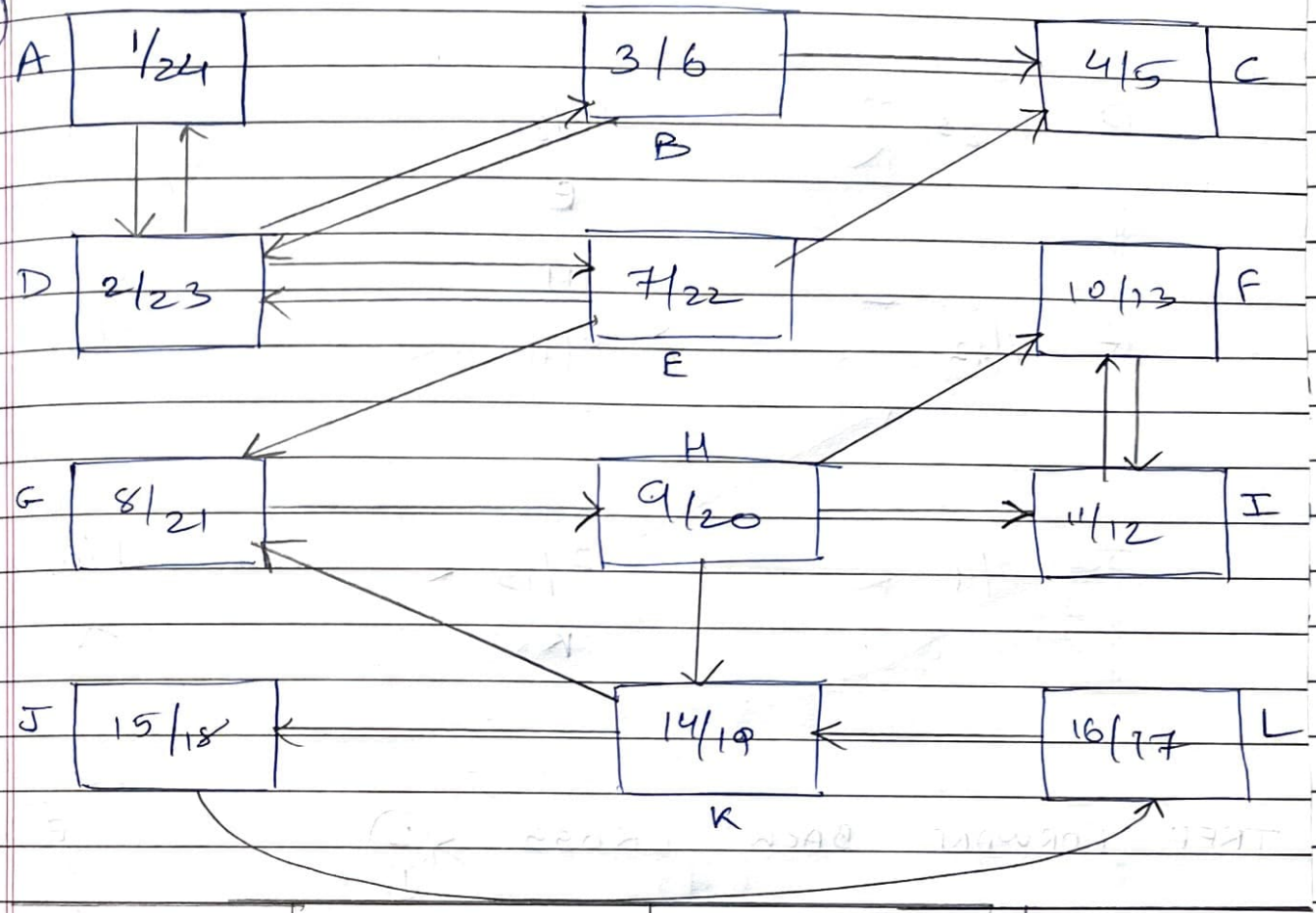
# HOMEWORK 5: GRAPHS EXPLORATION (BFS, DFS)

