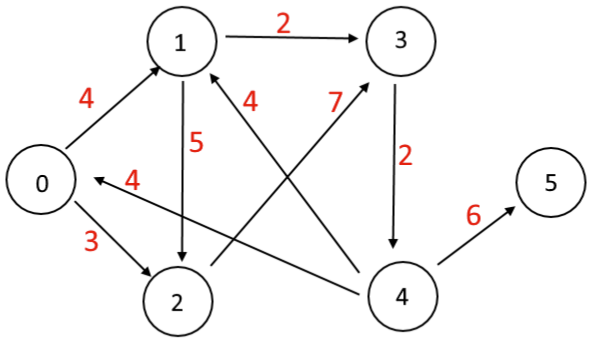
EL9343 Homework 12

Due: Dec. 15th 11:00 a.m.

1. Consider the following graph, with node 0 as the source node.
   (a) Run Dijkstra's algorithm. Write down the d array before each EXTRACT-MIN, also the final array.
   (b) Run Bellman-Ford algorithm. Write down the d distance array after each pass.



Solution:

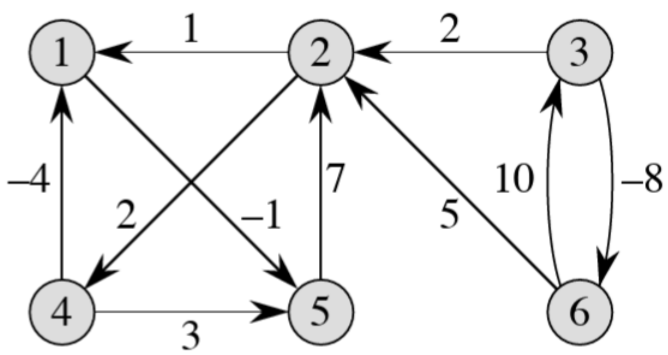
(a) The d array of running Dijkstra is as follows,

No.	0	1	2	3	4	5
1	0	∞	∞	∞	∞	∞
2	0	4	3	∞	∞	∞
3	0	4	3	10	∞	∞
4	0	4	3	6	∞	∞
5	0	4	3	6	8	∞
6	0	4	3	6	8	14

(b) The d array of running Bellman-Ford is as follows,

No.	0	1	2	3	4	5
1	0	∞	∞	∞	∞	∞
2	0	4	3	∞	∞	∞
3	0	4	3	6	∞	∞
5	0	4	3	6	8	∞
6	0	4	3	6	8	14

2. Run the Floyd-Warshall algorithm on the following graph. Show the matrix $D^{(k)}$ that results for each iteration of the outer loop.



Solution:

k	D^k
0	$\begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & \infty & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & 3 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{pmatrix}$
1	$\begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ \infty & 2 & 0 & \infty & \infty & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ \infty & 7 & \infty & \infty & 0 & \infty \\ \infty & 5 & 10 & \infty & \infty & 0 \end{pmatrix}$
2	$\begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{pmatrix}$
3	$\begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ 1 & 0 & \infty & 2 & 0 & \infty \\ 3 & 2 & 0 & 4 & 2 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 8 & 7 & \infty & 9 & 0 & \infty \\ 6 & 5 & 10 & 7 & 5 & 0 \end{pmatrix}$
4	$\begin{pmatrix} 0 & \infty & \infty & \infty & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & \infty & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix}$
5	$\begin{pmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ 0 & 2 & 0 & 4 & -1 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix}$
6	$\begin{pmatrix} 0 & 6 & \infty & 8 & -1 & \infty \\ -2 & 0 & \infty & 2 & -3 & \infty \\ -5 & -3 & 0 & -1 & -6 & -8 \\ -4 & 2 & \infty & 0 & -5 & \infty \\ 5 & 7 & \infty & 9 & 0 & \infty \\ 3 & 5 & 10 & 7 & 2 & 0 \end{pmatrix}$

3. Let $G(V, E)$ be a directed, weighted graph. The goal is to find the shortest path from a source node s to every nodes in the graph.
   (a) If all edges have unit weight, is there an algorithm faster than Dijkstra? Please just state the algorithm and its running time.
   (b) Now suppose the weight function has the form $w(\cdot) : E \to \{1, 2\}$, i.e. every edge has weight either 1 or 2. Design $O(|V| + |E|)$-time algorithm for the problem. Please show why your algorithm works and how to get the running time bound.

