

#

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 $2 \times 5 + 1 = 11$

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

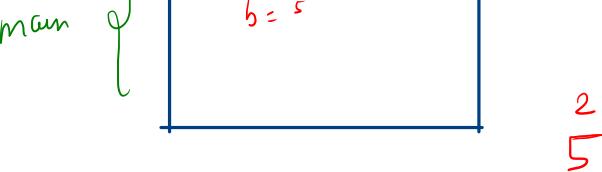
Stack

```
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);  
    }  
}
```



Reversion

How to solve any recursion problem

⇒ How to solve any recursion problem...
for a smaller problem (faith)

- 1) Keep a faith that it works.
 - 2) Solve for the ~~smallest~~ problem. (base case)
 - 3) Solve for the actual problem. (to reach your expectation)

Ques Print decreasing numbers from 'n' to 1 using recursion

$n=5$	public static void printDecreasing (int n){ System.out.print(n); printDecreasing (n-1); }
Expectation:	5 4 3 2 1
	loop

```

public static void printDecreasing (int n) {
    System.out.print (n);
    printDecreasing (n-1);
}

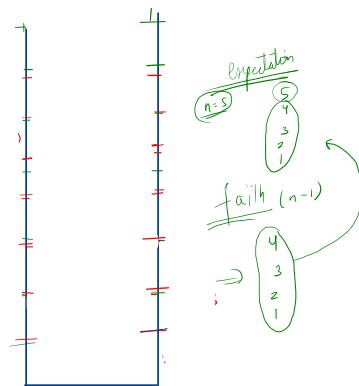
```

function with point numbers from $(n-1)$ to 1

Recursion (n)
 function will print numbers from
 n down to 1

you won't be able to learn recursion without dey-Hern

stack overflow



```
class Main {
    public static void printDecreasing(int n){
        if(n==0){①
            return;
        }
        System.out.println(n);
        printDecreasing(n-1);②
    }
}

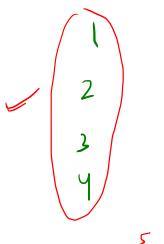
Run | Debug
public static void main(String[] args) {
    int n = ③; ④
    printDecreasing(n);
}
}
```

3
2

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.

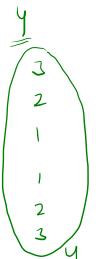
Faith $(n-1)$
It will print numbers
from $(n-1)$ to 1 and then 1 to $(n-1)$

$n=1$
3
2
1
1
2
3

$n=5$

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

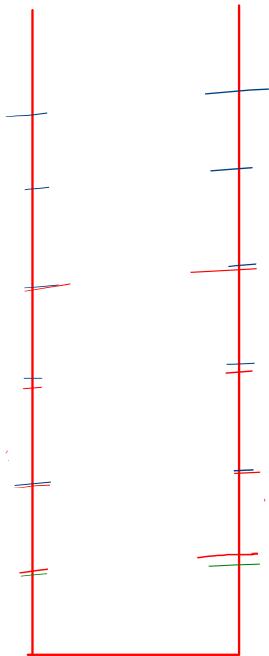
}



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

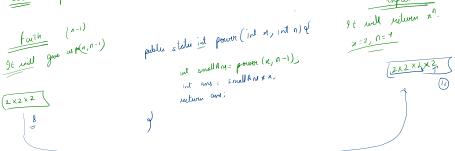
It will return factorial
of given number

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



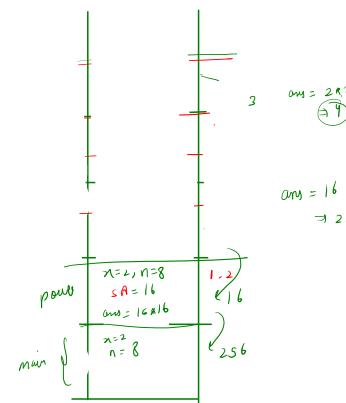
faith ($n/2$)
it will calculate
 $x^{\frac{n}{2}}$ correctly
 $x=2, n=6$

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 ($n/2 \cdot 2 = 0$)
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);  
}
```

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

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

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

$$\begin{aligned} n &= 2^x \\ 2^x &= n \\ \log_2 2^x &= \log_2 n \\ x \log_2 2 &= \log_2 n \\ x \cdot 1 &= \log_2 n \\ x &= \log_2 n \end{aligned}$$

$$\log_a b = b \log_a a$$

$$\log_a^* \Rightarrow$$

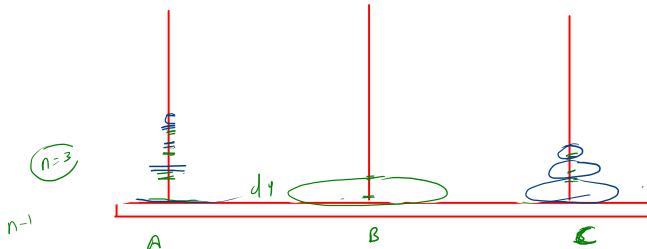
$$O(\log n)$$

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

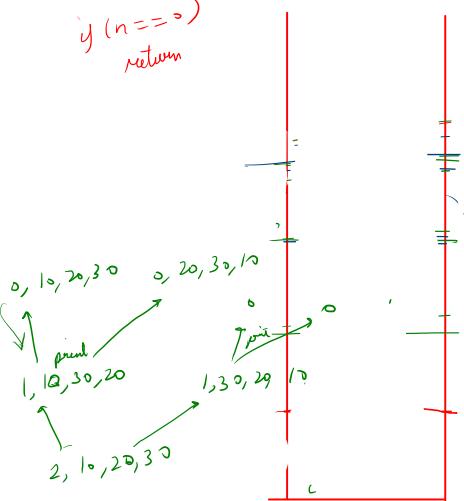
move $(n-1)$ discs from A to C using B
→ move nth disc from A to B
→ move nth disc from A to B (move 3rd disc)
move $(n-1)$ discs from C to B (move 2nd disc)
move $(n-1)$ discs from C to B (move 1st disc)
→ move $(n-1)$ discs from C to B (move 2nd disc)
→ move $(n-1)$ discs from C to B (move 3rd disc)

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

$y(n=2)$
return

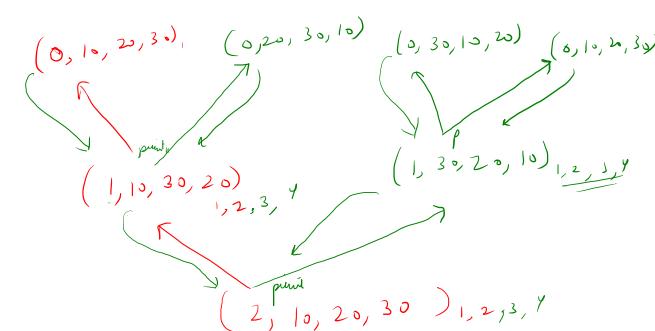


