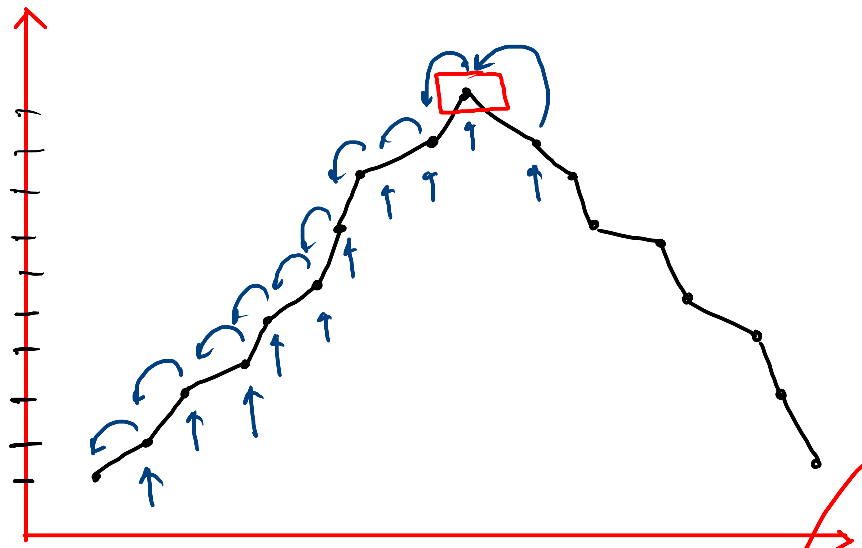
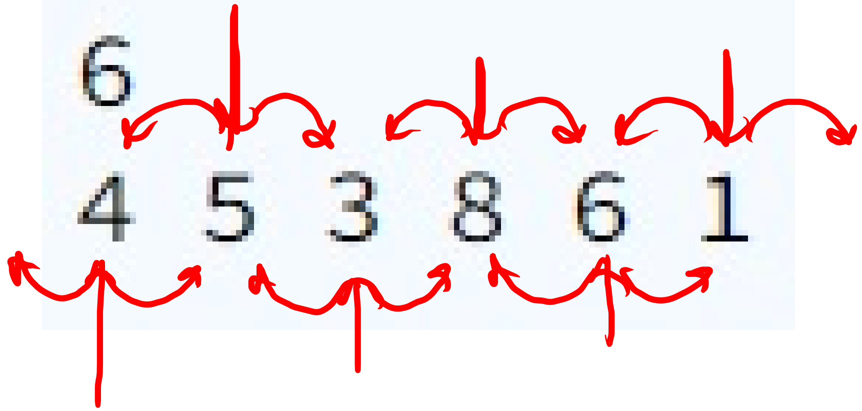
Peak Index in a Mountain Array 2



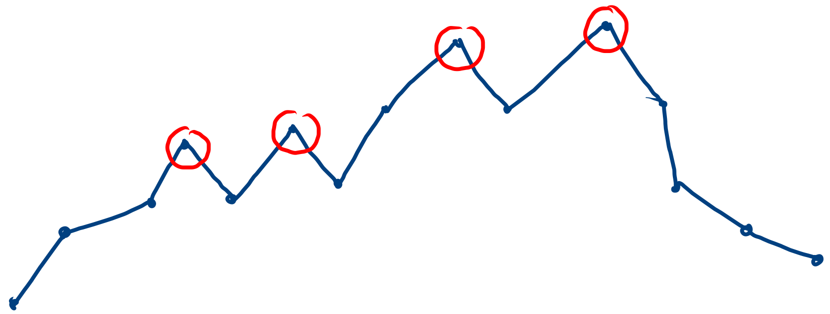
```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    System.out.println(findPeak(arr, n));
}

public static int findPeak(int[] arr, int n) {
    for (int i = 1; i < n - 1; i++) {
        if ( arr[i] > arr[i - 1] && arr[i] > arr[i + 1] ) {
            return i;
        }
    }
    return -1;
}
```

defination of peak element

Peak Elements



ans 5 8



```
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();
    }

    findPeak(arr, n);
}

public static void findPeak(int[] arr, int n) {
    for (int i = 1; i < n - 1; i++) {
        if ( arr[i] > arr[i - 1] && arr[i] > arr[i + 1] ) {
            System.out.print( arr[i] + " " );
        }
    }
}
```

$\Rightarrow$ **Subarray** (continuous sequence within our array)

