19201089

Section: 9

Task 1A

Here, we have two lists — q that stores the nodes with no prerequisite and f that stores the resulting sequence. The function ToposortBFS takes the graph and list of indegrees as input. While there are elements in the queue, they are popped and stored in f and the indegrees of the neighbours are decreased. If any of the indegrees are now 0, they are pushed in queue. Finally, we check if there was a cycle by comparing the count variable that was incrementing with each pop from queue, and the number of nodes. If they don't match, there exists a cycle

Task 1B

Here, there are two functions — DFS and isCycle. isCycle checks for cycle within the graph and DFS conducts the DFS algorithm. The isCycle function calls DFS function for the unvisited nodes and looks for cycle. DFS function also handles starting time and ending time. ~~in~~ First the isCycle function is called and it handles calling DFS. Lastly if there is a cycle, the isCycle function returns it. It also returns the ending time that is used to get the topological order.

~~Task1~~

Task2

This is similar to Task1A except that when popping from queue, we don't pop the latest element. We pop the smallest element so that it is lexicographically sorted to give the lexicographically smaller path.

Task3

Here, first DFS is run. Then the graph is transposed. The transposed graph is underwent DFS again but in order of the descending ending time of the nodes. From here, we can identify the different strongly connected components. ~~that can~~