main.c

Run

```c
#192210211
#include <stdio.h>
#define INFINITY 9999
#define MAX 10
void Dijkstra(int Graph[MAX][MAX], int n, int start);
void Dijkstra(int Graph[MAX][MAX], int n, int start) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (Graph[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = Graph[i][j];
  for (i = 0; i < n; i++) {
    distance[i] = cost[start][i];
    pred[i] = start;
    visited[i] = 0;
  }
  distance[start] = 0;
  visited[start] = 1;
  count = 1;
  while (count < n - 1) {
    mindistance = INFINITY;
    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i; }
    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
```

main.c

Run

```
    if (!visited[i])
      if (mindistance + cost[nextnode][i] < distance[i]) {
        distance[i] = mindistance + cost[nextnode][i];
        pred[i] = nextnode;}
    count++;}
  for (i = 0; i < n; i++)
    if (i != start) {
      printf("\nDistance from source to %d: %d", i, distance[i]);}}
int main() {
  int Graph[MAX][MAX], i, j, n, u;
  n = 7;
  Graph[0][0] = 0;
  Graph[0][1] = 0;
  Graph[0][2] = 1;
  Graph[0][3] = 2;
  Graph[0][4] = 0;
  Graph[0][5] = 0;
  Graph[0][6] = 0;
  Graph[1][0] = 0;
  Graph[1][1] = 0;
  Graph[1][2] = 2;
  Graph[1][3] = 0;
  Graph[1][4] = 0;
  Graph[1][5] = 3;
  Graph[1][6] = 0;
  Graph[2][0] = 1;
  Graph[2][1] = 2;
  Graph[2][2] = 0;
  Graph[2][3] = 1;
  Graph[2][4] = 3;
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
```

main.c

Run

```
60    Graph[2][4] = 3;
61    Graph[2][5] = 0;
62    Graph[2][6] = 0;
63    Graph[3][0] = 2;
64    Graph[3][1] = 0;
65    Graph[3][2] = 1;
66    Graph[3][3] = 0;
67    Graph[3][4] = 0;
68    Graph[3][5] = 0;
69    Graph[3][6] = 1;
70    Graph[4][0] = 0;
71    Graph[4][1] = 0;
72    Graph[4][2] = 3;
73    Graph[4][3] = 0;
74    Graph[4][4] = 0;
75    Graph[4][5] = 2;
76    Graph[4][6] = 0;
77    Graph[5][0] = 0;
78    Graph[5][1] = 3;
79    Graph[5][2] = 0;
80    Graph[5][3] = 0;
81    Graph[5][4] = 2;
82    Graph[5][5] = 0;
83    Graph[5][6] = 1;
84    Graph[6][0] = 0;
85    Graph[6][1] = 0;
86    Graph[6][2] = 0;
87    Graph[6][3] = 1;
88    Graph[6][4] = 0;
89    Graph[6][5] = 1;
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
```

programiz.com

Online C Compiler | jeevanmuchakarla/ds.csa0344-Data-structures

Programiz C Online Compiler

C Certification ›

main.c

Run

```
65    Graph[3][2] = 1;
66    Graph[3][3] = 0;
67    Graph[3][4] = 0;
68    Graph[3][5] = 0;
69    Graph[3][6] = 1;
70    Graph[4][0] = 0;
71    Graph[4][1] = 0;
72    Graph[4][2] = 3;
73    Graph[4][3] = 0;
74    Graph[4][4] = 0;
75    Graph[4][5] = 2;
76    Graph[4][6] = 0;
77    Graph[5][0] = 0;
78    Graph[5][1] = 3;
79    Graph[5][2] = 0;
80    Graph[5][3] = 0;
81    Graph[5][4] = 2;
82    Graph[5][5] = 0;
83    Graph[5][6] = 1;
84    Graph[6][0] = 0;
85    Graph[6][1] = 0;
86    Graph[6][2] = 0;
87    Graph[6][3] = 1;
88    Graph[6][4] = 0;
89    Graph[6][5] = 1;
90    Graph[6][6] = 0;
91    u = 0;
92    Dijkstra(Graph, n, u);
93    return 0;
94 }
```

Output

Clear

```
/tmp/dKzoRx1qjh.o
Distance from source to 1: 3
Distance from source to 2: 1
Distance from source to 3: 2
Distance from source to 4: 4
Distance from source to 5: 4
Distance from source to 6: 3
```