Problem 1 (10 points). Prove or give a counter-example for each of the following statements.

1. Let p be a shortest path from vertex s to vertex t in a directed graph. If the length of each edge in the graph is increased by 1, p will still be a shortest path from s to t.

   Solution. False. Consider a graph where $l(s,v_1) = 3, l(v_1,v_2) = 3, l(v_2,t) = 3, l(s,t) = 10$. The shortest path changed from (s,v_1,v_2,t) to (s,t) if we add 1 to every edge length.

2. Let p be a shortest path from vertex s to vertex t in a directed graph. If the length of each edge in the graph is decreased by 1, p will still be a shortest path from s to t.

   Solution. False. Consider a graph where $l(s,v_1) = 3, l(v_1,v_2) = 3, l(v_2,t) = 3, l(s,t) = 8$. The shortest path changed from (s,t) to (s,v_1,v_2,t) if we decrease every edge length by 1.

Problem 2 (10 points). Consider the following implementation of priority queue PQ with an array S. *insert (PQ, x)*: add x to the end of S; *decrease-key (PQ, x, key)*: set the key of element x as *key*; *empty (PQ)*: check whether the size of S is 0; *find-min (PQ)*: traverse S and return the element with smallest key; *delete-min (PQ)*: first traverse S to locate element with smallest key, then remove this element by shifting all elements on its rightside.

1. Analyze the running time of each above operation.

   Solution:
   insert (PQ, x): $O(1)$
   decrease-key (PQ, x, key): $O(1)$
   empty (PQ): $O(1)$
   find-min (PQ): $O(n)$
   delete-min (PQ): $O(n)$

2. What is the running time of Dijkstra's algorithm if this implementation of priority queue is used?

   Solution: The running time of Dijstra's algorithm is the sum of the following components (where $n = |V|$):
   insert (PQ, x): $|V| \cdot O(1) = O(|V|)$
   decrease-key (PQ, x, key): $|E| \cdot O(1) = O(|E|)$
   empty (PQ): $|V| \cdot O(1) = O(|V|)$
   find-min (PQ): $|V| \cdot O(n) = O(|V|^2)$
   delete-min (PQ): $|V| \cdot O(n) = O(|V|^2)$.
   Therefore, when using this implementation of priority queue, the running time of the Dijkstra's algorithm is $O(|V|^2)$.

Problem 3 (20 points). Let $G = (V,E)$ be a directed graph with positive edge length. Let $t \in V$. Give an algorithm runs in $O(|V|^2)$ time for finding shortest paths between all pairs of nodes, such that these paths pass through t. *(Hint: use the results in Problem 2.)*

Solution. Denote by $d_t(u,v)$ the the length of the shortest path from u to v passing through t We must have that $d_t(u,v) = d(u,t) + d(t,v)$, where $d(u,t)$ represents the distance from u to t and $d(t,v)$ represents the distance from t to v. Therefore, to determine $d_t(u,v)$ for all pairs of vertices, we only need to determine $d(u,t)$ for all $u \in V$ and $d(t,v)$ for all $v \in V$.

We can determine $d(t,v)$ for all $v \in V$ by running Dijkstra's algorithm with t as the source vertex. To compute $d(u,t)$ for all $u \in V$, consider the reverse graph G_R of G. Clearly, a shortest path from u to t in G will have a corresponding shortest path from t to u in G_R. Thus, we can run Dijkstra's algorithm in G_R with t as the source vertex, which will give us $d(u,t)$ for all $u \in V$. After having $d(u,t)$ for all $u \in V$ and $d(t,v)$ for all $v \in V$, we can then compute $d_t(u,v)$ for all $u,v \in V$ simply by adding $d(u,t)$ and $d(t,v)$.

In addition to the distance, to determine the shortest path from u to v passing through t, we can concatenate the shortest path from u to t and the shortest path from t to v, both of which can be obtained within the Dijkstra's algorithm through backtracing pointers.

The running time of the above algorithm is $O(|V|^2)$. In fact, we run the Dijkstra's algorithm twice to determine $d(u,t)$ for all $u \in V$ and $d(t,v)$ for all $v \in V$, which takes $O(|V|^2)$ time (see Problem 2). The next step of computing $d(u,t)$ for all $u, v \in V$ also takes $O(|V|^2)$ time, as it takes $O(1)$ time to compute $d_t(u,v)$ for each particular u and v.

