CS301-A4

Görkem Yar (Student)

May 2022

1 Tourist Problem

To find the shortest itineraries from a start city to every other city, one needs to consider every possible path (both train and bus, and multiple amount of intermediate cities) to every city. To illustrate, take Graph G with V vertices and E edges, and A and B are the vertices (cities) of the G. To find the shortest itinerary from A to B, every possible path between A and B **using at most k edges** should be examined. Since there are two possible ways to transport between vertexes and the transfer time in a city, the optimal substructure of the shortest path algorithm using at most k edges is a little bit different then using ordinary shortest path algorithm's optimal substructure.

1.1 Optimal Substructure:

1.1.1 Design of Optimal Substructure

The following notations will be used to define Optimal Substructure of the Algorithm:

- $\&_b^k(s,v)$ represents the shortest path from start city s to v, ending in the bus station of v (if it exists) with using at most k edges.
- $\&_t^k(s,v)$ represents the shortest path from start city s to v, ending in the train station of v (if it exists) with using at most k edges.
- $W_b(u,v)$ represents the distance (weight) between the cities u and v by using buses. If the edge does not exist then the weight is infinity.
- $W_t(u,v)$ represents the distance (weight) between the cities u and v by using trains. If the edge does not exist then the weight is infinity.
- TransferTime(v) represents the distance between the bus station and the train station in the given city v. If both of the stations do not exist then TransferTime(v)=∞.

At the beginning one needs to identify the terminating (or initial) condition of the Optimal Substructure. There are two possibilities of the initial condition depending on the starting station of starting city:

- First Possibility, If the starting position is the bus station of the starting city:
  - $\&_b^k(s,s) = 0$.

  - $\&_t^k(s,s)$ = Transfer time between the bus station and the train station of the first city (if train station does not exist it is infinity).

- Second Possibility, If the starting position is the train station of the starting city:
  - $\&_t^k(s,s) = 0$.
  - $\&_b^k(s,s)$ = Transfer time between the bus station and the train station of the first city (if bus station does not exist it is infinity).

These initial conditions represents the distance from starting city to the starting city using at most k edges is either zero or transfer time depends on the starting station.

Also there is another initial condition that applies for the both cases which is:

- $\&_b^0(s,v) = \infty$.
- $\&_t^0(s,v) = \infty$.

This initial condition states that the shortest path between the starting city s and a vertex v by using at most 0 edges is infinity.

Finally optimal substructure can be defined as following:

- $\&_b^k(s,v) = min_{(u,v)\in E}(min(\&_b^{k-1}(s,u) + W_b(u,v), \&_t^{k-1}(s,u) + W_t(u,v) + TransferTime(v)))$

  This notation can be explained as following: to arrive the **bus station of vertex v using at most k edges** one needs to examine every possible (u,v) pairs and find the minimum pair that gives the shortest path (v is the destination and u represents vertices that adjacent to v). In each pair there are 2 possibilities to find minimum path. The first possibility is coming from the bus station of u and the second possibility is coming from the train station of u. If we come from the train station of u, we need to add **transferTime(v)** since we want to go to bus station of v.

- $\&_t^k(s,v) = min_{(u,v)\in E}(min(\&_t^{k-1}(s,u) + W_t(u,v), \&_b^{k-1}(s,u) + W_b(u,v) + TransferTime(v)))$

  This notation can be explained as following: to arrive the **train station of vertex v using at most k edges** one needs to examine every possible (u,v) pairs and find the minimum pair that gives the shortest path (v is the destination and u represents vertices that adjacent to v). In each pair there are 2 possibilities to find minimum path. The first possibility is coming from the train station of u and the second possibility is coming from the bus station of u. If we come from the bus station of u, we need to add **transferTime(v)** since we want to go to train station of v.

One More Notation:

Up until now, we introduced two notations one of them finding the minimum distance to arrive the bus station of any city and the other one is the minimum distance to arrive the train station of any city.

Lastly, we need to introduce one more notation to find the minimum distance to every city from a starting city. This notation will compare the previous two notations for every city and will select the minimum one. As a result, we will find the minimum distance to any city by using train and bus stations.

Definition of New Notation:

- $\&_c^k(s,v)$ represents the shortest path from start city s to v.

