1. (15 points) Suppose Dijkstra’s algorithm is run on the following graph, starting at node A

- A : B(4), H(8)
- B : A(4), C(8), H(11)
- C : B(8), D(7), F(4), I(2)
- D : C(7), E(9), F(14)
- E : D(9), F(10)
- F : C(4), D(14), E(10), G(2)
- G : F(2), H(1), I(6)
- H : A(8), B(11), G(1), I(7)
- I : C(2), G(6), H(7)

Note that the numbers in brackets indicate the weight of the edge.

Answer the following questions.

(a) Draw a table showing the intermediate distance values and previous values of all the nodes at each iteration of the algorithm.

(b) Draw the final shortest path tree.

2. (20 points) Given a strongly connected directed graph, \( G = (V, E) \) with positive edge weights along with a particular node \( v_0 \in V \). You wish to pre-process the graph so that queries of the form "what is the length of the shortest path from \( s \) to \( t \) that goes through \( v_0 \)” can be answered in constant time for any pair of distinct vertices \( s \) and \( t \). The pre-processing should take the same asymptotic run-time as Dijkstra’s algorithm. Analyse the runtime and provide a proof of correctness.

3. (20 points) In cases where there are several different shortest paths between two nodes (and edges have varying lengths), the most convenient of these paths is often the one with fewest edges. For instance, if nodes represent cities and edge lengths represent costs of flying between cities, there might be many ways to get from city \( s \) to city \( t \) which all have the same cost. The most convenient of these alternatives is the one which involves the fewest stopovers. Accordingly, for a specific starting node \( s \), define

\[
\text{best}[u] = \text{Minimum number of edges in a shortest path from } s \text{ to } u
\]

In the example below, the best values for nodes S, A, B, C, D, E, F are 0, 1, 1, 1, 2, 2, 3, respectively.
Give an efficient algorithm for the following problem. Analyse the runtime and provide a proof of correctness.

**Input:** Graph $G = (V, E)$; positive edge lengths $l_e$; starting node $s \in V$.

**Output:** The values of $\text{best}[u]$ should be set for all nodes $u \in V$.

4. (20 points) You are driving down a very long highway, with gas stations at mile-posts $m_1, m_2, \ldots, m_n$, where $m_1 = 0$ is your starting point and $m_n$ is your final destination. You want to make as few gas stops as possible, but your car can only hold enough gas to cover $M$ miles. Give an algorithm to find the minimum number of stops you need to make. Argue the correctness of the algorithm, and analyze its running time.

5. (25 points) Alice wants to throw a party and is deciding whom to call. She has $n$ people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to a constraint: at the party, each person should have at least five other people whom they know.

   (a) Give a high-level description of an efficient algorithm that takes as input the list of $n$ people and the list of pairs who know each other and outputs the best choice of party invitees. Argue that this scheme is correct.

   (b) Give an efficient implementation (psuedocode) of the scheme from part (a), and analyze its running time in terms of $n$. 

2 of 2