COMPARISON OF SHORTEST PATH ALGORITHMS USING C#

Swati Vishnoi[1], Hina Hashmi[2]

[1]CCSIT, TMU, Moradabad
[2]CCSIT, TMU, Moradabad

Abstract

Many applications like transportation and communication network use shortest path algorithm to find out the shortest path between two or more nodes. In the Single source shortest path algorithm, a shortest path is calculated from one node to another node. In this paper, I have compared the results of the shortest path algorithms (Dijkstra, Bellman Ford) on the basis of running time. I used C# programming language to compare the algorithms. I compared the algorithms on the basis of complexity and space. I also tried to give some advantages and disadvantages of both the algorithms.

Keywords— *Shortest Path, Dijkstra, Bellman Ford, Run-time Analysis*

----------***----------

1. INTRODUCTION

Shortest Path Problem is the problem to find out the shortest path between two nodes. There are so many shortest path algorithms depending on the source and destination.

a) Single source Shortest Path
b) Single destination Shortest Path
c) All pair Shortest path Algorithm

In Single source shortest path algorithm, there is a source and we have to find the shortest path from this source to all the vertices. In single destination shortest path algorithm, there is a destination node and we have to find the shortest path from all nodes to a single node. In All pair shortest path algorithm, we have to find out the shortest path from all nodes to another node. We need efficient shortest path algorithm to run in an efficient time and take less memory to store the temporary results. In this paper, I am comparing single source shortest path algorithms (Dijkstra's and Bellman Ford).

2. LITERATURE REVIEW

As mentioned earlier, In Graph, vertices are representing the cities and edges are representing the routes from one city to another. A graph representation is explained further, and implementations of the shortest path algorithms being studied are presented.

3. WORKING OF DIJKSTRA'S AND BELLMAN FORD ALGORITHM

The working of Dijkstra's algorithm and bellman ford algorithm is as follows:

3.1 Dijkstra's Algorithm

The algorithm stores all nodes in a queue and the distance of the node from the root-First set the distance of source to zero and the distance of all the vertices except source is set to infinity. Select the least distance node from the queue and calculate the distance of all unprocessed adjacent nodes. This means that the algorithm checks for the following condition: [3]

distance +edgeweight <distance

3.2 Bellman Ford Algorithm

The Bellman-Ford algorithm is the relaxation operation algorithm. This procedure calculate the distance from the list of nodes to the adjacent nodes by checking the condition i.e. the distance is added to the edge length is less than distance to the adjacent node.

4. COMPARISON ON THE BASIS OF COMPLEXITY AND SPACE

We consider a graph[G] with the vertices or nodes [V] and the edges[E].Now If we find the complexity of Dijkstra's Algorithm and the Bellman Ford, we find that the complexity in terms of time is O(E+V(log V)) and in terms of space is O(V) for Dijkstra's Algorithm. And for Bellman Ford it is O(EV) and O(V).

4.1 Advantages and Disadvantages

4.1.1 Dijkstra's Algorithm

1. It is a Greedy Algorithm.
2. It doesn't work on negative weight.
3. It can work for directed and undirected graph.
4. It requires global information.

4.1.2 Bellman Ford Algorithm

1. It is a dynamic Algorithm.
2. It can work on negative weight.
3. It can only work for directed graph.
4. It only requires local information.

5. COMPARISON USING C# CODE

Now, I will determine the efficiency of shortest path algorithm. I created a window based application to find out the running time of both the algorithms. I created a application named Comparison1, in which I have created a Form and add a list box to display the running time of Dijkstra's and bellman ford algorithm. I implemented Dijkstra's algorithm and Bellman Ford algorithm using C# code. I created two functions for Dijkstra's and Bellman Ford algorithms. From the Form_Load () method, both functions are called and display the shortest path for every node from a single source. I used stopwatch to calculate the running time of Dijkstra's algorithm and Bellman Ford algorithm in microseconds. I used Random numbers to generate a graph.