```
public static void towerOfHanoi(int n, int A, int B, int C){  
    if(n==0){  
        return;  
    }  
  
    towerOfHanoi(n-1, A, C, B);  
    // move nth disc from A to B  
    System.out.println("Moving ring " + n + " from " + A + " to " + B + " using " + C);  
    towerOfHanoi(n-1, C, B, A);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

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

Recursive tree diagram

→ color path

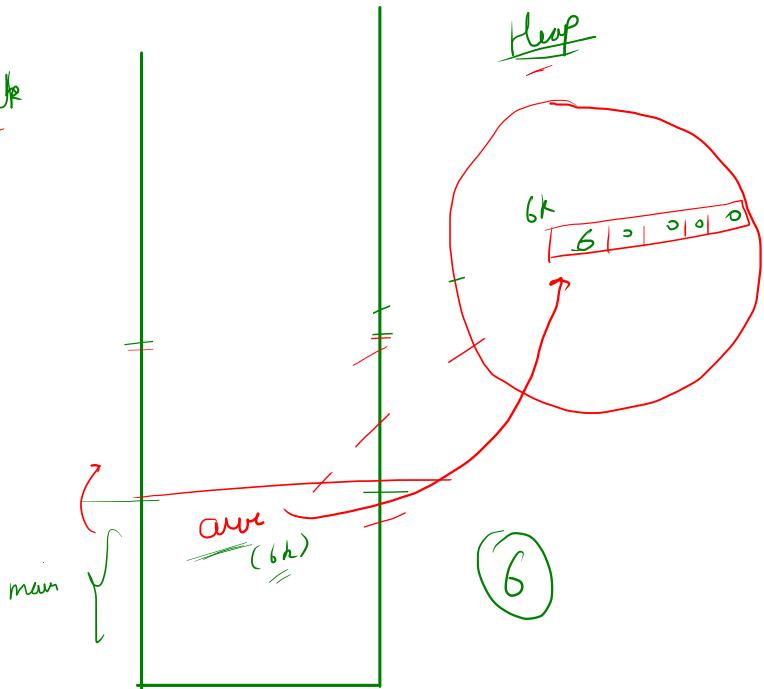


#

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;

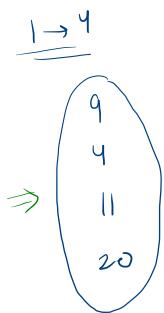
    int a = 5;
    increaseArray(arr);
    increaseInteger(a);

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

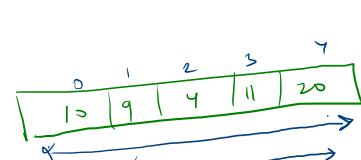
```

Ques Print array using recursion.

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

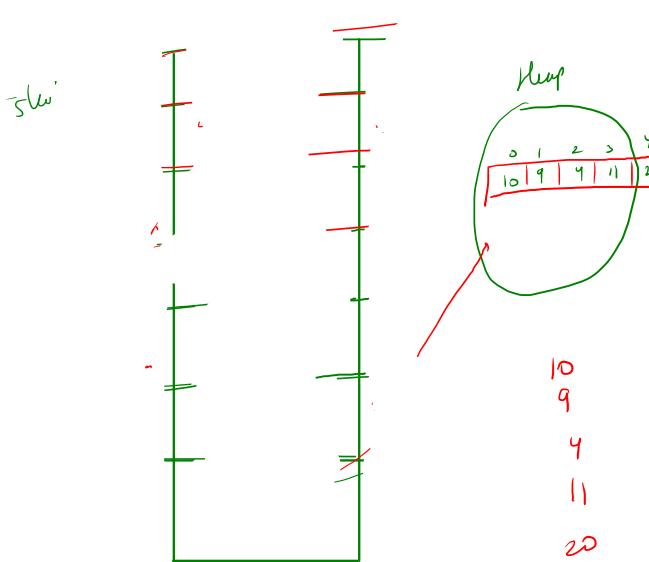
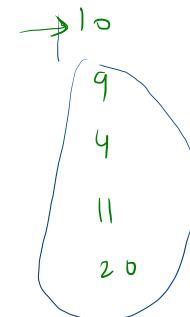


psu printArray(int[] arr, int idx){
 System.out.println(arr[idx]);
 printArray(arr, idx+1);
}



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

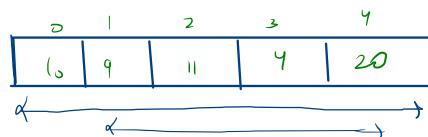
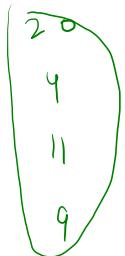
Fault

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

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

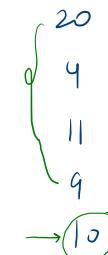
 printReverse (arr, idx+1);
 System.out.println (arr[idx]);

}



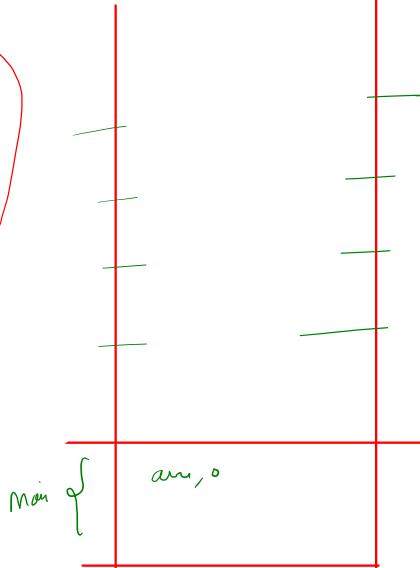
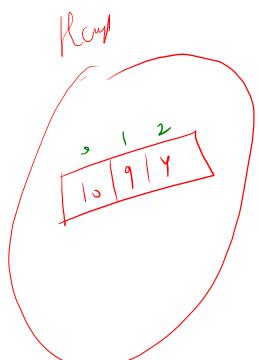
print reverse (arr, 0)

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);  
}
```



