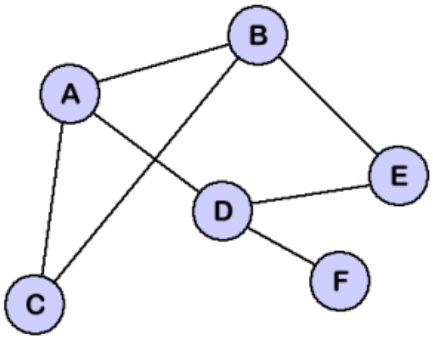
BCSE202P -
Data Structures and Algorithms

Digital Assignment – 4

Name: Dhruv Rajeshkumar Shah
Registration No – 21BCE0611

1. Construct the given graph using Adjacency Matrix and Perform breadth first search on it.

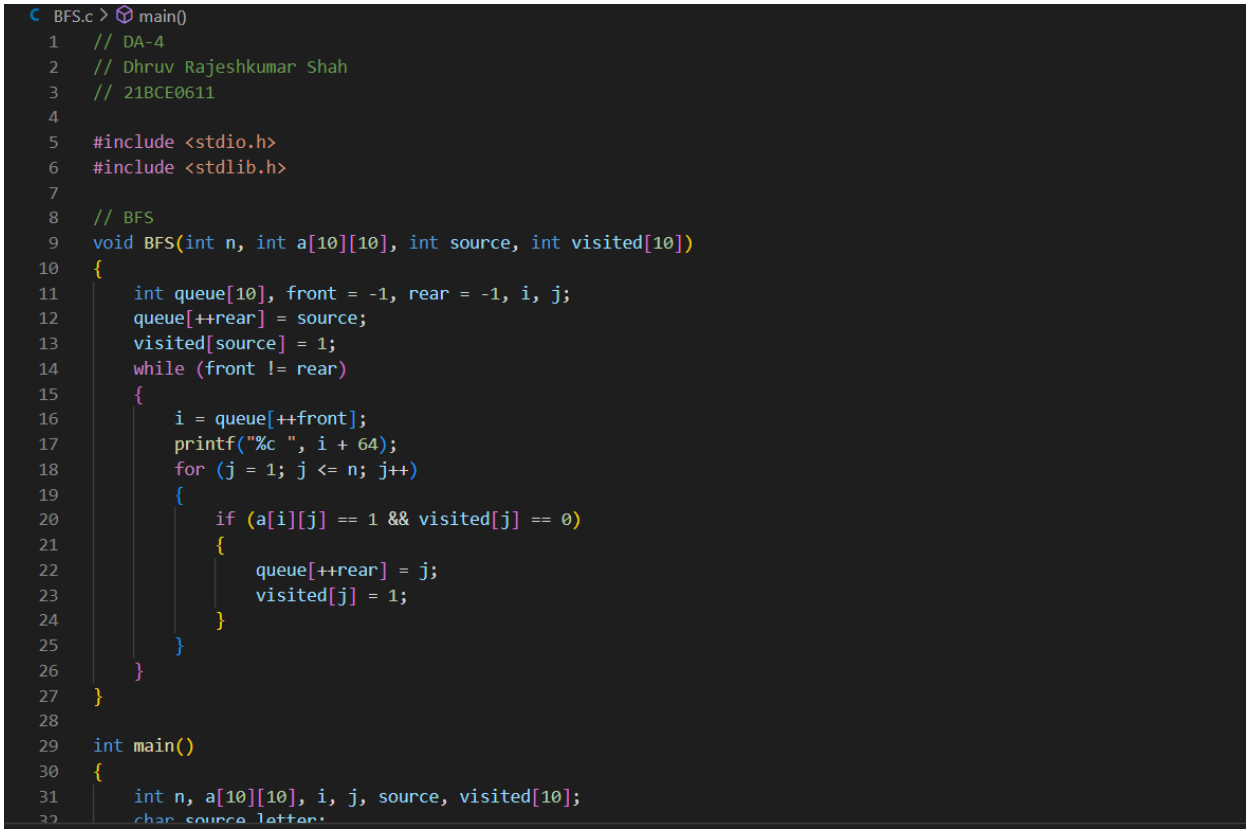


CODE

```c
// DA-4
// Dhruv Rajeshkumar Shah
// 21BCE0611

#include <stdio.h>
#include <stdlib.h>

// BFS
void BFS(int n, int a[10][10], int source, int visited[10])
{
    int queue[10], front = -1, rear = -1, i, j;
    queue[++rear] = source;
    visited[source] = 1;
    while (front != rear)
    {
        i = queue[++front];
        printf("%c ", i + 64);
        for (j = 1; j <= n; j++)
        {
            if (a[i][j] == 1 && visited[j] == 0)
            {
                queue[++rear] = j;
                visited[j] = 1;
            }
        }
    }
}

int main()
{
    int n, a[10][10], i, j, source, visited[10];
    char source_letter;
    printf("Enter the number of vertices: ");
```

```
    scanf("%d", &n);
    printf("Enter the adjacency matrix: \n");
    printf("    A B C D E F\n");
    printf("   -------------\n");
    for (i = 1; i <= n; i++)
    {
        printf("%c | ", i + 64);
