

- 1) Programming language ✓
- 2) Leetcode.
- 3) Project
↳ Upsolving (learn from others solution)
- 4) Make resume

In some question int, can't handle the value of the test cases so take long instead of int.

- To iterate from left and also not going outside the no. of bit we check \rightarrow while ($\text{bitmask} < N$) {

bitmask = bitmask << 1

Bit Manipulation

• Decimal to Binary

ex - 4

$$\begin{array}{r|l} & \text{remainder} \\ \hline 2 & 4 \\ 2 & 2 \\ \hline & 1 \end{array}$$

$$\Rightarrow (4)_{10} = (100)_2$$

0	=	000
1	=	001
2	=	010
3	=	011
4	=	100
5	=	101
6	=	110
7	=	111
8	=	1000

• Binary to Decimal

ex- 100

$$1 \times 2^e \quad 0 \times 2^1 \quad 0 \times 2^0$$

$$= 4 + 0 + 0$$

Bit-wise Operators

- Binary • AND &
 - OR |
 - XOR ^
 - One's Complement ~
 - Shift left <<
 - Shift Right >>

AND 8

080 = 0

ex-101

$$0.81 = 0$$

$$180 = D$$

181 = 1

$\Rightarrow (4)_n$

OR 1

$$0 \mid 0 = 0$$

11

$$110 = 1$$

— 1 —

(-1)

$$A = 0101$$

XOR ^

$$\begin{array}{r} \rightarrow 0 \wedge 0 \\ 0 \wedge 1 \\ 1 \wedge 0 \\ 1 \wedge 1 \end{array}$$

$$5 \wedge 6$$

$$A = 101 \quad B = 110$$

$$\begin{array}{r} 101 \\ 110 \\ \hline 011 \end{array}$$

part of 1's complement

$$\Rightarrow (3)_{10}$$

Binary One's Complement ~

$$\begin{array}{r} \text{ex-} \\ \sim 0 \quad 1 \quad \sim 5 \\ \sim 1 \quad 0 \end{array} \quad A = 0101$$

$$\sim (101) = 010$$

$$\Rightarrow 2_{(10)}$$

But computer gives us -6

Because in computer there 8 bits

$$\begin{array}{r} \text{ex-} \quad 00000101 \\ \text{MSB} \qquad \qquad \qquad \text{LSB} \\ \boxed{\text{Most significant bit}} \qquad \boxed{\text{least significant bit}} \end{array}$$

This method is followed when we convert the negative no.

if MSB = 0 \rightarrow the no. is positive
 MSB = 1 \rightarrow the no. is negative

So when we do ~ 0101 is actually does

$$\begin{array}{r} \sim 5 \rightarrow 00000101 \\ 2^{\text{complement}} \rightarrow \underline{111111010} \end{array}$$

This is for the value of MSB so it is 1

$$1^{\text{complement}} \rightarrow 00000101$$

\therefore no. is negative

$$\text{Add } 1 \rightarrow 00000101$$

\rightarrow This type of addition is done in bit wise

so it is -6

2^s Complement \rightarrow $\begin{cases} \textcircled{1} \text{ } 1^s \text{ complement not (n) } \sim n \\ \text{and } \textcircled{2} \text{ Adding 1} \end{cases}$

$$\text{ex- } \sim 0 = -1$$

$$\text{how } \rightarrow 0 \rightarrow 00000000$$

$$\sim 0 \rightarrow 11111111 \rightarrow \text{For MSB}$$

↓
One's complement 00000000

$$\begin{array}{r} + 1 \\ \hline 00000001 \end{array}$$

$\Rightarrow 1$ but negative

$$\cancel{\cancel{1}} \Rightarrow -1$$

MSB = 1
 \therefore negative

Binary Left Shift <<

this no.
is shifted $\underbrace{a}_{\downarrow} << b (2)$

$$\Rightarrow 1010 \quad 5 << 2$$

$$\Rightarrow 000101$$

after shift

$$\begin{array}{l} 010100 \\ \Rightarrow (10100)_2 \rightarrow 1 \times 2^4 + 1 \times 2^2 \\ \qquad \qquad \qquad \Rightarrow 16 + 4 \\ \qquad \qquad \qquad = 20 \end{array}$$

Shortcut
Formula

$$a << b = a * 2^b$$

$$\text{ex- } 5 * 2^2 \\ \Rightarrow 20$$

Binary Right shift >>

$$\text{ex- } 6 \Rightarrow \begin{array}{r} 000110 \\ \times \cancel{\cancel{1}} \\ 001000 \end{array}$$

$$6 >> 2$$

$$\Rightarrow (000001)_{10} = 1$$

\rightarrow shift the bits right side

formula

$$a >> b = a / 2^b$$

Question - Check if the number is Odd or Even

→ For this we will and (&) the no. with **1**

00000001 ← This number told us that is the number is odd or even that is by LSB
* Least significant bit

② But for this we need all other ~~0~~ 0

∴ We will and the no. with **1**

→ ex - & 01011011
00000001
00000001 ← ∴ this no. is odd.

index is counted using 0 so make bitmask make i-1

→ ex - & 100 (4)₁₀
001 ← even.

And this is called as bitmask

The operation we performed is & and bitmask we used is 1

Get ith bit

↳ Just to check the ^{ith} bit is 0 or 1

↳ we will take & with that bit,

Taking and operator with the ^{ith} position of this no.

only high code works

```
static int getithbit(int n, int i){  
    int bitmask = 1 << i;
```

↳ bit of which no. So far that will will input from user and take (1)₁₀.

→ 00000001

then in $\&(1 << i)$

Set i th bit \rightarrow Make a selected bit 1.

