CS21003 - Tutorial 8

October 10th, 2018

1. You are given an undirected graph where each edge has cost 1. A path p between u and v is a shortest path if the sum of cost of the edges on p is minimum among all the paths between u and v. Given two vertices u and v of the graph, you need to find the cost of shortest path between u and v. How can this be done? Can this idea be generalised if all the edges have same positive cost c?

2. There are N variables $x_1, x_2, \ldots, x_N$ and M relations of the form $x_i < x_j$ where $i \neq j$. A subset S of relations is called inconsistent if there does not exist any assignment of variables that satisfies all the relations in S. e.g, $\{x_1 < x_2, x_2 < x_1\}$ is inconsistent. You need to find if there is an inconsistent subset of M.

3. When an adjacency-matrix representation is used, most graph algorithms require time $\Omega(n^2)$, but there are some exceptions. Show that determining whether a directed graph G contains a universal sink (a vertex with in-degree $n-1$ and out-degree 0) can be determined in time $O(n)$, given an adjacency matrix for G.

4. You are given an undirected acyclic graph. You need to find a pair of vertices (i, j) such that the length of the path between i and j is maximum among all such pairs. The length of a path is the number of edges on the path. [**Hint:** There is a trivial $O(n.(n+e))$ algorithm to solve this. However, can you give an $O(n+e)$ algorithm?]

5. Prove or disprove: BFS and DFS algorithms on a undirected, connected graph $G = (V, E)$, produce the same tree if and only if G is a tree.

6. Given a directed acyclic graph G, design an $O(n+e)$ time algorithm which finds the length of the longest path of the graph. The sketch of the algorithm is as follows. Complete this.

   (a) Find a topological sort of the given DAG and let $v_1, v_2, \ldots, v_n$ be a topological sort, i.e., each edge is from a vertex v_i to another vertex v_j with $j > i$. Let $A[i]$ be the longest path of the graph starting at v_i. Find a formula for computing $A[i]$.

   (b) If we compute $A[1], A[2], \ldots, A[n]$, what would be the final solution?

   (c) Write a dynamic program for filling array A. What is the running time of this algorithm?

7. Suppose we have an undirected graph and we want to color all the vertices with two colors red and blue such that no two neighbors have the same color. Design an $O(n+e)$ time algorithm which finds such a coloring if possible or determines that there is no such coloring. (a) Prove that if the graph has a cycle of odd length, there is no such coloring. (b) Use BFS to find such a coloring. Show that if the graph does not have an odd length cycle, BFS gives an appropriate coloring.