

★★★★★

Recursion

- V.V.I ↗ while the func. is not finished executing it will remain in stack.
- ↗ when a func. finishes executing it is removed from stack and the flow of program is restored to where the fn. was called.

Stack 2

O/P
print5(5)
print4(4)
print3(3)
print2(2)
print1(1)
main

jaise sab ek-ex kar ke
store ek hu hai

i.e. main → print1() → ...

↗ usi hisab se execute hone ki baad nikalta jayega

→ Ultra. i.e. print5(5) → print4(4) → ... → main().

concept of
recursion in
stack

↑ main & print1() fir call kya & print1 ki ander ex aur call
kya tha jo ab wo execute na ha tab tak print1() mhi
niklega sly uske baad main niklega (last) mein,

↗ (main is the last fn. i.e. removed from
stack).

- ✓ Recursion → when a function calls itself.

Stack memory in case of recursion →

"print(n+1);

O/P
print(6)
print(5)
print(4)
print(3)
print(2)
print(1)
main()

∴ we need a base condition

base condition in recursion : condition where our recursion will stop making new calls. (Simple if condition).

if (n == 5){

 cout << 5;

} return; → mlb dont call now

Now will empty as :-

print(5) → print(4) → direct call before main

↓
main()

✓ if you are calling again & again, you can treat it as a separate call in the stack.

✓ No base condⁿ → func. call will be keep happening.

→ stack will be filled again and again.

(every call of func. takes memory)

→ after some time, memory of computer will exceed to certain limit.

i.e. Stack overflow error.

when there is no base condition

Q. Why recursion?

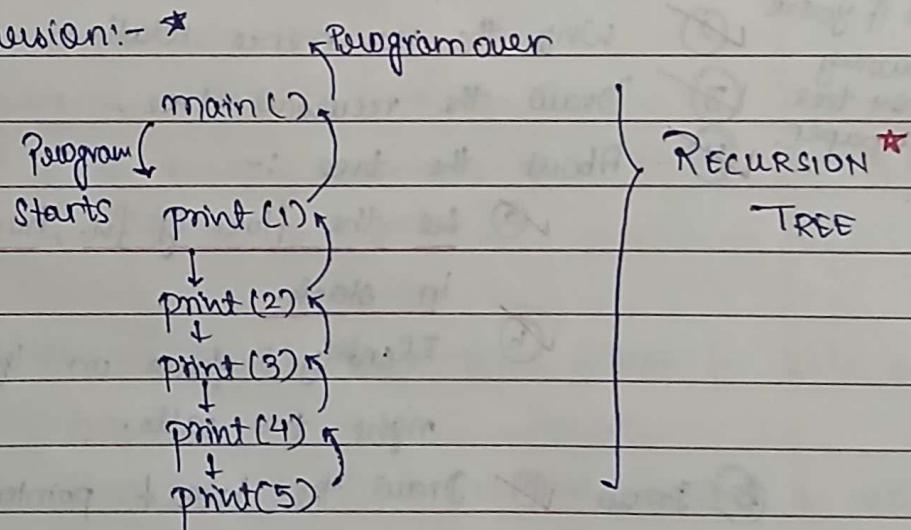
→ It helps us in solving bigger/complex problems in a simple way.

→ You can convert any recursion solⁿ into iteration & vice-versa.

→ Space complexity is not constant because of recursive calls.

→ It helps us in breaking down bigger problems into smaller problems.

✓ Visualising Recursion:- *

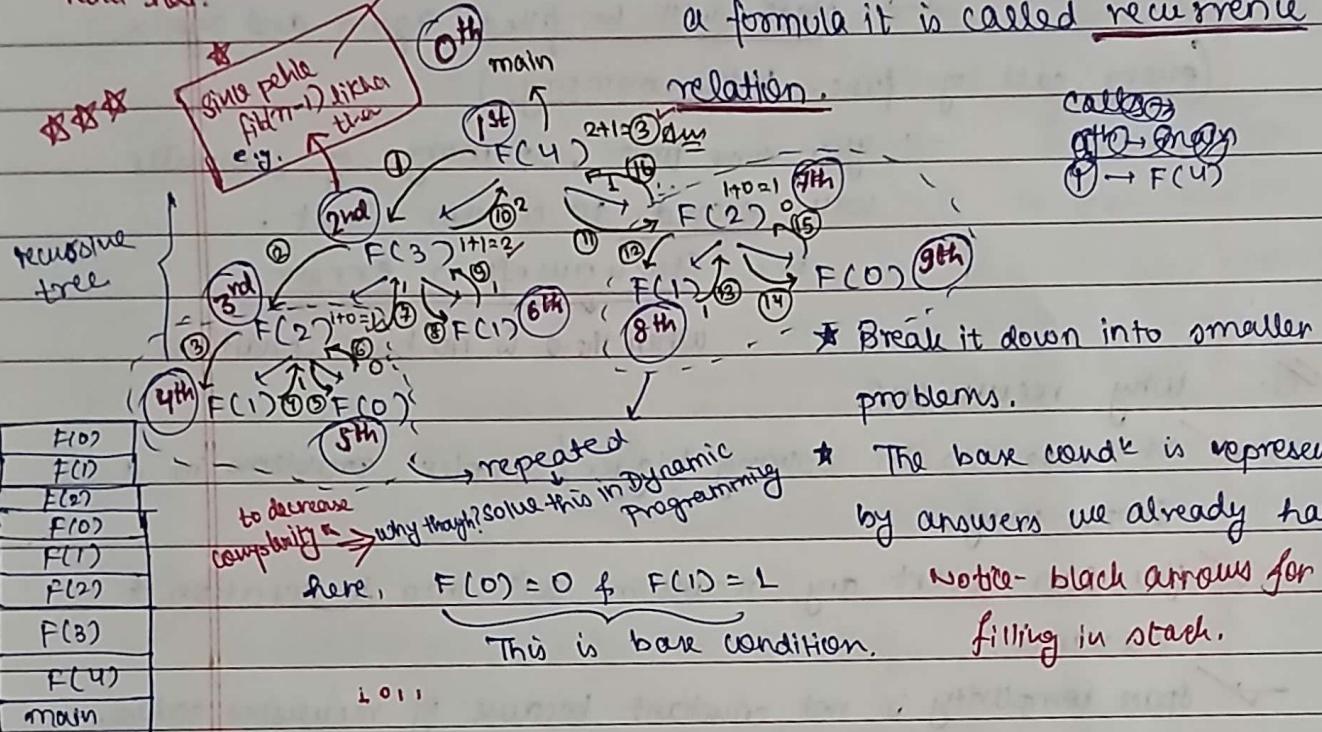


General concept for Fibonacci no.

is
basic
maths
koi sochne se jarurat
nhi hai → seedha dikh
raha hai.

$$\text{Fibo}(N) = \text{Fibo}(N-1) + \text{Fibo}(N-2)$$

when you write a recursion in
a formula it is called recurrence



VVI. How to understand and approach a problem?

No one can
explain recursion
to you if you're
not drawing
recursion tree
in pen + paper.

① Identify if you can break down problem into
smaller problems

② Write the recurrence relation if needed.

③ Draw the recursive tree.

④ About the tree :-

⑤ See the flow of fn., how they are getting
in stack.

⑥ Identify & focus on left tree calls &
right tree calls.

⑦ Draw the tree & pointer again & again
using pen + paper.

⑧ Use a debugger to see the flow.

⑨ See how values are returned at each step
& what type of values

See where the fn. call will come out of.

In the end you it will come out of the main fn.

Tip: Do not overthink.

VVVVVI

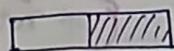
Variables: ① Arguments

② Return types

③ Body of func.

Understand the code

Q. Binary search with recursion.



N

$$C_{N/2} + 2 = \frac{2+2}{2}$$

- 1. ① comparison in

constant time i.e. $O(1)$

Search in the array of $N/2$.

- ② Dividing into 2 halfs

- 2. recurrence reln \rightarrow

$$F(N) = O(1) + F(N/2)$$

comparison

i.e. to find binary search in array of size 'n' \rightarrow take do a step that takes constant amount of time + search in $1/2$ of the array.

- Types of recurrence relation:-

Linear recurrence relation \rightarrow e.g. fibo

Divide & conquer recurrence reln \rightarrow e.g. Binary Search (reduced by a factor)

Whatever variables you need to pass in future fn. calls put inside the argument.

