Design and Analysis of Algorithms

ASSIGNMENT-3

Group-21 Section-C

IIB2019004 Saloni Singla

IIB2019005 Sandeep Kumar

IIB2019006 Amanjeet Kumar

Problem:

Single shortest distance problem -

It is about finding a path between vertices in a graph such that the total sum of the edges weights is minimum.

01

Approach

BFS(Breadth First Search)

ALGORITHM-1

- SET STATUS = 1 (ready state) for each node
- Enqueue the starting node A and set its STATUS = 2(waiting state)
- Repeat Steps 4 and 5 until QUEUE is empty
- Dequeue a node N. Process it and set its STATUS = 3 (processed state).
- Enqueue all the neighbours of N that are in the ready state
  (whose STATUS = 1) and set their STATUS = 2 (waiting state)
  [END OF LOOP]

Pseudo Code :

```
BFS(src, l)
    dist[src]←INT_MAX
    distance[src]←0
    q.push(src)
    while
        q.empty() =false do
        parent←q.f ront()
        q.pop()
        for auto nbr:
            l[parent] do
            if dist[nbr] = INT_MAX then
                q.push(nbr)
                distance[nbr] ← distance[parent] + 1
```

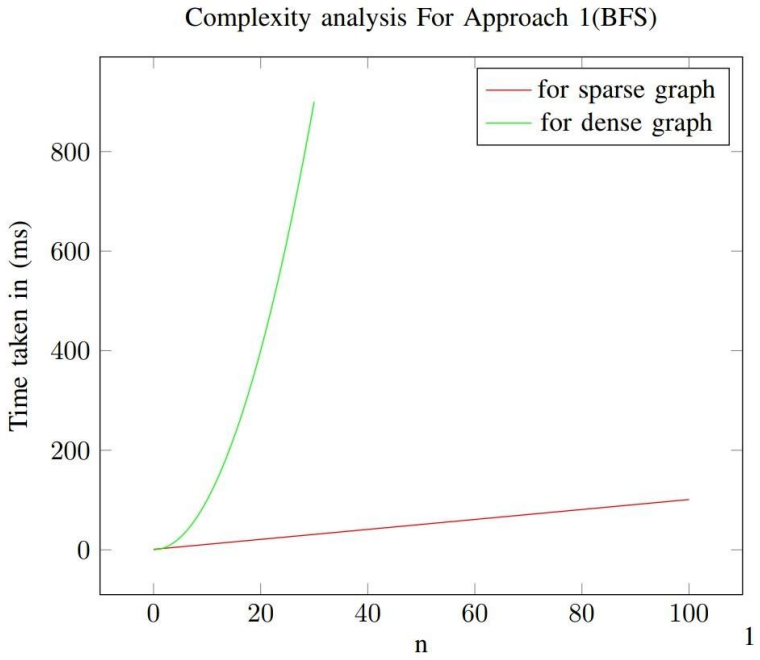
Time Complexity and Space Complexity Analysis

Time Complexity Analysis

Time complexity of breadth first search(BFS) is O(V+E), where V is the number of nodes and E is the number of edges.

Space Complexity Analysis

The space complexity for Dijkstra's algorithms is [O(V+E)(for adjacency list) + O(V) (for other arrays)]



02

Approach

Dijkstra's Algorithm

ALGORITHM-2

- Set all vertices distances = infinity except for the source vertex, set the source distance = 0.
- Push the source vertex in a min-priority queue in the form (distance , vertex), as the comparison in the min-priority queue will be according to vertices distances.
- Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).
- Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex
  with the new distance to the priority queue.
- If the popped vertex is visited before, just continue without using it.
- Apply the same algorithm again until the priority queue is empty.

Pseudo Code :

```
p.push({0,1})
    while p.empty = false do
     pair<int,int> t <- p.top()
        p.pop()
        visited[t.second] = true
        for i to less than vec[t.second].size();i++
          if (visited[vec[t.second][i].first] = false)
                and
             (t.first+vec[t.second][i].second<distance[vec[t.second][i].first]) then
                distance[vec[t.second][i].first]=t.first+vec[t.second][i].second
                p.push {distance[vec[t.second][i].first],vec[t.second][i].first
```

