Assignment 5

Aryan Choudhary - 170152
Abhinav Sharma - 180017

Algorithm

Assumption: We will consider d to be a fuel station in our algorithm for the sake of simplicity. Although, it is trivial to see that we will never fill the tank at d, since d is the last junction in the journey. So, even if there does not exist a fuel station at d, it does not matter.

We are required to find a path from s to d which satisfies the following two conditions-
Condition 1. The distance between any two consecutive fuel pumps on the path must be less than or equal to c.
Condition 2. The sum of the weights of edges on the path must be least among all such paths from s to d which satisfies condition 1. It need not be a simple path.

Definitions-
1. Path length: Sum of weights of all the edges in the path/route
2. Feasible path: Path satisfying condition 1
3. s: source, d: destination
4. c: maximum distance that can be travelled with full tank without refilling (as defined in the question)

Let the given graph be G(V,E), where-
V: set of all nodes (junctions) in G
E: set of edges (u,v) , such that u and v are junctions in G and there is a road connecting u and v. The weight (w(u,v)) of this edge is equal to the length of the corresponding road.

First of all, we will build a new graph ($G_1(V_1, E_1)$) which has following properties-
1. V_1 : Set of all junctions in G which are fuel stations (including d).
2. E_1 : Set of all edges (u,v), such that there exist a path of length not greater than c from u to v in G, where u and v are from the set V_1.
3. Weight of an edge (u,v) $\in E_1$ is the length of the shortest path from u to v in G.

Let $V_1 = (v_1, v_2, v_3, ..., v_k)$.
To build the graph (G_1), we will follow the following steps-
1. Run dijkstra algorithm from $v_i \in V_1$ in G, $\forall\ i \in [1, k]$
2. If the shortest path length $d(v_i, v_j)$ (computed using dijkstra) from v_i to v_j for any $j(i < j \leq k)$ is less than or equal to c, then add an edge (v_i, v_j) of weight $d(v_i, v_j)$ in G_1.

Note that, s is a fuel station, so s is in V_1 and also d is in V_1.
Now, we will run dijkstra algorithm from s in graph G_1 to find the shortest path from s to d.

Path restoration

We can modify the dijkstra algorithm to restore the shortest path from source vertex to any other vertex. We will maintain a Rb-tree in which the key at each node is i, where i is a node in G and also each node will contain the information parent(i), where parent(i) is the penultimate node in the shortest path from source to i. For each vertex in G there will be maximum 1 node in this Rb-tree. Note that parent(i) is just an information stored in the

node corresponding to vertex i, it is not an array.

Dijkstra-algo(s,G)

1. T $\leftarrow$ an empty RB-tree
2. U $\leftarrow$ set of all vertices reachable from s
3. L(v) $\leftarrow \infty$ for all v $\in$ U
4. L(s) $\leftarrow$ 0
5. while U is not empty do
6. y $\leftarrow$ vertex from U with minimum value of L;
7. δ(s,y) $\leftarrow$ L(y);
8. move y from U to S;
9. For each (y,v) $\in$ E with v $\in$ U do
10. if(L(v) > δ(s,y) + w(y,v))
11. L(v) $\leftarrow$ δ(s,y) + w(y,v)
12. erase node with key v from T if it exists
13. insert node with key v and parent(v) as y in T

The above pseudo code is taken directly from the lecture notes with the only changes in the for loop at step 9, in order to compute penultimate vertices. Step 2 in above pseudo code can be done with a dfs run from s and all the vertices which are visited in this dfs run are added to U.
To restore the shortest path from s to any vertex v in G -

Restore path(v, s)

1. path $\leftarrow$ initialise an empty array
2. Insert v to back of path
3. while node with key v exists in T
4. v $\leftarrow$ parent[v]
5. add v to back of path
6. if the last added node in path is not s
7. return "NO PATH"
8. else
9. reverse the array path
10. return path

If there exist a path from s to v in G, then an array will be returned , which contains the shortest path from s to v in G.

First we will restore path in G_1 by repeatedly finding ancestors of d until we reach s. Let $p_1, p_2, ..., p_l$ be ancestors of d in G_1 where $p_1 = s$ and $p_l = d$ such that p_i is parent of p_{i+1}. Also similarly we can restore path in G by finding nodes on shortest path from p_i to p_{i+1} for $\forall 1 <= i < l$.

Proof of Correctness

Let the shortest possible path (also referred to as optimal path, shortest feasible route) with the given constraint be $u_1, u_2, .., u_l$, where $u_1 = s$ and $u_l = d$ and l is the number of junctions in the path including s and d. Also let $u_{k1}, u_{k2}, .., u_{kr}$ be the fuel junctions in the above shortest path.

