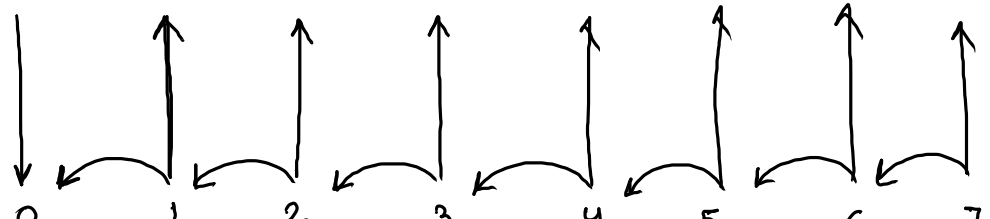
⇒ Prefix Sum (sum of all the elements on the left side including itself)

arr =

0	1	2	3	4	5	6	7
5	2	3	-4	0	7	1	-2



prefix sum array =

0	1	2	3	4	5	6	7
5	7	10	6	6	13	14	12

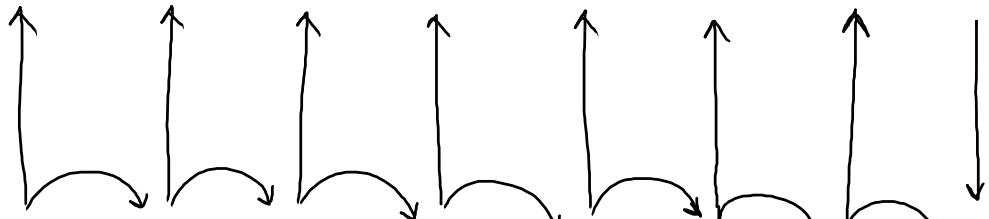
eq^n

$$prefix[i] = prefix[i-1] + arr[i]$$

⇒ Suffix Sum (sum of all the elements on the right side including itself)

arr =

0	1	2	3	4	5	6	7
5	2	3	-4	0	7	1	-2



suffix sum array =

0	1	2	3	4	5	6	7
12	7	5	2	6	6	-1	-2

eqn

$$suffix[i] = suffix[i+1] + arr[i]$$

code for prefix sum

```
pre[0] = arr[0];
for(int i = 1 ; i<n ; i++){
        pre[i] = pre[i-1] + arr[i];

}
```

code for suffix sum

```
suffix[n-1] = arr[n-1];

for(int i = n-2; i>=0; i--){

    suffix[i] = suffix[i+1] + arr[i];

}
```

dry run

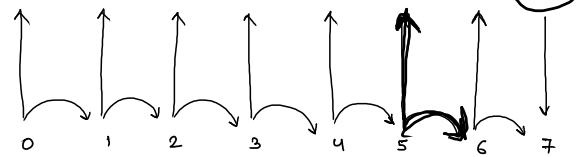
```
suffix[n-1] = arr[n-1];

for( int i = n-2 ; i>=0 ; i-- ){
    suffix[i] = suffix[i+1] + arr[i];
}
```

n = 8

arr =

0	1	2	3	4	5	6	7
5	2	3	-4	0	7	1	-2



suffix sum array =

0	1	2	3	4	5	6	7
12	7	5	2	6	6	-1	-2

i = 6, suf[6] = suf[7] + arr[6]

i = 5, suf[5] = suf[6] + arr[5]

i = 4, suf[4] = suf[5] + arr[4]

i = 3, suf[3] = suf[4] + arr[3]

i = 2, suf[2] = suf[3] + arr[2]

i = 1, suf[1] = suf[2] + arr[1]

i = 0, suf[0] = suf[1] + arr[0]

Note :-

$T.C = O(n)$

$S.C = O(n)$

code

```java
public static void main(String[] args) {
    int[] arr = {5, -4, 0, -1, 3};
    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];
    }
    // print
    for (int i = 0; i < n; i++) {
        System.out.print(prefix[i] + " ");
    }
}
```

Print Prefix Sum

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int[] ans = prefixSum(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.println(ans[i]);
    }
}
public static int[] prefixSum(int[] arr, int n) {
    int[] pre = new int[n];
    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = pre[i - 1] + arr[i];
    }
    return pre;
}
```

T.C = O(n)

S.C = O(n)

where n is size of array

Greatest Till Me

(maximum till the current index)

arr = [5 , 2 , 3 , 7 , -4 , 9 , 0]
0 1 2 3 4 5 6

prefix max array = [5 , 5 , 5 , 7 , 7 , 9 , 9]
0 1 2 3 4 5 6

eg")