5.1 To Strore a Graph

```
public struct Edge
{
public int u, v, w;
};
int NODES ;
int EDGES;
int[]d=new int [10000]; /* d[i] is the minimum distance from source node s to node i */
double[,] G = new double[1000, 1000]; /* graph to store the graph adjacency matrix */
```

5.2 To Store the Adjacency Matrix of Graph using Random Numbers

```
Random rn1 = new Random();

for (m = 0; m  length; m++)
{
for (n = 0; n < length; n++)
{
w[m, n] = rn1.Next(0, 10000);
G[m, n] = w[m, n];
}
}
```

5.3 To Store the Edges with their Weight

```
c = 0;
for (a = 0; a < NODES; ++a)
{
for (b = 0; b < NODES; ++b)
{
if (w[a, b] != 0)
{
edges[c].u = a;
edges[c].v = b;
edges[c].w = w[a, b];
c++;
}
l++;
}
}
EDGES = k;
```

5.4 To find Out the Running Time using Stopwatch:

```
Stopwatch s = new Stopwatch();
s.Start();
BellmanFord(source_vertex); /* Call for Bellman Ford Algorithm */
s.Stop();
long time = s.ElapsedTicks / Stopwatch.Frequency / (1000L * 1000L));
listBox1.Items.Add("time taken by Bellman ford is"+ time+" microseconds");
s.Start();
Dijkstra(source_vertex); /* Call for Dijkstra's Algorithm */
s.Stop();
long time = s.ElapsedTicks / Stopwatch.Frequency / (1000L * 1000L));
listBox1.Items.Add("time taken by Dijkstra's algorithm is"+ time+" microseconds");
```

Table I

First Run		
N	Dijkstra's Algorithm	Bellman Ford Algorithm
5	1577	741
10	1617	764
50	1853	4655
100	2777	32026
500	23923	4205010
1000	92550	33416106
Second Run		
N	Dijkstra's Algorithm	Bellman Ford Algorithm
5	1459	657
10	3570	687
50	1918	9631
100	2921	32822
500	23794	4224362
1000	96836	33603691
Third Run		
N	Dijkstra's Algorithm	Bellman Ford Algorithm
5	1567	667
10	1455	697
50	1758	4557
100	2644	37941
500	25087	4158252
1000	92149	33592017
Fourth Run		
N	Dijkstra's Algorithm	Bellman Ford Algorithm
5	1460	688
10	1411	678
50	1748	4476
100	2466	32904
500	24285	4196981
1000	92377	34340142
Fifth Run		
N	Dijkstra's Algorithm	Bellman Ford Algorithm

5	1506	670
10	1495	728
50	1659	4486
100	3479	31950
500	24323	4147961
1000	126932	33643137
Average		
5	1513.8	684.6
10	1909.6	710.8
50	1787.2	5561
100	2857.4	33528.6
500	24282.4	4186513.2
1000	100168.8	33719018.6

We can observe from this table that for the small number of vertices (N=5, 10) Bellman Ford is taking less time in comparison with Dijkstra's algorithm For the large number of vertices (N=50, 100, 500, 1000) Dijkstra's is taking less time in comparison with Bellman Ford.

6. CONCLUSION

In this paper I have given review about two single source shortest path algorithms and their comparison. There is some advantage of the algorithms as well as some disadvantage in each algorithms. I created a program for comparing the running time (in Microseconds).and execute the program five times with the different values (for each different value of N=5, 10, 50, 100, 500, 1000), And the output shown by the program(i.e. running time)is shown in the paper and then make a graph of that results. By these charts I have seen that for a small number of nodes (N=5, 10) Bellman Ford is the most efficient algorithm to find out the shortest path.

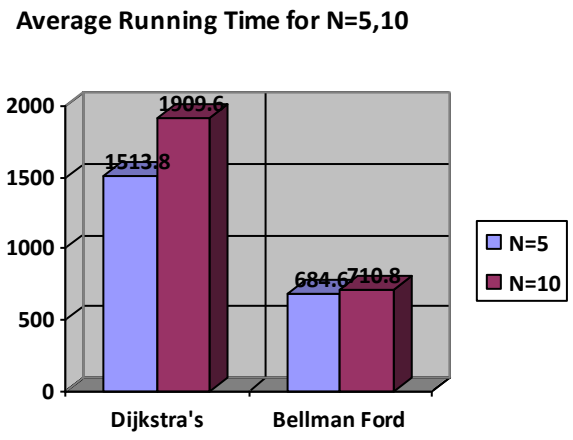

Fig 1

For N=50, Dijkstra's algorithm is the efficient algorithm.

