ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

SYSTEMATIC APPROACH IN COMPUTING OPTIMAL SHORTEST PATH IN A UNDIRECTED WEIGHTED NETWORK

CHANDANA N R
Dept. of CSE,
New Horizon College of Engineering
Bangalore, INDIA
chandana27797@gmail.com

DR. N GURUPRASAD
Professor, Dept. of CSE
New Horizon College of Engineering
Bangalore, INDIA
nguruprasad18@gmail.com
guruprasadn@newhorizonindia.edu

ABSTRACT

Shortest paths problems are familiar problems in computer science and mathematics. The computation of a shortest path between different locations appears to be a key problem in varieties of network applications. Shortest path algorithms play an important role in any path finding the application. These algorithms have been in existence since the mid-1960's. Year after year many of improvements in existing algorithms have led to new efficient algorithms. In this paper we present a survey review of an extremely simple, systematic and efficient approach in computing optimal shortest path between pair of vertices. It works similar to Dijkstra's algorithm but uses heuristic information.

Keywords – Dijkstra's, heuristic, optimal, shortest path

I INTRODUCTION

An algorithm is defined as computational procedure which takes a particular input and produces a particular output. Algorithms are used to solve wide range of problems. If G(V,E) is directed weighted graph, where V represents the set of vertices of graph & E represents the set of edges of graph. |V| represents the total number of vertices in graph & |E| represents the total number of edges in the graph. 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 Algorithm
b) Single destination Shortest Path Algorithm
c) All pair shortest path Algorithm

In Single source shortest path algorithm, we have to find out the shortest path from a source vertex to another vertex. In single destination shortest path algorithm, we have to find out the shortest path from all vertices to a single destination vertex. In All pair shortest path algorithm, we have to find out the shortest path from all vertices to another vertex. Due to the nature of routing applications, we need flexible and efficient shortest path procedures, both from a processing time point of view and also in terms of the memory requirements.[1]

The problem of computing shortest paths is indisputably one of the most wellstudied problems in computer science. It is thoroughly surprising that in the setting of real -weighted graphs, many basic shortest path problems have seen little or no progress since the early work by Dijkstra, Bellman and Ford, Floyd and Warshall, and others. For instance, no algorithm for computing single-source shortest paths (SSSPs) in arbitrarily weighted graphs has yet to improve the Bellman–Ford O(mn) time bound, where m and n are the number of edges and vertices, respectively. [4] proposed a system which is an innovative congestion control

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

algorithm named FAQ-MAST TCP (Fast Active Queue Management Stability Transmission Control Protocol) is aimed for high-speed long-latency networks. Four major difficulties in FAQ-MAST TCP are highlighted at both packet and flow levels. The architecture and characterization of equilibrium and stability properties of FAQ-MAST TCP are discussed. Experimental results are presented comparing the first Linux prototype with TCP Reno, HSTCP, and STCP in terms of throughput, fairness, stability, and responsiveness. FAQ-MAST TCP aims to rapidly stabilize high-speed long-latency networks into steady, efficient and fair operating points, in dynamic sharing environments, and the preliminary results are produced as output of our project. The Proposed architecture is explained with the help of an existing real-time example as to explain why FAQ-MAST TCP download is chosen rather than FTP download.

II Overview of Shortest Path Algorithm

Over the past decades there are various shortest path algorithm were developed. The shortest path algorithm are classified into three categories they are Single source shortest path algorithms, Single destination shortest path algorithms, All Pairs shortest path algorithms and it is shown in below figure 1.

