Implementation of Shortest Path Algorithms

Submitted by Group 12

- Anirudh K V
- Malini K Bhaskaran
- Neha Nirmala Srinivas
- Saumya Ann George

Executive summary

The shortest path problem is the problem of finding a path between two vertices in a graph such that the sum of weights of its edges is minimum. In a single source shortest path problem we have to find the shortest path from a source vertex to all the other vertices in the graph.

Problem statement

Level 1:

Develop a program that finds the shortest path from the source vertex to all the other vertices in the graph using the following algorithms : BFS, DAG shortest path, Dijkstra's algorithm and Bellman Ford. The program should run BFS if all the edge weights are equal otherwise if the graph is acyclic it should run DAG, if the edge weights are non negative it should run Dijkstra's otherwise it should run Bellman Ford.

Level 2:

Develop a program to count the number of simple shortest paths from the source.

Pseudocode

private boolean checkForSameEdgeWeights(Graph g) : This function checks if all the edges have the same weight. It returns true if all weights are equal and in this case we run BFS.

private long runBFS(Graph g): This function calculates the shortest paths from the source using BFS algorithm.

public List<Vertex> getTopologicalOrderList(Graph g) : This function calculates the topological order of a graph. If this function returns the topological order then the graph is a DAG and we will apply DAG shortest path.

private long findShortestDistance(List<Vertex> topOrderList, Vertex source, Graph g) : This function calculates the shortest distance based on the topological order of the graph.

private boolean checkIfAllEdgesArePositive(Graph g) : This function checks if all the edge weights in the graph are positive. If this function returns true we run Dijkstra's

public long dijkstra(Graph g) : This functions calculates the shortest path using Dijkstra's algorithm. It has been implemented using indexed heap.

public boolean bellmanFordTake3(Graph g) : This function calculates the shortest path using Bellman Ford's algorithm.

Level 2 :

We get the single source shortest path using Bellman Ford's algorithm. We create a new Directed graph whose edges belong to the shortest path graph and then we count the number of shortest Paths.

public long countShortestPaths(Graph g) : This function creates a new directed graph whose edges belong to the shortest path graph. This removes any cycles present in the current graph. Then we calculate the number of shortest paths in the DAG.

Test results

Sample Outputs

```
nehas-MacBook-Pro:src NehaSrinivas$ java Level1Driver /Users/
NehaSrinivas/Desktop/lp3-l1/lp3-l1-in-k1.txt
BFS 185

1 0 -
2 2 6
```

3 2 58
4 2 49
5 2 14
6 1 1
7 2 58
8 2 15
9 2 26
10 2 58
11 2 14
12 2 41
13 2 49
14 1 1
15 1 1
16 2 41
17 2 14
18 2 41
19 2 15
20 2 49
21 2 51
22 2 58
23 2 58
24 2 58
25 2 15
26 1 1
27 1 1
28 2 26
29 2 15
30 2 68
31 2 15
32 2 49
33 2 49
34 3 89
35 2 15
36 2 6
37 2 50
38 2 99
39 2 14
40 2 6
41 1 1
42 2 83
43 2 58
44 2 15
45 2 68
46 2 99
47 2 26
48 2 83
49 1 1
50 1 1
51 1 1

