

#

## Functions

⇒ Takes input and give some output -

$$f(a, b) = a^2 + 2ab + b^2$$

$$f(n) = \underline{2^{n+1}}$$

$f(x) = 2^{x+1}$

input  
 $n=5$

output  
 $2^{5+1} = 32$

parameters/arguments

```

public static int sum( int a, int b ) {
    int sum = a + b;
    return sum;
}

public static void fun( int a ) {
}
  
```

return type

$$\text{sum}(2, 3) = 5$$

$$\text{sum}(1, 4) = 5$$

$$\text{sum}(9, 11) = 20$$

RAM

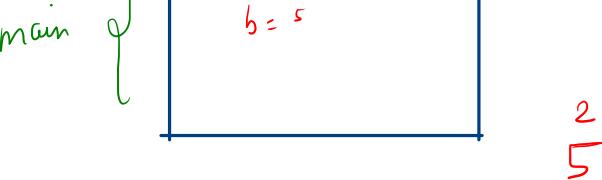
Stack

```
class Main {  
    public static int sum(int a, int b){  
        int sum = a + b;  
        return sum;  
    }  
    Run | Debug  
    public static void main(String[] args) {  
        int a= 2;  
        int b= 3;  
  
        int ans = sum(a,b);  
        System.out.println(ans);  
    }  
}
```

5

```
1- class Main {  
2-     public static int sum(int a, int b){  
3-         int sum = a + b;  
4-         return sum;  
5-     }  
6-  
7-     public static void fun(int a, int b){  
8-         int sum = a+b;  
9-         System.out.println(sum);  
10-  
11-         System.out.println("I am inside a void function");  
12-         System.out.println("I am inside a void function");  
13-     }  
14-  
15-     public static void main(String[] args) {  
16-         int a= 2;  
17-         int b= 3;  
18-         fun(a,b);  
19-     }  
20- }  
21 }
```

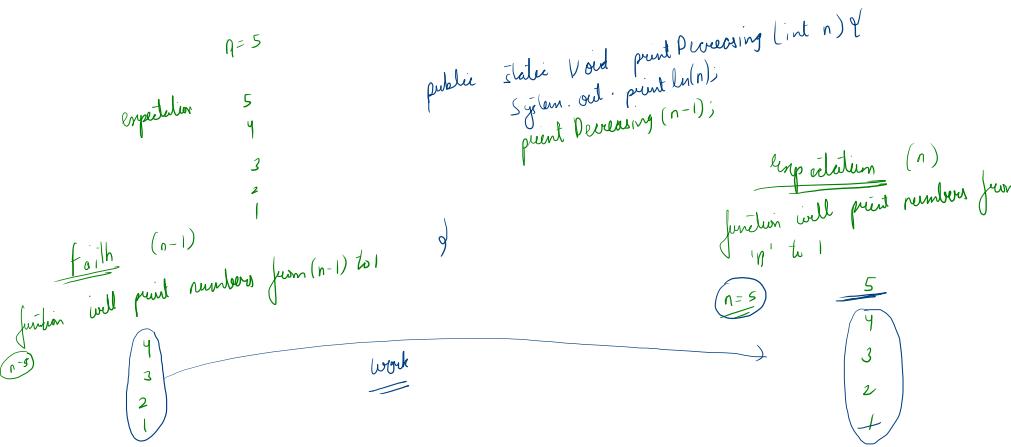
```
class Main {  
    public static void swap(int a, int b){  
        int temp = a;  
        a = b;  
        b = temp;  
    }  
    Run | Debug  
    public static void main(String[] args) {  
        int a= 2;  
        int b= 5;  
  
        swap(a, b);  
  
        System.out.println(a);  
        System.out.println(b);  
    }  
}
```



## # Recursion

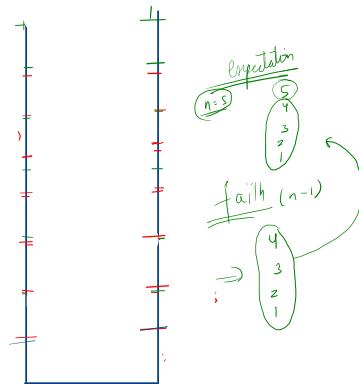
- How to solve any recursion problem
- 1) keep a faith that it works for a smaller problem (faith)
  - 2) solve for the smallest problem (base case?)
  - 3) solve for the actual problem (to reach your expectation)

Ours Print decreasing numbers from 'n' to 1 using recursion.



you won't be able to learn recursion without dry-Hum-

stack overflow



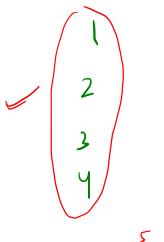
```
class Main {  
    public static void printDecreasing(int n){  
        if(n==0){  
            return;  
        }  
        System.out.println(n);  
        printDecreasing(n-1);  
    }  
}  
Run | Debug  
public static void main(String[] args) {  
    int n = 5;  $\underline{\text{printDecreasing}(n)}$   
}
```

3  
2  
1

Ques Print numbers in increasing order using recursion.

Faith ( $n-1$ )

It will work for ( $n-1$ )  
print numbers from 1 to ' $n-1$ '



```
psv printIncreasing(int n){  
    printIncreasing(n-1);  
    System.out.println(n);  
}
```

expectation (n)

function will print numbers  
from 1 to  $n$

$n=5$       |  
                2  
                3  
                4  
→ 5

```
public static void printIncreasing(int n){  
    if(n==0){  
        return;  
    }  
    printIncreasing(n-1);  
    System.out.println(n);  
}  
Run | Debug  
public static void main(String[] args) {  
    int n = 3;  
    printIncreasing(n);  
}
```

1

2

3

Ques Print numbers in decreasing order, then increasing using recursion.