Structure of New Notation:

- $\&_c^k(s,v) = min(\&_b^k(s,v),\ \&_t^k(s,v))$

1.1.2 Analyzing the Correctness of Optimal Substructures:

By using Cut and Paste Method this can be proven as follow

In the previous part we introduced $\&_b^k(s,v)$ as the minimum distance between s and v's bus station t.
$\&_b^k(s,v) = min_{(u,v)\in E}(min(\ \&_b^{k-1}(s,u) + W_b(u,v),\ \&_t^{k-1}(s,u) + W_t(u,v) + TransferTime(v)))$

Assume that for some u the notation should be:
$\&_b^k(s,v) = min(\ \&_b^{k-1}(s,u) + W_b(u,v),\ \&_t^{k-1}(s,u) + W_t(u,v) + TransferTime(v))$
Without lost of generality we can assume pick one side of the equation as minimum.

Two Possibilities for Picking Minimum:

- $\&_b^k(s,v) = \&_b^{k-1}(s,u) + W_b(u,v)$

  Assume that there exist another $\&_b^{'k-1}(s,u)$ such that it is smaller than $\&_b^{k-1}(s,u)$. In this case, if we cut and paste this new element in the equation, we will find a $\&_b^{'k}(s,v)$ which is smaller than $\&_b^k(s,v)$. This will contradict our first assumption as declaring $\&_b^k(s,v)$ as the minimum distance. Therefore $\&_b^{k-1}(s,u)$ needs to be smallest distance to u's bus station from s.

- $\&_b^k(s,v) = \&_t^{k-1}(s,u) + W_t(u,v) + TransferTime(v)$

  Assume that there exist another $\&_t^{'k-1}(s,u)$ such that it is smaller than $\&_t^{k-1}(s,u)$. In this case, if we cut and paste this new element in the equation, we will find a $\&_b^{'k}(s,v)$ which is smaller than $\&_b^k(s,v)$. This will contradict our first assumption as declaring $\&_b^k(s,v)$ as the minimum distance. Therefore $\&_t^{k-1}(s,u)$ needs to be smallest distance to u's train station from s.

Similar to this we can show the correctness of $\&_t^k(s,v)$ by using Cut and Paste Method.

1.1.3 Termination of the Algorithm

In the previous part, shortest distance to a city's bus station and train station were defined as

- $\&_b^k(s,v) = min_{(u,v)\in E}(min(\ \&_b^{k-1}(s,u) + W_b(u,v),\ \&_t^{k-1}(s,u) + W_t(u,v) + TransferTime(v)))$
- $\&_t^k(s,v) = min_{(u,v)\in E}(min(\ \&_t^{k-1}(s,u) + W_t(u,v),\ \&_b^{k-1}(s,u) + W_b(u,v) + TransferTime(v)))$

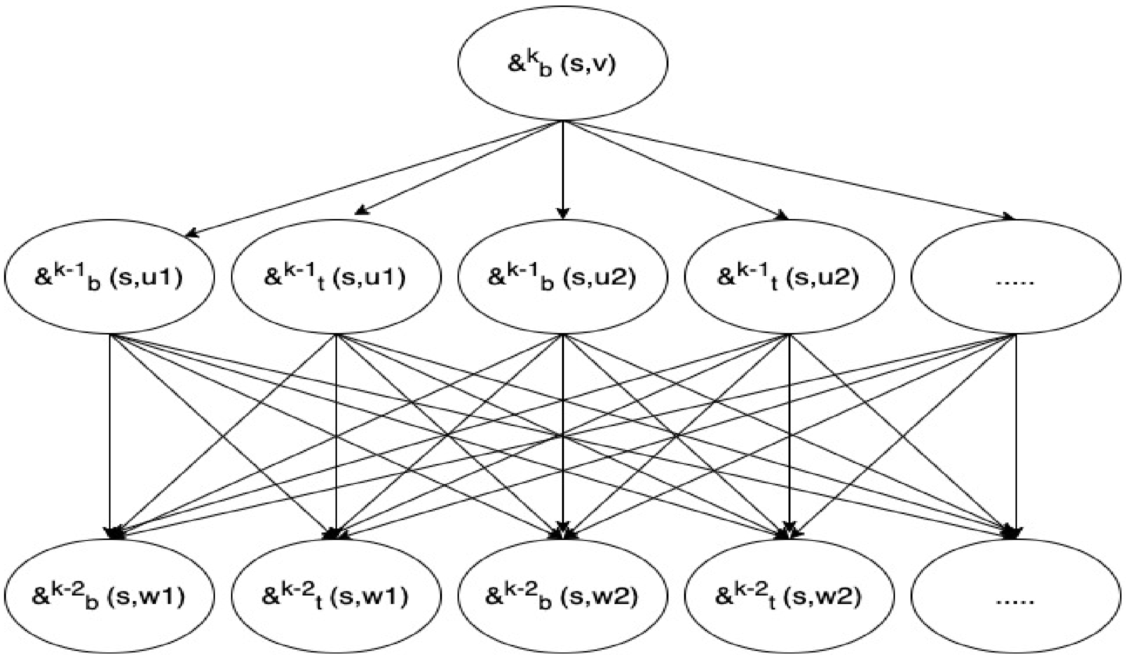
In both of the notations the k factor which is indicating the amount of edges what at most can be used to find shortest distance is decreasing by 1. In the previous chapters, the initial conditions are declared as

