

11/2/24

C program for Dijkstra's algorithm

```
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#define V 6
```

```
int mindistance (int dist[V], bool sptset[V])
```

```
{
    int min = INT_MAX, min_index;
```

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

```
if (!sptset[v] && (v) < min) {
```

```
    min = dist[v];
```

```
    min_index = v;
```

```
}
```

```
return min_index;
```

```
void dijkstra (int graph[V][V], int src)
{
    int dist[V];
    bool sptset[V];
```

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

```
if (!sptset[v] && graph[v][v])
```

```
dist[v] = INT_MAX
```

```
dist[v] = graph[v][v];
```

```
}
```

```
print Solution (dist);
```



```

int main()
{
    int graph[V][V];
    printf("Enter the values of adjacency Matrix:");
    for(int i=0; i<V; i++) {
        for(int j=0; j<V; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    dijkstra(graph, 0);
    return 0;
}

```

Output :-

Enter the values of adj. Matrix :-

0	15	10	∞	45	∞
∞	0	15	∞	20	∞
20	∞	0	20	∞	∞
∞	10	∞	0	35	∞
∞	∞	∞	30	0	∞
∞	∞	∞	4	∞	0

Vertex	Distance from source
0	0
1	15
2	10
3	30
4	35
5	∞



Kruskals //

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#define size 100
```

```
struct Edge {
```

```
    int v, v, weight;
```

```
};
```

```
void Kruskals (int c [size] [size], int m);
```

```
int find (int parent [], int i);
```

```
void unionset (int parent [], int rank [], int x, int
```

```
int main () {
```

```
    int c [size] [size];
```

```
    printf ("Enter the no of Vertices ");
```

```
    scanf ("%d", &n);
```

```
}
```

```
void Kruskals (int c [size] [size], int m);
```

```
struct Edge [size] * Edge;
```

```
int edgeCount = 0;
```

```
int minCost = 0;
```

```
printf ("Edges in Minimum Spanning Tree");
```

```
for (int i = 0; i < edgeCount; i++);
```

```
int v = edges[i].weight;
```



```

int find (int parent[], int i) {
    if (parent[i] == 0)
        return i;
    return find (parent, parent[i]);
}

```

Output:-

Ent no of vertices : 5

Ent no of edges : 5

Ents no of edges (u, v, wt);

0 1 10

0 2 6

0 3 5

1 3 15

2 3 4

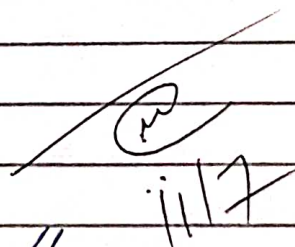
Shortest paths

2 - - 3 = 4

0 - 3 = 5

0 - 1 = 10

Minimal Spanning Tree = 19 //

  
11/7