

Revision

* Kadane's Algo.

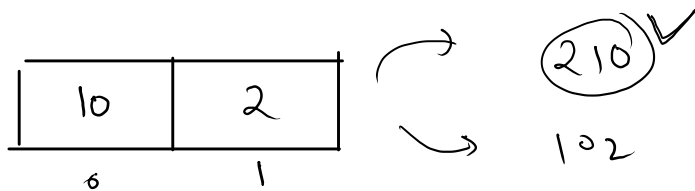
↳ Maximum sum subarray

179. Largest Number

Example 1:

Input: nums = [10,2]

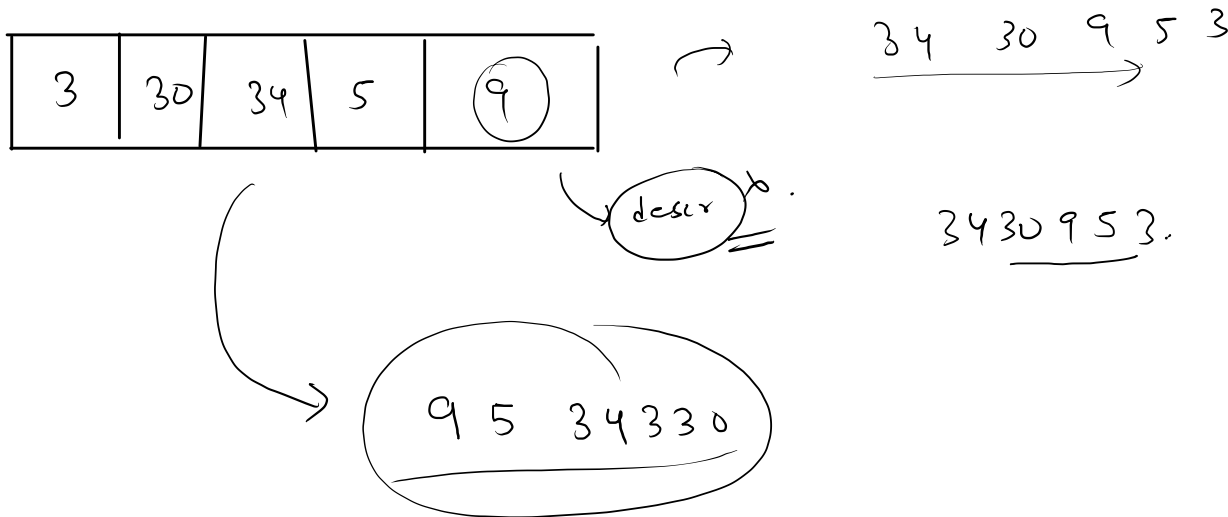
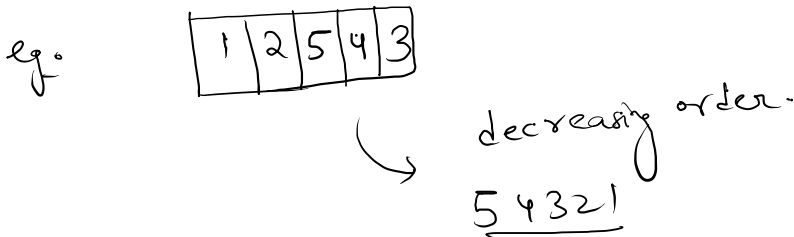
Output: "210"



Example 2:

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

Output: "9534330"



$$\begin{array}{|c|c|} \hline 3 & 30 \\ \hline \end{array}$$

330

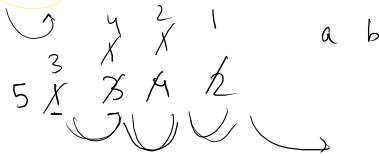
>

303

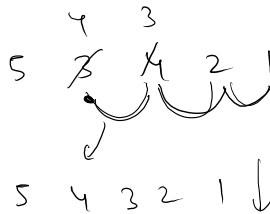
Sort \rightarrow descending

return b - a

1 5 3 4 2



a b



$$\begin{array}{|c|c|c|} \hline \cancel{3} 54 & \cancel{30} 3 & \cancel{34} 30\cancel{3} \\ \hline \end{array}$$

sorting

1.
2. ✓
3.