Problem 4 (20 points). Let $G = (V,E)$ be a directed graph with positive edge length. Design an algorithm runs in $O(|V|^3)$ to find the length of the shortest cycle in G.

Solution. We can run Dijkstra's algorithm $|V|$ times with each $v \in V$ as the starting source vertex. Then to find the length of the shortest cycle in the graph we calculate minimum of $d(u,v) + d(v,u)$ for all pairs of vertices $u, v \in V$.

Following Problem 2, the Dijkstra's algorithm can be implemented as having running time of $O(|V|^2)$. Therefore, the above algorithm runs in $O(|V|^3)$ time as we run Dijkstra's algorithm $|V|$ times and the last step of determing the minimum between all pairs of vertices takes $O(|V|^2)$ time.

Problem 5 (20 points). Given directed graph $G = (V,E)$ with positive edge length, vertex $s \in V$, describe how to modify Dijkstra's algorithm so that the algorithm also sets a binary array $unique[1 \cdots |V|]$, where $unique[u] = 1$ if there exists a unique shortest path from s to u, and $unique[u] = 0$ if there are more than one shortest paths from s to u. (If v cannot be reached from s, define $unique[v] = 1$.)

Solution. To find if there is a unique shortest path by modifying Dijkstra's algorithm, we do an additional checking when examining edge (v,w). If the new distance of $d[v] + l(v,w)$ is equal to $d[w]$ so far, then we have multiple possible paths from s to w with same shortest distance. So, we set $unique[w] = 0$. When we find a better shortest distance for vertex w, i.e., the case where $d[w] > d[v] + l(v,w)$, then we set $unique[w]$ to $unique[v]$. The highlighted lines indicate modification in original Dijkstra's algorithm.

Algorithm 1 ModifiedDijkstra ($G = (V,E)$, $s \in V$):

```
for each vertex v ∈ V do
    d[v] = ∞
    insert (PQ, v, ∞)
    unique[v] = 1
end for
d[s] = 0
decrease-key (PQ, s, 0)
while PQ is not empty do
    v = find-min (PQ)
    delete-min (PQ)
    for each (v,w) ∈ E do
        if d[v] + l(v,w) < d[w] then
            d[w] = d[v] + l(v,w)
            unique[w] = unique[v]
            decrease-key (PQ, w, d[v] + l(v,w))
        else if d[v] + l(v,w) == d[w] then
            unique[w] = 0
        end if
    end for
end while
```

Problem 6 (20 points). Let $G = (V, E)$ be a directed graph with positive edge length $l(e) > 0$ for any $e \in E$. For vertex $v \in V$ there is also an associated positive *vertex weight* $w(v) > 0$. For any path we define its *length* as the sum of the length of its all edges plus the sum of the weights of its all vertices. Given $s \in V$, design an algorithm runs in $O((|V| + |E|) \cdot \log |V|)$ for computing the shortest paths (in terms of this new definition of length) from s to all vertices.

Solution. For each edge $e = (u, v)$ in G, we add the weight of node v to edge e. The weight of e would be $l'(e) = l(e) + w(v)$. Then we can run Dijkstra's algorithm on G starting from s with these updated edge length. After that for every vertex $v \in V$, its eventual distance from s needs to be augmented by $w(s)$. If we use binary heap within the Dijkstra's algorithm, the running time of the above algorithm is $O((|V| + |E|) \cdot \log |V|)$.

There is an alternative method by creating a new graph $G' = (V', E')$. For each (weighted) vertex $v \in V$ in G, we split it into two unweighted vertices v_1 and v_2 and add them to V'. Then we add an edge (v_1, v_2) to E', which is assigned to have weight $w(v)$. For each edge $e = (u, v) \in E$ in the original graph G, we replace it with edge (u_2, v_1) in the new graph G'. Therefore in the new graph G' we has $|V'| = 2 \cdot |V|$ and $|E'| = |V| + |E|$. Now we can run Dijkstra algorithm on G' to find the shortest paths from s_1, and it takes time $O((|V'| + |E'|) \cdot \log |V'|) = O((2 \cdot |V| + |V| + |E|) \cdot \log(2 \cdot |V|)) = O((|V| + |E|) \cdot \log |V|)$.