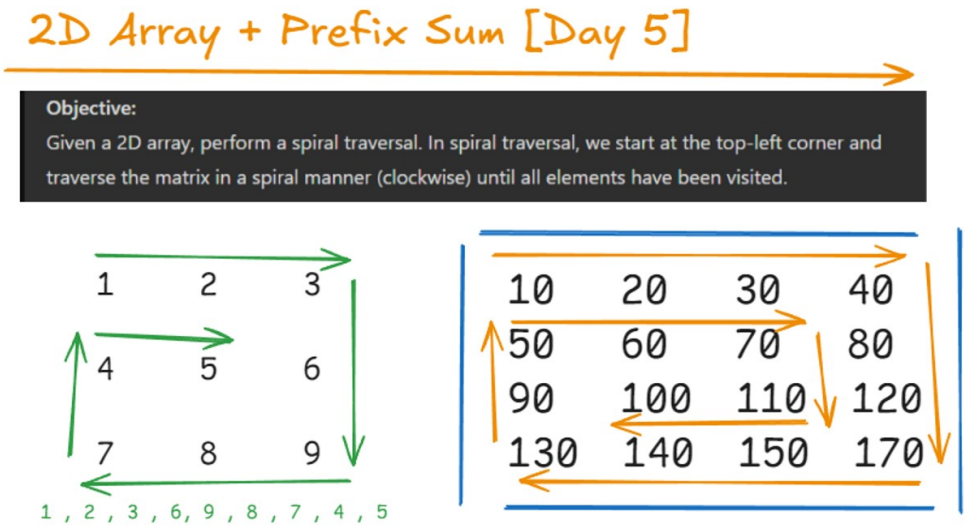
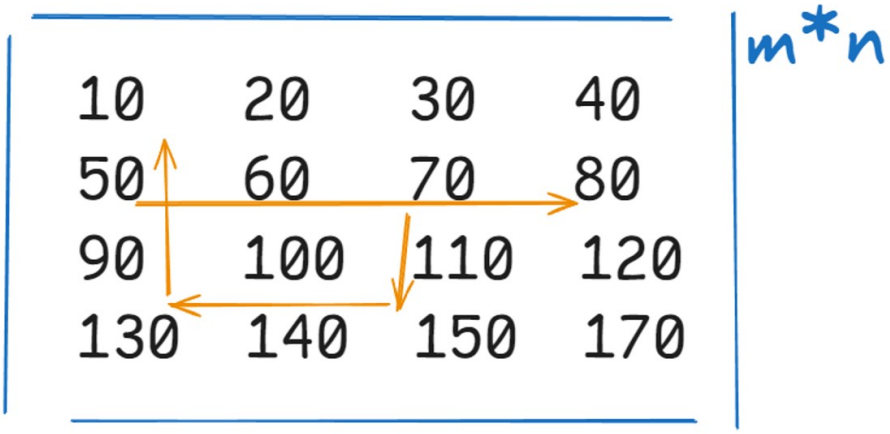
2D Array + Prefix Sum[Day 5]

06 September 2024 19:00





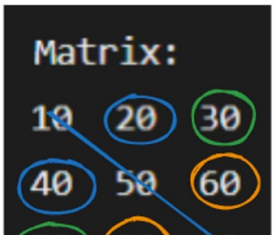
top = 0 , right = n -1;
left = 0 ; bottom = m - 1;

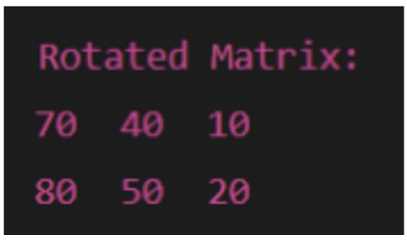
```
Spiral Traversal
10 20 30 40 80 120 170 150 140 130 90 50 60 70 110 100
PS C:\Users\krish\VSCode\Java DSA>
```