$\Rightarrow n | (1 \ll i)$ to make change make that bit 1

\Rightarrow ex-

$$\begin{array}{r} 10 \rightarrow 1010 \\ 0100 \text{ or} \\ \hline 1110 \end{array}$$

we done $10 | (1 \ll 2)$

$$\therefore 1110 = (14)_{10}$$

Clear i th bit \rightarrow making that bit zero

$n \& \sim(1 \ll i)$

ex- $10 \rightarrow 1010$
~~1000~~
 $\hline 1000$
 $\Rightarrow (8)_{10}$

Update i th bit

\hookrightarrow position will be given (i)

and the user will tell he want 1 or 0 on that place

\Rightarrow _____

Update ~~the bit~~ Just add a if ~~else~~ else \rightarrow if ($i=0$) or
if ($i=1$)

\hookrightarrow use of Another Approach

int newbit \rightarrow bit you
want on
that place
0 or 1

i) clear the bit first

ii) bitmask = [newbit $\ll i$]

\Rightarrow $\boxed{\text{clear_}i\text{-th-bit}(n: i);}$
 $\boxed{\text{int bitmask} = \text{newBit} \ll i;}$
 $\boxed{\text{return } n \mid \text{bitmask};}$

Clear last i bits

ex $n = 1111_0, i=2$

$\Rightarrow 1100$

\Rightarrow for this we use $(-1 \ll i)$ or $(\sim 0) \ll i$

$$\Rightarrow (\sim 0)_0 = 1111$$

Shiftin left i times

$$n \& (-1 \ll 2)$$

$$\Rightarrow 11111110$$

$$\begin{array}{r} 1111 \\ \times 100 \\ \hline 1100 \end{array}$$

$$\rightarrow 11111100$$

$$= 1100 = (2)_0$$

Clear Range of bits

$\Rightarrow 100111010011, i=2, j=7$

\Rightarrow clear bits from j to i

that is

$$\begin{array}{r} 100111010011 \\ \downarrow \quad \downarrow \\ 100100000011 \end{array}$$

$$\Rightarrow 100100000011$$

from j to i

Logic

$$a \Rightarrow (\sim 0) \ll j+1$$

$$\Rightarrow 111100000000$$

$$b \Rightarrow 01 = (1)_0 = 2^1 - 1$$

$$011 = (3)_0 = 2^2 - 1$$

$$0111 = (7)_0 = 2^3 - 1$$

$$01111 = (15)_0 = 2^4 - 1$$

for 011

\rightarrow

\therefore for $011 = 2^2 - 1$

$$\Rightarrow ((1 \ll i)) - 1$$

$$2^b \rightarrow 1 \ll b$$

$\therefore a \mid b = \text{clear bits bit}$

bitmask $\Rightarrow (\sim 0) \ll j+1 \mid (1 \ll i) - 1$

\therefore bitmask

\rightarrow required answer

$$ex - (10)_{10} \rightarrow \begin{array}{r} 000 \\ \text{J} \end{array} \begin{array}{r} 10 \\ \text{J} \end{array} \begin{array}{r} 10 \\ \text{J} \end{array} \quad \begin{array}{l} i=2 \\ j=4 \end{array}$$

$$ans \Rightarrow 0000010 = (2)_0$$

Check if a number is power of 2 or not ~~*~~

$$16 \rightarrow 2^4 \rightarrow \begin{array}{r} 1000 \\ 3 \end{array} \rightarrow 0$$

$$8 \rightarrow 2^3 \rightarrow \begin{array}{r} 1000 \\ 7 \end{array} \rightarrow 0$$

$$\boxed{n \& (n-1) == 0}$$

if this is true
then the no. is power
of 2.

Fast Exponentiation \rightarrow to take the exponentiation fast
 \rightarrow to take out power fast

ex - $3^5 \rightarrow$ for this we need to multiply $3 \times 3 \times 3 \times 3 \times 3$

but we will use bit here

$$\Rightarrow 3^{101} \rightarrow ① a = 3, ans = 1$$

$$ans = 3$$

$$a \rightarrow a^2 \rightarrow 9$$

$$\begin{array}{r} a^4 a^2 a^1 \\ 101 \\ 3 \downarrow (a^1) \\ at 0 \\ multiply with 1 \end{array}$$

$$② a = 9, ans = 3$$

$$ans = 3 \times 1$$

$$a \rightarrow a^2 = 81$$

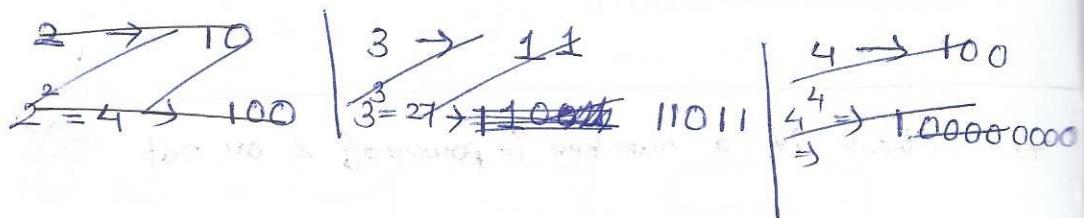
$$③ a = 81, ans = 3$$

$$ans = 3 \times 81 = 243$$

$$a \rightarrow a^2 = 81^2 = 243$$

H.W. # Find the value of x^* if x is given? (Bit manipulation)

→



~~5~~ → ~~101~~
5 →

$\wedge \rightarrow (\text{XOR})$
not of 01

$$x = 100$$

$$100 \wedge 100$$

$$\begin{array}{r} 100 \\ 100 \\ \hline 000 \end{array}$$

$x \wedge x$ will always be 0

Swaping two number's without using 3rd variable.

$$\text{Ex - } x = 3 \rightarrow 11 \\ y = 4 \rightarrow 100$$

i) $x = x \wedge y$

ii) $y = x \wedge y$

$x = x \wedge y$

$$\left| \begin{array}{l} x = \frac{100}{111} = 7 \text{ (dec)} \\ y = \frac{111}{100} \rightarrow 3 \text{ (dec)} \\ x = \frac{111}{011} \rightarrow 4 \text{ (dec)} \end{array} \right.$$

$$x = 4, y = 3$$

after swaping.

$$\# \quad -x = \sim x + 1$$

- Negative of a number invert its bit and add 1 to it

\therefore To add 1 using bit

$$\Rightarrow -\sim x = x + 1$$

just replaced
x by $\sim x$

To Convert Uppercase \rightarrow Lowercase Character

~~char~~ char a = 'A';
 Sys0((char)(a | ' '));
 type cast
 from bit
 to character.

ch = 'A';
 ch1 = 'b';

\Rightarrow For Uppercase to lowercase \rightarrow

For Lower to uppercase \rightarrow

(char)(ch | ' ')
 (char)(ch ^ ' ')

Object oriented programming

OOP's

* Questions in interview

→ Blueprint of an object.

Classes → group of this entities / collection of objects of similar properties.

Objects → ~~entity~~ entities in real world.

