June 14, 2025

CPP DRILL SESSION

* Steps to keep in mind when solving a problem:

→ Read the problem statement.

→ Visualization, visualizing flow from input → output.

→ Problem solving, toying to solve the problem using specific test cases to build strategy | logic.

→ Formulate, generalize the strategy for all test cases and building a pseudocode. (Thinking of what data structure is suitable, how many loops are needed, what is the time complexity)

→ Code

→ Debug

PROBLEM 1 https://atcoder.jp/contests/abc104/tasks/abc104_b

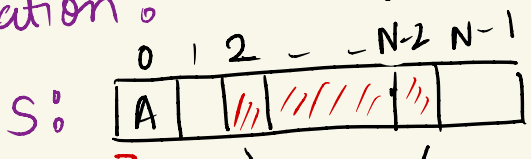
Problem Statement

You are given a string S. Each character of S is uppercase or lowercase English letter. Determine if S satisfies all of the following conditions:

- The initial character of S is an uppercase `A`.
- There is exactly one occurrence of `C` between the third character from the beginning and the second to last character (inclusive).
- All letters except the `A` and `C` mentioned above are lowercase.

Property to use : String is simply a character array.

Visualization :



① is fixed && ② C can only occur once b/w S[2] _ _ S[N-2] && ③ All characters except A & C from S[1] to S[N-1] should be lowercase

① && ② && ③ ⇒ VALID

otherwise ⇒ INVALID

Solution link https://atcoder.jp/contests/abc104/submissions/66713397

Problem 2: https://atcoder.jp/contests/abc075/tasks/abc075_b

Problem Statement

You are given an $H \times W$ grid.

The squares in the grid are described by H strings, $S_1, \ldots, S_H$.

The j-th character in the string S_i corresponds to the square at the i-th row from the top and j-th column from the left $(1 \leq i \leq H, 1 \leq j \leq W)$.

`.` stands for an empty square, and `#` stands for a square containing a bomb.

Dolphin is interested in how many bomb squares are horizontally, vertically or diagonally adjacent to each empty square. ①

(Below, we will simply say "adjacent" for this meaning. For each square, there are at most eight adjacent squares.)

He decides to replace each `.` in our H strings with a digit that represents the number of bomb squares adjacent to the corresponding empty square. ②

Print the strings after the process.

This can seem very complex to read. So, for problems like these, we can go to the given test cases with inputs & outputs in order to understand the problem statement better.

Sample Input 1 Copy

```
3 5
.....
.#.#.
.....
```

Sample Output 1 Copy

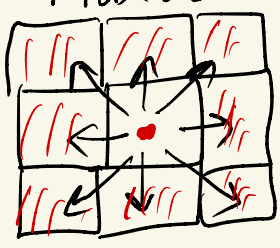
```
11211
1#2#1
11211
```

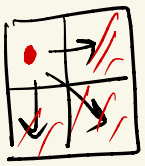
For example, let us observe the empty square at the first row from the top and first column from the left.
There is one bomb square adjacent to this empty square: the square at the second row and second column.
Thus, the `.` corresponding to this empty square is replaced with `1`.

(1) A bomb square is a square with '#'

(2) For every square that contain a (.), we have to calculate how many bomb square or '#' are surrounding it from every side.

Max: 8 neighbour





Min: 3

(3) Replace the (.) square with count of bomb square that surrounds it.

* Brute force approach can be:

→ Loop through the matrix (2 loops)

→ For every a[i][j], go through each of its neighbour [depending on position it can vary from Max: 8 to Min: 3] and count the cells containing '#'.

