Revision :-

- ✓ Sorting , lambda function
- ✓ arrays, subarray, Kadane's algo
- ✓ 2 pointers
- ✓ Prefix array
- ✓ Arrays as hashmap
- ✓ 2d array
- ✓ String & substring
- ✓ Binary Search (BSLB & BSUB)
- ✓ ArrayList
- Stacks
- Hashmap
- Queue
- PQ

⇒ Binary Search

```
int si=0; ei=n-1;
while( si <= ei ){
    int mid = (si+ei)/2;
    if( arr[mid] == target){


    } else if( arr[mid] < target){
        si = mid+1;
    } else{
        ei = mid-1;
    }
}
```

UB

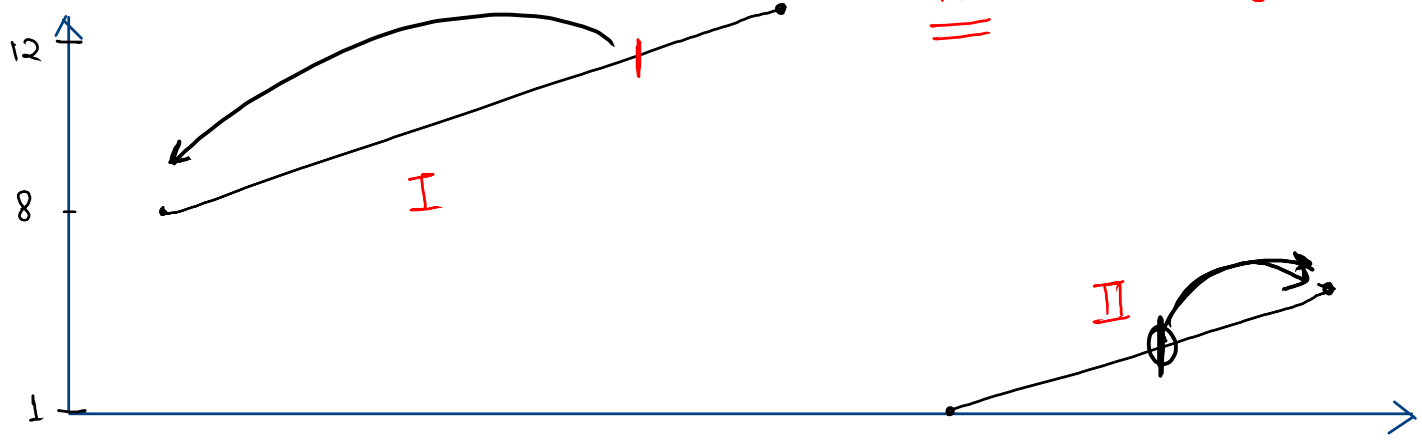
```
if(mid< n-1 && arr[mid] == arr[mid+1])
        si = mid+1;
else
        return mid;
```

LB

```
if(mid>0 && arr[mid] == arr[mid-1])
        ei = mid-1;
else
        return mid;
```

Find The Index of Rotation

$$\left[\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & \underset{\overset{\uparrow}{si}}{12} & \underset{\overset{\uparrow}{\underline{\underline{mid}}}}{\textcircled{1}} & 3 & \underset{\overset{\uparrow}{ei}}{4} \end{array}\right]$$



