

ASSIGNMENT #4 – Dynamic Programming-Floyd Algorithm

C Code of Floyd–Warshall Algorithm

```
#include <iostream>
#include <vector>
using namespace std;
void showvectortable(vector < vector<int> > floydvector) {
    for (int i = 0; i < floydvector.size(); i++) {
        for (int j = 0; j < floydvector[i].size(); j++) { // D table
            if (floydvector[i][j] == 100000) // checks infinity
                cout << "#" << "\t"; // # for infinity
            else
                cout << floydvector[i][j] << "\t";
        }
        cout << "\n\n";
    }
}
int min(int x, int y) {
    if (x < y)
        return x;
    return y;
}
void floydAlgorithms(int node, vector < vector <int> > connection) {
    vector< vector<int> > d0(node, vector<int>(node)),
        d1(node, vector<int>(node)),
        b0(node, vector<int>(node)),
        b1(node, vector<int>(node));
    for (int i = 0; i < node; i++) { // D and B table
        for (int j = 0; j < node; j++) {
            if (i != j)
                d0[i][j] = connection[i][j]; // Create D table
            else
                d0[i][j] = 100000; // Infinity Value
            b0[i][j] = j + 1; // B table
        }
    }
    for (int r = 0; r < node; r++) {
        cout << "\t D" << r << " TABLE:\n\n";
        showvectortable(d0);
        cout << "\n\t B" << r << " TABLE:\n\n";
        showvectortable(b0);
        for (int i = 0; i < node; i++) {
            for (int j = 0; j < node; j++) {
                if (d0[i][j] <= d0[i][r] + d0[r][j]) //
                    b1[i][j] = b0[i][j];
                else
                    b1[i][j] = b0[i][r]; // B1
                if (r == i || r == j) {
                    d1[i][j] = d0[i][j];
                    continue; // next iter
                }
                d1[i][j] = min(d0[i][j], d0[i][r] +
d0[r][j]); // Floyd Algorithms Formula
            }
        }
        cout << "\n\n\n";
    }
}
```

```

        d0 = d1;
        b0 = b1;
    }
    int target = 4;
    int start = 1;
    int i = 0;
    cout << "PATH = "<< start << " - ";
    while (true) {
        if (b0[i][target - 1] == target) {
            cout << b0[i][target - 1];
            break;
        }
        cout << b0[i][target - 1] << " - ";
        i = b0[i][target - 1] - 1;
    }
}

int main()
{
    int length;
    cout << "Enter the Number of Nodes " << "\n";
    cin >> length;
    int nodes;
    cout << "Enter the Length of Nodes " << "\n";
    cin >> nodes;
    // {100000, 5, 1, 100000},{2, 100000, 100000, 1},{5, 2, 100000, 4},
    // {100000, 100000, 2, 100000}
    vector<vector<int>> connection;
    for(int i=0;i<nodes;++i) {
        cout << "Enter the " << i+1 << ". Vector values " << "\n";
        vector<int> row;
        for(int j=0;j<length;++j){
            int value;
            cout << j << ". index = ";
            cin >> value;
            row.push_back(value);
        }
        //Push the row in matrix
        connection.push_back(row);
    }

    cout << "\n\n";
    floydAlgorithims(nodes, connection);
}

```

Output of Program

```
CA: LutfiArda-Ge... - □ ×
Enter the Number of Nodes
4
Enter the Length of Nodes
4

Enter the 1. Vector values
0. index = 100000
1. index = 5
2. index = 1
3. index = 100000

Enter the 2. Vector values
0. index = 2
1. index = 100000
2. index = 100000
3. index = 1

Enter the 3. Vector values
0. index = 5
1. index = 2
2. index = 100000
3. index = 4

Enter the 4. Vector values
0. index = 100000
1. index = 100000
2. index = 2
3. index = 100000
```

#1

D0 TABLE:			
#	5	1	#
2	#	#	1
5	2	#	4
#	#	2	#

B0 TABLE:			
1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

#2

D1 TABLE:			
#	5	1	#
2	7	3	1
5	2	6	4
#	#	2	#

B1 TABLE:			
1	2	3	4
1	1	1	4
1	2	1	4
1	2	3	4

#3

D2 TABLE:			
7	5	1	6
2	7	3	1
4	2	5	3
#	#	2	#

B2 TABLE:			
2	2	3	2
1	1	1	4
2	2	2	2
1	2	3	4

#4

D3 TABLE:			
5	3	1	4
2	5	3	1
4	2	5	3
6	4	2	5

B3 TABLE:			
3	3	3	3
1	1	1	4
2	2	2	2
3	3	3	3

PATH = 1 - 3 - 2 - 4
Press any key to continue . .

#5