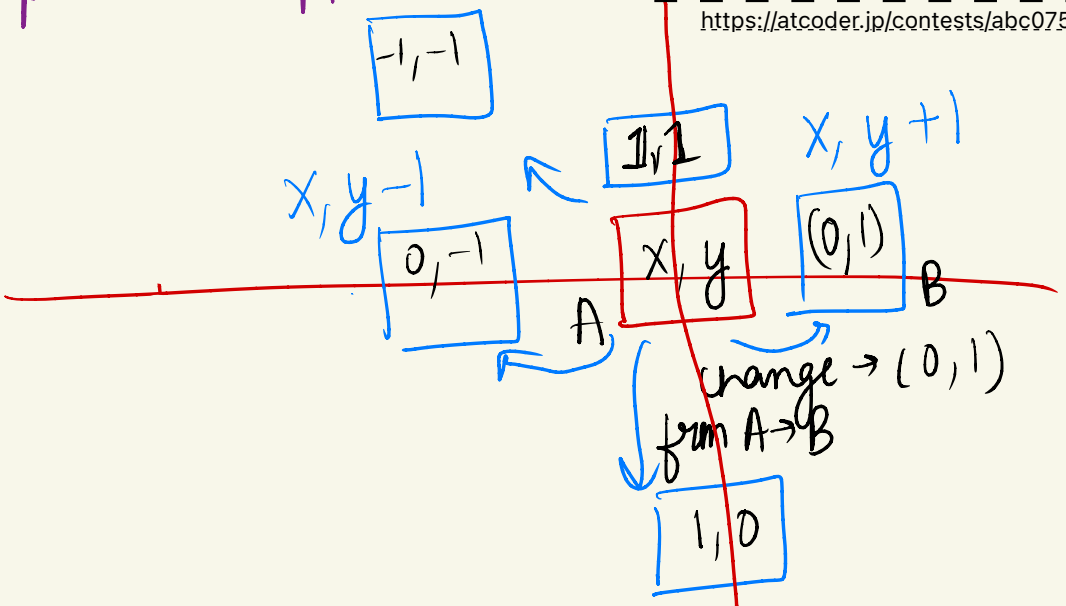
Have to build direction() & count() logic

↓ trace neighbours ↓ count no. of '#'

* Optimized approach:

Solution link

https://atcoder.jp/contests/abc075/submissions/66714266

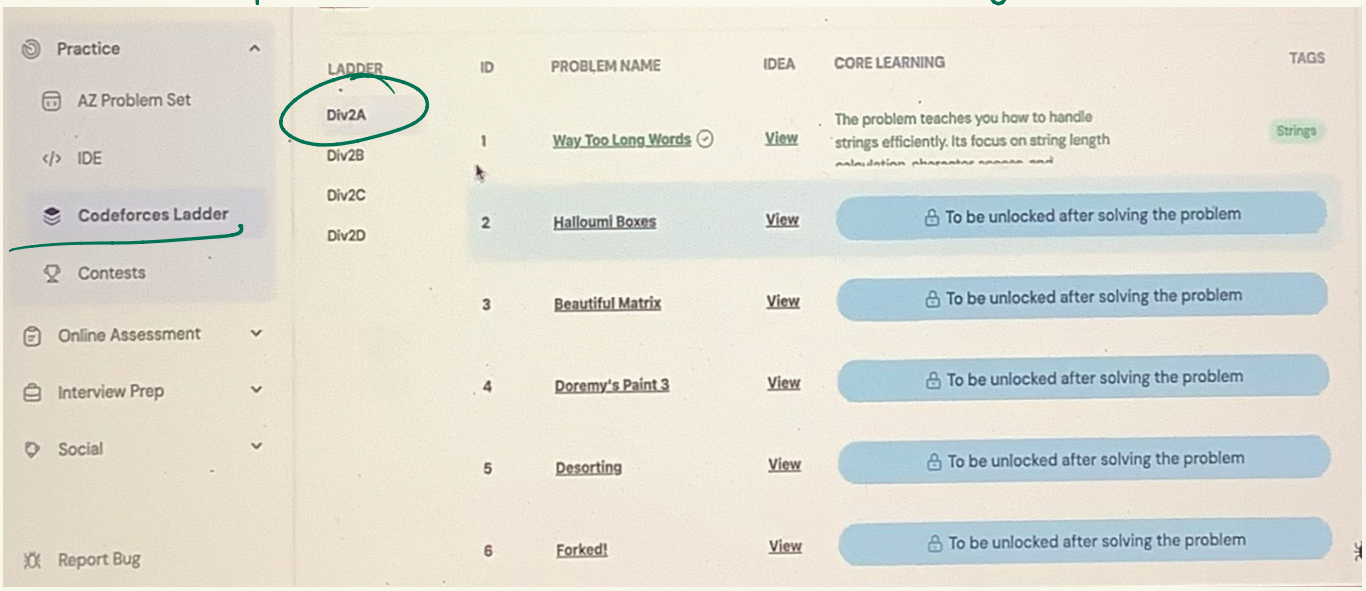


Use direction vectors derived from above to trace neighbors

dx[] (change in x) = {0, 1, 1, 1, 0, -1, -1, -1}

dy[] (change in y) = {1, 1, 0, -1, -1, -1, 0, 1}

Side Note: To get good at implementation, try solving Div2A problems (50 problems should get one started)