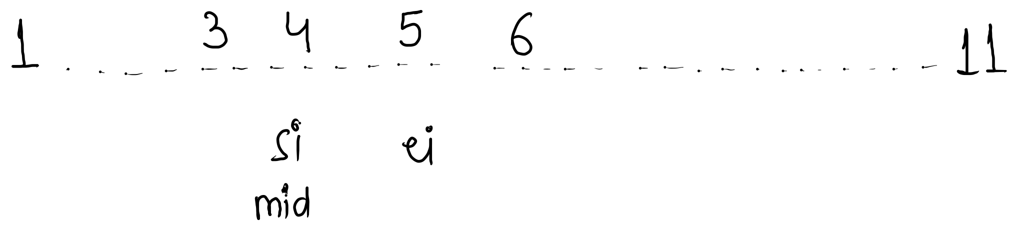
```
si=0, ei=n-1;
while (si<=ei){
    int mid=(si+ei)/2;
    if( arr[mid] <=arr[mid-1] && arr[mid]<=arr[mid+1]){
        return mid-1;
    } else if( arr[mid] <=arr[ei]){
        ei=mid-1;
    } else if(arr[mid] >=arr[si]) {
        si=mid+1;
    }
}
```

prev = (mid-1+n) % n;
next = (mid+1) % n;

The banana challenge

h = 8

arr	3	6	7	11



check → 1) 3/6 = 0
3%6 != 0 , time = 1
2) 6/6 = 1
6%6 != 0 ~~d~~time = 1
3) 7/6 = 1
7%6 != 0 , time = 2
4) 11/6 = 1
11%6 != 0 , time = 2
sum= 6

check → 1) 3/3 = 1
3 % 3 != 0 time = 1
2) 6/3 = 2
6%3 != 0 , time = 2
3) 7/3 = 2
7%3 != 0 time = 3
4) 11/3 = 3
11%3 != 0 time = 4
sum= 10

check → 1) 3/4 = 0
3 %4 != 0, time = 1
2) 6/4 = 1
6%4 != 0 , time = 2
3) 7/4 = 1
7%4 != 0, time = 2
4) 11/4 = 2
11%4 != 0 time = 3
sum= 8

Code

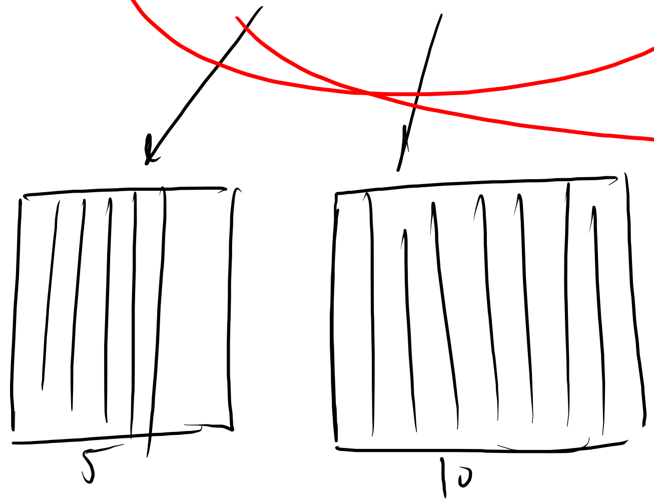
```java
public static int kokoEatingBananas(int n, int[] arr, int totalTime) {
    int si = 1;
    int ei = max(arr);
    while ( si <= ei ) {
        int mid = (si + ei) / 2;    // speed
        if ( check(arr, mid, totalTime) == true ) {
            ei = mid - 1;
        } else {
            si = mid + 1;
        }
    }
    return si;
}

public static boolean check(int[] arr, int speed, int totalTime) {
    int time = 0;
    for (int i = 0; i < arr.length; i++) {
        time += arr[i] / speed;
        if ( arr[i] % speed != 0 ) {
            time++;
        }
    }
    if ( time > totalTime ) {
        return false;
    } else {
        return true;
    }
}
```