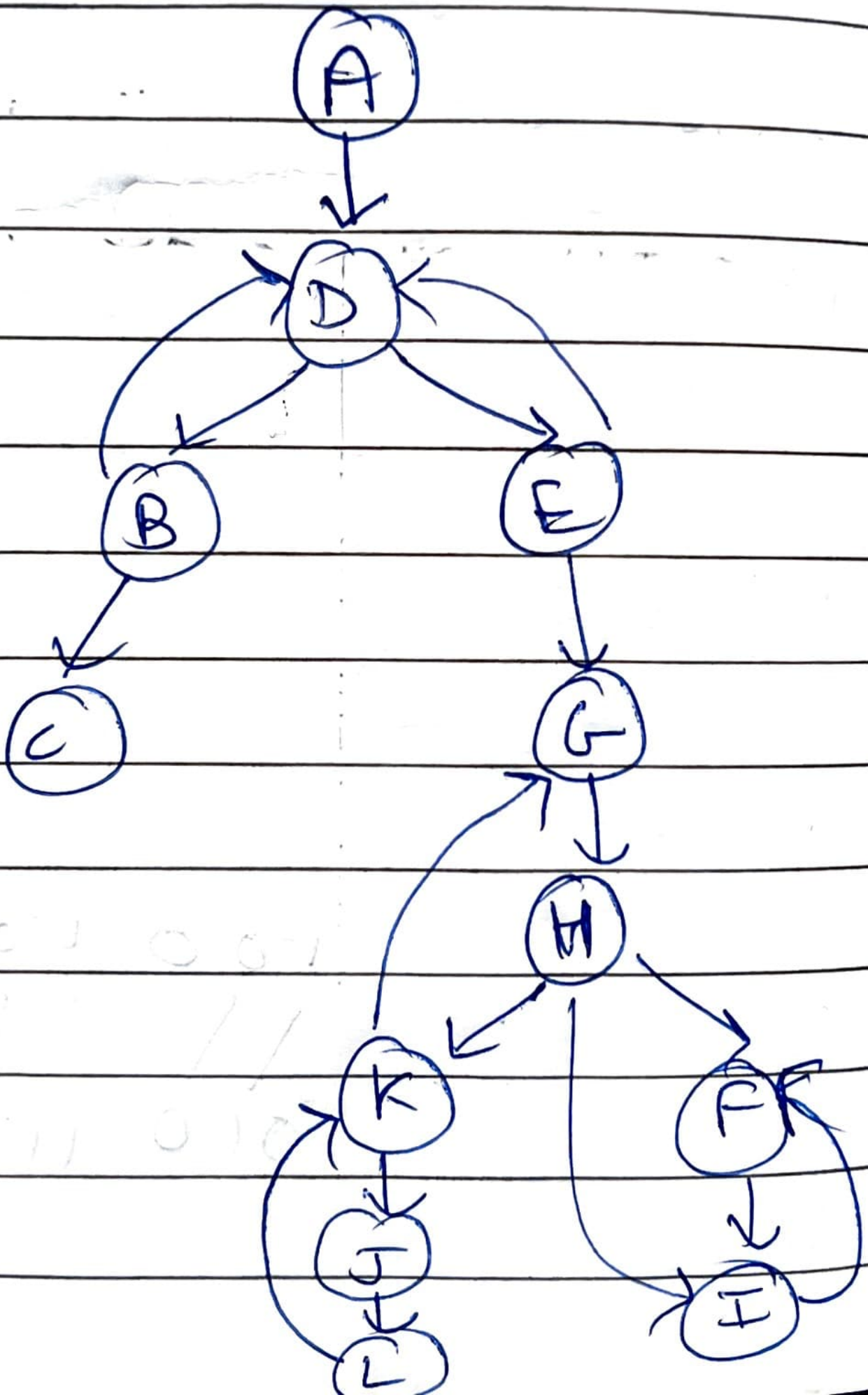
+ DAKSH BHUVA
+ 10475468

## * Question 1:

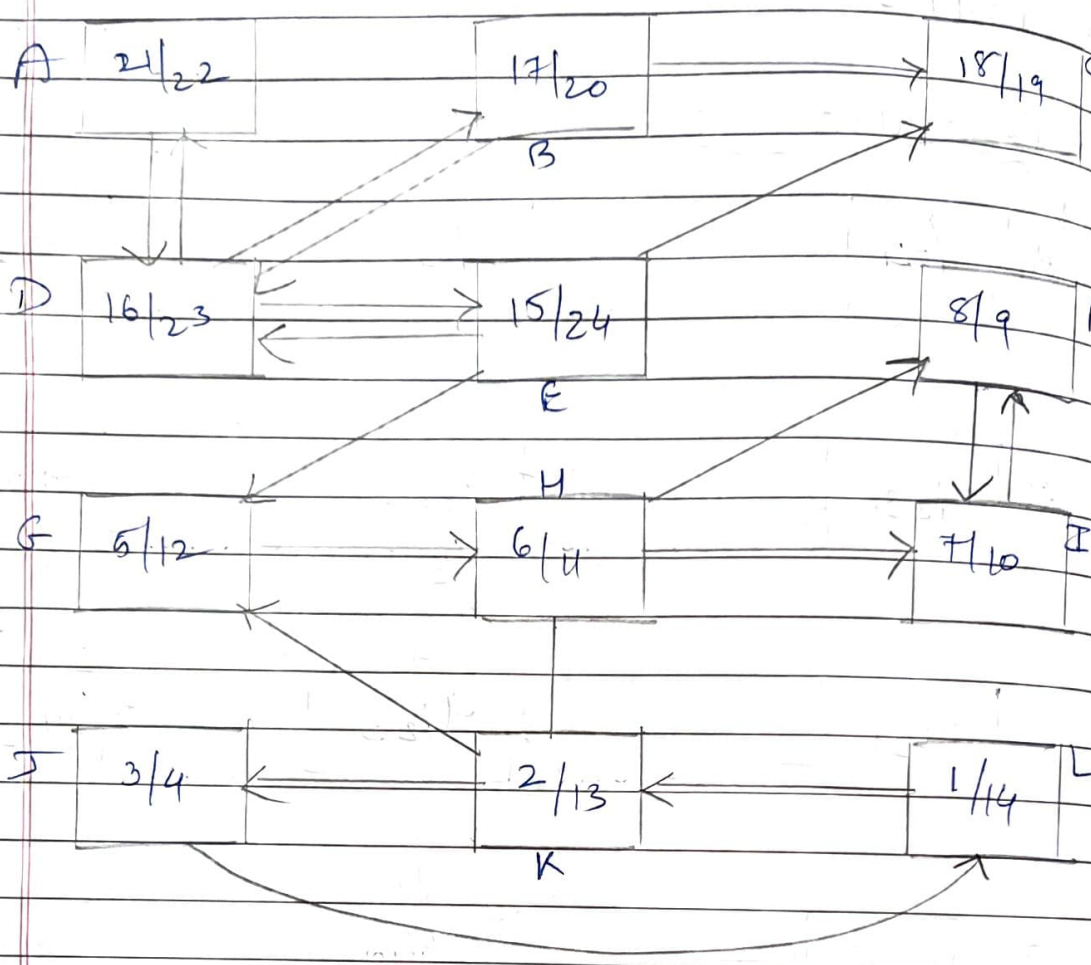
a)



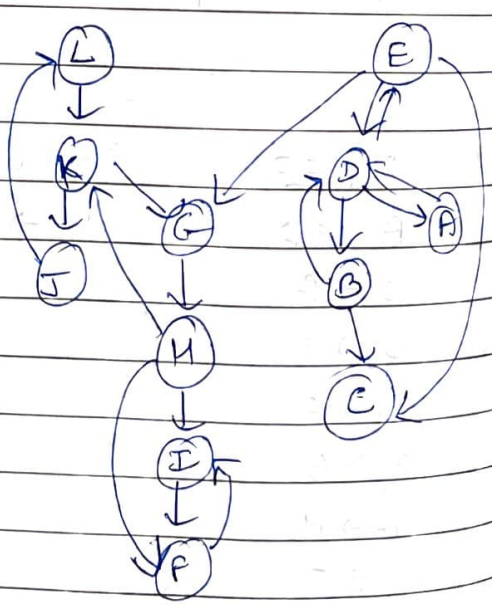

| TREE | FORWARD | BACK | CROSS |
|---|---|---|---|
| A → D | H → I | D → A | E → C |
| D → B | | B → D | |
| D → E | | E → D | |
| B → C | | L → K | |
| E → G | | K → G | |
| G → H | | I → F | |
| H → F | | | |
| F → I | | | |
| H → K | | | |
| K → J | | | |
| J → L | | | |


