<u>OS_LAB ASSIGNMENT-3</u>

Group-17:
Pranav Nyati (20CS30037)
Vibhu (20CS10072)
Ashwani Kumar Kamal (20CS10011)
Prerit Paliwal (20CS10046)

Optimization Approach:

- In the optimized version, when new nodes are added to the graph by the producer, we will only compute the shortest path of all these new nodes to all the nodes in the graph using Dijkstra (instead of recomputing the shortest path from all the nodes, including the previous set of nodes in the updated mapped set of a consumer) and update the already computed shortest paths between the previous (older) set of mapped nodes of a consumer to the previous total nodes.

- Let the total no of nodes after the previous update = V, new nodes added = N, No of consumers = C

- **The time complexity of the unoptimized approach:**

  Each consumer applies Dijkstra roughly ((V+N)/C) times, and so for all consumers: Dijkstra applied (V + N)

  Each Dijkstra takes O(E*log(V)) , so Total complexity = **O((V+N)*E*log(V + N));**

- **The approach of optimized version:**

  - First, compute the shortest path of all the new nodes as a source to all the nodes in the graph:
    Complexity: O(N*E*log(V + N))

  - Now, for each pair of nodes, one node from the old mapped set of a consumer, and one from all the nodes in the graph, compare their previous distance to the sum of their distances from the new node. The

new node that minimized the distance b/w these old nodes will lead to a new shorter path for the pair of old nodes:
Complexity for one consumer: O((V/C)*(V)*N)
Therefore, total complexity for all consumers: O((V*V*N));

- The total complexity of the optimized version: **O (N*E*log(V + N) + V*V*N)**

- Improvement factor for the given dataset:

  Roughly,
  V = 4000;
  N = 30 (max 30 new nodes are added in each interaction);
  C = 10;
  E = 90,000 = 9*10^4;

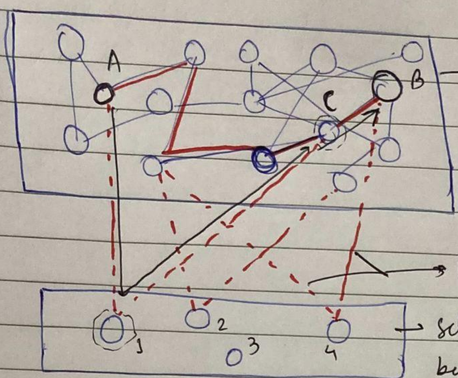
Unoptimised version: No of ops = **O((V+N)*E*log(V + N))**
= (4030)*(90000)*log(4030) = 1.3 * 10^9 ops

Optimised version: No of ops = **O(N*E*log(V + N) + V*V*N)** = 0.49 * 10^9 ops

Therefore, an improvement of a factor of approx **(1.3)/(0.49) = 2.7** is observed.

- **The reasoning for correctness:** The shortest path between the old nodes can only change due to the new edges introduced. All the new edges introduced have one of their ends in the new nodes, and we have already computed the Dijkstra for the new nodes over all of the graph. The first step thus finds the shortest path of the new nodes to all the nodes. For the old nodes, if a new path involving one or more of the new edges decreases the cost, that is the updated path; otherwise, the path b/w old notes remains unchanged.

Explanatory figure:



On adding the new nodes, calculate dist (A, new node) and dist (B, new node).

dist (B, new node) → dist of shortest path b/w B & new node

dist (A, new node) → distance of shortest path b/w A & new node (computed from dijkstra on new node)

If dist (A, B) (from previous step) > dist (A, new node) + dist (B, new node) → then dist (A, B) = dist (A, new) + dist (B, new),

then update shortest path from A to B to the path A ⟶ newnode ⟶ B.,

else no change.

In above example, the cost of A to B path from previous itrⁿ = 4
using the newly introduced nodes & edges →

Dist (A, new node 1) = 1

Dist (new node 1, B) = 2 (via node C)

∴ updated path from A to B ⇒ $\boxed{A \to \text{new node } 1 \to C \to B}$

Hence, this way older path updated,