⇒ Prefix Sum

:- when each element is the sum of all left side elements including itself

arr =

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

←

prefix sum array =

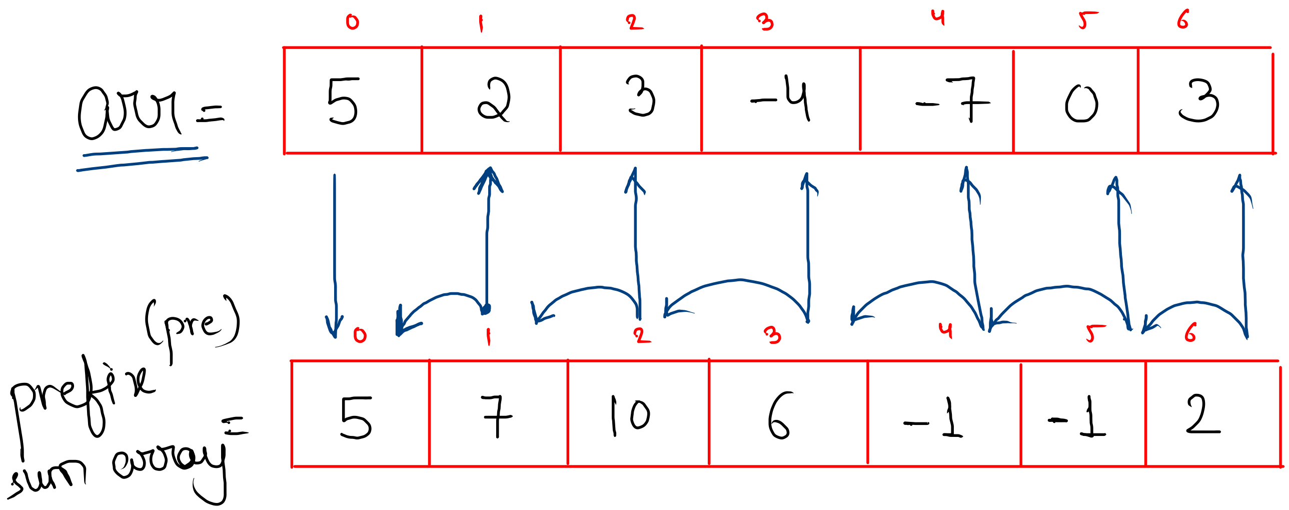
0	1	2	3	4	5	6
5	7	10	6	-1	-1	2

⇒ Suffix Sum :-

:- when each element is the sum of all right side elements including itself

suffix sum array =

0	1	2	3	4	5	6
2	-3	-5	-8	-4	3	3



Equation:- $pre[i] = arr[i] + pre[i-1]$, $i \in [1, n-1]$

$suf[i] = arr[i] + suf[i+1]$, $i \in [n-2, 0]$

Print Prefix Sum between L and R

n = 8

arr =

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

left = 3
right = 6

pre =

0	1	2	3	4	5	6	7
5	7	3	6	13	17	16	14

psudo code

```
1) create prefix sum array of size n
2) pre[0] = arr[0]
3) loop from 1 to n-1
    3.1) pre[i] = arr[i] + pre[i-1]
```

code

T.C = O(N) & S.C = O(N)

```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 left = scn.nextInt();
    int right = scn.nextInt();
    printLtoR(arr, n, left, right);
}
public static void printLtoR(int[] arr, int n, int left, int right) {
    int[] pre = new int[n];
    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = arr[i] + pre[i - 1];
    }

    for (int i = left; i <= right; i++) {
        System.out.println( pre[i] );
    }
}
```

Greatest Till Me

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

	0	1	2	3	4	5	6	7
pre =	5	5	5	5	7	7	7	7

(prefix max array)

