

Análise de Algoritmos:

→ grafos

1) $G = (V, E)$ → grafo Direcionado

- a) Based on the Strongly Connected Algorithm in Lecture 5 (Graphs), we should test to see if there is a node in V connected to all other nodes from the graph.

Algorithm: ~ Based on Strongly Connected Algorithm in Lecture 5

exists ← false

u ~ node

for all u in V run DFS(u)

if the search visits all nodes

exists ← true

return exists

end if

end

return exists

Analysis: $\Omega(n+m)$ ~ runs DFS once. $O(n(m+n))$, $n = |V|$ and $m = |E|$, ~ runs DFS
 m times.b) $G \sim$ DAG (Directed Acyclic Graph)

As seen in Lecture 5 (Graphs),

"If G is a DAG, then G has a topological ordering."Therefore, we should run a topological ordering in $O(m+n)$ time and then run BFS to see if the node with

15/05/20

no incoming edges visits all other nodes from G_{II}

Analysis: $\text{charavith minima em } (E, V) = \Theta(|V|)$

$\{ \text{Topological ordering} \rightsquigarrow O(m+n) \}$

$\{ \text{BFS} \rightsquigarrow O(m+n) \}$

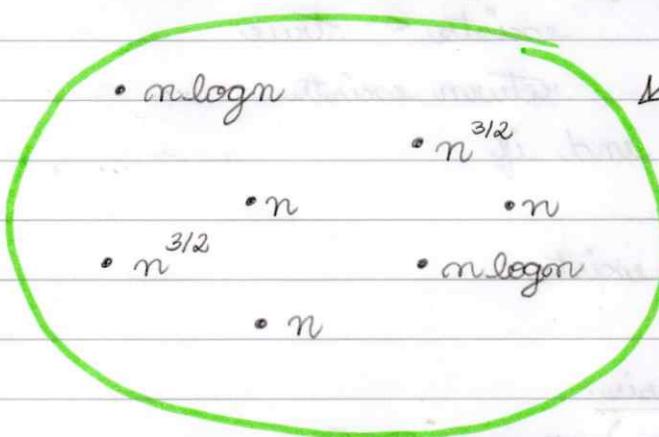
Algorithm Complexity $\rightsquigarrow O(m+n)$, $m = |V|$ and $m = |E|$

2) Basic Notions of Complexity

a) FALSO, pois se o algoritmo possui complexidade de pior caso em $\Omega(n^{1.5})$, então é possível afirmar que há pelo menos um input para o qual a complexidade do algoritmo tem ordem $n^{1.5}$

Vide exemplo na aula Basic Notions of Complexity:

inputs of size n
for algorithm A



b) $A \rightsquigarrow \Theta(n^2 \log n)$ $\left\{ \begin{array}{l} \Omega(n^2 \log n) \\ O(n^2 \log n) \end{array} \right.$

$B \rightsquigarrow \Theta(n^2 \sqrt{n})$ $\left\{ \begin{array}{l} \Omega(n^2 \sqrt{n}) \\ O(n^2 \sqrt{n}) \end{array} \right.$

FALSO, pois o fato do algoritmo B ter complexidade de pior caso em $\Theta(n^2 \sqrt{n})$ nos permite dizer apenas que há um input para o qual a complexidade do algoritmo B tem ordem $n^{5/2}$ e não há inputs cu

já complexidade tende ordem maior que $m^2 \log n$

$$\begin{aligned} c) A &\sim O(m^2 \log n) \\ B &\sim O(m^2 \sqrt{m}) \end{aligned}$$

FALSO, pois é possível apenas afirmar que, para o algoritmo A, não há inputs cuja complexidade tenha ordem maior do que $m^2 \log n$ e, para o algoritmo B, não há inputs cuja complexidade tenha ordem maior do que $m^2 \sqrt{n}$.

3) $A = \{a_1, a_2, \dots, a_n\}$

$$m_j, j = 1, \dots, n$$

$$1 \leq \alpha < \beta \leq n$$

$$\sum_{i=\alpha}^{\beta} m_i = ?$$

$$\underbrace{m_\alpha + \dots + m_\beta}_{n \text{ elementos}}$$

Algoritmo de somatório:

soma = \emptyset //Inicializa a soma zerada.

