Prefix Array

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & 5 \\ arr = & [1, & 8, & 2, & 6, & 7, & 4] \end{array}$$

$$\begin{array}{lcccccc} prefix = & [1 & 8 & 8 & 8 & 8 & 8] \\ & 0 & 1 & 2 & \boxed{3} & 4 & 5 \end{array}$$

prefix[0] = arr[0]

We can use prefix array for these typc of questions:-

1. Greatest till current value
2. Smallest till current value
3. Sum of all the value from beginning to current values

arr = [1, 8, 2, 6, 7, 4]

res = [1, 8, 8, 8, 8, 8].

Example 2.

Greatest till current value

arr = [1, 4, 2, 6, 8]
0 1 2 3 4

[1, 4, 4, 6, 8] → Answer.

prefix[0] = arr[0];

Prefix = [1, 4, 4, 6, 8]
0 1 2 3 4

Question:-

Smallest till current Value

arr = [1, 0, 14, 2, 3]
0 1 2 3 4

Prefix = [1, 0 0 0 0]
0 1 2 3 4

Greatest till me.

n = 7

	0	1	2	3	4	5	6
arr =	1	88	3	2	16	10	9
prefix =	1	88	88	88	88	88	88

arr = [1, 88, 3, 2, 16, 10, 9]

prefix = [1, 88, 88, 88, 88, 88, 88]

Algorithm:

1. Greate a prefix array with length same as original array
2. First value of prefix array will be first value of original array
3. Compare the value stored at previous index with current value, whichever is greater will current value in prefix array.

```
int prefix[] = new int[arr.length]
                       or n

prefix[0] = arr[0];
```

```
for(int i=1; i<n;i++){
    if (prefix[i-1]>arr[i]){
      prefix[i] = prefix[i-1];
    }else {
      prefix[i]=arr[i];
    }
  }
  for (int i=0;i<n;i++){
     S.o.pln(prefix[i]);
   }
```

Time Complexity→ O(n)

Print Prefix Sum

n = 5

arr = [1, 2, 8, 4, 10]
0 1 2 3 4

Output :

[1, 3, 11, 15, 25]

prefix = [1, 3, 11, 15, 25]
0 1 2 3 4

Example 2 .

arr = [2, 3, 7, 4, 1]
0 1 2 3 4

prefix = [2, 5, 12, 16, 17]
0 1 2 3 4

```
int prefix[] = new int[n];
  prefix[0] = arr[0];
for(int i=1; i<n; i++){
  prefix[i] = prefix[i-1] + arr[i];
  }
for (int i=0; i<n; i++) {
    S.o.pln(prefix[i]);
  }
```

Find Pivot Index

n = 6

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

At index 0.

leftsum = 0 × pivot
rightsum = 27

At index 1.

leftsum = 1 × pivot
rightsum = 20

At index 2.

leftsum = 8 × Pivot
rightsum = 17

At index 3.

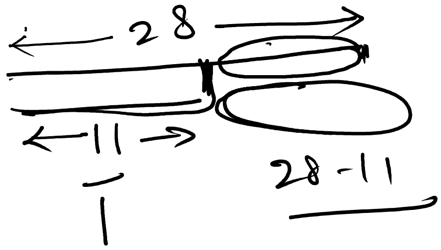
leftsum = 11 ✓ Pivot.
rightsum = 11

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

28 T

0 1 2 3 4 5

prefixSum: [1, 8, 11, 17, 22, 28]
0 1 2 3 4 5



At any index

leftsum = prefix[i-1];

rightsum = prefix[n-1] - prefix[i]

Code.

```
int prefix[] = new int[n];
  prefix[0] = arr[0]
for(int i=1; i<n; i++){
  prefix[i] = prefix[i-1] + arr[i];
 boolean hasPivot = false;
for(int i=0; i<n; i++){
  int leftsum=0, rightsum=0;
 if(i==0){
  leftsum=0;
 }else{
}else{
 leftsum = prefix[i-1];
 }
 if(i==n-1){
  rightsum=0;
 }else{
 rightsum= prefix[n-1]-prefix[i];
   }
 if(leftsum==rightsum){
   s.o.pln(i);
   haspivot=true;
   break;
}
}
if(haspivot==false){
  s.o.pln(-1);
```

}

Test Case 2 .

n = 3

arr = [2, 1, -1]

0 1 2

At index 0 .

Leftsum = 0

rightsum = 0

✓ pivot .