CS21003 - Tutorial 12

November 17, 2017

1. Given a directed graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, n\}$, we define the *transitive closure* of Gas the graph $G^* = (V, E^*)$, where

$$E^* = \{(i, j) : \text{there is a path from vertex } i \text{ to vertex } j \text{ in } G\}$$

   Can you propose an $O(n^3)$ algorithm to compute the transitive closure of the graph?

2. Let D be the shortest path matrix of an undirected weighted graph G. Thus $D(u, v)$ is the length of the shortest path from vertex u to vertex v, for every two vertices u and v. Graph G and matrix D are given. Assume the weight of an edge $e = (a, b)$ is decreased from w_e to w'_e. Design an algorithm to update matrix D with respect to this change. The running time of your algorithm should be $O(n^2)$. Describe all details and write a pseudo-code for your algorithm.

3. Let $G = (V, E)$ be a weighted graph and T be the array containing the parents (previous vertices) corresponding to the shortest distance from source s to all the other vertices. Assume all weights in G are increased by the same amount c. Is T still the parent array (from source s) of the modified graph? If yes, prove the statement. Otherwise, give a counter example.

4. Let $G = (V, E)$ be a weighted undirected graph. Let $s, t \in V$ and $s \neq t$. Design an $O(ElogV)$ algorithm to find all vertices v such that v lies on at least one of the shortest paths between s and t.

5. A *maximum spanning tree* of G is a spanning tree T of G such that the sum of costs of the edges in T is as large as possible. Modify Kruskal's algorithm to compute a maximum spanning tree of G. What is the running time of your algorithm?

6. Let $G = (V, E)$ be a directed graph with n vertices. A *Hamiltonian path* in G is a path in G of length $n - 1$ (a path on which all vertices of G appear). Given that G is a DAG, propose an $O(|V| + |E|)$-time algorithm to determine whether G contains a Hamiltonian path. Also prove the correctness of your algorithm.

7. Let $G = (V, E)$ be a directed graph. A vertex s in G is called a *source* if its in-degree is zero. Likewise, a vertex t is called a target (or sink) if its out-degree is zero.

   - Propose an $O(|V| + |E|)$-time algorithm to locate all the sources and all the targets in G.
   - Prove that a DAG must contain at least one source and at least one target.
   - Propose an $O(|V| + |E|)$-time algorithm to count the total number of paths from all the sources in G to all the targets in G.