A
D
B
E
C
G
H
K
F
J
I
L

(b)

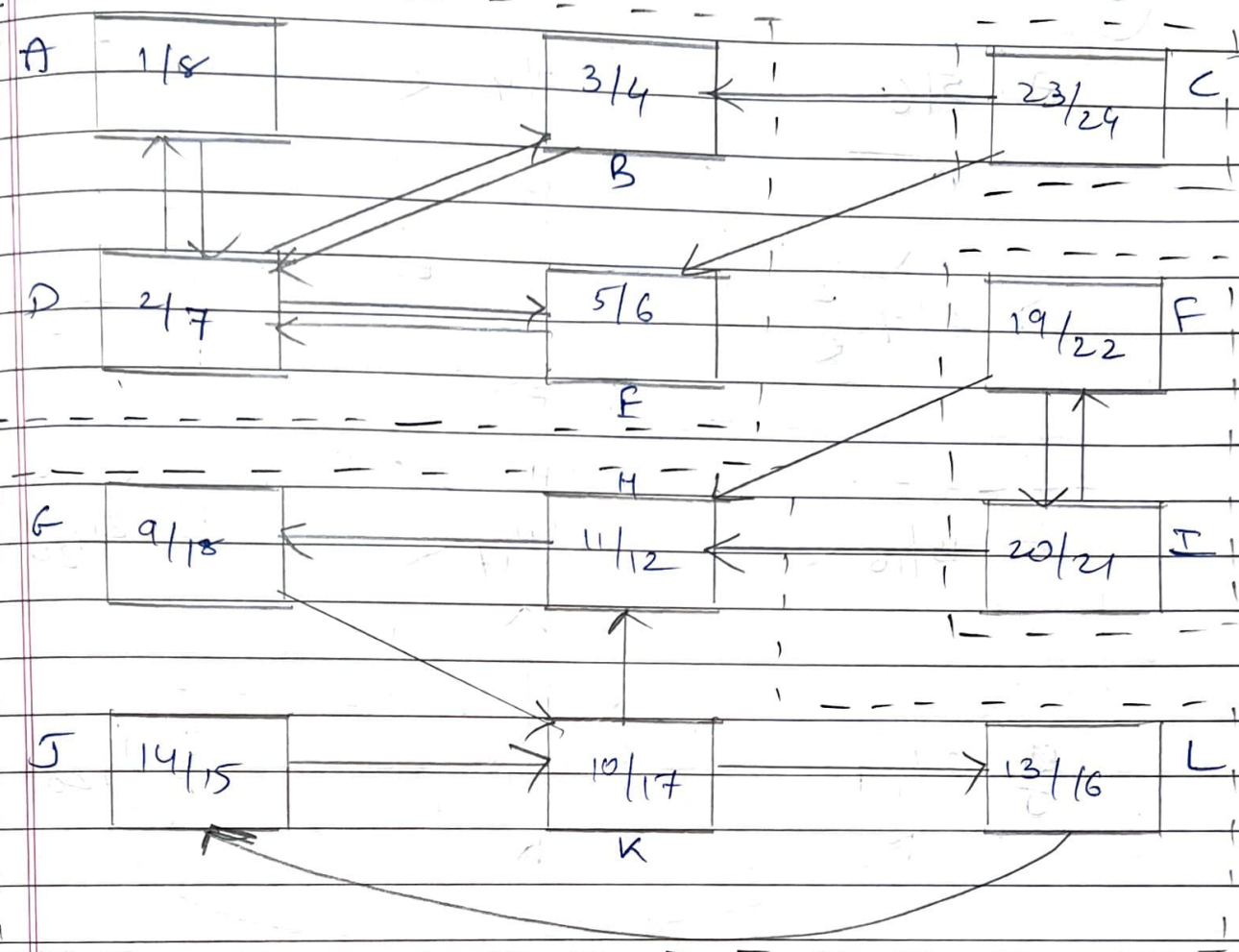



| TREE | FORWARD | BACK | CROSS |
|---|---|---|---|
| L→K | H→F | J→L | E→G |
| K→J | E→C | H→K | |
| K→G | | D→E | |
| G→H | | A→D | |
| H→I | | B→D | |
| I→F | | F→I | |
| E→D | | | |
| D→B | | | |
| B→C | | | |
| D→A | | | |



