**Report on Mandatory Project 4: Implementation of Critical paths**

Critical paths in PERT charts:

The **critical path method** (**CPM**) is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for [scheduling](https://en.wikipedia.org/wiki/Schedule_(project_management)) a set of activities. Critical path is constructed as follows:

1. A list of all activities required to complete the task.
2. The time ([duration](https://en.wikipedia.org/wiki/Duration_(project_management))) that each activity will take to complete
3. The [dependencies](https://en.wikipedia.org/wiki/Dependency_(project_management)) between the activities and
4. Logical start and end points.

Using these values, CPM calculates the [longest path](https://en.wikipedia.org/wiki/Longest_path) of planned activities to logical end points and the earliest and latest that each activity can start and finish. This process determines which activities are "critical" (i.e., on the longest path). A critical path is the sequence of project network activities which add up to the longest overall duration, regardless if that longest duration has float or not. This determines the shortest time possible to complete the project.

**Code base:**

Skeleton implementation: Java library, mp4.zip

References: Pseudo code from Professor’s notes, https://en.wikipedia.org/wiki/Critical\_path\_method

**The following have been implemented:**

1. Algorithms for finding critical paths (DAGs)
2. Algorithms for finding topological orders in DAGs.
3. Computation of EC and LC using the output from both the above algorithms

**Input format**

A directed graph suitable for readDirectedGraph() in the Graph class is given, followed by the durations of the nodes. If the input graph has N vertices, the last two (N-1 and N) are dummy nodes and are not incident to any edges in the input. These two nodes are to be used for s and t.

**Output format**

Line 1: Length of a critical path and no of critical nodes

Line 2: A critical path and no of critical paths

Line 3: blank

Line 4: header of table (Task EC LC Slack)

Next n lines: Task number, its earliest completion time, lastest

completion time, and slack.

Next k lines: one critical path per line.

|  |  |
| --- | --- |
| **Sample input**  11 12  1 3 1  1 4 1  2 4 1  2 5 1  3 6 1  4 6 1  4 7 1  5 7 1  5 8 1  6 9 1  7 9 1  8 9 1  3 2 3 2 1  3 2 4 1 0 0 | **Sample output**  10 # Length of critical path  1 3 6 9 # A critical path  Task EC LC Slack  1 3 3 0  2 2 4 2  3 6 6 0  4 5 6 1  5 3 5 2  6 9 9 0  7 7 9 2  8 7 9 2  9 10 10 0  Additional output :  4 # Number of critical nodes  1 # Number of critical paths  1 3 6 9 # Critical path 1 |

**Results**

|  |  |  |  |
| --- | --- | --- | --- |
| **File** | **No. of CP** | **Critical Path** | **Time/Memory** |
| **Channel-x.txt** | **2** | 1 16 27 3 3 3 25 24 21 30 12 7 7 7 26 28 5 10 15 15 14 18 | Time: 35 msec.  Memory: 1 MB / 61 MB. |
| **In.txt** | **1** | 1 3 6 9 | Time: 17 msec.  Memory: 1 MB / 61 MB. |
| **Pert.10.15.txt** | **1** | 3 8 4 9 | Time: 21 msec.  Memory: 1 MB / 61 MB. |
| **Pert.100.150.txt** | **1** | 59 3 68 53 9 19 76 91 | Time: 40 msec.  Memory: 1 MB / 61 MB. |
| **Pert.100.500.txt** | **1** | 23 34 4 17 68 35 92 80 93 94 65 58 21 83 77 36 85 24 | Time: 69 msec.  Memory: 2 MB / 61 MB. |
| **Pert.1000.5000.txt** | **1** | 533 840 71 58 803 354 199 723 717 976 838 912 384 518 395 311 350 600 223 291 141 922 787 625 564 773 425 | Time: 371 msec.  Memory: 13 MB / 61 MB. |