BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
YELAHANKA, BENGALURU - 560064

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Report on Dijkstra’s Algorithm

Name	Raghavendra K M
USN	1BY18IS093
Semester/Section	5B
Course Code	18CSL57
Course Name	Computer Network Laboratory
Faculty	Prof. Gireesh babu C N
Title	Dijkstra’s Algorithm
Date	27-10-2020

Signature of a Student **Signature of a Faculty**

Dijkstra's Algorithm

Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.

It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

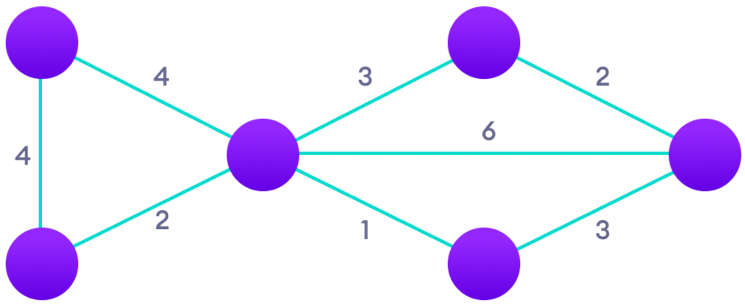
How Dijkstra's Algorithm works

Dijkstra's Algorithm works on the basis that any subpath B -> D of the shortest path A -> D between vertices A and D is also the shortest path between vertices B and D.

Djikstra used this property in the opposite direction i.e we overestimate the distance of each vertex from the starting vertex. Then we visit each node and its neighbors to find the shortest subpath to those neighbors.

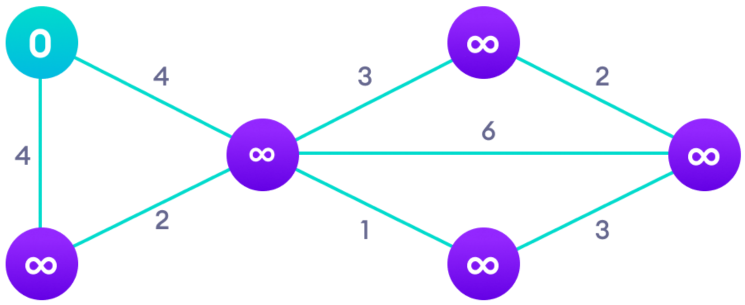
The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.

