Revision

* Kadane's Algo.

↳ Maximum sum subarray

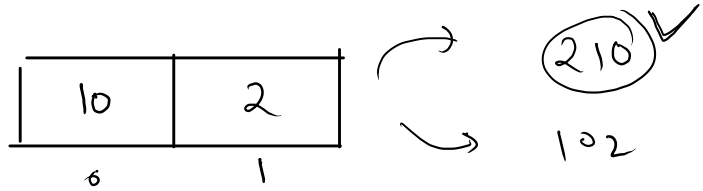
179. Largest Number

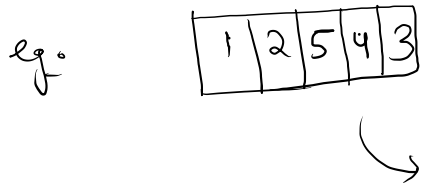
Example 1:

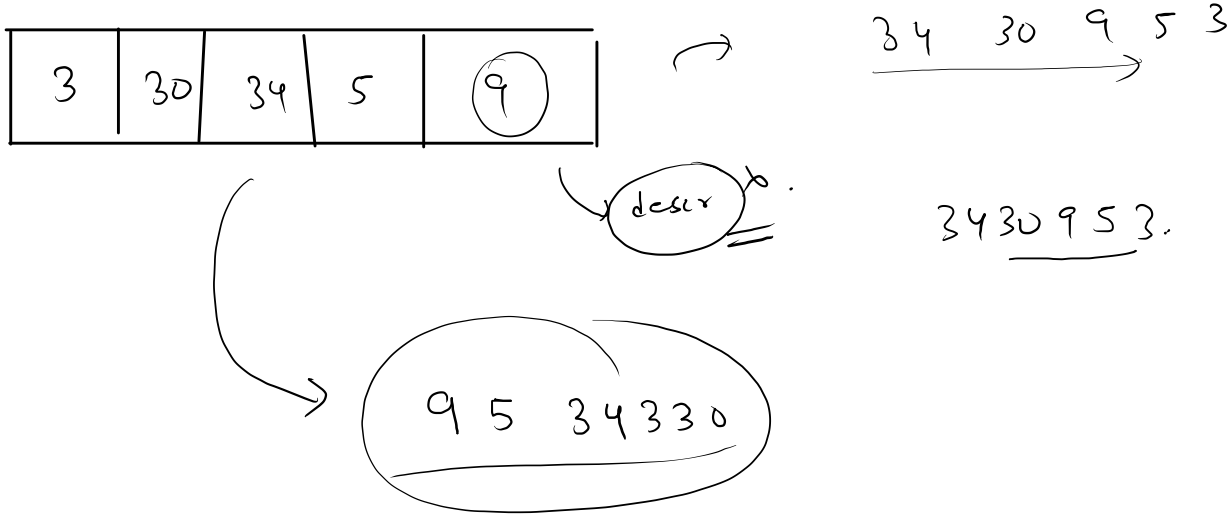
```
Input: nums = [10,2]
Output: "210"
```

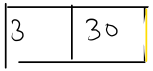
Example 2:

```
Input: nums = [3,30,34,5,9]
Output: "9534330"
```





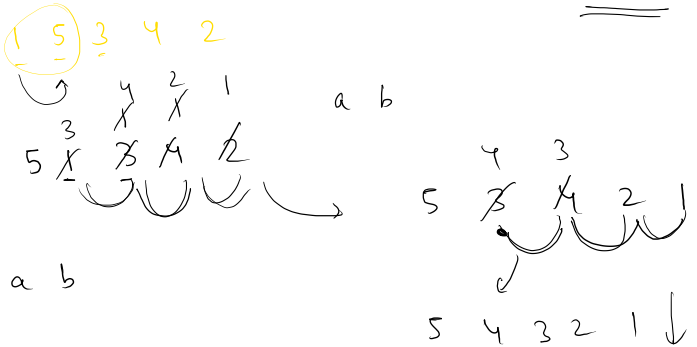


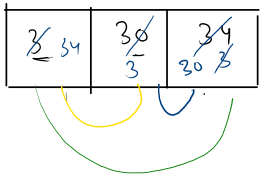


330 > 303

Sort → descending

return b-a;





sorting.