Painters = 2

total boards = 70

arr

0	1	2	3	4
5	10	15	35	5



$\overbrace{5 + 10 + 15}^{p=1} + \overbrace{35+5}^{p=2}$

~~30~~ ~~65~~

$\boxed{P=2}$

p = 1

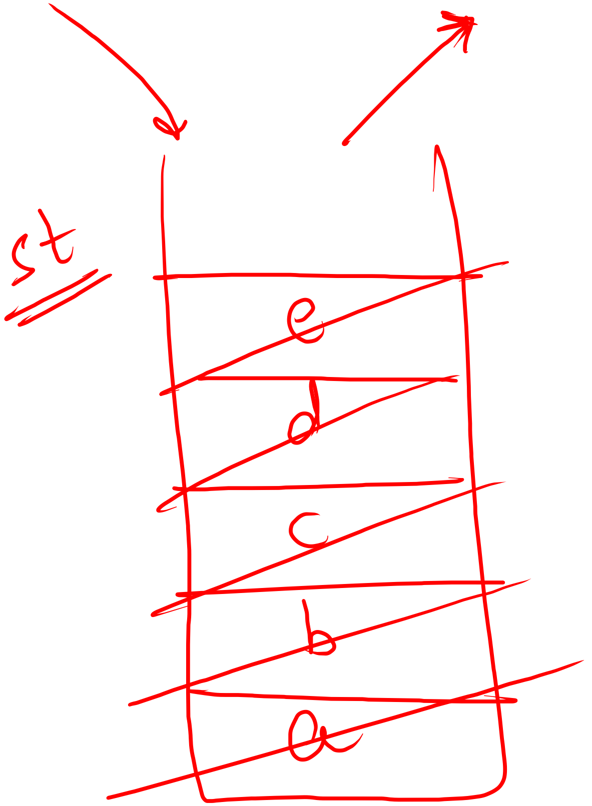
p = 5

35 - - - - - - - 52 - - - - - - - 70 time

si mid ei

⇒ Stack

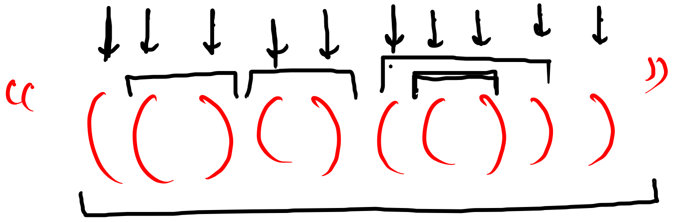
(LIFO) same
(FILO)