And variables that are considered valuable in that fn. call only

that you don't need to pass into future recursion calls put inside body of function.

Here middle 'm' is in body & start 's' & end 'e' in argument.

for future rec. calls

`return 0;` \Rightarrow signifies success or that the operation completed successfully without errors.

e.g. file closed successfully

Note: this & `return -1;` will return 0 & -1 only respectively but it indicates their meaning as value 0 & -1 \Rightarrow convention in C/C++/Java.

- ✓ Make sure to return the result of a fn. call of the return type.

arr: [1, 2, 3, 4, 55, 66, 78] target: 78 m: 3

Left takes ($s \rightarrow m$)

Right takes ($m+1 \rightarrow e$)

thought
only
but checks
value!!

code off

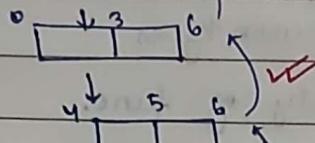
debug mode

same file

else say

nhi aayega.

main() 6 Am



way of showing error
in form of integer.

* `return -1;` \Rightarrow indication that

a certain condition has not been met;

$$\frac{s+e}{2} = s + \frac{(e-s)}{2}$$

e.g. error opening file e.g. invalid input data
here \rightarrow e.g. element not found

Time and Space Complexity

- ✓ What is time complexity?

reference ex. \rightarrow

Old Computer

MI MacBook

✓ data: 1,000,000 elements in array

"

✓ Algorithm, linear search for target
that doesn't exist in array

"

✓ Time taken \rightarrow 10 sec

Time: 1 sec.

Both have same

Time Complexity i.e. $O(n)$

because ∇ Time Complexity \neq Time Taken

↓

It's basically get a function that gives us the relationship about how the time will grow as input grows

$O(N^2)$ upper bound
not meant to be tightest upper bound
ex: $N^3, N^2, N^{\log N}$

Worst case complexity: $O(N^2)$

Best " " : $O(N)$

Use case: Adaptive i.e. steps get reduced if array is sorted.

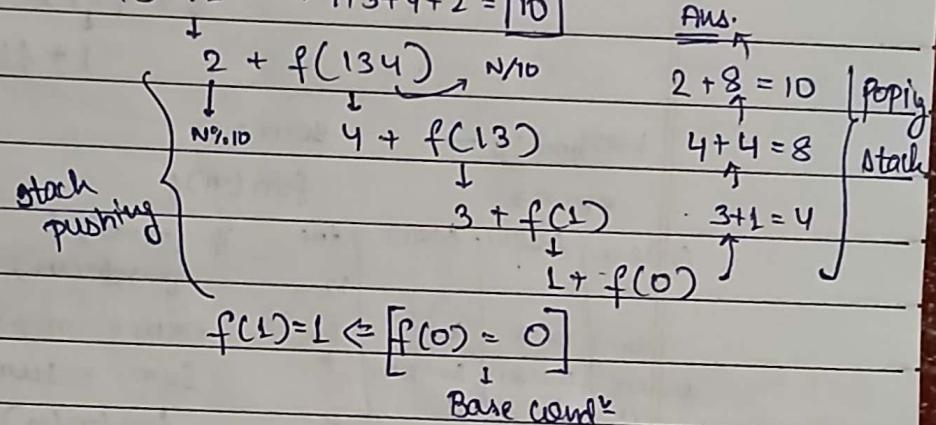
No. of swaps reduce as comparison to Bubble sort

Stable Algo

Used for smaller values of $N \Rightarrow$ works good when array is partially sorted + it takes part in hybrid sorting algorithms (e.g. quick sort, merge sort, heapsort then insertion)

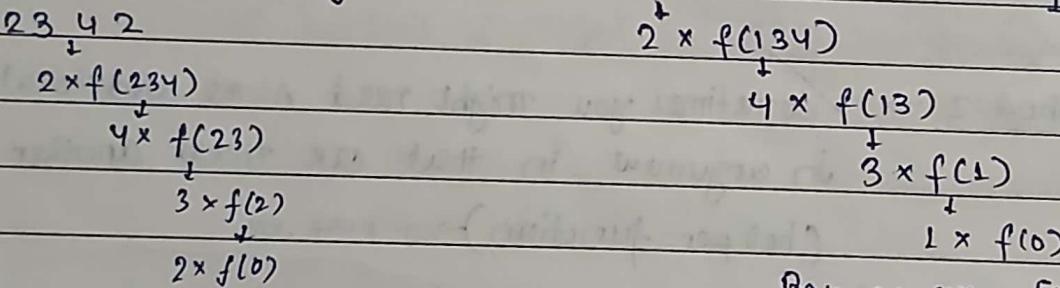
* Revision Lvl-1 Ques.

Sum Dgit \rightarrow e.g. $n = 1342 \rightarrow 1+3+4+2 = 10$



Similarly Prod Digit.

$$n = 1342 \rightarrow 1 \times 3 \times 4 \times 2 = 24$$



Base condition $[f(0) = 1]$

Alternate base condition $\rightarrow (n \% 10 == n)$

return $n;$

e.g. $n = 505$

$$5 \times f(50)$$

$$\leftarrow 0 \times f(5)$$

Concept:-

If in recursion call $m--$;

then means pass m then sub. 1

∴ it will call m again & again & again...

∴ use $(-n)$.

reverse a no. :-

$$\text{eg. } m = 1892 \Rightarrow 4281$$

$$\begin{array}{c} 4 + f(182) \\ \downarrow \\ 2 + f(18) \\ \downarrow \\ 8 + f(1) \\ \downarrow \\ 1 + f(0) \end{array}$$

Method:1 → $\text{sum} = 0$

eg. 123 $\text{fun}(m) \{$

(123)	false	false(12)	false(1)	true	if ($m=0$) {
rem=2	rem=2	rem=1	rem	} return;	
sum=0+3	sum=3+2	sum=32+1	sum	mrem = m % 10;	
12	1	0	32	sum = sum * 10 + rem	

} $\text{fun}(m/10);$

Method:2 → Sometimes you might need some additional variables in argument in that we make another function.
(helper function) → e.g. swap, etc.

$$n = 1234$$

$$4 + 123$$

$$4 \times 1000 + 123$$

$$[4 \times 10^3 + f(123)]$$

$$\hookrightarrow [3 \times 10^2 + f(12)]$$

$$\hookrightarrow [2 \times 10 + f(1)]$$

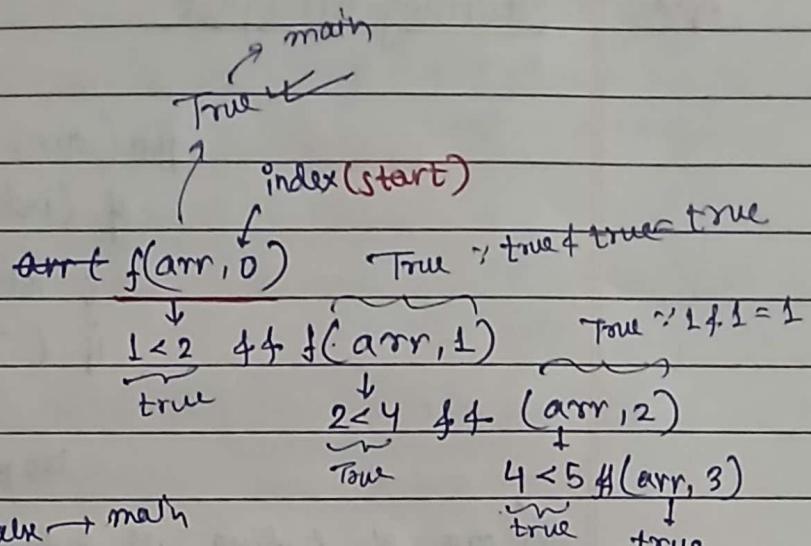
$$f(m, \text{arg}) = \text{rem} * 10^{\text{arg}-1} + f(m/10, \text{arg}-1)$$

base condition → if ($n \% 10 == n$)
return n ;

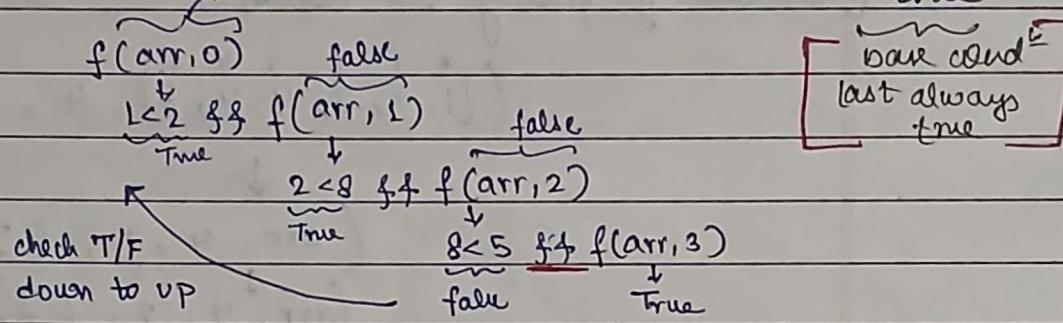
Recursion - Array Ques.

