1. **Problem Description**: We have a stack of n textbooks that need to be sorted, such that the frontmatter of each textbook is facing up. For example, initially, we may have the following stack (n=7):

1	0
4	1
5	1
7	0
3	1
2	1
6	0

where the first number in each row shows the desired order of the textbook, and the second number shows whether the cover is frontmatter of the textbook is facing up (1) or facing down (0).

The goal is to sort the textbooks so that they form the following stack:

1	1
2	1
3	1
4	1
5	1
6	1
7	1

However, to sort the stack, we are only allowed to select one of the textbooks, lift that textbook and all textbooks on top of it,[1] and flip them altogether. For example, assume that we select textbook 7 in the following stack:

1	0
4	1
5	1
7	0
3	1
2	1
6	0

Flipping that textbook and everything above it will give us the following stack of textbooks:

7	1
5	0
4	0
1	1
3	1
2	1
6	0

[1] The top textbook has no textbook on top of it, so in case it is selected, it will be the only textbook that flips.

We wish to start from an arbitrary stack, and do a finite number of flips and sort the textbooks.

2. The goal of this Lab is that you write a program that takes an initial stack of textbooks of size n, and sorts them using Depth-First Search and Breadth-First Search. We have provided a skeleton code with extensive comments in the README.md file that you should read. Please use the skeleton code and submit your code to the GitHub repository that will be provided to you.

   Here is how your code will be graded:

   - General soundness of DFS and BFS code: 2×30 pts.
   - Passing multiple test cases: 2×20 pts.

3. (20 pts **Extra Credit**) For $n = 1, 2, 3, 4, 5$, generate all $2^n \times n!$ possible initial book stacks and solve the problem starting from each of them using DFS and BFS. For each n, calculate the average number of flips needed when using DFS and BFS, over all possible initial textbook stacks. For example, when $n = 3$, there are $2^3 \times 3! = 48$ possible initial textbook stacks. For each initial textbook stack, calculate how many flips are needed to sort the textbooks for each of the DFS and BFS algorithms, and calculate the average number of flips over all 48 possible initial configurations for each of DFS and BFS algorithms, and report the average for each n and for each algorithm in a table. Analyze and interpret the results.

   Note 1: as there may be more than one solution for each initial textbook stack, consider only one solution (the first solution your algorithms encounter) for each initial configuration when calculating averages.

   Note 2: You may need to submit the results of the extra credit part separately. Instructions for submission will be given later.