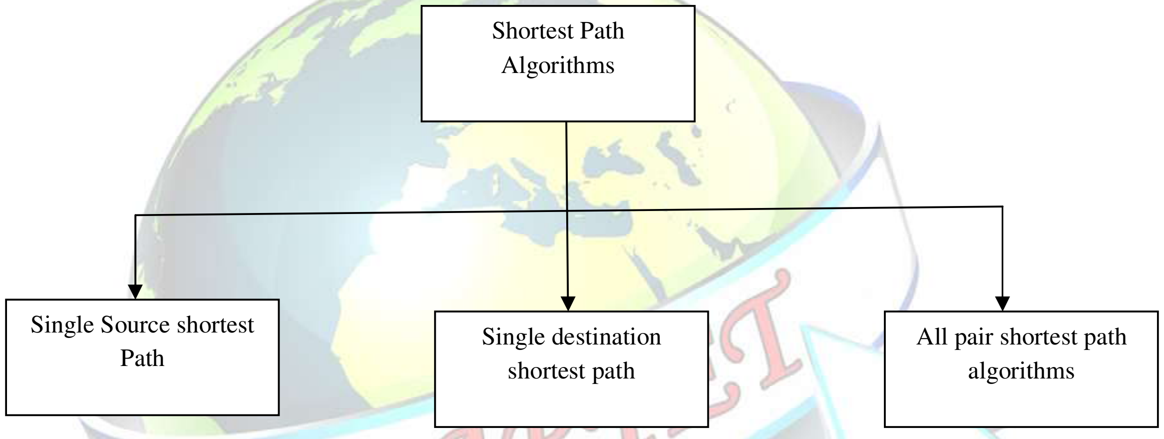
Shortest Path Algorithms
Single Source shortest Path
Single destination shortest path
All pair shortest path algorithms

Figure1. Classification of Shortest Path Algorithm

III AVOIDING CONFUSIONS ABOUT SHORTEST PATH

There are few points we would like to clarify before we discuss the algorithm.

i) There can be more than one shortest path between two vertices in a graph.
ii) The shortest path may not pass through all the vertices.
iii) It is easier to find the shortest path from the source vertex to each of the vertices and then evaluate the path between the vertices we are interested in.
iv) This algorithm can be used for directed as well as un-directed graphs
v) For the sake of simplicity, we will only consider graphs with non-negative edges.

Note: This is not the only algorithm to find the shortest path, few more like Bellman-Ford, Floyd-Warshall, Johnson's algorithm are interesting as well.

IV TIME COMPLEXITY OF SHORTEST PATH ALGORITHMS

The time efficiency of Dijkstra's algorithm is considered with cost(weight) adjacency matrix and finding minimum in an unordered array. Therefore time complexity of this algorithm is given by $\Theta(V^2)$. When it is considered with cost(weight) adjacency linked list the time complexity comes out to be $\Theta(E \log V)$. Complexity of the Floyd's algorithm is given by $\Theta(N^3)$. Time complexity of Bellman Ford is O(VE)The time

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

complexity of Johnson's algorithm becomes same as FLOYD WARSHALL when the graphs is complete (For a complete graph E = O(V^2).

V SYSTEMATIC APPROACH ALGORITHM

Step 1) Represent details of the distance network in the form of a table. Each node is provided with a column and nodes that are emerging from that are arranged in increasing order of their distances.
Step 2) Select node 1 and set cumulative distance as zero. This is shown at the top of that column
Step 3) Delete all nodes in the first table that are pointing towards node 1.
Step 4) Include recently selected node in List A of next table. (second table)
Step 5) Find nearest node for each of the nodes in List A and write in third column of second table. (If all nodes are deleted in first table corresponding to any node in List A, then remove that node from the List A of second table)
Step 6) For each node in List A of second table, calculate cumulative distance upto its nearest node as shown in last column of second table. Then select nearest node which has least cumulative distance (X) and mark a square around it in the third column of second table. Write cumulative distance covered upto to the selected node as X at the top of the respective column of the first table.
Step 7) Check whether the recently selected node is same as the required destination. If so, goto Step 9; else go to Step 8
Step 8) Delete all nodes in the first table that are pointing towards the recently selected node and then goto Step 4.
Step 9) The last selected node will be the last node of the shortest path.
Step 10) Find nodes in List A corresponding to recently selected / prefixed* nde and prefix that node in the partially formed shortest path.
Step 11) Check whether the prefixed node is the original source node. If no goto Step 12; otherwise goto Step 13
Step 12) Move to iteration in which recently prefixed node is selected (bold ones) in the third column of second tableGoto Step 10.
Step 13) Path constricted is the shortest path.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

