Lista 3

Pedro Daniel da Silva Gohl

August 2016

1 Questão

Algoritmo 1 busca em Profundidade

```
função buscaEmProfundidade(vértice V, j)                    ▷ O(a^2)
    visitados[i] = 1
    se i = j então
        retorna j
    fim se
    para todo w adjacente a i faça
        se visitados[w] = 0 então
            função buscaEmProfundidade(w,j)
            fim função
        senão retorna j
        fim se
    fim para
fim função
```

$$T(n) = \begin{cases} 1, & \text{se } n = 0 \\ aT(n-1), & \text{se } n > 1, a \text{ é a quantidade de adjacentes} \end{cases}$$

k=1

$$T(n) = aT(n-1)$$

$$T(n-1) = aT(n-2)$$

k=2

$$T(n) = a * aT(n-2)$$

$$T(n) = a^2T(n-2)$$

$$T(n-2) = aT(n-3)$$

k=3

$$T(n) = a^2 * aT(n-3)$$
$$T(n) = a^3 T(n-3)$$

F.R

$$a^k T(n-k)$$

pior caso é quando ele percorre todos os vértices e arestas k=n

$$a^n T(0)$$

$$a^n * 1$$

$$O(a^n)$$

2 Questão

2.1 A

$$\sum_{i=2}^{n} C$$

$$c = 2, n + c - 2$$
$$O(n)$$

2.2 B

$$T(n) = \begin{cases} 1, & \text{se } n = 0 \\ 2T(n-1)+1, & \text{se } n > 1 \end{cases}$$

k=1

$$T(n) = 2T(n-1) + 1$$
$$T(n-1) = 2T(n-2) + 1$$

k=2

$$T(n) = 2(2T(n-2) + 1) + 1$$
$$T(n) = 2^2 T(n-2) + 2 + 1$$
$$T(n) = 2T(n-3) + 1$$

k=3

$$T(n) = 2^2(2T(n-3) + 1) + 2 + 1$$
$$T(n) = 2^3 T(n-3) + 2^2 + 2 + 1$$
$$T(n) = 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i$$
$$n = k$$
$$2^{n+1} - 1$$
$$O(2^n)$$

3 Questão

3.1 A

É um algoritmo procedural que toma suas decisões pela escolha que mais trará beneficíos. Esses algoritmos não analisam um problema como um todo, apenas o próximo passo para tentar informar a saída o mais rápido possível. Uma vez que ele escolhe o próximo passo, ele não volta atrás.

Exemplo: Algoritmo de *Dijkstra* com complexidade $O(|E| + |V|log|V|)$

3.2 B

Backtracking é um conceito dos algoritmos de busca que permite refazer uma escolha passada para permitir novas possibilidades de escolhas futuras

Exemplos: Algoritmo de custo uniforme com complexidade $O(V)$

4 Questão

4.1 A

Mostre que X está na classe NP. *verificar se a solução é válida em tempo polinomial*

Mostre que X está na classe NP-Difícil *apresentando uma redução polinomial qye transforme instâncias de um problema Y (conhecidamente NP-Difícil) no problema X.*

Exemplo: Redução do SAT pro CLIQUE

$$a = (X \vee Y \vee Z)(Y \vee Z)(\neg X \vee Y \vee Z)$$

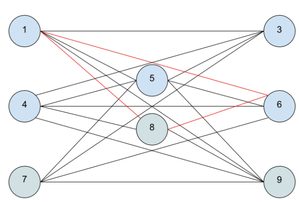

Figura 1: SAT pra CLIQUE

4.2 B

P: Solucionável em tempo polinomial Exemplo: Quicksort

NP: Existe solução não polinomial conhecida; Toda solução pode ser verificada em tempo polinomial. Exemplo: O problema do caixeiro viajante

NP-Difícil: Problemas que não existem soluçõe, indedutiveis e intrataveis. Exemplo: o Problema da parada

$NP-Completo$: Problemas os quais existem soluções conhecidas, entretanto nunca foi provada existência de algoritmos polinomiais, mas também nunca provaram a existência. Exemplo: Problema 3SAT

5 Questão

Um grafo G é um par ordenado $G = (V, E)$ onde E é um conjunto de pares de elementos V, onde V são chamados vértices e os elementos E são chamados de arestas.

Problema do caixeiro viajante: o problema do caixeiro viajante consiste na procura de um caminho que possua a menor distância começando em uma cidade qualquer, visitando cada uma delas apenas uma única vez retornando no final à cidade inicial.

Em um grafo G partindo de um vértice V qualquer o problema do caixeiro viajante retorna um caminho P passando por todos os vértices de G apenas uma vez.

Coloração de grafos: uma coloração de uma grafo G é uma função $F : V \to C$ onde C é um conjunto de cores, tal que cada aresta (V, U) de E, tem-se $f(v) \neq f(u)$.

Clique: seja um grafo G, um clique C é um subgrafo completo de G onde para cada vértice de C existem arestas para todos os demais vértices de C.