- $\&_b^0(s,v) = \infty$.
- $\&_t^0(s,v) = \infty$.
- Also, Optimal Substructure Design Part explains briefly the other initial conditions such as $\&_b^k(s,s) = 0$.

Since all initial conditions are defined and the k factor goes through to the 0 the algorithm is **terminating**.

1.1.4 Overlapping Subproblems

In this chapter, the overlapping subproblems will be explained for $\&_b^k(s, v)$, and also the same explanation applies for $\&_t^k(s, v)$ as well.



Overlapping Subproblems exist in each level of k.

Since there is V vertices in the graph and visiting a city more than once does not improve the shortest path, k factor can be limited by V. There is shortest path from s to every other city. So the v in $\&_b^k(s, v)$ is V as well. So, the total amount of subproblems is $O(V^2)$.

1.1.5 Computational Efficiency of the Algorithm

Time Complexity: Since there is V vertices in the graph and visiting a city more than once does not improve the shortest path, k factor can be limited by V.

In each level to find all minimum $\&_b^k(s, u_i)$, every adjacent edge to every u_i should be examined. Since all the adjacent elements to every vertex is examining and there are two possible adjacent elements this step will take O(2E)= O(E).

So the total time complexity is $V * O(E) = O(EV)$.

1.2 Algorithm Implementation

1.2.1 Pseudocode for the Algorithm

```
def BellmanFord(self, src, initialB, initialT):

       #Initialize everything except the start as Infinity
       bus = [float("Inf")] * self.V
       bus[src] = initialB
       train = [float("Inf")] * self.V
       train[src] = initialT
       dist = [float("Inf")] * self.V
       dist[src] = 0

       for _ in range V:
           for u, v, wb, wt, ttv in Graph:
               # wb= bus weight, wt train weight, ttv transfer time in the city v
               if bus[u] + wb < bus[v]:
                       bus[v] = bus[u] + wb
               if train[u] + wt + ttv < bus[v]:
                       bus[v] = train[u] + wt + ttv

               if train[u] + wt < train[v]:
                       train[v] = train[u] + wt
               if bus[u] + wb + ttv < train[v]:
                       train[v] = bus[u] + wb + ttv

               #Update Distance to a city
               if dist[v]>bus[v]:
                   dist[v]=bus[v]
               if dist[v]> train[v]:
                   dist[v]=train[v]

       for u, v, wb, wt, ttv in self.graph:
           #Many statements here just one example
           if bus[u] + wb < bus[v]:
               print("Graph contains negative weight cycle")
               return
```

1.2.2 Computational Efficiency of the Algorithm

Time Complexity: As it mentioned in the previous chapter, k is equal to Vertex count. Number of subproblems for each k_i is Vertex count as well. On the other hand, each k_i takes O(E) time because all of the edges need to be traversed to update shortest paths. As a result the Overall time complexity of the algorithm as it stated before is O(EV).

Space Complexity: For the shortest bus path an array of size V allocated, for the train shortest path an array of size V allocated and lastly for the shortest path to a city an array of size V allocated. Overall space complexity is O(3V) which is O(V).

1.3 Experimental Evaluation