Ex - Pen
 Properties :
 - Med
 - long
 - function

Example - Pen

Class

① attributes (Properties) color (String)

② functions (behaviour) changeColor()

- Conventional
 - Name of class start with Capital Letter
 - Name of function start with Small Letter

↳ public class is at top above

↳ So that no error in program

Ex:-

public class OOPS {

 public static void main(String args []) {

 Pen p1 = new Pen(); ← created a pen object

 p1.setColor("Blue"); ← accessing the color in pen object

 System.out.println(p1.color);

}

class Pen { ← another class

 String color;

 int tip;

 void setColor(String newColor) { ←
 color = newColor;

}

Function inside
the class

* Classes cannot be protected or private because it will be of no use then.

#	<u>Access</u>	<u>Modifiers</u>	<u>outside class by subclass only</u>	<u>outside package</u>
1) Private	✓	X	X	X
2) Default	✓	✓	X	X
3) Protected	✓	✓	✓	X
4) Public	✓	✓	✓	✓

Getters & Setters

- Get → to return the value.]
- Set → to modify the value.]

→ If the properties of the object are private we need to use functions to use the value and change the values.

- this ↳ this keyword is used to refer to the current object.
- ↑
 keyword ↳ Usefull when objects or parameters are of same names.

OOP's

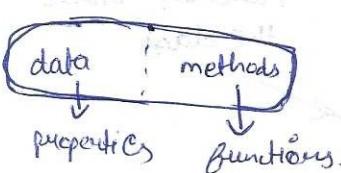
- Encapsulation
- Inheritance
- Abstraction
- Polymorphism,

Learn definition

*

1. Encapsulation →

↳ Just like a capsule



Defined as the wrapping up of data and methods under a single unit. It also implements data hiding.

↳ by using access modifiers.

PDPP
private - default - protected - public

Constructor

↳ It is a special method which is invoked automatically at the time of object creation.

- 1) have same name as class or structure
- 2) don't have a return type (Not even void)
- 3) Are only called once → At object creation.
- 4) Memory allocation happens when constructor is called.

Ex -

Pen p1 = new Pen();

Types of Constructors

① Non-parameterized → No parameter is used.

Ex - Student();

② Parameterized → with parameters.

We can have as many types of constructor as many type of parameter and their sequence we give.

③ Copy Constructor:

↳ by default in C++

↳ need to made in Java

② And default constructor will only work if we give no constructor.

↳ To copy the properties from one object to other.

① Shallow Copy

Ex - main()

student s1 = new Student();

s1.name = "ABC";

s1.roll = 456;

Student s2 = new Student(s1);

② Deep Copy

this.name = s1.name;
this.roll = s1.roll;

}

In shallow the reference get copied. therefore if we make changes in first or original it will also reflect in copy.

Class Student {

String name;

int roll;

Student(Student s1){

Deep Copy

* constructor overloading
↳ When we many functions and any of that is used as per the requirement.

Destructors

- ↳ In C++ their need to make this functions.
- ↳ In java it is automatically done using garbage collector.

Inheritance

→ Inheritance is when properties and methods of base class are passed on to a derived class.

Class A Base/Parent

Class B Derived/Child

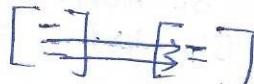
Class A → Class B

Example :

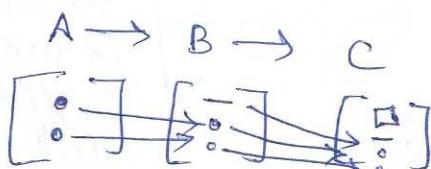
```
class A extends B {  
    ...  
}
```

Types

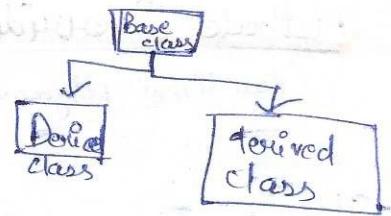
• Single level Inheritance : example → A → B



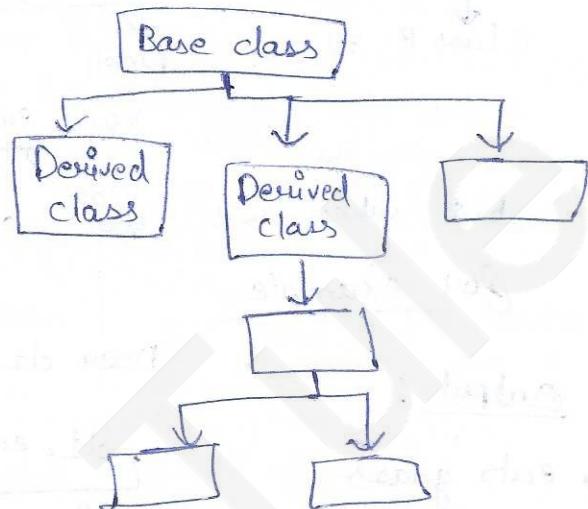
• Multi level Inheritance : example



Hierarchical Inheritance :-



Hybrid Inheritance :-



* Multiple inheritance → available in C++ but not in Java

Poly morphism

→ When try to do the same thing in multiple forms.

many forms
(static)



① Compile Time Poly morphism (Method Overloading)

② Run Time Poly morphism (Dynamic)

Type → Method Overriding

Method Overloading

→ Multiple function with same name but different parameters

ex- Calculator {

But all have same name. {

$\text{sum}(\text{int } a, \text{int } b)$ $\text{sum}(\text{float } a, \text{float } b)$ $\text{sum}(\text{int } a, \text{int } b, \text{int } c)$	Here the data type is different. Here the count is different.
---	--

Methode Overriding → Parent and child classes both contains the same function with different definition.
(Run time polymorphism)

Class A =

Animal
void eat() → "eat anything"

↓
Class B =

Deer

void eat() → "eats grass";

but when we will call the function

for example

Output:

→ eats grass

Prints parent class output.

Deer d1 = new Deer();

{ d1.eat(); }

this will call the function made inside the deer function.

Packages in Java:

classes
subpackages interfaces

is group of similar types of classes, interfaces and sub packages.

Types :

1) Inbuilt

2) User defined.

Inbuilt

ex - Scanner sc = new Scanner();

This is inside utility package → import java.util.*;

Userdefined

Package name ;

Abstraction → Hiding all the unnecessary details and showing only the important parts to the user.

• Difference b/w

* Only data can be hidden that we don't want to show

Encapsulation and Abstraction

* Here we can do both things
Hide the data that we don't want to see and also show the data that we want to show.

Concept

- ① Abstract Classes
- ② Interfaces

* Constructor are called hierarchical wise

A
↓
B
↓
C

Here first it will call A

then B

then C

↳ Just add abstract word in front of class.