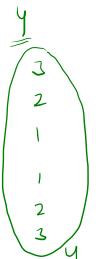
Faith  $(n-1)$   
It will print numbers  
from  $(n-1)$  to 1 and then 1 to  $(n-1)$

$n=1$   
3  
2  
1  
1  
2  
3

$n=5$

psv pdi(int n){  
System.out.println(n);  
pdi(n-1);  
System.out.println(n);

}



-

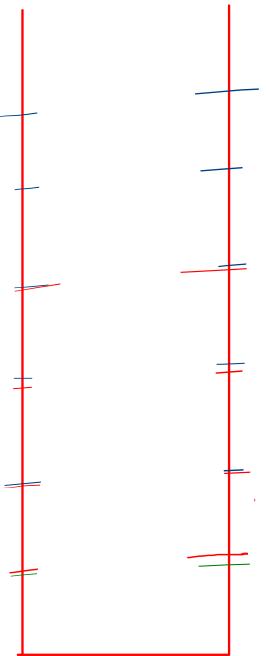
Expectation

It will print numbers in decreasing  
then increasing order

$n=4$

4  
3  
2  
1  
1  
2  
3  
4

```
public static void pdi(int n){  
    if(n==0){  
        return;  
    }  
    System.out.println(n);  
    pdi(n-1);  
    System.out.println(n);  
}  
Run | Debug  
public static void main(String[] args) {  
    int n = 4; 2  
    pdi(n);  
}
```



Ques find factorial of given number 'n'.

Faith  $(n-1)$

It will return  
factorial  $(n-1)$

$1 \times 2 \times 3 \times 4$   
smallAns

public static int fac(int n) {

int smallAns = fac(n-1); → faith  
int ans = smallAns \* n; → work  
return ans;

}

work

lexipetation

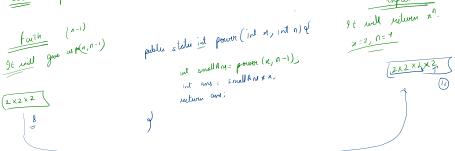
It will return factorial  
of given number

n=5  
P  
 $1 \times 2 \times 3 \times 4 \times 5$

$2^4$

```
public static int factorial(int n){  
    if(n==0){  
        return 1;  
    }  
    int smallAns = factorial(n-1);  
    int ans = smallAns * n;  
    return ans;  
}  
Run | Debug  
public static void main(String[] args) {  
    int n = 4;  
    int ans = factorial(n);  
    System.out.println(ans);  
}
```

Ques Find  $x^n$  using recursion



```
public static int power(int x, int n){  
    if(n==0){  
        return 1;  
    }  
    int smallAns = power(x, n-1);  
    int ans = smallAns * x;  
    return ans;  
}  
  
Run | Debug  
public static void main(String[] args) {  
    int x = 2;  
    int n = 4;  
    int ans = power(x, n);  
    System.out.println(ans);  
}
```

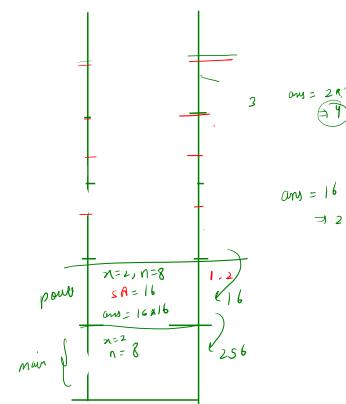
faith ( $n/2$ )  
it will calculate  
 $x^{\frac{n}{2}}$  correctly  
 $x=2, n=6$

actual  
 $2 \times 2 \times 2$

public static int power(int x, int n){  
it will return  $x^n$ .  
int smallAns = power(x, n/2);  
int ans = smallAns \* smallAns;  
y = (n/2)! = 0  
y = ans \* x;  
return ans;

expectation  
 $x=2, n=6$

$2 \times 2 \times 2 \times 2 \times 2 \times 2$



```
public static int power2(int x, int n){  
    if(n==0){  
        return 1;  
    }  
    int smallAns = power2(x, n/2);  
    int ans = smallAns * smallAns;  
  
    if(n%2 != 0){  
        ans = ans*x;  
    }  
    return ans;  
}  
  
Run | Debug  
public static void main(String[] args) {  
    int x = 2;  
    int n = 6;  
    int ans = power2(x, n);  
    System.out.println(ans);  
}
```

$$\begin{aligned} 2^8 &\Rightarrow 2^2 \times 2^2 \\ &\Rightarrow 2^4 \times 2^4 \\ &\Rightarrow 2^8 \\ \\ 2^1 &\Rightarrow 2^{\frac{1}{2}} \times 2^{\frac{1}{2}} \\ &\Rightarrow 2^{\frac{3}{2}} \times 2^{\frac{3}{2}} \\ &\Rightarrow 2^6 \times 2^2 \end{aligned}$$

$$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \frac{n}{8} \rightarrow \frac{n}{16} \dots \underset{x}{\cancel{1}} \quad O(\alpha)$$

$$\frac{n}{2^0} \rightarrow \frac{n}{2^1} \rightarrow \frac{n}{2^2} \rightarrow \frac{n}{2^3} \rightarrow \frac{n}{2^4} \rightarrow \frac{n}{2^5} \rightarrow \frac{n}{2^6} \dots \underset{x}{\cancel{1}} \quad O(\alpha)$$