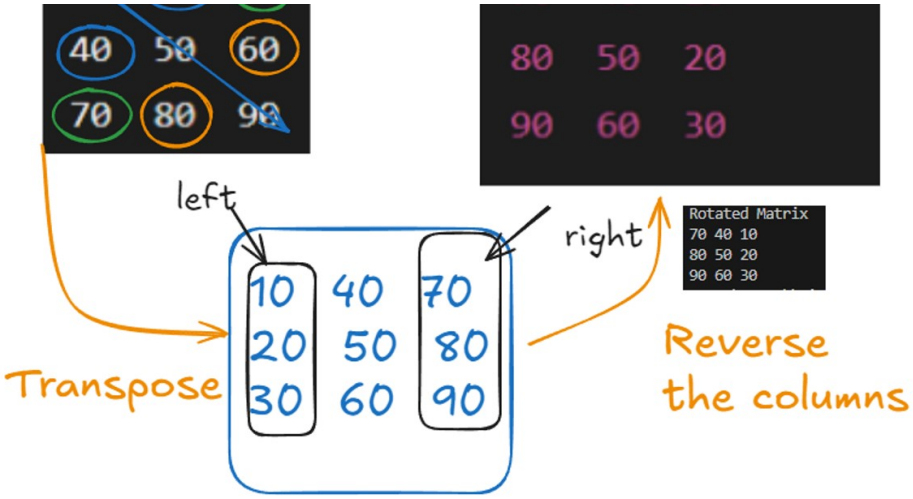
```java
import java.util.Scanner;
public class SpiralMatrixTraversal {

    public static void spiralTraversal(int[][] matrix , int m , int n){
        int top =0;
        int bottom = m -1;
        int left = 0;
        int right = n -1;
        while(top <= bottom && left <= right){
            // Traverse left -> right making top fixed
            for(int i = left; i<=right; i++){
```

```
        while(top <= bottom && left <= right){
            // Traverse left -> right making top fixed
            for(int i = left; i<=right; i++){
                System.out.print(matrix[top][i] + " ");
            }
            top++;
            // Traverse top -> bottom making right fixed
            for(int i = top; i<=bottom; i++){
                System.out.print(matrix[i][right] + " ");
            }
            right--;
            // Traverse right -> left making bottom fixed
            if(top <= bottom){
                for(int i = right; i>=left; i--){
                    System.out.print(matrix[bottom][i] + " ");
                }
            }
            bottom--;
            // Traverse bottom -> top making left fixed
            if(left <= right){
                for(int i =bottom; i>=top; i--){
                    System.out.print(matrix[i][left] + " ");
                }
            }
            left++;
        }
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the number of rows:");
        int m = scn.nextInt();
        System.out.println("Enter the number of columns:");
        int n = scn.nextInt();

        int[][] matrix = new int[m][n];
        System.out.println("Enter the elements of an matrix");
        for(int i = 0 ; i<m ; i++){
            for(int j =0; j<n; j++){
                matrix[i][j] = scn.nextInt();
            }
        }

        System.out.println("Spiral Traversal");
        spiralTraversal(matrix , m ,n);

    }
}
```

- Rotate an 2D array by 90 degree in clockwise direction.







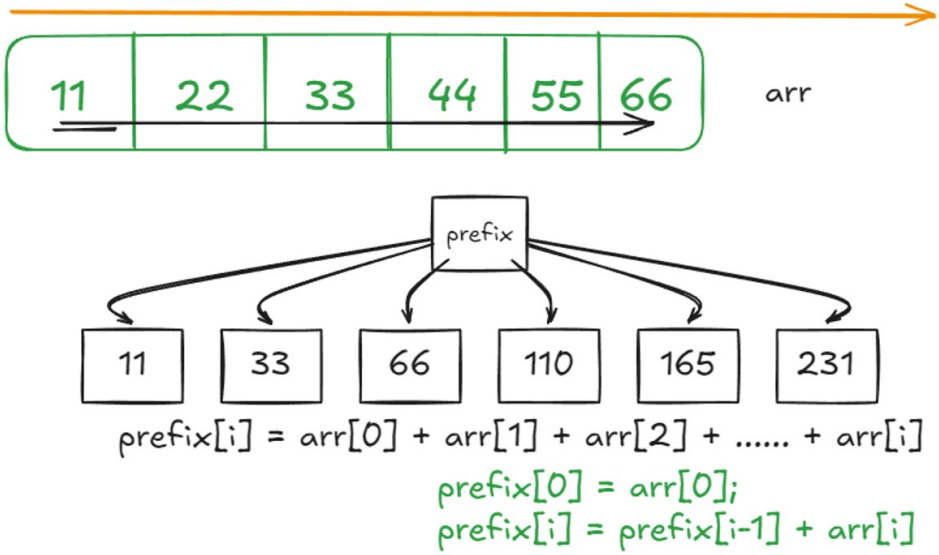
Transpose = Changing row into columns & column into row

