Analysis of Algorithms II

Project 1

Report

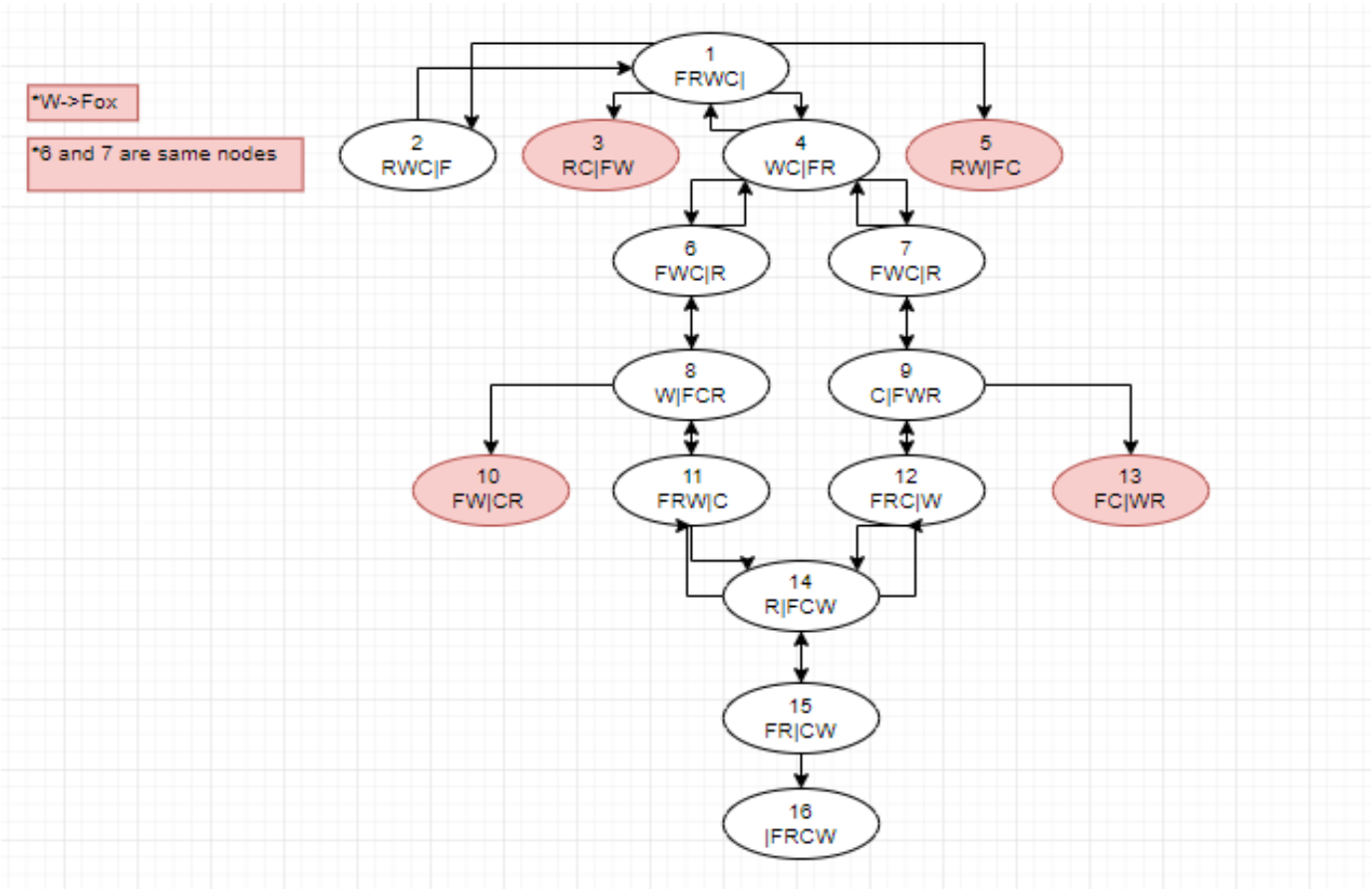
Ahmet Göktuğ SEVİNÇ

150140120

18.03.2018

1)

To solve this problem, firstly I've considered all of the situations as a node and tried to find the connections between these nodes. For this purpuse, I've chosen the situation that all the objects are on the deparute point as the first node and the situation that all the object are on the arrival point as the final node. And the purpose was to find the path between that first and last nodes. Then, I have drawn the following graph considering suitable options. The red nodes implies the forbidden situations.



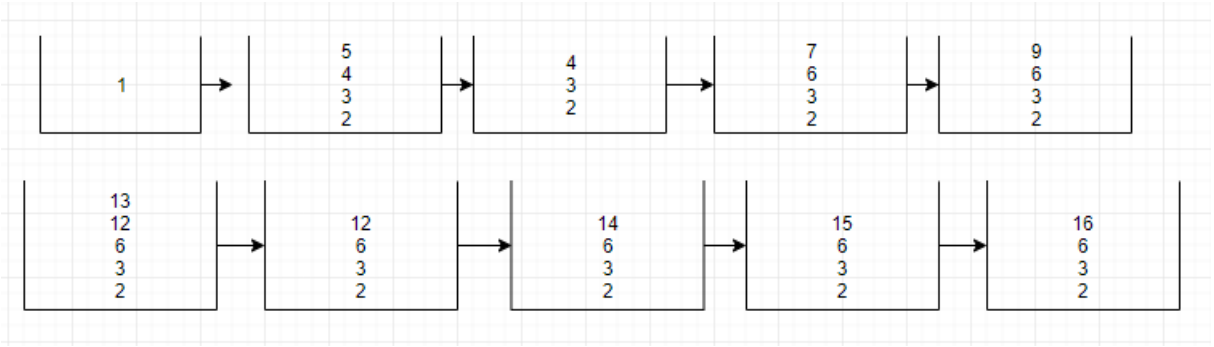
3) For example, at the beginning of my graph, in DPS algorithm 2, 3, 4 and 5 are pushed on to stack. After that 5 will be popped, then 4 will be popped and 1, 6, 7 will be pushed on the stack. Now if we come back to 1 and if we didn't mark 1 as visited, 2, 3, 4, 5 will be pushed to stack again. To avoid from this situation we need to keep track of visited nodes.

4) No edge joins two nodes in the same layer and there is no odd cycle, so this graph is bipartite graph.

5) In DFS algorithm depth is searched first, and in BFS algorithm we move layer by layer, since our problem requires to move from the first node to the final node, DFS algorithm is giving better results.

In DFS stack elements are going to be:

***Note that since I check if a node is visited or not when I push it to the stack, parent nodes won't be pushed again. For example at the 4th step, 4th node has a connection with 1st node but since 1 is marked as visited it won't be pushed.**



In BFS queue elements are going to be:

***Note that since I check if a node is visited or not when I push it to the queue, same nodes won't be pushed again. For example since parent of 12th and 13th nodes is 14th node, when I pop 12 it 14 will be pushed but then when I pop 13 14 won't be pushed again because it is marked as visited.**