Example of Dijkstra's algorithm



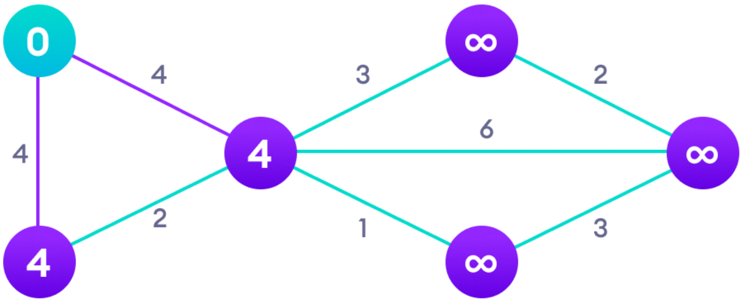
Step: 1

Start with a weighted graph



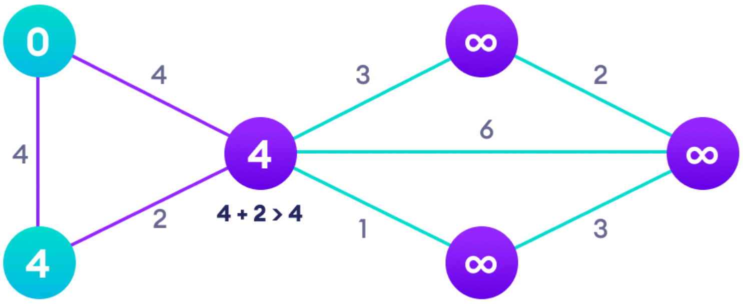
Step: 2

Choose a starting vertex and assign infinity path values to all other devices



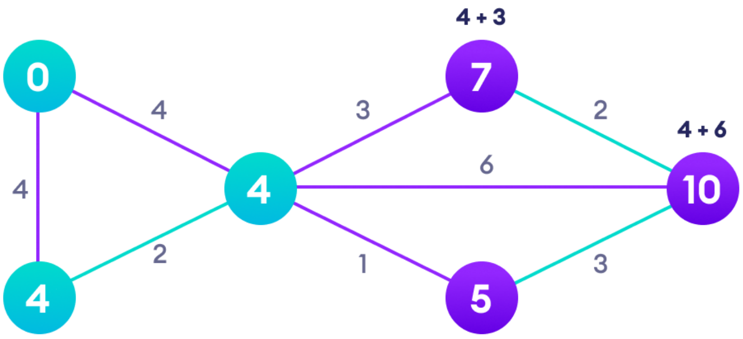
Step: 3

Go to each vertex and update its path length



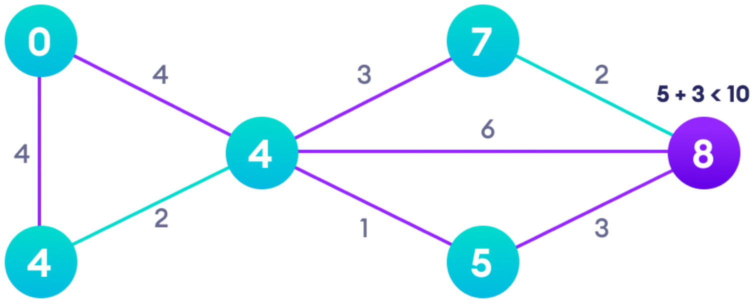
Step: 4

If the path length of the adjacent vertex is lesser than new path length, don't update it



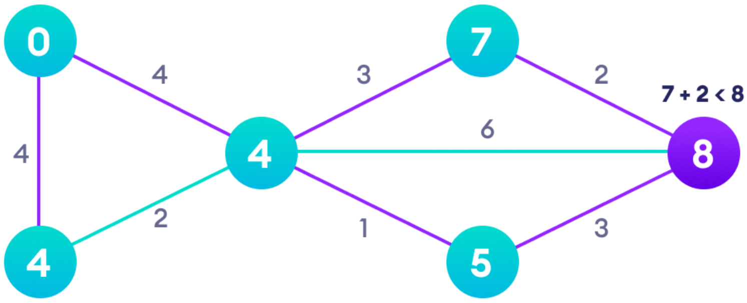
Step: 5

Avoid updating path lengths of already visited vertices



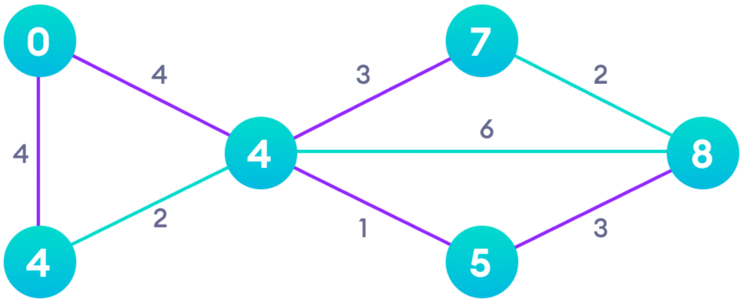
Step: 6

After each iteration, we pick the unvisited vertex with the least path length. So, we choose 5 before 7



Step: 7

Notice how the rightmost vertex has its path length updated twice



Step: 8

Repeat until all the vertices have been visited

Dijkstra's Algorithm Complexity

Time Complexity: O(E Log V)

where, E is the number of edges and V is the number of vertices.