Y
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;

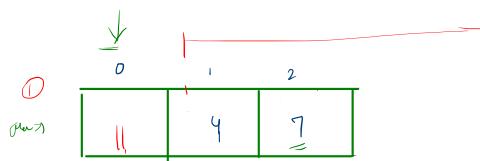
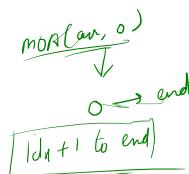
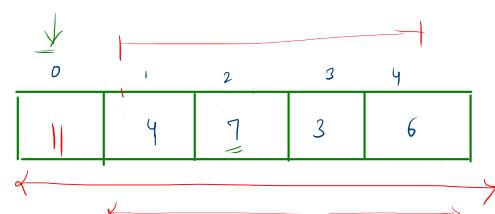
}

expectation

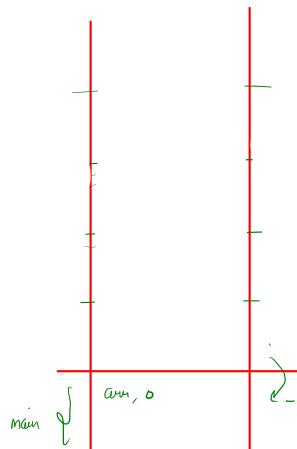
It will return 'max' from

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

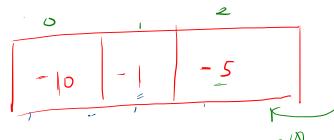
ans = 11



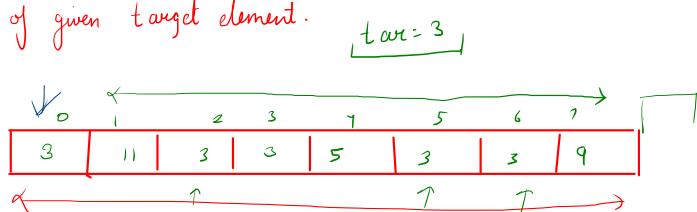
```
public static int maxOfArray(int[] arr, int idx){  
    if(idx == arr.length){  
        return 0; // Integer Min Value  
    }  
  
    int smallAns = maxOfArray(arr, idx+1);  
  
    int ans = Math.max(arr[idx], smallAns);  
    return ans;  
}  
Run | Debug  
public static void main(String[] args)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

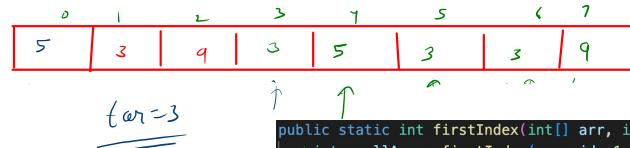
It will return first index of tar from 'idx+1' to end

```
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;
}
```

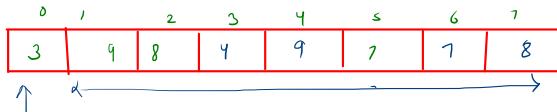
```
Run | Debug
public static void main(String[] args) {
    int[] arr = {10, 9, 4, 11, 20};
    System.out.println(maxOfArray(arr, idx: 0));
}
```

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

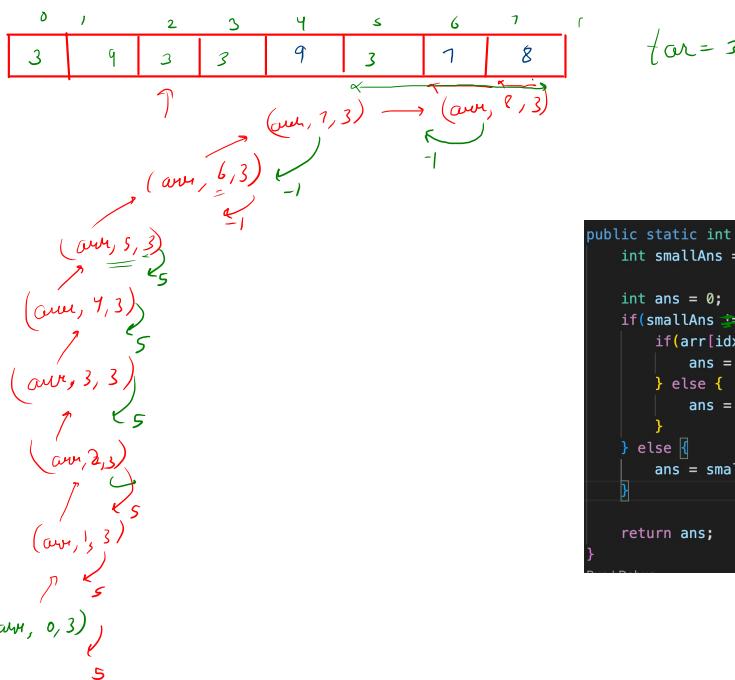
smallAns = -1

```
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);

