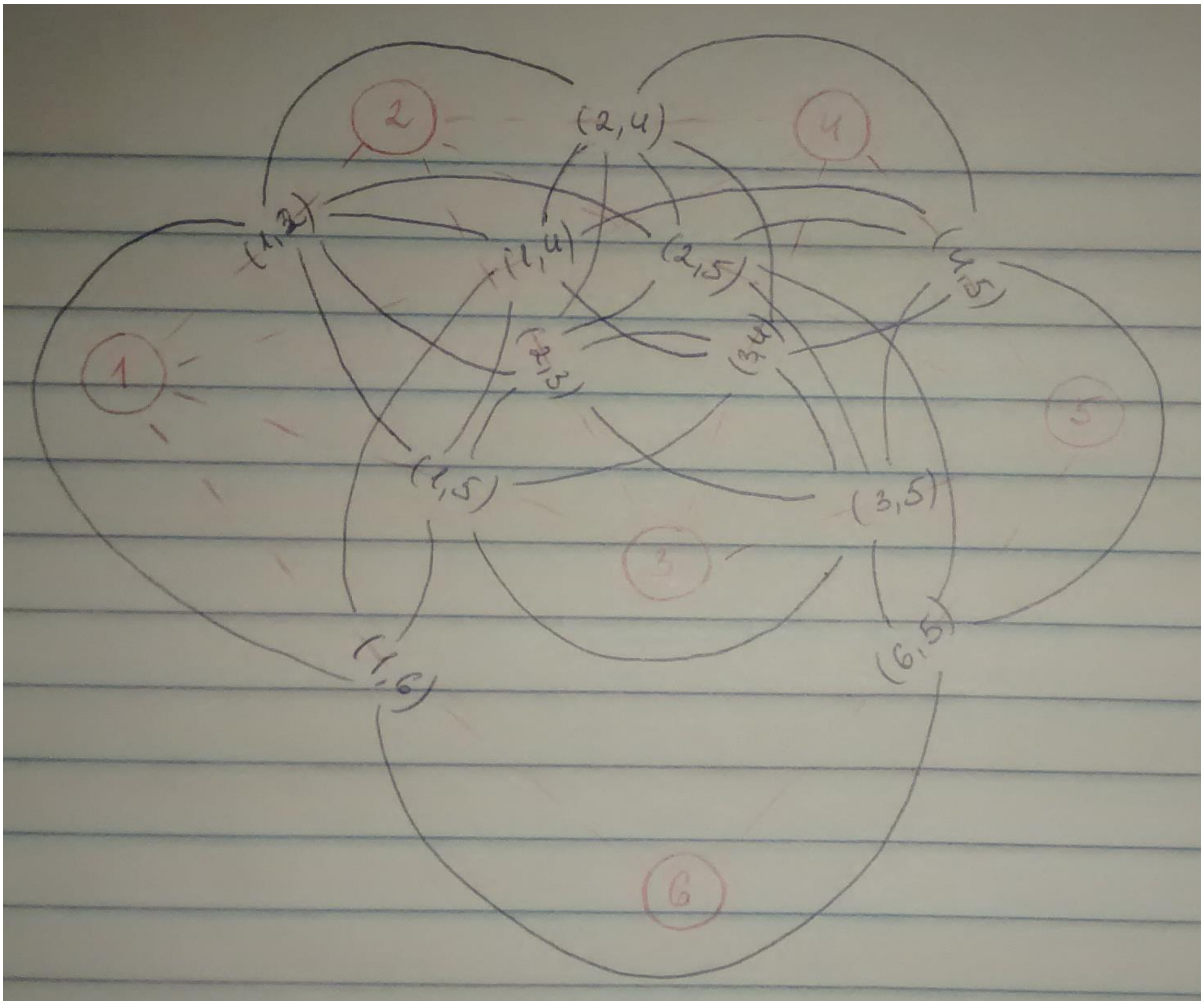
Homework #9

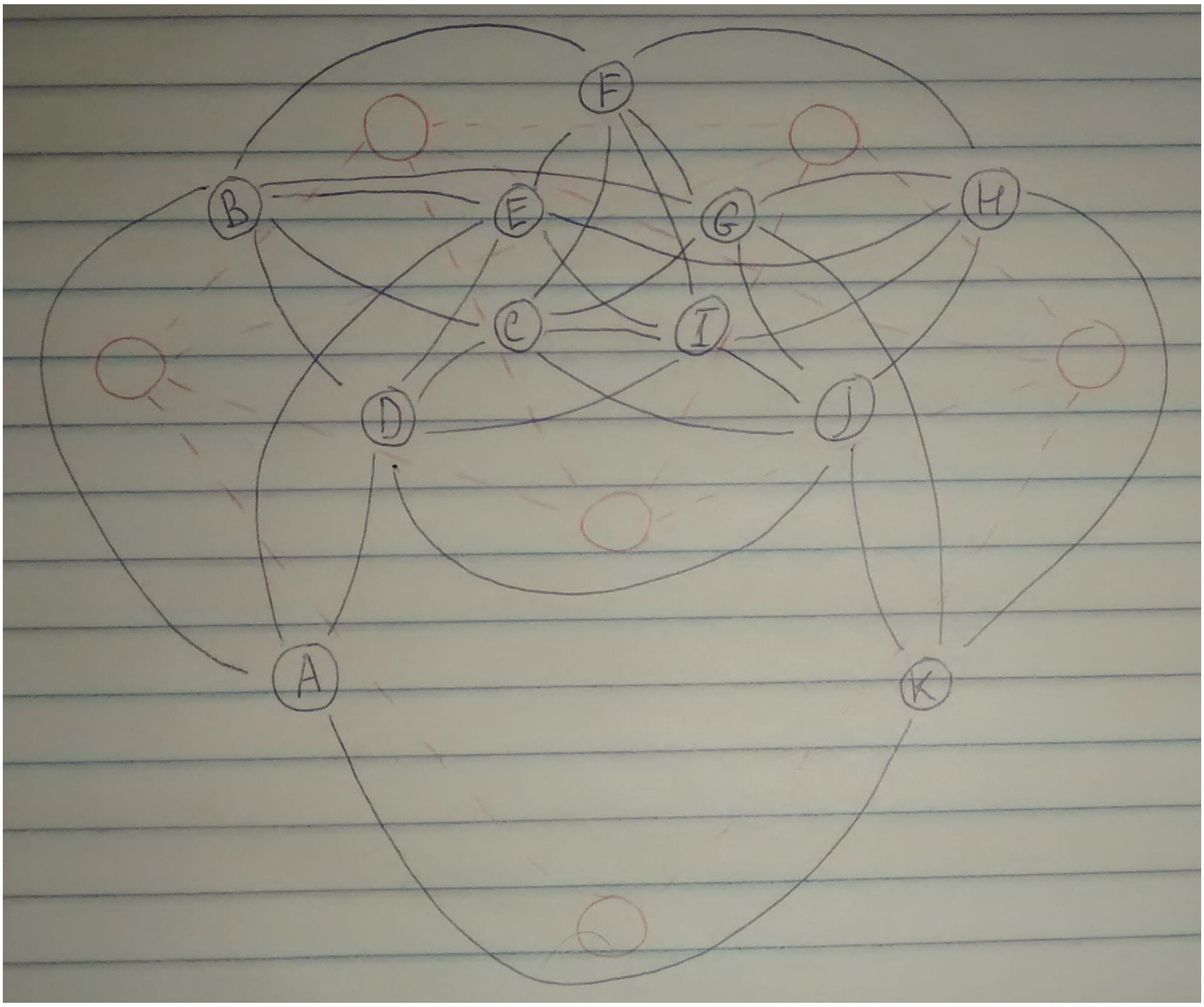
Graphs II

Anastasiia Konoplina

1. Study the line graph. Use graph from lecture - below. Form a line graph. Formulate the Euler and Hamilton path tasks on original graph and line graph. How do these relate to the Euler and Hamilton paths in the original graph?

I didn't know if I can add labels to nodes or I can use edges' labels, so just in case I have done both options:





For original graph:

- Euler path – path to visit each edge exactly once
- Hamilton path – path to visit every node exactly once

For line graph:

- Euler path – path to visit every node exactly once
- Hamilton path – usually node in line graph contains edges between to nodes of original graph. So Hamilton path in L(G) will be path of nodes which includes only “unique” values of original graph nodes. For example path (3,1), (2,3) but not (1,2), (2,3), (3,1) – one node is redundant.

2. Implement yourself the main graph traversal methods (alternatively, simulate all explicitly). Use the graph from the bottom of the page (and the edge list).

- *Run the BFS traversal starting from A*
- *Run the recursive depth first search (DFS) algorithm outputting nodes in order of processing. Print the discovery and finishing timestamps d(u) and f(u).*
- *Make the graph undirected*
- *and run the same BFS and DFS on the undirected graph*

- BFS traversal: ['A', 'C', 'D', 'G', 'E', 'B', 'N', 'F', 'H', 'K', 'I', 'J', 'M', 'L']
- DFS recursive: I think it cannot work, since some nodes do not have output edges, so it raise an error which I do not know how to avoid.
- BFS undirect: ['A', 'D', 'C', 'N', 'G', 'B', 'K', 'E', 'H', 'F', 'M', 'I', 'L', 'J']
- DFS undirect: ['A', 'D', 'C', 'E', 'N', 'G', 'H', 'F', 'M', 'L', 'I', 'J', 'B', 'K']

