[CS252]-Algorithms II		Spring 2020-21
	Homework: 1	
Name: Sudhir Sharma	Roll No: 12041500	email: sudhirsharma@iitbhilai.ac.in
Collaborators Names:		

Solution of Problem 1

I just use the process use in the class (proof by induction). In the following, G is the input graph, s is the source vertex, $l(uv)$ is the length of an edge from u to v, and V is the set of vertices.

Pseudo Code

```
Dijkstra(G,s) for all u ∈ V \ {s}, d(u)=∞
    d(s) = 0
    R = {}
    while R! = V
     pick u ∉ R with smallest d(u)
        R = R ∪ {u}
        for all vertices v adjacent to u
            if d(v) > d(u) + l(u,v)
               d(v) = d(u) + l(u,v)
```

Let $d(v)$ be the label found by the algorithm and let $\delta(v)$ be the shortest path distance from s-to-v. We want to show that $d(v) = \delta(v)$ for every vertex v at the end of the algorithm, showing that the algorithm correctly computes the distances. We prove this by induction on $|R|$ via the following lemma:

We have to show For each $x \in R$, $d(x) = \delta(x)$.

Proof by Induction: *Base case* ($|R| = 1$)*:* Since R only grows in size, the only time $|R| = 1$ is when $R = \{s\}$ and $d(s) = 0 = \delta(s)$, which is correct.

Inductive hypothesis: Let u be the last vertex added to R. Let $R` = R \cup \{u\}$. Our I.H. is: for each $x \in R`$, $d(x) = \delta(x)$.

Using the I.H.: By the inductive hypothesis, for every vertex in R^0 that isn't u, we have the correct distance label. We need only show that $d(u) = \delta(u)$ to complete the proof.

Suppose for a contradiction that the shortest path from s-to-u is Q and has length

$$l(Q_x) < d(u).$$

Q starts in $R`$ and at some leaves $R`$(to get to u which is not in $R`$). Let $xy$ be the first edge along $Q$ that leaves $R`$. Let Q_x be the s-to-x subpath of Q. Clearly:

$$l(Q_x) + l(xy) \leq l(Q).$$

Since $d(x)$ is the length of the shortest s-to-x path by the I.H., $d(x) \leq l(Q_x)$, giving us

$$d(x) + l(xy) \leq l(Q_x).$$

Since y is adjacent to x, $d(y)$ must have been updated by the algorithm, so

$$d(y) \leq d(x) + l(xy).$$

Finally, since u was picked by the algorithm, u must have the smallest distance label:

$$d(u) \leq d(y).$$

Combining these inequalities in reverse order gives us the contradiction that $d(x) < d(x)$. Therefore, no such shorter path Q must exist and so $d(u) = \delta(u)$. □

This lemma shows the algorithm is correct by "applying" the lemma for $R = V$.

Solution of Problem 2:

The idea is to traverse all vertices of graph using BFS and we will use a Min Heap to store the vertices. Min Heap is used as a priority queue to get the minimum distance vertex from set of not yet included vertices. Time complexity of operations like extract-min and decrease-key value is O(logV) for Min Heap.

Steps for implementation of Dijkstra's algorithm using min-priority queue

1) Create a Min Heap of size V where V is the number of vertices in the given graph. Every node of min heap contains vertex number and distance value of the vertex.

2) Initialize Min Heap with source vertex as root (the distance value assigned to source vertex is 0). The distance value assigned to all other vertices is INF (infinite).

3) While Min Heap is not empty, do following

a) Extract the vertex with minimum distance value node from Min Heap. Let the extracted vertex be u.

b) For every adjacent vertex v of u, check if v is in Min Heap. If v is in Min Heap and distance value is more than weight of u-v plus distance value of u, then update the distance value of v.

Time Complexity:

This case is valid when-

- The given graph G is represented as an adjacency list.
- Priority queue Q is represented as a binary heap.

Here,

- With adjacency list representation, all vertices of the graph can be traversed using BFS in O(V+E) time.
- In min heap, operations like extract-min and decrease-key value takes O(logV) time.
- So, overall time complexity becomes O(E+V) x O(logV) which is O((E + V) x logV) = O(ElogV)
- This time complexity can be reduced to O(E+VlogV) using Fibonacci heap.