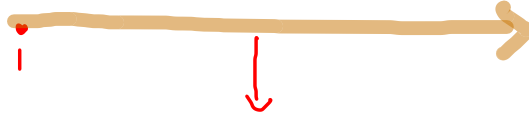


Recursion
with arrays

virtually short
array

10	20	30	40	50	60
0	1	2	3	4	5



10	20	30	40	50	60
0	1	2	3	4	5



10	20	30	40	50	60
0	1	2	3	4	5



10	20	30	40	50	60
0	1	2	3	4	5



Display
array
L to R

10	20	30	40	50	60
0	1	2	3	4	5

i



faith —, display(arr, i+1);

expectation [print(arr[i]);
display(arr, i+1);

arr =

10	20	30	40	50
0	1	2	3	4

arr = null

```
public static void displayArr(int[] arr, int idx){
    if(idx == arr.length) {
        return;
    }

    1 System.out.println(arr[idx]);
    2 displayArr(arr, idx+1);
}
```

10 20 30 40 50

dA	arr = null, i = 5	
dA	arr = null, i = 4	1 2
dA	arr = null, i = 3	1 2
dA	arr = null, i = 2	1 2
dA	arr = null, i = 1	1 2
dA	arr = null, i = 0	1 2

arr =

10	20	30	40	50
0	1	2	3	4



50 40 30 20

with

10

self work

```
public static void displayArrReverse(int[] arr, int idx) {
    if (idx == arr.length) {
        return;
    }
    1 displayArrReverse(arr, idx+1);
    2 System.out.println(arr[idx]);
}
```

50 40 30 20 10

do	arr = 41, i = 5	
do	arr = 41, i = 4	12
do	arr = 41, i = 3	12
do	arr = 41, i = 2	12
do	arr = 41, i = 1	12
do	arr = 41, i = 0	12

11	7	6	13	2	5
0	1	2	3	4	5



$Sam = \max(arr, idx + 1);$
 (small array
 max)

$Omax = \text{Math.max}(arr[idx], Sam);$

return Omax;

```

public static int maxOfArray(int[] arr, int idx){
    if(idx == arr.length) {
        return Integer.MIN_VALUE; //max identity -> -Infinity
    }

    1int sam = maxOfArray(arr,idx+1); //smaller array max
    2int omx = Math.max(arr[idx],sam);

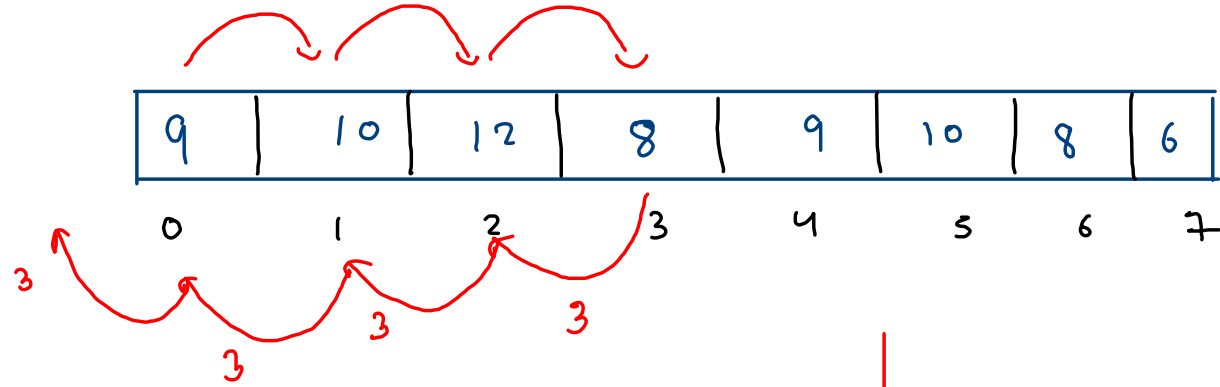
    3return omx;
}

