Gabarito da Prova 1

`MAC338 - 25/4/2000`

Resposta da questão **1.**

(i)

Sim, pois

$$|n \log n| = n \log n \leq n^2 \qquad \text{para todo } n \geq 1.$$

(ii)

Não, pois a notação O fornece apenas uma limitação superior para a complexidade de tempo do algoritmo. No caso, o algoritmo B poderia ter complexidade de tempo $O(n)$ enquanto o algoritmo A não.

(iii)

Como $n(\log n)^3 = O(n^2)$ mas $n^2 \neq O(n(\log n)^3)$, o mais informativo é dizer que a complexidade de tempo do algoritmo é $O(n(\log n)^3)$, pois isto fornece uma estimativa melhor do tempo que o algoritmo consome.

Resposta da questão **2.**

(i)

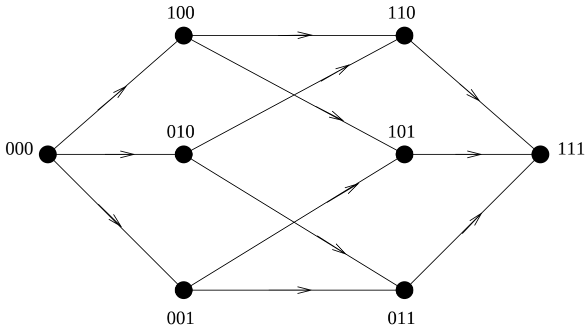

Figura 1: Grafo G_1 para $k = 2$ e $n = 3$.

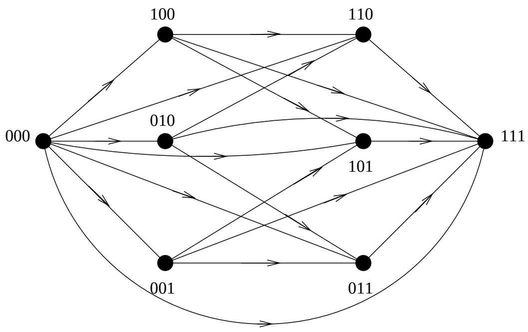

Figura 2: Grafo G_2 para $k = 2$ e $n = 3$.

(ii)

Note que $|V(G_1)| = |V(G_2)| = k^n$. Para cada $v \in V(G_1)$, denote por $d(v)$ o número de arcos que saem de v. Claramente, $d(v) \leq n$ para todo $v \in V(G_1)$. Então

$$|A(G_1)| = \sum_{v \in V(G_1)} d(v) \leq nk^n.$$

Assim, G_1 tem no máximo nk^n arcos.

Vamos agora contar o número de arcos de G_2. Seja S o conjunto dos pares $(\mathbf{x}, \mathbf{y})$ tais que $\mathbf{x} = (x_1, \ldots, x_n), \mathbf{y} = y_1, \ldots, y_n$ com $x_i, y_i \in [k]$ e $x_i \leq y_i$ para todo i. Seja T o conjunto dos vetores $(p_1, \ldots, p_n)$ tais que $p_i = (x_i, y_i)$ com $x_i, y_i \in [k]$ e $x_i \leq y_i$ para todo i. É fácil ver que $|S| = |T|$. Além disso, cada entrada p_i de um vetor em T pode assumir $k + \binom{k}{2}$ possíveis valores. Logo,

$$|T| = \left\{ \binom{k}{1} + \binom{k}{2} \right\}^n = \binom{k+1}{2}^n.$$

Por outro lado, o conjunto dos arcos de G_2 consiste dos elementos $(\mathbf{x}, \mathbf{y})$ de S tais que $\mathbf{x} \neq \mathbf{y}$. Assim,

$$|A(G_2)| = |S| - k^n = \binom{k+1}{2}^n - k^n.$$

(iii)

É suficiente mostrar que $(3/2)k^n \leq \binom{k+1}{2}^n$. Isto vale se

$$\log \frac{3}{2} k^n \leq \log \binom{k+1}{2}^n,$$

ou seja,

$$\log \frac{3}{2} + n \log k \leq n \log \frac{k+1}{2} + n \log k.$$

Como $\log 3/2 \leq \log(k+1)/2$ para $k \geq 2$ e $n \geq 1$, segue o resultado.

(iv)

Como G_1 é um subgrafo de G_2, toda ordenação topológica de G_2 também é uma ordenação topológica de G_1.

Suponha que $v_1, v_2, \ldots, v_{k^n}$ seja uma ordenação topológica de G_1. Basta verificar agora que se $i > j$ então não existe um arco de v_i para v_j em G_2. Tome $v_j = (y_1, \ldots, y_n)$ e $v_i = (x_1, \ldots, x_n)$, e suponha por absurdo que exista um arco de v_i para v_j em G_2, ou seja, que $x_l \leq y_l$ para todo l. Como $v_i \neq v_j$, deve existir um caminho orientado de v_i a v_j em G_1 (ele pode ser obtido incrementando uma a uma as entradas do vetor v_i até "atingir" o vetor v_j). Isto contraria o fato de que $v_1, v_2, \ldots, v_{k^n}$ é uma ordenação topológica de G_1. Logo, não existe arco de v_i para v_j em G_2, e $v_1, v_2, \ldots, v_{k^n}$ também é uma ordenação topológica de G_2.

