**ساختار کلی برنامه**

برای پیاده سازی این پروژه از دو کلاس به نام های Network و Graph استفاده کردیم. کلاس Network دارای یک فیلد از نوع کلاس Graph است که با دریافت دستور از کنسول، متد مناسب از کلاس Graph را Call می کند.

class Network

{

private:

    Graph\* graph;

public:

    void run();

};

کلاس Graph به نوعی هسته ی اصلی برنامه بوده به طوری که راس ها را نگهداری کرده و الگوریتم ها را اجرا می نماید.

class Graph

{

private:

    set<int> nodes;

    map<pair<int, int>, int> weight;

public:

    Graph(string topology);

    void add\_edge(int v, int u, int w);

    void show();

    void delete\_edge(int v, int u);

    void modify\_edge(int v, int u, int cost);

    void link\_state(int source);

    void distance\_vector(int source);

};

همچنین تعدادی تابع خارج از این دو کلاس به عنوان توابع کمکی (که در اجرا و چاپ نتیجه به ما کمک می کنند) تعریف شده اند.

**الگوریتم Link State (LSRP)**

کد این الگوریتم به صورت زیر است:

    mark[source] = true;

    distance[source] = 0;

    int size = 1;

    while (size < number\_of\_nodes)

    {

        int mn = INF, source;

        for (auto node: nodes)

        {

            if (mark[node])

                continue;

            if (distance[node] < mn)

            {

                mn = distance[node];

                source = node;

            }

        }

        cout << "   |Iter " + to\_string(size) << ":" << endl;

        cout << "Dest|";

        for (auto node: nodes)

        {

            int column = 4 - get\_digits\_count(node);

            for (int i = 0; i < column; i++)

                cout << " ";

            cout << node << "|";

        }

        cout << endl;

        cout << "Cost|";

        for (auto node: nodes)

        {

            int column = 4 - get\_digits\_count(distance[node]);

            if (distance[node] == INF)

                column = 2;

            for (int i = 0; i < column; i++)

                cout << " ";

            cout << (distance[node] == INF ? -1 : distance[node]) << "|";

        }

        cout << endl;

        for (int i = 0; i < (nodes.size() + 1) \* 5; i++)

            cout << "-";

        cout << endl;

        mark[source] = true;

        size += 1;

        for (auto node: nodes)

        {

            if (mark[node])

                continue;

            if (weight.find(make\_pair(source, node)) == weight.end())

                continue;

            if (distance[source] + weight[make\_pair(source, node)] < distance[node])

            {

                distance[node] = distance[source] + weight[make\_pair(source, node)];

                parent[node] = source;

            }

        }

    }

**الگوریتم Distance Vector (DVRP)**

ین الگوریتم مشابه الگوریتم Ford-Bellman عمل میکند. آنقدر روی یالها عمل relaxation انجام میدهد، تا موقعی که تغییر جدیدی در فاصله رخ ندهد. کد این الگوریتم به صورت زیر است:

    vector<bool> mark(number\_of\_nodes + 1, false);

    vector<int> distance(number\_of\_nodes + 1, INF);

    vector<int> parent(number\_of\_nodes + 1, -1);

    distance[source] = 0;

    while (1)

    {

        bool updated = false;

        for (auto it = weight.begin(); it != weight.end(); it++)

        {

            int source = it->first.first, destination = it->first.second, w = it->second;

            if (distance[source] + w < distance[destination])

            {

                distance[destination] = distance[source] + w;

                parent[destination] = source;

                updated = true;

            }

        }

        if (!updated)

            break;