1.
2.
3.

a = 3

b = 30

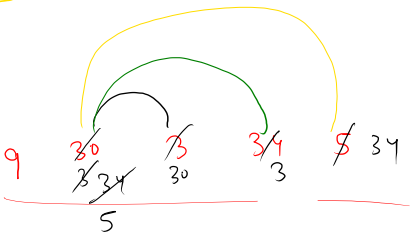
A S→ 303

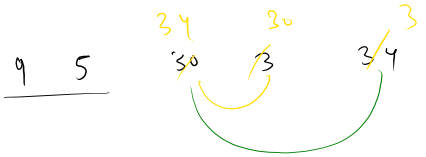
B r→ 330

34330

Input: nums = [3,30,34,5,9]

Output: "9534330"





summarize

→ s → "a+b"

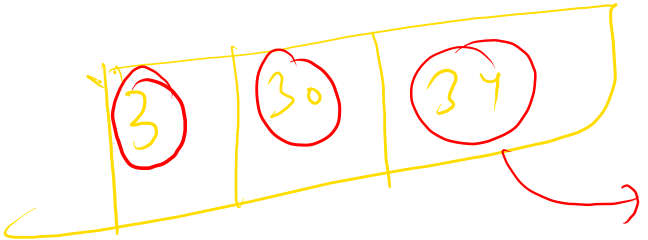
= → concatenation.

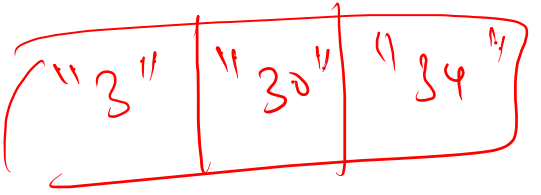
r → "b+a"

