## Difficulty

Easy

## Categories

Observations, Ad-Hoc, Arrays, Sorting, Implementation

## Problem 1: Number Sequences

Given an array of 3 integers in random order. The question labels the smallest of the array values as A, second largest in the array of 3 integers as B and largest of the array values as C. I have to print the values in a specific order as explained by a string of 3 uppercase letters (A,B,C).

For example:
18 9 7
CAB
Expected Output: 18 7 9
Explanation: I was given three values. There were not given in any order. A got the value of 7, B became 9 and C became 18, following the description. I had to print them in the order specified by the following string: CAB. So, the output became: 18 7 9

## Explanation of the Code

I got these 3 values in an array of size 3. Using the library function of Java, Arrays.sort(), I sorted this array. Now, the value of A or smallest value is array[0], the value of B or the second largest value is array[1] and the value of C or the largest value is array[2].
I traversed the given string, through a loop with looping variable named as i. If the character at index i is C, then I printed the largest value or array[2], followed by a space (because of Output format), if the character at index i is B, then I printed the second largest value or array[1] or if the character at index i is A, then I printed the smallest value or array[0].

## Problem Setter's Code

```
import java.util.Arrays;
import java.util.Scanner;

public class Main {

        public static void main(String[] args) {
                Scanner scn=new Scanner(System.in);
                int a = scn.nextInt();
                int b = scn.nextInt();
                int c = scn.nextInt();
        int[] arr=new int[3];
        arr[0]=a;
        arr[1]=b;
        arr[2]=c;
        Arrays.sort(arr);

        String str=scn.next();
        char[] char_arr=str.toCharArray();
        for(int i=0; i<char_arr.length; i++) {
            if(char_arr[i]=='A') {
                System.out.print(arr[0]+" ");
            }else if(char_arr[i]=='B') {
                System.out.print(arr[1]+" ");
            }else if(char_arr[i]=='C') {
                System.out.print(arr[2]+" ");
            }
        }

        scn.close();
    }

}
```

## Difficulty

Medium

## Categories

Observations, Handling indices, Arrays

## Problem 2: Row of Cows

You are given 2 strings of sizes N and M. Let us suppose the strings are JLA and CRUO. You have to view these strings facing each other. So, for the example, the view becomes: A -> L -> J and C <- R <- U <- O. These strings move toward each other and in each second, character of string moving towards left gets swapped with character of string moving towards right. We have to tell the resulting string after T seconds.

## Explanation of the Code

I made following observations in our running example.
At t=0 seconds, the resulting string becomes ALJCRUO.
At t=1 seconds, the resulting string becomes ALCJRUO.
At t=2 seconds, the resulting string becomes ACLRJUO
At t=3 seconds, the resulting string becomes CARLUJO [C would not move further]
At t=4 seconds, the resulting string becomes CRAULOJ [R and J cannot move further]
At t=5 seconds, the resulting string becomes CRUAOLJ [U and J cannot move further]
At t=6 seconds, the resulting string becomes CRUOALJ [O and A cannot move further] [There cannot be any more swaps]
At t>7 seconds, the resulting string becomes CRUOALJ.

There is a relation between index in the resulting string and time t (in seconds).
For example: for the first character J of the first string, the index in the resulting string is N-1, at t=0 seconds.
At t=1 seconds, the index in the resulting string is N-1+1 or 3, as given in ALCJRUO. And at t=x seconds, the position in the resulting string is N-1+x, but N-1+x cannot be greater than last index of resulting string.
In general, for any character of first string at an index i, it's index in the resulting string, after t seconds, is given by N-1+i+t.
Hence, we can directly place the character in that index.
Similarly, for the second string. For any character in the second string at an index i, it's index in the resulting string, after t seconds, is N+i-t.

Also, we know that if the string is JLA and if the character is J, then it will be swapped for t seconds, but if the character is A, then it will swapped for t-2 seconds. So, the general equation of the index of characters of first string, after t seconds, becomes N-1+i+t-i and for the characters of the second string, it becomes N+i-(t-i).

## Problem Setter's Code

```
import java.util.Scanner;

public class Main {

            public static void main(String args[]) {
            Scanner scn=new Scanner(System.in);
            int onelength=scn.nextInt();
            int twolength=scn.nextInt();
            String one=scn.next();
            String two=scn.next();
            int t=scn.nextInt();
            char[] onestr_array=one.toCharArray();
            char[] twostr_array=two.toCharArray();

            char[] thirdarray=new char[onelength+twolength];
            int P=0;
            for(int i=0; i<one.length(); i++) {
                  P=onelength-i-1;
                  P+=Math.min(Math.max(t-i, 0), twolength);
                  thirdarray[P]=onestr_array[i];
            }
            for(int i=0; i<two.length(); i++) {
                  P=onelength+i;
                  P-=Math.min(Math.max(t-i, 0), onelength);
                  thirdarray[P]=twostr_array[i];
            }

            for(char ch: thirdarray) {
                  System.out.print(ch);
            }

      }
}
```