Q. Check array is sorted?

$$\rightarrow \text{arr} = [1, 2, 4, 5]$$



e.g. arr [1, 2, 8, 5] , false → math

Linear search with recursion

$$\text{arr} = [3, 2, 1, 18, 9] , \text{target} = 18$$

$f(\text{arr}, \text{target}, 0)$, index

aur sinhe agar ek bhi jaga target mil gya to True!

✓ $\text{arr}[0] == \text{target}$ || $(\text{arr}, \text{target}, 1)$

$\text{arr}[1] == \text{target}$ || $(\text{arr}, \text{target}, 2)$

false → till index becomes out of bound

* Returning ArrayList

✓ fun(arr, target, index, list)

if (index == end) {

✓ return list ;

✓ if (target == arr[index]) {

✓ add in list

keep going

✓ in case of finding all indexes.

main

([1, 2, 3, 4, 4, 8], 4, 0, [])

concept

(arr, 4, 1, [])

(arr, 4, 2, [])

∴ list.add Yes! ← (arr, 4, 3, [])

return

(arr, 4, 6, [3, 4])

adding on the same
object just via diff.
reference variable

e.g.

1. PASSING IT IN
ARGUMENT

implementation

static ArrayList< Integer > findAllIndex(int[] arr, int target,
[return type])

int index, ArrayList< Integer >
list) {

if (index == arr.length) {
// last take check hoga
// agar nahi milti to wala
// aage wala & return
return list ;
pahuch kar

if (arr[index] == target) {

list.add(index);

return findAllIndex(arr, target, index + 1, list);

type ArrayList

returned

easy

Rotated Binary Search

watch B.S. video if you

$$\text{arr} = [5, 6, 7, 8, 9, 1, 2, 3]$$

don't know this is rotated array.

① if $\text{arr}[s] \leq \text{arr}[\text{mid}]$ e.g. $\frac{7}{2} = 3.5$

$$\text{mid} = \frac{7}{2} = 3.5$$

if $\text{key} \geq \text{arr}[s] \& \leq \text{arr}[\text{mid}]$

floorvalue = 3 i.e. '8'

$$\text{end} = \text{mid} - 1$$

else

$$s = m + 1$$

② if $\text{key} \geq \text{arr}[\text{mid}] \& \leq \text{arr}[e]$

e.g. 2

$$s = m + 1$$

else

$$e = m - 1$$

* Pattern + Bubble + Selection

Q. 4 ⚫ ⚫ ⚫ ⚫

$f(r, c)$ column

3 ⚫ ⚫ ⚫

$f(4, 0)$

2 ⚫ ⚫

proved if $0 < 4$

1 ⚫

$f(4, 1)$

South:

$f(4, 2)$

$f(4, 3)$

$row - 1$

$(3, 0)$

new line, again start with 0.

illy for correct triangle check code.

Q. What does `Arrays.copyOfRange()` do?

→ Sol 1 ctrl + click to check

Sol 2 Copies the specified range into a new array.

→ Check code! + & mostly check for merge in place code!

★ 2. (Continuing return ArrayList) Putting it in body (not Argument)

Very Imp.

Goal: return the list but don't take in argument. reason

→ ∵ Problem: Every call will have new ArrayList.

main → [3, 4]

e.g. arr[1, 2, 3, 4, 4, 8]

, target = 4, index = 0

① (arr, target, 0)

[list = []] → body of function ✓

② (arr, t, 1)

[list = []] → empty since, ptho 4 → 3rd index se aayeg.

③ (arr, t, 2)

[list = []] → new lists are created at every function call.

④ (arr, t, 3)

[list = [3]] → 3 added in the list will be there for only this function; no dependency with past & future.

⑤ (arr, t, 4)

[list = [4]]

return type list: []

(arr, t, 5) → So the answer is present in individual function call: Problem

How to connect them? see red pen answers. base condition if (index == curr.length){

 return list;

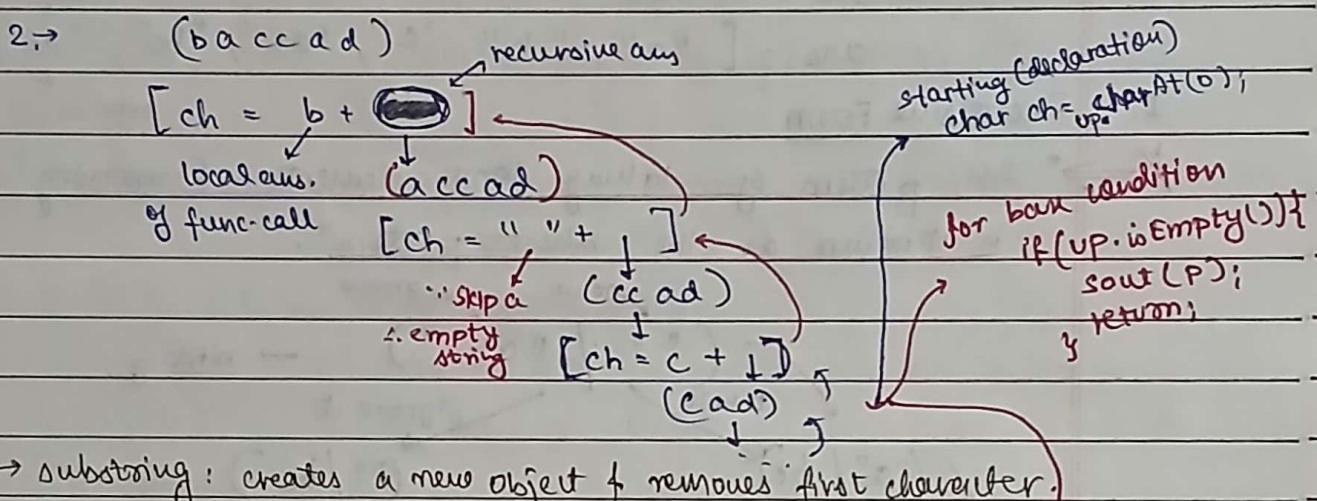
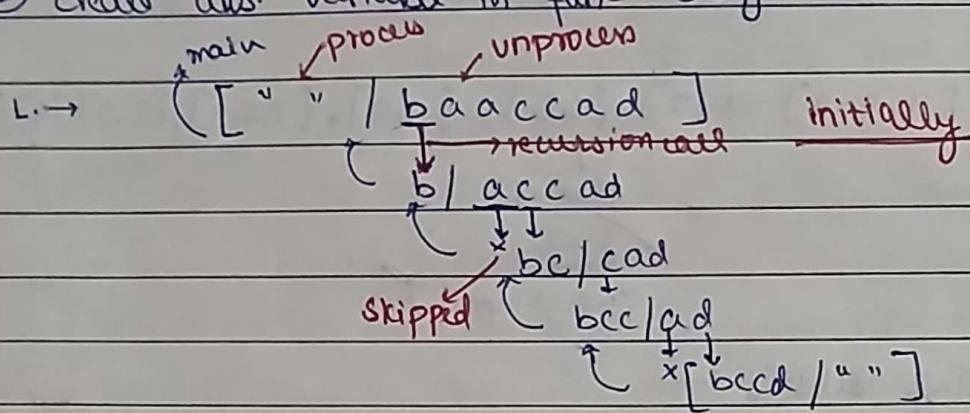
CHECK CODE!

~~V.~~ Remove 'a' from a String.

eg. str = baccad ans = bccd .

→ ① Pass the ans. string in argument → passed for future calls .