Space Complexity: O(V)

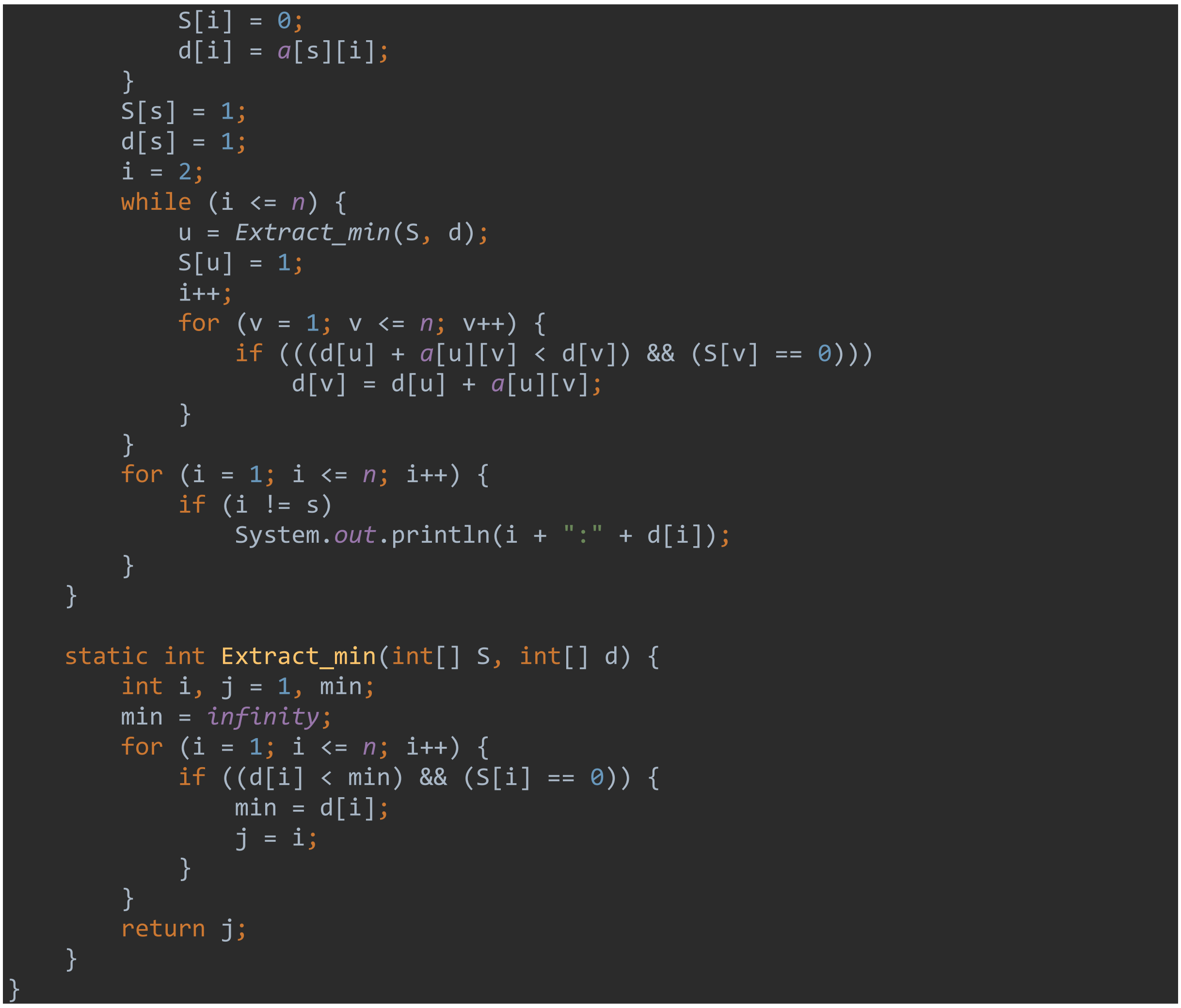
Dijkstra's Algorithm Applications

- To find the shortest path
- In social networking applications
- In a telephone network
- To find the locations in the map

Source code:

