**Q.6**

**Solution if we can only fill full fuel tank at a given city –**

To solve this question, we need to re-model and create a new graph. Let us make the new graph as follows – In the new graph, if there is an edge from A to B, it means that a car can directly travel from city A to city B without refuelling in between. Thus, it will be a directed graph. We can make this graph as follows – In our original graph, ignore the fuel costs and call Dijkstra for every node. Our fuel tank capacity is c units. Let’s say that we called Dijkstra for node A. Dijkstra will calculate the shortest distances from node A. Note that, we can only travel from A to any city B if the distance between A and B is at most c. So, while running standard Dijkstra, when the minimum distance of any non-processed node exceeds c, we can automatically terminate the algorithm. We are allowed to terminate it from the proof of standard Dijkstra property “After any vertex v becomes marked, the current distance to it d[v] is the shortest, and will no longer change.” (Source : cp-algorithms.org). This will make the implementation a bit faster. After calling Dijkstra for A, we have the shortest distances to all other edges at a distance of upto c from A. In our new graph, we start a directed edge from A to these edges. The weight of the edge in this graph will be the fuel cost at A.

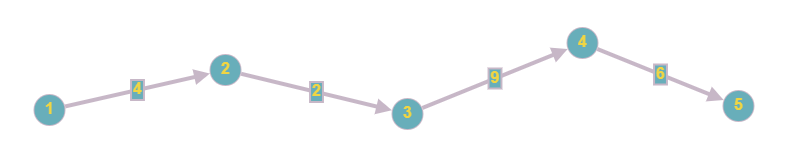
We will have to run Dijkstra like this for every city (except the ending city e). This will lead us to the new graph. Since there are V cities and running Dijkstra takes O(VlogV) worst case time (although it should be lower since we are terminating Dijkstra when dist > c), the worst case time complexity of this re-modelling is O(V^2logV), which *should* pass the constraints V<1000.

So now, we just need to call Dijkstra on our new graph. Our new graph connects every city with the cities it can reach to, and the weight of an edge is the fuel cost at the starting city of the edge. This will give us the cheapest trip cost from starting city s to ending city e. However, if e is not connected to s in this graph, there is no possible way to reach the target city.

**Now considering fuel costs are per litre and it is possible to fill a fraction of fuel tank at any stage –**

Using Dijkstra’s, we have arrived at the best case path from s to e. So our current situation is this – we have a set of cities, each of them has a distance from starting city, and a cost of fuel tank. We wish to fill fuel in such a way that the cost is minimised.

Consider the shortest path achieved –



And let the fuel costs at the cities, in order, be [3, 1, 2, 9]. (Fuel cost of target city 5 does not matter).

We want to fill the maximum amount at the lowest fuel cost, i.e., at city 2. This means the fuel at reaching 2 should be 0 and the we should fill as much fuel as needed for the rest of the journey, or the max tank capacity, i.e., min(c, dist[5]-dist[2]). Let c = 12. We will maintain an array Fuel[i], which will store the amount of fuel filled at any city i.

Now, we go to the next lowest cost city, which is city 3 where fuel costs 2. Once again, let’s fill the fuel here to the value min(c, dist[5]-dist[3]). In this case, c is less, so we fill it to 12. Currently, our fuel tank was filled at city 2 to 12 units, so it has 10 units at city 3. We will fill two more units of fuel here.

Our next city is City 1, at fuel cost 3. Here, we fill just enough fuel to reach city 2, which has lowest fuel cost. So, at city 1, we fill 4 units of fuel. Finally, we have City 4, with fuel cost 9. Our previous city with filled fuel was City 3, where there were 12 units of fuel (Previous city can be checked in O(N) worst case time, where N is number of cities in this path). So the current fuel at city 4 is 3 units. We need 6 units to reach destination (min(c, dist[5]-dist[4]), so we fill 3 more units here. This entire algorithm can run in O(N^2) time, and will give us the minimum fuel cost.