Algorithm Design and Analysis

Abhimanyu Gupta : - 2019226

Abhinav Gudipati :- 2019227

Bonus Problem

Solution:

Assumptions-

- There are no multiple edges between any two pairs of vertices.
- There are no self loops.
- All edge weights are positive.
- **G** is stored as an adjacency list representation.
- There is a unique shortest path between **s**-**t**, which contains all the other vertices.
- All edges have distinct weights.

Algorithm-

➔ Initialise two hashmaps namely **ds** and **dt** with key and value for both being a vertice and a distance respectively.

➔ **ds** would store the shortest distance from **s** to every other vertex of **G**.

➔ While, **dt** would store the shortest distance from every other vertex of **G** to **t**.

➔ Run the Dijkstra algorithm with the source being s in order to fill **ds**.

➔ Now, as the shortest path for **s-t** includes every other vertex of **G**, all the vertices must lie in a line. Therefore dt can be filled as **dt(v) = ds(t) - ds(v),** $\forall$ **v** $\in$ **V**

➔ Let **E`** be a set containing all the edges in the single shortest path from **s** to **t**. These edges can be found by Dijkstra shortest path algorithm itself.

➔ Let's create a **distRange** a minimum segment tree of size **|E`|**, and initialise all entries to infinity.

- ➔ Order all the vertices in the single shortest path **s-t** as V_i for $i \in [1, |E`|]$,
- ➔ Therefore **s** will be labeled V_1 and **t** will be labeled $V_{|E`|}$
- ➔ Also, character the edge between V_i and V_{i+1} in **E`** to be e_i.
- ➔ Initialising an array **dist** of size **|E|** for storing the final result of our algorithm, where an entry for edge **e** would contain the shortest **s−t** path distance in **G \ e**.
- ➔ Now, $\forall\ e \notin E`$ $dist(e) = ds(t)$.
- ➔ Let, $e(V_i, V_j)$ denotes an edge between V_j and V_j
- ➔ And, $\forall$ edge $e(V_i, V_j)$ in **E`**, we in turn operate on **distRange** and update $e_k\ \forall\ k \in [i,j-1]$ as $ds(V_i) + |e_k| + dt(V_j)$, this would obtain us $distRange(e_k)$ as $\min_{\forall\ k \in [i,j-1]}(ds(V_i) , + |e| + dt(V_j))$.
- ➔ After the above, operations **dist(e) = distRange(e)**.

Complexity-

The vital computations in the above described algorithm are the Dijkstra shortest path algorithm to fill the two distance arrays in the starting, and the range updates in the Segment tree for each concerned edge.

Dijkstra, given adjacency list representation using heap runs in **O(|E|log|V|)**

We perform range updates in the segment tree **|E|** times, and each update requires **log|E|** computations.

Therefore, the total complexity of our algorithm is

O(|E|log|V| + |E|log|E|) ≤ O(|E|log|V| + |E|log $|V^2|$) (as $|E| \le |V^2|$) ≤ O(|E|log|V| + 2*|E|log |V|) ≤ O(|E|log|V|)