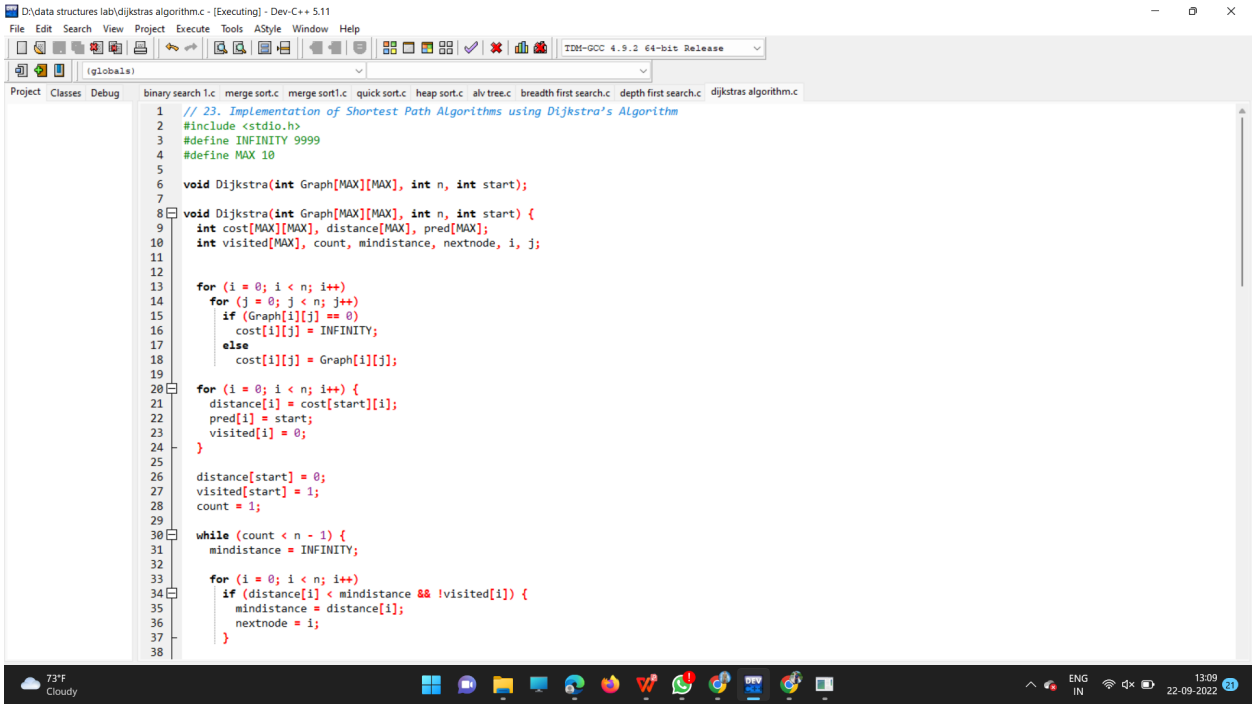
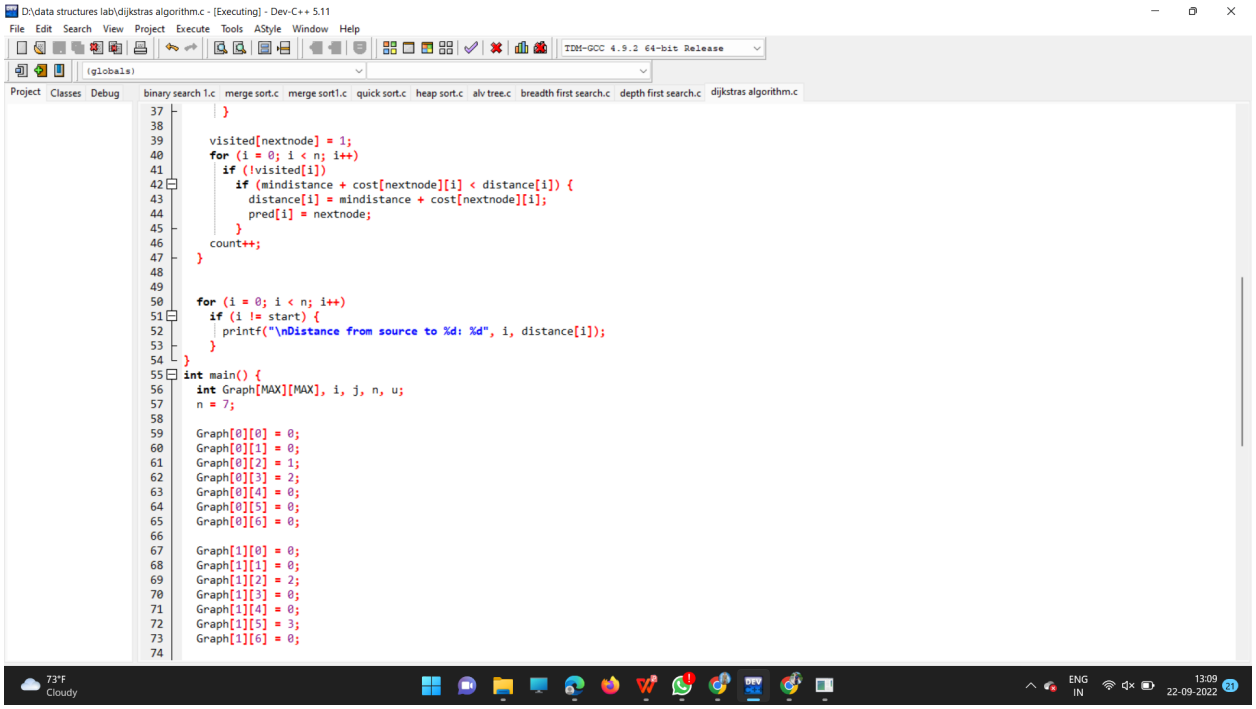
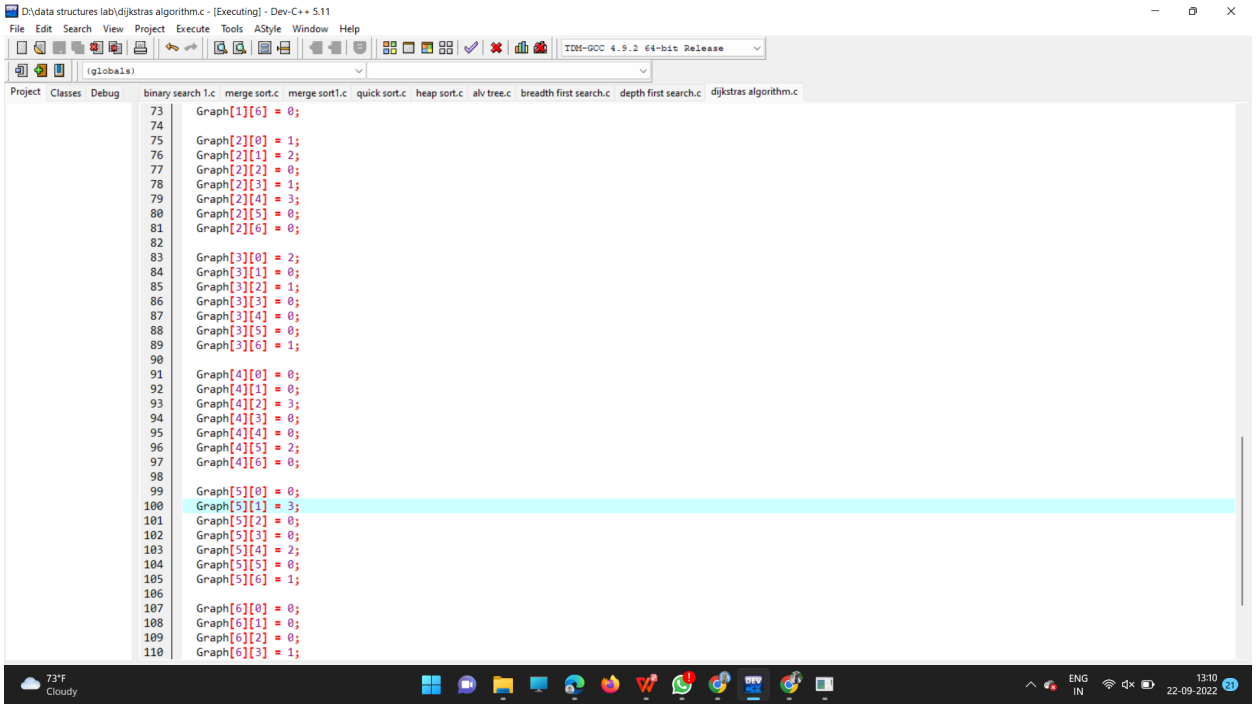
```
// 23. Implementation of Shortest Path Algorithms using Dijkstra's Algorithm
#include <stdio.h>
#define INFINITY 9999
#define MAX 10

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++)
      if (!visited[i])
        if (mindistance + cost[nextnode][i] < distance[i]) {
          distance[i] = mindistance + cost[nextnode][i];
          pred[i] = nextnode;
```

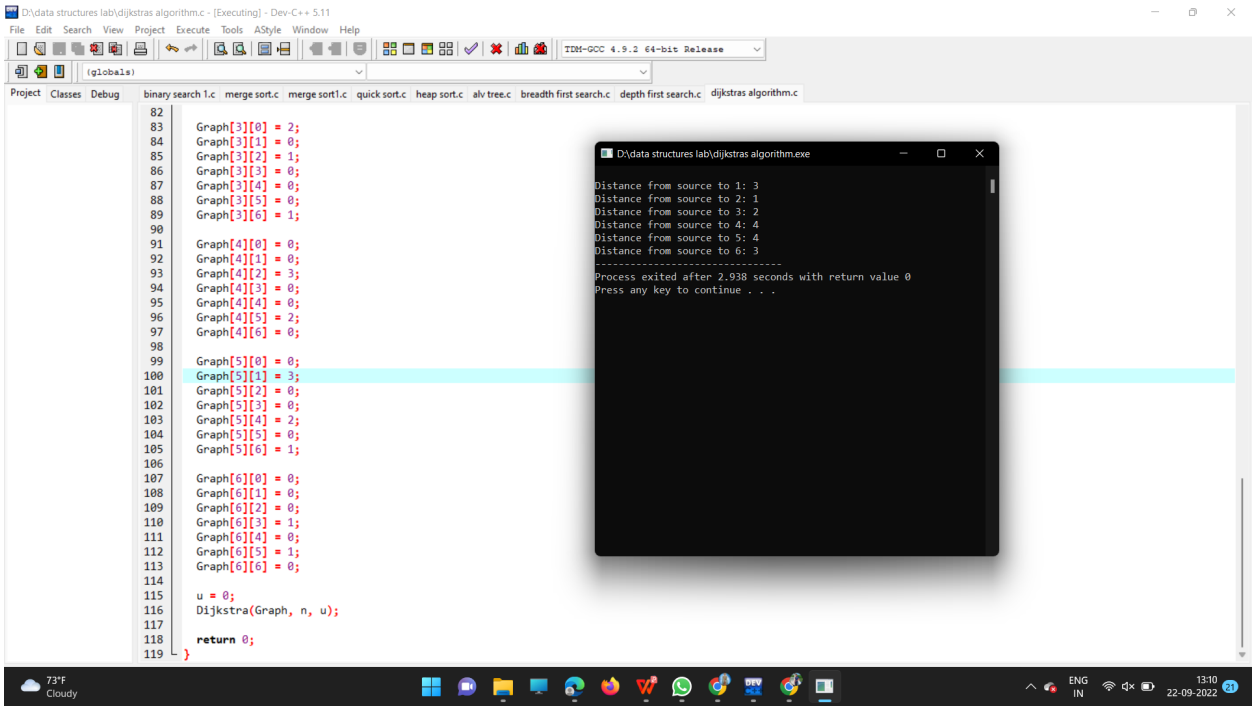
```
      }
    count++;
  }


  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;
  Graph[2][5] = 0;
  Graph[2][6] = 0;

  Graph[3][0] = 2;
  Graph[3][1] = 0;
  Graph[3][2] = 1;
  Graph[3][3] = 0;
  Graph[3][4] = 0;
  Graph[3][5] = 0;
```

```
  Graph[3][6] = 1;

  Graph[4][0] = 0;
  Graph[4][1] = 0;
  Graph[4][2] = 3;
  Graph[4][3] = 0;
  Graph[4][4] = 0;
  Graph[4][5] = 2;
  Graph[4][6] = 0;

  Graph[5][0] = 0;
  Graph[5][1] = 3;