Properties:

- i) Cannot create an instance/object of abstract class.

- ii) Can have abstract/non-abstract methods

- iii) Can have constructor (functions)

↳ This helps to pre-initialise any value or properties in all the classes.

↳ While making the abstract function we cannot ~~call~~ assign the value there. And if we extends the class in which the abstract function is made then it is compulsory for the other classes to use the abstract function that was made using abstract keyword.

Interfaces

ex- [Car, Car2]-Object

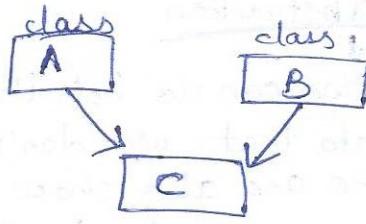
↳ It is a blueprint of class.

[Multi 800] Class (blueprint of object)

↑ ↓
Car Interfaces (blueprint of class)
[Wheels speed engine]

1. Multiple Inheritance (5th) type of inheritance,

- ↳ by default done in C++.
- ↳ but we need to do it in Java.



ex -

class Bear implements Herbivore, Carnivore { }

2. Total Abstraction

- ↳ 100% abstraction we get
- ↳ Not like abstract classes ~~which~~ in which it is needed to make each function as abstract function to get 100% abstraction.

Interface

class

- | | |
|---------------------------|------------------------|
| 1) Keyword - interface | 1) keyword - class |
| 2) extension - implements | 2) extension - extends |

Properties

- ① All methods are public, abstract and without implementation.
- ② Used to achieve total abstraction.
- ③ Variables in Interface are final, public, static
↓
value will be same

Note - If it is like this

class vehicle();
class car extends vehicle();

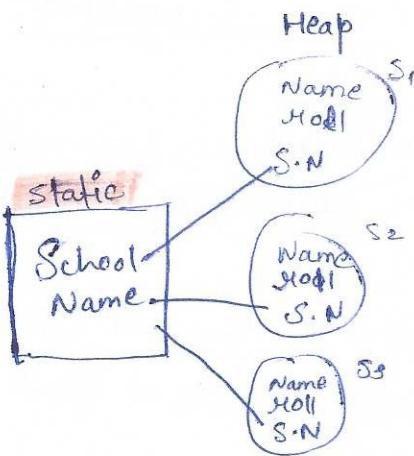
Then we can also write like this

Vehicle v = ~~new vehicle();~~ new car();

Vehicle v = new car();

Static Keyword →

static keyword in java used to share the same variable or method of a given class



- ↳ we can make static
 - Properties
 - Functions
 - Blocks
 - Nested Classes
- ↳ basically anything we can make static

↳ If any object declare school name means initialise it then it will be initialised for all objects

↳ If any object modify school name then it will be modified for all objects

* Our main function is also a static.

↳ Using static keyword multiple things of same name are not created. (No multiple creation)

↳ Saves memory

Super. Keyword →

~~super~~ keyword is used to refer immediate parent class object

- ① to access parents properties
- ② to access parents functions
- ③ to access parents constructor.

In function overriding Derive class function will be output.

We can access static variables directly with classes also.
ex- class Book{
 static int count;
}

We can write like this
→ `System.out.println(Book.count);` ✓
without making object.



type of access method.

a) less effective at same effective

While method / function overloading we can use

Recursion → It is a method of solving a computational problem where the solution depends on solutions to smaller instances of the same problem.

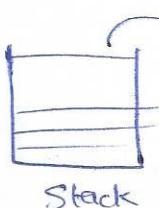
- Base Case → Where recursion ends.
- Recursive function → The functions that call themselves again and again.
 - 1) Top to base → calculate go down repeat
 - 2) base to top → add the solution and go up repeat
- a) Define base case
b) Work we want to do
c) (Inner call)

Print numbers from decreasing order

```
fun(int n){  
    sys0 Point(n);  
    fun(n-1);  
    if(n == 1){ sys0(n);  
        return; } }
```

Stack Overflow

- 1) Too many parameters ↑
- 2) Too many calls.



error →

Print numbers from n to 1 (Increasing Order)

function (int n)
 if ($n == 1$) {
 System.out.print (n);
 return;
 }
 function (n - 1);

Print factorial of number n.

int Fact (int n) {
 if ($n == 0$) {
 return 1; }
 here 0 because
 factorial of
 1 = 0
 int fact = $n * \text{Fact}(n - 1)$;
 return fact; }

Time Complexity $\rightarrow \Theta(n)$ because we done n calls

Space complexity $\rightarrow \Theta(n)$ because in worst case we store n stack cases.

print Sum of n natural numbers

Sum (int n) {
 if ($n == 1$) {
 return 1; }
 {

 int sum = $n + \text{Sum}(n - 1)$;
 return sum;
 }

}

Point N^{th} Fibonacci number

- ↳ Each next number is sum of last two numbers.
- ↳ Two base cases:

$\text{fib}(\text{int } n) \{$

if ($n=0 \rightarrow 0$)
 $n=1 \rightarrow 1$ } Base cases

$$\text{fib}_{n-1} = \text{fib}(n-1);$$

$$\text{fib}_{n-2} = \text{fib}(n-2);$$

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

return $\text{fib } N$

$$\text{fib } 5 = \text{fib } 4 + \text{fib } 3$$

$$\downarrow$$

$$\text{fib}_3 + \text{fib}_2$$

$$\downarrow$$

$$\text{fib}_2 + \text{fib}_1$$

$$\downarrow$$

$$\text{fib}_1 + \text{fib}_0$$

* Space complexity : $O(n)$ → no. of calls

* Time complexity : $O(2^n)$ → because ~~many~~ many leaf nodes were called.

Check if array is sorted or not.

ex - 1 2 3 4 5

check for

$$i > i-1$$

→ isSorted (int arr[], int i)

if { $i == \text{arr.length}$ } { }

return true;

}

~~arr[i] < arr[i+1]~~

if ($\text{arr}[i] > \text{arr}[i+1]$) { }

return false;

}

return isSorted (arr, i+1); }

i is the value from which we are starting to check is it sorted or not

First Occurrence \rightarrow

→ Here we need to return the position of key

ex -

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

key = 5

Output = 4.

if no key then -1.

