Computer Networks

Experiment No: 4

Name: Ayush Jain

SAP Id: 60004200132

Batch: B2

Computer Engineering

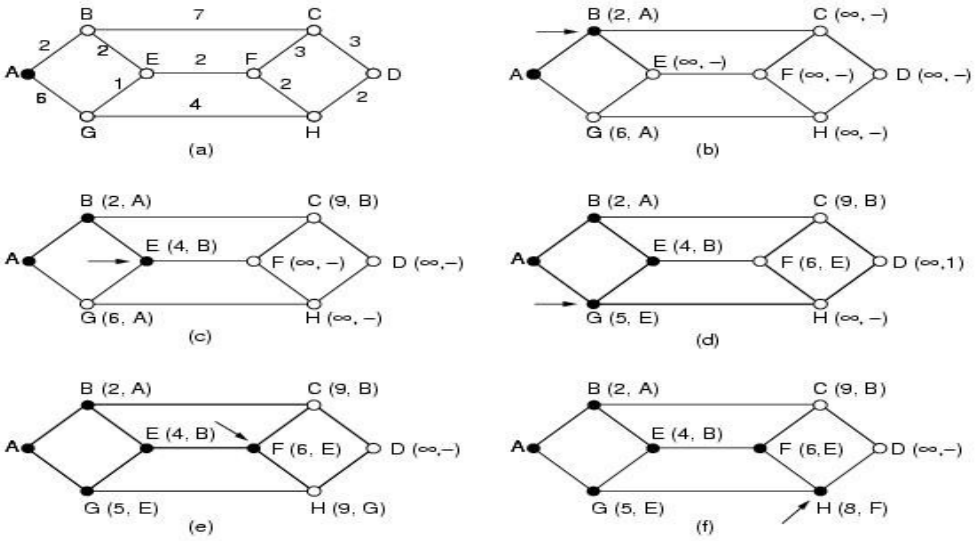
Aim: To implement Dijikstras shortest path algorithm.

Theory:

At the end each node will be labeled (see *Figure.1*) with its distance from source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change reflecting better paths. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Look at the weighted undirected graph of *Figure.1(a)*, where the weights represent, for example, distance. We want to find shortest path from A to D. We start by making node A as permanent, indicated by a filled in circle. Then we examine each of the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can construct the final path later. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in *Figure.1(b).* This one becomes new working node.

We now start at B, and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on the node, we have a shorter path, so the node is relabeled. After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. The *Figure. 1* shows the first five steps of the algorithm.



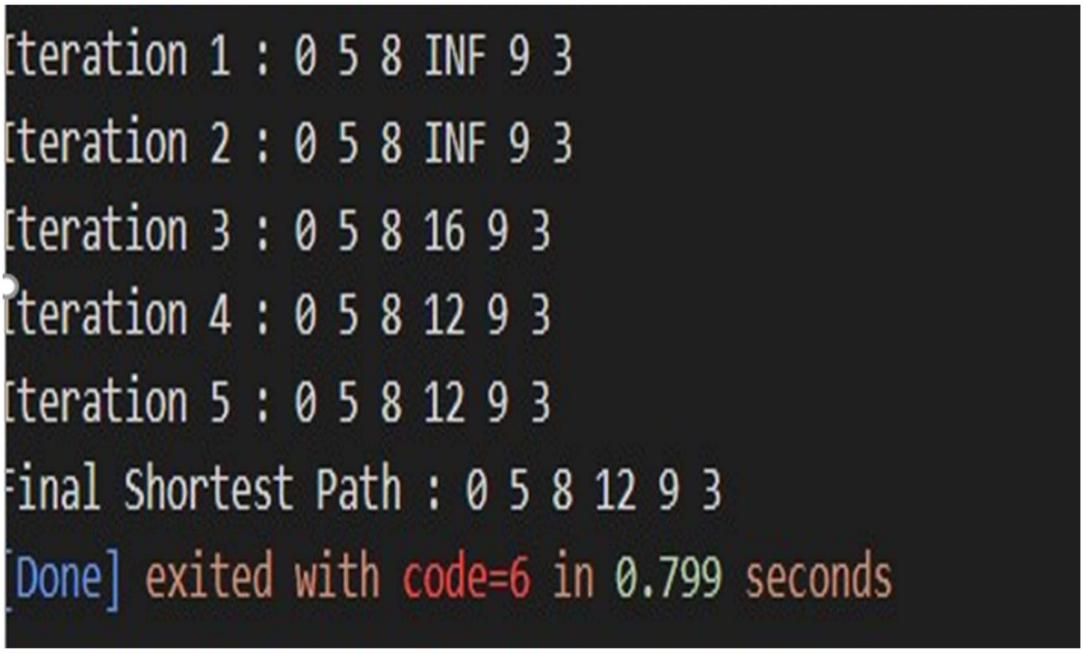
Code:

```
#include <stdio.h>

int visited[6] = {0, 0, 0, 0, 0, 0};

int distance[6];

void Dijkstra(int Graph[6][6], int n, int start)
{
  int cost[6][6];
  int count, min, next, i, j;

  for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
    {
      if (Graph[i][j] == 0)
      {
        cost[i][j] = 721;
      }      else      {
        cost[i][j] = Graph[i][j];
      }
    }
  }

  for (i = 0; i < n; i++)
  {
    distance[i] = cost[start][i];     visited[i] = 0;
  }
```

```
  distance[start] = 0;   visited[start] = 1;
count = 1;

  while (count < n)
  {
    min = 721;

    for (i = 0; i < n; i++)
    {
      if (distance[i] < min && !visited[i])
      {
        min = distance[i];
        next = i;
}
      }

    visited[next] = 1;
    printf("Iteration %d : ",count);
    for (i = 0; i < n; i++)
    {
      if (min + cost[next][i] < distance[i])
      {
        distance[i] = min + cost[next][i];
      }

      if (distance[i] == 721)         printf("INF ");
      else
        printf("%d ", distance[i]);
    }
    printf("\n");
    count++;
  }
  printf("Final Shortest Path : ");   for (i = 0; i <
n; i++)
  {
    printf("%d ", distance[i]);
  }
}

void main()
{

  int arr[6][6] = {{0, 7, 0, 0, 0, 3},
                   {7, 0, 4, 0, 0, 2},
                   {0, 4, 0, 8, 5, 5},
                   {0, 0, 8, 0, 3, 0},
                   {0, 0, 5, 3, 0, 6},
```

```
                {3, 2, 5, 0, 6, 0}};
  Dijkstra(arr, 6, 0);
}
```

Output:



Conclusion: Successfully implemented Dijikstras shortest path algorithm.