CS-2009 Design and Analysis of Algorithms, Spring-2024
Project

Due Date and Time: 30th April 2024 (2:25 pm) **Weight: 10%**

Instructions:

1. Late submission will not be accepted.
2. Project can be done in a group. Max. group size is 2 members. Cross-section group is allowed. In case of a group size of 2 members, each member is required to solve both problems of a part. In case of group size of 1 member, the student is required to solve all problems of both parts. Carefully form your group. Complaints related to group members not performing their tasks will not be entertained.
3. Groups are required to list their information on the following spreadsheet latest by the submission deadline (you are required to lock the cells on the spreadsheet related to your group): https://docs.google.com/spreadsheets/d/1YC4_50ie-x9u9slYXCAR41ffbDKklY_GNhZs4g1NIjI/edit?usp=sharing
4. Changes in the groups are possible till the deadline. The schedule of demonstrations will be made on the basis of groups' information submitted till the deadline.
5. Only the listed student (on the spreadsheet) will give the demo. related to a part, i.e., other member of the group (if any) cannot give demo. on behalf of the listed group member. The other member of the group must not be present at the time of demo. List yourself on the spreadsheet under both problems even if you are the only member in the group.
6. Secured project marks of each group member = (Secured marks in the assigned problem(s) × 5 / Total marks of the assigned problem(s)) + (group marks in both problems × 5 / 400).
   For example, if there are 2 members in a group; the 1st member secures 120 marks out of 200 marks points and the 2nd group member secures 80 marks out of 200 marks, then the 1st group member will obtain (120×5/200 + 200×5/400=5.50) points out of the 10 points of the project and the 2nd group member will obtain (80×5/200 + 200×5/400=4.50) points out of the 10 points of the project.
7. There will be no credit if the given requirements are changed.
8. Your solution will be evaluated in comparison with the best solution. ***There will be no credit for a non-polynomial time solution.***
9. Plagiarism in any problem may result in zero marks in the whole project regardless of the percentage plagiarized. If you use a website link (URL) for help, provide it in your project report. You should have a complete understanding of the solution provided by you and should be able to fully demonstrate it during evaluation.
10. Implement your solutions using anyone of these programming languages C++/Java/Python.
11. Additional datasets can be used to test your work so make a generic solution of each problem.
12. You are required to submit the source code files only (and not the complete project folder).
13. Submit the PDF file of a "single" integrated project report having the algorithms (in pseudocode form) designed for the project and asymptotic time complexity analysis of your algorithms. Each group member will contribute to the report related to his/her allocation of problem(s). Handwritten reports will not be accepted. *The report will constitute 20% weight of the project*.
14. Each group member will submit the source code file(s) of the problem(s) allocated to him/her. However, the single integrated project report will be submitted by both group members. So each group member will submit a zipped folder having; i) source code file(s), ii) PDF of the project report. *You are required to double-check the zipped folder that you are going to submit. Excuses like corrupt zipped folder, submitting a wrong or earlier version, etc., will not be accepted.*

Part A

Problem 1: [100 Marks; Weight = 2.5%]

Each type of structure should consist of 2 or more stages. Each stage consists of a natural number of blocks. No two stages are allowed to be at the same height. Each stage must be lower in height than the previous one. All stages must contain at least one block. The height of a stage is defined by the number of blocks that make up that stage.

For example, when N = 3, you have only 1 choice of how to build the structure, with the first stage having a height of 2 and the second step having a height of 1 as follows:

□
□□
21

When N = 4, you still only have 1 structure choice:
□
□
□□
31

But when N = 5, there are two ways you can build a structure from the given blocks.

The two staircases can have heights (4, 1) or (3, 2), as shown below:
□
□
□
□□
41

□
□□
□□
32

Write a C++/Java/Python program that takes a positive integer n and returns the number of different number of structures that can be built from exactly n blocks, where $3 \leq n \leq 200$.

Test Case	N	Number of structures	Marks
1	10	9	05
2	100	444792	10
3	150	19406015	15
4	180	141231779	25
5	200	487067745	25
Algorithm in pseudocode form in report			10
Correct time--complexity analysis of the algorithm			10

Problem 2: [100 Marks; Weight = 2.5%]

Following requirements must be fulfilled in the report.

1. Describe the space of sub-problems, most notably by providing clear notation that identifies the parameterization of sub-problems that will be used, and how the overall goal is reflected in those sub-problems. Make sure to justify the optimality substructure condition that is required for dynamic programming.

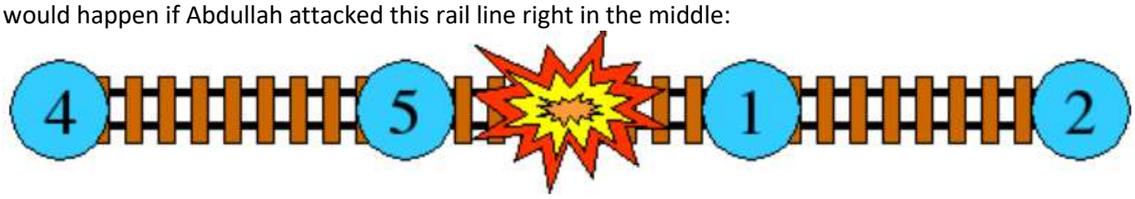
2. Give a clear and concise recursive formula that can be used to compute the values of the sub-problems (including any relevant base cases).
3. If using a bottom-up implementation of dynamic programming, in what order would you solve the sub-problems?
4. What is the asymptotic running time and space usage of the overall algorithm, in terms of the relevant problem parameters?