$$\log a^b = b \log a$$

$$\frac{n}{2^k} = 1$$

$$2^k = n$$

$$\log_2 n = \log n$$

$$x \log_2 x = \log n$$

$$x \log_2 x = \log n$$

$$x \cdot 1 = \log n$$

$$(x = \log n)$$

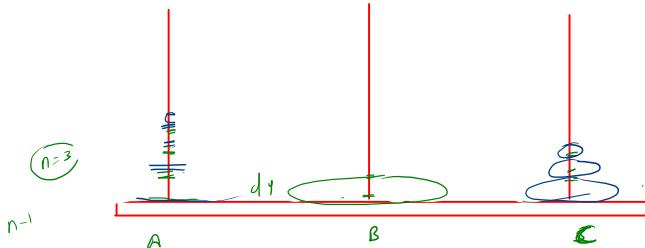
$$O(\log n)$$

Ques

## Tower of Hanoi

shift  $n$  discs from tower A to tower B  
with the help of tower C

- you can move one disc at a time
- you can not put a larger disc on top of a smaller disc



Path

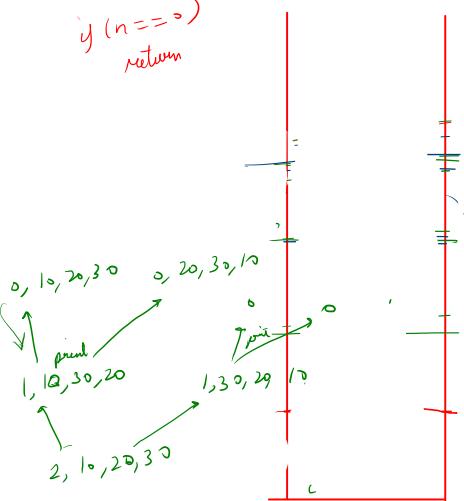
move  $(n-1)$  discs from A to C using B  
→ move nth disc from A to B  
→ move nth disc from A to B (move 3rd disc)  
move  $(n-1)$  discs from C to B (move 2nd disc)  
move  $(n-1)$  discs from C to B (move 1st disc)  
→ move  $(n-1)$  discs from C to B (move 2nd disc)  
→ move  $(n-1)$  discs from C to B (move 3rd disc)

psv  $\text{ToH}(n, \underline{\text{A}}, \underline{\text{B}}, \underline{\text{C}})$   
→  $\text{ToH}(n-1, \underline{\text{A}}, \underline{\text{C}}, \underline{\text{B}}) \rightarrow$  Path  
move nth disc from A to B  
→  $\text{ToH}(n-1, \underline{\text{C}}, \underline{\text{B}}, \underline{\text{A}})$

Expectation

function will move  $n$  discs from 'A' to 'B' using 'C'  
 $n=3$   
with steps to move 3 discs from A to B

$y(n=3)$   
return

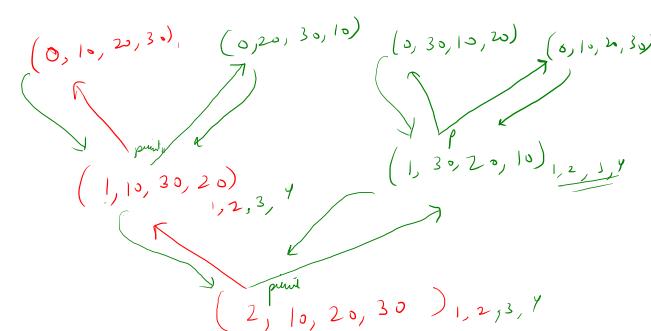


Moving ring 1 from 10 → 30 using 20  
Moving ring 2 from 10 → 20  
Moving ring 1 from 30 → 20

```
public static void towerOfHanoi(int n, int A, int B, int C){  
    if(n==0){  
        return;  
    }  
  
    towerOfHanoi(n-1, A, C, B);  
    // move nth disc from A to B  
    System.out.println("Moving ring " + n + " from " + A + " to " + B + " using " + C);  
    towerOfHanoi(n-1, C, B, A);  
}  
  
Run | Debug  
public static void main(String[] args) {  
    int n = 3;  
    towerOfHanoi(3, A: 10, B: 20, C: 30);  
}
```