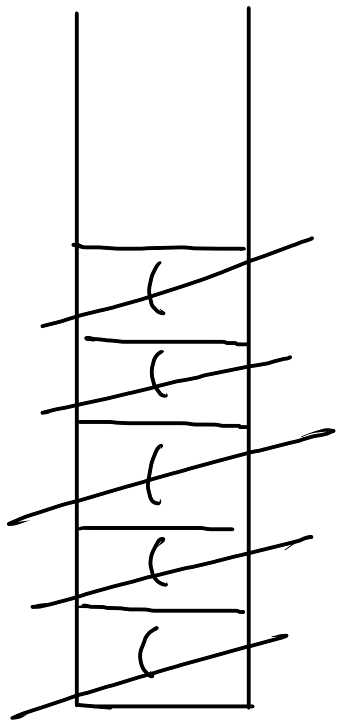


abcde

edcba

valid parentheses 10





<u>when to pop</u>

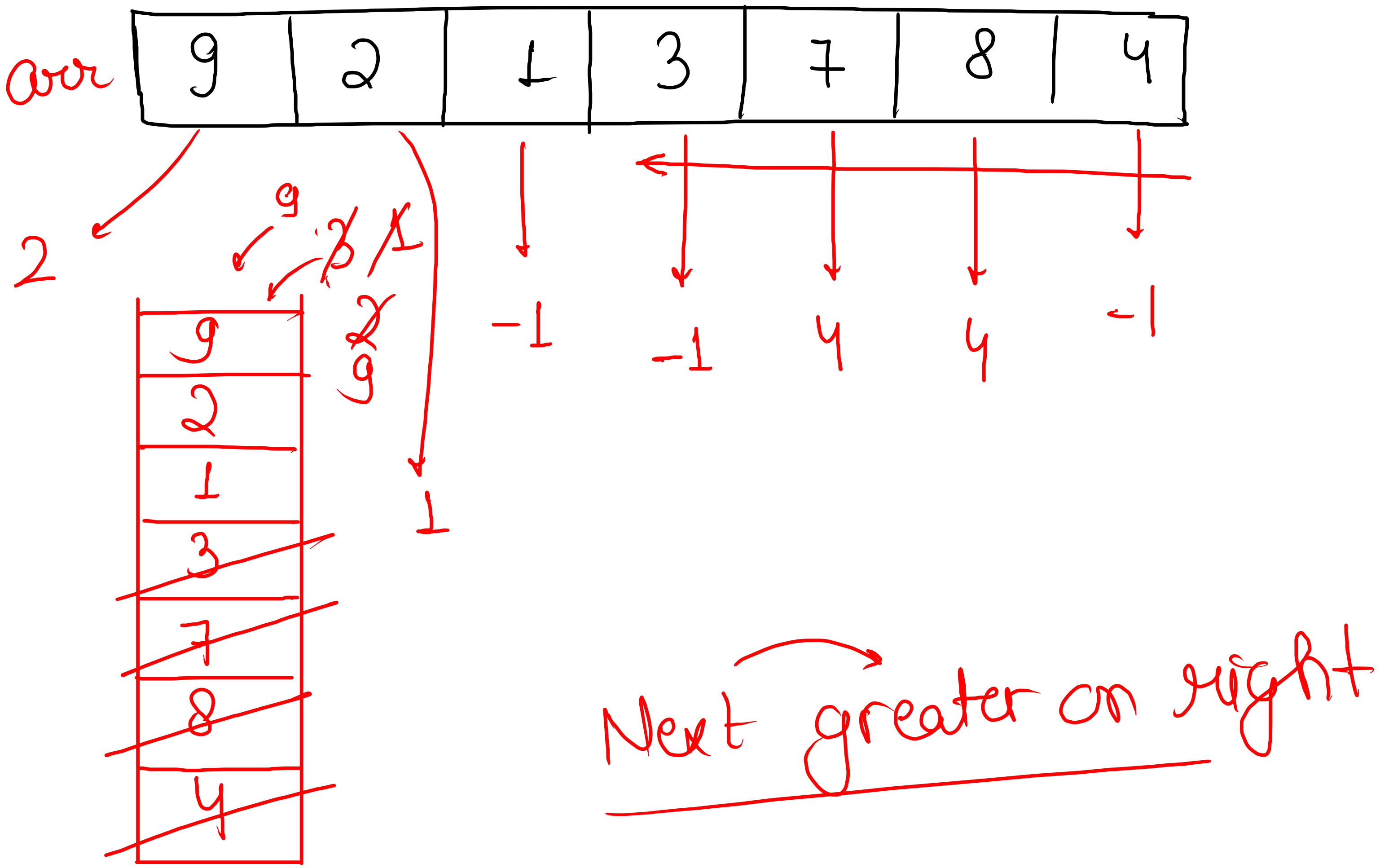
1) stack should not be empty

2) curr = ')'

