Assignment 2 – PS03

DSA

Abhishek Mondal

Anuj Arora

Akash Chowdhury

Contents

GitHub Link:-

https://github.com/AkashChowdhury2000/DSA_Assignment_2.git

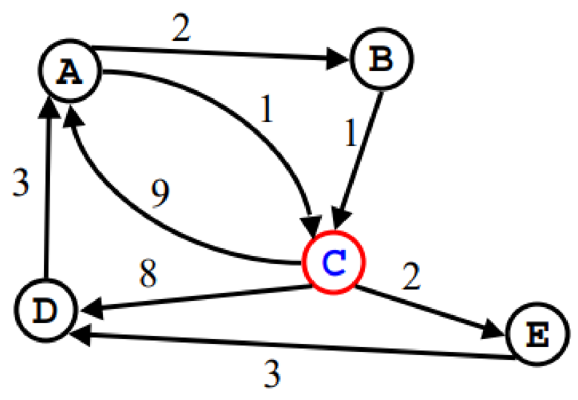
Problem Statement

Suppose there are multiple coffee vendors spread out in your office campus. During break time you want to visit the coffee stall, as you must return to your workstation as soon as possible, you want to visit the nearest coffee vendor. Your task is to find the fastest way to the coffee vendors from your workstation.

Sample Input graph:

Source node C

Coffee vendor nodes: A, B



Overview

To get an optimal path, Dijkstra Algorithm has been used.

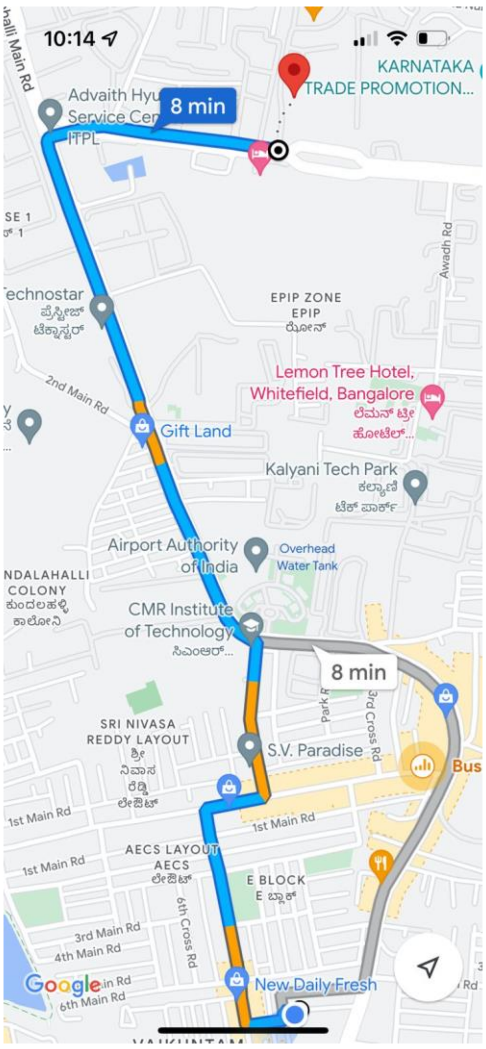
Basics of Dijkstra's Algorithm:

Dijkstra's Algorithm basically starts at the node that you choose (the source node) and it analyzes the graph to find the shortest path between that node and all the other nodes in the graph.

The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path.

Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path.

The process continues until all the nodes in the graph have been added to the path. This way, we have a path that connects the source node to all other nodes following the shortest path possible to reach each node.



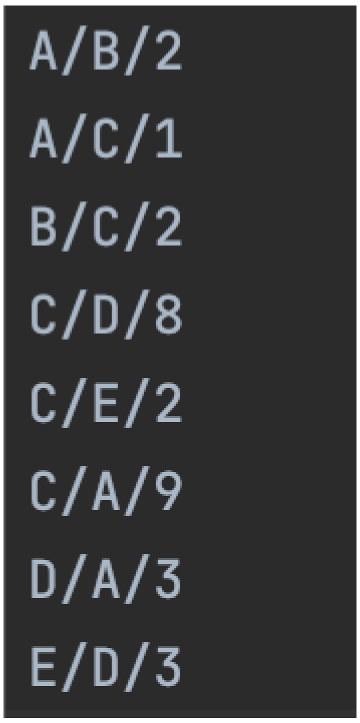
As it can be seen here, from point A to point B, there are several routes which can be chosen. However, as per google suggestion, the route with blue color is the shortest and should be preferred for the commute.