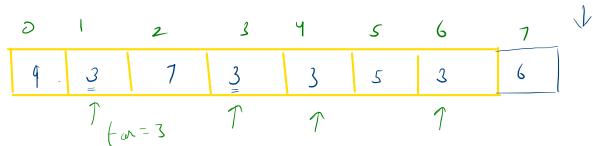
    int ans = 0;
    if(smallAns == -1){
        if(arr[idx]==tar){
            ans = idx;
        } else {
            ans = smallAns;
        }
    } else {
        ans = smallAns;
    }

    return ans;
}
```

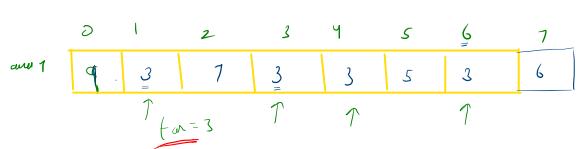
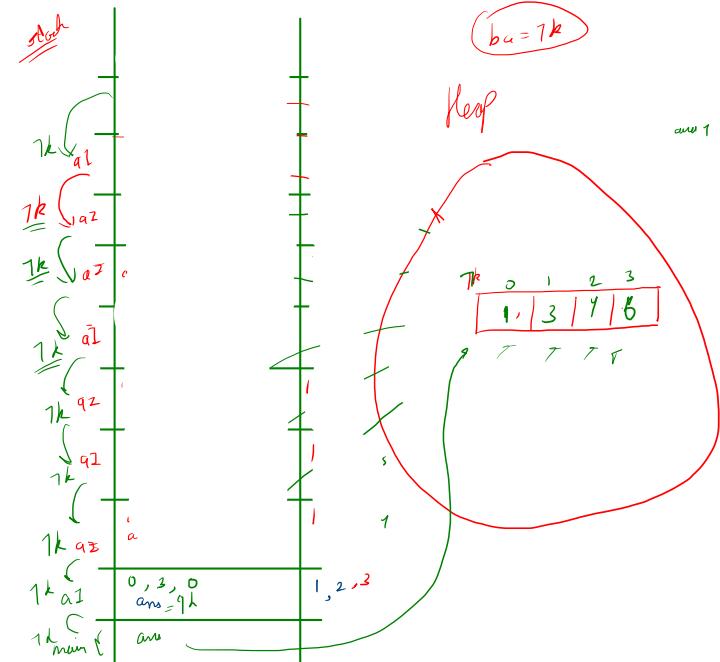
Ques Find all indices of tar in array

(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^{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

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

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

$$(n-1)$$

$$(n-2)$$

$$n=6$$

$$n=5$$

$$n=5$$

$$f(n-1)$$

$$f(n-2)$$

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

$$lt = 2$$

length calculation

It will calculate n^{th} fibonacci term.

$$n=6$$

$$ans = ?$$

return ans;

public static int fib (int n) {

$$\text{int } lt = \underline{fib(n-1)};$$

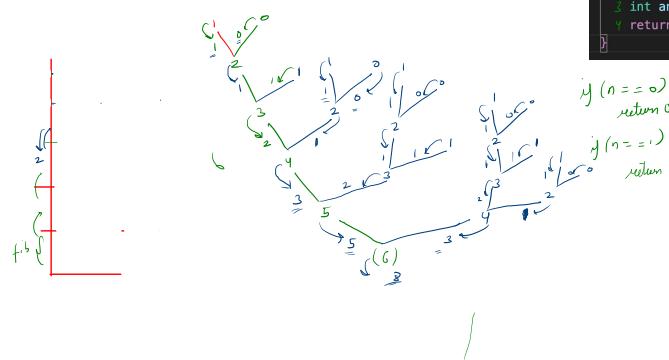
$$\text{int } lst = \underline{fib(n-2)};$$

$$\text{int } ans = lt + lst;$$

$$\text{return } ans;$$

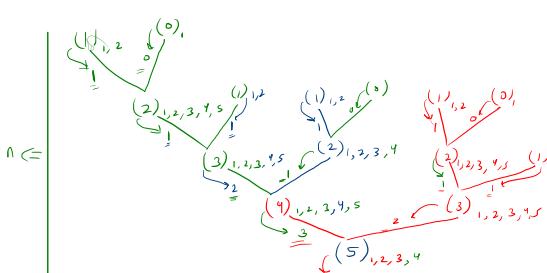
```
// fibonacci number
public static int fib(int n){
```

- | int lt = fib(n-1);
- | int lst = fib(n-2);
- | int ans = lt + lst;
- | return ans;



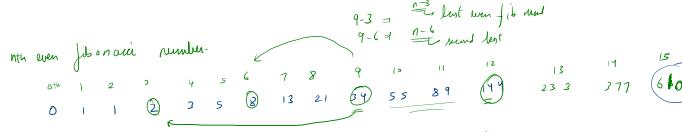
$T(n) \Rightarrow (\text{number of calls})^{\text{height of tree}}$

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

3rd even fib number = 9th fib number
 4th even fib number = $3 \times n$ fib number

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

$$f(n-2) + f(n-3) + f(n-4) + f(n-5)$$

$$f(n-3) + f(n-4) + f(n-5) + f(n-6) =$$

$$2f(n-3) + f(n-4) + f(n-5)$$

$$3f(n-3) + f(n-4) + f(n-5) + f(n-6)$$

$$\rightarrow 3f(n-3) + f(n-4) + f(n-5) + f(n-6)$$

even

$$f(n) = 3f(n-3) + f(n-6) \Rightarrow n^{\text{th}} \text{ even fib number}$$

$$\Rightarrow 4 \times (\text{last 3 even fib number}) + (\text{second last even fib number})$$

$f_E(n) = 4f_E(n-1) + f_E(n-2)$

last even fib number
 second last even fib number

$\boxed{1}$

$$1 \xrightarrow{a=8} 10 \xrightarrow{\text{length } 10 \text{ (mod)}}$$

$$a = 10 \xrightarrow{a=10} 10 \xrightarrow{10+10=20}$$

$$\boxed{2}$$

$$\text{ans} = \boxed{n+1, 50}$$

$$n+2 \xrightarrow{1} 1$$

$$n+9 \xrightarrow{0}$$

$$10 \xrightarrow{2}$$

$$27 \xrightarrow{27+2=0}$$

$$37 \xrightarrow{37+2=0}$$

last even $\geq 10^9$
 last $\geq 10^9$

$$\text{ans} \geq 4 \times 10^4 + 10^2$$

$$\geq \boxed{410^4}$$

$$7 \times 10^4$$

$$10 \xrightarrow{-2^{12}}$$

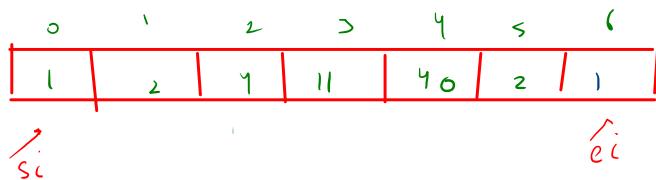
$$\begin{array}{c} \text{last even} \\ -2^{12} \\ \hline \boxed{2^{12}-1} \\ \hline 10^8 \end{array}$$

$$10^n \not\mid 1 \pmod{10^8}$$

(mod)

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

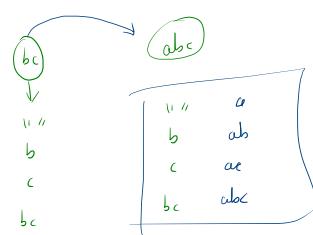
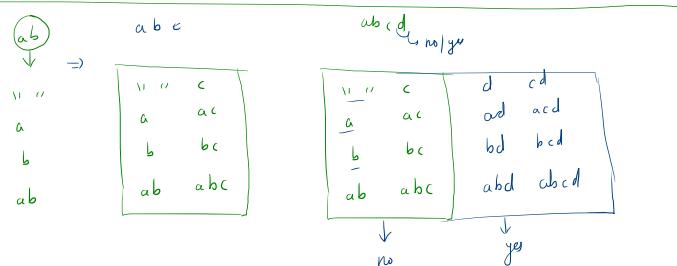
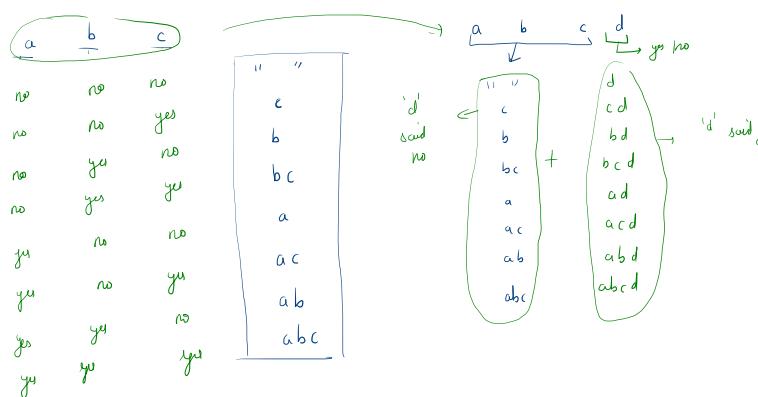
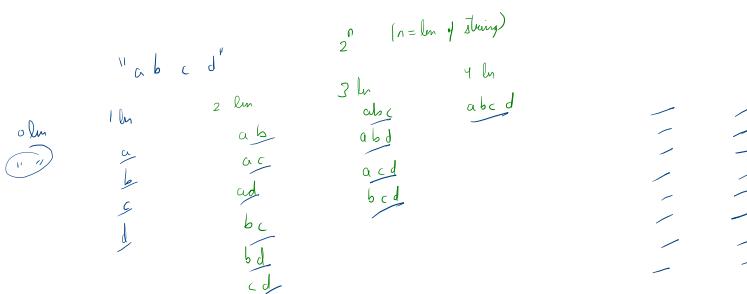
arr[1] = 45

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?

("")
↓
{ ("",) }

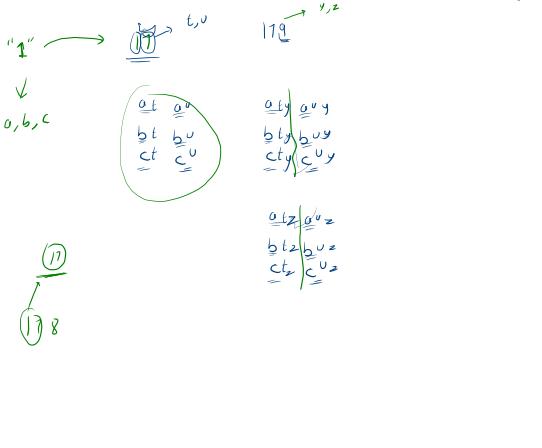
(c)
↓
{ (), (c) }

(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;  
}
```

