13. Graph – Shortest Path (Dijkstra's Algorithm)

Aim: To find the shortest path in a graph using Dijkstra's algorithm.

Algorithm:

1. **Input adjacency matrix of weighted graph.**
2. **Initialize distance[] with infinity, dist[source]=0.**
3. **Use visited[] to track processed nodes.**
4. **For each unvisited node, pick min distance node.**
5. **Update distances of neighbors if shorter path is found.**
6. **Repeat until all nodes are visited.**

```
#include <stdio.h>

#define INF 9999

#define V 5


void dijkstra(int G[V][V], int start) {

   int dist[V], visited[V]={0};

   for(int i=0;i<V;i++) dist[i]=INF;

   dist[start]=0;


   for(int c=0;c<V-1;c++) {

      int min=INF,u=-1;

      for(int i=0;i<V;i++)

         if(!visited[i] && dist[i]<min){ min=dist[i]; u=i; }

      visited[u]=1;

      for(int v=0;v<V;v++)

         if(!visited[v] && G[u][v] && dist[u]+G[u][v]<dist[v])

            dist[v]=dist[u]+G[u][v];
```

```
    }

    printf("Vertex Distance from Source:\n");
    for(int i=0;i<V;i++) printf("%d -> %d\n",i,dist[i]);
}

int main() {
    int G[V][V]={
        {0,10,0,30,100},
        {10,0,50,0,0},
        {0,50,0,20,10},
        {30,0,20,0,60},
        {100,0,10,60,0}};
    dijkstra(G,0);
    return 0;
}
```

Output:

Vertex Distance from Source:

0 -> 0

1 -> 10

2 -> 50

3 -> 30

4 -> 60

Result:

Shortest paths from source node were successfully computed using Dijkstra's algorithm.