② Create ans. variable in func. body → new



→ substring : creates a new object & removes first character.

↳ same as slice method in JS.

e.g. String s = "Hello World";
sout(s.substring(6)); // World
sout(s.substring(0,5)); // Hello

for 1st element always inclusive for it till end.

for skip :- if (ch == 'a') {
 skip(p, up.substring(1));
 will remain as it is
}

[inclusive] exclusive

Method 2 using return type :- (take only up as arg)

base condition → if (ch == 'a') { if (up.isEmpty()) {

return "";

if a → if (ch == 'a') {

return skipRet(up.substring(1));

else {

return ch + skipRet(up.substring(1));

for not 'a' :-

else {

skip(p+ch, up.substring(1))

now consider next index i.e. 1
to check (skip 0th index later
a is)

added in process

Note :- To skip a word directly → e.g. Apple

↳ If (up.startsWith("apple")) {

return skipApple(up.substring(5));

else {
 return skipApple(up.substring(1));
 up.charAt(0);
 skipApple(up.substring(1));

for Arrays for strings

* Subsets / Subsequence

- ✓ deals with permutations & Combinations
- ✓ Subsets → Non-adjacent collection.

e.g. $[3, 5, 9] \rightarrow [3], [3, 5], [3, 9], [3, 5, 9], [5, 9], [5], [9]$

\downarrow
same as
 $[5, 3]$
same
(so don't repeat)

- ✓ Creating subsets :-

str = "abc"

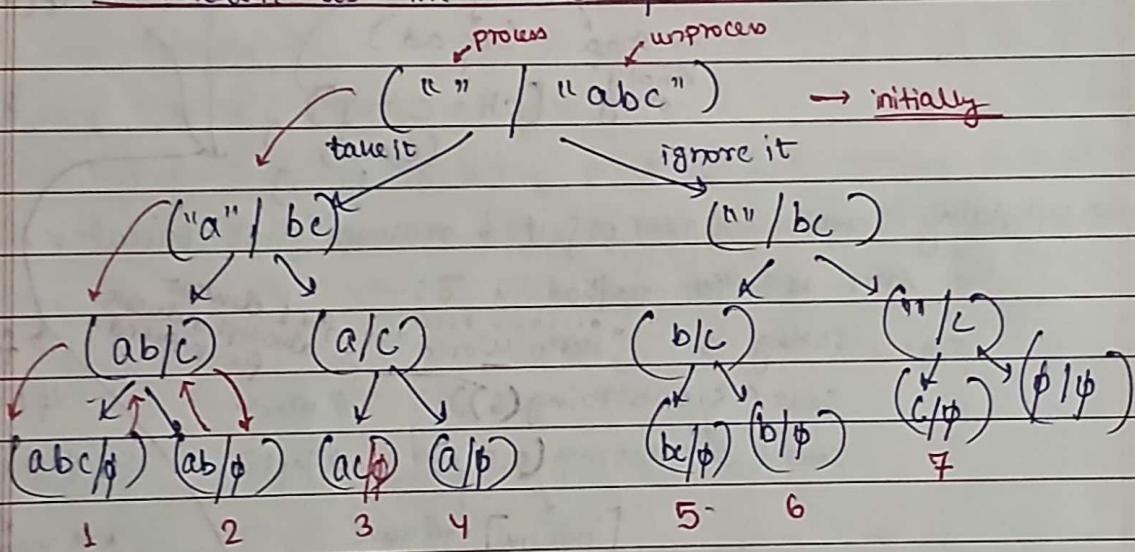
ans = ["a", "b", "c", "ab", "bc", "ca", "abc"]

i.e either ac or
ca
↑ not both

order can't be i.e ac)

I. Recursive Form

★ → This pattern of taking some elements & removing some is known as the subset pattern.



→ (return) base condition: if (up.isEmpty())

 tjf
 down(p);

 return;

char ch = up.charAt(0);

tak it: → subseq(prch, up.substring(1));

ignore it: → " (p,);

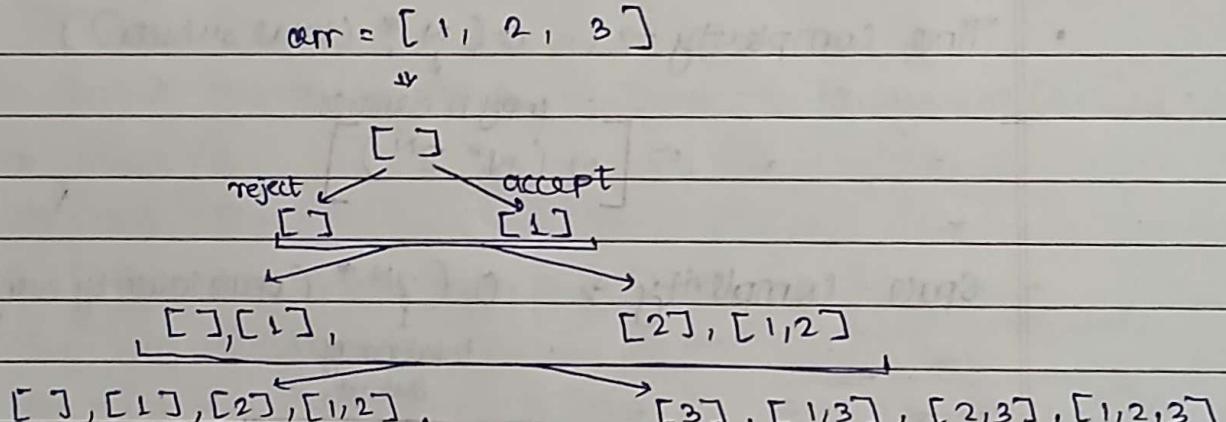
to print ASCII value → " (pt(ch+0), up.substring(1));
along with

→ for return type: `ArrayList<String>` ↗
 ↗ 1. Pass it in argument
 ↗ 2. Pass it in body

```

  static ArrayList<String> subseqSet (String p, String up) {
    base cond → if (up.isEmpty()) {
      ArrayList<String> list = new ArrayList<>();
      list.add(p);
      return list;
    }
    char ch = charAt(0);
    ArrayList<String> left = subseqSet (ch+p, up.substring(1));
    " " " right = " " (p, " );
    select both & add → left.addAll(right); / or right.addAll(left);
    return left; / return right;
  }
  for ascii → ArrayList<String> ascii = subseqSet (pt(ch+b), up.substring(1));
  left.addAll(right);
  left.addAll(ascii);
  return left;
  }
  
```

II. Iterative Form



From above observation:- second half is added in the array.

Date _____
Page _____

↓ list of list

\Rightarrow static `List<List<Integer>> subset(int[] arr)`

`List<List<Integer>> outer = new ArrayList<>();` list outside wala list

initially $\rightarrow []$ \leftarrow `outer.add(new ArrayList<>());`

just an empty list

`for (int num : arr) {` \rightarrow for every no. in my array

Jitna outer(left) \leftarrow int n = outer.size();

list hai utna ~~internal~~ (right)

list banao.

`for (int i=0; i < n; i++) {`

`List<Integer> internal = new ArrayList<>();`

add the current \leftarrow internal.add(num);

no. you are at.

`outer.add(internal);` \rightarrow now combine

for that make a copy of
outerlist in internal list + add
no. in it \cdot (outer.get(i))

COPY
Outer.get(i)

\downarrow

\Rightarrow How to call? `int[] arr = \{1, 2, 3\};`

`List<List<Integer>> ans = subset(arr);`

for every lists of list in answer \rightarrow `for (List<Integer> list : ans) {`

print that list $\}$ `sout(list);`

• Time complexity $\rightarrow O(N * \downarrow \text{total subsets})$

$$\Rightarrow [O(N * 2^N)]$$

copy n elements

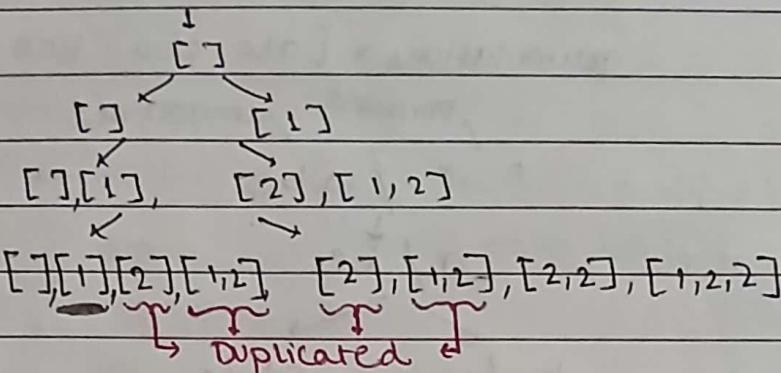
Space Complexity $\rightarrow O(2^N * \downarrow \text{space taken by every subset})$

total no. of
subsets

$$\Rightarrow O(2^N * N)$$

- Subsequence of strings with duplicate elements

```
arr = [ 1, 2, 2 ]
```



How to solve this?