VI ILLUSTRATIVE EXAMPLE

Let us illustrate the algorithm with a network consisting of 10 vertices

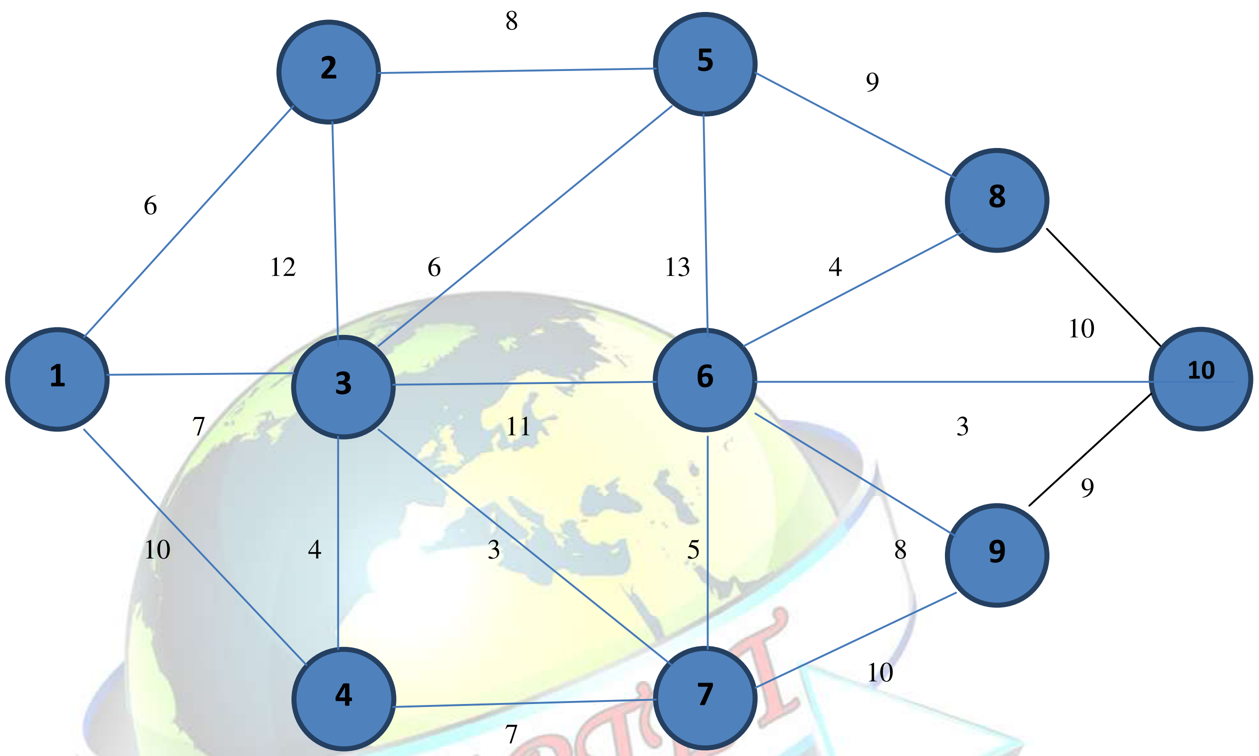

Figure2. Distance Network

Step 1) The distance network shown in Figure 2 is represented in the form of matrix as shown in Table1 below.