1.**In optimal path, we visit a fuel station only once i.e. $u_{ki} \neq u_{kj}$ for any pair (i, j), $i \neq j$ and $1 \leq i, j \leq r$.**
Lets prove it by contradiction. Lets suppose there exists two index $i < j$ such that $u_{ki} = u_{kj}$. Because u_{ki} is a fuel station. Fuel tank is full when we reach u_{ki} and again when we reach u_{kj}. Distance between two cities is positive implies edge weights are positive, which in turn implies $u_{ki}, u_{ki+1}...u_{kj-1}, u_{kj}$ is a positive weight cycle. Consider the path which removes this cycle i.e $u_1, u_2, .., u_{ki-1}, u_{ki}, u_{kj+1}, .., u_l$. This path just removes some edges from the optimal path. Positive weight edges implies sum of edge weights along this path is smaller than the optimal path. Also note that it is a valid path between s and d. Hence we found a valid path with sum of edge weights lower than optimal path. This contradicts the fact that it was optimal path. Hence, we can say that optimal path visits fuel stations only once.

2. **In optimal path, path between two consecutive fuel stations is shortest path among them. i.e. $u_{ki}, u_{ki+1}, ..., u_{kj-1}, u_{kj}$ $(j = i + 1)$ is shortest path between u_{ki} and u_{kj} for all $1 \leq i < r$.**
$j = i + 1$ in the proof.
Using similar argument we used in lemma 1. Proof by contradiction. Suppose there exists two consecutive fuel station u_{ki}, u_{kj} such that the path H $(u_{ki}, u_{ki+1}, ..., u_{kj-1}, u_{kj})$ is not the shortest path among them. Let path P $(u_{ki} = p_1, p_2, ..., p_{m-1}, p_m = u_{kj})$ be the shortest path. Consider the path S $(u_1, u_2, ..., u_{ki-1}, u_{ki} = p_1, p2, ..., p_{m-1}, p_m = u_{kj}, u_{kj+1}, ..., u_{l-1}, u_l)$. Because sum of edge weights along P is smaller than H implies edge weights along P is also less than c. Hence S is also a feasible route. Also sum of edge weights along S is smaller than the optimal path. This contradicts the fact that it was optimal path. Hence, we can say that path between any two fuel stations in optimal path is shortest path among them.

3. **For every feasible path from s to d in G, there exist a path from s to d in G_1, having path length less than or equal to original path in G.**
Consider any feasible path P $(s = p_1, p_2, .., p_x = d)$ in G, and let $(s = p_{k1}, p_{k2}, ..., p_{ky} = d)$ be the sequence of fuel stations in the order as they appear in P. Since, P is a feasible path , therefore for any i $(1 \leq i < y)$, the length of the path $(p_{ki}, p_{ki+1}...., p_{k(i+1)})$ is always less than or equal to c, i.e. the path between two consecutive fuel stations is less than or equal to c. We know that $p_{ki} \in V_1$ and $p_{k(i+1)} \in V_1$, since these are fuel stations. Also, according to our algorithm, we add an edge between every two fuel stations whose edge weight is shortest path length between these two fuel stations, if this length is not greater than c. Therefore, there exist an edge p_{ki} to $p_{k(i+1)}$ in G_1 $\forall\ i, 1 \leq i < y$. Hence, there exist the path $(s = p_{k1}, p_{k2}, ..., p_{ky} = d)$ in G_1.
For all i, $(1 \leq i < y)$, weight of the edge $(p_{ki}\ p_{k(i+1)})$ is equal to the shortest path length in G. Therefore,

$\rightarrow w(p_{ki}, p_{k(i+1)})$ in $G_1 \leq$ Path length of path $(p_{ki}, p_{ki+1}...., p_{k(i+1)})$ in G $\forall\ i, 1 \leq i < y$
$\rightarrow \sum_{i=1}^{y-1} w(p_{ki}, p_{k(i+1)})$ in $G_1 \leq \sum_{i=1}^{y-1}$ Path length of path $(p_{ki}, p_{ki+1}...., p_{k(i+1)})$ in G
$\rightarrow$ Path length of path $(s = p_{k1}, p_{k2}, ..., p_{ky} = d)$ in $G_1 \leq$ Path length of path P in G

Hence, proved.