Problem Description

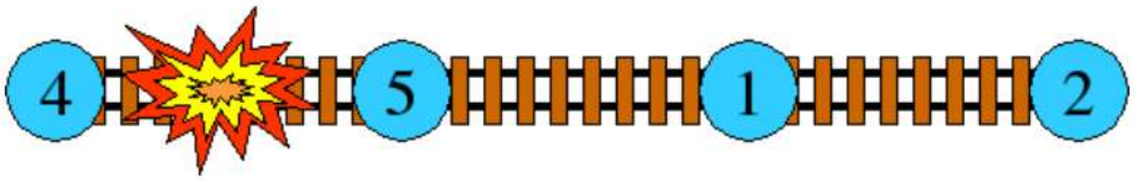
Abdullah is a freedom fighter, fighting against the occupant forces. As a part of his fighting strategy, his primary target is the destruction of railroads.

In this problem, the task is to assist Abdullah to make best use of his limited resources. He has some information available from intelligence wing of his organization. First, the rail line is completely linear which means that there are no branches, no spurs. Next, the Intelligence wing has assigned a Strategic Importance to each depot which is an integer from 1 to 5. Moreover, a depot is of no use on its own, it only has value if it is connected to other depots. The Strategic Value of the entire railroad is calculated by adding up the products of the Strategic Values for every pair of depots that are connected, directly or indirectly, by the rail line. Consider the following railroad:

The strategic value of above railroad is computed as: 4*5 + 4*1 + 4*2 + 5*1 + 5*2 + 1*2 = 49.
Now, suppose that Abdullah only has enough resources for one attack. He cannot attack the depots themselves because they are too well defended. He must attack the rail line between depots, in the middle of the desert. Consider what would happen if Abdullah attacked this rail line right in the middle:



The Strategic Value of the remaining railroad is 4*5 + 1*2 = 22. But, suppose Abdullah attacks between the 4 and 5 depots:



The Strategic Value of the remaining railroad is 5*1 + 5*2 + 1*2 = 17. This is the Abdullah's best option.
Given a description of a railroad and the number of attacks that Abdullah can perform, figure out the smallest Strategic Value that he can achieve for that railroad.

Input
There will be several data sets. Each data set will begin with a line with two integers, n and m. n is the number of depots on the railroad ($1 \le n \le 1000$), and m is the number of attacks Abdullah has resources for ($0 \le m < n$). On the next line will be n integers, each from 1 to 5, indicating the Strategic Value of each depot in order. End of input will be marked by a line with n=0 and m=0, which should not be processed.
Output
For each data set, output a single integer, indicating the smallest Strategic Value for the railroad that Abdullah can achieve with his attacks. Output each integer in its own line.

Sample Input
4 1
4 5 1 2
4 2
4 5 1 2

0 0

Sample Output
17
2
For demo. a different input can be given to test your solution.

Part B

Problem 3: [100 Marks; Weight = 2.5%]

In the Advanced Data Structures course, Laiba, a diligent university student, received a challenging assignment in informatics. The task at hand is to manage an array A of length n by dividing it into k segments where k>1. The goal is to ensure that each segment shares the same property: they all have the Minimum Not Presented Number (MNPN).

The MNPN, a concept crucial for data analysis, refers to the smallest non-negative integer that does not appear within the segment.

Laiba seeks assistance in determining whether such a division is possible. If feasible, Laiba needs to identify a suitable division. However, if no such division exists, Laiba should report -1.

For instance, in one scenario, Laiba could divide the array into segments such that the MNPN within each segment remains consistent. This entails careful selection of segment boundaries to ensure the property holds true. Laiba eagerly awaits guidance to navigate through this intriguing computational challenge.

input: The first line of input has integer N, the next line contains N integers.

output: output the number of segments followed by the range of the segments i.e. the starting and ending indices of segments.

Input	Output
5 1 2 3 4 5	2 1 1 2 5
5 0 1 2 1 0	-1
8 0 1 7 1 0 1 0 3	2 1 5 6 8

For test case 1, we can note that the two segments do not have 0 in them.

for test case 2, no such division can be done.

Problem 4: [100 Marks; Weight = 2.5%]

In the world of solo leveling, a new S class Gate has emerged, attracting the attention of Laiba. With excitement, Laiba decides to host a rigorous training session for Hunters. A total of **M*N** hunters arrive at Laiba's session, forming **M** rows of equal size, each containing exactly n individuals. The adventurers are numbered from 1 to n in each row, following the order from left to right.

Eager to assemble an unbeatable team, Laiba plans to select members with a strategic approach. The selection process involves choosing adventurers from left to right, ensuring that the column index of each chosen member (excluding the first column) is strictly greater than the column index of the previously chosen hunter. To maintain fairness and balance,

Laiba aims to avoid consecutive selections from the same row. The first hunter can be chosen from any of the **M * N** participants without any additional constraints. The team can accommodate no more than N members.

You are provided with the power level of each hunter, write an optimal solution of complexity O(N*M) that calculates the maximum power that the newly arranged team can have.

input: The first line in each test case contains two integers, M (rows) and N columns, followed by M rows each having N values.

Input	Output
1 10 **1 2 3 4 5 5 4 3 2 1**	**5**
2 3 **1 10 35** **10 3 11**	**45**
2 5 **11 5 7 9 5** **7 10 3 6 7**	**39**

For test case 1, we have only 1 row so we can only pick 1 value that maximizes the power.

For test case 3, **11 + 10 + 7 + 6 + 5 = 39**

11	5	**7**	9	**5**
7	**10**	3	**6**	7