Recursive tree diagram

→ color path

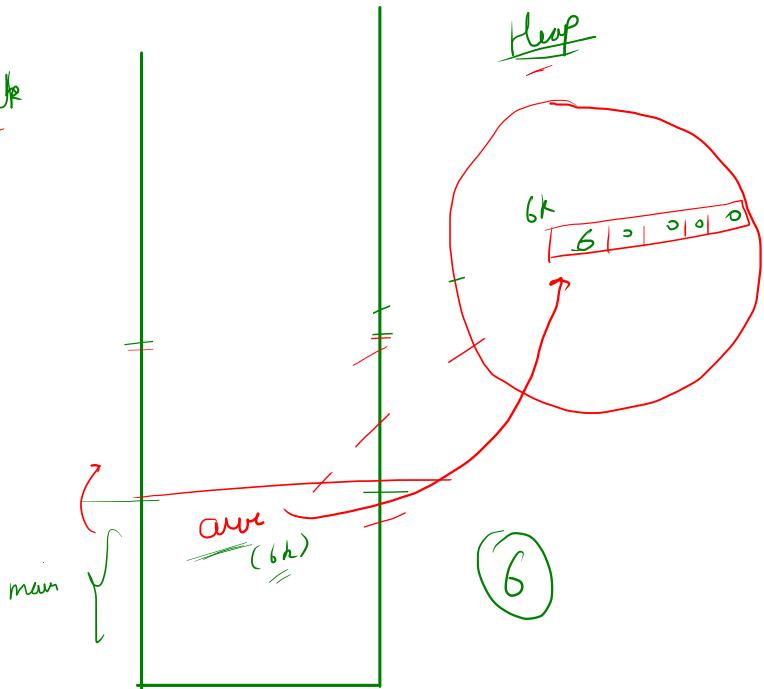


#

# Recursion with Arrays

RAM

stack



(NEW)

```

public class Main {
    public static void increaseArray(int[] arr){
        arr[0]++;
    }

    public static void increaseInteger(int a){
        a++;
    }
}

Run | Debug
public static void main(String[] args) {
    int[] arr = new int[1];
    arr[0] = 5;

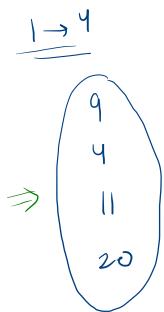
    int a = 5;
    increaseArray(arr);
    increaseInteger(a);

    System.out.println(arr[0]);
    System.out.println(a);
}

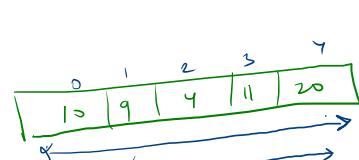
```

Ques Print array using recursion.

Faith (smaller area)  
It will print array from  
of  $idx+1$  to end

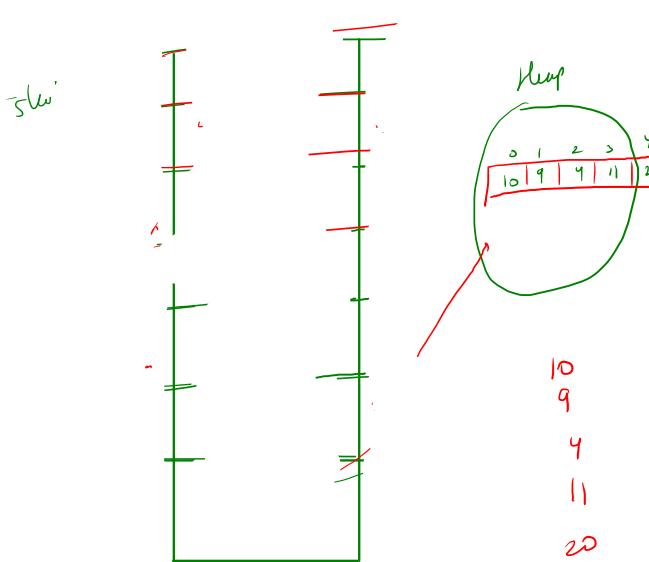
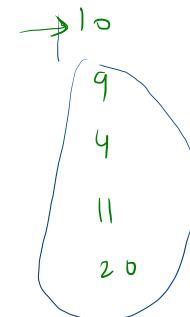


psu printArray(int[] arr, int idx){  
 System.out.println(arr[idx]);  
 printArray(arr, idx+1);  
}



printArray(arr, 0)

expectation  
function will print  
array from 'idx' to end  
(n-1)  
from 0 → 4



```
public static void printArray(int[] arr, int idx){  
    if(idx == arr.length){  
        return;  
    }  
    System.out.println(arr[idx]);  
    printArray(arr, idx+1);  
}  
Run | Debug  
public static void main(String[] args) {  
    int[] arr = {10, 9, 4, 11, 20};  
    printArray(arr, 0);  
}
```

Ours

Print array in reverse

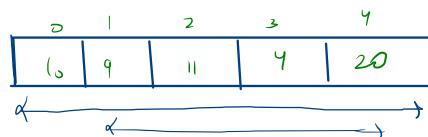
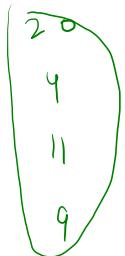
Fault

It will print from  
( $idx+1$ ) to end in reverse.

PSV  
printReverse (int [] arr, int idx) {

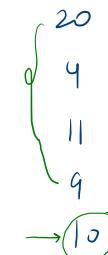
    printReverse (arr, idx+1);  
    System.out.println (arr[idx]);

}



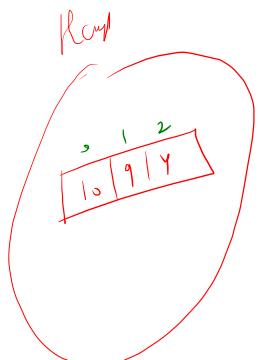
print reverse (arr, 0)

Expectation  
It will print from 'idx'  
to end in reverse  
 $0 \rightarrow 4$  in reverse



if ( $idx == arr.length$ )  
    return;

```
public static void printReverse(int[] arr, int idx){  
    1 printReverse(arr, idx+1);  
    2 System.out.println(arr[idx]);  
}  
Run | Debug  
public static void main(String[] args) {  
    int[] arr = {10, 9, 4, 11, 20};  
    printReverse(arr, idx: 0);  
}
```



Man of { arr, 0

4  
9  
10

Ours Find max of array using recursion.

