EECS4020 Design and Analysis of Algorithms

Homework 6

(On a voluntary basis: No need to submit)

Exam 3: June 15, 2022 (2 hours)

You are encouraged to work in groups. Please submit your solution (of any question) to our tutors on or before June 7, 2022, so that they may have a chance to give feedback of your work.

I. Minimum Spanning Trees

1. Let $G = (V, E)$ be a undirected connected graph such that each edge is associated with a distinct non-negative weight. Recall that the *minimum spanning tree problem* is to find a spanning tree T in G such that the *total* weight of all the edges in T is minimized.

   Here, we consider a different problem, called the *bottleneck spanning tree*, where our target is to find a spanning tree T' in G such that the *maximum* weight of all the edges in T' is minimized.

   Note that a bottleneck spanning tree is not necessarily a minimum spanning tree. See Figure 1 for an example.

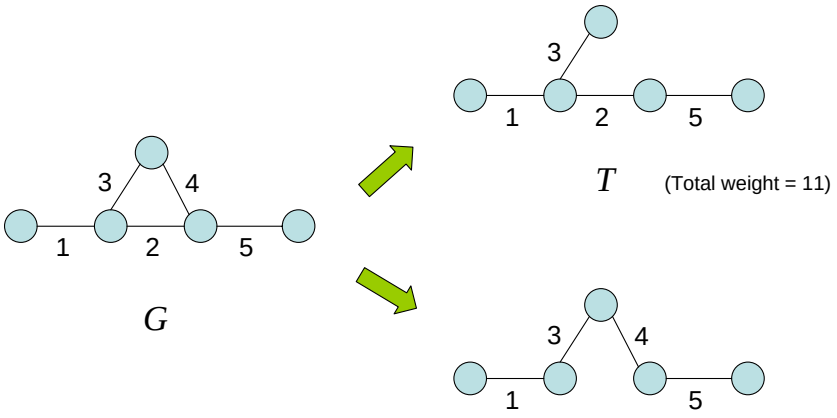

Figure 1: Examples of a minimum spanning tree and a bottleneck spanning tree.

   Show that the bottleneck spanning tree problem can be solved in $O(|V| + |E|)$ time.

   Hint: DFS/BFS, binary search on the maximum weight, contract connected components

2. Let $G = (V, E)$ be a undirected connected graph such that each *vertex* v is associated with a non-negative weight $w(v)$. For any spanning tree T in G, the weight of the spanning tree is defined to be the following value:

$$\sum_{v \in T} w(v) \times \deg_T(v),$$

   where $\deg_T(v)$ denotes the degree of v in T. Our target is to find a spanning tree in G whose weight is minimized.

   Recall that the traditional minimum spanning tree problem, whose input graph is edge-weighted, can be solved in $O(|E| + |V| \log |V|)$ time. Design an efficient algorithm for this vertex-weighted version of the minimum spanning tree problem such that its running time is still $O(|E| + |V| \log |V|)$ time.

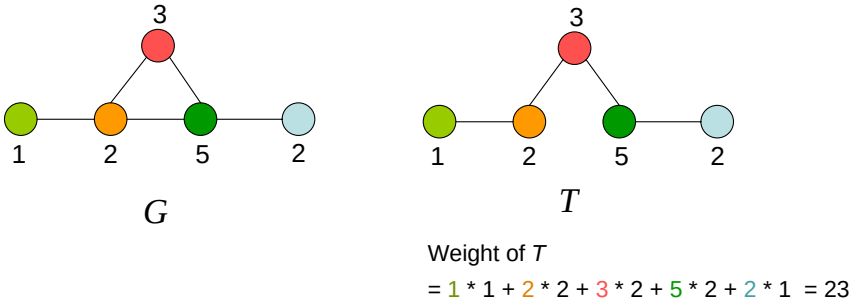

Figure 2: Example of the weight of a spanning tree.

II. Single-Source Shortest Path

1. Suppose G is a directed acyclic graph, and s and t are two vertices in G. Design a linear-time algorithm that counts the number of different paths from s to t.

   Hint: Consider applying topological sort and then dynamic programming.

2. Let $G = (V, E)$ be an edge-weighted, directed graph such that each edge has an integral weight chosen from $\{1, 2, 3, \ldots, W\}$. Modify Dijkstra's algorithm to compute the shortest paths from a given source vertex s in $O(W\,|V| + |E|)$ time. Also, give an alternative algorithm that runs in $O((|V| + |E|) \log W)$ time.

3. Given a weighted, directed graph $G = (V, E)$ with no negative-weight cycles. Suppose that for any vertex v, there exists a shortest path from s to v using at most m edges. (Here, the shortest path is by weight, not the number of edges.) Suggest a simple change to the Bellman-Ford algorithm that allows it to terminate in at most $m + 1$ passes (i.e., $m + 1$ RelaxAll operations), even if m is not known in advance.