        for (j = 1; j <= n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    getchar();
    printf("Enter the source vertex: ");
    scanf("%c", &source_letter);
    source = (int)source_letter - 64;
    for (i = 1; i <= n; i++)
    {
        visited[i] = 0;
    }
    printf("The BFS traversal is: ");
    BFS(n, a, source, visited);
    return 0;
}
```

SCREENSHOT



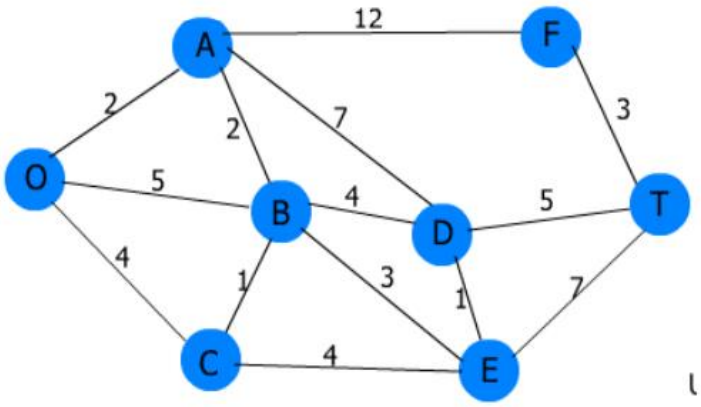
BFS.c > main()

```
1   // DA-4
2   // Dhruv Rajeshkumar Shah
3   // 21BCE0611
4
5   #include <stdio.h>
6   #include <stdlib.h>
7
8   // BFS
9   void BFS(int n, int a[10][10], int source, int visited[10])
10  {
11      int queue[10], front = -1, rear = -1, i, j;
12      queue[++rear] = source;
13      visited[source] = 1;
14      while (front != rear)
15      {
16          i = queue[++front];
17          printf("%c ", i + 64);
18          for (j = 1; j <= n; j++)
19          {
20              if (a[i][j] == 1 && visited[j] == 0)
21              {
22                  queue[++rear] = j;
23                  visited[j] = 1;
24              }
25          }
26      }
27  }
28
29  int main()
30  {
31      int n, a[10][10], i, j, source, visited[10];
32      char source_letter;
```

