Exercises - Single Source Shortest Paths - Algoritmiek Tutorial

1. **Explorer:** Consider a graph that contains m paths P_i for $1 \leq i \leq m$, each with infinite length, and has a vertex s connecting all P_i's by being adjacent to the first vertex of each P_i. More precisely, s is adjacent to vertex $v_{i,1}$ for $1 \leq i \leq m$ and vertex $v_{i,j}$ is adjacent to $v_{i,j+1}$ for $1 \leq i \leq m$ and $j \geq 1$. Someone left a treasure in one of the vertices in the graph. Our goal is to design an algorithm that starts from vertex s and is able to find the treasure. Keep in mind that there are infinite number of vertices and any algorithm cannot reach the end of path P_i for any i.

   (a) Does DFS necessarily find the treasure? Why?

   (b) Does BFS necessarily find the treasure? Why?

   Solution:

   (a) No. If DFS chooses a wrong path (having no treasure) to search at the beginning, then it never reaches the end of the path since the number of vertices planned to be searched is infinite, and thus never reaches the path containing the treasure.

   (b) Yes. The algorithm searches $v_{1,1}, v_{2,1}, \ldots, v_{m,1}$ firstly, and then $v_{1,2}, v_{2,2}, \ldots, v_{m,2}$, and so on. It eventually finds the treasure.

2. **Subpaths of shortest paths are shortest paths:** Consider any directed weighted graph $G = (V, E)$ which has no negative cycle. Prove that the subpaths of shortest paths are also shortest paths. More formally, consider any shortest path from v_0 to v_k, $v_0 \to v_1 \to \cdots \to v_k$, for any $0 \leq i \leq j \leq k$, the subpath $v_i \to v_{i+1} \to \cdots \to v_j$ is a shortest path from v_i to v_j.

   Solution:
   See textbook Lemma 24.1

3. **Shortest path subgraph:** Show that the predecessor subgraph of $G = (V, E)$ after any single source shortest path algorithm is a tree. (Note: the predecessor subgraph G_π is defined as (V, E_π), where $E_\pi = \{(\pi[v], v) \mid v \in V\}$.)

   Solution:
   By contradiction

4. Prove the correctness of DIJKSTRA's algorithm formally.

   Solution:
   The correctness can be proven formally by induction in the number of vertices v with $f[v] = true$ that $d[v] = \delta(s, v)$.

   When there is only one vertex with f-value is $true$, that is, there is only the vertex s. The value $d[s] = 0$. It is correct since the path from s to s has no edge.

   Induction hypothesis: When there are k vertices with their f-value as $true$, each of these vertices has its d-value equal to the minimum distance of any path from s to it.

   By the inductive hypothesis, consider the vertex u with $f[u] = false$ and the smallest $d[u]$, which is chosen to be the $k+1$-th vertex whose f-value is going to be set as $true$.

   Suppose for contradiction that the shortest path P from s to u has length $\ell < d[u]$. Since $f[s] = true$ and $f[u] = false$ at this moment, there is at least one edge (x, y) where $f[x] = true$ and $f[y] = false$. Consider the first of these (x, y) edges on P. By the algorithm, $d[y] \leq d[x] + w(x, y) = \delta(s, x) + w(x, y) = \ell - w(P_{yu}) < d[u] - w(P_{yu})$, where P_{yu} is the sub-path from y to u on P. According to the algorithm, u is chosen because $d[u] \leq d[y]$. Thus, $d[u] \leq d[y] < d[u] - w(P_{yu})$, which contradicts the fact that all the edge weights are non-negative. It completes the proof.

5. **Path counting:**

   (a) Consider a DAG (directed acyclic graph) $G = (V, E)$ and two vertices $s, t \in V$, give an $O(|V| + |E|)$-time algorithm to count the total number of paths from s to t. Analyze your algorithm and give the correctness proof.

   (b) Consider an arbitrary directed graph $G = (V, E)$ with weighted edges, where all the weights are non-negative. Given two vertices $s, t \in V$, determine the number of shortest paths from a source vertex s to the target vertex t. This algorithm should run in the same time as Dijkstra's algorithm.

Solution:

(a) The algorithm is described as follows.

Topologically sort the vertices of G
Remove the vertices before s and the vertices after t and their incident edges
Set $s.count = 1$
for each vertex v, taken in topological-sorted order **do**
    $v.count = \Sigma_{(u,v)\in E}\ u.count$
return $t.count$

For each vertex v, we can go from s to u for some $(u, v) \in E$ and go from u to v to obtain a path from s to v. Thus the number of paths from s to v, i.e., $v.count$, is the sum of $u.count$ for all $(u, v) \in E$. Since topological sort has no back edges, we can count the number of paths in the order. Topological sort takes $O(|V| + |E|)$ and the for-loop takes $O(|V| + |E|)$ so the total runtime is $O(|V| + |E|)$.

(b) Run the DIJKSTRA algorithm on G and remove any edge that is not a predecessor edge in the result shortest path tree. The tree is a DAG. Using a similar procedure in (a), the number of shortest paths from s to t is the final $t.count$.

6. Consider a directed graph $G = (V, E)$ with non-negative edge lengths and a starting vertex $s \in V$. The *bottleneck* of a path is defined as the minimum length of one of its edges. Show how to compute correctly, for each vertex $v \in V$, the maximum bottleneck of any $s - v$ path. Your algorithm should run in $O(|V|^2)$ time.

Solution:

Correctness. The correctness can be proven formally by induction in the number of vertices v with $f[v] = true$ that $b[v]$ is indeed the maximum bottleneck of any $s - v$ path.

When there is only one vertex with f-value is $true$, that is, there is only the vertex s. The value $b[s] = \infty$. It is correct since the path from s to s has no edge.

Induction hypothesis: When there are k vertices with their f-value as $true$, each of these vertices has its b-value equal to the maximum bottleneck on any path from s to it.

Algorithm 1 Modified`DIJKSTRA`(s)

For every vertex v, initialize $b[v] \leftarrow \infty$, $\pi[v] \leftarrow \phi$, and $f[v] \leftarrow false$
$\triangleright$ $b[v]$ is the lower bound of $s - v$ path's bottleneck, and $f[v]$ indicates if the bottleneck of $s - v$ path is fixed
$f[s] \leftarrow true$
while there is at least one vertex still unfinished **do**
    $u \leftarrow$ the vertex with $f[u] = false$ and the largest $b[u]$
    $f[u] \leftarrow true$
    for all edges (u, v) **do**
        if $b[v] < \min\{b[u], w(u, v)\}$ **then**
            $b[v] \leftarrow \min\{b[u], w(u, v)\}$
            $\pi[v] \leftarrow u$

By the inductive hypothesis, consider the vertex u with $f[u] = false$ and the largest $b[u]$, which is chosen to be the $k + 1$-th vertex whose f-value is going to be set as $true$.

Suppose for contradiction that a path Q from s to u with bottleneck $q > b[u]$. Since $f[s] = true$ and $f[u] = false$, there is at least one edge (x, y) where $f[x] = true$ and $f[y] = false$. Consider the first of these (x, y) edges on Q. Note that $q \leq b[x]$, which is the bottleneck of the vertex x. Also, due to the algorithm's update policy, $b[y] \geq \min\{b[x], w(x, y)\} \geq b[x]$. According to the algorithm, u is chosen because $b[u] \geq b[y]$. Therefore, $b[u] < q \leq b[x] \leq b[y] \leq b[u]$, which causes a contradiction. Thus the proof is completed.