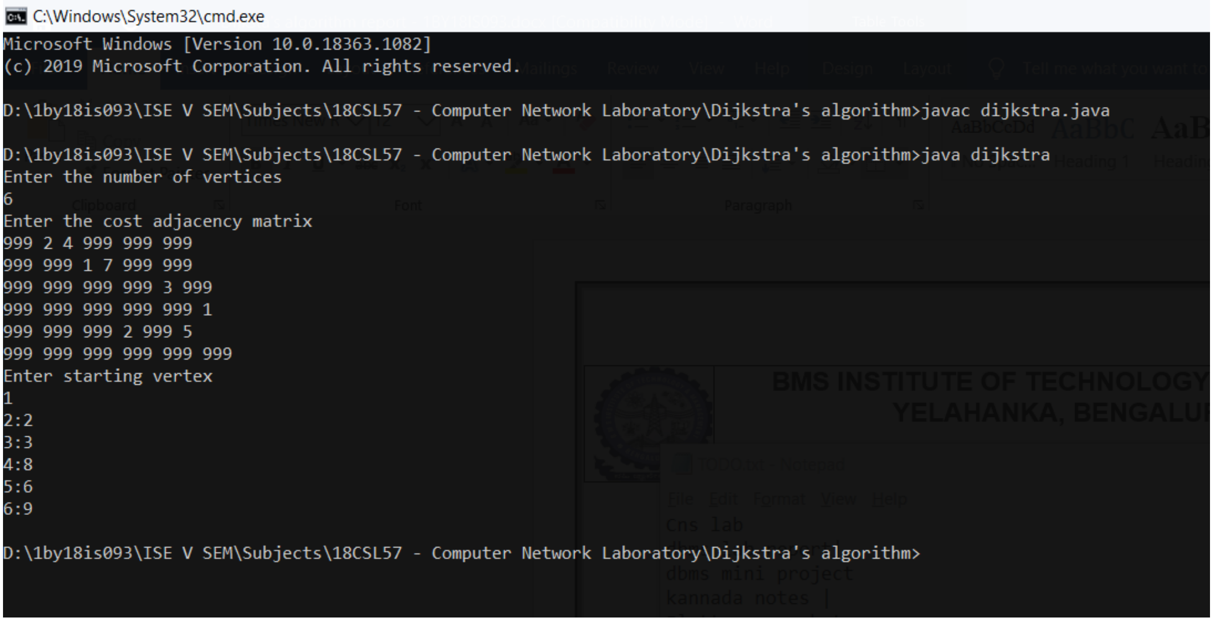
dijkstra.java

```
import java.util.Scanner;

public class dijkstra {

    final static int MAX = 20;
    final static int infinity = 9999;
    static int n;           //no of vertices of G
    static int[][] a;       //cost matrix
    static Scanner scan = new Scanner(System.in);

    public static void main(String[] args) {
        readMatrix();
        int s;           //starting vertex
        System.out.println("Enter starting vertex");
        s = scan.nextInt();
        dijkstrasMethod(s);     //find shortest path
    }

    static void readMatrix() {
        a = new int[MAX][MAX];
        System.out.println("Enter the number of vertices");
        n = scan.nextInt();
        System.out.println("Enter the cost adjacency matrix");
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++)
                a[i][j] = scan.nextInt();
    }

    static void dijkstrasMethod(int s) {
        int[] S = new int[MAX];
        int[] d = new int[MAX];
        int u, v, i;
        for (i = 1; i <= n; i++) {
```



```
            S[i] = 0;
            d[i] = a[s][i];
        }
        S[s] = 1;
        d[s] = 1;
        i = 2;
        while (i <= n) {
            u = Extract_min(S, d);
            S[u] = 1;
            i++;
            for (v = 1; v <= n; v++) {
                if (((d[u] + a[u][v] < d[v]) && (S[v] == 0)))
                    d[v] = d[u] + a[u][v];
            }
        }
        for (i = 1; i <= n; i++) {
            if (i != s)
                System.out.println(i + ":" + d[i]);
        }
    }

    static int Extract_min(int[] S, int[] d) {
        int i, j = 1, min;
        min = infinity;
        for (i = 1; i <= n; i++) {
            if ((d[i] < min) && (S[i] == 0)) {
                min = d[i];
                j = i;
            }
        }
        return j;
    }
}
```

Outputs:



```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Dijkstra's algorithm>javac dijkstra.java

D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Dijkstra's algorithm>java dijkstra
Enter the number of vertices
6
Enter the cost adjacency matrix
999 2 4 999 999 999
999 999 1 7 999 999
999 999 999 999 3 999
999 999 999 999 999 1
999 999 999 2 999 5
999 999 999 999 999 999
Enter starting vertex
1
2:2
3:3
4:8
5:6
6:9

D:\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Dijkstra's algorithm>
```