→ base case = -1

if key found return i;

```
firstOccur (array, i, key) {  
    if (array[i] == key) {  
        return i;  
    }  
    return firstOccur (array, i+1, key);  
}
```

Last Occurrence \rightarrow

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

→ ① Look forward

② Then compare with yourself

→ If there is no key in forward then the current
i ~~is~~ will be last occurrence if it is ~~same~~ as key

```
lastOccurrence (array, key, i+1) {  
    if (array[i+1] == key) {  
        return i+1;  
    }  
    return lastOccurrence (array, key, i+1);  
}
```

```
int isFound = lastOccurrence (array, key, i+1);  
if (isFound == -1 || array[i] != key) {  
    return i;  
}  
return isFound;
```

Point x to the power n

Example - point x^n

$$2^{10} = 2 \times 2^{10-1}$$

\downarrow

$$2^{(10-1)} \times 2^{(10-2)} \dots$$

$\text{power}(x, n) \{$
if ($n == 1$) { return $x; }$

int value = $x + \text{power}(x, n-1)$;
return value;

base case \rightarrow if ($i == 1$) { return n }

Time complexity = $O(n)$

if ($i == \text{arr.length}$) {
return -1; }

key, $i+1$;

Point x to the power n (Optimized)

int Time complexity $\rightarrow O(\log n)$

① Case one \rightarrow when $n = \text{even}$

$$\text{ex} - 2^8 = 2^{10}$$

$$\Rightarrow (2^5) \times (2^5) = 2^{5+5} = 2^{10}$$

② Case two \rightarrow when $n = \text{odd}$.

$$\text{ex} - x^7 = 2^5$$

$$\Rightarrow x \times (2^2) \times (2^2) = 2^{1+2+2} = 2^5$$

$\text{power}(x, n) \{$

if ($n == 1$) { return $x; }$

int ans = $\text{power}(x, n/2)$;

int ansSq = ans * ans;

// both odd

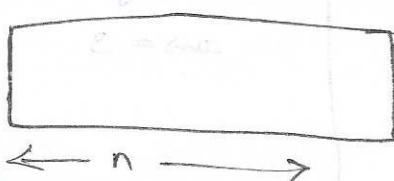
if ($n \% 2 != 0$) {

ansSq = $x * \text{ansSq};$

}

return ansSq.

Tiling Problem



breadth = 2

tiles = 2×1

Either horizontally
or vertically,
or using both



Q. Find total no. of ways to place tiles?

\Rightarrow we should know ① base case

② learn

③ call inner f(x)

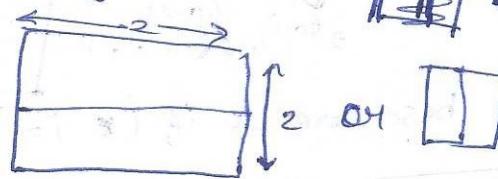
$n = \text{no. of files}$

① For $n=0$, 2×0 , ways = 1

② $n=1$, 2×1 = ways = 1

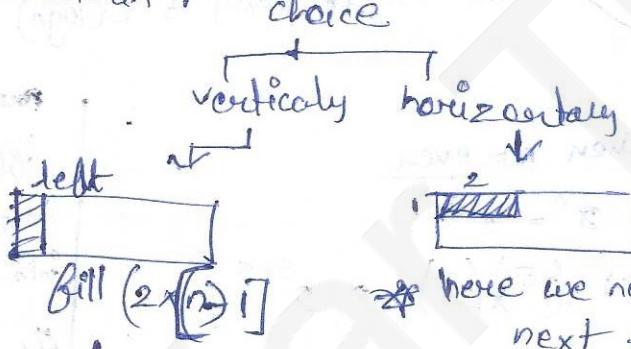
③ $n=2$, 2×2 = ways = 2

ways = 2.

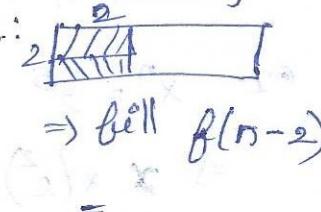


① # base case : if ($n=0$ || $n=1$) {

② Kam but for more than 1 choice



here we need to place next file horizontally only



③ Total ways = $f(n-1) + f(n-2)$

int tiling(int n) { floor length = n, breadth is 1

if ($n==0$ || $n==1$) {

return 1;

// vertical case

int fm1 = tiling(n-1);

// Horizontal case

int fm2 = tiling(n-2);

int NO_Ways = fm1 + fm2;
return NO_Ways; } }

For length = 3 ans = 3

①

②

③

Remove Duplicates in a String ; amazon

"appnacollege"

there are only lower case characters.

① Create new string string Builder sb

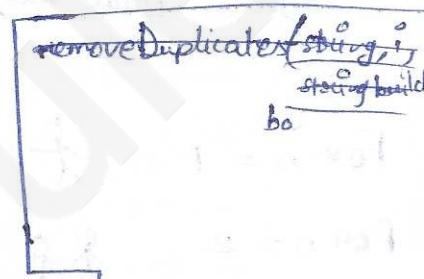
② Index variable

③ map (boolean) []

0	1	2	3	.	.	25
---	---	---	---	---	---	----

index = currentchar - 'a'
(map)

⇒ example 'z'-'a' = 25
 'b'-'a' = 1



if the character is their

map [i] = true;

 sb.append(i);

and if (map[i] = true) {
 skip off continue;
}

like this only unique characters will be their
in the new string.

① base if (i == string.length) {
 return sb.toString(); }

② kam → is the char is present or not
 ✓ → newStrX
 X → add newStrY

③ index → index + 1;

function(stairg start, StairBucker newStair, boolean map[][])

Friends Pairing Problem:

Goldman Sachs

We have n friends. A person can remain single or make a pair.

Find total ways in which they can be

⇒ total ways → Standing insingle + in pair.

For $n = 1$ ♂ (single) 1 way

For $n = 2$ ♂♂ either a, b
 single] - 2 ways
 or pair (a, b)

For $n = 3$ ♂♂♂ (a, b), c ↗
 a, (b, c) ↗ 4 ways
 (a, c), b
 a, b, c.

choose ↗
 ↓ ↗ Pair because it can make pair
 Single ↗ with any member
 f(n-2) = (n-1) * f(n-2)

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

$$\text{total ways} = f(n-1) + (n-1) * f(n-2);$$

$\Rightarrow \text{Friend}(\text{int } n) \{$ $n = \text{no. of people}$

// base case

if ($n = 2 \text{ || } n = 1$) {

return n; }

// choice

// single

int fnm1 = Friend(n-1);

// pair

int fnm2 = Friend(n-2);

int pairways = (n-1) * fnm2;

return fnm1 + ~~pairways~~ pairways;

}

Binary String Problem

Output

Print all binary strings of size N without consecutive
one. that is, empty.