# Difficulty

Hard

## Categories

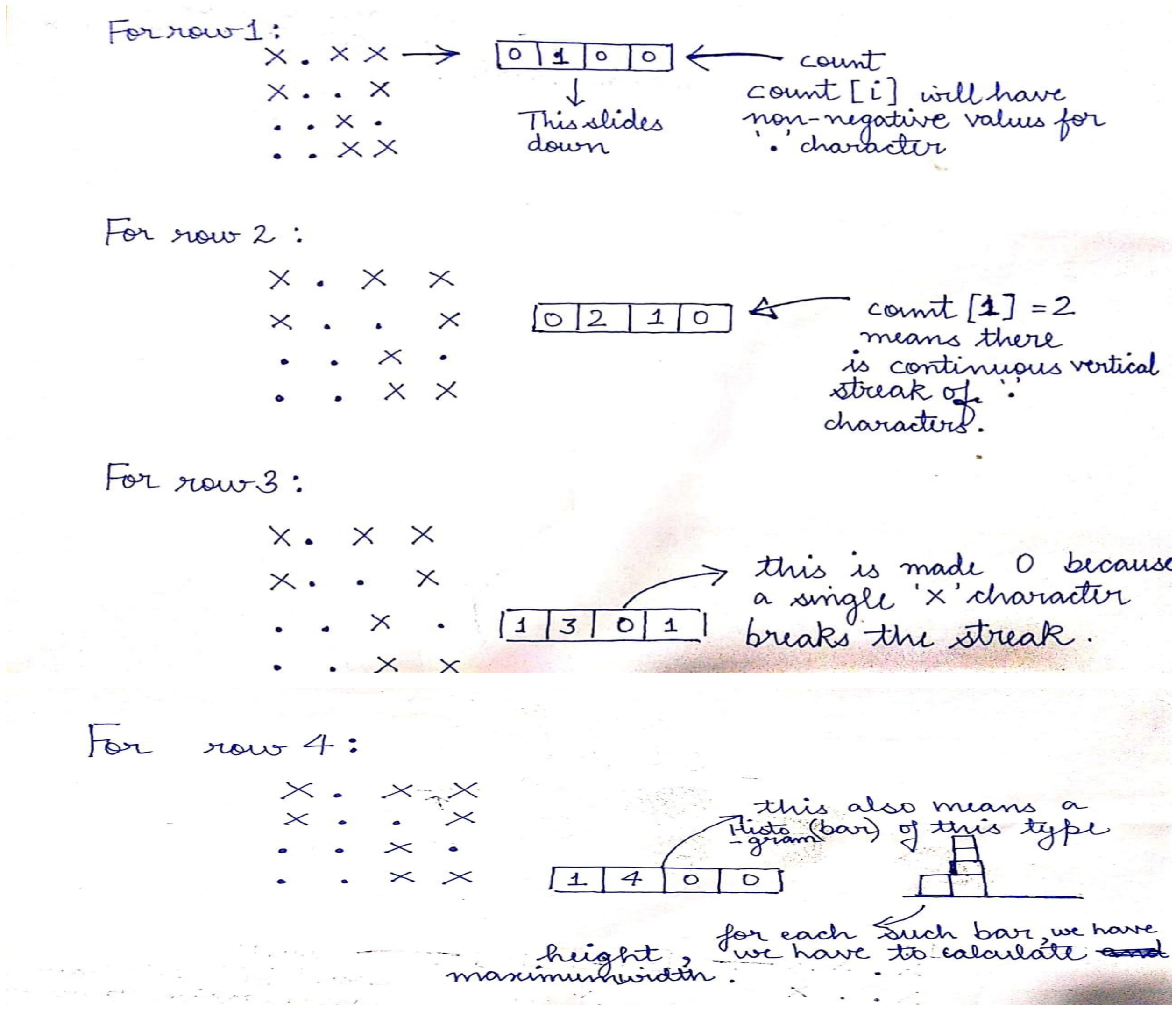
2D arrays, Matrices

## Problem 3: Maximum Invitations

You have to find maximum rectangular matrix of all character ‘.’ in a binary-matrix of 2 characters ‘.’ and ‘x’.

## Explanation of Code

I will use a count[] array, which will slide as explained in the picture below:


For row 1:
count
This slides down
count [i] will have non-negative values for '.' character
For row 2 :
count [1] = 2 means there is continuous vertical streak of '.' characters.
For row 3 :
this is made 0 because a single 'x' character breaks the streak.
For row 4 :
this also means a Histo-gram (bar) of this type
for each such bar, we have we have to calculate ~~and~~ height, maximum width.

Note: A small correction for row 4: count[] will following values = 2 4 0 0. count[0] must be 2 in the figure.

For each row, I will have my value in the count array populated. For each value in the count array, I will check for maximum width rectangle that can be formed from height equal to count[i]. After I have width and height values, I will calculate perimeter.
I will use another variable to find maximum of all the perimeters. My answer will maximum perimeter minus 1 because 1 seat or space will be used by host itself.

## Problem Setter's Code

```
import java.util.*;
public class Main {

    static int peri(int w, int h) {
        return w*2+h*2;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int m = sc.nextInt();
        sc.nextLine();
        char[][] arr = new char[n][m];
        int[] cnt = new int[m];
        int max = 0;
        for (int i=0; i<n; i++) {
            arr[i] = sc.nextLine().toCharArray();
            for (int j=0; j<m; j++) {
                if (arr[i][j]=='.') cnt[j]++;
                else cnt[j] = 0;
            }
            for (int j=0; j<m; j++) {
                int h = cnt[j];
                int w = 1;
                if (h>0) {
                    for (int k=j+1; k<m; k++) {
                        if (cnt[k]<h) break;
                        w++;
                    }
                    for (int k=j-1; k>0; k--) {
                        if (cnt[k]<h) break;
                        w++;
                    }
                }
                max = Math.max(max, peri(w, h));
            }
        }
        System.out.println(max-1);
    }

}
```