Problem 3

https://codeforces.com/gym/102697/problem/040

Sudoku is a popular number puzzle involving placing digits (1-9) in a 9 by 9 grid. A sudoku is considered `valid` if all of the following are true:

-All digits are integers from 1 to 9

-No number occurs in the same row twice

-No number occurs in the same column twice

-No number occurs in the same 3 by 3 box twice.

Given a sudoku board, print whether or not it is valid.

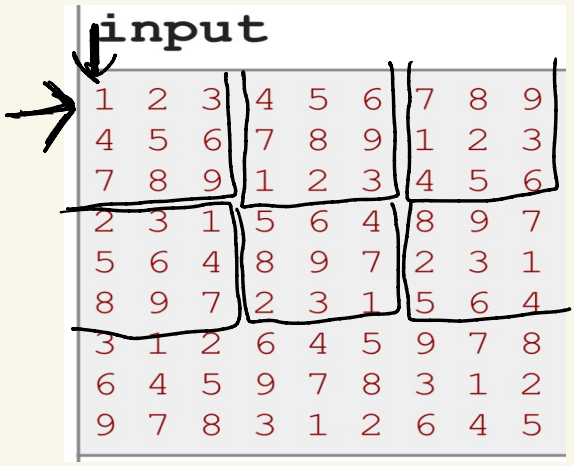
3 checks need to be done in order to check for above conditions.

→ per row

→ per column

→ 3X3

check(row, col, box)

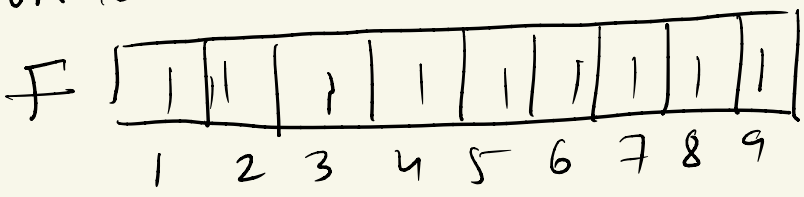


AlgoZenith

→ check from 1 — 9, no. appeared once

use frequency array

If 9 numbers (diff) are put in freq. array, it will look like:



→ Meaning the index of the array appeared only once in 3X3

If any one of them is not 1 then its wrong. (INVALID)

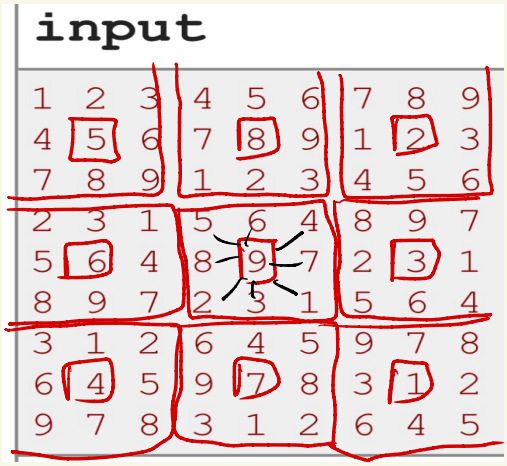
→ Tracing through row & col

If i is same, then its a row.

If j is same, then we are talking column.

→ Tracing 3X3 box

Go to centre cells one by one and then use direction vectors dx[] & dy[] discussed earlier to trace each box.



