CSE208L Object Oriented Programming Lab

LAB # 8

2021

Submitted to:

Engr. Sumayyea Salahuddin

Submitted by:

TAYYABA

Registration No :

19PWCSE1854

Semester: 3rd

Class Section: C

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

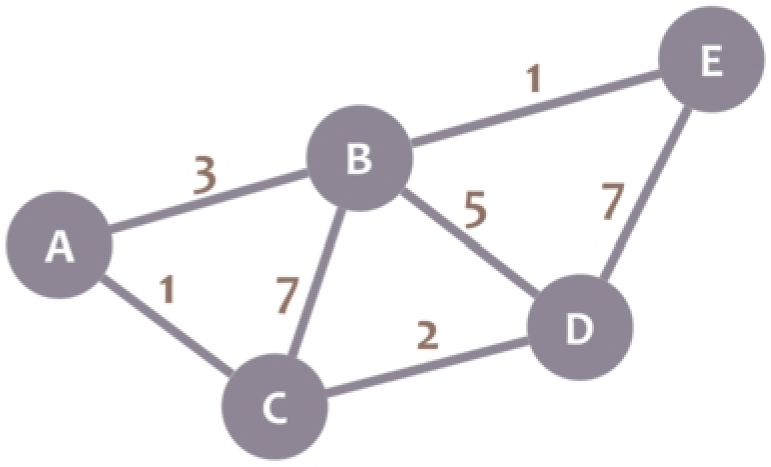
January 14, 2021

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

Algorithm
Minimum Distance (MD)

Dijkstra's Algorithm allows you to calculate the shortest path between one node (you pick which one) and *every other node in the graph*. You'll find a description of the algorithm at the end of this page, but, let's study the algorithm with an explained example! Let's calculate the shortest path between node C and the other nodes in our graph



During the algorithm execution, we'll mark every node with its minimum distance to node C (our selected node). For node C, this distance is 0. For the rest of nodes, as we still don't know that minimum distance, it starts being infinity (∞):

We'll also have a current node. Initially, we set it to C (our selected node).

Now, we check the neighbours of our current node (A, B and D) in no specific order. Let's begin with B. We add the minimum distance of the current node (in this case, 0) with the weight of the edge that connects our current node with B (in this case, 7), and we obtain 0 + 7 = 7. We compare that value with the minimum distance of B (infinity); the lowest value is the one that remains as the minimum distance of B (in this case, 7 is less than infinity):

So far, so good. Now, let's check neighbour A. We add 0 (the minimum distance of C, our current node) with 1 (the weight of the edge connecting our current node with A) to obtain 1. We compare that 1 with the minimum distance of A (infinity), and leave the smallest value

OK. Repeat the same procedure for D:

We now need to pick a new *current node*. That node must be the unvisited node with the smallest minimum distance (so, the node with the smallest number and no check mark). That's A.

And now we repeat the algorithm. We check the neighbours of our current node, ignoring the visited nodes. This means we only check B.

For B, we add 1 (the minimum distance of A, our current node) with 3 (the weight of the edge connecting A and B) to obtain 4. We compare that 4 with the minimum distance of B (7) and leave the smallest value: 4.

Afterwards, we mark A as visited and pick a new current node: D, which is the non-visited node with the smallest current distance.

We repeat the algorithm again. This time, we check B and E.

	A	B	C	D	E
c{C}	**infinity**	**infinity**	**[0]**	**infinity**	**infinity**
C{A}=0+1=1	**[1]**	**7**		**2**	**infinity**
C{B}=1+3=4		**[4]**		**2**	**infinity**
C{D}=0+2=2				**[2]**	**infinity**

We got minimum distances from point C to all other points

From C to C = 0

From C to A = 1

From C to B = CAB(line)

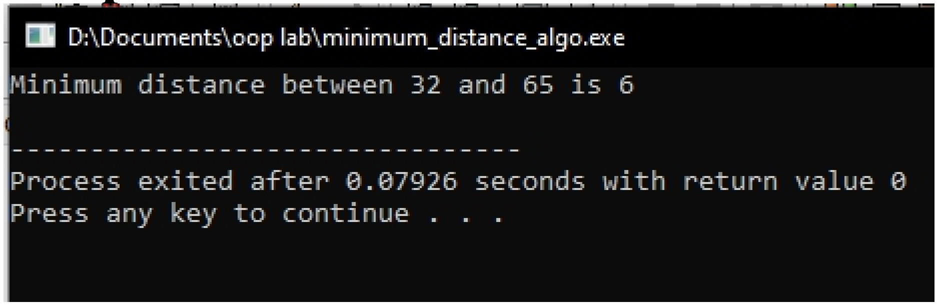
From C to D= 2

Code:

```
#include<iostream>

#include <cstdlib>

class Distance {

  private:

        int * arr, n, x, y;

  public:

        Distance(int * arr, int n, int x, int y) {

              this->arr = arr;

              this->x = x;

              this->n = n;

              this->y = y;

        }

        int minDist()

{

        int i, j;

        int min_dist = INT_MAX;

        for (i = 0; i < n; i++)

        {

                for (j = i+1; j < n; j++)

                {

                        if( (x == *(arr+i) && y == *(arr+j) ||

                                y == *(arr+i) && x == *(arr+j)) &&

                                min_dist > abs(i-j))

                        {

                                min_dist = abs(i-j);
```

```
                    }
                }
            }
            return min_dist;
        }
};
using namespace std;
int main()
{
            int arr[] = {3, 5, 4, 2, 6, 65, 12, 6, 5, 14, 8, 32};
            int n = sizeof(arr)/sizeof(arr[0]);
            int x = 32;
            int y = 65;
            Distance obj(arr, n, x, y);

            cout << "Minimum distance between " << x <<
                                        " and " << y << " is " <<
                                       obj.minDist() << endl;
                                        return 0;
}
```

Output:



```
D:\Documents\oop lab\minimum_distance_algo.exe
Minimum distance between 32 and 65 is 6

--------------------------------
Process exited after 0.07926 seconds with return value 0
Press any key to continue . . .
```