0 -> . ;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz
(y,z)



$$\text{Schn} = \underline{\underline{189}}$$

public static ArrayList<String> getKpc(String site) {

expectation
will return all
possible strings

$\underline{t} \underline{v} \underline{y}$, $t \underline{v} z$, $t w y, t w z, t v y, t x z$
 $v v y, v v z, u w y, u w z, u v y, u x z$

"89" = "789"

(789)

$t_{VY}, t_{VZ}, t_{WY}, t_{WZ}, t_{XY}, t_{XZ}$
 $u_{VY}, u_{VZ}, t_{WY}, u_{WZ}, u_{XY}, u_{XZ}$

	0	1	2	3	4	5
sum =	uy	vz	wy	wz	yz	xz

(111)
 \downarrow
 / (111)
 ↓ (111)
 (9)
 \downarrow
 ↓ (9, z)
 (89)
 \downarrow
 (789)
 \downarrow
 (789)

sum = $"y z"$
 add = $"y z"$
 $b = ab$, $i < z$
 to Add = $"y^i z"$
 $j=0 : j < 1$
 $y + z = y^0 + z^0$
 y z

```

// get all keypad combinations
static String[] sarr = {"", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tu", "vwx", "yz"};
public static ArrayList<String> getKPC(String str){
    if(str.length()==0){
        ArrayList<String> ba = new ArrayList<String>();
        ba.add("");
        return ba;
    }

    int firstDigit = str.charAt(index) - '0';
    String ss = str.substring(beginIndex+1);

    ArrayList<String> sans = getKPC(ss);

    ArrayList<String> ans = new ArrayList<String>();
    String add = sarr[firstDigit]; // string that we can replace this digit with

    for(int i=0; i<add.length(); i++){
        char toAdd = add.charAt(i);
        for(int j=0; j<sans.size(); j++){
            String sansString = sans.get(j);

            String ansString = toAdd + sansString;
            ans.add(ansString);
        }
    }

    return ans;
}

```

The diagram shows a sequence of four tables representing the states of climbing n stairs:

- Table 1 (Left):** Shows all possible ways to climb 3 stairs. The paths are categorized by the last step taken:
 - 1 step: 1, 1, 1
 - 2 steps: 1, 1, 2; 1, 2, 1; 2, 1, 1
 - 3 steps: 1, 2, 2; 2, 1, 2; 2, 2, 1; 3, 1
- Table 2 (Middle Left):** Shows all possible ways to climb 4 stairs, derived from Table 1 by adding one more step to each path in Table 1.
- Table 3 (Middle Right):** Shows all possible ways to climb 3 stairs, derived from Table 1 by removing the last step from each path in Table 1.
- Table 4 (Right):** Shows all possible ways to climb 2 stairs, derived from Table 3 by removing the last step from each path in Table 3.