OUTPUT

```
dhruv@Titan /c/Dhruv/VIT/Semester-3/DSA/Lab/DA-4 (main)
$ ./BFS.exe
Enter the number of vertices: 6
Enter the adjacency matrix:
Enter row 1: 0 1 1 1 0 0
Enter row 2: 1 0 1 0 1 0
Enter row 3: 1 1 0 0 0 0
Enter row 4: 1 0 0 0 1 1
Enter row 5: 0 1 0 1 0 0
Enter row 6: 0 0 0 1 0 0
Enter the source vertex: A
The BFS traversal is: A B C D E F
dhruv@Titan /c/Dhruv/VIT/Semester-3/DSA/Lab/DA-4 (main)
$ ./BFS.exe
Enter the number of vertices: 6
Enter the adjacency matrix:
Enter row 1: 0 1 1 1 0 0
Enter row 2: 1 0 1 0 1 0
Enter row 3: 1 1 0 0 0 0
Enter row 4: 1 0 0 0 1 1
Enter row 5: 0 1 0 1 0 0
Enter row 6: 0 0 0 1 0 0
Enter the source vertex: B
The BFS traversal is: B A C E D F
dhruv@Titan /c/Dhruv/VIT/Semester-3/DSA/Lab/DA-4 (main)
$ ./BFS.exe
Enter the number of vertices: 6
Enter the adjacency matrix:
Enter row 1: 0 1 1 1 0 0
Enter row 2: 1 0 1 0 1 0
Enter row 3: 1 1 0 0 0 0
Enter row 4: 1 0 0 0 1 1
Enter row 5: 0 1 0 1 0 0
Enter row 6: 0 0 0 1 0 0
Enter the source vertex: C
The BFS traversal is: C A B D E F
```

4. Write a program that creates the following graph and finds the shortest path from the vertex O to all the other vertices using Dijikstra's Algorithm.