Solution link (need slack login)

https://azpremiumb11.slack.com/archives/C09026YB5ME/p1749894836976609

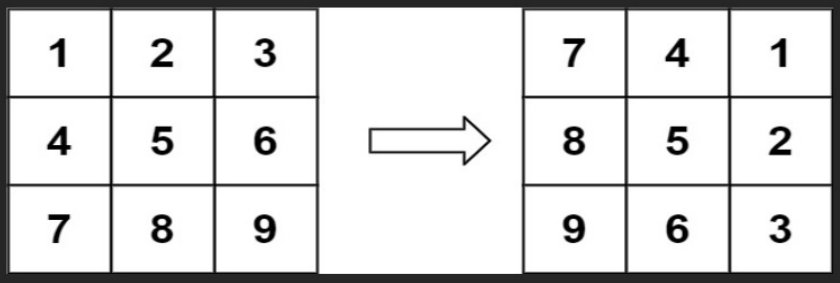
AlgoZenith

Problem 4: https://leetcode.com/problems/rotate-image/description/

You are given an `n x n` 2D `matrix` representing an image, rotate the image by **90** degrees (clockwise).

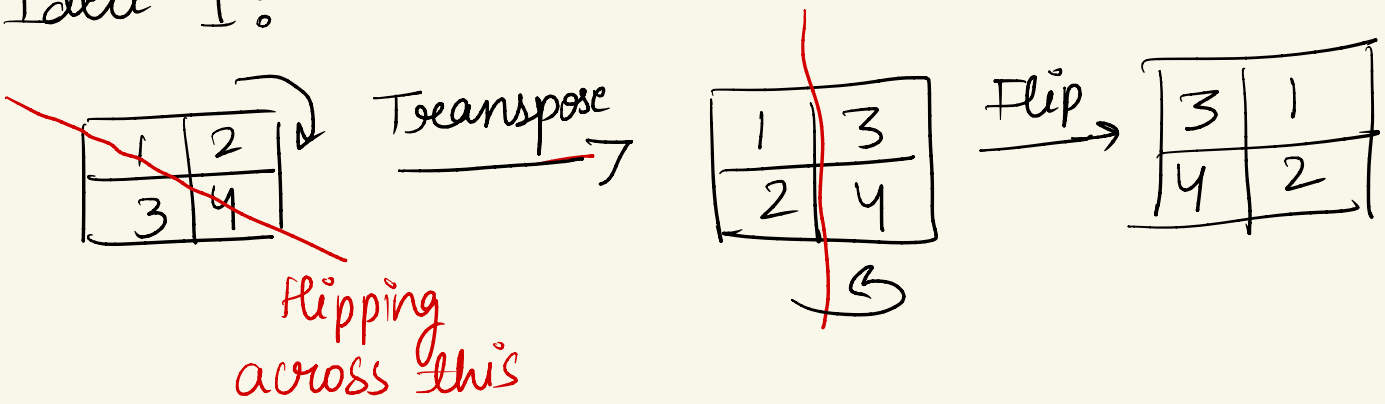
You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

Example 1:




```
Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]
Output: [[7,4,1],[8,5,2],[9,6,3]]
```

Idea 1:



Idea 2:

Swap positions manually.

Solution:

```java
class Solution {
    public void rotate(int[][] matrix) {
        // mat transpose
        for(int i=0; i<matrix.length; i++){
            for(int j=i; j<matrix[i].length; j++){
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }

        // rev row
        for(int i=0; i<matrix.length; i++){
            for(int j=0; j<matrix.length/2; j++){
                int temp = 0;
                temp = matrix[i][j];
                matrix[i][j] = matrix[i][matrix.length-1-j];
                matrix[i][matrix.length-1-j] = temp;
            }
        }

    }
}
```

Swapping for Transpose

Problem 5: Solution link

https://atcoder.jp/contests/abc054/tasks/abc054_b

https://atcoder.jp/contests/abc054/submissions/66883044

Problem Statement

You are given an image A composed of N rows and N columns of pixels, and a template image B composed of M rows and M columns of pixels.
A pixel is the smallest element of an image, and in this problem it is a square of size 1×1.
Also, the given images are binary images, and the color of each pixel is either white or black.

In the input, every pixel is represented by a character: `.` corresponds to a white pixel, and `#` corresponds to a black pixel.
The image A is given as N strings $A_1, \ldots, A_N$.
The j-th character in the string A_i corresponds to the pixel at the i-th row and j-th column of the image A $(1 \leq i, j \leq N)$.
Similarly, the template image B is given as M strings $B_1, \ldots, B_M$.
The j-th character in the string B_i corresponds to the pixel at the i-th row and j-th column of the template image B $(1 \leq i, j \leq M)$.

Determine whether the template image B is contained in the image A when only parallel shifts can be applied to the images.

AlgoZenith