Solution:

(a) BFS can find the shortest path in such a graph. The time complexity is $O(|V| + |E|)$.

(b) We are going to extend the original graph so that there are only unit-weighted edges and we can directly apply BFS.

Construct a new graph $G'(V', E')$, initially empty. First copy all the nodes in V to add to V'. Next let's consider the edges.

For each edge $e = (u, v) \in E$, if $w(e) = 1$, we will add a new unit-weighted edge $e' = (u', v')$ to E'. If $w(e) = 2$, we will add a new node n_e to set V', and two edges $e'_1 = (u', n_e)$ and $e'_2 = (n_e, v')$ to set E', both being unit-weighted.

It is obvious that the path from some node u to v in G and the corresponding path through same sequence of nodes in G' have the same length. Besides, in G', there are only unit-weighted edges. To get the shortest path, we just apply BFS.

For time complexity, in the new graph G', $|V'| \leq |V| + |E|, |E'| \leq 2|E|$, where the maximum values occur when every edge in original graph has weight 2. Therefore, the BFS in the new graph takes $O(|V'| + |E'|) \to O(|V| + 3|E|) \to O(|V| + |E|)$.

4. The goal of this problem is to travel from home to a store, purchase a gift, and then get back home, at minimal cost.

   Let us model this problem using a directed graph. Let $G(V, E)$ be a directed, weighted graph, with non-negative edge weights $w : E \to \mathbb{R}^+$. The weight of an edge represents the cost of traversing that edge. Each vertex $v \in V$ also has an associated cost $c(v) \in \mathbb{R}^+$ which represents the cost of purchasing the desired gift at that location.

   Starting from "home base" $h \in V$, the goal is to find a location $v \in V$ where the gift can be purchased, along with a path p from h to v and back from v to h. The cost of such a solution is the cost $c(v)$ of the location v plus the weight $w(p)$ of the path p (i.e., the sum of edge weights along the path p).

   Design an algorithm that on input $G(V, E)$, including edge weights $w(\cdot)$ and costs $c(\cdot)$, and home base $h \in V$, finds a minimal cost solution. Assuming G is represented using adjacency lists, and Dijkstra's algorithm runs in $O(|E| \log |V|)$. Your algorithm should run Dijkstra **exactly once** and in time $O((|V| + |E|) \log |V|)$. Please show why your algorithm works and analyze the time complexity.

   Hints: If you try to solve with the original graph G, then most likely you will have to run multiple times of Dijkstra. You can try to extend the graph G into two layers. What is the number of nodes and edges?

   Solution:

   Construct a new layered graph $G'(V', E')$ as follows.

   First we copy the original graph G (including nodes and edges) together with the edge weights **twice**, to make two layers. Let's call the two layers $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ respectively. Now $V' = V_1 \cup V_2$ and $E' = E_1 \cup E_2$.

   Between the two layers, add some additional edges. For each node $v \in V$, connect its two copies $v_1 \in V_1$ and $v_2 \in V_2$ using a directed edge $e_v = (v_1, v_2)$, and set its weight as $c(v)$. Add this e_v to E'.

   (If one cannot buy the gift at some location v, we assume the corresponding $c(v) = \infty$ or any sufficiently large numbers.)

   Note that there are two copies of h in the graph. To find the optimal path, we do a Dijkstra to find the shortest path from node h_1 and to h_2.

   This path must firstly goes to some node v_1 in layer 1, then through directed edge (v_1, v_2), and finally back to h_2 in layer 2. This is because all the edges connecting the two layers are pointing from layer 1 to layer 2, making it impossible to get back to layer 1 from layer 2.

   In this path, the first part from h_1 to v_1 means starting from home and go to some location v. Second part going through v_1 to v_2 means buying gift at v. The last part from v_2 to h_2 means going back home. All the costs of traversing edges and purchasing gift are on the edge weights in the constructed graph G', so we can use standard shortest path algorithm to solve it.

   Consider the number of nodes and edges, $|V'| = 2|V|$ and $|E'| = 2|E| + |V|$. The running of Dijkstra is $O(|E'| \log |V'|) \to O((|E| + |V|) \log |V|)$.

   Note: There are also other solutions, like some using the transpose graph. The requirements are to run Dijkstra exactly once and the algorithm has the correct output.