(v)

Topological-Sort(G_1) consome no máximo $k^n + nk^n$ unidades de tempo enquanto Topological-Sort(G_2) consome $\binom{k+1}{2}^n$ unidades de tempo. Quando $k = 2$ e n é grande, Topological-Sort(G_2) demora 3^n unidades de tempo, enquanto Topological-Sort(G_1) demora $(n+1)2^n$. Portanto, neste caso Topological-Sort(G_2) demora mais que Topological-Sort(G_1).

Para $n = 15$, Topological-Sort(G_2) demora 10 vezes mais que Topological-Sort(G_1), e para $n = 27$, Topological-Sort(G_2) demora 1000 vezes mais que Topological-Sort(G_1). (é necessário fazer umas continhas ...). Note que não foram pedidos os menores valores de n que satisfazem essas condições (para calcular isso seria preciso uma calculadora), mas era esperado que procurassem por valores pequenos.

Resposta da questão **3.**

Grafo orientado			
	Branca	*Cinza*	*Preta*
Branca	Sim	Sim	Sim
Cinza	Sim	Sim	Sim
Preta	Não	Sim	Sim

Na execução de DFS(G) para um *grafo orientado* G, não pode haver aparecer um arco (x, y) com x de cor preta e y de cor branca; um vértice só recebe cor preta quando todos os seus vizinhos já foram visitados e/ou pesquisados (isto é, têm cor cinza ou preta).

Grafo não-orientado			
	Branca	*Cinza*	*Preta*
Branca	Sim	Sim	Não
Cinza	Sim	Sim	Sim
Preta	Não	Sim	Sim

Na execução de DFS(G) para um *grafo não-orientado*, não pode aparecer um arco com uma das pontas de cor preta e a outra de cor branca; o motivo é o mesmo do caso orientado.

Resposta da questão **4.**

Suponha que executamos DFS(G) para um grafo orientado G sem laços, e obtivemos os valores $d[v]$, $f[v]$ para cada vértice v de G. Sabemos pela Estrutura de Parênteses (ER) que para quaiquer dois vértices u e v de G ocorre exatamente uma da seguintes possibilidades (veja CLR):

- $d[u] < d[v] < f[v] < f[u]$ ou $d[v] < d[u] < f[u] < f[v]$,
- $d[v] < f[v] < d[u] < f[u]$,
- $d[u] < f[u] < d[v] < f[v]$.

(i)

Se (u, v) é um arco da floresta de busca ou um arco longitudinal descendente então v é descendente de u na floresta de busca. Isso significa que v foi descoberto depois de u e antes de u ser finalizado. Portanto, pela ER temos que $d[u] < d[v] < f[v] < f[u]$.

Por outro lado, se $d[u] < d[v] < f[v] < f[u]$ então v foi descoberto enquanto u ainda estava sendo pesquisado (tinha cor cinza). Então v é um descendente de u na floresta de busca, e portanto, (u, v) é um arco da floresta de busca ou um arco longitudinal descendente.

(ii)

Se (u, v) é um arco longitudinal ascendente então u é um descendente de v na floresta de busca. Portanto, por um argumento análogo ao apresentado no item anterior, temos que $d[v] < d[u] < f[u] < f[v]$.

Por outro lado, se $d[v] < d[u] < f[u] < f[v]$ então u é um descendente de v na floresta de busca, e portanto, (u, v) é um arco longitudinal ascendente.

(iii)

Se (u, v) é um arco transversal então ele não pertence à floresta de busca e não é um arco longitudinal (descendente ou ascendente). Pela ER e pelos itens anteriores, segue que os intervalos $[d[v], f[v]]$ e $[d[u], f[u]]$ são disjuntos. Além disso, temos que $d[v] < d[u]$, senão v teria que ser um descendente de u. Logo, $d[v] < f[v] < d[u] < f[u]$.

Por outro lado, se $d[v] < f[v] < d[u] < f[u]$ então pelos itens anteriores, temos que (u, v) não pode ser um arco da floresta nem um arco longitudinal (descendente ou ascendente). Portanto, (u, v) é um arco transversal.

(iv)

A paridade de $f[v] - d[v]$ é ímpar para todo vértice de G. Basta observar que a variável `time` no algoritmo DFS é incrementada um número ímpar de vezes a partir do instante em que um vértice v é descoberto até ele ser finalizado. De fato, `time` é incrementada imediatamente antes de $d[v]$ ser atualizado. Quando DFS vai colorir v de preto e atualizar $f[v]$, possivelmente vários vértices foram descobertos e finalizados desde o instante em que v foi descoberto. Para cada um desses são contadas duas modificações de `time`. Note que `time` não é modificada quando são encontrados vértices de cor cinza ou preta. Assim, o número de incrementos de `time` ocorridos é ímpar. Logo, $f[v] - d[v]$ é ímpar.

(v)

Não podemos ter $d[v] < d[u] < f[v] < f[u]$ para nenhum par u, v de vértices, por causa da ER.