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

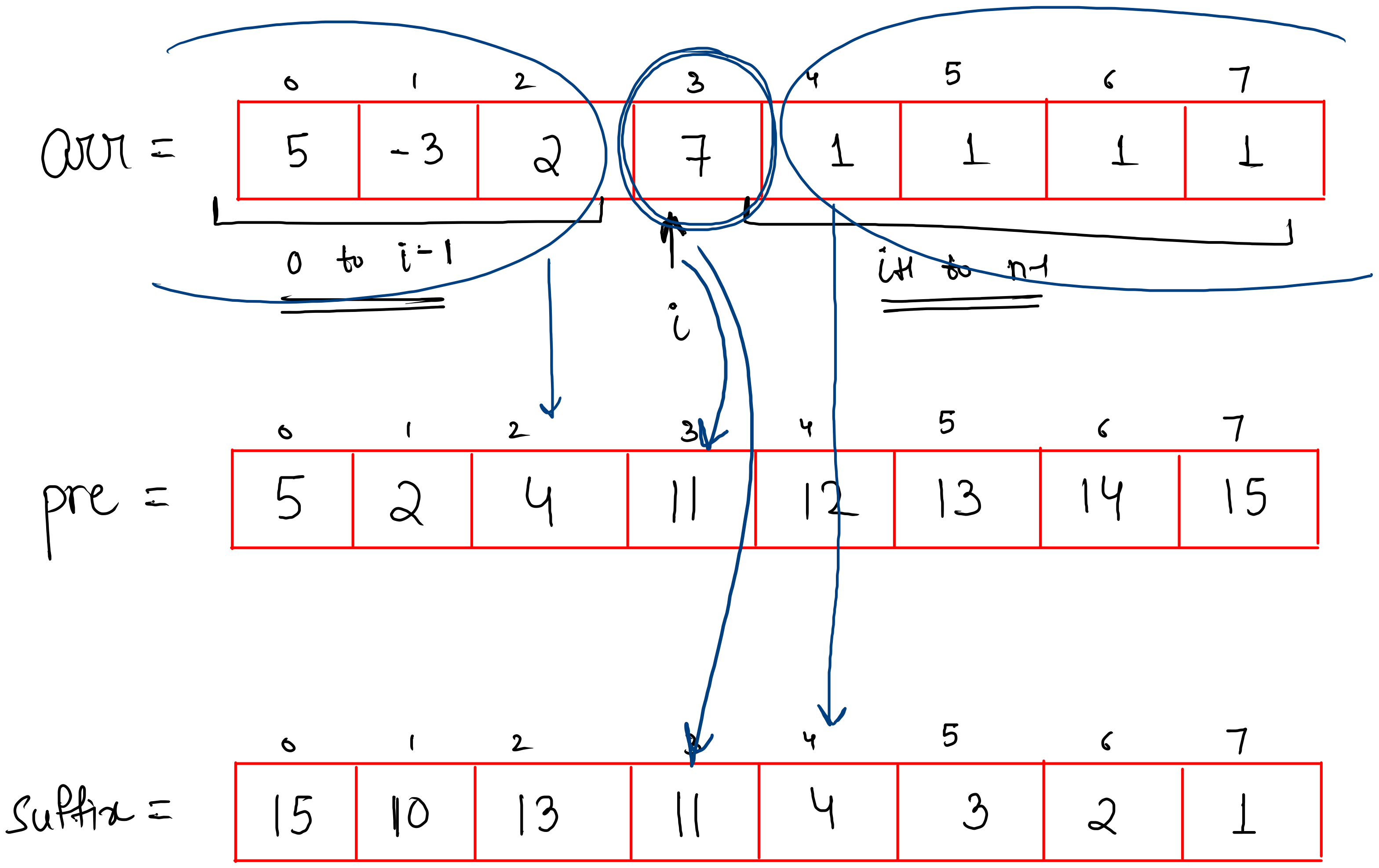
code

T.C = O(N) , S.C = O(N)

```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();
    }
    greatestTillMe(arr, n);
}
public static void 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( arr[i], pre[i - 1] );
    }

    for (int i = 0; i < n; i++) {
        System.out.println(pre[i]);
    }
}
```