1, 2, 3, 4, 5

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

2
2 3
2 3 4
2 3 4 5

3
3 4
3 4 5

4
4 5

5

$\boxed{2 \quad 3 \quad 5}$ $\propto$ subsequence

Subarray :- part of array in same order and need to continuous

Subsequence :- part of array in same order and need not to be continuous

Print All Subarrays

```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();
    }

    printSubarrays(arr, n);
}

public static void printSubarrays(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            print(arr, i, j);
        }
    }
}

public static void print(int[] arr, int si, int ei) {
    for (int i = si; i <= ei; i++) {
        System.out.print(arr[i] + " ");
    }
    System.out.println();
}
```

$$\begin{array}{cccc} & 0 & 1 & 2 \\ arr & 1 & 2 & 3 \end{array}$$

printing

$i=0, j=0$ — 1

$j=1$ — 1 2

$j=2$ — 1 2 3

$i=1, j=1$ — 2

$j=2$ — 2 3

$i=2, j=2$ — 3

first f^n

Sum Equals Zero

2 3 -3 5

subarray

2

2 3

2 3 -3

2 3 -3 5

3

✓ $\boxed{3 \quad -3}$

3 -3 5

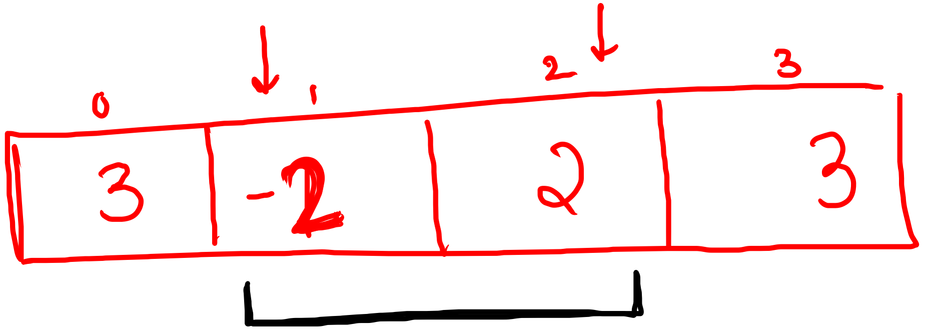
-3

-3 5

5

complete

```java
public static void subarraySum(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        int sum = 0;
        for (int j = i; j < n; j++) {
            sum += arr[j];
            if (sum == 0) {
                System.out.println(true);
                return;
            }
        }
    }
    System.out.println(false);
}
```



$i=0$, $sum=0$, $j=0$ $sum=3$

$j=1$ $sum=1$

$i=0$, $j=2$ $sum=3$

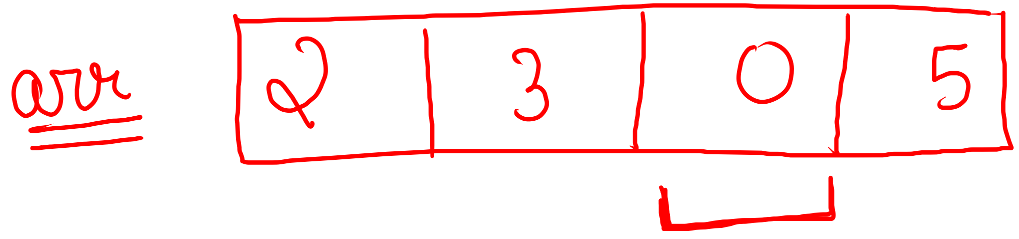
$j=3$ $sum=6$

$i=1$, $j=1$, $sum=-2$

$i=1$, $j=2$, $\boxed{sum=0}$

```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();
    }

    subarraySum(arr, n);
}

public static void subarraySum(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        int sum = arr[i];
        if (sum == 0) {
            System.out.println(true);
            return;
        }
        for (int j = i + 1; j < n; j++) {
            sum += arr[j];
            if (sum == 0) {
                System.out.println(true);
                return;
            }
        }
    }
    System.out.println(false);
}
```

complete



⇒ Kadane's Algo (works in linear time) O(N)

↳ used to find "maximum subarray sum"

1	2	3	-2

(1) 1

(3) 1 2

✓ (6) 1 2 3

(4) 1 2 3 -2

(2) 2

(5) 2 3

(3) 2 3 -2

(3) 3

(1) 3 -2

(-2) -2

HW_Olympiad Team Selection

(Game Theory)

take turn optimally