SOLⁿ → only add 2 in internal list (right one). ✓

Hence, → When you find a duplicate element, only add it in the newly created subset (intertial) of previous step.

But, condition → Duplicates have to be together.

How to solve this set problem now?

↳ Sol? → SORT the array.

~~check code!~~

Note:- Direct sorting of array in Java → `Arrays.sort(arr);`

```
int start = 0;
```

```
int end = 0;
```

```
for (int i = 0; i < curr.length; i++) {
```

start = 0;

bcn condition (otherwise out of bound)

if current
same

base condition (otherwise out of bound)
 $i > 0$ & $\text{arr}[i] == \text{arr}[i-1]$ }
 Start = end + 1 ; (end while outer loop
 for last - the so abs
 End = outer.size() - 1 ; start internal loop as

```
int n = number.size();
```

```
for (int j=0; j<m; j++) {
```

```
List<Integer> listInternal = new ArrayList<>(outer.get(j));
```

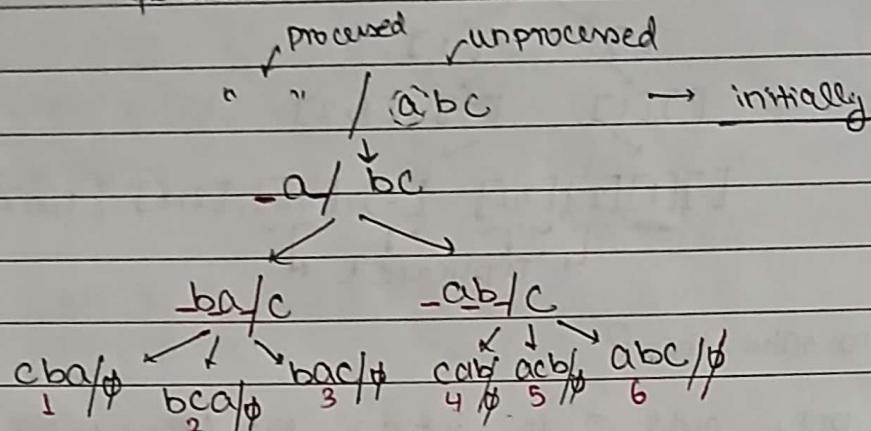
internal.add(larr[i]);

return outer; } Outer.add(Internal);

* Permutations

e.g. str = abc

permutations = [abc, cba, bca, acb, ...]



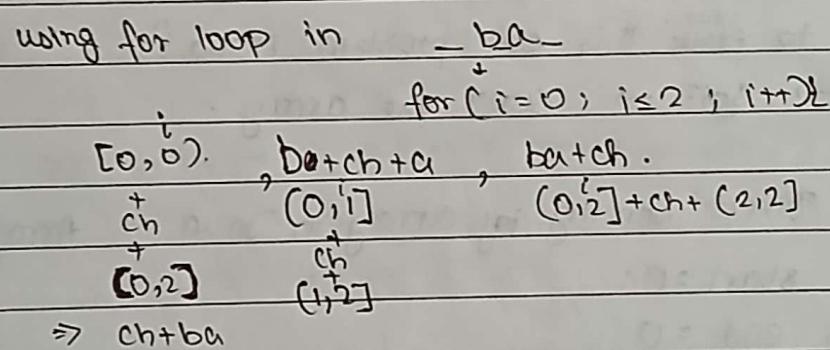
$$3! = 6 \checkmark$$

Different from rest others → Here the no. of function call is dependent on size of process.

variable
no. of
recursive
calls.

And no. of func./recursive call is size of process + 1.

e.g. using for loop in



Hence, first = p.substring(0, i)

second = p.substring(i, p.length())

p = first + ch + second

eg. abc → " " "abc"
(1) up="abc"
 char ch = up.charAt(0); → ch='a'
 f="" , i=0
 s=" "
 p=" "+a+" "+

(2) up="bc"
 ch='b'
 f=""
 s="a"
 p=cba

(3) up="c"
 ch='c'
 f="" → (0)0
 s="ba"
 p=cba
 UP=" " → empty
 SOUT(p) → return
 UP="c", P="ba"
 (earlier)
 (4) ch = c'
 NOW for i=1,

~~✓~~ static void permutations(String p, String up) {

if (up.isEmpty()) {

 sowt (p);

 return;

 char ch = up.charAt(0); ✓

variable ← for (int i=0; i <= p.length(); i++) {
rec. calls

 String f = p.substring(0, i);

 String s = " " (i, p.length());

 permutations (f + ch + s, up.substring(1));

}

⇒ do yourself for returning in case of ArrayList (by own)

↳ check screenshot if gave up.

→ check rec. level & checking counting zeros (with helper func.).

→ find counts → static int permutationsCount (String p, String up)

if (up.isEmpty()) {

 return 1;

 int count = 0;

 char ch = up.charAt(0);

 for (int i=0; i <= p.length(); i++) {

 String f = " ";

 " s = " "

 count = count + permutationsCount (f + ch + s, up.substring(i));

 return count;

* Numpad Ques. GOOGLE Q.

leetcode : letter combination of a phone no.

for simplicity, →
 (solve actual by
 your own)

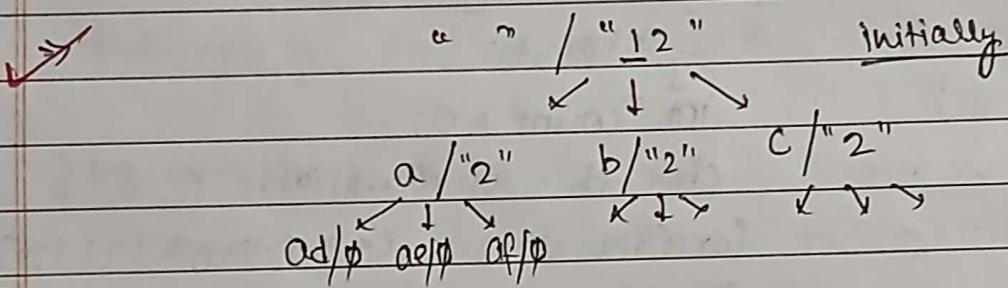
1	2	3
cde 012	def 345	ghi 678
jkl 4	mno 5	pqr 6
stu 7	vwx 8	yz 9

e.g. "12"

ad, ae, af, bd, be, bf, cd, ce, cf



→ Same as earlier concept of taking ~~d~~ & removing ~~bc~~ ←
 (process/unprocess).



⇒ range of each digit → $[(\text{digit}-1)^3, \text{digit}^3]$
 (index 0 to 8)
 e.g. 5 → $[12, 15)$ e.g. 1 → $[0, 3)$

$\begin{matrix} 12, 13, 14 \\ \hline m \quad n \quad o \end{matrix}$

$\begin{matrix} 0, 1, 2 \\ \hline a \quad b \quad c \end{matrix}$

base → When index = 26 STOP.
 (only)

e.g. here i will run from $[0, 3) \rightarrow 0, 1, 2$

$\begin{matrix} 0, 1, 2 \\ \hline a \quad b \quad c \end{matrix} \leftarrow$

to get character from no.
 just do = 'a' + index

~~debug & check
which is doing what~~

static void pad (string P, string UP) {

if (UP.isEmpty ()) {
cout (P)
return.

int digit = UP.charAt (0) - '0'; → convert '2' into 2.

converting first
placed string into
no.

for (int i = (digit - 1) * 3; i < digit * 3; i++) {

char ch = (char) ('a' + i);

pad (P + ch, UP.substring (1));

↳ ↳ UP = "12" (String)

only do for list in
arguments
(easy)

→ In this Ques. we've to return in list. (already covered just copy paste).

static ArrayList<String> padList (" ", " ") {

if (UP.isEmpty ()) {

ArrayList<String> list = new ArrayList<>();

list.add (P);

return list;

int digit = UP.charAt (0) + '0' ;

ArrayList<String> list = new ArrayList<>();

for (int i = ...; ...; ...) {

char ch = " ";

list.addAll (padList (P + ch, UP.substring (1)));

}

return list;

→ can also perform for count of pad.