52 3 87
53 2 14
54 2 14
55 2 68
56 2 15
57 2 6
58 1 1
59 2 83
60 1 1
61 2 15
62 2 14
63 2 58
64 2 68
65 3 43
66 2 15
67 2 41
68 1 1
69 2 49
70 1 1
71 2 68
72 2 14
73 2 14
74 2 14
75 2 70
76 2 58
77 2 51
78 1 1
79 2 68
80 2 60
81 2 41
82 2 15
83 1 1
84 2 15
85 2 49
86 2 15
87 2 58
88 2 41
89 2 14
90 2 78
91 2 41
92 2 58
93 2 14
94 2 26
95 2 70
96 2 15
97 2 14
98 2 99
99 1 1
100 2 58

```
nehas-MacBook-Pro:src NehaSrinivas$ java Level1Driver /Users/
NehaSrinivas/Desktop/lp3-l1/lp3-l1-in-k2.txt
DIJ 2096

1 0 -
2 16 57
3 13 1
4 23 21
5 23 7
6 25 86
7 19 51
8 24 18
9 27 23
10 24 53
11 19 2
12 20 7
13 21 1
14 26 53
15 17 59
16 24 64
17 17 80
18 17 52
19 20 57
20 21 93
21 12 28
22 21 7
23 20 51
24 20 53
25 21 35
26 29 42
27 25 31
28 11 35
29 28 64
30 22 13
31 21 23
32 16 51
33 18 52
34 30 94
35 6 1
36 21 24
37 27 77
38 20 83
39 21 3
40 18 94
41 20 11
42 22 45
43 22 13
44 22 33
```

45 20 28
46 29 65
47 26 51
48 25 16
49 21 94
50 28 92
51 14 28
52 13 1
53 17 51
54 19 61
55 7 35
56 25 42
57 15 51
58 19 21
59 7 35
60 26 72
61 17 87
62 21 35
63 19 17
64 16 3
65 25 49
66 26 42
67 21 3
68 24 80
69 33 8
70 27 2
71 22 25
72 21 17
73 23 11
74 21 89
75 33 35
76 23 30
77 25 98
78 28 35
79 27 44
80 16 59
81 26 55
82 23 17
83 19 40
84 32 88
85 15 28
86 23 15
87 16 89
88 25 55
89 9 1
90 16 85
91 26 80
92 23 83
93 16 57

```
94 14 55
95 19 57
96 27 65
97 26 93
98 23 33
99 26 86
100 24 39

nehas-MacBook-Pro:src NehaSrinivas$ java Level1Driver /Users/
NehaSrinivas/Desktop/lp3-l1/lp3-l1-in-k3.txt
B-F 1286

1 0 -
2 11 47
3 15 69
4 20 31
5 1 1
6 2 5
7 20 93
8 12 24
9 16 94
10 13 53
11 9 43
12 12 57
13 11 6
14 10 16
15 15 27
16 6 1
17 26 27
18 20 62
19 9 32
20 17 8
21 18 9
22 14 61
23 17 39
24 10 78
25 18 15
26 14 89
27 10 32
28 24 12
29 15 82
30 17 74
31 18 24
32 7 5
33 17 88
34 5 82
35 6 42
36 8 34
37 14 67
```

38 22 44
39 9 30
40 11 41
41 10 35
42 10 5
43 5 2
44 11 47
45 4 12
46 31 27
47 5 67
48 20 34
49 14 57
50 10 16
51 17 27
52 18 98
53 12 98
54 20 75
55 16 49
56 14 11
57 11 32
58 11 61
59 16 2
60 12 96
61 10 36
62 14 88
63 8 78
64 13 13
65 17 34
66 14 53
67 13 99
68 12 82
69 13 60
70 10 35
71 12 44
72 9 45
73 16 87
74 13 77
75 14 71
76 14 27
77 15 34
78 6 82
79 14 89
80 15 12
81 15 24
82 1 1
83 10 92
84 13 32
85 16 3
86 13 36

```
87 7 89
88 11 32
89 10 71
90 15 27
91 20 98
92 9 78
93 14 36
94 13 53
95 15 49
96 11 20
97 18 69
98 11 39
99 12 34
100 14 12


nehas-MacBook-Pro:src NehaSrinivas$ java Level1Driver /Users/
NehaSrinivas/Desktop/lp3-l1/lp3-l1-in-k4.txt
DAG -1998

1 0 -
2 INF -
3 -41 55
4 -12 76
5 -36 40
6 -57 21
7 INF -
8 -7 1
9 -59 12
10 14 1
11 INF -
12 -51 71
13 INF -
14 INF -
15 INF -
16 -46 43
17 -44 54
18 -60 83
19 -55 56
20 INF -
21 -51 62
22 -36 70
23 INF -
24 -43 40
25 INF -
26 INF -
27 INF -
28 -29 81
29 INF -
```