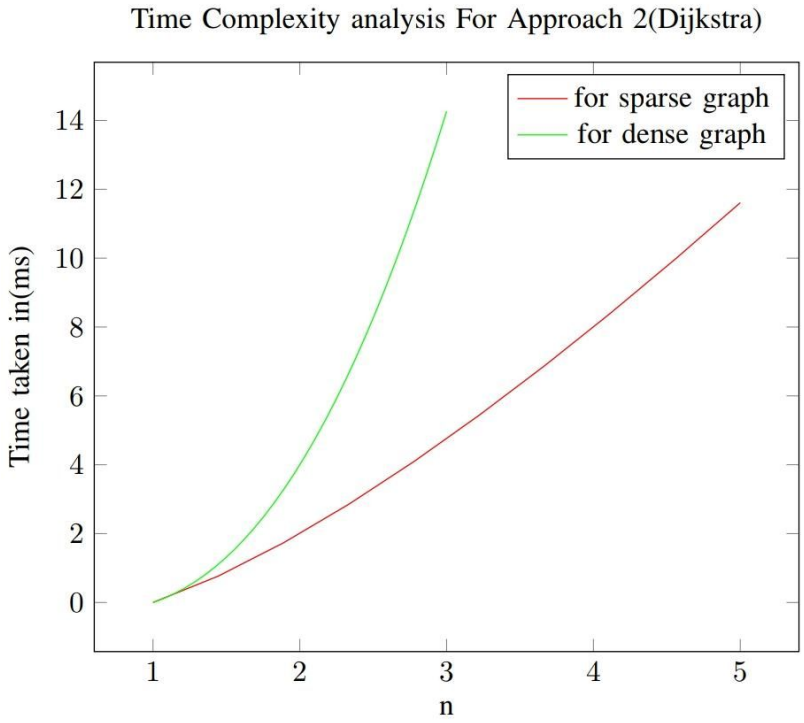
Time Complexity and Space Complexity Analysis

Time Complexity Analysis

Time complexity of Dijkstra's algorithm is O(V+ElogE), because it goes through all nodes of the graph and adds the distance for maximum of one time when at the current node.

Space Complexity Analysis

The space complexity for BFS algorithms is [O(V+E)(for adjacency list) + O(V) (for other arrays)].



03

Approach

Bellman Ford's Algorithm

ALGORITHM-3

Bellman Ford's algorithm is used to find the shortest paths from the source vertex to all other vertices in a weighted graph. It depends on the following concept: Shortest path contains at most edges, because the shortest path couldn't have a cycle.

- The outer loop traverses from 0 : n-1
- Loop over all edges, check if the next node distance > current node distance + edge weight, in this case update the next node distance to "current node distance + edge weight".

Pseudo Code :

```
SSSP(s, V, AdjList):
for i←0 to V - 1 do
    dist[i] ← Inf
dist[s] ← 0
for i←0 to V-2 do
    for u←0 to V-1 do
        for j←0 to (int)AdjList[u].size() do
        v←AdjList[u][j]
        dist[v.f irst] ← min(dist[v.f irst], dist[u] +v.second)
```

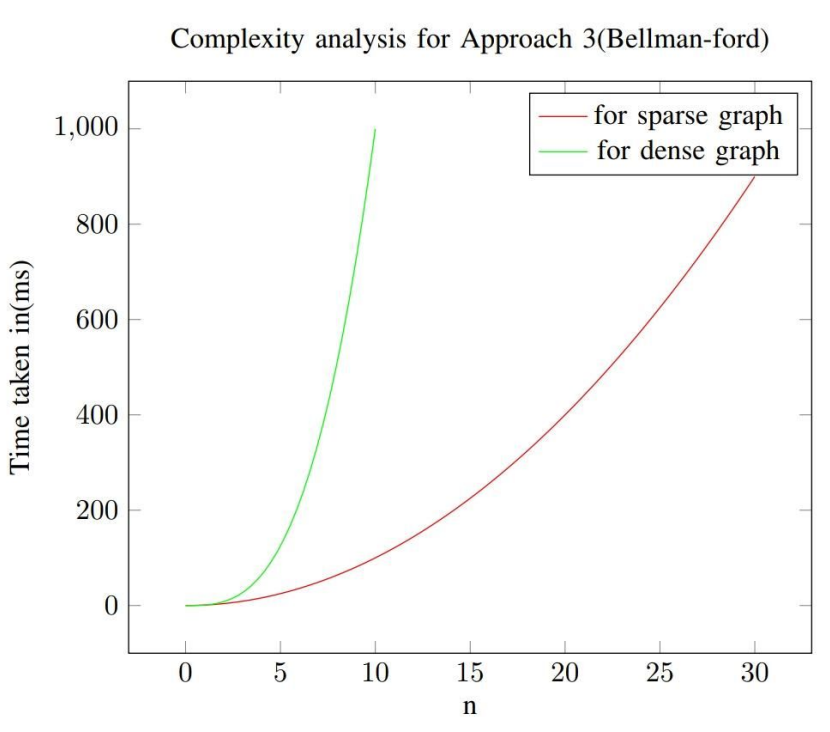
Time Complexity and Space Complexity Analysis

Time Complexity Analysis

Time complexity of Bellman Ford's algorithm is O(V*E), because it consists of (V−1) rounds and iterates through all E edges during a round.

Space Complexity Analysis

The space complexity is O(V) +O(E).



CONCLUSION

Here we discussed algorithms to solve SSSP for undirected graph and also for Unweighted graph, the distances can be computed in almost linear time complexity but for weighted graph, there exist several algorithms for different types of graphs.

REFERENCES

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/
https://www.geeksforgeeks.org/shortest-path-unweighted-graph/