Problem set 8, problem 3 and 2

Written by: Mert Denizciler

The Algorithm:

I construct a graph G' using G. Then, I run Dijkstra's algorithm on G'. The path P' that I find corresponds to a path P in G, which the answer. Alternatively, there is no path in G', then there is no path in G either. I start with the construction of G'.

Let m_v be the number of incoming edges of node v.

For each node v in G:

- Sort the incoming edges from smallest to biggest by weight and store them in a list E_{in}.
- Sort the incoming and outgoing edges from smallest to biggest by weight and store them in a list E.
- For i in $[0,\ m_v - 1]$:
    - Add a node v_i and adjust the endpoint of $E_{in}[i]$ to be v_i.
    - Add an edge from v_i to v_{i+1} with zero weight.
    - Traverse the sorted list E from $E_{in}[i]$ to $E_{in}[i+1]$ to find outgoing edges with weight w_{out} such that:
        - $w(E_{in}[i]) < w_{out} \leq w(E_{in}[i+1])$ for problem 2
        - $w(E_{in}[i]) \leq w_{out} < w(E_{in}[i+1])$ for problem 3

        where $w(E_{in}[i])$ denotes the weight of $E_{in}[i]$.
    - Adjust the starting point of these outgoing edges to be v_i.
- Remove v from G.

Transforming P' into P:

For all v in G, replace the copies v_i of v with v. Remove all edges with zero weight that were added during the construction of G'.

Proof of correctness:

We prove that each path P' in G' is in one-to-one correspondence with a path P in G such that P satisfies the condition in problem 3. The proof can also be adapted to problem 2.

Assume P' is a path in G'. Note that for all v' in P', the weights of the outgoing edges of $v'$are bigger than or equal to the incoming edge weight of v', except for the newly added outgoing edge with zero weight. This is because of the constraint $w(E_{in}[i]) \leq w_{out}$. This newly added edge is removed when transforming P' into P, and the new edge leads to a node v'' with outgoing edge weights that are all bigger than those of v'. So, path P in G satisfies the constraint.

Conversely, assume a path P in G satisfies the constraint. This implies the existence of a path P' in G' that can be transformed into P. To construct P' from P, traverse the edges that are both in P and G'. When this is not possible, traverse a newly added edge instead.

Therefore, each path P' from s to t in G' corresponds to one path P from s to t in G that satisfies the constraint. Since Dijkstra's algorithm selects the shortest path among these, the algorithm is correct.

Runtime:

Each edge participates in the sorting at most three times, so all of the sorting is at most $O(m\log(m))$. The list E is traversed once, finding each w_{out} at most once. Each edge is adjusted at most three times. At most m nodes are added and at most n nodes are removed. At most m edges are added. So, the overall construction of G' is $O(n+m\log(m))$. G' has $O(n+m)$ nodes and $O(m)$ edges. Dijkstra's algorithm runs on G' in $O((n+m)\log(n))$. Therefore, the whole algorithm is also $O((n+m)\log(n))$.

Example: V in G, incoming edge weights are: 1, 2, 6 Outgoing edge weights are: 2, 5, 7.

Sorted list: [1, 2, 2, 5, 6, 7] = E