Find Pivot Index 1

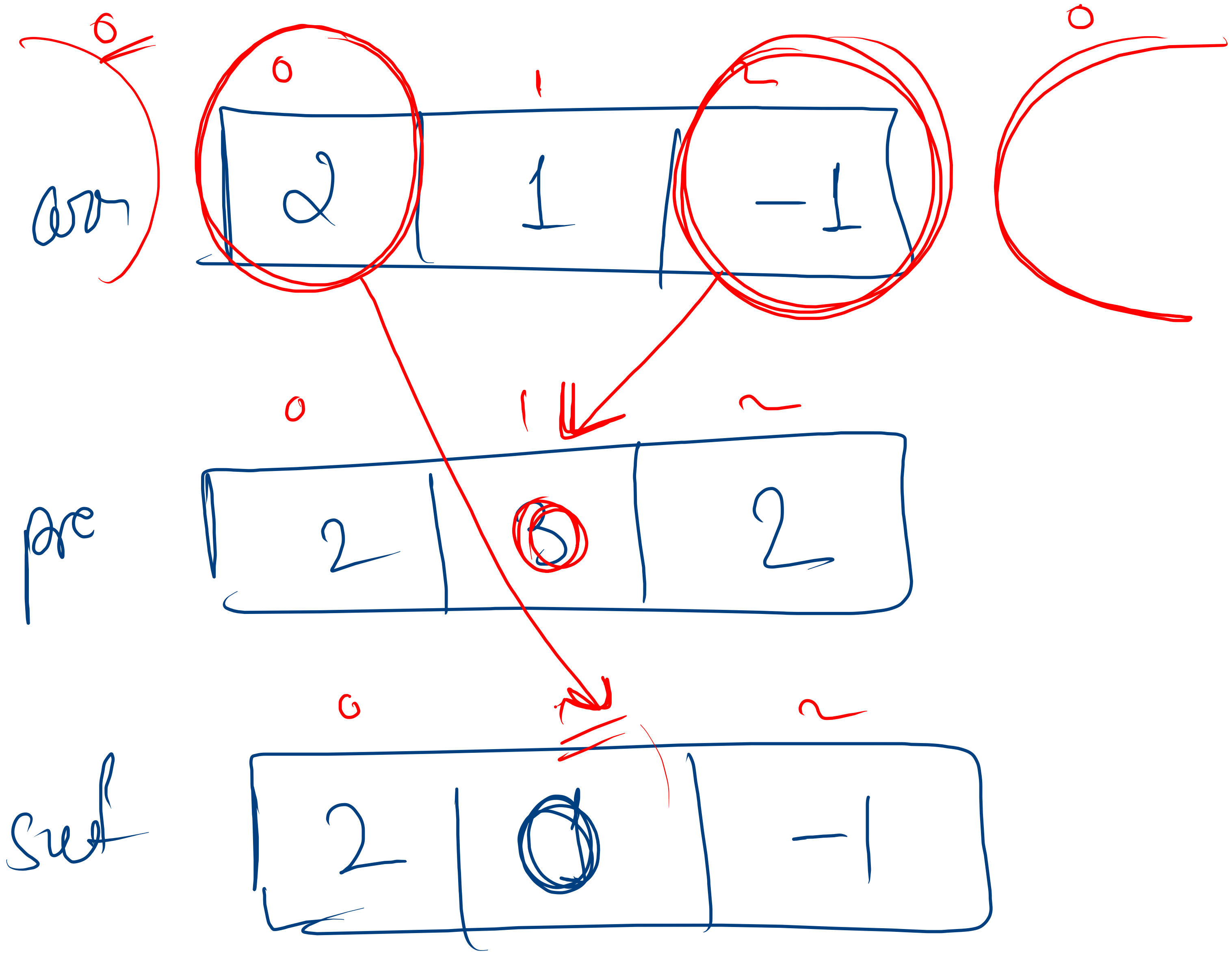


Code

```
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();
    }
    System.out.println(findPivot(arr, n));
}
public static int findPivot(int[] arr, int n) {
    // prefix sum array
    int[] prefix = new int[n];
    prefix[0] = arr[0];
    for (int i = 1; i < n; i++) {
        prefix[i] = arr[i] + prefix[i - 1];
    }

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

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


0
arr
0 1 2
2 1 -1
0
pre
0 1 2
2 3 2
suf
0 1 2
2 0 -1

another way

```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();
    }
    System.out.println(findPivot(arr, n));
}
public static int findPivot(int[] arr, int n) {
    // prefix sum array
    int[] prefix = new int[n];
    prefix[0] = arr[0];
    for (int i = 1; i < n; i++) {
        prefix[i] = arr[i] + prefix[i - 1];
    }

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

    if ( n > 1 && suffix[1] == 0 ) return 0;

    for (int i = 1; i < n - 1; i++) {
        if ( prefix[i - 1] == suffix[i + 1] ) {
            return i;
        }
    }

    if ( n > 1 && prefix[n - 2] == 0 ) return n - 1;

    return -1;
}
```

edge cases