CS21003: Algorithms-I (Theory)
Tutorial – 8 (Graph Algorithms)
Date: 31-March-2020

1. Prof. Kevin devises an alternative technique to compute the minimum spanning tree (MST) in a connected undirected graph $G = (V, E)$ with a cost $c(e)$ associated with each edge $e \in E$. Your task is to assist Prof. Kevin by judging the efficiency and correctness of his idea. Assume that G has n ($n = |V|$) nodes and m ($m = |E|$) edges.

   (i) Firstly, given an edge $e \in E$, Prof. Kevin provides a solution (see below) to determine whether or not e belongs to a cycle in G. His idea is based on the fact that e lies on a cycle if and only if $(G - e)$ is connected.

   (a) Delete the edge e from the given graph $G = (V, E)$. Call this graph G'.

   (b) Run breadth-first or depth-first traversal in the graph G' obtained in Step-(a).

   (c) If the traversal indicates that G' is disconnected, return false; else return true.

   Determine the running time of this proposed procedure (calculate the time-complexity of each step and then the overall time-complexity).

   (ii) The MST algorithm of Prof. Kevin goes as follows.

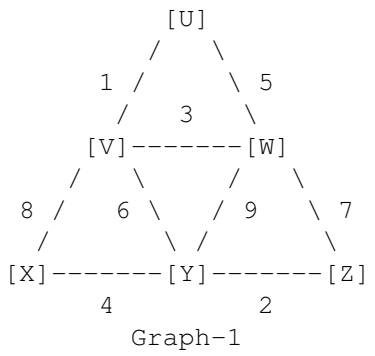
   (a) Sort E in non-increasing order of the edge costs. Store this sorted list in T.

   (b) Repeat the following three steps, as long as G contains more than $(n - 1)$ edges:

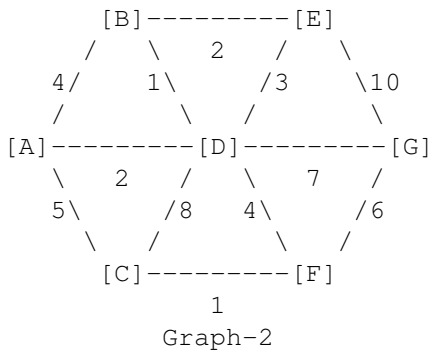
   - Pick the edge e from T with largest cost.
   - If e belongs to a cycle in G, remove e from E.
   - Remove e from T.

   (c) Output the reduced graph (V, E).

   Demonstrate stepwise (edge-by-edge) how the approach proposed by Prof. Kevin works on the two example graphs given in the following.



Graph-1



Graph-2

   Describe the running time of this procedure as proposed by Prof. Kevin (calculate the time-complexity of each step and then the overall time-complexity).

   (iii) Now, the more serious concern is to check whether Prof. Kevin's approach produces correct MST. Prove (with proper reasoning) or disprove (with a valid counter-example): Prof. Kevin's approach always outputs a minimum spanning tree of a connected graph.

2. In the game of Snakes-and-Ladders, we are given a $n \times n$ board where each cell is marked with a number from 1 (start position) to n^2 (destination position) and there are few ladders and snakes placed arbitrarily in the board. From a particular cell c_1, a ladder can make the player to climb up (jump) to a particular cell c_2 (obviously, $c_2 > c_1$), whereas a snake can make the player to climb down (slip) to a particular cell, c_2 (obviously, $c_2 < c_1$). You are also given a dice marked from 1 to 6 which you throw and make your moves in the board based on the (random) number you get from the dice. Given a particular Snakes-and-Ladders board to you, your task is to find out the minimum number of dice throws required to reach to the destination position in the board from its start position, given that you always get best/favourable dice outcomes as desired.

   (i) Formulate the above problem into a graph-theoretic problem.

   (ii) Propose an algorithm to derive the minimum number of throws required.

   (iii) Derive the time complexity of your algorithm.