arr[i][j] = arr[j][i];
Swapping

```
import java.util.Scanner;
public class RotateMatrix90Degree {
    public static void rotateMatrixBy90(int[][] matrix , int n){
        // Transpose
        for(int i = 0 ; i<n; i++){
            for(int j = i+1; j<n ; j++){
                // Swap the matrix Element
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }
        // Reverse

        for(int i =0; i<n ; i++){
            int start = 0;
            int end = n -1;

            while(start < end){
                int temp = matrix[i][start];
                matrix[i][start] = matrix[i][end];
                matrix[i][end] = temp;

                start++;
                end--;
            }
        }
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the size of the matrix:");
        int n = scn.nextInt();

        int[][] matrix = new int[n][n];

        System.out.println("Enter the elements of an Matrix : ");
```

```
        int[][] matrix = new int[n][n];

        System.out.println("Enter the elements of an Matrix : ");
        for(int i=0; i<n; i++){
            for(int j = 0; j<n; j++){
                matrix[i][j] = scn.nextInt();
            }
        }

        rotateMatrixBy90(matrix , n);

        System.out.println("Rotated Matrix");
        for(int i =0; i<n ; i++){
            for(int j =0 ; j<n; j++){
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Prefix Sum

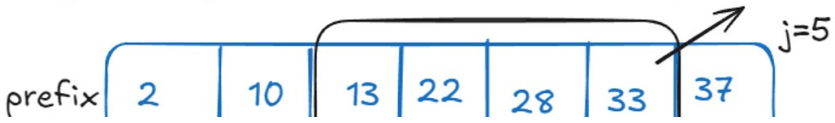


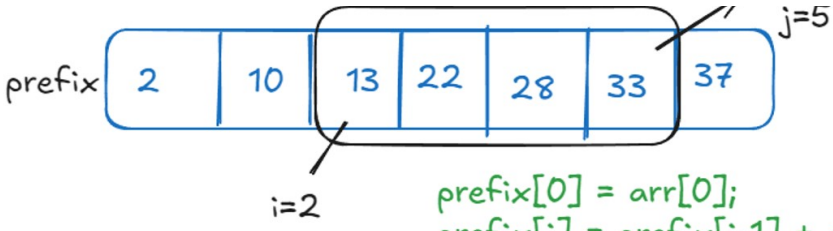
Problem 1: Find the Sum of a Subarray

Problem Statement: Given an array arr[] and two indices i and j, find the sum of the elements between the indices i and j (inclusive).

Example:

Input: arr[] = {2, 8, 3, 9, 6, 5, 4}
Query: (i = 2, j = 5)
Output: 23 (because 3 + 9 + 6 + 5 = 23)





prefix[0] = arr[0];
prefix[i] = prefix[i-1] + arr[i]

SUM(2,5) = 33 - 10 = 23
SUM(i,j) = Prefix[j] - prefix[i-1];

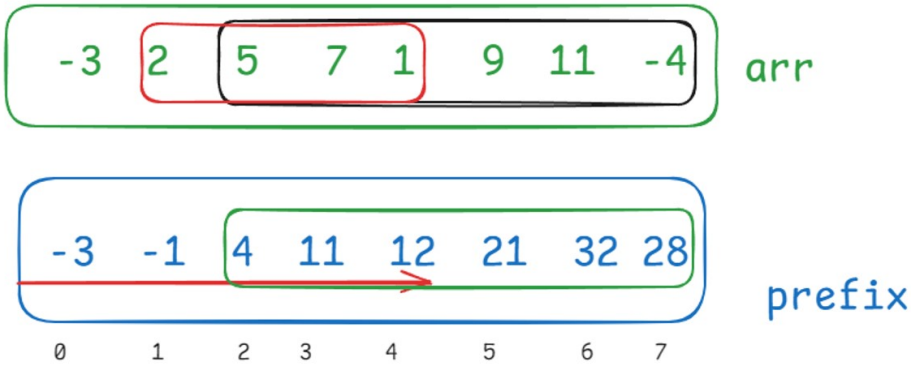
2, 8, 3, 9, 6, 5, 4

prefix[0] = 2;
prefix[1] = prefix[0] + arr[1] = 2 + 8 = 10
prefix[2] = prefix[1] + arr[2] = 10 + 3 = 13.
prefix[3] = prefix[2] + arr[3] = 13 + 9 = 22.
prefix[4] = prefix[3] + arr[4] = 22 + 6 = 28
prefix[5] = prefix[4] + arr[5] = 28 + 5 = 33
prefix[6] = prefix[5] + arr[6] = 33 + 3 = 37

SUM(2,5) = prefix[5] - prefix[2-1]
= 33 - 10 = 23;

