**PROJECT REPORT**

1. **Task - 1: Link State Algorithm**

**Fig-1 : Source Node v/s Running Time (in nanoseconds)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Source Node** | **small-net** | **large-net-a** | **large-net-b** |
| 1 | 21601.8 | 129903.6 | 105760.4 |
| 2 | 15639.2 | 137332 | 112993.6 |
| 3 | 13977.8 | 125896 | 100579.8 |
| 4 | 14857 | 129316.6 | 102828 |
| 5 | 13879.6 | 147106.6 | 99699.8 |
| 6 | 13880 | 127362 | 117783.2 |
| 7 | 14173.2 | 124332 | 104684.8 |
| 8 | 13782.2 | 119347 | 101557.6 |
| 9 | 13391.4 | 115828.2 | 101068.4 |
| 10 | 13782.2 | 114459.6 | 99895.8 |
| 11 |  | 114948.4 | 96572.2 |
| 12 |  | 115925.8 | 97451.8 |
| 13 |  | 143099 | 105076.4 |
| 14 |  | 137625.2 | 93835.2 |
| 15 |  | 140264.4 | 98234 |
| 16 |  | 139189.4 | 94617.2 |
| 17 |  | 141046.2 | 101557.2 |
| 18 |  | 140264.2 | 98722.8 |
| 19 |  | 141144 | 92858.2 |
| 20 |  | 137527.6 | 105662.4 |
| 21 |  | 140850.6 | 105662.8 |
| 22 |  | 136061.2 | 99309 |
| 23 |  | 137039 | 99504.6 |
| 24 |  | 135767.8 | 94617.4 |
| 25 |  | 135475 | 96181.6 |
| 26 |  | 134497.4 | 94226.2 |
| 27 |  | 132542.6 | 94128.4 |
| 28 |  | 138993.6 | 94715 |
| 29 |  | 135768.2 | 104001.2 |
| 30 |  | 136647.6 | 98234 |
| 31 |  | 138895.8 | 96181.2 |
| 32 |  | 138309.6 | 98429.2 |

**Table Of Values (time in nanoseconds)**

From the above graph, it can be concluded that a network with less number of nodes requires less time to find the shortest path from each node to every other node in the network. Since the running time of Link State algorithm for *n* nodes is *O(n3)*, hence as the number of nodes in a graph increases, the running time of the algorithm also increases. For a network with very large number of nodes, it can be concluded that for a network with same number of nodes, it takes less time to find the shortest path between nodes when the edge costs between them is a function of distance between them (large-net-b.txt) in comparison with a network of nodes which have unit edges costs between them (large-net-a.txt).

1. **Task - 2: Distance Vector Algorithm**

**Fig-2 : Initial Node v/s Number of Iteration taken to converge**

SSSS

|  |  |  |  |
| --- | --- | --- | --- |
| **Initial Node** | **small-net** | **large-net-a** | **large-net-b** |
| 1 | 3 | 5 | 7 |
| 2 | 3 | 6 | 8 |
| 3 | 3 | 5 | 7 |
| 4 | 4 | 6 | 7 |
| 5 | 4 | 5 | 7 |
| 6 | 4 | 7 | 7 |
| 7 | 4 | 7 | 8 |
| 8 | 2 | 5 | 6 |
| 9 | 3 | 5 | 7 |
| 10 | 3 | 6 | 7 |
| 11 |  | 5 | 7 |
| 12 |  | 5 | 7 |
| 13 |  | 6 | 7 |
| 14 |  | 5 | 6 |
| 15 |  | 5 | 7 |
| 16 |  | 6 | 7 |
| 17 |  | 6 | 6 |
| 18 |  | 6 | 7 |
| 19 |  | 5 | 6 |
| 20 |  | 4 | 6 |
| 21 |  | 6 | 8 |
| 22 |  | 6 | 7 |
| 23 |  | 6 | 7 |
| 24 |  | 5 | 6 |
| 25 |  | 5 | 6 |
| 26 |  | 5 | 6 |
| 27 |  | 5 | 5 |
| 28 |  | 6 | 7 |
| 29 |  | 5 | 7 |
| 30 |  | 6 | 7 |
| 31 |  | 5 | 6 |
| 32 |  | 5 | 6 |

**Table Of Values (Initial node v/s No of iterations)**

In our program, whenever a node performs Distance vector calculation and its results in change in its distance vector, the node sends its modified Distance Vector to all its neighbors. This is counted as one iteration. If there is no change in distance vector after the calculation, it is not counted as new iteration.

From the above graph, it can be concluded that a network with small number of nodes, a node's distance vector takes less number of iterations after which convergence is achieved. For a network with large number of nodes, the network with unit edge costs (large-net-a.txt), converges with less number of iterations when compared relatively to a network which has edge costs as a function of distance between nodes (large-net-b.txt).

It can be seen from the graph that the initial condition, that is each node is selected as a source node every time, the convergence time varies a little but this is not a very significant change for e.g. in above test it deviated only by 1 iteration. When each node is selected as a source node, the Distance Vector algorithm achieves convergence after performing almost the same number of iterations for each node.