Table 1: Details of distances (Iteration 1)

1	2	3	4	5	6	7	8	9	10
1-2 6	2-1 6	3-7 3	4-3 7	5-3 6	6-10 3	7-3 6	8-6 4	9-6 8	10-6 3
1-3 7	2-5 8	3-4 4	4-7 7	5-2 8	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10	2-3 12	3-5 6	4-1 10	5-8 9	6-7 5	7-4 7	8-10 10	9-7 10	10-8 10
		3-1 7		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11				
		3-2 12			6-5 13				

Step 2) City 1 is selected as shown in Table 2 and the cumulative distance travelled up to City 1 is set to 0 (zero) as shown in the last column of Table 2 as well at the top of the column 1 of Table 3 of the next step.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

Table 2: Details of selection of Node in Iteration 1

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0

Step 3) delete all the arcs in Table 1 that are pointing towards City 1 (2-1, 3-1 and 4-1) as shown in Table 3 (deletion of arc is indicated using a '*' by the side of that arc)

Table 3: Updated distance (Iteration 2)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6	2-1 6 *	3-7 3	4-3 7	5-3 6	6-10 3	7-3 6	8-6 4	9-6 8	10-6 3
1-3 7	2-5 8	3-4 4	4-7 7	5-2 8 *	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10	2-3 12	3-5 6	4-1 10 *	5-8 9	6-7 5	7-4 7	8-10 10	9-7 10	10-8 10
		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11				
		3-2 12			6-5 13				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 4

Table 4: Details of selection of Node in Iteration 2

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*

Step 7) the recently selected City 2 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 2 (1-2, 3-2 and 5-2) as shown in Table 5, and then go to step 4

Table 5: Updated distance (Iteration 3)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3	4-3 7	5-3 6	6-10 3	7-3 6	8-6 4	9-6 8	10-6 3
1-3 7	2-5 8	3-4 4	4-7 7	5-2 8	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10	2-3 12	3-5 6	4-1 10 *	5-8 9	6-7 5	7-4 7	8-10 10	9-7 10	10-8 10
		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11				
		3-2 12 *			6-5 13				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 6

Table 6: Details of selection of Node in Iteration 3

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3,** 5	0+7=7* 6+8-14

Step 7) the recently selected City 3 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 3 (1-3, 2-3, 5-3, 6-3, 7-3 and 4-3) as shown in Table 7, and then go to step 4

Table 7: Updated distance (Iteration 4)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3	4-3 7 *	5-3 6	6-10 3	7-3 6 *	8-6 4	9-6 8	10-6 3
1-3 7 *	2-5 8	3-4 4	4-7 7	5-2 8	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10	2-3 12 *	3-5 6	4-1 10 *	5-8 9	6-7 5	7-4 7	8-10 10	9-7 10	10-8 10
		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11 *				
		3-2 12 *			6-5 13				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 8

Table 8: Details of selection of Node in Iteration 4

Iteration	**List A (nodes included)**	**Nearest nodes**	**Distance calculation**
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3,**5	0+7=7* 6+8=14
4	1, 2, 3	4, 5, **7**	0+10=10* 6+8=14 7+3=10*

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

In Table 8, City 7 is selected by breaking tie between City 4 and City 7 randomly.

Step 7) the recently selected City 7 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 7 (4-7, 3-7, 6-7 and 9-7) as shown in Table 9, and then go to step 4

Table 9: Updated distance (Iteration 5)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3*	4-3 7 *	5-3 6	6-10 3	7-3 6 *	8-6 4	9-6 8	10-6 3
1-3 7 *	2-5 8	3-4 4	4-7 7*	5-2 8	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10	2-3 12 *	3-5 6	4-1 10 *	5-8 9	6-7 5*	7-4 7	8-10 10	9-7 10*	10-8 10
		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11 *				
		3-2 12 *			6-5 13				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 10

Table 10: Details of selection of Node in Iteration 5

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3**, 5	0+7=7* 6+8=14
4	1, 2, 3	4, 5, **7**	0+10=10* 6+8=14 7+3=10*
5	1, 2, 3, 7	**4**, 5, 4, 6	0+10=10* 6+8=14 7+4=11 10+5=15

