**IMPLEMENTATION OF DATA STRUCTURES AND ALGORITHMS**

**(MP4 REPORT)**

**By,**

**PRIYANKA BURA**

**(pxb144230)**

**Project Description –**

Find the critical paths within a graph, a critical path whose edges are tight and vertices are critical.

Condition for vertex to be critical is **Vertex.LC = Vertex.EC** or **Slack = 0** (LC – Latest Completion time, EC – Expected Completion Time).

Condition for Edge to be tight is Vertices connecting the edge should be critical and fromVertex.lc + toVertex.d == toVertex.lc (d – duration of a task)

**CriticalPaths.java** contains following functions:

* **findCriticalPaths** – Function to find all the critical paths within a given graph

**Graph.java** implemented the following functions:

* **findIndegreeTopologicalOrder** – function to get the list of vertices in its topological order using indegree property of vertices.
* **findDFSTopologicalOrder** - function to get the list of vertices in its topological order using DFS. Topological order will be the reverse of DFS finish order.
* **DFSVisit** – function to run DFS on given vertex and all its adjacent vertices recursively and add to topological order.
* **calculateEC** – function to calculate estimated completion time of all vertices, as the updation of EC should start from source we use IndegreeTopologicalOrder list generated above.
* **calculateLC** – function to calculate latest completion time of all vertices, as the LC needs to be updated from target, we need reverse order so will make use of DFSTopologicalOrder list genereated above.
* **calculateCriticalPaths** – function to find the number of critical paths, this can be achieved by iterating through all the vertices in IndegreeTopologicalOrder whose incident edges are tight and updating the other end vertex’s #critical paths by adding to its current vertex’s #critical paths.
* **enumeratePaths** – function to get an array of all the critical paths possible within a given graph, starting from the source we keep on iterating through all the adjacent tight edges and their other end vertices till the target is reached by recursively calling enumeratePaths function with other end vertex and incremented index.

**Output for all input files –**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input File** | **Length of a critical path** | **critical path** | **#critical nodes** | **#critical paths** | **Run Time (msec)** |
| in.txt | 10 | 1 3 6 9 | 4 | 1 | 1 |
| pert.10.15.txt | 61 | 3 8 4 9 | 4 | 1 | 2 |
| pert.100.150.txt | 100 | 59 3 68 53 9 19 76 91 | 8 | 1 | 4 |
| pert.100.500.txt | 237 | 23 34 4 17 68 35 92 80 93 94 65 58 21 83 77 36 85 24 | 18 | 1 | 4 |
| pert.1000.5000.txt | 263 | 533 71 723 976 838 912 384 518 395 311 350 600 223 291 141 922 787 625 564 773 425 | 27 | 2 | 23 |
| channel-x.txt | 17 | 1 3 25 30 12 7 26 10 15 14 | 17 | 54 | 7 |