Dijkstra algorithm can also be implemented for the same purpose.

Pseudo Code

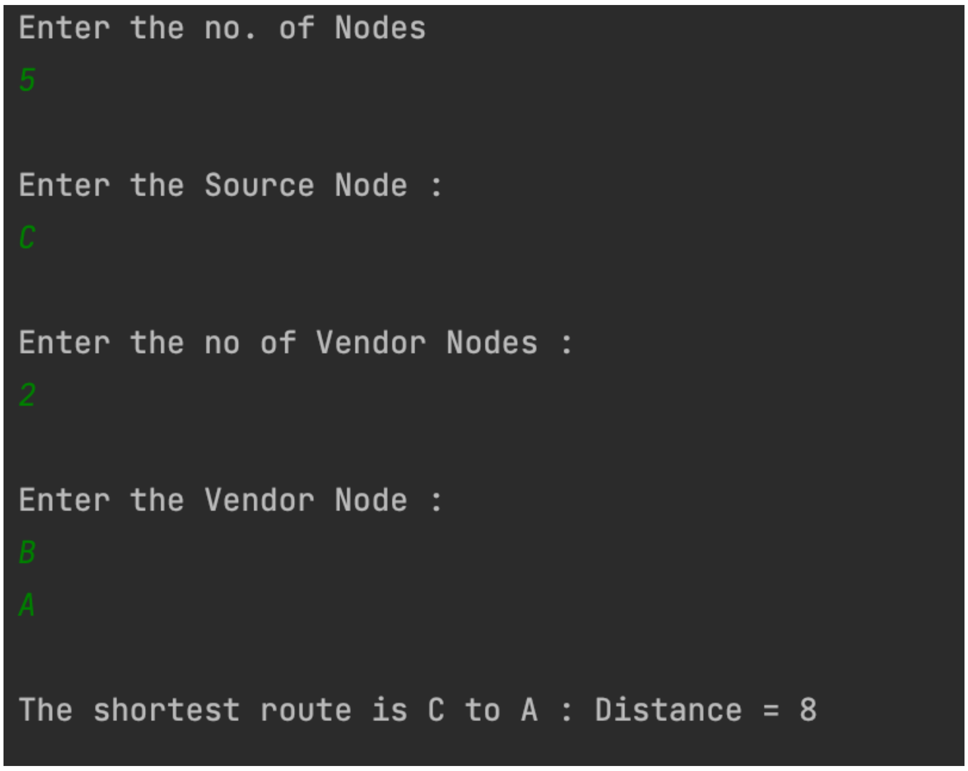
```
Function shortestPath(Int s, ArrayList<ArrayList<Node>>, int N, int vendorNodes[]): void ->
    Int dist[] = new int[N]
    For(I between 0 and N)
        dist[i] = infinite
    dist[s] = 0
    PriorityQueue(Node) pq = new PriorityQueue(N, new Node())
    Pq->add(new Node, 0)

    While pq->size() > 0:
        Node node = pq->poll()
        For(I in adj->get(node->getV)) :
            If dist[ it->getV() ] + it.getWeight() < dist[ it->getV() ]] :
                Dist[ it->getV() ] = dist[ node->getV() ] + it->getWeight()
                Pq->add(new Node(it->getV(), dist[it->getV()]))
    Int shortest = infinite, temp = 0;
    For(I in vendorNodes) :
        If dist[i] < shortest :
            Shortest = dist[i]
            Temp = i
    Print " The Shortest Path is " + char (s + 65) + "to" + char(temp + 65) + shortest
```

Input



```
A/B/2
A/C/1
B/C/2
C/D/8
C/E/2
C/A/9
D/A/3
E/D/3
```

Output



```
Enter the no. of Nodes
5

Enter the Source Node :
C

Enter the no of Vendor Nodes :
2

Enter the Vendor Node :
B
A

The shortest route is C to A : Distance = 8
```