Fault

get will return max of a smaller area.

smallAns = 1

ps int manOfArray (int[] arr, int idx) {

    int smallAns = manOfArray(arr, idx+1);  
    int ans = Math.max(arr[idx], smallAns);  
    return ans;

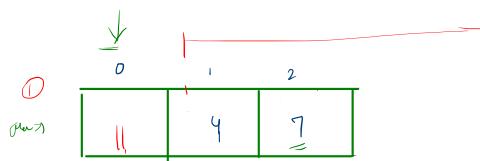
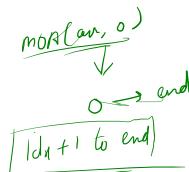
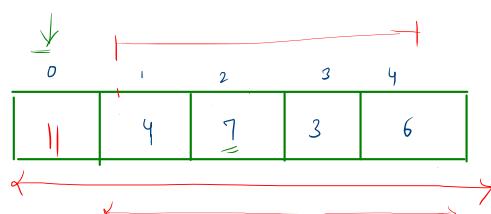
}

expectation

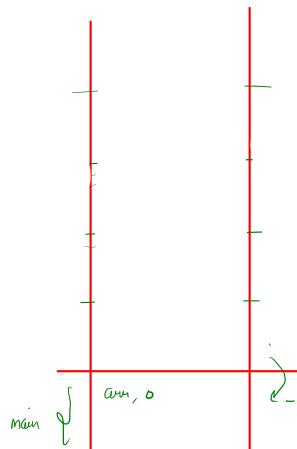
It will return 'max' from

(idx) to end.  
 $idx = 0$  to  $n-1$

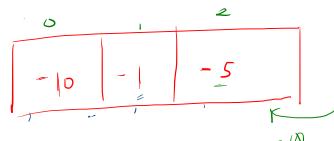
ans = 11



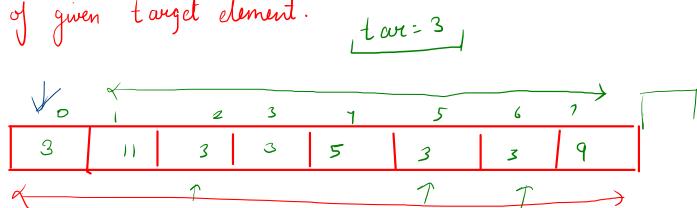
```
public static int maxOfArray(int[] arr, int idx){  
    if(idx == arr.length){  
        return 0; // Integer Min Value  
    }  
  
    int smallAns = maxOfArray(arr, idx+1);  
  
    int ans = Math.max(arr[idx], smallAns);  
    return ans;  
}  
Run | Debug  
public static void main(String[] args) {  
    int[] arr = {10, 9, 4, 11, 20};  
    System.out.println(maxOfArray(arr, idx: 0));  
}
```



arr  $\Rightarrow$



Ques Find first index of given target element.



fun (arr, 0, 3) = 0  
fun (arr, 3, 3)

Expectation

It will return the first index of the given target element.

ans = 0

Faith

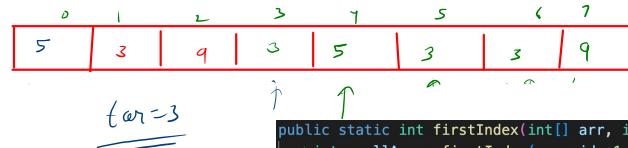
It will return first index of tar from 'idx+1' to end

```
public static int firstIndex(int[] arr, int idx, int tar) {
    int smallAns = firstIndex(arr, idx+1, tar);
    if (arr[idx] == tar) {
        ans = idx;
    } else {
        ans = smallAns;
    }
    return ans;
}
```

return ans;

)

(arr, 8, 3)  
(arr, 7, 3)  
(arr, 6, 3)  
(arr, 5, 3)  
(arr, 4, 3)  
(arr, 3, 3)  
(arr, 2, 3)  
(arr, 1, 3)  
(arr, 0, 3)



```
public static int firstIndex(int[] arr, int idx, int tar){
    int smallAns = firstIndex(arr, idx+1, tar);
    int ans = 0;

    if(arr[idx] == tar){
        ans = idx;
    } else {
        ans = smallAns;
    }

    return ans;
}
```

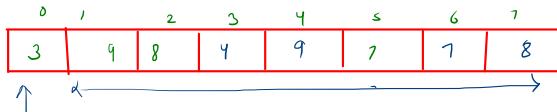
```
Run | Debug
public static void main(String[] args) {
    int[] arr = {10, 9, 4, 11, 20};
    System.out.println(maxOfArray(arr, idx: 0));
}
```

