Assignment 3 (Dijkstr's algorithm part A):

Cpp code

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

int Total_v = -1;

void print_paths(int index, vector<int> paths)
{
   char c=65+index;
        if (index == Total_v) {
                return;
        }
        print_paths(paths[index], paths);
        cout << c << "->";
}

void catch_sol(int first, vector<int> arr_dist, vector<int> paths)
{
        int n = arr_dist.size();
        for (int i = 0; i < n; i++) {
                if (i != first) {
                    char c=65+first, e=65+i;
                        cout << "\n" << c << " -> ";
                        cout << e << "  \t  ";
                        cout << arr_dist[i] << " \t\t ";
                        print_paths(i, paths);
                }
        }
}

int main()
{
        vector<vector<int> > arr = {
      {0, 10, 0, 0, 3},
      {0, 0, 2, 0, 4},
      {0, 0, 0, 9, 0},
      {0, 0, 7, 0, 0},
      {0, 1, 8, 2, 0}};

        int n = arr[0].size(), first = 0;
        vector<int> arr_dist(n);
        vector<bool> arr_add(n);
        vector<int> paths(n);

        for (int i = 0; i < n; i++) {
                arr_dist[i] = INT_MAX;
                arr_add[i] = false;
        }

        arr_dist[first] = 0;
        paths[first] = Total_v;
```

```
        for (int i = 1; i < n; i++) {
                int v = -1;
                int dist = INT_MAX;
                for (int j = 0; j < n; j++) {
                        if (!arr_add[j] && arr_dist[j] < dist) {
                                v = j;
                                dist = arr_dist[j];
                        }
                }
                arr_add[v] = true;
                for (int i = 0; i < n; i++) {
                        int e_dist = arr[v][i];
                        if (e_dist > 0 && ((dist + e_dist) < arr_dist[i])) {
                                paths[i] = v;
                                arr_dist[i] = dist + e_dist;
                        }
                }
        }
        cout << "Vertex\t||\tDistance\t|| Path"<<endl;
        catch_sol(first, arr_dist, paths);
        return 0;
}
```

- Test cases passed
- Completed on 12/10/22

Q/A:

1. How long did you spend on this assignment?
    a. 3hr
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. While printing the whole path from source to destination.
    b. At first I tried a naive approach by storing a dictionary of paths from source to all vertices with their minimum distance, later I tried to store the distance of one vertex by using previous vertex distance.