~~11~~

be pair
number

Base case if ($n=0$) \rightarrow "-"

if ($n=1$) \rightarrow "1", "0"

if ($n=2$) \rightarrow "00"

"01"

"10"



if ($n=3$)

$l_p = \text{last place}$

\rightarrow	$\begin{array}{ccc} - & - & - \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{array}$	$l_p = 0$	"000"	5 total ways.
		$l_p = 1$	"001"	
		$l_p = 1$	"010"	
			"101"	
			"100"	

PrintBinString(int n, int lastPlace, string str){

// base case

if (n == 0) {

System.out.print(str);
return;

// Kaam

PrintBinString(n-1, 0, str + "0");

if (lastPlace == 0) {

PrintBinString(n-1, 1, str + "1");

}

Divide and Conquer

algorithm

* # Merge Sort // Depth first

→ time complexity - $n(\log n)$ like inbuilt sort

Approach

① to divide

↳ mid

to find out mid

mid \rightarrow

$$\text{Starting index} + \frac{(\text{Ending index} - \text{Starting index})}{2}$$

$$\rightarrow S_i^0 + \frac{(E_i - S_i)}{2}$$

② mergeSort (left)

mergeSort (right)

$S_i = 0$

6

3

9

5

2

8

$E_i = 5$

$E_i = \text{mid}$

$E_i = 2$

$E_i = 5$

<

function MergeSort(int arr[], int si^o, int ci^o)

//base case
if (si^o = ci^o) {
 return;
}

dividing
each
element

// Kaam

int mid = si^o + (ci^o - si^o) / 2; // or you can use $\frac{si^o + ci^o}{2}$

MergeSort(arr, si^o, mid); // passed ci^o as mid

MergeSort(arr, ~~right~~, ci^o); // passed si^o as mid + 1

// Another function for merging them

merge [arr, si^o, mid, ci^o]

}

Public static void merge (int arr[], int si^o, int mid, int ci^o)

// creating temporary for saving array
int temp[] = new int [ci^o - si^o + 1];

int i = si^o;

int j = mid + 1;

int k = 0; // iterator for temp
while (i <= mid && j <= ci^o) {

 if (arr[i] > arr[j]) {

 temp[k] = arr[j]; // add smaller
 j++;

 } else {
 temp[k] = arr[i];
 i++;

 k++;

 merging elements
 while (i <= mid) {

 temp[k+i] = arr[i++];
 }

 while (j <= ci^o) {

 temp[k+j] = arr[j++];
 }

 considered good

time complexity = $O(n \log n)$

✓ & in sorting

Space complexity = $O(n)$

∴ should not be used where
space is left.

Quick Sort

Time complexity \rightarrow Average cases $\rightarrow O(n \log n)$

Worst case $\rightarrow O(n^2)$

Space complexity $\rightarrow O(1)$

less than merge sort.

=> Approach

↳ Pivot and partition

① choose a pivot (the point around which we will do sorting)

but naturally we take pivot = last element

② Partition (parts) around pivot

a) will take pivot element as mid

and then sort the small element in front of pivot

b) and big elements behind the pivot
(bigger than pivot)

③ Quick Sort (left part)

(right part)

At base case return;

Partition

here we take $i = -1$
the if we found a element smaller the pivot element
we $i++$

② then swap the element at i and the larger element position

③ if not we do nothing and move forward

④ at last $i++$ and bring pivot at i and to the pivot

Quick sort's Works Time complexity

↳ When pivot is always smallest or the largest element

For example

When we have ~~a sorted array~~ a sorted array.

Sorted and Rotated Array Search

* medium level
* important
* interview

(at pivot)

(no duplicates)

Q. We have sorted and rotated array with distinct numbers

Find the index of given element, key = 0

4	5	6	7	0	1	2
---	---	---	---	---	---	---

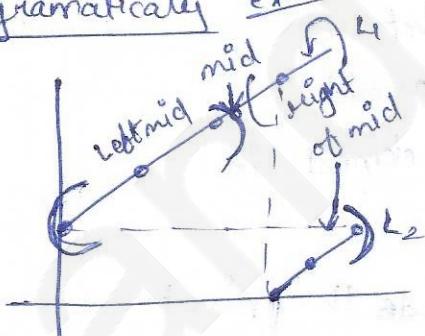
here we can see pivot

is now 1

→ to solve this we will use divide and conquer.

→ Modified binary search.

• Diagrammatically example



Case 1: mid on L1

Step 1. we will find mid
 $mid = s^{\circ} + \frac{(e^{\circ} - s^{\circ})}{2}$

Case 2: mid on L2

case a: we only need to search

arr [s^{\circ}] < [mid] range

L1 left

$s^{\circ} < key < mid$

Case b:

else
right of mid.

Case c: mid left

Case d: mid right

Initial step: find position

mid = arr[0] to arr[n-1]

mid = arr[0] to arr[n-1]

Time and Space Complexity

↳ These are two parameters to judge or compare same programs.

Time Complexity

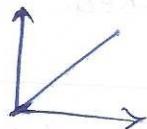
↳ Amount of time taken up by an algorithm/code as a function of input size.

Not the actual time taken,

time \propto Input size

- linear Search

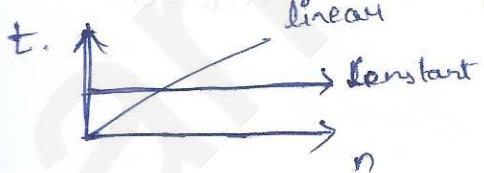
$$\text{time complexity} = O(n)$$



• In sorted array time complexity is constant

∴ time complexity = $O(1)$

Constant



because same for every program

No loop
No function.

Big O notation \rightarrow upper bound $O()$

↳ this the worst case time complexity

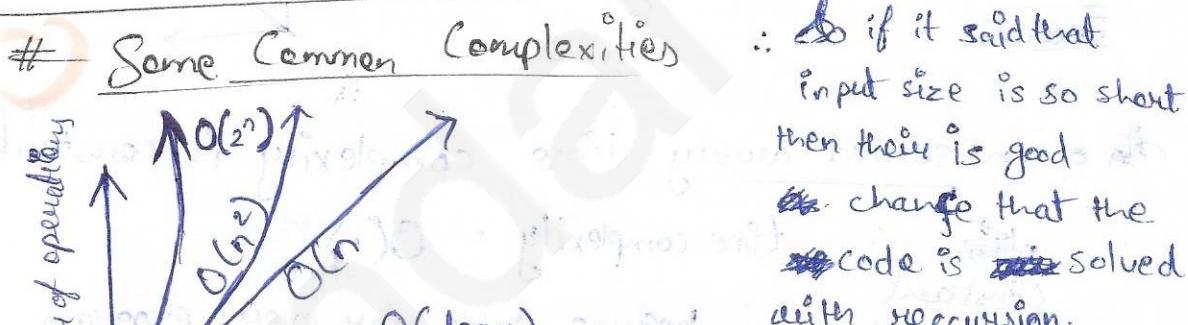
↳ program may ~~not~~ run faster than that
but not slower than that.