calculation in the way down

smallest problem  
{ if (idx = arr.length)  
 return -1

Ques Find last index of tar

tar = 3



Faith

from 'idx+1' to end  
it will find the  
last index to tar

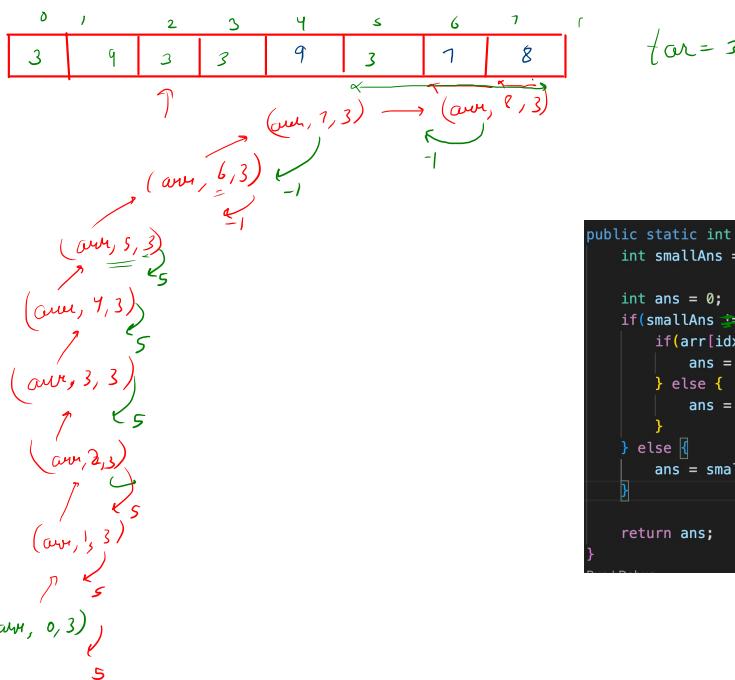
smallAns = -1

```
ps int lastIndex(int[] arr, int idx, int tar) {
    int smallAns = lI(arr, idx+1, tar);
    int ans = 0;
    if (smallAns == -1) {
        if (arr[idx] == tar)
            ans = idx;
    } else
        ans = smallAns;
    return ans;
}
```

Intuition

From '0' to end, it  
will return last index  
of target.

ans = 5



```
public static int lastIndex(int[] arr, int idx, int tar){
    int smallAns = lastIndex(arr, idx+1, tar);

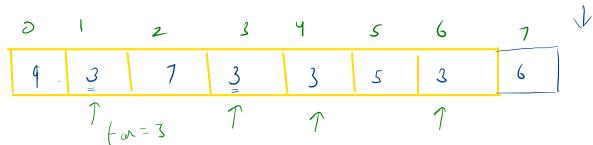
    int ans = 0;
    if(smallAns == -1){
        if(arr[idx]==tar){
            ans = idx;
        } else {
            ans = smallAns;
        }
    } else {
        ans = smallAns;
    }

    return ans;
}
```

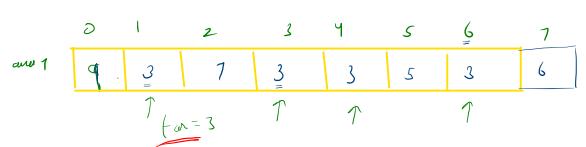
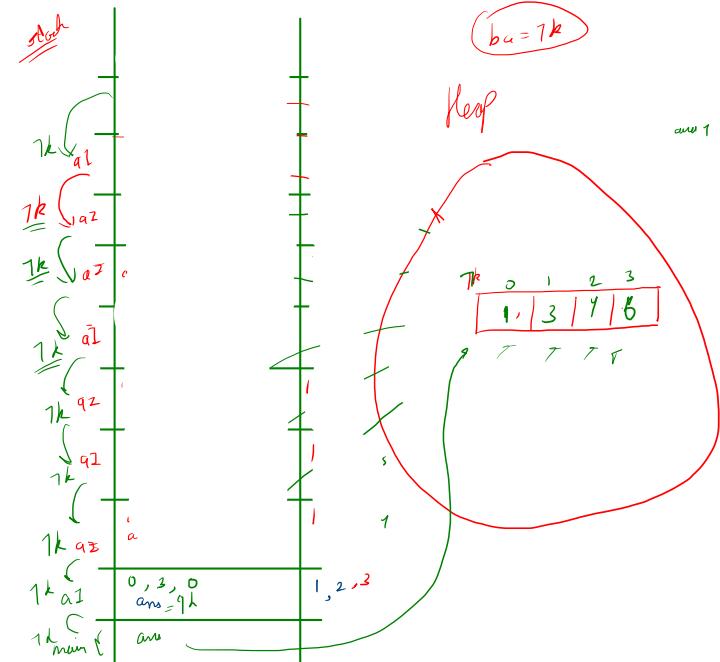
Ques Find all indices of tar in array

(arr, 8, 3, 4)  
(arr, 7, 3, 4)  
(arr, 6, 3, 3)  
(arr, 5, 3, 3)  
(arr, 4, 3, 2)  
(arr, 3, 3, 1)  
(arr, 2, 3, 1)  
(arr, 1, 3, 0)  
(arr, 0, 3, 0)

public static int[] allIndices(int[] arr, int idx, int tar, int count) {



ans = [1 | 3 | 4 | 6]  
← (arr, 0, 3)



```
public static int[] allIndices(int[] arr, int idx, int tar, int count){  
    if(idx==arr.length){  
        int[] ba = new int[count];  
        return ba;  
    }  
  
    int[] ans;  
    if(arr[idx]==tar){  
        ans = allIndices(arr, idx+1, tar, count+1);  
    } else {  
        ans = allIndices(arr, idx+1, tar, count);  
    }  
  
    if(arr[idx]==tar){  
        ans[count] = idx;  
    }  
    return ans;  
}  
  
Run | Debug  
public static void main(String[] args) {  
    int[] arr = {0,3,7,3,3,5,3,6};  
    int[] ans = allIndices(arr, idx: 0, tar: 3, count: 0);  
  
    for(int i=0; i<ans.length; i++){  
        System.out.println(ans[i]);  
    }  
}
```

Ques Find  $n^{\text{th}}$  fibonacci number

| nth    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8  | 9  | 10 | 11 | 12 |     |
|--------|---|---|---|---|---|---|---|----|----|----|----|----|-----|
| $f(n)$ | 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 |

$$f(n) = f(n-1) + f(n-2)$$

$$f(0) = f(1) + f(0)$$

$$(n-1)$$

$$\begin{array}{l} n=6 \\ \hline n-1=5 \\ \hline n=5 \end{array}$$

$$f(b(n-1))$$

$$f(b(n-2))$$

length calculation

It will calculate  $n^{\text{th}}$  fibonacci term.