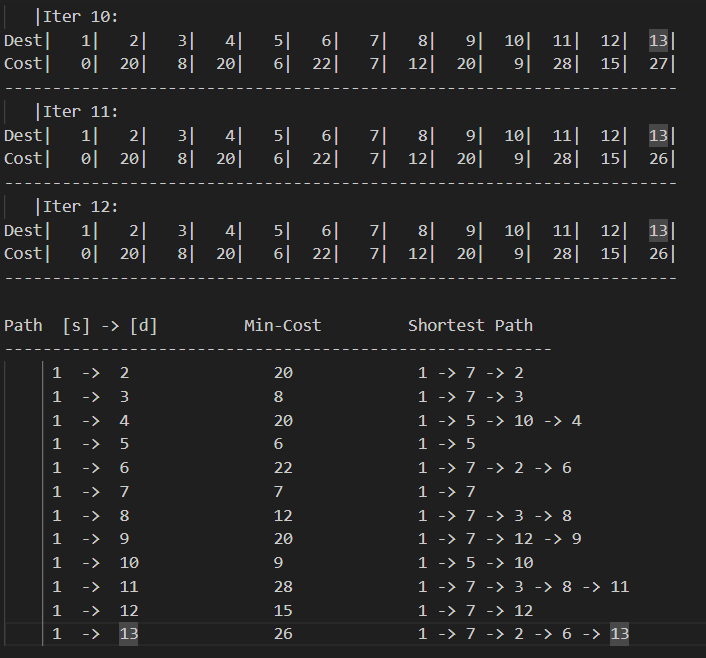
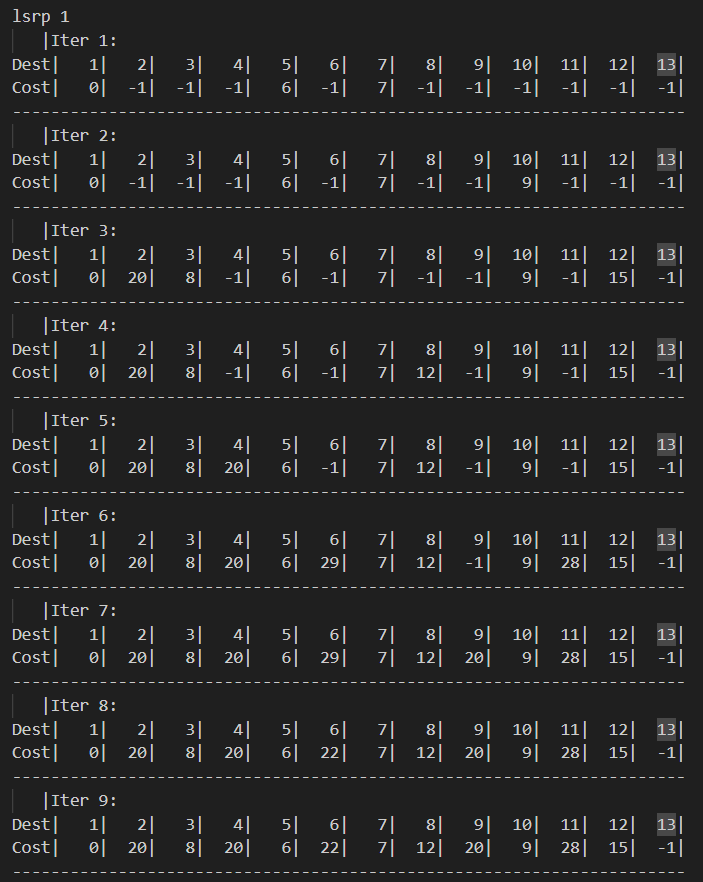
    }

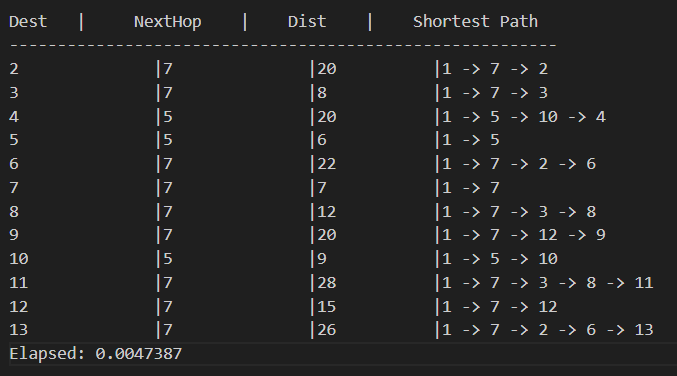
**نتایج**

با توجه به گراف داده شده در صورت پروژه، ورودی اولیه جهت ساخت گراف به صورت زیر است:

topology 1-5-6 1-7-7 2-6-2 2-7-13 3-7-1 3-8-4 3-12-8 4-6-19 4-10-11 5-10-3 6-8-17 6-11-25 6-13-4 7-12-8 8-11-16 9-12-5 9-13-7 10-12-12

**خروجی الگوریتم link state به ازای راس مبدا 1**



**خروجی الگوریتم Distance Vector به ازای راس مبدا 1**

**مقایه زمان اجرا (زمان ها به میلی ثانیه درج شده اند.)**

|  |  |  |
| --- | --- | --- |
| Link State | Distance Vector | Source |
| **24.014** | **4.722** | 1 |
| **25.387** | **5.012** | 2 |
| **22.772** | **5.430** | 3 |
| **23.866** | **5.323** | 4 |
| **22.473** | **5.716** | 5 |
| **23.369** | **5.584** | 6 |
| **29.591** | **5.568** | 7 |
| **24.504** | **5.108** | 8 |
| **22.280** | **4.947** | 9 |
| **22.537** | **5.125** | 10 |
| **23.875** | **5.722** | 11 |
| **23.031** | **4.782** | 12 |
| **23.076** | **5.577** | 13 |

**پیش از حذف یال 10-4 :**

**پس از حذف یال 10-4:**

|  |  |  |
| --- | --- | --- |
| Link State | Distance Vector | Source |
| **25.028** | **4.922** | 1 |
| **23.339** | **5.546** | 2 |
| **37.084** | **5.181** | 3 |
| **21.815** | **3.828** | 4 |
| **53.405** | **5.711** | 5 |
| **20.845** | **5.585** | 6 |
| **14.583** | **5.810** | 7 |
| **14.063** | **5.777** | 8 |
| **15.541** | **5.162** | 9 |
| **14.998** | **6.388** | 10 |
| **15.538** | **5.526** | 11 |
| **47.921** | **17.739** | 12 |
| **17.591** | **24.723** | 13 |