Big $\Omega()$ → ↳ Best or ~~the~~ lowest time complexity
Big omega ↳ lower bound.
 ↳ Program cannot be faster than this time.

Big $\Theta()$ → Average time complexity
Big Theta

Small 'O' of time complexity are the time complexities that are loosely packed.

↳ Any Big 'O' of time complexities are tightly packed.



↳ if it said that input size is so short then there is good chance that the code is solved with recursion.

↳ $O(2^n)$ is the worst time complexity and we never use this in our code written in interviews.

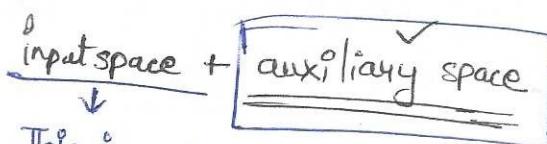
↳ If comes when we use recursion with ~~memo~~ optimization.

↳ This time complexity is only applicable for small input size.

Space Complexity → Memory Space

Heap(1)

Stack(2)



most of the time we are talking about this

For ex- If we have two

Merge sort

↑ Space comple

↓ Time comple

Quick sort

↓ Space comple

↑ Time comple

But we always give the priority to time complexity most of the time.

Theoretical Analysis

① Simple for loop

for(int i=0; i<n; i++) {
 ↓
 } (K)

Time does not depend
on the code inside the loop

$$\Rightarrow \text{Time complexity} = O(n \times k)$$

but ignore the constant

$$\text{new of above} \quad O(n)$$

② Nested Loop 1

for(i=0; i<n) {
 ↓
 } (n)

for(j=i+1; j<n) { }
 ↓
 } (n-i)

$$\frac{n(n+1)}{2}$$

$$\Rightarrow O\left(\frac{n^2}{2} - \frac{n}{2}\right)$$

ignoring constant and taking larger no.

$$= O(n^2)$$

here sometimes the inner loop work but some times not

by a pattern we get $(n-1) + (n-2) + \dots + 1$ (P)
and this is an A.P. ∴

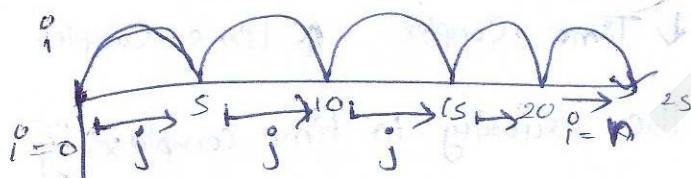
nested loop 2

$$T.C = O(n^2)$$

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

nested loop 3

```
for (int i=0; i<n; i=i+k) {  
    for (int j=i+1; j<=k; j++) {  
        }
```



as j take the longer path \therefore time complexity = $O(n)$

Sorting

a) Bubble sort.

a) Worst case (array will be in reverse sorted)

$$T.C = O(n^2)$$

b) Best case \rightarrow here also T.C is $O(n^2)$

because the code needs to run

the inner loop.

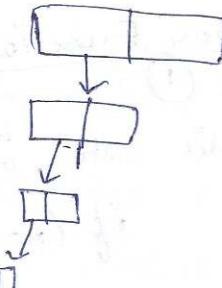
$$T.C = O(n^2)$$

\hookrightarrow to make it fast you need to write a optimised sort.

And optimised sort will give us less time complexity

Binary Search

↳ As it gets divide is two
and then again divide int two



$$\Rightarrow T.C = \log(n)$$

Recursive Algorithms

* Factorial code →

$$n+k \Rightarrow T.C = O(n)$$

$$S.C = \text{max depth} * \underbrace{\text{each memory level occupied}}$$

$$S.C = O(n)$$

* Sum of n

$$T.C = O(n)$$

$$S.C = O(n)$$

* Fibonacci

↳ ~~here~~ divide and conquer is used
~~max~~ $\Rightarrow T.C = 2^n$

$$S.C = O(n)$$

* Merge Sort

↳ (1) For merge function $T.C = O(n)$
(2) ~~ditto~~

$$T.C = n \times \log n$$

$$S.C = O(n)$$

because of temp array,

→ Power Function (Recursion)

Normal: ①

```
public static int power(int a, int n) {
    if (n == 0) {
        return 1;
    }
}
```

$$T.C = O(n)$$

$$S.C = O(1)$$

return $a * \text{Power}(a, n-1);$

}

[Opt]

optimized.

②

```
public static int power2(int a, int n) {
    if (n == 0) {
        return 1;
    }
    int halfPowerSq = power2(a, n/2) * power2(a, n/2);
}
```

$$T.C = O(n)$$

if ($n \% 2 != 0$) { // a is odd
 return $a * \text{halfPowerSq};$
}

more optimized

③

```
public static int power3(int a, int n) {
    if (n == 0) {
        return 1;
    }
}
```

$$T.C = O(\log n)$$

$$S.C = O(1)$$

int halfPower = power3(a, n/2);
 int halfPowerSq = halfPower * halfPower;

if ($n \% 2 != 0$) { // a is odd

return $a * \text{halfPowerSq};$

}

return halfPowerSq;

How to approach Question ?

① Brute force (logical)

↓
optimize

B.F

↔ linear search
↔ Binary search
if sorted direct.

→ in 1 sec 10⁸ operations can be performed.

- watch input size for analysing time complexity.

~~Backtracking~~

Backtracking

↳ Backtracking happens only after function call.