Para $i=0$ até $n-1$ faça

aux $\leftarrow a_i$

soma \leftarrow soma + aux

Fim Para

Devolve soma

Análise:

As cada loop são realizadas $O(1)$ operações elementares. Logo, o tempo é linear e a complexidade

17/05/20

de do algoritmo é $O(n)$. //

Se ordenássemos A,

$(n \log^2 n) O \sim A$ (S)

Quicksort $\sim O(n^2)$

$(\frac{1}{2} n \log n) O \sim S$

Mergesort $\sim O(n \log n)$

Shellsort $\sim O(n \log n)$

Bubble Sort $\sim O(n^2)$

Insertion Sort $\sim O(n^2)$

Selection Sort $\sim O(n^2)$

→ (Median of Medians)

Algoritmo escolhido \sim MOM-algo $\sim O(n)$ //

$a = \text{MOM_algo}(A, \alpha)$; // Encontra o α -ésimo menor elemento de A.

$b = \text{MOM_algo}(A, \beta)$; // Encontra o β -ésimo menor elemento de A.

soma = $a + b$;

Para $i = 0$ até $n-1$ faça

se $a_i > a$ e $a_i < b$

soma += a_i ;

Fim Para

Análise:

MOM-algo $\sim O(n)$

Para $\sim O(n)$

Complexidade do Algoritmo $\sim O(n) \sim$ É linear. //

} Próxima Pág...

4)

```

MisterM( $n$ )
if  $n = 1$  then _____
    Return 1 _____
else _____
     $Z = 0$  _____
    for  $i = 1; i \leq n - 1; i++$ 
        for  $j = i + 1; j \leq n; j++$ 
             $Z = Z + \text{Magic}(i, j)$ 
        end for
    end for
     $Z = Z + \text{MisterM}(n - 1)$ 
    Return  $Z$  _____
end if

```

$\text{Magic}(i, j)$

```

X = 0
for k = i; k ≤ j; k++ do
    if k é um múltiplo de 7 then
        X = X + 1
    end if
end for
Return X

```

→ O(1)

→ O(n)

→ O(1)

→ O(1)

→ O(1)

$\rightarrow O(1)$

$\rightarrow O(1)$

$\rightarrow O(1)$

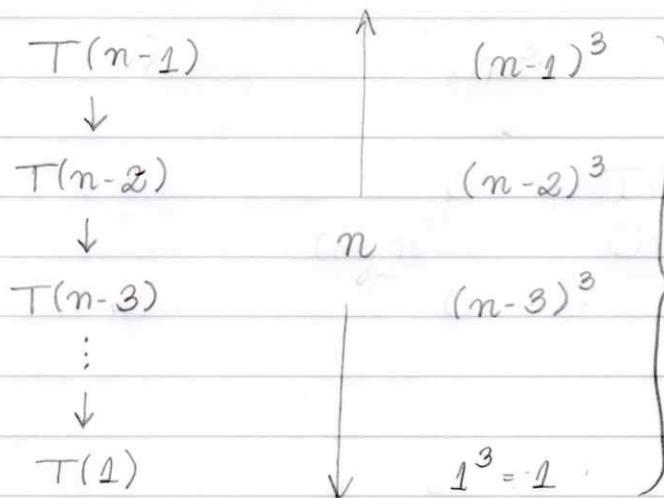
$\rightarrow O(n) \quad \left\{ n^2 \times (n) = n^3 + O(4) \right.$

$\rightarrow O(n)$

$\rightarrow O(1)$

Considerei todos os $O(1) = c_j$

$$T(n) = c + n^3 + T(n-1) \quad \text{pr Recursão}$$



$$T(n) = \sum_{k=1}^{n-1} (n-k)^3 \leq \sum_{k=1}^n k^3$$

$= n^4 // \sim T(n) \sim O(n^4)$
 Logo, $T(n)$ será
 $O(n^4) //$

Melhor caso $\sim m = 1 \sim \Omega(2) = \Omega(c)$

19/05/20

5) $G = (V, E)$

↳ Fortemente conectado

↳ Direcionado

↳ Com pesos $\sim \{1, 2, 3, 4\}$

↳ $O(|E| + |V|)$

↳ Caminho de menor peso entre um par de vértices (u, v)

Teria necessário executar o algoritmo de Dijkstra para encontrar o caminho de menor peso de u a v .

Initialize ~ 1
ExtractMin $\sim n$
ChangeKey $\sim m$

BFS \sim caminho mínimo $\sim O(m+n)$

Percorro o grafo e substituo as arestas de peso 2 por um novo vértice entre essas arestas, de peso 3 por 2 novos vértices e peso 4 por 3 novos vértices. Em seguida, executo a BFS para determinar o caminho mínimo.