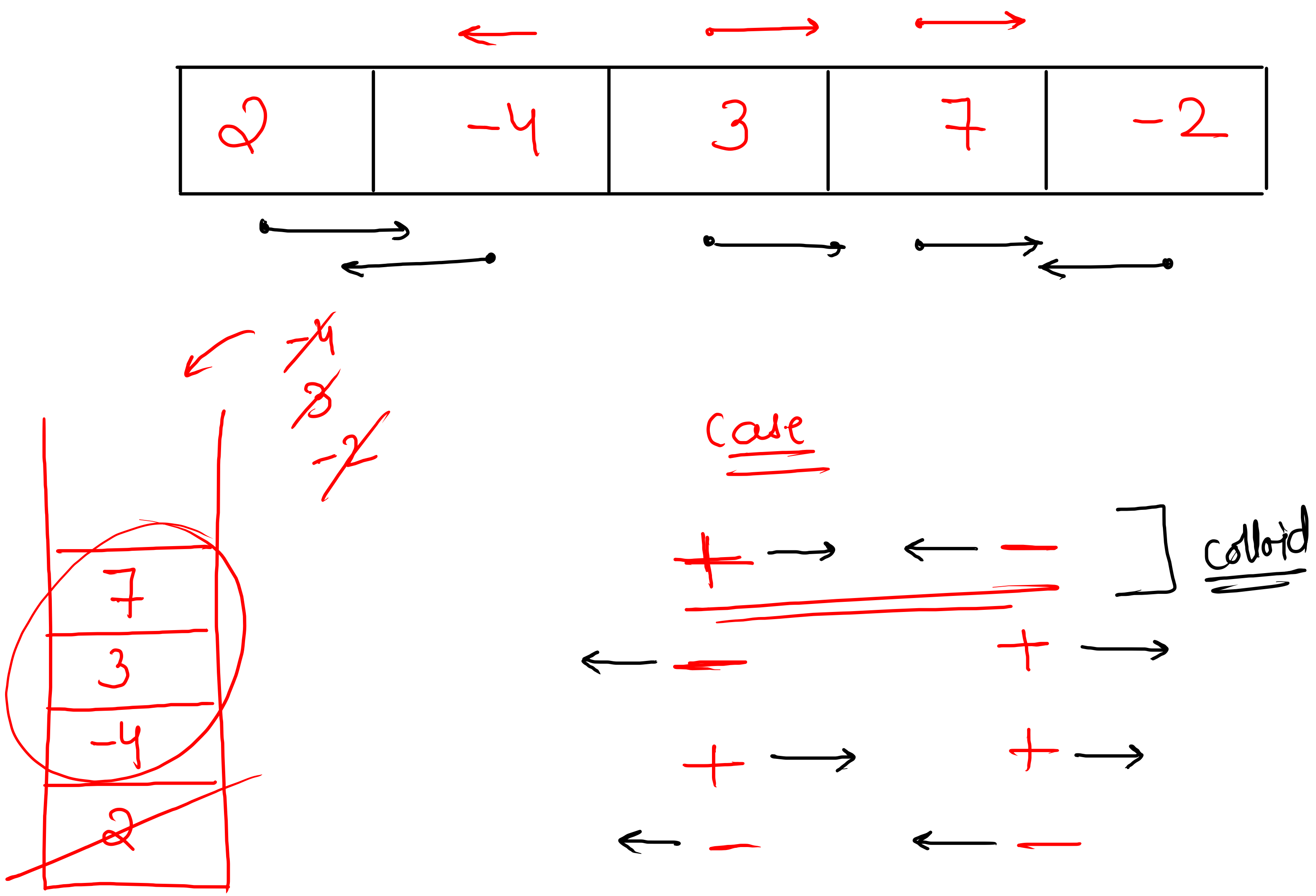
top = '('

then pop

Next Smaller Element To The Right



Asteroid Collision



Code

```
public static void astroidCollision(int[] arr, int n) {
    Stack<Integer> st = new Stack<>();
    for (int i = 0; i < n; i++) {
        if ( arr[i] > 0 ) {
            st.push( arr[i] );
        } else {
            while ( !st.isEmpty() && st.peek() > 0 && st.peek() < -1 * arr[i] ) {
                st.pop();
            }
Imp         if ( !st.isEmpty() && st.peek() == -1 * arr[i] ) {
                st.pop();
            } else if ( st.isEmpty() || st.peek() < 0 ) {
                st.push( arr[i] );
            }
        }
    }
    ArrayList<Integer> ans = new ArrayList<>();
    while ( st.size() > 0 ) {
        int ele = st.pop();
        ans.add( 0, ele );
    }

    for (int i : ans) {
        System.out.print(i + " ");
    }
}
```