1 Shortest Paths with Negative Weights

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

(a) Add a large number $M$ to every edge so that there are no negative weights left.
(b) Run Dijkstra’s to find the shortest path in the new graph.
(c) 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.

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

![Graph Diagram]
(c) 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?

(d) Let \(G = (V, E)\) be a directed graph. Under what condition does the Bellman-Ford algorithm return the same shortest path tree (from source \(s \in V\)) regardless of the ordering on edges? (Hint: Think about how there could be multiple shortest path trees).

2 Unique Shortest Path

Shortest paths are not always unique: sometimes there are two or more different paths with the minimum possible length. Show how to solve the following problem in \(O(|V| + |E| \log |V|)\) time.

Input: An undirected graph \(G = (V, E)\); edge lengths \(l_e > 0\); starting vertex \(s \in V\).

Output: A Boolean array \(usp[\cdot]\): for each node \(u\), the entry \(usp[u]\) should be \text{true} if and only if there is a \textit{unique} shortest path from \(s\) to \(u\). (Note: \(usp[s] = \text{true}\).)
3 Dijkstra Tiebreaking

We are given a directed graph $G$ with positive weights on its edges. We wish to find a shortest path from $s$ to $t$, and, among all shortest paths, we want the one in which the longest edge is as short as possible. How would you modify Dijkstra’s algorithm to this end? Just a description of your modification is needed.

(If there are multiple shortest paths where the longest edge is as short as possible, outputting any of them is fine).

4 MST practice

Let $G = (V, E)$ be an undirected, connected graph.

(a) Prove that there is a unique MST if all edge weights are distinct.

(b) True or False? If $G$ has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a MST.

(c) True or False? If the lightest edge in $G$ is unique, then it must be a part of every MST.
5 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 $t$.

6 A Divide and Conquer Algorithm for MST

Is the following algorithm correct? If so, prove it. Otherwise, give a counterexample and explain why it doesn’t work.

\begin{verbatim}
procedure FindMST(G: graph on $n$ vertices)
    If $n = 1$ return the empty set
    $T_1 \leftarrow$ FindMST($G_1$: subgraph of $G$ induced on vertices \{1, $\ldots$, $n/2$\})
    $T_2 \leftarrow$ FindMST($G_2$: subgraph of $G$ induced on vertices \{$n/2 + 1, \ldots, n$\})
    $e \leftarrow$ cheapest edge across the cut \{1, $\ldots$, $n/2$\} and \{$n/2 + 1, \ldots, n$\}.
    return $T_1 \cup T_2 \cup \{e\}$.
\end{verbatim}