3\} \Rightarrow \{S, 1, 0, 2, 3, 9\} \Rightarrow \{S, 1, 0, 2, 3, 9\}$

Próxima partição

$\{9\}$ → não faz nada

$\{3, 1, 0, 2\}$ → Pivô: 3 $\Rightarrow \{3, 1, 0, 2\} \Rightarrow \{3, 1, 0, 2\} \Rightarrow \{3, 1, 0, 2\}$

$\Rightarrow \{3, 1, 0, 2\} \Rightarrow \{2, 1, 0, 3\}$

Depois da 2 partição: $\{2, 1, 0, 3, 5, 9\}$

b) Construindo Heap:

$\{S, 1, 0, 2, 9, 3\} \Rightarrow l = 6, r = 7, small = 3 \Rightarrow$ não troca

$\{S, 1, 0, 2, 9, 3\} \Rightarrow l = 1, r = S, small = 2, \Rightarrow$ não troca

$\{S, 1, 0, 2, 9, 3\} \Rightarrow l = 2, r = 3, small = 3 \Rightarrow$ troca 1 e 3

$\{0, 1, S, 2, 9, 3\} \Rightarrow l = 6, r = 7, small = 6 \Rightarrow$ troca 3 e 6

$\{0, 1, 3, 2, 9, S\} \Rightarrow l = 12, r = 13, small = 6 \Rightarrow$ não troca.

O heap construído no vetor foi: $\{0, 1, 3, 2, 9, S\}$

04-b) Ordenando

{0, 1, 3, 2, 9, 5} ⇒ Troca 1 e 6 e reajusta

{5, 1, 3, 2, 9 | 0} ⇒ l=2, r=3, small=2 ⇒ troca 1 e 2

{1, 5, 3, 2, 9 | 0} ⇒ l=4, r=5, small=4 ⇒ troca 2 e 4

{1, 2, 3, 5, 9 | 0} ⇒ não troca, reajustado

{1, 2, 3, 5, 9, 0} ⇒ Troca 1 e 5 e reajusta

{9, 2, 3, 5 | 3, 0} ⇒ l=2, r=3, small=2 ⇒ troca 1 e 2

{2, 9, 3, 5 | 1, 0} ⇒ l=4, r=5, small=4 ⇒ troca 2 e 4

{2, 5, 3, 9 | 1, 0} ⇒ não troca, reajustado

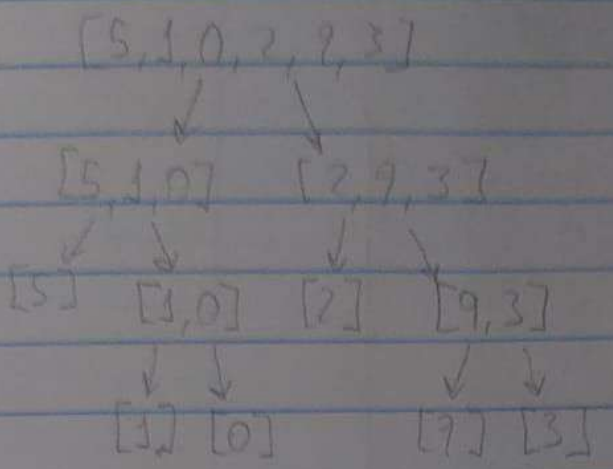
{2, 5, 3, 9, 1, 0} ⇒ Troca 1 e 4 e reajusta

{9, 5, 3 | 2, 1, 0} ⇒ l=2, r=3, small=3 ⇒ Troca 1 e 3

{3, 5, 9 | 2, 1, 0} ⇒ não troca, reajustado

Ao final das 3 execução do HeapSort, temos v = {3, 5, 9, 2, 1, 0}

04-c) Dividindo



Conquistando:

[5] e [0]

[0,5] → 1 comp, 2 mov

[9] e [3]

[3,9] → 1 comp, 2 mov

[5] e [0,1]

[0,1,5] → 2 comp, 3 mov

[2] e [3,9]

[2,3,9] → 2 comp, 3 mov

[0,1,5] e [2,3,9]

[0,1,2,3,5,9] → 5 comp, 6 mov

Quantidade de Comparações = 1+1+2+2+5 = 11

Quantidade de Movimentos = 2+2+3+3+6 = 16

Continuação - Questão 03

O algoritmo segue a mesma linha de raciocínio que o ShellSort padrão, mas tem uma falha na lógica quando chamou o algoritmo interno utilizado (SelectionSort). A ideia original era diminuir o número de comparações e trocas fazendo com que o vetor parcialmente ordenado ao final, Entretanto, o selection não diminui a quantidade de comparações por não ter um "melhor caso". Portanto, o algoritmo permanece $\Theta(n^2)$, só que com mais passos.

2,0

Questão 05

a)
```
int aux1 = 0;                    1+
int aux2 = 0;                    1+
for (i = 1; i < M; i *= 3)       log3M *(
  for (j = 0; j < N; j++)        N*(
    for (k = 0; K < N; k = K++)  N*(
      aux1++;                    1 ))) 1+
print (aux1);                    1
```

Complexidade: $T(n) = 3 + N^2 \log_3 M$

$\Rightarrow \lim_{n\to\infty} \frac{n^2 \log_3 m + 3}{n^2 \log_3 m} \quad \lim_{n\to\infty} \frac{n^2 \log_3 m}{n^2 \log_3 m} + \frac{3}{n^2 \log_3 m} = 1$

$\therefore T(n) \in \Theta(n^2 \log_3 m) \in \Theta(n^2 \log m)$

05-b)

```
aux1 = 0;
aux2 = 0;
for (i = M; i > 0; i = i/3)
    for (j = 0; j < M; j++)
        aux1++;
print (aux1);
for (i = 1; i < M; i *= 2)
    aux2++;
print (aux2);
```

$T(n) = 4 + M \log_3 M + \log_2 M$

$\Rightarrow \lim_{n \to \infty} \frac{M \log_3 M}{M \log M} + \frac{\log_2 M}{n \log M} + \frac{4}{n \log M} = \lim_{n \to \infty} \frac{\frac{\log M}{\log 3}}{\log M} + \frac{\frac{\log M}{\log 2}}{\log M} = \frac{1}{\log 3} + \frac{1}{\log 2}$

$\therefore T(n) \in \Theta(M \log M)$