Annotations explain the process:

- Top left:** "Get all the stair paths" and "put to den".
- Top right:** "n = 4" and "add this path to set to nth step".
- Middle right:** "in one move" followed by "1 step", "2 step", and "3 step".
- Bottom right:** "3 steps" and "put to climb 2 step".
- Bottom left:** "3, 1" and "3, 2" circled.

Faith
It will find all
the possible path

$$\text{for } (n-1)(n-2)(n-3)$$

$$\begin{aligned} n=3 \\ \text{Answer} &= \{ "111", "12", "21", "3" \} \\ n=2 \\ \text{Answer} &= \{ "11", "2" \} \\ n=1 \\ \text{Answer} &= \{ "1" \} \end{aligned}$$

public static ArrayList<String> getStairPath (int n) {

longest distance

13

```

public static ArrayList<String> getStairPaths(int n){

    if(n==0) // no possible path
        return new ArrayList<String>();
    else
        return ba;
}

if(n==0){
    ArrayList<String> ba = new ArrayList<String>();
    ba.add("");
    return ba;
}

ArrayList<String> ans = new ArrayList<String>();
ArrayList<String> sans = getStairPaths(n-1);

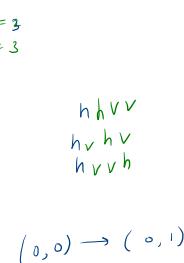
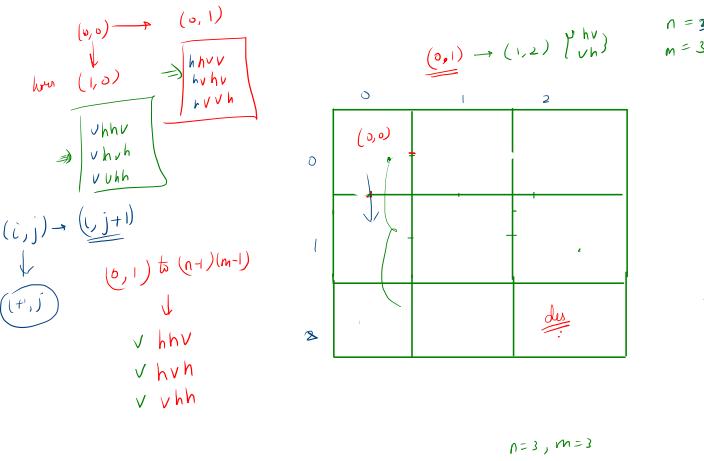
for(int i=0; i<sans.size(); i++){ // take a jump of 1
    String spath = sans.get(i);
    String path = "1" + spath;
    ans.add(path);
}

sans = getStairPaths(n-2); // take a jump of 2
for(int i=0; i<sans.size(); i++){ // take a jump of 2
    String spath = sans.get(i);
    String path = "2" + spath;
    ans.add(path);
}

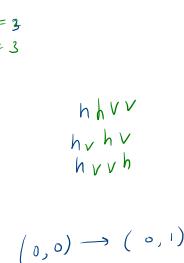
sans = getStairPaths(n-3); // take a jump of 3
for(int i=0; i<sans.size(); i++){ // take a jump of 3
    String spath = sans.get(i);
    String path = "3" + spath;
    ans.add(path);
}

return ans;
}

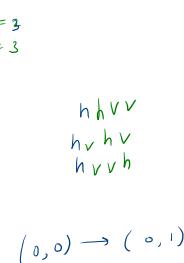
```



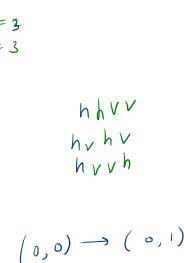
$n=3$



$m=3$



$n=3$

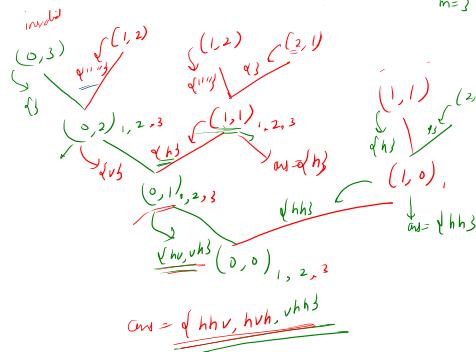
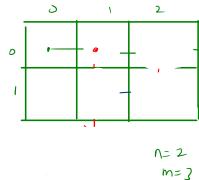


$m=3$

Fact
we will get all the paths
from $(0,1)$ to $(n-1, m-1)$
and $(1,0)$ to $(n-1, m-1)$

$\begin{array}{c} +h \\ +v \end{array}$

expectation
We will get all the
paths from (i,j)
to $(n-1, m-1)$

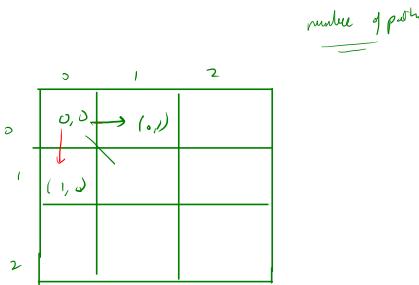


```
public static ArrayList<String> getMazePaths(int i, int j, int n, int m){  
    1 ArrayList<String> allPaths = new ArrayList<>();  
  
    2 ArrayList<String> hpaths = getMazePaths(i, j + 1, n, m);  
    for(int k=0; k < hpaths.size(); k++){  
        String hpath = hpaths.get(k);  
        String path = "h" + hpath;  
        allPaths.add(path);  
    }  
  
    3 ArrayList<String> vpaths = getMazePaths(i + 1, j, n, m);  
    for(int k=0; k < vpaths.size(); k++){  
        String vpath = vpaths.get(k);  
        String path = "v" + vpath;  
        allPaths.add(path);  
    }  
    return allPaths;  
}
```

(n, m)
 2^{n+m}

Ques Find all the different paths from $(0,0)$ to $(n-1, m-1)$.

