Homework 11, additional problems, solution set

1.i. We modify Dijkstra's algorithm to incorporate the cost of the vertices as well as the edges, which means that Dist[s] needs to be initialized to $c(s)$ rather than 0, and whenever we extend a path from u to v, we have to add in the cost $w(u,v)+c(v)$ rather than just $w(u,v)$. Pseudo code for the modified algorithm is shown below with the modifications underlined.

Dijkstra(G, s)
for $v = 1$ to n **do** Dist[v] $\leftarrow \infty$
end for
$\underline{\text{Dist}[s] \leftarrow c(s)}$
BuildQueue(Q, V, Dist);
while Q is not empty **do**
  u = DeleteMin(Q)
  for each edge (u, v) **do**
    if $\underline{\text{Dist}[u] + w(u,v) + c(v) > \text{Dist}[v]}$ **then**
      $\underline{\text{Dist}[v] \leftarrow \text{Dist}[u] + w(u,v) + c(v)}$;
      ReduceKey(Q, v, Dist[v])
    end if
  end for
end while

The processing per edge remains bounded by an $O(1)$ cost plus at most one ReduceKey. The processing per vertex is unchanged. Therefore the algorithm has the same runtime as Dijkstra's algorithm up to constant factors.

ii. Let G be the input graph. We build the following graph $H = (W, F)$. For each vertex $v \in V$, W has two vertices v^i and v^o (for in and out); it also has an edge (v^i, v^o), with weight $c(v)$. For each edge $(u, v) \in E$, W has an edge (u^o, v^i) with weight $w(u, v)$. Clearly, given G, H can be build it $O(n + m)$ time. We let s^i be the start vertex.

We claim that the length of the shortest path from s^i to v^o in H equals the length of the shortest path from s to v in G. To see this, suppose that $s = v_1, v_2, \ldots, v_k = v$ is a shortest path from s to v in G. Then $v_1^i, v_1^o, v_2^i, v_2^o, \ldots, v_k^i, v_k^o$ is a path from s^i to v^o in H of the same length as the shortest path from s to v in G. We conclude that the shortest path from s^i to v^o in H is no longer than the shortest path from s to v in G.

Now let's consider paths in H, starting at s^i. Any such path must alternate in and out vertices. Therefore any shortest path from s^i to v^o must have the form $s^i = v_1^i, v_1^o, v_2^i, v_2^o, \ldots, v_k^i, v_k^o = v^o$ for some k. But the path $s = v_1, v_2, \ldots, v_k = v$ is an equal length path from s to v in G. We conclude that the shortest path from s to v in G is no longer than the shortest path from s^i to v^o in H.

With these two observations, we conclude that the length of the shortest path from s to v in G equals the length of the shortest path from s^i to v^o in H.

This leads to the following algorithm.

a. Build H.
b. Run Dijkstra(H, s^i).
c. For each $v \in G$, report the length of the shortest path to v^o in H.

We now bound the running time of our algorithm. The time to build H is $O(n+m)$. H has $2n$ vertices and $m + n$ edges. Therefore, the overall runtime is at most a constant factor larger than the runtime of Dijkstra's algorithm on a graph of n vertices and m edges (note that by assumption, $n \le m$).

2. We begin by building the following graph $H = (F, W)$. H is going to have two copies of each vertex in G, and the paths in H will alternate between "even" copy and "odd" copy vertices, with the effect that paths between even copy vertices consist of an even number of edges. More precisely, for each vertex $v \in V$, F has the two vertices v^e and v^o. For each edge $(u, v) \in E$, W has the two edges (u^e, v^o) and (v^o, u^e).

Suppose that $s = v_1, v_2, \ldots, v_{2k+1} = v$ is a shortest even path from s to v in G. Then $v_1^e, v_2^o, v_3^e, \ldots, v_{2k+1}^e$ is a path of the same length from s^e to v^e in H. We conclude that the shortest path from s^e to v^e in H is no longer than the shortest even path from s to v in G.

Now let's consider paths in H, starting at s^e. Any such path must alternate even and odd vertices. Therefore any shortest path from s^e to v^e must have the form $s^e = v_1^e, v_2^o, v_3^e, \ldots, v_{2k+1}^e = v^e$ for some k. But the path $s = v_1, v_2, \ldots, v_{2k+1} = v$ is an equal length path from s to v in G. We conclude that the shortest even path from s to v in G is no longer than the shortest path from s^e to v^e in H.

With these two observations, we conclude that the length of the shortest even path from s to v in G equals the length of the shortest path from s^e to v^e in H.

This leads to the following algorithm.

a. Build H.

b. Run Dijkstra(H, s^e).

c. For each $v \in G$, report the length of the shortest path to v^e in H.

We now bound the running time of our algorithm. The time to build H is $O(n + m)$. H has $2n$ vertices and $2m$ edges. Therefore, the upper bound on the runtime of Dijkstra's algorithm increases by at most a factor of 2, and the overall runtime multiplies the bound on the runtime of Dijkstra's algorithm on a graph of n vertices and m edges by at most a constant factor.