1 Fixing Dijkstra’s Algorithm with Negative Weights

Dijkstra’s algorithm doesn’t work on graphs with negative edge weights. Here is one attempt to fix it:

1. Add a large number \( M \) to every edge so that there are no negative weights left.
2. Run Dijkstra’s to find the shortest path in the new graph.
3. Return the path found by Dijkstra’s, but with the old edge weights (i.e. subtract \( M \) from the weight of each edge).

Show that this algorithm doesn’t work by finding a graph for which it must give the wrong answer.

2 Bellman-Ford Practice

(a) Run the Bellman-Ford algorithm on the following graph, from source \( A \). Relax edges \((u, v)\) in lexicographic order, sorting first by \( u \) then by \( v \).

(b) What problem occurs when we change the weight of edge \((H, A)\) to 1? How can we detect this problem when running Bellman-Ford? Why does this work?

(c) Let \( G = (V, E) \) be a directed graph. Under what condition does the Bellman-Ford algorithm returns the same shortest path tree (from source \( s \in V \)) regardless of the ordering on edges?
3 Service scheduling

A server has $n$ customers waiting to be served. Customer $i$ requires $t_i$ minutes to be served. If, for example, the customers were served in the order $t_1, t_2, t_3, \ldots, t_n$, then the $i$-th customer would wait for $t_1 + t_2 + \cdots + t_i$ minutes.

We want to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i).$$

Given the list of the $t_i$’s, give an efficient algorithm for computing the optimal order in which to serve the customers.

4 MST Basics

For each of the following statements, either prove or give a counterexample. Always assume $G = (V, E)$ is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

(a) Let $e$ be any edge of minimum weight in $G$. Then $e$ must be part of some MST.

(b) If $e$ is part of some MST of $G$, then it must be a lightest edge across some cut of $G$.

(c) If $G$ has a cycle with a unique lightest edge $e$, then $e$ must be part of every MST.

(d) For any $r > 0$, define an $r$-path to be a path whose edges all have weight less than $r$. If $G$ contains an $r$-path from $s$ to $t$, then every MST of $G$ must also contain an $r$-path from $s$ to $t$.

5 Updating a MST

You are given a graph $G = (V, E)$ with positive edge weights, and a minimum spanning tree $T = (V, E')$ with respect to these weights; you may assume $G$ and $T$ are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to quickly update the minimum spanning tree $T$ to reflect this change, without recomputing the entire tree from scratch. There are four cases. In each, give a description of an algorithm for updating $T$, a proof of correctness, and a runtime analysis for the algorithm. Note that for some of the cases these may be quite brief.

(a) $e \notin E'$ and $\hat{w}(e) > w(e)$

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$

(c) $e \in E'$ and $\hat{w}(e) < w(e)$

(d) $e \in E'$ and $\hat{w}(e) > w(e)$