```
pre[i] = Math.max(pre[i-1], arr[i]);
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    int[] ans = greatestTillMe(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.println(ans[i]);
    }
}
public static int[] greatestTillMe(int[] arr, int n) {
    int[] pre = new int[n];
    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = Math.max( pre[i - 1], arr[i] );
    }
    return pre;
}
```

dry run

arr = [5 , 2 , 3 , 7 , -4 , 9 , 0]

pre = [5, 5, 5, 7 , 7 , 9, 9]

i = 1, pre[i] = max(5 , 2);

i = 2, pre[i] = max(5 , 3);

i = 3, pre[i] = max(5, 7);

i = 4, pre[i] = max(7, -4);

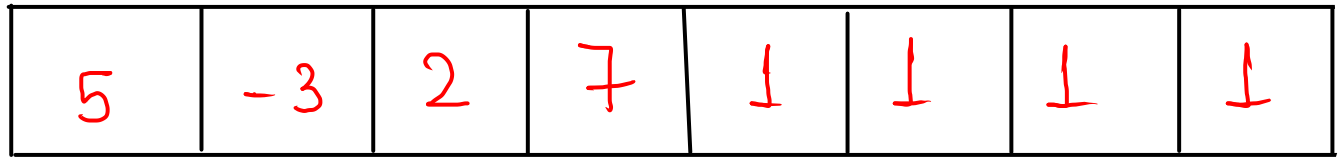
i = 5, pre[i] = max(7, 9);

i = 6, pre[i] = max(9, 0);

Find Pivot Index 1

(9mp)

arr =



l = 0
r = 10

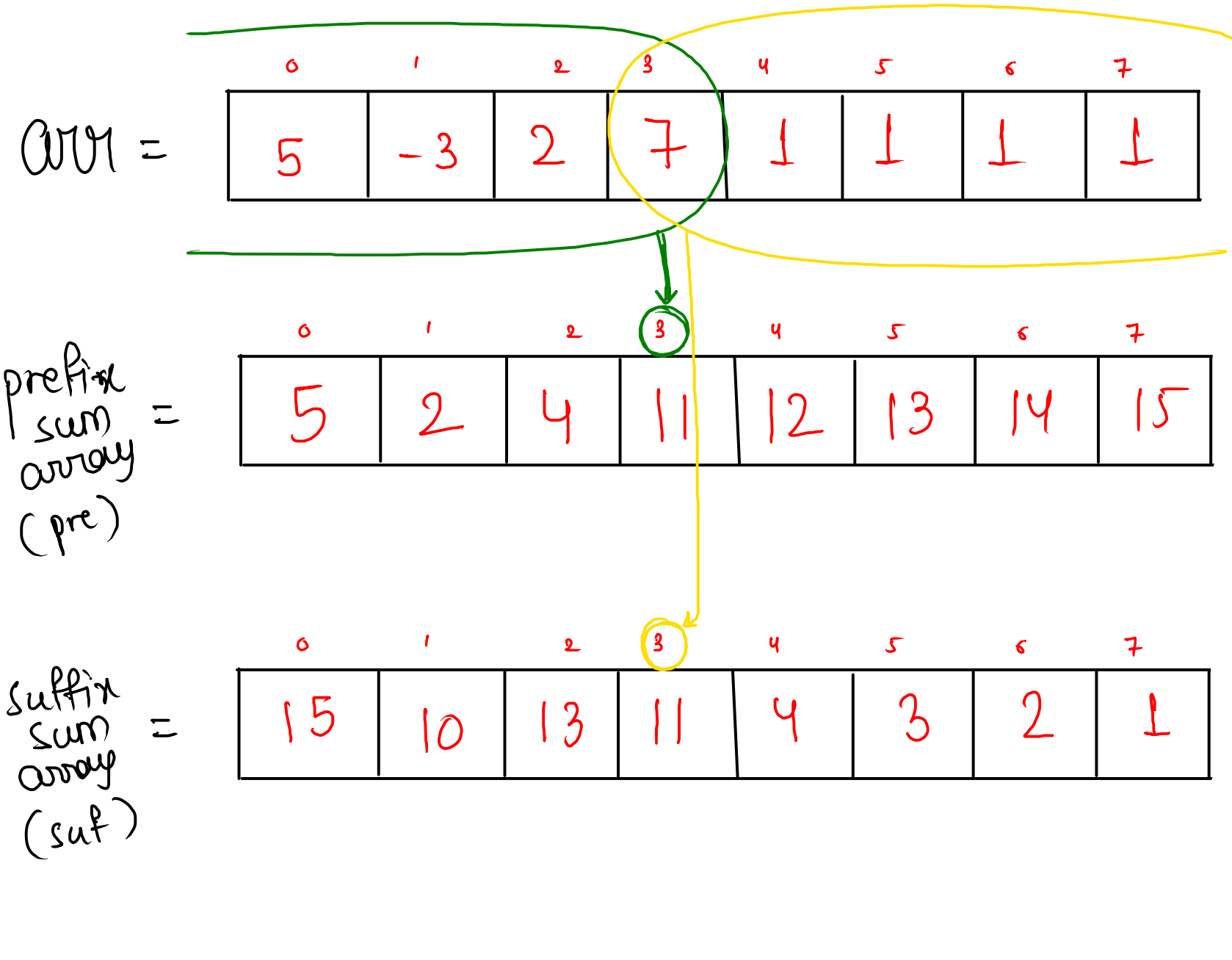
l = 5
r = 13

l = 2
r = 11

l = 4
r = 4

pivot index

ans = 3


arr =
0 1 2 3 4 5 6 7
5 -3 2 7 1 1 1 1
prefix sum array (pre) =
0 1 2 3 4 5 6 7
5 2 4 11 12 13 14 15
suffix sum array (suf) =
0 1 2 3 4 5 6 7
15 10 13 11 4 3 2 1

psudo code

1) create prefix array

pre[0] = arr[0];

loop from 1 to n

pre[i] = pre[i-1] + arr[i]

2) create suffix array

suf[n-1] = arr[n-1]

loop from (n-2) to 0

suf[i] = suf[i+1] + arr[i]

3) loop from 0 to n

check if pre[i] == suf[i]

return i;

return -1;

code

```java
public static int pivotIndex(int[] arr, int n) {
    int[] pre = new int[n];
    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = pre[i - 1] + arr[i];
    }

    int[] suf = new int[n];
    suf[n - 1] = arr[n - 1];
    for (int i = n - 2; i >= 0; i--) {
        suf[i] = suf[i + 1] + arr[i];
    }

    for (int i = 0; i < n; i++) {
        if ( pre[i] == suf[i] ) {
            return i;
        }
    }
    return -1;
}
```

operations = 3*n

T.C = O(n)

---

m/m = 2n

S.C = O(n)

where n is size of array