Flip The Bits and Fall Down

Akshath Suresh R Rao
Dept. Of Computer Science (Artificial Intelligence)
Amrita Vishwa Vidya Peetham
amenu4aie20009@am.students.amrita.edu

Natte Sai Bharath
Dept. Of Computer Science (Artificial Intelligence)
Amrita Vishwa Vidya Peetham
amenu4aie20052@am.students.amrita.edu

Raja Pavan Karthik
Dept. Of Computer Science (Artificial Intelligence)
Amrita Vishwa Vidya Peetham
amenu4aie20060@am.students.amrita.edu

Seshabhattar Srivastav
Dept. Of Computer Science (Artificial Intelligence)
Amrita Vishwa Vidya Peetham
amenu4aie20065@am.students.amrita.edu

I. Introduction

In this project we have solved two problems taken from codeforces. Both problems with Difficulty 1200. The code was written in Python. Problem 1 is named Flip the Bits and Greedy algorithm was used to solve the problem. Problem 2 is named Fall Down and DFS was used to solve it. Through this project we have learnt to implement algorithms efficiently and also solve challenging problems in competitive programming platforms.

II. Problem Statements

1) Flip The Bits:: There is a binary string a of length n. In one operation, you can select any prefix of a with an equal number of 0 and 1 symbols. Then all symbols in the prefix are inverted: each 0 becomes 1 and each 1 becomes 0.

2) Fall Down:: There is a grid with n rows and m columns, and three types of cells:

- *empty cell, denoted with '.'.*
- *A stone, denoted with '*'.*
- *An obstacle, denoted with the lowercase Latin letter 'o'.*

All stones fall down until they meet the floor (the bottom row), an obstacle, or other stone which is already immovable. (In other words, all the stones just fall down as long as they can fall.)

Simulate the process. What does the resulting grid look like?

III. Algorithms

A. Depth First Search

Depth-first search is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking. We used this algorithm for the *Fall Down* problem.

B. Greedy Algorithm

A greedy algorithm works by choosing the best possible answer in each step and then moving on to the next step until it reaches the end, without regard for the overall solution. It only hopes that the path it takes is the globally optimum one, but as proven time and again, this method does not often come up with a globally optimum solution. In fact, it is entirely possible that the most optimal short-term solutions lead to the worst possible global outcome. This algorithm was used to solve the *Flip the Bits* problem.

IV. Flip The Bits

There is a binary string a of length n. In one operation, you can select any prefix of a with an equal number of 0 and 1 symbols. Then all symbols in the prefix are inverted: each 0 becomes 1 and each 1 becomes 0.

A. Input-Output

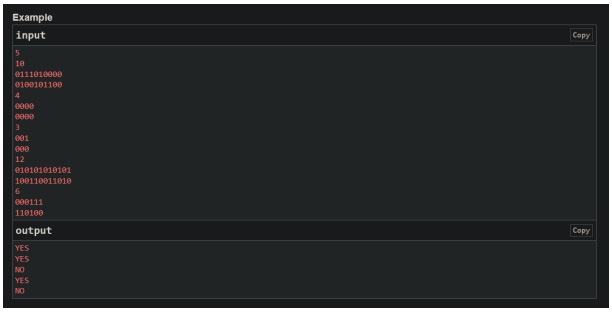


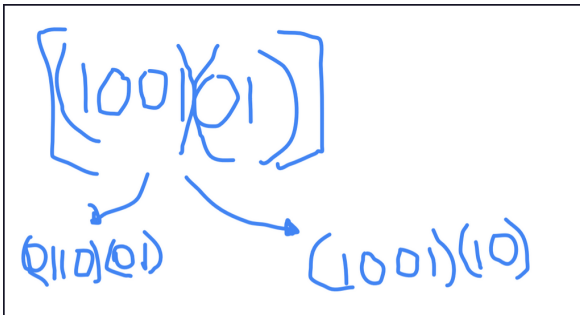
Fig. 1. Example of Input-output

- The first line contains an integer t which signifies the number of test cases
- The second line contains integer n which specifies the length of two strings a and b
- The next two lines contains strings a and b of length n given in binary format.

The output is either YES or NO

B. Theory

Let us take an example string $a = 100101$
In this string we can divide a into two sections which contain equal number of 0s and 1s. These two prefixes can be considered for flipping which means that there are two possibilities in which we can flip the bits. We can also take the entire string as prefix since number of 0s and 1s are same.



So, we can take a counter from left to right. Whenever the number of 0s and 1s are same, we separate it as one section.

C. Implementation

- We have to take the number of test cases
- We ask for the input size of the strings A and B
- We have to take String A of length n (only binary)
- We have to take String B of length n (only binary)
- It will check for the prefix and if it found a prefix it performs the flip
- It will be printed "YES" if string *a* can be converted into string *b* if not "NO"

V. Fall Down

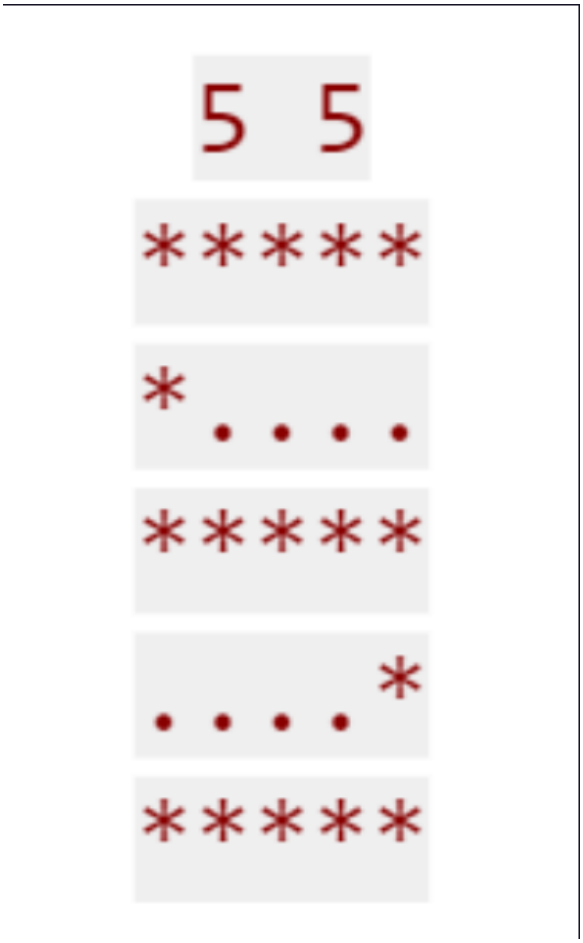
In this problem, we must find the resulting grid.

A. Theory

Here If we see the input has been given the size of 5 x 5 In the next line there is a matrix of 5 x 5 size. Now the process begins by checking the symbols In the first column there are no obstacles and there is only one space left in 4th row, so we can move the above stones to the down one by one We repeat this process for the remaining rows and columns until the stones fall down.

B. Implementation

- Enter the total no of test cases want to perform.
- Enter the size of matrix n x m.
- Input the patterns which should be only containing these three symbols (* . o).
- Traverse at each column and every row



- Check for obstacles and change the position of the stone to final position.
- Print the matrix

VI. Question Links

- https://codeforces.com/problemset/problem/1669/G
- https://codeforces.com/contest/1504/problem/B