Step 7) the recently selected City 4 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 4 (1-4, 3-4 and 7-4) as shown in Table 11, and then go to step 4

Table 11: Updated distance (Iteration 6)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3*	4-3 7 *	5-3 6	6-10 3	7-3 6 *	8-6 4	9-6 8	10-6 3
1-3 7 *	2-5 8	3-4 4*	4-7 7*	5-2 8	6-8 4	7-6 5	8-5 9	9-10 9	10-9 9
1-4 10*	2-3 12 *	3-5 6	4-1 10 *	5-8 9	6-7 5*	7-4 7*	8-10 10	9-7 10*	10-8 10

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11 *				
		3-2 12 *			6-5 13				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 12

In table 11, all the arcs with respect to node 1 and node 4 are deleted. (* is marked) hence in table 12 under List A, these nodes are deleted by marking 'X' against each of them. The cumulative distance up to the nearest neighbour of each of the remaining nodes i.e, 2, 3 and 7 are shown in the last row of table 12.

Table 12: Details of selection of Node in Iteration 6

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3,** 5	0+7=7* 6+8=14
4	1, 2, 3	4, 5, **7**	0+10=10* 6+8=14 7+3=10*
5	1, 2, 3, 7	**4**, 5, 4, 6	0+10=10* 6+8=14 7+4=11 10+5=15
6	X X 1, 2, 3, 7, 4	5, **5**, 6	6+8=14 7+6=13 10+5=15

Step 7) the recently selected City 5 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 5 (2-5, 3-5, 6-5 and 8-5) as shown in Table 13, and then go to step 4

Table 13: Updated distance (Iteration 7)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3*	4-3 7 *	5-3 6	6-10 3	7-3 6 *	8-6 4	9-6 8	10-6 3
1-3 7 *	2-5 8 *	3-4 4*	4-7 7*	5-2 8	6-8 4	7-6 5	8-5 9 *	9-10 9	10-9 9
1-4 10*	2-3 12 *	3-5 6 *	4-1 10 *	5-8 9	6-7 5*	7-4 7*	8-10 10	9-7 10*	10-8 10
		3-1 7 *		5-6 13	6-9 8	7-9 10			
		3-6 11			6-3 11 *				
		3-2 12 *			6-5 13 *				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 14

Table 14: Details of selection of Node in Iteration 7

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3**,5	0+7=7* 6+8=14
4	1, 2, 3	4, 5, **7**	0+10=10* 6+8=14 7+3=10*
5	1, 2, 3, 7	**4**, 5, 4, 6	0+10=10* 6+8=14 7+4=11 10+5=15
6	X X 1, 2, 3, 7, 4	5, **5**, 6	6+8=14 7+6=13 10+5=15
7	X 2, 3, 7, 5	6, **6**,8	7+11=18 10+5=15 * 13+9=22

Step 7) the recently selected City 6 is not the required City 10. So go step 8.

Step 8) delete all the arcs that are pointing towards City 6 (3-6, 5-6, 8-6, 10-6, 9-6 and 7-6) as shown in Table 15, and then go to step 4

Table 13: Updated distance (Iteration 7)

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1-2 6 *	2-1 6 *	3-7 3*	4-3 7 *	5-3 6	6-10 3	7-3 6 *	8-6 4	9-6 8*	10-6 3*
1-3 7 *	2-5 8 *	3-4 4*	4-7 7*	5-2 8	6-8 4	7-6 5*	8-5 9 *	9-10 9	10-9 9
1-4 10*	2-3 12 *	3-5 6 *	4-1 10 *	5-8 9	6-7 5*	7-4 7*	8-10 10	9-7 10*	10-8 10
		3-1 7 *		5-6 13*	6-9 8	7-9 10			
		3-6 11*			6-3 11 *				
		3-2 12 *			6-5 13 *				

Step 4, 5 & 6) the calculations and updations with respect to these steps are shown in Table 16

