15/05/20

Análise de Algoritmos:

1) $G = (V, E)$ ~> grafos ~> 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 n times.

b) G ~> 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 exists all other nodes from G.//

Analysis:

$\begin{cases} \text{Topological ordering} \rightsquigarrow O(m+n) \\ \text{BFS} \rightsquigarrow O(m+n) \end{cases}$

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

- $n \log n$
- $n^{3/2}$
- n
- n
- $n^{3/2}$
- $n \log n$
- n

b) $A \rightsquigarrow \Theta(n^2 \log n) \begin{cases} \Omega(n^2 \log n) \\ O(n^2 \log n) \end{cases}$

$B \rightsquigarrow \Theta(n^2 \sqrt{n}) \begin{cases} \Omega(n^2 \sqrt{n}) \\ O(n^2 \sqrt{n}) \end{cases}$

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

15/05/20

ja complexidade tenda ordem maior do que $n^{5/2}$.

c) $A \rightsquigarrow O(n^2 \log n)$
$B \rightsquigarrow O(n^2\sqrt{n})$

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

3) selection
$A = \{a_1, a_2, \dots, a_n\}$
$m_j \ , \ j = 1, \dots, n$
$1 \leq \alpha < \beta \leq n$
$\sum_{i=\alpha}^{\beta} m_i = ?$ $\rightarrow \underbrace{m_\alpha + \dots + m_\beta}_{n \text{ elementos}}$

Algoritmo do somatório:

soma = $\emptyset$ // Inicializa a soma zerada.

Para i = 0 até n-1 faça
    aux ← ai
    soma ← soma + aux
Fim Para
Devolva soma

Análise:
A cada loop são realizadas O(1) operações elementares. Logo, o tempo é linear e a complexida

17/05/20

de do algoritmo é $O(n)$.//

Se ordenássemos A,
→ Quicksort → $O(n^2)$
→ Mergesort → $O(n \log n)$
→ Heapsort → $O(n \log n)$
→ Bubble Sort → $O(n^2)$
→ Insertion Sort → $O(n^2)$
→ Selection Sort → $O(n^2)$

(Median of Medians)

<u>Algoritmo escolhido</u> → MOM_algo → $O(n)$//

```
a = MOM_algo(A, α);   // Encontra o α-ésimo menor elemento de A.
b = MOM_algo(A, β);   // Encontra o β-ésimo
soma = a + b;         menor elemento de A.
Para i = 0 até n-1 faça
    se ai > a e ai < b
        soma += ai;
Fim Para
```

<u>Análise</u>:

MOM_algo → $O(n)$
Para → $O(n)$
Complexidade do Algoritmo → $O(n)$ → É linear.//

Próxima Pág...

19/05/20

4)

```
MisterM(n)
  if n = 1 then                                  → O(1)
    Return 1                                     → O(1)
  else
    Z = 0                                        → O(1)
    for i = 1; i ≤ n − 1; i++ do                 → O(n)
      for j = i + 1; j ≤ n; j++ do               → O(n)   } n² × n = n³ + O(4)
        Z = Z+Magic(i, j)
      end for
    end for
    Z = Z+MisterM(n − 1)
    Return Z                                     → O(1)
  end if

Magic(i, j)
  X = 0                                          → O(1)
  for k = i; k ≤ j; k++ do                       → O(n)
    if k é um múltiplo de 7 then                 → O(1)
      X = X + 1                                  → O(1)
    end if
  end for
  Return X                                       → O(1)
```

Considerei todos os $O(1) = c$

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

$T(n-1)$	n	$(n-1)^3$
$\downarrow$		
$T(n-2)$		$(n-2)^3$
$\downarrow$		
$T(n-3)$		$(n-3)^3$
$\vdots$		
$\downarrow$		
$T(1)$		$1^3 = 1$

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

$$= n^4$$

Logo, $T(n)$ será $O(n^4)$

Melhor caso $\rightsquigarrow n = 1 \rightsquigarrow \Omega(2) = \Omega(c)$

19/05/20

5) $G = (V, E)$

↳ Fortemente conectado

↳ Direcionado

↳ Com pesos ⇝ $\{1, 2, 3, 4\}$

↳ $O(|E| + |V|)$

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

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

$$\left.\begin{array}{l} \text{Initialize} \rightsquigarrow 1 \\ \text{ExtractMin} \rightsquigarrow n \\ \text{ChangeKey} \rightsquigarrow m \end{array}\right\} O(n+m)$$

BFS ⇝ caminho mínimo ⇝ $O(m+n)$

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 de peso 4 por 3 novos vértices. Em seguida, executo a BFS para determinar o caminho mínimo

$$a \xrightarrow{1} b$$

$$u \xrightarrow{2} v \rightarrow u \xrightarrow{1} v' \xrightarrow{1} v$$

$$w \xrightarrow{3} z \rightarrow w \xrightarrow{1} z' \xrightarrow{1} z'' \xrightarrow{1} z$$

$$d \xrightarrow{4} e \rightarrow d \xrightarrow{1} d' \xrightarrow{1} d'' \xrightarrow{1} d''' \xrightarrow{1} d$$