```
import java.util.Scanner;
public class SubarraySum {
    public static int[] prefixSum(int[] arr){
        int n = arr.length;
        int[] prefix = new int[n];
        prefix[0] = arr[0];
        for(int i =1; i<n ; i++){
            prefix[i] = prefix[i-1] + arr[i];
        }
        return prefix;
    }
    public static int findSubarraySum(int[] prefix , int i , int j){
        if(i == 0){
            return prefix[j];
        }else{
            return prefix[j] - prefix[i-1];
        }
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the number of elements in the array");
        int n = scn.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the elements of an array: ");
        for(int i =0 ; i<n ; i++){
            arr[i] = scn.nextInt();
        }
```

```
        for(int i =0 ; i<n ; i++){
            arr[i] = scn.nextInt();
        }
        int[] prefix = prefixSum(arr);
        System.out.println("Starting Index : ");
        int i = scn.nextInt();
        System.out.println("Ending Index : ");
        int j = scn.nextInt();
        int result = findSubarraySum(prefix , i ,j);
        System.out.println(result);

    }
}
```



SUM(0,4) = 12
SUM(2,7) = 28 - (-1) = 29
SUM(1,4) = 12 - (-3) = 12 + 3 = 15

Problem: Find Last Occurrence Index

Problem Statement:

Given an array `arr[]` and an element `x`, find the last occurrence index of `x` in the array. If `x` is not present in the array, return `-1`.

Example:

- **Input:**
  - `arr[] = {2, 8, 3, 9, 6, 5, 4, 9, 3}`
  - `x = 9`
- **Output:**
  - `7` (because the last occurrence of `9` is at index `7`)

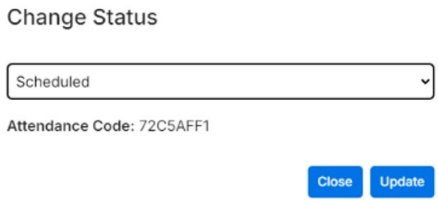
keep on iterating from end to start
return the first occurrence of that index
if found else return -1.

```
Enter the element to find :
```

it found else return -1.

```
Enter the element to find :
9
The last occurence of 9 is at index: 7
```

```java
import java.util.Scanner;
public class LastOccurenceIndex {
    public static int FindLastOccurence(int[] arr , int x){
        for(int i =arr.length-1; i>=0; i--){
            if(arr[i] == x){
                return i;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        System.out.println("Enter the number of elements in the array");
        int n = scn.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the elements of an array: ");
        for(int i =0 ; i<n ; i++){
            arr[i] = scn.nextInt();
        }

        System.out.println("Enter the element to find : ");
        int x = scn.nextInt();

        int index = FindLastOccurence(arr , x);

        if(index != -1){
            System.out.println("The last occurence of " + x + " is at index: " + index);
        }else{
            System.out.println(x + " is not present in an array");
        }
    }
}
```



Problem Statement:

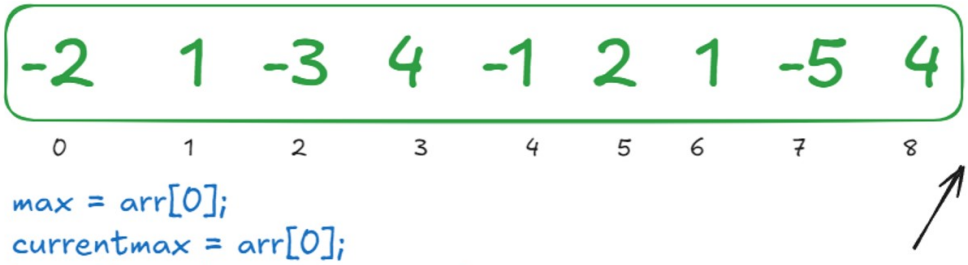
Given an array of integers, find the maximum sum of any contiguous subarray.

Example:

- **Input:**
  - **`arr[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4}`**
- **Output:**

- `arr[] = {-2, 1, -3, 4, -1, 2, 1, -5, 4}`

- **Output:**
  - `6` (because the contiguous subarray `[4, -1, 2, 1]` has the maximum sum `6`)




```
max = arr[0];
currentmax = arr[0];
loop through i = 1 ;  i < n -1;
currentmax = Max(arr[i] , currentSum + arr[i]);
max = Max(max , currentmax);
return max;
```

[Kadane's Algorithm]

Memory

```
max = 6
currentMax = 5
```

cur = max(4, 1+4) = 5

return max;