  Graph[5][2] = 0;
  Graph[5][3] = 0;
  Graph[5][4] = 2;
  Graph[5][5] = 0;
  Graph[5][6] = 1;

  Graph[6][0] = 0;
  Graph[6][1] = 0;
  Graph[6][2] = 0;
  Graph[6][3] = 1;
  Graph[6][4] = 0;
  Graph[6][5] = 1;
  Graph[6][6] = 0;

  u = 0;
  Dijkstra(Graph, n, u);

  return 0;
}
```


D:\data structures lab\dijkstras algorithm.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
binary search 1.c merge sort.c merge sort1.c quick sort.c heap sort.c alv tree.c breadth first search.c depth first search.c dijkstras algorithm.c
// 23. Implementation of Shortest Path Algorithms using Dijkstra's Algorithm
#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;
}
73°F Cloudy
ENG IN
13:09 22-09-2022


D:\data structures lab\dijkstras algorithm.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
binary search 1.c merge sort.c merge sort1.c quick sort.c heap sort.c alv tree.c breadth first search.c depth first search.c dijkstras algorithm.c
}
visited[nextnode] = 1;
for (i = 0; i < n; i++)
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;
73°F Cloudy
ENG IN
13:09 22-09-2022


D:\data structures lab\dijkstras algorithm.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
binary search 1.c merge sort.c merge sort1.c quick sort.c heap sort.c alv tree.c breadth first search.c depth first search.c dijkstras algorithm.c
Graph[1][6] = 0;
Graph[2][0] = 1;
Graph[2][1] = 2;
Graph[2][2] = 0;
Graph[2][3] = 1;
Graph[2][4] = 3;
Graph[2][5] = 0;
Graph[2][6] = 0;
Graph[3][0] = 2;
Graph[3][1] = 0;
Graph[3][2] = 1;
Graph[3][3] = 0;
Graph[3][4] = 0;
Graph[3][5] = 0;
Graph[3][6] = 1;
Graph[4][0] = 0;
Graph[4][1] = 0;
Graph[4][2] = 3;
Graph[4][3] = 0;
Graph[4][4] = 0;
Graph[4][5] = 2;
Graph[4][6] = 0;
Graph[5][0] = 0;
Graph[5][1] = 3;
Graph[5][2] = 0;
Graph[5][3] = 0;
Graph[5][4] = 2;
Graph[5][5] = 0;
Graph[5][6] = 1;
Graph[6][0] = 0;
Graph[6][1] = 0;
Graph[6][2] = 0;
Graph[6][3] = 1;
73°F Cloudy
ENG IN
13:10 22-09-2022


