

Lesson:



Problems on Array-4



Pre-Requisites

- Arrays in Java

Today's Checklist

- Array Problems based on concept of prefix sum

Pattern: Prefix Sums

In this approach, we create an array by taking the sum of values from the beginning of the array and keep adding them. Prefix[i] represents the values from the beginning of the array to all elements till index 'i'. Using this prefix sums technique, we can calculate range sums easily that will be seen in the problems.

Problem 1: Given an integers array 'a', return the prefix sum/ running sum in the same array without creating a new array.

Input :

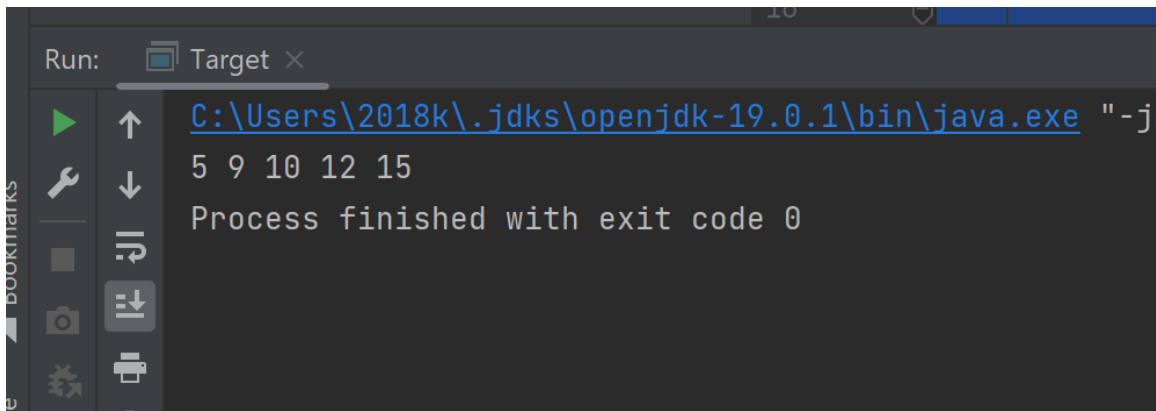
5 4 1 2 3

Output :

5 9 10 12 15

Code:

```
import java.io.*;
import java.util.*;
public class Target {
    public int[] runningSum(int[] a) {
        for (int i = 1; i < a.length; ++i){
            a[i] += a[i - 1];
        }
        return a;
    }
    public static void main(String[] args){
        int[] a={5, 4, 1, 2, 3};
        Target obj1 = new Target();
        int[] ans=obj1.runningSum(a);
        for (int element: ans) {
            System.out.print(element + " ");
        }
    }
}
```



Explanation: Add the previous element of the prefix sum to the current element. This way the current element contains the current element + all the values of previous elements.

Problem 2: Check if we can partition the array into two subarrays with equal sum. More formally, check that the prefix sum of a part of the array is equal to the suffix sum of rest of the array.

Input :

5 2 3 4

Output :


True

Explanation:

We can simply calculate the total sum of the whole array in the first traversal. Then in the second traversal, check at every point that the sum of the prefix part of the array is equal to the suffix part of array. Calculate the suffix using total sum - current prefix.

Code:

```
import java.io.*;
import java.util.*;
public class Target {
    public static boolean check(int[] a) {
        int n = a.length;
        int pref = 0, total_sum = 0;
        for (int i = 0; i < n; i++) {
            total_sum += a[i];
        }
        for (int i = 0; i < n; i++) {
            pref += a[i];
            int suff = total_sum - pref;
            if (pref == suff) return true;
        }
        return false;
    }
    public static void main(String[] args){
        int[] a={5, 2, 3,4};
        System.out.println(check(a));
    }
}
```



```

17
return false;
run: Target x
C:\Users\2018k\.jdk\openjdk-19.0.1\bin\java.exe "-javaagent:
true
Process finished with exit code 0

```

Problem 3: Given an array of integers of size n . Answer q queries where you need to print the sum of values in a given range of indices from l to r (both included).

Note: The values of l and r in queries follow 1-based indexing.