4. **For every path from s to d in G_1, there exist a feasible path of equal length in G.**
Consider any path P $(s = p_1, p_2, .., p_x = d)$ in G_1. We know that, there is an edge $(p_i, p_{i+1})\ \forall\ i, (1 \leq i < x)$ in G_1, only if there exist a path from p_i to p_{i+1} in G and the weight of this edge is the shortest path length between p_i and p_{i+1} in G. Let, $p_i, p_i^1, p_i^2, ..p_i^{ki}, p_{i+1}$ denote the sequence of vertices in the shortest path from p_i to p_{i+1}, in G. For simplicity, let $[p_i]$ denote the sequence $(p_i^1, p_i^2, ..p_i^{ki})$. Consider another path P1 $(s = p_1, [p_1], p_2, [p_2], p_3, ..., p_{x-1}, [p_{x-1}], p_x = d)$ in G. From our algorithm , we can see that edge weight of all the edges in G_1 is less than or equal to c and also all the vertices $p_i\ \forall\ i, (1 \leq i \leq x)$ are fuel junctions, therefore the distance between any two fuel junctions in path P1

is less than or equal to c. Hence, path P1 is feasible path.
Now, lets see length of path P1 -
length of path P1 = $\sum_{i=1}^{i=x-1}$ path length of path $(p_i, [p_i], p_{i+1})$ in G
length of path P1 = $\sum_{i=1}^{i=x-1}$ shortest path length between p_i and p_{i+1} in G
length of path P1 = $\sum_{i=1}^{i=x-1}$ weight of edge (p_i, p_{i+1}) in G_1
length of path P1 = path length of path P in G_1.

Hence, proved.

Now, we will use the above lemmas to prove that our algorithm generates the shortest feasible path from s to d in G.
Let A be the shortest path from s to d in G_1, that we get after running dijkstra algorithm on graph G_1. Using lemma 4, we know that there exist a feasible path A1 in G , whose path length is same as that of path A in G_1. Hence path length of A1 is an upperbound on shortest feasible path in G. Now lets prove that it is lowerbound as well (i.e. it is the required path).
We will prove this using contradiction.
Lets assume there exists a feasible path be B1 in G such that path length of B1 is less than path length of A1. Using lemma 3, we can know that there exist a path B in G_1 corresponding to path B1 in G, such that path length of B is less than or equal to path length of B1.

Path length of A = Path length of A1 (Lemma 4)
Path length of A1 > path length of B1 (Assumption for proof by contradiction).
Path length of B1 $\geq$ Path length of B (Lemma 3)
or, Path length of A = Path length of A1 > Path length of B1 $\geq$ Path length of B
or, Path length of A > Path length of B

This contradicts the fact that A was the shortest path from s to d in $G1$ which we obtained using dijkstra algorithm. Therefore there doesnt exists a feasible route from s to d in G of path length smaller than A. Hence this is also an lowerbound on shortest feasible route. Hence path corresponding to A in $G1$ (i.e. $A1$) is the shortest feasible route from s to d in G.

Complexity Analysis

Running dijkstra from a specific source vertex in a graph G=(V,E) visits each edge only once. Hence time taken for dijkstra is bounded by $O(|E| \log |V|)$. Since no of fuel stations are bounded by $|V|$. Therefore, running all dijkstras from all fuel stations to create $G1 = (V1, E1)$ will take $O(|V||E|log|V|)$.

No of fuel stations reachable from a particulate station is bounded by $|E|$ because starting dijkstra from this fuel station visits each edge only once. In worst case we will find new fuel station along with an edge. Also no of total fuel stations are bounded by $|V|$. Hence bound is $min(|V|, |E|)$. Hence, $|E1| \leq |V| * min(|E|, |V|)$. Running dijkstra on $G1$ will take $O(|E1| * \log |V1|)$ i.e. $O(|V| * |E| * \log |V|)$.

For finding all nodes on shortest path from u to v we run one dijkstra along with keeping track of penultimate node. Step 12 and 13 can be performed in $O(log|V|)$ each. Therefore, overall running this dijkstra on G=(V,E) takes $O(|E| \log |V|)$. Let L be the no of fuel stations on shortest feasible route. L is bounded by $|V|$ (no of fuel stations) (lemma 1). We need $L-1$ dijkstra on G on consecutive fuel stations in shortest path in $G1$. Bounded by $O(L * |E| * \log |V|)$ i.e. $O(|V| * |E| * \log |V|)$.

Let X be the number of junctions in the shortest path between two consecutive fuel stations. Then while loop in Restore path(v,s) runs X times and in each iteration it takes $O(log|V|)$ to search the parent of v in rb-tree T. X is bounded by number of junctions reachable from v, which is further bounded by $|E|$. Therefore, Running time of Restore path(v,s) is $O(|E| \log |V|)$. For all $L-1$ pairs of consecutive fuel station, the overall running time is bound by $O(L * |E| \log |V|)$ i.e. $O(|V| * |E| \log |V|)$.

We also need one dijkstra on $G1$ to restore path from s to d which will take $O(|E1| \log |V1|)$ i.e. $O(|E| * \log |V|)$. Therefore our algorithm is $O(|E| * |V| * \log |V|)$, the required time complexity.