Q. now I → no letter, abc from 2, 7 has pqr & g has xyz.
↓ abc from 2, abc from 2, 7 has pqr & g has xyz.

Start from 2. char ch = (char) ('a' + i - 3)

↳ only for g → [24, 27]

a=0, b=1, c=2

:(-2) ↗

∴ (digit - 1) * 3 = 18 ↓ 18 = 15 ↓ for 15 :- 3

pqr's

∴ 8 → 19 20 21

[21, 24]

char → 'a' + i - 3

1 - 2 4 i < digit * 3

(not equal)

∴ 11 8 →

∴ digit * 3 = 21
↳ to get 18 → i < s .

* Numbers on a Dice AMAZON Q.

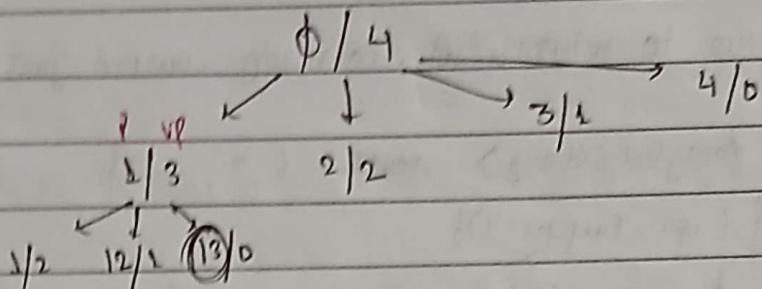
Die has $\rightarrow [1, 2, 3, 4, 5, 6]$ \rightarrow some initial data

e.g. 4 on a die possible combinations:-

$[4, 122, 22, 31]$

sum \rightarrow
Should be 4.

\rightarrow taking something
ignoring
already used!



Condition: $i \leq \text{target}$

\Rightarrow static void dice (String p, ~~int~~ int target) {
 if (target == 0) {
 cout(p);
 return;
 }
 for (int i = 1; i <= 6 && i <= target; i++) {
 dice(p + i, target - i);
 }
}

\downarrow debug
4 check
what's happening
 \downarrow target = 4 i = 3
 \downarrow p = "1" + 4 = "4" + "1" = "41" + "1" = "411" + "1" = "4111"

\downarrow 4 3 2 1 ..
 \downarrow 3 6 10

CHECK

\Rightarrow Why, do for returning via list. (yourself)

\Rightarrow If die face is more than 6.

\hookrightarrow pass another argument of int face.
 \hookrightarrow for (int i = 1; i <= face && i <= target; i++) {
 diceFace (p + i, target - i, face);

* Backtracking

- Maze Problem

↑	○		○
○			
1			○
2	○	○	

can move only right & down.

$$\text{Ans.} \Rightarrow \begin{array}{c} RRDD \\ DDRR \\ \hline RDDR \end{array}$$

Q. How many ways of going from $(0,0)$ to $(2,2)$?

Breaking down into diff smaller approaches :-

\Rightarrow if you are tots at last column \rightarrow only possible way is to DOWN
 \Rightarrow " " " " row \rightarrow " " " " Right.

\Rightarrow if $RD + C$

$DR + C \rightarrow$ Path till that point.

For no. of ways:-

⇒ Note:- same curr. will be less opposite.

$\therefore \Rightarrow$

(3,3)

Down →

3.2 ← right

1

113

RR i.e. right only !.L

A handwritten musical staff on five-line paper. The first measure contains a C major chord (C, E, G) in the treble clef. The second measure contains a D major chord (D, F#, A) in the treble clef. The third measure contains a G major chord (G, B, D) in the treble clef.

→ I can only ~~do~~

We've already calculated for
(2,2) still checking here redundancy
will be covered in DP ~~in~~

11

static int count (int r, int c) {

if ($r == 1$ || $c == 1$) {

, return 1;

int left = count(r-1, c); DOWN

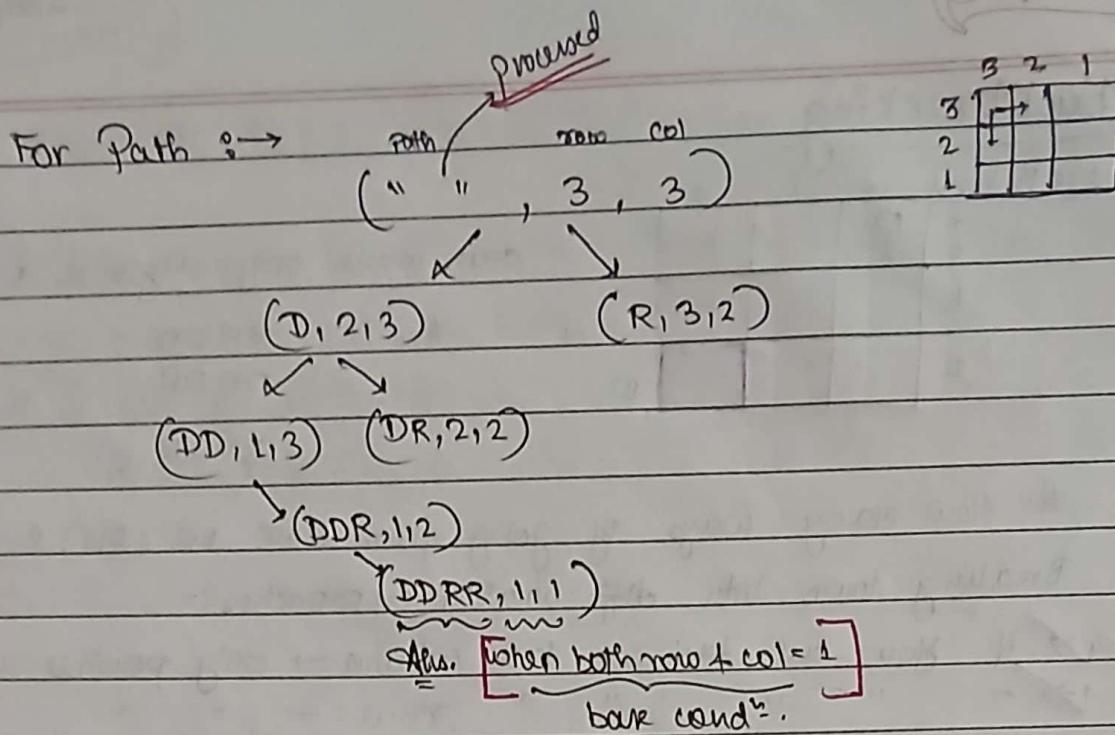
int right = " (r, c-1); right

return left + right;

H.W (memorization + tabulation)

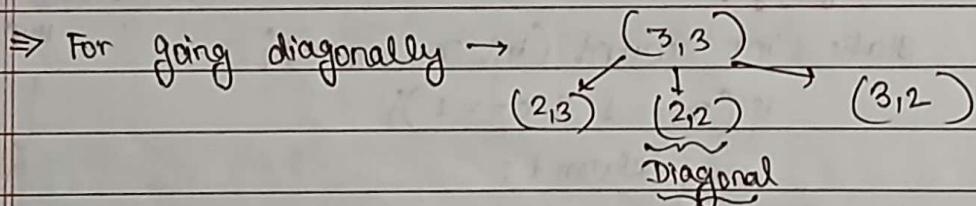
Gives

of ways.



✓ static void path (String p, int r, int c) {
 if (r == 1 && c == 1) {
 cout (p), earlier for cout it was "0111"
 return;
 if (r > 1) {
 path (p + 'D', r - 1, c);
 } if (c > 1) {
 path (p + 'R', r, c - 1);
 }

⇒ Do for returning in ArrayList.



i.e. you can't go diagonal ← condition: ($r > 1 \text{ and } c > 1$)
 ✓ when you are at last col and row.