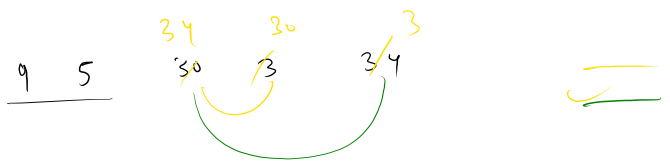
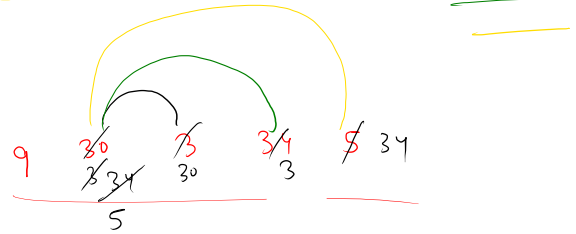
$$a = 3$$

$$b = 30$$

A \rightarrow 303 ✓
B \rightarrow 330 ✓

34330

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



a 3
 b 34
 s → 334
 r → (343)



a = 30
 b = 3
 s → 303
 r → (330)

9534330

summary

$s \rightarrow "a+b"$
↳ concatenation.

$r \rightarrow "b+a"$

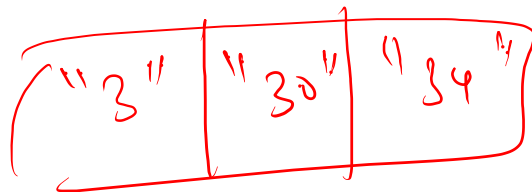
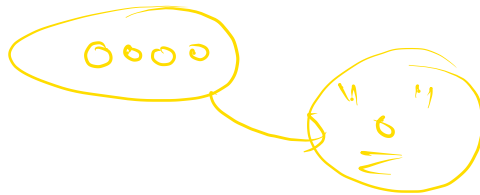
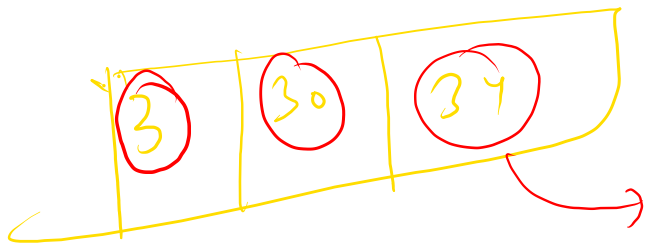
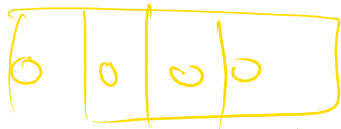
$b - a$

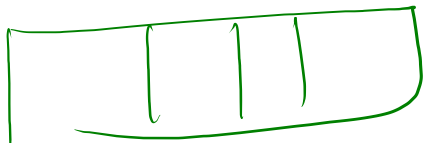
$r - a$

\rightarrow

$r + s$ → con
 $r - s$ /

$r \cdot \text{compareTo}(s)$ ✓

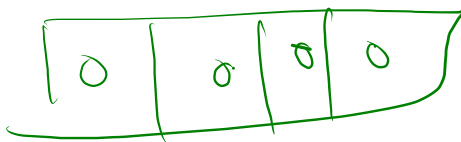
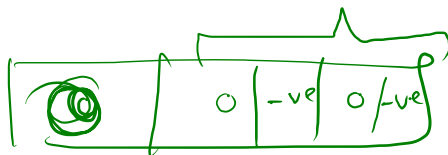
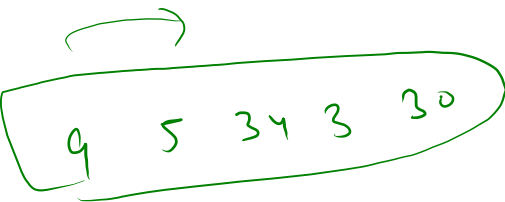




≥ 0
 ≥ 0

-ve

Sort. \rightarrow cont. / logic ✓




```

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;

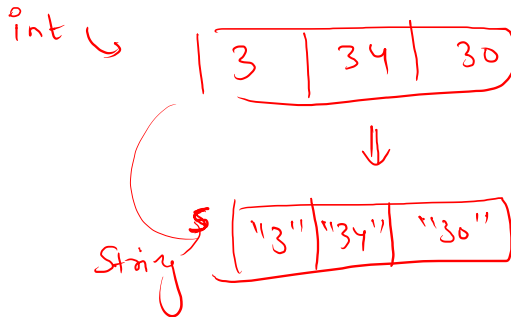
            return r.compareTo(s);
        });

        if(S[0].equals("0")){
            return "0";
        }

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

        return ans;
    }
}

```

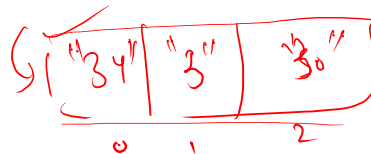


$$(b+a) - (a+b)$$



$$s - r \rightarrow s.compareTo(r)$$

$$r - s \rightarrow r.compareTo(s)$$



i

ans = ""