$(0,0) \rightarrow (0,1)$
h h v v
h v h v
h v v h



$(0,0) \rightarrow (1,0)$
v h h v
v v h h
v h v h

path \Rightarrow hpaths + vpaths

Ans

It will return
correct number of paths
from $(i+1, j)$ to $(n-1, m-1)$
and $(i, j+1)$ to $(n-1, m-1)$.

ps int manPaths (int i, int j, int n, int m) ✓

int hpaths = mp(i, j+1, n, m);
int vpaths = mp(i+1, j, n, m);
int ans = hp + vp;
return ans;

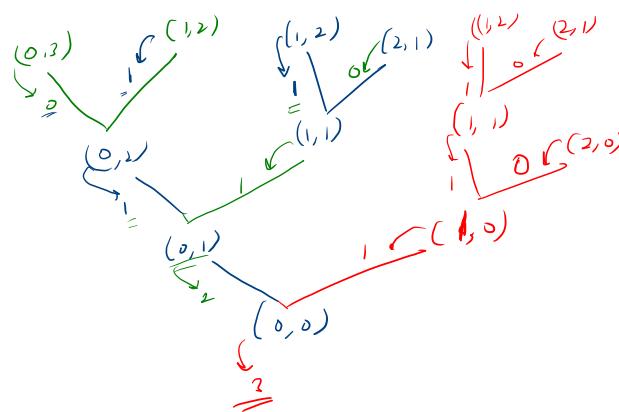
)

Explanation

it will return number of
paths from (i, j) to $(n-1, m-1)$.

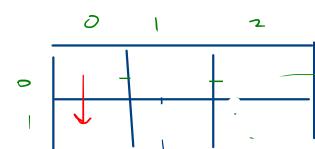
```
// number of maze Paths =====
public static int mazePaths(int i, int j, int n, int m){
    1 int hpaths = mazePaths(i, j+1, n, m);
    2 int vpaths = mazePaths(i+1, j, n, m);

    3 int allPaths = hpaths + vpaths;
    4 return allPaths;
}
```

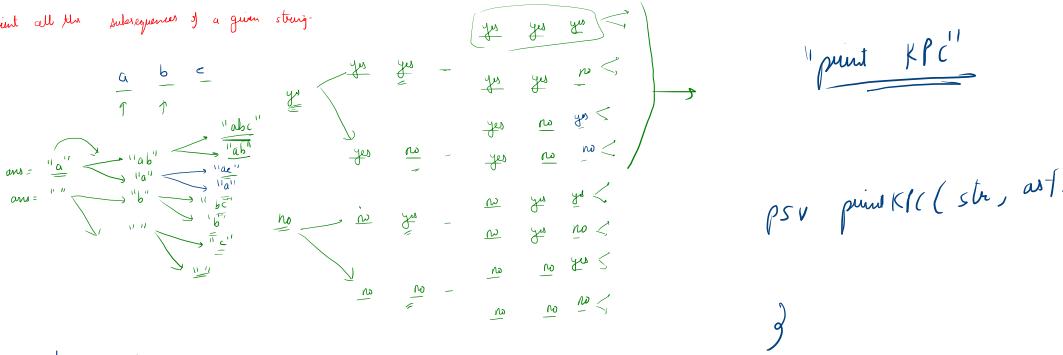


$j = m$

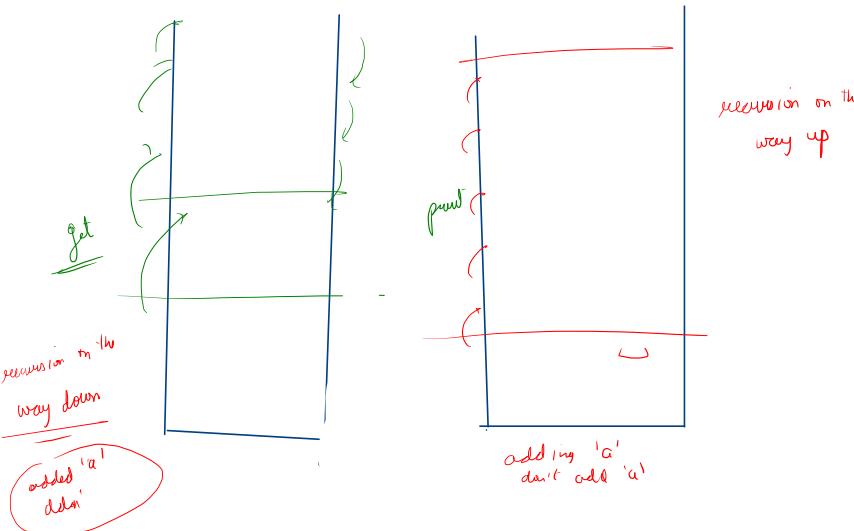
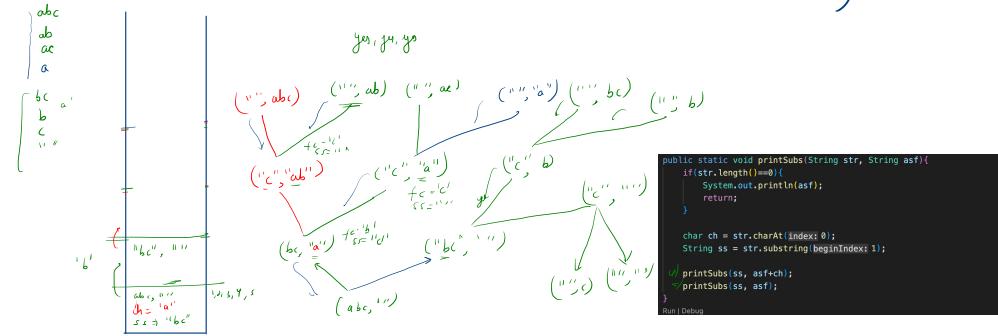
$n = 2$
 $m = 3$