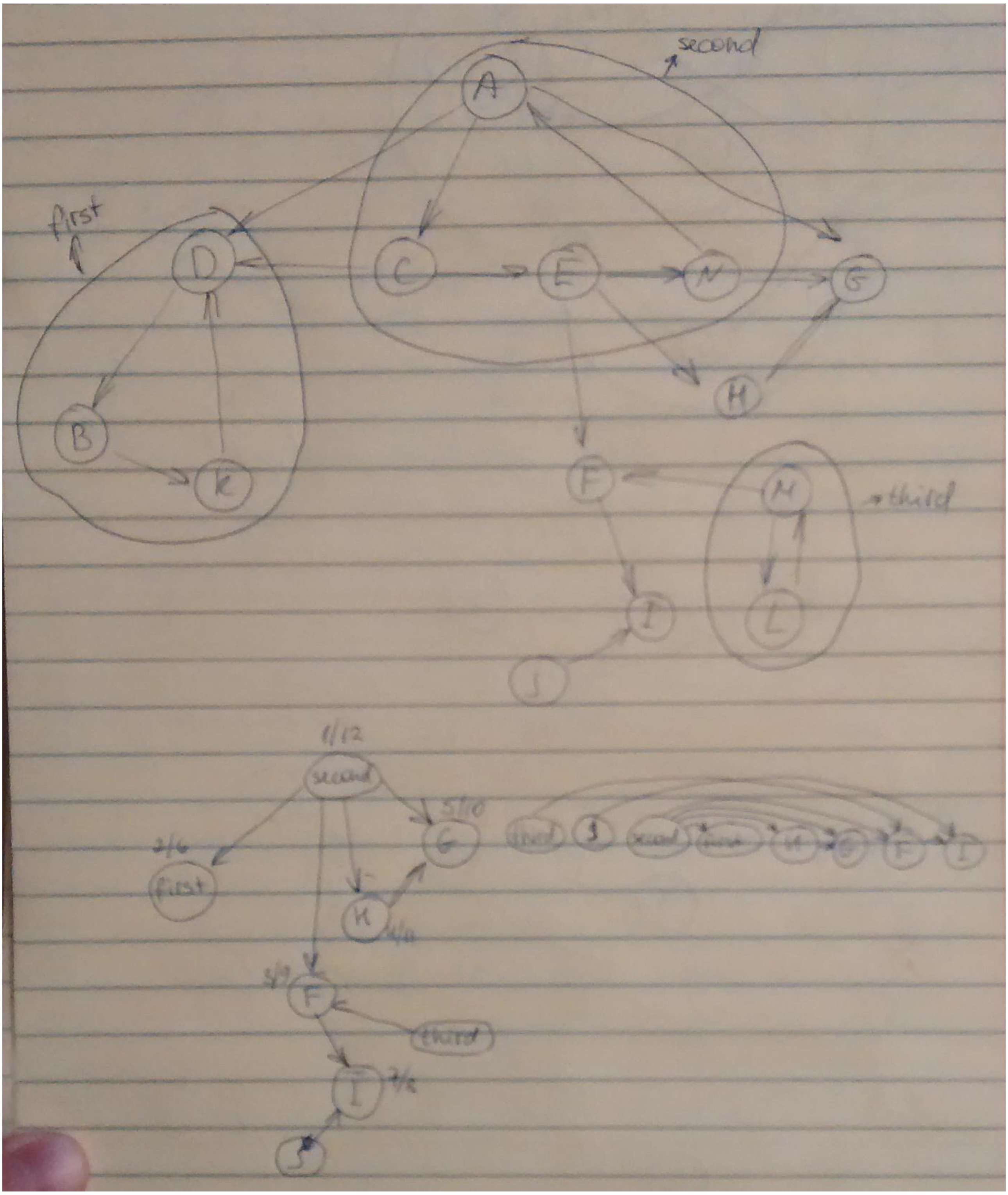
Code is here. Functions: “bfs” and “dfs_rec”

3. Use the same graph. Implement the DFS but now without the recursion. Instead use the explicit stack (one-ended queue). Make sure to get exactly the same results as with the recursion in 2.

Returns: ['A', 'C', 'E', 'N', 'G', 'F', 'I', 'H', 'D', 'B', 'K', 'J', 'M', 'L']

Code is here. Function: “dfs_stack”

5. *Discover the Strongly Connected Components and output the component graph; provide the Topological sort ordering of the SCC graph. In the first DFS traversal pass the graph in reverse alphabetical order (the start node, and always also the adjacency list).*


second
A
first
D
C
E
N
G
H
B
K
F
M
third
I
L
J
1/12
second
5/10
G
2/6
first
K
4/8
3/9
F
third
I
7/8
J
third
J
second
first
H
G
F
I