"30"

ans = "3430"

```

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]);
    }

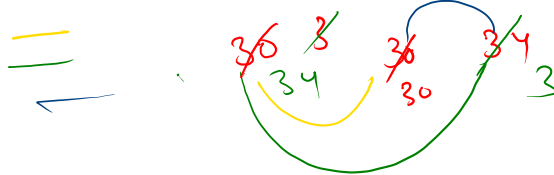
    Arrays.sort(S, (a,b) -> {
        String s = a + b;
        String r = b + a;

        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;
}

```



$$a = 3$$

$$b = 30$$

$$s \rightarrow 334$$

$$r \rightarrow 343$$

r.comp s

r.c s

$$\underline{343} > 334$$



$$a - b \rightarrow$$

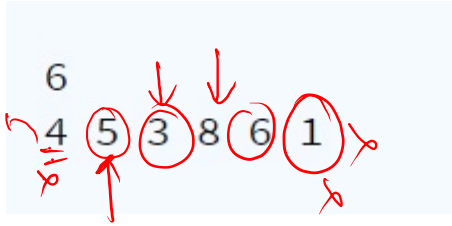
$$b - a \rightarrow$$

$$s - r \rightarrow$$

$$r \ominus s \rightarrow$$

Peak Elements

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



$$\underline{A[i-1]} < \underline{A[i]} > \underline{A[i+1]}$$

→ peak ←

✓
 $4 < 5 > 3$

5 8

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         Scanner scn = new Scanner(System.in);
8         int n = scn.nextInt();
9         int [] A = new int[n];
10        for(int i = 0; i < n; i++){
11            A[i] = scn.nextInt();
12        }
13        |
14        //logic
15        for(int i = 1; i < n-1; i++){
16            if(A[i] > A[i+1] && A[i] > A[i-1]){
17                System.out.print(A[i] + " ");
18            }
19        }
20
21    }
22
23 }
24 }
```

Peak Index in a Mountain Array

QUESTION

A mountain array is an array that increases to a peak and then decreases. More formally, an array arr is a mountain array if and only if:

- $arr[0] < arr[1] < \dots < arr[i - 1] < arr[i] > arr[i + 1] > \dots > arr[arr.length - 1]$

Given an integer mountain array arr , return the index i that corresponds to the peak of arr . Your implementation must run in $O(\log n)$ time.

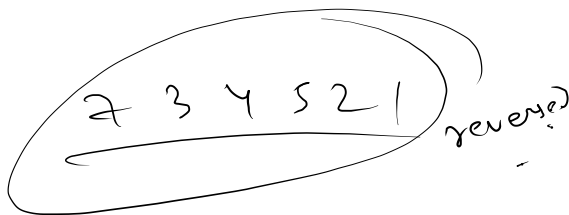
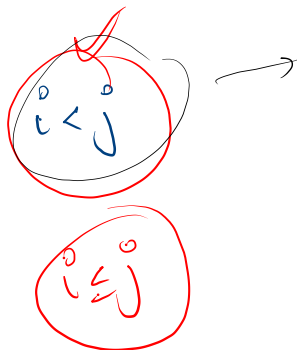
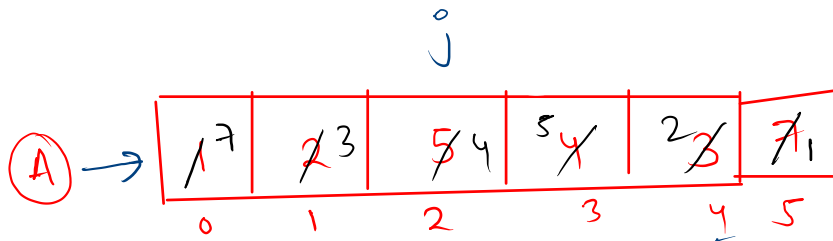
ANSWER

```
class Solution {
public:
    int peakIndexInMountainArray(vector<int> &arr) {
        int left = 0, right = arr.size() - 1;
        while (left < right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] < arr[mid + 1])
                left = mid + 1;
            else
                right = mid;
        }
        return left;
    }
};
```

GKSTR32 Reverse_Array

0
1

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



```

public 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);
        }
    }
}

```

NO. 1 2 3 4 5 6 7

↳ 1 2 6 5 4 3 7

reverse(A, start, end)
index.

Type Casting.

implicit

* done by java

automatic

eg- converting
char → int

int v = 'a';

explicit

done by programmer

manual

eg- converting
int → char

char ch = (char) (72);

int = char