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 3, January 2018

Table 16: Details of selection of Node in Iteration 8

Iteration	List A (nodes included)	Nearest nodes	Distance calculation
1	-	**1**	0
2	1	**2**	0+6=6*
3	1, 2	**3**,5	0+7=7* 6+8=14
4	1, 2, 3	4, 5, **7**	0+10=10* 6+8=14 7+3=10*
5	1, 2, 3, 7	**4**, 5, 4, 6	0+10=10* 6+8=14 7+4=11 10+5=15
6	X X 1, 2, 3, 7, 4	5, **5**, 6	6+8=14 7+6=13 10+5=15
7	X 2, 3, 7, 5	6, **6,**8	7+11=18 10+5=15 * 13+9=22
8	X 3, 7, 5, 6	9, 8, **10**	10+10=20 13+9=22 15+3=18*

Step 7) the recently selected City 10 is the required Destination City. So go step 9.

Step 9) the partial path: 10

Step 10) the city in List A, corresponding to city 10 is 6. So prefix city 6 in the partial path. The partial path is 6-10

Step 11) the recently prefixed city is not the source city. So, go to step 12

Step 12) Move to iteration 7 where city 6 is found in bold and go to step 10.

Step 10) The city in list A, corresponding to the city 6 is 7. So prefix city 7 in the partial path the partial path is 7-6-10

Step 11) the recently prefixed city is not the source city. So, go to step 12

Step 12) Move to iteration 4 where city 7 is found in bold and go to step 10.

Step 10) The city in list A, corresponding to the city 7 is 3. So prefix city 3 in the partial path the partial path is 3-7-6-10

Step 11) the recently prefixed city is not the source city. So, go to step 12

Step 12) Move to iteration 3 where city 3 is found in bold and go to step 10.

Step 10) The city in list A, corresponding to the city 3 is 1. So prefix city 1 in the partial path the partial path is 1-3-7-6-10

Step 11) the recently prefixed city is the source city. So, go to step 13

Step 13) the path obtained is the shortest path. It is 1-3-7-6-10. **The shortest distance is 7+3+5+3=18**

VII CONCLUSION

The time complexity for each of the Dijkstra's, Floyd-Warshall and Bellman-Ford algorithms show that these algorithms are acceptable in terms of their overall performance in solving the shortest path problem. All of these algorithms produce only one solution. An attempt has been made by us in getting optimal solution. There is tremendous scope for improvement – in terms of how this approach behaves in real time for large number of vertices.Clearly, the choice of shortest-path algorithm for a particular problem will involve complex tradeoffs between flexibility, scalability, performance, and implementation complexity. The performance models developed in this case study provide a basis for evaluating these tradeoffs.

REFERENCES

[1] Faramroze Engineer, Fast Shortest Path Algorithms for LargeNetworks
[2]KairanbayMagzhan, Hajar Mat JaniShortest Path Algorithms
[3] C. Xi, F. Qi, and L. Wei, —A New Shortest Path Algorithm based on Heuristic Strategy,Proc. of the 6th World Congress on Intelligent Control and Automation, Vol. 1, pp. 2531–2536, 2006.
[4] Christo Ananth, S.Esakki Rajavel, I.AnnaDurai, A.Mydeen@SyedAli, C.Sudalai@UtchiMahali, M.Ruban Kingston, "FAQ-MAST TCP for Secure Download", International Journal of Communication and Computer Technologies (IJCCTS), Volume 02 – No.13 Issue: 01 , Mar 2014, pp 78-85
[5] R. Sophia Porchelvi and G. Sudha. 2014. Intuitionistic Fuzzy Critical path in a network. International Conference on Mathematical Methods and Computations Proceedings.
[6] Leong Hou U, Hong Jun Zhao, Man Lung Yiu, Yuhong Li, and Zhiguo Gong, "Towards Online Shortest Path Computation", IEEE Transactions On Knowledge And Data Engineering, Vol. 26, No. 4, April 2014.