Projeto e análise de algoritmos II

Laboratório 1 - Blueprints

Erik Perillo, RA135582

18 de setembro de 2016

1 Abordagem

Dados grafo B (do arquivo morto) e A (nome borrado), Duas condições devem ser satisfeitas para serem da mesma cidade:

1. Todos os vértices de B devem estar em A. Basta fazer uma busca exaustiva como descrita no pseudo-código abaixo:

Algorithm 1

```
1: procedure CONTEM(A, B)
2:     {V_A, E_A} ← A, {V_B, E_B} ← B
3:     if |V_B| > |V_A| then return false
4:     for v in V_B do
5:         if v ∉ V_A then return false
       return true
```

2. Para cada vértice (u, v) em B, existe um passeio em A $P = \{u, w_1, \ldots, w_n, v\}$ tal que $\{w_1, \ldots, w_n\}$ não estão em B. A ideia então é fazer uma busca em profundidade [1] para toda aresta $(u, v) \in B$ de u até o possível momento em que o vértice v seja alcançado, sempre checando se os vértices na ''jornada" não estão em B. O pseudo-código a seguir descreve.

Algorithm 2

```
1: procedure NOVOS-VERTICES-EM-VELHAS-CONEXOES(A, B)
2:     {V_A, E_A} ← A, {V_B, E_B} ← B
3:     for (u, v) in E_B do
4:         visited ← ∅ ∪ u
5:         for w in Adj_A[u] do
6:             if not visited[w] and Novos-vertices-em-caminho(w, v, B, visited) then return true
       return false
```

Algorithm 3

```
1: procedure NOVOS-VERTICES-EM-CAMINHO(u, v, B, visited)
2:     visited[u] ← true
3:     if u = v then return true
4:     if u ∈ B then return false
5:     for w in Adj_A[u] do
6:         if not visited[w] and Novos-vertices-em-caminho(w, v, B, visited) then return true
       return false
```

2 Complexidade

Vamos definir n_1, m_1 o número de vértices e arestas de B, respectivamente, e n_2, m_2 os mesmos de A.

2.1 Algoritmo 1

O primeiro algoritmo na linha 4 executa $O(n_1)$ vezes a linha 5, que por sua vez pode ser implementada em $O(n_2)$. Nas outras linhas, tem tempo constante. Assim, sua complexidade assintótica é $O(n_2 n_1)$. Como todo vértice de B está em A, $n_2 \geq n_1$ e temos o resultado final de $O(n_2^2)$.

2.2 Algoritmos 2 e 3

O *loop* da linha 3 do algoritmo 2 é executado no máximo m_1 vezes. O algoritmo 3 é executado no máximo uma vez para cada vértice de A (pois depois o vértice é marcado como visitado), passando por suas adjacências. Assim, o tempo total passado chamando o Algoritmo 3 é $\sum_{v \in A} d(v) = O(m_2)$ para cada iteração da linha 3 do algoritmo 2. A checagem da linha 4 do algoritmo 3 pode ser guardada, sendo executada no máximo n_2 vezes, levando n_1 quando acontece, tendo como tempo $O(n_2 n_1)$. O total então é $O(m_1 m_2 + n_1 n_2)$. Agora, sabe-se que há limites entre o número de nós e arestas: $m \leq \frac{n^2 - n}{2}$, m e n o número de arestas e vértices de um grafo simples. Assim, $m_1 m_2 + n_1 n_2 \leq m_1 n_2^2 + 2n_2^2, (n \geq 0)$ E o tempo é então $O(m_1 n_2^2)$.

2.3 Total

O tempo total é a soma dos tempos, que dá $O(n_2^2 + m_1 n_2^2) = O(m_1 n_2^2)$.

Referências

[1] Cormen et al. *Introduction to Algorithms, 3th ed.* The MIT Press, 2009, pp. 603–607.