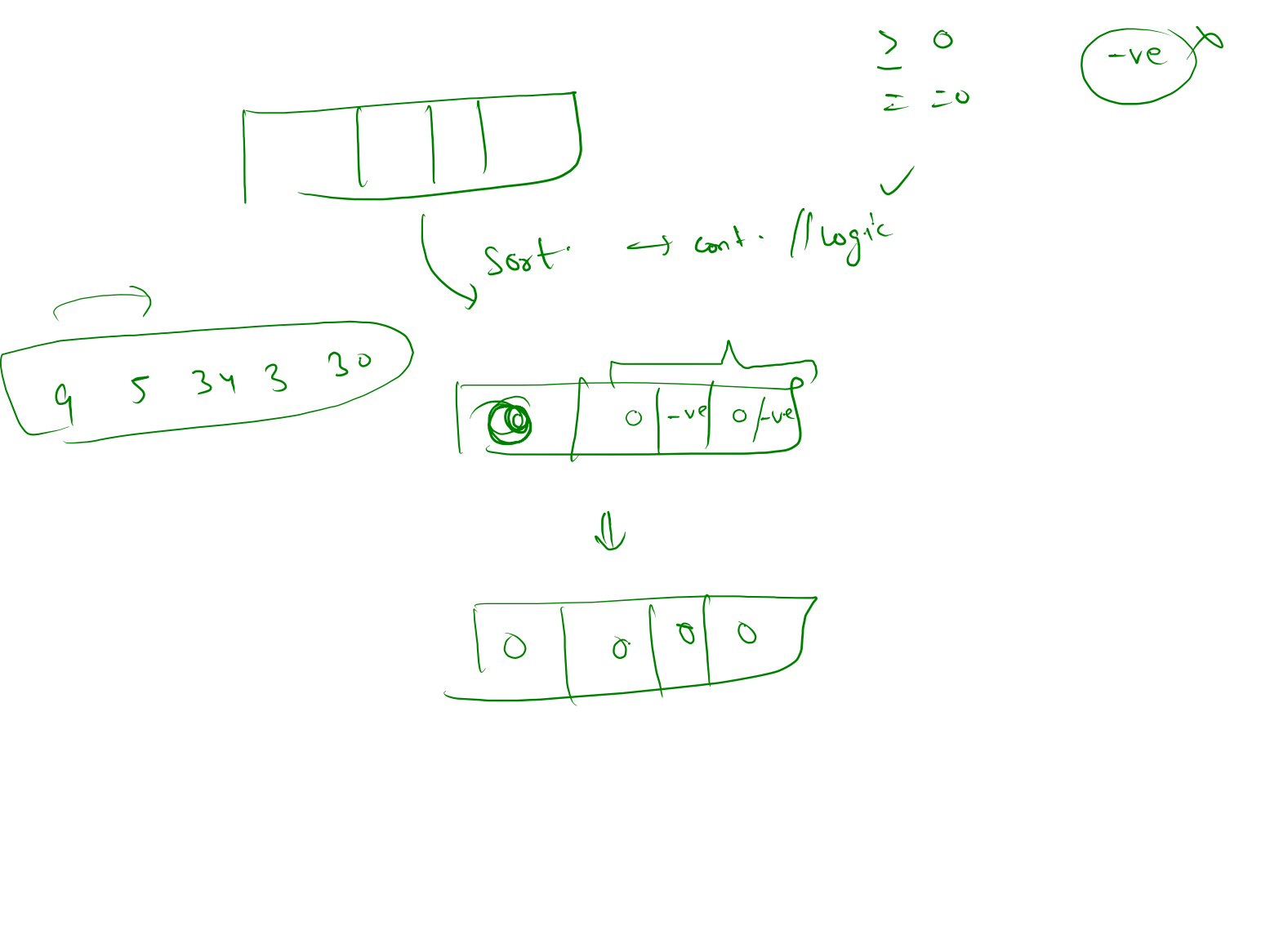
b - a

→

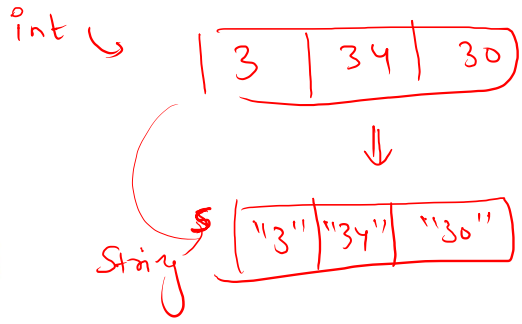
r. compareTo(s)

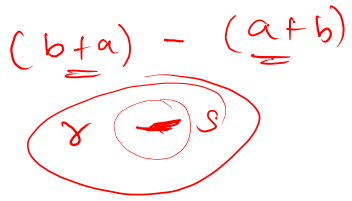

3
30
34


"3"
"30"
"34"

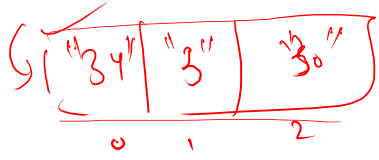

≥ 0
= =0
-ve
✓
Sort → cont. // logic
9 5 34 3 30
0 -ve 0 -ve
0 0 0 0

```
class Solution {
    public String largestNumber(int[] nums) {
        int n = nums.length;
        String [] S = new String[n];

        for(int i = 0; i < n; i++){
            S[i] = "" + nums[i];
            // S[i] = String.valueOf(nums[i]);
        }

        Arrays.sort(S, (a,b) -> {
            String s = a + b;
            String r = b + a;

            return r.compareTo(s);
        });

        if(S[0].equals("0")){
            return "0";
        }
        String ans = "";
        for(int i = 0; i < n; i++){
            ans += S[i];
        }

        return ans;
```





s - r → s.compare(r)
r - s → r.compare(s)

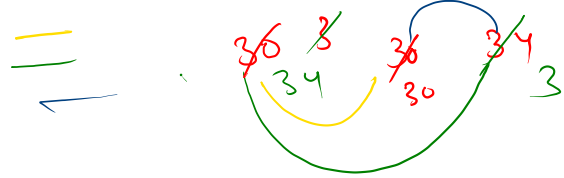


"30"

ans = ""

ans = "343030"

```
class Solution {
    public String largestNumber(int[] nums) {
        int n = nums.length;
        String [] S = new String[n];

        for(int i = 0; i < n; i++){
            S[i] = "" + nums[i];
            // S[i] = String.valueOf(nums[i]);
        }

        Arrays.sort(S, (a,b) -> {
            String s = a + b;
            String r = b + a;

            return r.compareTo(s);
        });

        if(S[0].equals("0")){
            return "0";
        }
        String ans = "";
        for(int i = 0; i < n; i++){
            ans += S[i];
        }

        return ans;
```



r.comp s.

