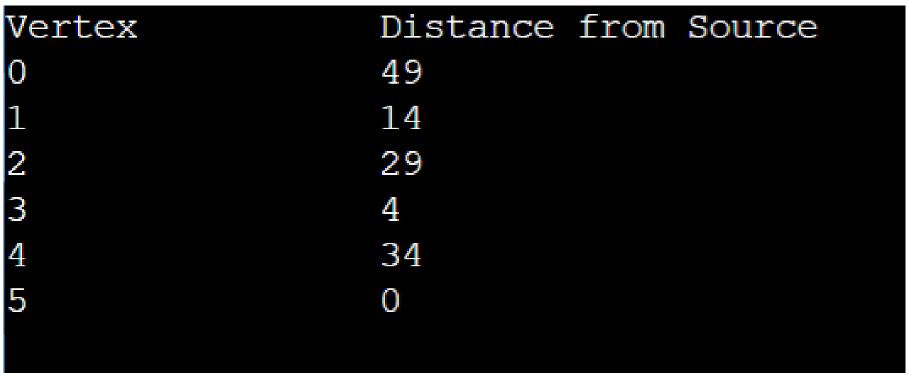
Dijkstra Algorithm

```
#include <stdio.h>

#define INF 9999
#define N 6

void dijkstra(int cost[N][N], int src) {
   int dist[N];
   int vis[N];
   int count, min_dist, u;

   for (int i = 0; i < N; i++) {
      dist[i] = INF;
      vis[i] = 0;
   }

   dist[src] = 0;

   for (count = 0; count < N-1; count++) {
      min_dist = INF;
      for (int v = 0; v < N; v++) {
         if (!vis[v] && dist[v] <= min_dist) {
            min_dist = dist[v];
            u = v;
         }
      }

      vis[u] = 1;

      for (int v = 0; v < N; v++) {
         if (!vis[v] && cost[u][v] && dist[u] != INF && dist[u] + cost[u][v] < dist[v]) {
            dist[v] = dist[u] + cost[u][v];
         }
      }
   }

   printf("Vertex \t\t Distance from Source\n");
   for (int i = 0; i < N; i++) {
      printf("%d \t\t %d\n", i, dist[i]);
   }
}

int main() {
   int cost[N][N] = {
      {0, 15, 10, INF, 45, INF},
      {INF, 0, 15, INF, 20, INF},
      {20, INF, 0, 20, INF, INF},
      {INF, 10, INF, 0, 35, INF},
      {INF, INF, INF, 30, 0, INF},
      {INF, INF, INF, 4, INF, 0}
```

```
    };

    int src = 5;
    dijkstra(cost, src);

    return 0;
}
```

Output:

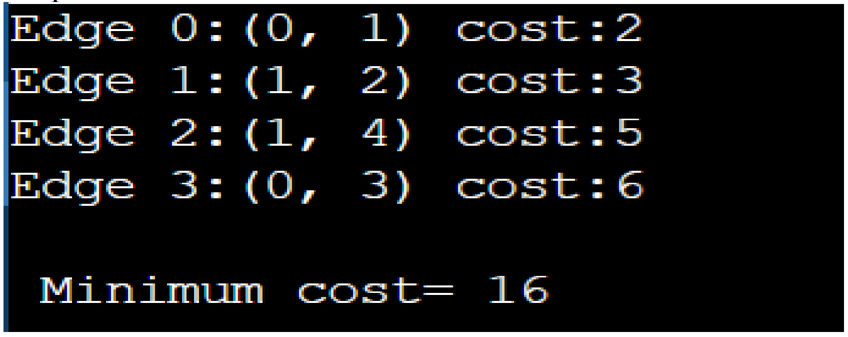


```
Vertex                Distance from Source
0                     49
1                     14
2                     29
3                     4
4                     34
5                     0
```

Kruskal Algorithm

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define V 5

int parent[V];

int find(int i)
{
    while (parent[i] != i)
        i = parent[i];
    return i;
}

void union1(int i, int j)
{
    int a = find(i);
    int b = find(j);
    parent[a] = b;
}

void kruskalMST(int cost[][V])
{
    int mincost = 0; // Cost of min MST.

    for (int i = 0; i < V; i++)
        parent[i] = i;
    int edge_count = 0;
    while (edge_count < V - 1) {
        int min = INT_MAX, a = -1, b = -1;
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (find(i) != find(j) && cost[i][j] < min) {
                    min = cost[i][j];
                    a = i;
                    b = j;
                }
            }
        }

        union1(a, b);
        printf("Edge %d:(%d, %d) cost:%d \n",
               edge_count++, a, b, min);
        mincost += min;
    }
    printf("\n Minimum cost= %d \n", mincost);
}

int main()
{
    int cost[][V] = {
        { INT_MAX, 2, INT_MAX, 6, INT_MAX },
        { 2, INT_MAX, 3, 8, 5 },
        { INT_MAX, 3, INT_MAX, INT_MAX, 7 },
        { 6, 8, INT_MAX, INT_MAX, 9 },
        { INT_MAX, 5, 7, 9, INT_MAX },
    };

    kruskalMST(cost);

    return 0;
}
```

Output:



```
Edge 0:(0, 1) cost:2
Edge 1:(1, 2) cost:3
Edge 2:(1, 4) cost:5
Edge 3:(0, 3) cost:6

 Minimum cost= 16
```