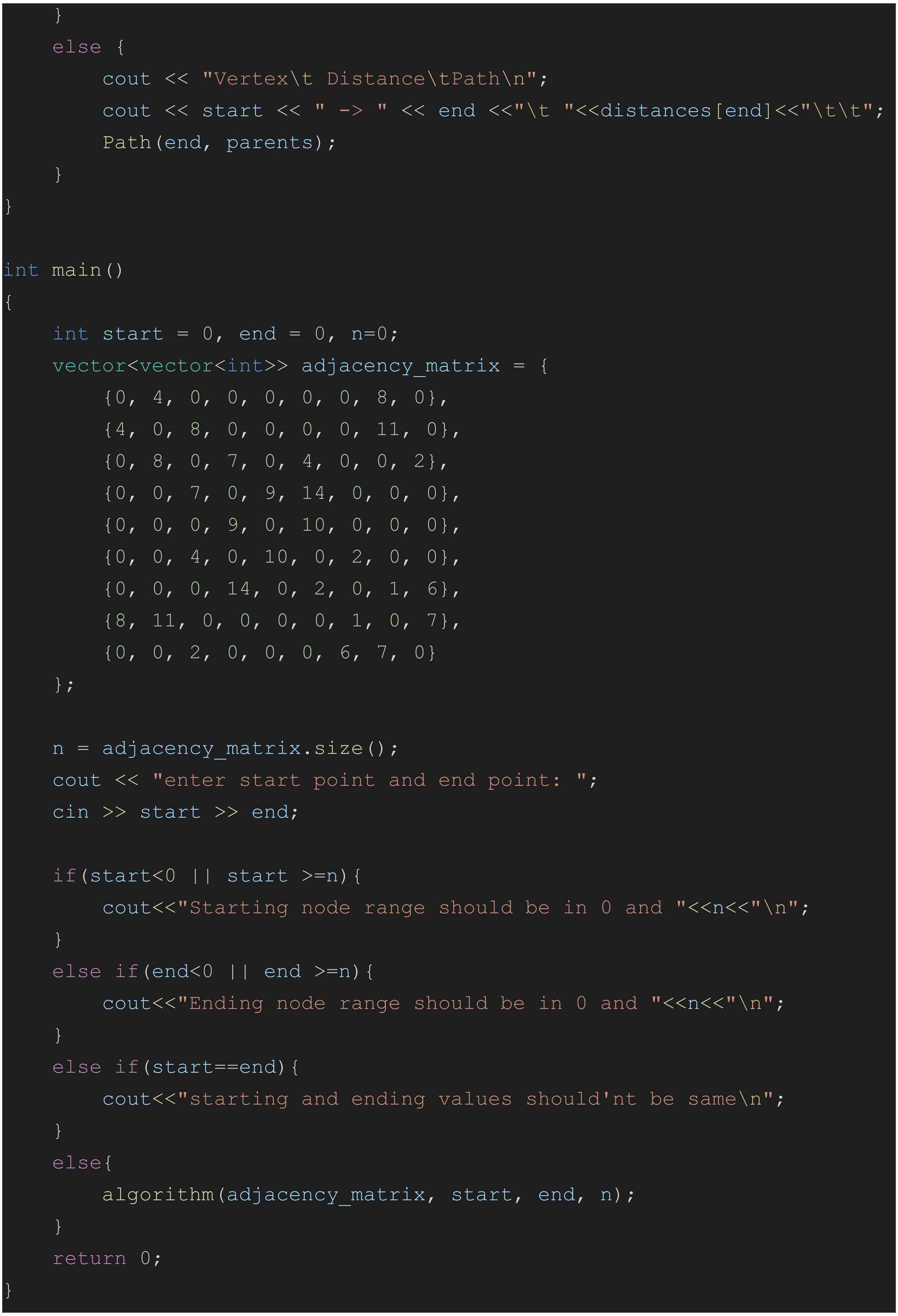
Assignment 3(Dijkstra's):

Cpp code

```cpp
#include <bits/stdc++.h>
using namespace std;

void Path(int current_pointer, vector<int>& parents) {
    if (current_pointer == -1) {
        return;
    }
    Path(parents[current_pointer], parents);
    cout << current_pointer << " ";
}

void algorithm(vector<vector<int>>& matrix, int start, int end, int n)
{
    vector<int> distances(n, INT_MAX);
    vector<int> parents(n, -1);
    vector<bool> visited(n, false);

    distances[start] = 0;
    for (int i = 0; i < n - 1; i++) {
        int u = -1;
        for (int j = 0; j < n; j++) {
            if (!visited[j] && (u == -1 || distances[j] <
distances[u])) {
                u = j;
            }
        }

        visited[u] = true;
        for (int v = 0; v < n; v++) {
            int weight = matrix[u][v];
            if (weight != 0 && distances[u] + weight < distances[v]) {
                distances[v] = distances[u] + weight;
                parents[v] = u;
            }
        }
    }

    if (distances[end] == INT_MAX) {
        cout << "No path found between " << start << " and " << end <<
"\n";
```



```
    }
    else {
        cout << "Vertex\t Distance\tPath\n";
        cout << start << " -> " << end <<"\t "<<distances[end]<<"\t\t";
        Path(end, parents);
    }
}

int main()
{
    int start = 0, end = 0, n=0;
    vector<vector<int>> adjacency_matrix = {
        {0, 4, 0, 0, 0, 0, 0, 8, 0},
        {4, 0, 8, 0, 0, 0, 0, 11, 0},
        {0, 8, 0, 7, 0, 4, 0, 0, 2},
        {0, 0, 7, 0, 9, 14, 0, 0, 0},
        {0, 0, 0, 9, 0, 10, 0, 0, 0},
        {0, 0, 4, 0, 10, 0, 2, 0, 0},
        {0, 0, 0, 14, 0, 2, 0, 1, 6},
        {8, 11, 0, 0, 0, 0, 1, 0, 7},
        {0, 0, 2, 0, 0, 0, 6, 7, 0}
    };

    n = adjacency_matrix.size();
    cout << "enter start point and end point: ";
    cin >> start >> end;

    if(start<0 || start >=n){
        cout<<"Starting node range should be in 0 and "<<n<<"\n";
    }
    else if(end<0 || end >=n){
        cout<<"Ending node range should be in 0 and "<<n<<"\n";
    }
    else if(start==end){
        cout<<"starting and ending values should'nt be same\n";
    }
    else{
        algorithm(adjacency_matrix, start, end, n);
    }
    return 0;
}
```

- Test cases passed
- Completed on 28/3/23

Q/A:

1. How long did you spend on this assignment?
   a. 1day
2. Based on your effort, what letter grade would you say you earned?
   a. On a scale of 1 to 10. I would grade this as 10/10.
3. Based on your solution, what letter grade would you say you earned?
   a. On a scale of 1 to 10. I would grade this as 9/10.
4. Provide a summary of what doesn't work in your solution, along with an explanation of how you attempted to solve the problem and where you feel you struggled?
   a. The program takes input as an adjacency matrix that represents a graph and start and end nodes for which the shortest path is to be calculated.

   b. The program optimises the printing of the shortest path between start and end nodes by using a recursive function that backtracks from the end node to the start node along the path with minimum distance