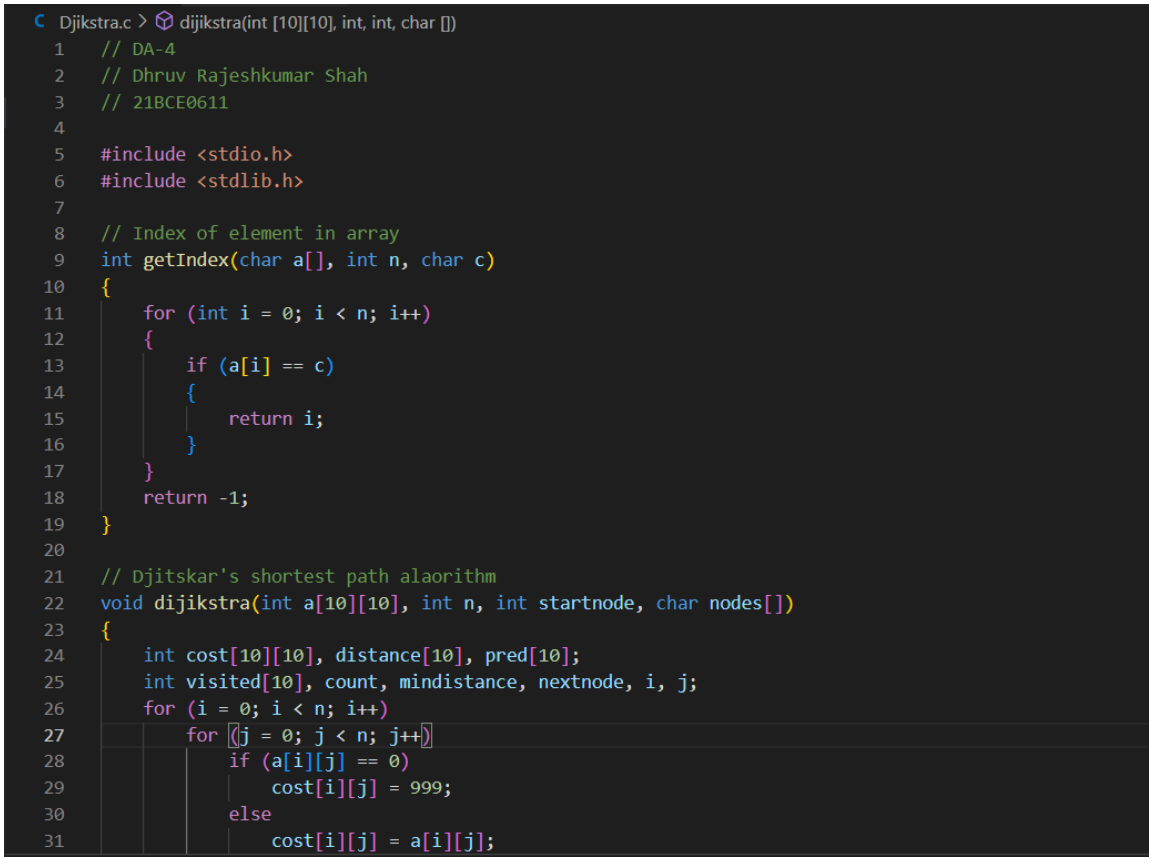
CODE

```
// DA-4
// Dhruv Rajeshkumar Shah
// 21BCE0611

#include <stdio.h>
#include <stdlib.h>

// Index of element in array
int getIndex(char a[], int n, char c)
{
    for (int i = 0; i < n; i++)
    {
        if (a[i] == c)
        {
            return i;
        }
    }
    return -1;
}

// Djitskar's shortest path alaorithm
void dijikstra(int a[10][10], int n, int startnode, char nodes[])
{
    int cost[10][10], distance[10], pred[10];
    int visited[10], count, mindistance, nextnode, i, j;
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            if (a[i][j] == 0)
```

```
                cost[i][j] = 999;
            else
                cost[i][j] = a[i][j];
    for (i = 0; i < n; i++)
    {
        distance[i] = cost[startnode][i];
        pred[i] = startnode;
        visited[i] = 0;
    }
    distance[startnode] = 0;
    visited[startnode] = 1;
    count = 1;
    while (count < n - 1)
    {
        mindistance = 999;
        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 != startnode)
        {
            printf("\nDistance of node %c = %d", nodes[i], distance[i]);
            printf("\nPath = %c", nodes[i]);
            j = i;
            do
            {
                j = pred[j];
                printf("<-%c", nodes[j]);
            } while (j != startnode);
        }
        printf("\n");
    }
}
```

```c
void main()
{
    int a[10][10], j, n, startnode;
    char source;
    char nodes[] = {'A', 'B', 'C', 'D', 'E', 'F', 'O', 'T'};
    printf("\nEnter the number of vertices: ");
    scanf("%d", &n);
    printf("\nEnter the adjacency matrix:\n");
    printf("    A B C D E F O T\n");
    printf("   ----------------\n");
    for (int i = 0; i < n; i++)
    {
        printf("%c | ", nodes[i]);
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
    getchar();
    printf("\nEnter the source: ");
    scanf("%c", &source);
    startnode = getIndex(nodes, n, source);
    dijikstra(a, n, startnode, nodes);
}
```

SCREENSHOT



OUTPUT

```
dhruv@Titan /c/Dhruv/VIT/Semester-3/DSA/Lab/DA-4 (main)
$ ./Djikstra.exe
Enter the number of vertices: 8

Enter the adjacency matrix:
    A B C D E F O T
   ----------------
A | 0 2 0 7 0 12 2 0
B | 2 0 1 4 3 0 5 0
C | 0 1 0 0 4 0 4 0
D | 7 4 0 0 1 0 0 5
E | 0 3 4 1 0 0 0 7
F | 12 0 0 0 0 0 0 3
O | 2 5 4 0 0 0 0 0
T | 0 0 0 5 7 3 0 0

Enter the source: O

Distance of node A = 2
Path = A<-O

Distance of node B = 4
Path = B<-A<-O

Distance of node C = 4
Path = C<-O

Distance of node D = 8
Path = D<-B<-A<-O

Distance of node E = 7
Path = E<-B<-A<-O

Distance of node F = 14
Path = F<-A<-O


Distance of node T = 13
Path = T<-D<-B<-A<-O
```