↳ just add one more condition → if ($r > 1 \text{ and } c > 1$) {
 path (p + 'X', r - 1, c - 1);
 } * *

Maze with obstacles

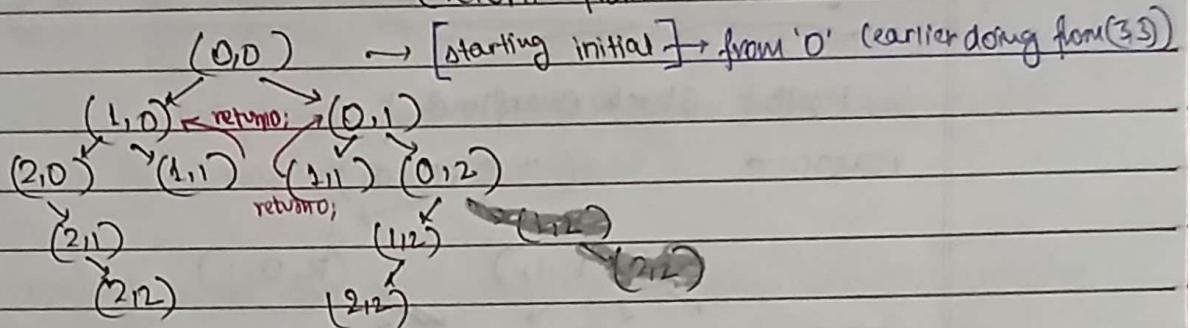
	0	1	2
0	.	.	.
1		Wavy lines	
2			

(false → river) → Boolean

✓ Note:- when you land on a new cell,
check whether that is river or not.

If you land on river,
stop recursion for that call.

(return from river)



Now this time we are starting from (0,0) to (2,2) earlier we were doing the opp.

```
⇒ static void pathRestrictions( String p, boolean[][] maze, int r, int c)
    if (r == maze.length - 1 && c == maze[0].length - 1) {
        Sout(p);
        ↗ reached at the end by
        return;
        ↗ maze.
```

if (maze[r][c] == false) {

 return;

 if (r < maze.length - 1) {

, PathRestriction(p + 'D' → maze, r+1, c);

 if (c < maze[0].length - 1) {

 "path" (p + 'R' , maze , r, c+1);

 }

Create board in main: → boolean[][] board = { {true, true, true},
 { " ", false, " " },
 { " ", true, " " } }

pathRestrictions("", board, 0, 0);

Maze for all Directions

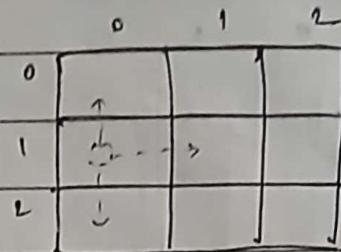
✓ Left, right, up, down.

✓ If you are going:-

up \Rightarrow $\tau - 1, c$ (condition: $\text{row} > 0$)

left \Rightarrow $\tau, c - 1$ (" : $\text{col} > 0$)

right + down \rightarrow we know.



result: Stack Overflow!

reason: \rightarrow

$(\swarrow, \searrow, 0, 0)$

$(D, L, 0)$ $(R, U, 1)$

$(DL, 2, 0), (DR, 1, 1), (DU, 0, 0)$

✓ Problem \rightarrow again reached $(0, 0)$ hence never ending!

✗ Sol: \rightarrow Do not move back the same path.

✓ All cells that are visited, mark those as false, so that it does not go there.

↳ same concept as we did in Maze with obstacle.

✓ put all those (row, col) as false which are already visited

✓ Another Problem \rightarrow To find $(DL, 2, 0)$ if all the false $\text{row} - \text{col}$ ke wo to false rhe gya, so when we traverse from $(DR, 1, 1)$ \therefore we had down aane pr false marked hai but ye uspe kabhi gaya hi nahi still!

* \therefore so when previous path is over & now you're in another rec^{to}. case, then these cells should not be false.]

Solution:-

→ While you're moving back, you restore the maze as it was.

So, when do we go back? → When fn. is returned
when you come out of recursive func. → You're now
in above recursion call. Hence, while going
back (↑) → remark the cells as true!

THIS IS KNOWN AS BACKTRACKING

(What if I had not taken this path what would
my array look like).

You're making some changes while going from below recursive
calls, so when you go outside those recursion calls,
changes that were made via those recursion calls
should also not be available. This is known as
Backtracking. → Thought process: make a change, reverse the
change when that work is done.

↳ Check all paths code!!

Print all paths with numbers

→ Take a step variable.

e.g.

1		
2	5	6
3	4	7

→ Update the path array.

→ Print it in back cond.

⇒ Backtracking, as array, integer DDRURD

⇒ pass path & step in argument.

⇒ with 'every' rec. call step+1.

⇒ Initially path[r][c] = step, after (for) backtrack → path[r][c] = 0;

→ for base cond → for every path in arr → print that array.

→ Note: in main declare path as array of size board!

→ check code.

* N-Queen Problem

e.g. N=4 (place 4 Queens)

Place 2 Queens on board

so that no 2 Queens with
their path eliminate each
other.

0	X	Q	
1		X	Q
2	Q		
3			Q

↳ one of the ~~solutions~~ answers.

How, recursion? → checking again + adding/ignoring.

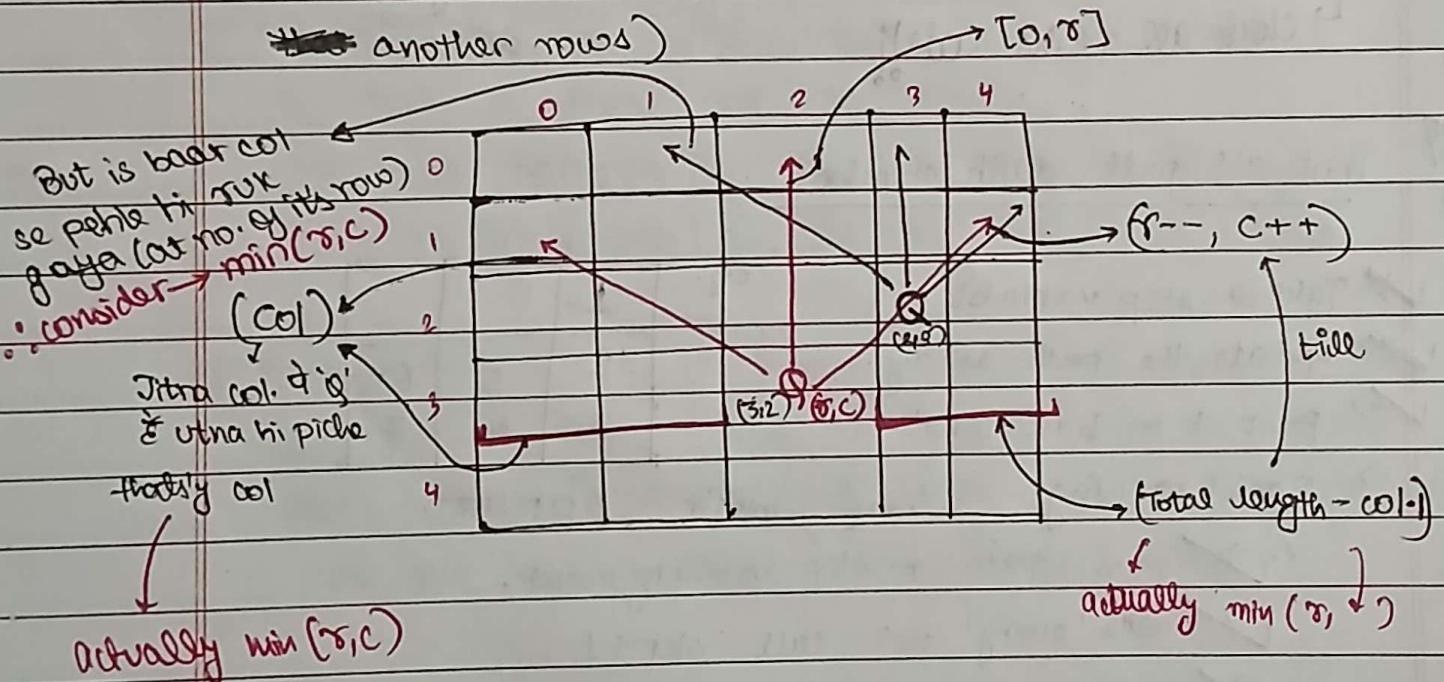
How, Backtracking? → (what changes you're making in that ~~rec~~ rec².
call when that ~~rec~~ call is over those
changes should also be gone).

here, (1,2) was not a feasible path so
Q placed there should remove.