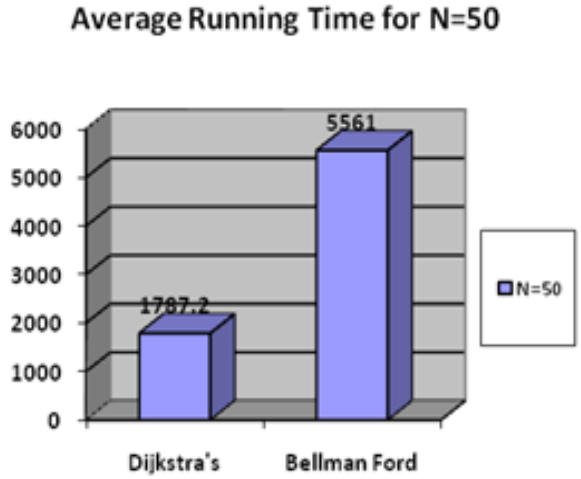

Fig 2

For N=100,again Dijkstra's algorithm is efficient algorithm, there is a very big difference in running time of Bellman Ford running time and Dijkstra's algorithm.

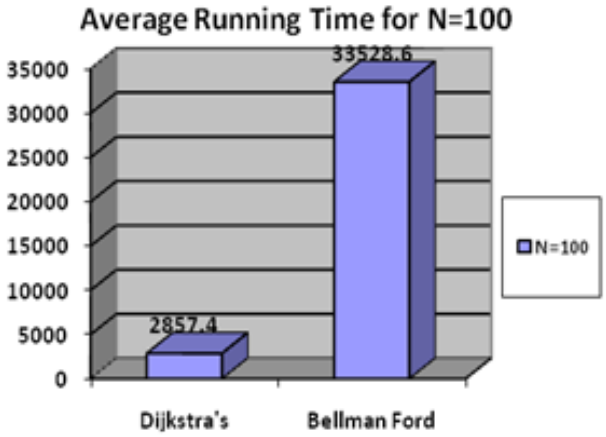

Fig 3

For N=500, 1000, Dijkstra's algorithm is more efficient algorithm than Bellman Ford.

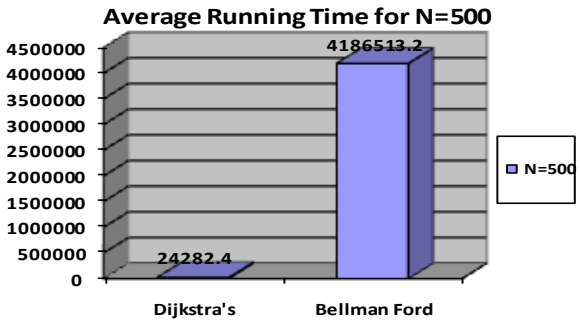

Fig 4

By these all charts, we can conclude that for small number of nodes (N < 50) Bellman Ford perform better than Dijkstra's algorithm. Dijkstra's algorithm takes twice the running time of Bellman Ford algorithm. But a large number of nodes (N>50) Dijkstra's algorithm becomes more efficient. For N=50, Bellman Ford algorithm is three times to Dijkstra's running time. For N=100, Bellman Ford is 11 times to Dijkstra's algorithm.

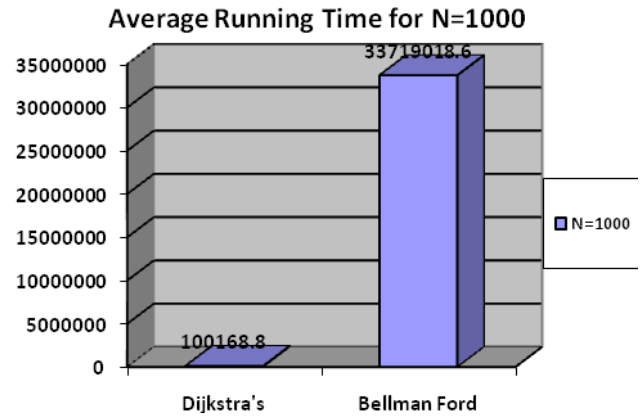

Fig 5

For N=500, 1000, Dijkstra's algorithm outperforms in comparison to Bellman Ford algorithm.

REFERENCES

[1] Faramroze Engineer, Fast Shortest Path Algorithms for Large Road Networks

[2] Kairanbay Magzhan, Hajar Mat Jani, A Review And Evaluations Of Shortest Path Algorithms

[3] http://en.algoritmy.net/article/45514/Dijkstras-algorithm

[4] http://en.algoritmy.net/article/47389/Bellman-Ford-algorithm