# PROBLEMS OF THE DAY – 5

## 1.Maximum sum Rectangle

Given a 2D matrix M of dimensions RxC. Find the maximum sum submatrix in it.

**Example**:

**Input**:

R=4

C=5

M=[[1,2,-1,-4,-20],

[-8,-3,4,2,1],

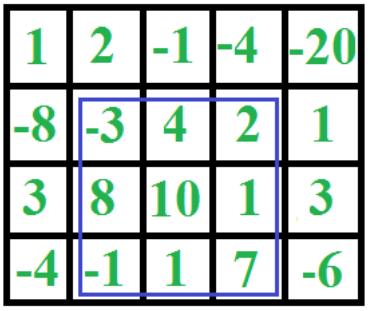
[3,8,10,1,3],

[-4,-1,1,7,-6]]

**Output**:

29

**Explanation**: The matrix is as follows and the blue rectangle denotes the maximum sum rectangle.




**Input**:

R=2

C=2

M=[[-1,-2],[-3,-4]]

**Output**:

-1

**Explanation**: Taking only the first cell is the optimal choice.

Expected Time Complexity:**O(R*R*C)**

Expected Auxillary Space:**O(R*C)**

## 2.Rotate by 90 degree

Given a square matrix of size n x n. The task is to rotate it by 90 degrees in an anti-clockwise direction without using any extra space.

**Examples**:

**Input**: matrix[][] = [[1, 2, 3],

[4, 5, 6]

[7, 8, 9]]

**Output**: Rotated Matrix:

[3, 6, 9]

[2, 5, 8]

[1, 4, 7]

**Input**: matrix[][] = [[1, 2],

[3, 4]]

**Output**: Rotated Matrix:

[2, 4]

[1, 3]

---

## 3.Edit Distance

Given two strings str1 and str2. Return the minimum number of operations required to convert str1 to str2.

**The possible operations are permitted:**

Insert a character at any position of the string.

Remove any character from the string.

Replace any character from the string with any other character.

**Examples**:

**Input**: str1 = "geek", srt2 = "gesek"

**Output**: 1

**Explanation**: One operation is required, inserting 's' between two 'e'.

**Input** : str1 = "gfg", str2 = "gfg"

**Output**: 0

**Explanation**: Both strings are same.

Expected Time Complexity: **O(|str1|*|str2|)** Expected Space Complexity: **O(|str1|*|str2|)**

## 4.Longest valid Parentheses

Given a string str consisting of opening and closing parenthesis '(' and ')'. Find length of the longest valid parenthesis substring.

**Examples** :

**Input**: str = (((

**Output**: 2

**Explaination**: The longest valid parenthesis substring is "()".

**Input**: str = )()())

**Output**: 4

**Explaination**: The longest valid parenthesis substring is "()()".

Expected Time Complexity: **O(|str|)** Expected Auxiliary Space: **O(|str|)**

---

## 5.Palindromic Partitioning

Given a string str, a partitioning of the string is a palindrome partitioning if every sub-string of the partition is a palindrome. Determine the fewest cuts needed for palindrome partitioning of the given string.

**Example** :

**Input**: str = "ababbbabbababa"

**Output**: 3

**Explaination**: After 3 partitioning substrings are "a", "babbbab", "b", "ababa".

**Input**: str = "aaabba"

**Output**: 1

**Explaination**: The substrings after 1 partitioning are "aa" and "abba".

Expected Time Complexity: **O(n*n)** [n is the length of the string str]

Expected Auxiliary Space: **O(n*n)**