D:\data structures lab\dijkstras algorithm.c - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
binary search 1.c merge sort.c merge sort1.c quick sort.c heap sort.c alv tree.c breadth first search.c depth first search.c dijkstras algorithm.c
Graph[3][0] = 2;
Graph[3][1] = 0;
Graph[3][2] = 1;
Graph[3][3] = 0;
Graph[3][4] = 0;
Graph[3][5] = 0;
Graph[3][6] = 1;
Graph[4][0] = 0;
Graph[4][1] = 0;
Graph[4][2] = 3;
Graph[4][3] = 0;
Graph[4][4] = 0;
Graph[4][5] = 2;
Graph[4][6] = 0;
Graph[5][0] = 0;
Graph[5][1] = 3;
Graph[5][2] = 0;
Graph[5][3] = 0;
Graph[5][4] = 2;
Graph[5][5] = 0;
Graph[5][6] = 1;
Graph[6][0] = 0;
Graph[6][1] = 0;
Graph[6][2] = 0;
Graph[6][3] = 1;
Graph[6][4] = 0;
Graph[6][5] = 1;
Graph[6][6] = 0;
u = 0;
Dijkstra(Graph, n, u);
return 0;
}
D:\data structures lab\dijkstras algorithm.exe
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
Process exited after 2.938 seconds with return value 0
Press any key to continue . . .
73°F Cloudy
ENG IN
13:10 22-09-2022