s → 334
r → 343

a = 3
b = 30

r.c s

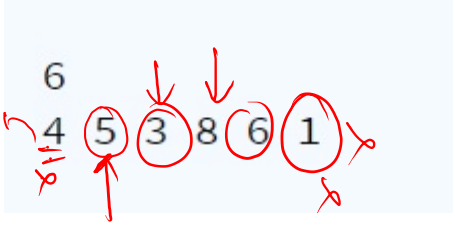
343 > 334

+ve
-ve
.0

a − b →
b − a →
s − r →
r ⊖ s →

Peak Elements

arr[i] is a peak element only if **arr[i-1] < arr[i] > arr[i+1]**.



peak ele

A[i-1] < A[i] > A[i+1]

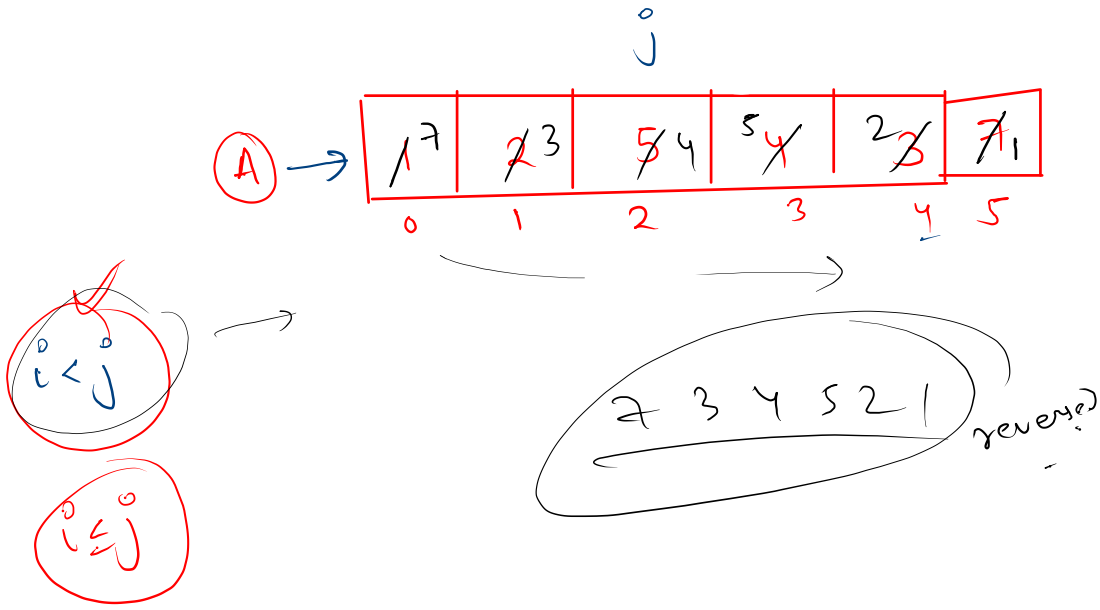
4 < 5 > 3

5 8

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        //logic
        for(int i = 1; i < n-1; i++){
            if(A[i] > A[i+1] && A[i] > A[i-1]){
                System.out.print(A[i] + " ");
            }
        }


    }
}
```

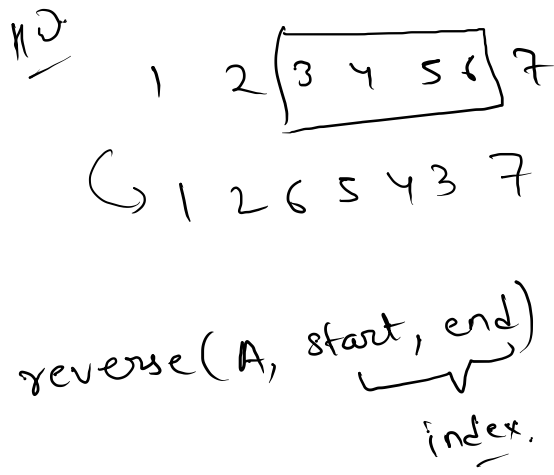
Peak Index in a Mountain Array

GKSTR32 Reverse_Array



★ do not take extra space in terms of 'n'

```java
ublic class Solution {
    public static void reverse(int [] A){
        int i = 0;
        int j = A.length-1;

        while(i < j){
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;

            i++;
            j--;
        }

        for(  i = 0; i < A.length; i++){
            System.out.println(A[i]);
        }
    }

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

        reverse(A);

    }
```



Type Casting