Input :

```

5
5 1 2 3 4
4
11
13
4 5
15

```

Output :

```

5
8
7
15

```

Code:

```

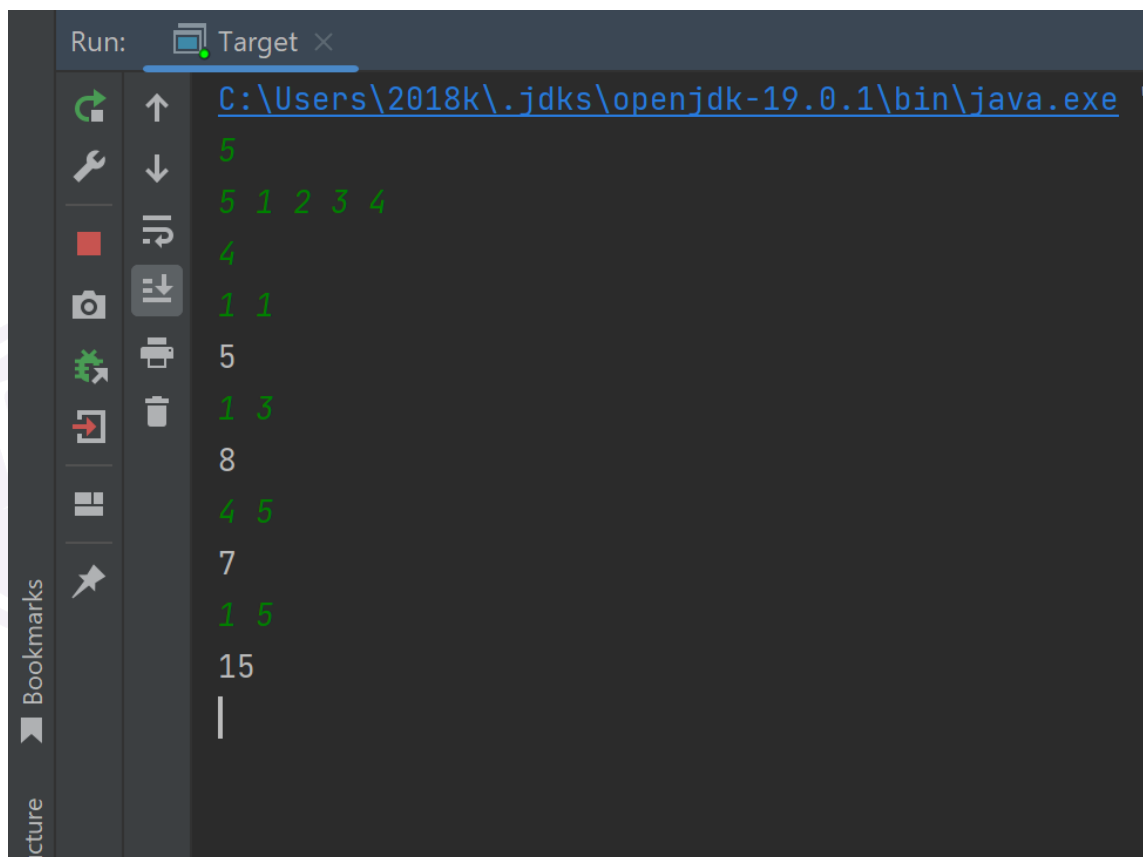
import java.io.*;
import java.util.*;
public class Target {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n;
        n=sc.nextInt();
        int[] a = new int[n+1];
        for (int i = 1; i ≤ n; i++) {
            a[i]=sc.nextInt();
        }
        for (int i = 1; i ≤ n; i++) {
            a[i] += a[i - 1];
        }
        // making a prefix sum array out of given array
    }
}

```

```

int q; // no of queries
q = sc.nextInt();
while (q ≥ 0) {
    int l, r;
    l = sc.nextInt();
    r = sc.nextInt();
    // we need to find sum of values of indices from l to r (both included)
    // so that is equal to (total sum till r - total sum till l-1 )
    // note we also need to include the value at index l so subtracting only till (l-1)
    int ans = (a[r] - a[l-1]);
    System.out.println(ans);
    q--;
}
}
}

```



```

Run: Target x
C:\Users\2018k\.jdk\openjdk-19.0.1\bin\java.exe "
5
5 1 2 3 4
4
1 1
5
1 3
8
4 5
7
1 5
15
|

```

Explanation: We first construct the prefix array from the given array as we need to answer queries for sum calculation from l to r. This way (total sum till r - total sum till l-1) becomes `prefix[r] - prefix[l-1]`. At each query input the value of left and right and return this.

Upcoming Class Teasers:

- Problems based on 2D Arrays