30 INF –
31 INF –
32 INF –
33 −31 80
34 INF –
35 −57 19
36 −29 78
37 INF –
38 INF –
39 −66 46
40 −45 43
41 INF –
42 INF –
43 −40 33
44 −38 33
45 INF –
46 −60 21
47 −44 81
48 INF –
49 −56 91
50 INF –
51 INF –
52 INF –
53 −3 76
54 −47 71
55 −35 78
56 −49 90
57 INF –
58 −66 59
59 −60 82
60 INF –
61 INF –
62 −48 43
63 INF –
64 INF –
65 24 53
66 INF –
67 −66 91
68 INF –
69 INF –
70 −26 76
71 −43 81
72 INF –
73 INF –
74 INF –
75 INF –
76 −18 86
77 INF –
78 −28 76

```
79 INF -
80 -27 78
81 -35 78
82 -51 90
83 -53 54
84 INF -
85 -17 86
86 -11 8
87 -51 90
88 -63 91
89 INF -
90 -46 94
91 -60 12
92 INF -
93 INF -
94 -39 55
95 -2 1
96 -8 70
97 9 10
98 INF -
99 INF -
100 INF -
```

Level 2 Outputs :
———————————

```
nehas-MacBook-Pro:src NehaSrinivas$ java Level2Driver /Users/
NehaSrinivas/Desktop/lp3-l2/1.txt
17
1 0 1
2 4 2
3 1 1
4 2 1
5 1 1
6 2 2
7 3 2
8 3 3
9 4 3
10 2 1
```

```
nehas-MacBook-Pro:src NehaSrinivas$ java Level2Driver /Users/
NehaSrinivas/Desktop/lp3-l2/2.txt
819
1 0 1
2 248 55
3 92 1
4 77 1
```

5 10 1
6 122 3
7 85 1
8 20 1
9 211 8
10 233 21
11 205 8
12 289 55
13 112 2
14 183 8
15 68 1
16 113 2
17 45 1
18 221 13
19 268 55
20 231 21
21 152 5
22 132 3
23 58 1
24 29 1
25 114 3
26 243 34
27 128 3
28 137 5
29 236 34
30 193 8
31 286 55
32 168 8
33 87 1
34 162 8
35 214 8
36 180 8
37 181 8
38 198 8
39 235 21
40 51 1
41 222 13
42 293 55
43 259 55
44 255 55
45 143 5
46 102 1
47 37 1
48 175 8
49 276 55
50 294 89

```
nehas-MacBook-Pro:src NehaSrinivas$ java Level2Driver /Users/
NehaSrinivas/Desktop/lp3-l2/rchannel-100-2-3.txt
13

nehas-MacBook-Pro:src NehaSrinivas$ java Level2Driver /Users/
NehaSrinivas/Desktop/lp3-l2/3.txt
4550
1 0 1
2 209 22
3 20 1
4 1 1
5 138 6
6 211 55
7 145 6
8 11 1
9 36 1
10 257 683
11 28 1
12 179 10
13 267 683
14 73 1
15 41 1
16 224 132
17 92 2
18 231 132
19 15 1
20 263 683
21 64 1
22 112 4
23 82 2
24 101 2
25 234 191
26 226 132
27 210 22
28 151 6
29 172 6
30 163 6
31 156 6
32 59 1
33 30 1
34 33 1
35 184 10
36 233 191
37 39 1
38 110 4
39 201 22
40 121 4
41 192 22
42 214 55
```

```
43 49 1
44 243 246
45 131 6
46 248 246
47 26 1
48 129 6
49 89 2
50 274 929


nehas-MacBook-Pro:src NehaSrinivas$ java Level2Driver /Users/
NehaSrinivas/Desktop/sp.txt
Non-positive cycle in graph. DAC is not applicable
(5,1) -8
(4,5) 1
(2,4) 5
(1,2) 2
```