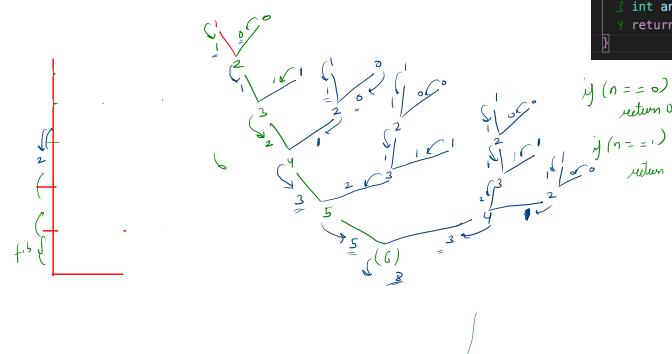
$$n=6$$

$$ans = ?$$

return ans;

$$ans = 1+1 \Rightarrow 2$$

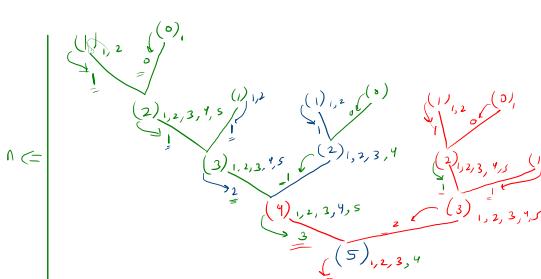
$$lt = 2$$



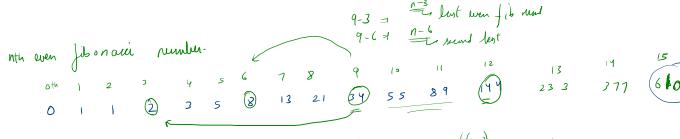
```
// fibonacci number
public static int fib(int n){
    int lt = fib(n-1);
    int lst = fib(n-2);
    int ans = lt + lst;
    return ans;
}
```

$T(n) \Rightarrow (\text{number of calls})^{\text{height of tree}}$

$2^n$



```
// fibonacci number
public static int fib(int n){
    if(n==0){
        return 0;
    }
    if(n==1){
        return 1;
    }
    int lt = fib(n-1);
    int lst = fib(n-2);
    int ans = lt + lst;
    return ans;
}
```



1st even fib number = 3rd fib number.  
2nd even fib number = 6th fib number.  
3rd even fib number = 9th fib number.  
4th even fib number =  $3 \times n$  fib number.

$$\begin{aligned}
 f(n) &\rightarrow f(n-1) + f(n-2) \\
 &\downarrow \quad \downarrow \\
 f(n-2) + f(n-3) + f(n-4) + f(n-5) &= \\
 f(n-3) + f(n-4) + f(n-5) + f(n-6) &= \\
 2f(n-3) + f(n-4) + f(n-5) &= \\
 3f(n-3) + f(n-4) + f(n-5) + f(n-6) &= \\
 \boxed{f(n)} &\Rightarrow 4(f(n-3) + f(n-6)) \Rightarrow \text{n}^{\text{th}} \text{ even fib number} \\
 &\quad \Rightarrow 4 \times (\text{last 3 even fib numbers}) \\
 &\quad + (\text{second last even fib number}) \\
 f_E(n) &\Rightarrow 4f_E(n-1) + f_E(n-2) \\
 &\quad \downarrow \quad \downarrow \\
 &\quad \text{last even fib number} \quad \text{second last even fib number.}
 \end{aligned}$$

$$\begin{array}{c}
 \boxed{1} \xrightarrow{\boxed{1+8}} \boxed{9} \xrightarrow{\boxed{1+9}} \boxed{18} \xrightarrow{\boxed{1+18}} \boxed{27} \xrightarrow{\boxed{2+27}} \boxed{30} \\
 \text{ans} = x + 4 \Rightarrow \boxed{\text{ans} = 12} \\
 \text{ans} / 10
 \end{array}$$

$$\begin{array}{r}
 10^{-2} \xrightarrow{\sim} 1 \\
 10^{-1} \cdot 9 \\
 10^{-2} \\
 27 \cdot 10^{-2} \\
 30 \cdot 10^{-2}
 \end{array}$$

$$\text{last even } \Rightarrow 10^9$$

$$\text{last } \Rightarrow 10^8$$

$$\begin{array}{l}
 \text{ans} \Rightarrow 4 \times 10^8 + 10^2 \\
 \Rightarrow 4 \times 10^8
 \end{array}$$

$$7 \times 10^8$$

$$\begin{array}{l}
 \text{ans } \xrightarrow{\sim} 10^8 \\
 -2^{12} \xrightarrow{\sim} \left( \frac{2^{12}-1}{2^{12}} \right) \cdot 10^8
 \end{array}$$

$$\begin{array}{l}
 \text{ans } \xrightarrow{\sim} 10^{10} \cdot 1 \cdot (\text{mod}) \\
 \xrightarrow{\sim} 2^{12-4} \cdot 10^{10} \cdot 1 \cdot (\text{mod})
 \end{array}$$

$$\text{last even } \Rightarrow 10^9$$

$$\text{last } \Rightarrow 3 \times 10^8$$

$$7 \times 10^8$$

$$-2^{12}$$

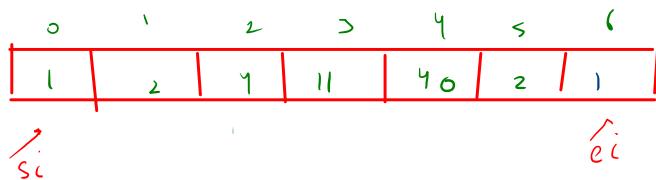
$$\left( \frac{2^{12}-1}{2^{12}} \right) \cdot 10^8$$

$$10^{10} \cdot 1 \cdot (\text{mod})$$

$$2^{12-4}$$

Ques

Check whether the given array is pallindromic or not.