∴ putting there we
can't put 4 Queens

⇒ Some imp. checks e.g. if Q is in center of 3×3 matrix

then no other Queen will be placed i.e. (no need to check
~~another rows~~)



*Date _____
Page _____*

⇒ static int queens (boolean[][] board, int row) {

 if 'Q' placed ← if (row == board.length) {

 display (board);
 return 1;
 }

 add a new line after displaying
 board for another

 int count = 0; → local variable,

 placing the queen + → for (int col=0; col < board.length; col++) {

 checking for each row
 + col.

 if (isSafe(board, row, col)) { → place 'Q' if safe

 board [row][col] = true; → placed, now for below
 ones can be placed
 or not

 count = count + queens (board, row+1);

 coming out of func. → board [row][col] = false;

 calls 'Backtrack'

}

{ return count;

}

static boolean isSafe (boolean[][] board, int row, int col) {

 check vertical →

 for (int i = 0; i < row; i++) {

 row → [0, r]

 if (board[i][col]) {

 mtlb if this is true
 = board[i][col] == true

 return false;

}

 Not safe to place it
 'cause it's a 'Q' over there

for max. time you can → int maxLeft = Math.min (row, col);

go diagonal left

min (r, c)

for (int i = 1; i <= maxLeft; i++) {

 if (board[row-i][col-i]) {

 return false

 if, diagonal right →

(length - col - 1)

with (r++, c++)

int maxRight = Math.min (row, board.length - col - 1);

for (int i = 1; i <= maxRight; i++) {

 if (board[row-i][col-1]) {

 return false;

If none of the check returns → return true,

false :- ans. is true

display \Rightarrow static void display (boolean[][] board) {
 for every row on board \Rightarrow for (boolean[] row : board) {
 for (boolean element : row) {
 if (element) {
 sout("Q");
 } else {
 sout("X");
 }
 sout(); \rightarrow next line
 }
 }
}

Main \Rightarrow int n = 4; \rightarrow no. of Q's \rightarrow on 8x8 or 4x4 board
 boolean[][] board = new boolean[n][n];
 sout(queens(board, 0));
 } \downarrow row initially.

\rightarrow Space complexity $O(n \times m)$.

\rightarrow Recurrence relation:

$$T(N) = N * T(N-1) + O(N^2)$$

for every col it will check for m^2 elements

for n columns (check in $n-1$ again & again)

some Time Complexity using Agra-Bazzi formula.

$$O(N^3 + N!) = [O(N!)]_{\text{Ans}}$$

\rightarrow Note: For these problems: You can eliminate for loops with conditions, but then you need another variable in argument.

e.g. (board, row, col, target)

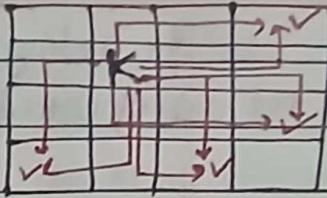
Ans. found when
 $\text{target} = 0$.

(0, 0, 4)
 (1, 0, 3)

* N-Knights Problem

	0	1	2	3
0	K	K	K	K
1				
2				
3				

rows cols
 ↓ ↓
 (0,0,4)
 ↓
 (0,1,3)
 ↓
 (0,2,2)
 ↓
 (0,3,1)
 ↓
 (0,4,0)



∴ One of the ans. is found

Now, whenever you reached end → try for a new line.

∴ $(0,0,4) \rightarrow (0,1,3), (0,2,2), (1,1,1)$ ↗

$$\text{safe} \Rightarrow \begin{bmatrix} r-2, & c-1 \\ r-2, & c+1 \\ r-1, & c+2 \\ r-1, & c-2 \end{bmatrix}$$

⇒ static void knight (boolean[][] board, int row, int col, int knights)

if (knights == 0) {

display(board); → make display same as of Queen
 } return;

make if reach end
 row++ col just return

if (col == board.length) { → if reached end of col, increase
 row by 1 & check for another.

Knight(board, row+1, 0, knights);

} return;

if (!isSafe (board, row, col))

place it when → board[row][col] = true;

safe + col++ knight (board, row, col+1, knights - 1);

Backtracking → board[row][col] = false;

If not safe : move →

knight (board, row, col+1, knights);

already & do not

}

reduce knight : You didn't

placed it

for checking places are → static boolean isValid (boolean[][] board,
not out of bounds in Matrix int row, int col) {
if (row >= 0 && row < board.length && col >= 0
&& col < " ") {
} return true;
} return false;
}

for checking safe in → static boolean isSafe ("", "", "") {
valid places only.
if (isValid (board, row-2, col-1)) {
if (board[row-2][col-1]) {
} return false;
}
if (isValid ("", row-1, col-2)) {
" " [row-1][col-2]);
" " " (" , row-2, col+1);
" " (row-2)[col+1]);
" " " (" , row-1, col+2);
" " [row-1][col+2]);
} return true; → if all not ~~not~~ valid.
}

* Sudoku Solver

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

Rules: You know

We know how to

Check rows + col but
for each block.

e.g. I'm at (4,4) we've
to find starting of
the box where '1' lies



$$x = x \% 3$$

$$y = y \% 3$$

$$\Rightarrow 4 - 4 \cdot 1 \cdot 3 = 1$$

$$4 - 4 \cdot 1 \cdot 3 = 1$$

e.g. we're at (6,8)

remainder
that's 'y' this

formula.

$$\Rightarrow (6,3) \checkmark$$

and 3 bcoz $\sqrt{9}$.

$$\text{starting} \Rightarrow 6 - 6 \% 3 = 0$$

$$8 - 8 \% 3 = 2$$

$$\Rightarrow (6,2) \times$$

to check if same
no. already here

static boolean isSafe(int[][] board, int row, int col, int num){

1. Check the row → for (int i=0; i < board.length; i++) {

check if the no. is → if (board[row][col] == num) {

in the row

} return false;

2. Check the col → for (int[] nums: board) {

check if the no. is in → if (nums[col] == num) {

the col (by checking

diff. row of same col)

} return false;

```

3. int sqrt = (int)(Math.sqrt(board.length));
int rowStart = row - row % sqrt;
int colStart = col - col % sqrt;
    c.g.g → sg=3 ek block में 3x3 एवं एक matrix
    ban payega
for (int r = rowStart; r < rowStart + sqrt; r++) {
    for (int c = colStart; c < colStart + sqrt; c++) {
        if (board[r][c] == num) {
            return false;
        }
    }
}
return true;
}

```

Now, static boolean solve(int[][] board)

```
int n = board.length;
```

```
int row = -1;
```

```
int col = -1;
```

```
boolean emptyLeft = true;
```

```
for (int i = 0; i < n; i++) {
```

```
    for (int j = 0; j < n; j++) {
```

```
        if (board[i][j] == 0) {
```

```
            row = i;
```

```
            col = j;
```

value stored there \leftarrow emptyLeft = false;

```
        break;
```

empty

if some element is found \rightarrow if (emptyLeft == false)
in row or already present \rightarrow break;

if all filled
(sudoku solved)

\rightarrow if (emptyLeft == true){
return true;
}

```

backtrack → for (int number = 1; number <= 9; number++) {
    if (isSafe(board, row, col, number)) {
        put no. if safe
        board[row][col] = number;
        if (solve(board) == true) "or" (solve(board)) {
            print board if answer ←
            "found answer" ← display(board);
            return true;
        } else {
            backtracking (pichha-jakar ←
            ke hata te jao)(jab tak
            sati wala no. sati
            place pe na ho)
            board[row][col] = 0;
        }
    }
}
sudoku can't be solved ← return false;
    
```

```

display → static void display (int[][] board) {
for every row in board → for (int[] row : board) {
    " " no. in row → for (int num : row) {
        sout (num + " ");
        sout(); → new line (arranged)
    }
}
    
```

```

main → int[][] board = new int[][] {
    {3, 0, 6, - - -}
}
    
```

```

    };
if (!solve (board)) sout ("cannot solve");
else {
    display (board);
}
    
```

- Complexity : Total g^m nos.
for every no. $\rightarrow n^2$ possibility.

Time : $O(g^m)$, Space : $O(n^2)$