classmate
Date
Page

(c)




{A, D, B, E}

SCC 1

SCC 4

{C}

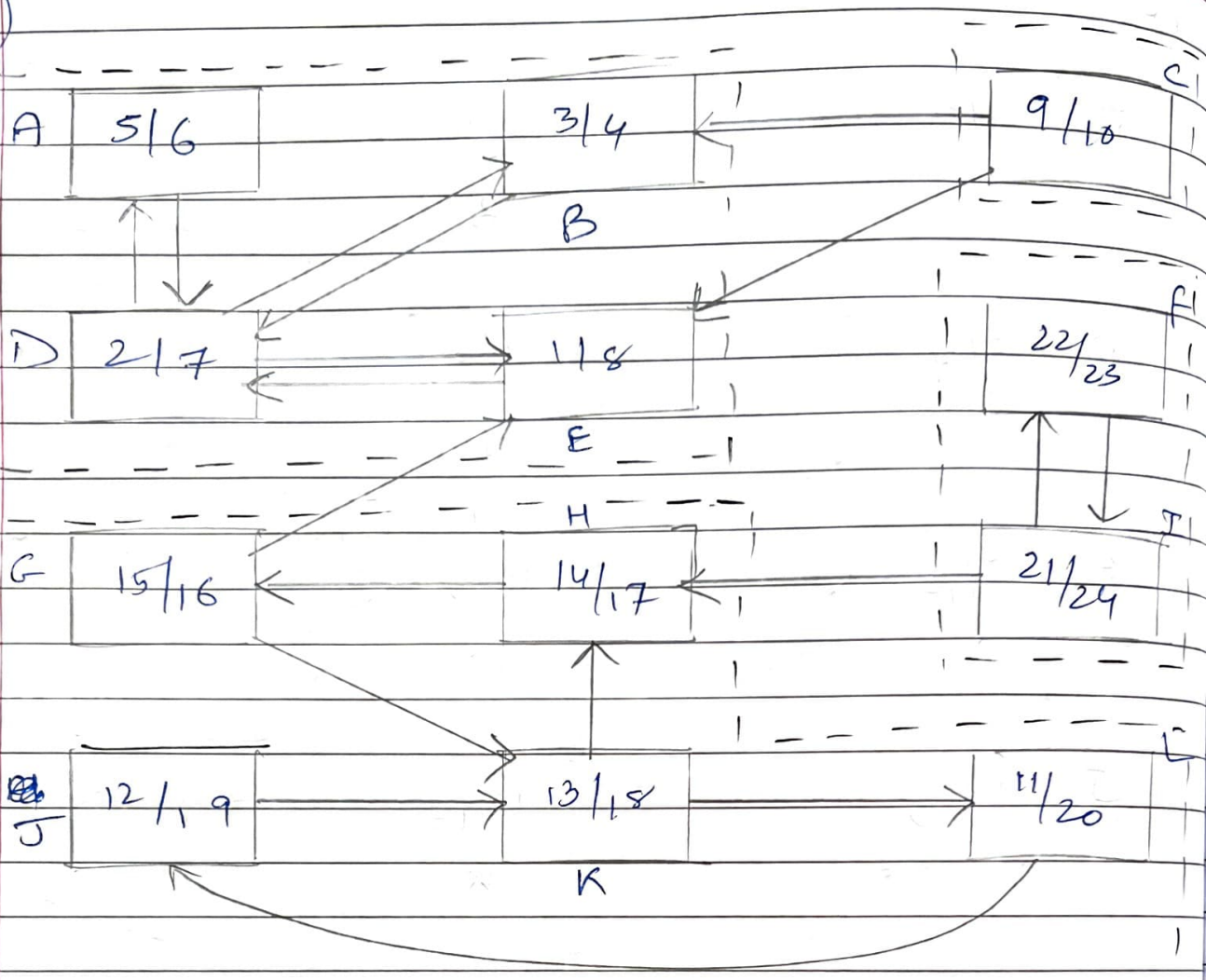
SCC 2

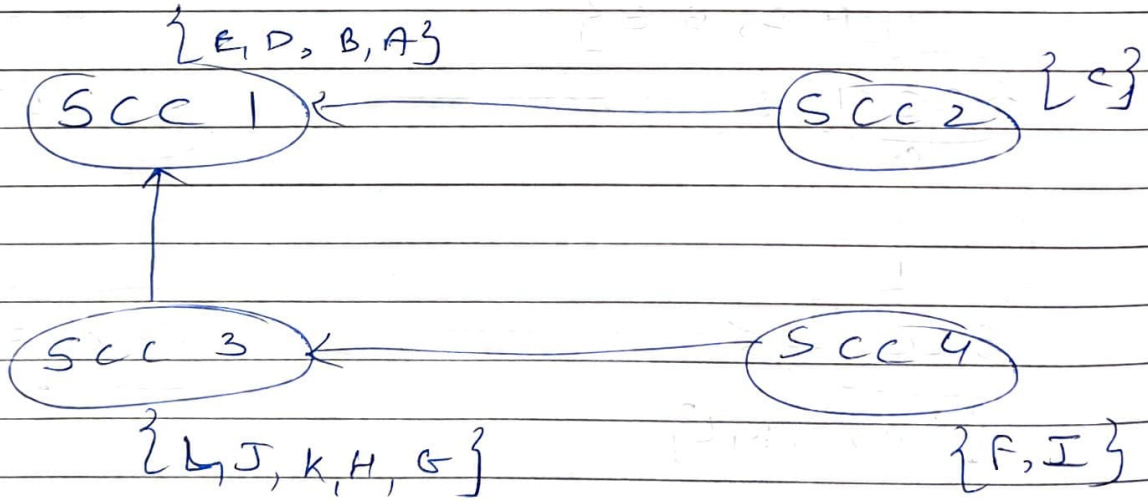
{G, H, J, K, L}

SCC 3

{F, I}

(d)


A
5/6
3/4
9/10
B
D
2/7
11/8
22/23
E
H
G
15/16
14/17
21/24
J
12/1, 9
13/18
11/20
K


{E, D, B, A}
SCC 1
SCC 2
{C}
SCC 3
SCC 4
{L, J, K, H, G}
{F, I}

# Assignment 5: Graphs Exploration (BFS, DFS)

## Abstract:

I have implemented the DFS and BFS algorithms in C++ and printed the nodes travelled both for DFS and BFS graph staring from the lowest element given to us in the main file. In the skeleton code provided, an Adjacency Matrix representation is used for the graphs. The nodes are represented with unique IDs that are integer values starting from 0 to match the indexes in the Adjacency Matrix.

## Result:

A. BFS & DFS

```
(base) dakshbhuva@Dakshs-Air cs590_hw5_code % ./graph
BFS output:
 -> 0 -> 1 -> 2 -> 3 -> 5 -> 4 -> 7 -> 6
DFS output:
 -> 0 -> 1 -> 2 -> 5 -> 7 -> 3 -> 4 -> 6 -> 8 -> 9%
```

## Discussion:

**a. Breadth-First Search (BFS)**

- I have implemented the pseudo-code as given on the lecture slides for Breadth-First Search with some minor changes.
- The time-complexity for BFS algorithm is O (V + E) where V is vertices and E is edges.

**b. Depth-First Search (DFS)**

- I have implemented the pseudo-code as given on the lecture slides for Depth-First Search with some minor changes.
- The time-complexity for DFS algorithm is O (V + E) where V is vertices and E is edges.

## Conclusion:

Thus, the Breadth-First Search and Depth-First Search algorithm is an algorithm for searching a tree data structure for a node that satisfies a given property.