```

11	7	6	13	2	5
0	1	2	3	4	5

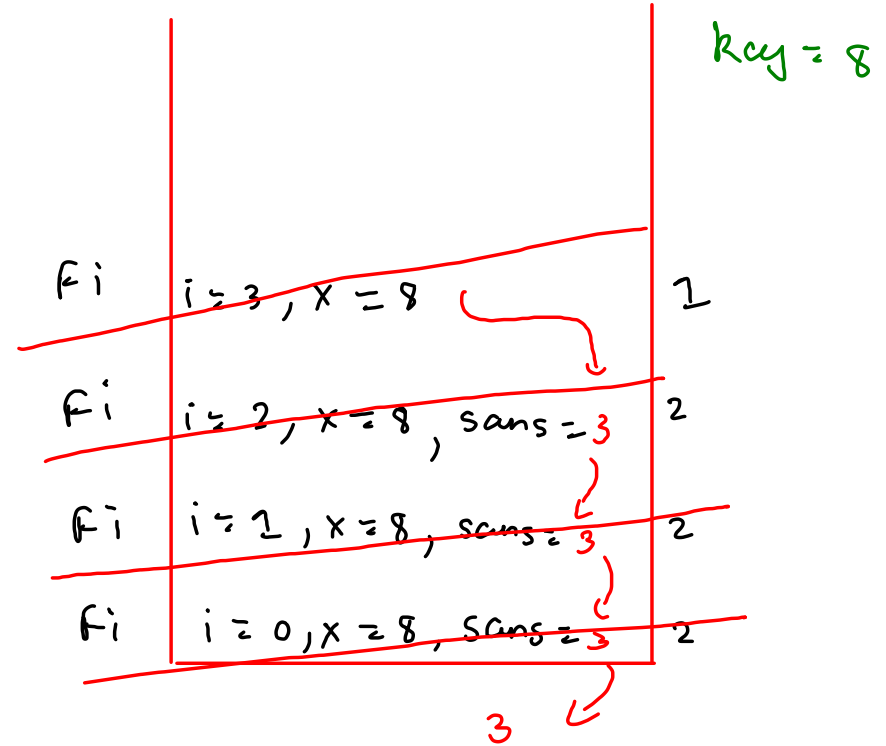
max	$i = 6$		base case
max	$i = 5$	$sam = -\infty, om = 5$	1 2 3
max	$i = 4$	$sam = 5, om = 5$	1 2 3
max	$i = 3$	$sam = 5, om = 13$	1 2 3
max	$i = 2$	$sam = 13, om = 13$	1 2 3
max	$i = 2$	$sam = 13, om = 13$	1 2 3
max	$i = 0$	$sam = 13, om = 13$	1 2 3
		13	

First index



```
public static int firstIndex(int[] arr, int idx, int x){
    if(idx == arr.length) {
        return -1;
    }

    if(arr[idx] == x) {
        return idx;
    }
    else {
        int sans = firstIndex(arr, idx+1, x); //smaller array ans
        return sans;
    }
}
```



last index

9	8	12	8	9	10	8	6
0	1	2	3	4	5	6	7

$x = 8$

```

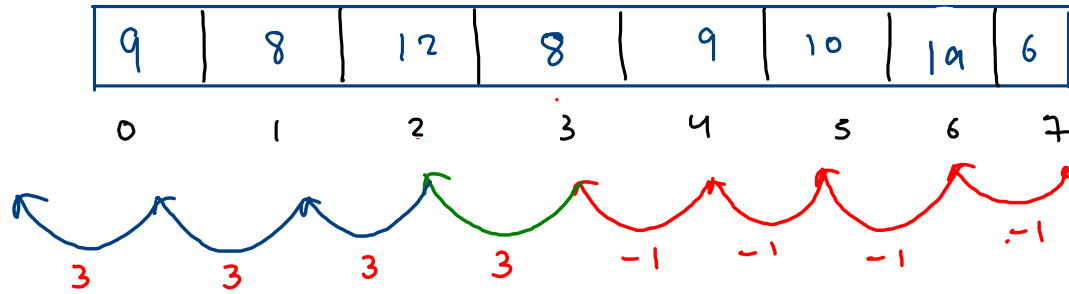
public static int lastIndex(int[] arr, int idx, int x){
    if(idx == arr.length) {
        return -1;
    }

    2 int lisa = lastIndex(arr, idx+1, x); //last index in smaller array

    2 if(lisa != -1) {
        return lisa;
    }
    else {
        3 if(arr[idx] == x) {
            return idx;
        }
        else {
            return -1;
        }
    }
}

```

li	i = 8		base
li	i = 7, lisa = -1		1 36
li	i = 6, lisa = -1		1 3a
li	i = 5, lisa = 6		1 2
li	i = 4, lisa = 6		1 2
li	i = 3, lisa = 6		1 2
li	i = 2, lisa = 6		1 2
li	i = 1, lisa = 6		1 2
li	i = 0, lisa = 6		1 2
		6	



$x = 8$

```

public static int lastIndex(int[] arr, int idx, int x){
    if(idx == arr.length) {
        return -1;
    }

    1 | int lisa = lastIndex(arr, idx+1, x); //last index in smaller array

    2 | if(lisa != -1) {
        return lisa;
    }

    3 | else {
        if(arr[idx] == x) {
            return idx;    a
        }
        else {
            return -1;    b
        }
    }
}

```

— → 3b

— → 3a

— → 2