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 é checar, para toda aresta (u, v) em B, se há um passeio que satisfaz tais condições. `DFS-VISIT` [1] em A a partir de u pode ser usado para produzir uma lista π de pais. Checa-se então essa lista, a partir de v, indo até u (se existe passeio), verificando se os vértices do meio não estão em B.

Algorithm 2

```
1: procedure NOVOS-VERTICES-EM-VELHAS-CONEXOES(A, B)
2:     {V_A, E_A} ← A, {V_B, E_B} ← B
3:     checked_A ← ∅
4:     for (u, v) in E_B do
5:         π = DFS-VISIT(A, u)
6:         w = π[v]
7:         if w = NULL then return false
8:         while w ≠ u do
9:             if not(w ∈ checked_A) and w ∈ B then return false
10:            checked_A ← checked_A ∪ w
11:            w = π[w]
       return true
```

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 Algoritmo 2

`DFS-VISIT` na linha 5 leva $O(m_2)$. O loop da linha 8 é executado em $O(m_2)$ também, todos dentro do *loop* da linha 4 que acontece em $O(m_1)$. A checagem da linha 9, salvando-se os resultados prévios em $checked_A$, é executada em no máximo $O(n_2)$ por todo o algoritmo e, quando acontece, leva $O(n_1)$. As outras linhas são em $O(1)$. O tempo total de execução é então $O(m_1(2m_2) + n_1 n_2)$. Agora, sabe-se que há limites entre o número de nós e arestas: $n \leq 2m$, sendo m e n o número de arestas e vértices de um grafo simples qualquer. Assim, $m_1(2m_2) + n_1 n_2 \leq m_1 n_2 + n_2^2$ e então o tempo é $O(m_1 n_2 + n_2^2)$.

2.3 Total

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

Referências

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