MO417 – Complexidade de Algoritmos
Segundo Semestre de 2008
Sétima Lista de Exercícios

1 Representação de Grafos

1. Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?
2. The transpose of a directed graph G = (V, E) is the graph $G^T = (V, E^T)$, where E^T = (v, u) $\in$ V x V : (u, v) $\in$ E. Thus, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.
3. The square of a directed graph G = (V, E) is the graph $G^2 = (V, E^2)$ such that (u, w) $\in E^2$ if and only if for some v $\in$ V, both (u, v) $\in$ E and (v, w) $\in$ E. That is, G^2 contains an edge between u and w whenever G contains a path with exactly two edges between u and w. Describe efficient algorithms for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.
4. When an adjacency-matrix representation is used, most graph algorithms require time $\Omega(V^2)$, but there are some exceptions. Show that determining whether a directed graph G contains a universal sink (a vertex with in-degree $|V| - 1$ and out-degree 0) can be determined in time $\mathrm{O}(V)$, given an adjacency matrix for G.

2 Buscas em Grafos

5. Give a counterexample to the conjecture that if there is a path from u to v in a directed graph G, and if d[u] $<$ d[v] in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.
6. Show that a depth-first search of an undirected graph G can be used to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that each vertex v is assigned an integer label cc[v] between 1 and k, where k is the number of connected components of G, such that cc[u] = cc[v] if and only if u and v are in the same connected component.
7. Give an algorithm that determines whether or not a given undirected graph G = (V, E) contains a cycle. Your algorithm should run in $\mathrm{O}(V)$ time, independent of $|E|$.
8. Another way to perform topological sorting on a directed acyclic graph G = (V, E) is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $\mathrm{O}(V + E)$. What happens to this algorithm if G has cycles?