Análise de Algoritmos - 2020.1

Exame 1 - Gabarito

1 - a) Para essa questão, modificaremos o algoritmo da BFS(G, v) visto em aula para retornar o número de vértices alcançados por v mais 1 (incluindo v).

```
for each v ∈ V do
  if BFS(G, v) == |V| then
    return SIM
  end if
end for
return NAO
```

Complexidade final: $O(|V|(|V| + |E|))$.

1 - b) Note que uma DAG sempre possuirá pelo menos um vértice cujo grau de entrada $(d(\cdot)^-)$ é zero. Logo, para resolver essa questão, basta verificar se o primeiro vértice de grau de entrada 0 alcança todos os outros. Caso contrário, a resposta é NAO (ou o grafo é desconectado ou existe mais de um vértice com grau de entrada 0).

```
for each v ∈ V do
  if d(v)^- == 0 then
    if BFS(G, v) == |V| then
      return SIM
    end if
    return NAO
  end if
end for
```

Complexidade final: $O(|V|+|E|)$ (note que a BFS é executada somente uma vez).

2 - a) FALSO. A complexidade $\Omega(n^{1.5})$ se aplica às entradas de pior caso. Para qualquer n, pode existir entradas de "melhor caso" onde a complexidade para tais entradas podem ser, por exemplo, $O(n)$ ou mesmo $O(1)$.
Contra-exemplo: considere um algoritmo de ordenação $O(n \log n)$ que tenha uma etapa em tempo $O(n)$ que verifica se a entrada é ordenada. Logo, esse algoritmo não precisa ordenar a entrada.

2 - b) FALSO. Podemos também levar em consideração os argumentos da letra a). Além disso, quando lidamos com complexidade assintótica, estamos nos referindo a valores muito grandes de n. Para valores pequenos de n, um algoritmo $O(n^3)$ pode ser facilmente muito mais rápido que um algoritmo $O(n)$ devido as constantes.

2 - c) FALSO. Um algoritmo A com complexidade $\Theta(n)$ também pertence a $O(n^2 \log n)$. Da mesma forma, um algoritmo B com complexidade $\Theta(1)$ também pertence a $O(n^2\sqrt{n})$. É possível que B seja $\Theta(1)$ para qualquer entrada, e ainda assim pertença a $O(n^2\sqrt{n})$. Nesse caso, B nunca iria realizar mais operações que A.

3 - Usaremos o algoritmo $kSelect(\cdot)$ visto em aula.

```
alpha ← kSelect(A, α)
beta ← kSelect(A, β)
soma ← 0
for i = 1 ... n do
  if alpha ≤ a_i ≤ beta then
    soma ← soma + a_i
  end if
end for
return soma
```

Complexidade final: Tanto $kSelect(\cdot)$ quanto o loop pertencem a $O(n)$. Logo, esse algoritmo pertence a $O(n)$.

4 - Importante: apresentar corretamente o $O(\cdot)$ é suficiente para valer os dois pontos.

Seja $T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (j - i + 1) + T(n-1)$

$T(n) \leq n \times n \times n + T(n-1) = n^3 + T(n-1)$

$T(n) \leq \sum_{k=1}^{n} k^3 \leq \sum_{k=1}^{n} n^3 \therefore T(n) \in O(n^4)$.

Seja $T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (j - i + 1) + T(n-1)$

$T(n) \geq \sum_{i=1}^{\frac{n}{3}} \sum_{j=\frac{2n}{3}+1}^{n} (\frac{n}{3}) + T(n-1) \geq \frac{n^3}{27} + T(n-1)$

$T(n) \geq \sum_{k=1}^{n} (\frac{k^3}{27}) \geq \sum_{k=\frac{n}{2}+1}^{n} (\frac{k^3}{27}) \geq \sum_{k=\frac{n}{2}+1}^{n} (\frac{(\frac{n}{2})^3}{27}) = (\frac{n}{2})^4 \times \frac{1}{27} \therefore T(n) \in \Omega(n^4)$

Por fim, $O(n^4)$ e $\Omega(n^4)$ implicam em $\Theta(n^4)$.

5 - Seja w_e a notação para peso de uma aresta $e \in E$. Aplica-se uma transfor-

mação em G de modo a transformar todas arestas cujos pesos são maiores que um em sequências "vértice-aresta" de tamanho $w_e - 1$.
Seja G' o grafo G transformado, executa-se $\text{BFS}(G', u)$ com critério de parada ao alcançar v e uma variável auxiliar que armazenará o número de arestas necessárias para alcançar v. Ao encerrar a $\text{BFS}(\cdot)$, retorna-se a variável auxiliar. A complexidade final é $O(|V| + |E|)$.