Ques Print all the subsequences of a given string.



psv printKPC(str, asf)



QuesPrint all KPC

return

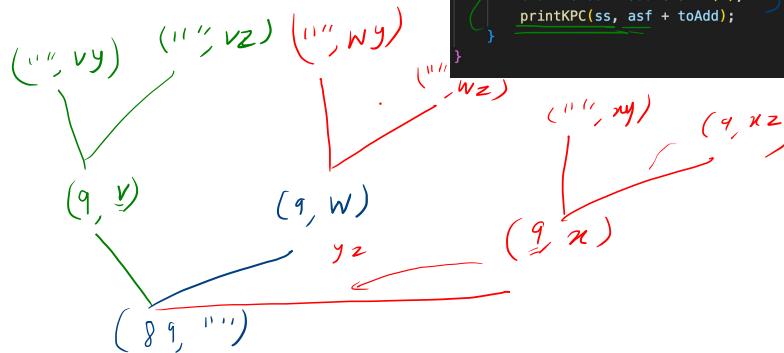
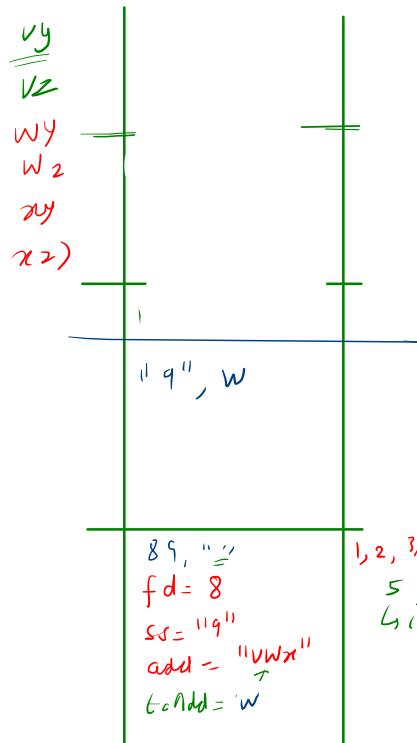
7
"789" → {t, u}
fc = '7'
ss → "89"

(89)
6 vy u vy
t vz u vz
6 wy u wy
6 wz u wz
t xy u xy
t zz u zz



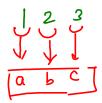
6 1 2 3 4 5 6 7 8 9

```
static String[] sarr = {".","abc","def","ghi","jkl","mno","pqrs","tu","vwx","yz"};
```



0 1 2 3 4 5 6 7 8

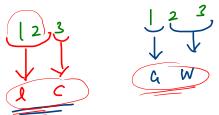
```
static String[] sarr = {".","abc","def","ghi","jkl","mno","pqrs","tu","vwx","yz"};  
public static void printKPC(String str, String asf){  
    if(str.length()==0){  
        System.out.println(asf);  
        return;  
    }  
    int fd = str.charAt(index) - '0';  
    String ss = str.substring(beginIndex: 1);  
  
    String add = sarr[fd];  
  
    for(int i=0; i<add.length(); i++){  
        char toAdd = add.charAt(i);  
        printKPC(ss, asf + toAdd);  
    }  
}
```



1 → 'a'
2 → 'b'
3 → 'c'
4 → 'd'

1 2 3 4

(34) > 26



11 → 'l'
12 → 'l'

fc = '0'
return;

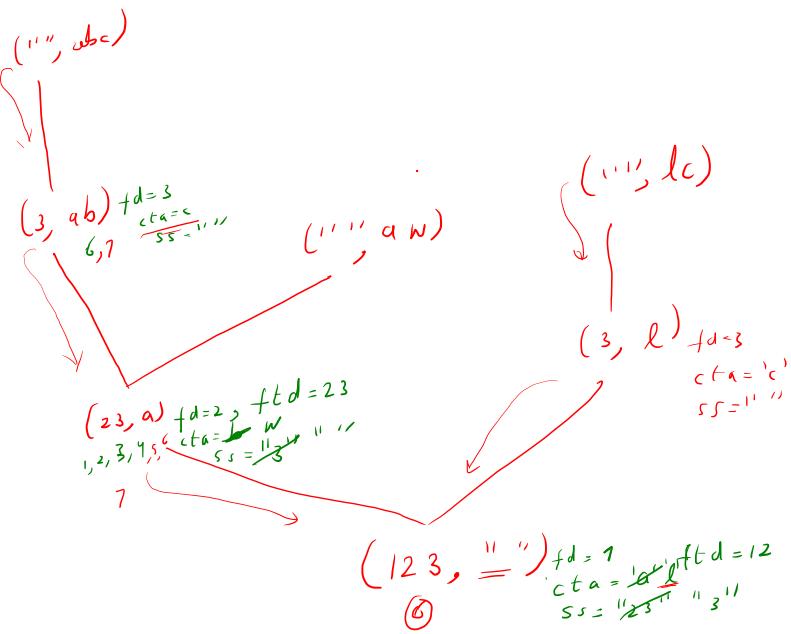
w
w
w
y
26 → 'z'

(2, 3, 4, a)
(3, 4, l)
(1, 2, 3, 4)

(3, 4)
Joint 2 digits <= 26

(123)
(1034)
(303)

abc
aw
lc



```
public static void printEncodings(String str, String asf) {
    1 if(str.length()==0){
        System.out.println(asf);
        return;
    }
    2 if(str.charAt(index)==0){
        return;
    }

    // taking first character
    3 int firstDigit = str.charAt(index) - '0';
    4 char charToAdd = (char)(a' + firstDigit - 1);
    5 String ss = str.substring(beginIndex: 1);

    6 printEncodings(ss, asf+charToAdd);

    // taking first 2 char
    7 if(str.length() >= 2){
        8 int firstTwoDigits = Integer.parseInt(str.substring(beginIndex: 0,endIndex: 2));
        9 if(firstTwoDigits<=26){
            10 charToAdd = (char)(a' + firstTwoDigits - 1);

            11 ss = str.substring(beginIndex: 2);
            12 printEncodings(ss, asf+charToAdd);
        }
    }
}
```

