Assignment Sheet

Competitive Programming Workshop – Even 2017

Course Coordinator: Manish K Thakur

Q1. Set of problems based on Minimum Spanning Tree (MST):

For a given weighted graph G = {V, E}, we already computed the MST represented as T. WAP to perform following:

(a) Let us consider that weight of an edge in the given Graph, G is **decreased**. Call this graph as G'. Now, there can be two possibilities, either T is still the MST or T may not be the MST. It is now needed to identify the MST (either T or T', where T' is the new MST) without starting from the scratch.

(b) Let us consider that weight of an edge in the given Graph, G is **increased**. Call this graph as G'. Now, there can be two possibilities, either T is still the MST or T may not be the MST. It is now needed to identify the MST (either T or T', where T' is the new MST) without starting from the scratch.

(c) Let us consider that an edge in the given Graph, G is **removed / deleted**. Call this graph as G'. Now, there can be two possibilities, either T is still the MST or T may not be the MST. It is now needed to identify the MST (either T or T', where T' is the new MST) without starting from the scratch.

(d) Let us consider that a new edge (with some weigh) in the given Graph, G is **inserted** between two vertices. Call this graph as G'. Now, there can be two possibilities, either T is still the MST or T may not be the MST. It is now needed to identify the MST (either T or T', where T' is the new MST) without starting from the scratch.

(e) After analysing the entire cost to construct the MST, T, it has been found that spending such cost is not feasible, hence decided to identify such nodes which are responsible for high costing in T. Let us say we need to identify one such node, say N, and remove it from the MST, T and graph G. It is needed to **identify the node, N**, and new MST, T' without starting from the scratch.

(f) It is required to construct the **MST, T2 for K mandatory vertices** out of total N vertices of the given graph, G. In this process, you are free to use remaining (N-K) vertices, if needed to construct the MST. WAP to construct the MST, T2.

(g) WAP to identify the second MST (i.e. next best MST).

(h) Many times it is needed to identify the **Maximum Spanning Tree**, which is a tree of all the modes with maximum combined weights of all the involved edges. WAP to compute the MaxST for a given graph.

Q2. Set of problems based on Shortest Path:

For a given weighted graph G = {V, E}, we already computed the shortest distance, ShortDist between a source, S1 and a destination, D1 using Dijkastra's Algorithm. WAP to perform following:

(a) Let us consider that weight of an edge in the given Graph, G is **decreased**. Call this graph as G'. Now, there can be two possibilities, either ShortDist is still the shortest distance between S1 and D1 or it may not be the shortest distance. It is now needed to identify the shortest distance between S1 and D1 without starting from the scratch.

(b) Let us consider that weight of an edge in the given Graph, G is **increased**. Call this graph as G'. Now, there can be two possibilities, either ShortDist is still the shortest distance between S1 and D1 or it may not be the shortest distance. It is now needed to identify the shortest distance between S1 and D1 without starting from the scratch.

(c) Let us consider that an edge in the given Graph, G is **removed / deleted**. Call this graph as G'. Now, there can be two possibilities, either ShortDist is still the shortest distance between S1 and D1 or it may not be the shortest distance. It is now needed to identify the shortest distance between S1 and D1 without starting from the scratch.

(d) Let us consider that a new edge (with some weigh) in the given Graph, G is **inserted** between two vertices. Call this graph as G'. Now, there can be two possibilities, either ShortDist is still the shortest distance between S1 and D1 or it may not be the shortest distance. It is now needed to identify the shortest distance between S1 and D1 without starting from the scratch.

(e) WAP to compute the **second smallest path** between S1 and D1.

(f) WAP to compute the **maximum distance between S1 and D1** such that one vertex is to be visited at most once.