(arr, 2, 4)  
f  
(arr, 1, 5)  
f  
(arr, 0, 6)  
f

```
public static boolean isPalindromic(int[] arr, int si, int ei) {
    if(arr[si] != arr[ei]){
        return false;
    }

    boolean ans = isPalindromic(arr, si + 1, ei - 1);
    return ans;
}
```

dynamic array

→ data type

ArrayList<String>

[dynamic array]

al = new ArrayList<>();

add

al.add("abc")

al.add("def")

al.add("abcd")

remove

al.remove(idn)

al.remove(1)

size

al.size()

→ [3]

get

al.get(idn);

cur →

| 0  | 1  | 2  |
|----|----|----|
| 10 | 13 | 19 |

set 45 at 1st index

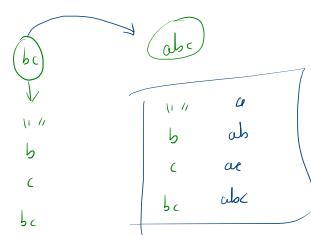
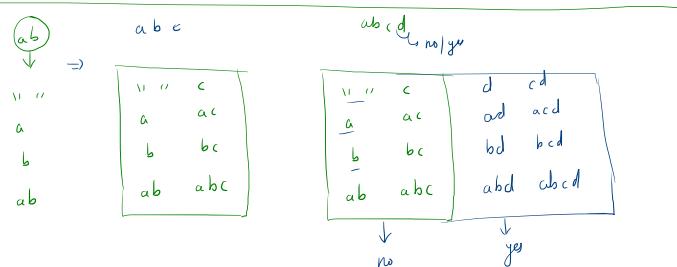
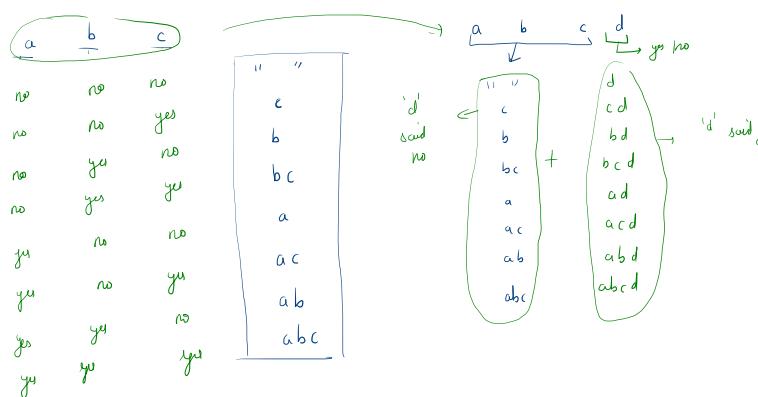
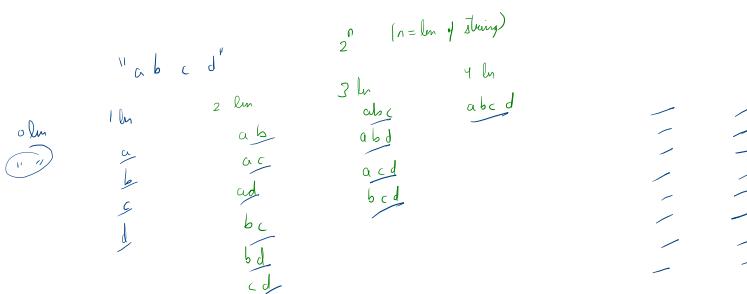
arr[1] = 45

set

al.set(idx, value)

# Strings

str = "a b c d e"

substring  $\Rightarrow$  continuous segment of string.subsequence  $\Rightarrow$  discontinuous segment of string

Ques Get all the subsequences of a given string.

Fault (smaller value)  
It will return all the  
subsequences for 'bc'

smallAns = { "", b, c, bc }

public static ArrayList<String> getSS (String str) {

abc

expectation

It will return all  
the subsequences of given  
string

str = abc

ans = { "", a, b, c, ab, bc, ac, abc }

work?

( "")  
↓  
{ ("") }  
  
(c)  
↑  
↓ ("", c)  
  
(bc)  
↑  
↓ ("", c, b, bc)  
  
( abc )  
↓  
↓ { ("", c, b, bc, a, ac, ab, abc) }

```
public static ArrayList<String> getSS(String str){  
    char fc = str.charAt(index: 0);  
    String ss = str.substring(beginIndex: 1);  
  
    ArrayList<String> sans = getSS(ss);  
  
    ArrayList<String> ans = new ArrayList<>();  
    // copy sans  
    for(int i=0; i<sans.size(); i++){  
        String ssub = sans.get(i);  
        ans.add(ssub);  
    }  
  
    // add first char  
    for(int i=0; i<sans.size(); i++){  
        String ssub = sans.get(i);  
        String sub = fc + ssub;  
        ans.add(sub);  
    }  
  
    return ans;  
}
```