Projeto e Análise de Algoritmos

Flávio L. C. de Moura*

31 de março de 2022

1 Componentes fortemente conexas

Nesta seção veremos como determinar as componentes fortemente conexas de um digrafo. É fácil ver que a estratégia que utilizamos para determinar as componentes conexas de um grafo não funciona para digrafos. De fato, ao executarmos DFS a partir de um vértice u, percorremos todos os vértices alcançáveis a partir de u, e terminamos com uma árvore que tem u como raiz e cujos vértices não estão fortemente conectados. De fato, uma componente fortemente conexa de um digrafo $G = (V, E)$ é um conjunto maximal de vértices $C \subseteq V$ tal que para qualquer par de vértices u e v em C, temos que $u \rightsquigarrow v$ e $v \rightsquigarrow u$, ou seja, u e v são mutuamente alcançáveis. O algoritmo que apresentaremos a seguir utiliza o digrafo G^T, chamado de *transposto de* G, definido por $G^T = (V, E^T)$, onde $E^T = \{(u, v) : (v, u) \in E\}$.

Exercício 1.1. *Construa um algoritmo eficiente para computar o digrafo transposto G^T de G a partir da lista de adjacências de G, e em seguida faça a análise assintótica do algoritmo.*

Exercício 1.2. *Construa um algoritmo eficiente para computar o digrafo transposto G^T de G a partir da matriz de adjacências de G, e em seguida faça a análise assintótica do algoritmo.*

A ideia do algoritmo que veremos a seguir está baseada no fato de que, para qualquer digrafo G, tanto G quanto G^T possuem as mesmas componentes fortemente conexas. De fato, u e v são mutualmente alcançáveis em G, se e somente se, u e v são mutualmente alcançáveis em G^T.

Algorithm 1: SCC(G)

1 Execute DFS(G) para computar o tempo de finalização $u.f$ para cada vértice u;
2 Compute G^T;
3 Execute DFS(G^T) escolhendo os vértices em ordem decrescente do tempo de finalização;
4 Retorne os vértices de cada árvore da floresta computada na linha anterior como uma componente fortemente conexa separada.

Cada conjunto de vértices retornado pelo algoritmo SCC constitui um vértice do grafo $G^{scc} = (V^{scc}, E^{scc})$, chamado de *grafo das componentes fortes* de G, cujos vértices correspondem às componentes fortemente conexas de G, digamos $C_1, C_2, \ldots, C_k$ e, $(C_i, C_j) \in E^{scc}$ se $i \neq j$ e existe alguma aresta de G que sai de um vértice de C_i e chega em um vértice de C_j. O grafo das componentes conexas tem como propriedade fundamental não possuir ciclos. Um digrafo sem ciclo é normalmente chamado de DAG (*directed acyclic graph*).

A seguir, estenderemos a noção de tempo de início e finalização de um vértice para conjuntos de vértices. Esta extensão será necessária para provarmos a correção do algoritmo SCC.

Definição 1.3. *Se $U \subseteq V$ então definimos $d(U) = \min_{u \in U}\{u.d\}$ e $f(U) = \max_{u \in U}\{u.f\}$*

Ou seja, $d(U)$ representa o tempo do primeiro vértice que foi visitado do conjunto U, enquanto que $f(U)$ denota o tempo do último vértice de U a ter a visita finalizada.

Lema 1.4. *Sejam C e C' duas componentes fortemente conexas de um digrafo $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E$ tal que $u \in C$ e $v \in C'$. Então $f(C) > f(C')$.*

*flavio@flaviomoura.info

Corolário 1.5. *Sejam C e C' duas componentes fortemente conexas de um digrafo $G = (V, E)$. Suponha que exista uma aresta $(u, v) \in E^T$ tal que $u \in C$ e $v \in C'$. Então $f(C) < f(C')$.*

Teorema 1.6. *O algoritmo SCC computa corretamente as componentes fortemente conexas de um digrafo G dado como argumento.*