Types

- ① Decision (solution exist or not)
- ② Optimization (best solution)
- ③ Enumeration (listing all possible solutions)

Backtracking of an Array

↳ First we will make an array-

1	2	3	4	5
0	1	2	3	4

if ($i = n-1$) { back track

and decrease each value by -1

code
public change(int arr[], int i, int n){

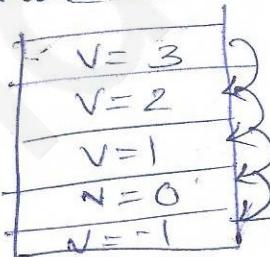
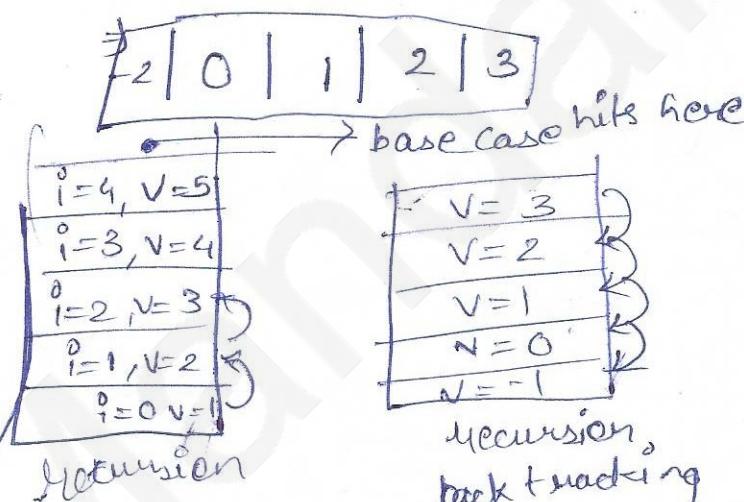
// base case

if ($i = arr.length$) {
printArr(arr);
return;

}

// recursion

arr[i] = val;
change(arr, i+1, val+1);
arr[i] = arr[i]-1;



Time Complexity

$\Rightarrow O(n)$

Find Subsets

→ Q. Find and print all subsets of a given string.
"abc"

a, b, c, ab, bc, ac, abc, " "
1 2 3 4 5 6 7

↓
null set

8 subsets
 $n = \text{no. of elements}$
Total Subsets = 2^n

Sometimes it doesn't work instead of this
use $i+1$

⇒ "abc" is a string
① we will start with a
each character has its choice to come in subset or not
⇒ $\begin{matrix} a \\ b \\ c \end{matrix}$

↙ ↘ ↗
bc ac ab

↙ ↗
bc ac ab

↙ ↗
bc ac ab

↙ ↗
empty

if (int arr[I], int i, int val)

$$2^n = 2^3 = 8$$

⇒ abc, ab, ac, a, bc, b, c, empty.

• Time Complexity

$$\Theta(n \times 2^n)$$

• Space Complexity

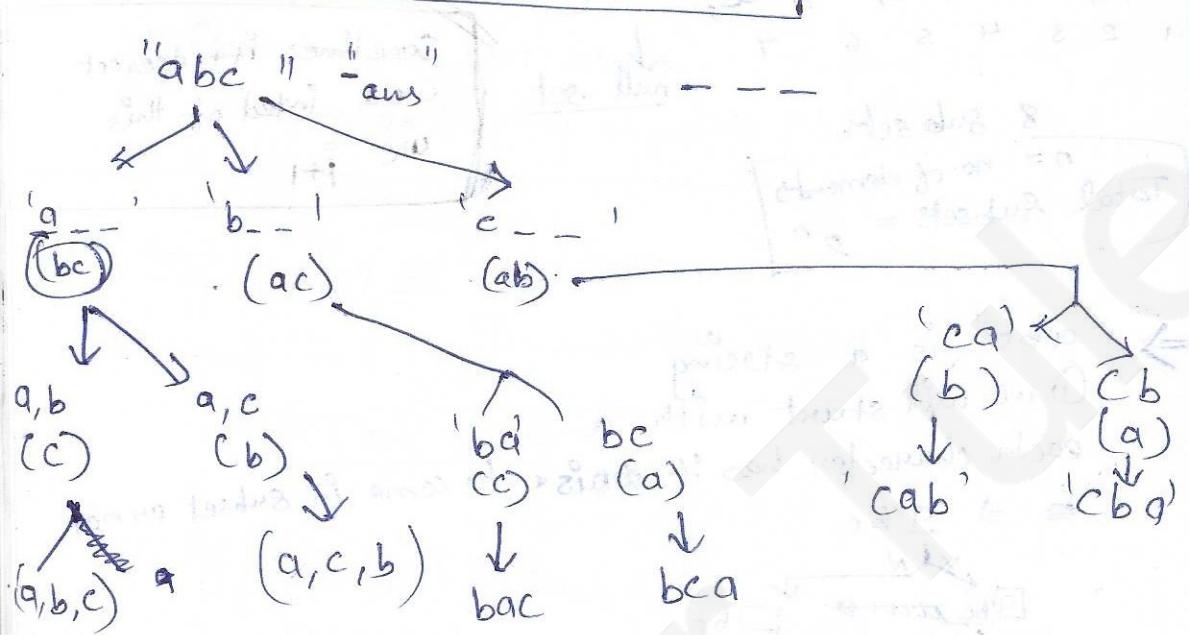
$$\Theta(n)$$

→ If we solved this using String Builder then we also need to add a step to delete the element from the string when we come back.

Find Permutations

Q. Find and print all the permutations of a string?

$$\text{No. of permutation} = \frac{\text{size of string}!}{}$$



abc, acb, bac, bca, cab, cba

$$3! = 6$$

$$T.C = O(n \times n!)$$

N-Queens

* Place N queens on $N \times N$ chess board, such that
"no 2 queens can attack each other"

In this questions three thing is usually asked

- 1) all solutions
- 2) Solution exists or not
- 3) Count all solutions.

• Conditions need to check. [for safe]

- ① Is an queen above
- ② Is a queen in above left diagonal
- ③ Is a queen in above right diagonal

Time Complexity : As the first

Queen has n , other queen $(n-1)$
like this

$$n(n-1)(n-2)(n-3)$$

$$\Rightarrow n!$$

$$T.C = O(n!)$$

#) N-Queens - Count ways.

⇒ [in base case just make the
count ++]

↳ Make a static variable because in recursion
in each level a new parameter is formed

#) Check if there is a solⁿ and print one solution.

⇒ Just make the pattern function as boolean
and if true then print pattern

Grid Ways

Q. Find the number of ways to reach from $(0,0)$ to
 $(N-1, M-1)$ in a $N \times M$ grid.

↳ Allowed moves → i) right
ii) down

$$\text{Time complexity} = O(n+m)$$

$$\text{Time complexity} = O(m)$$

So how do we do it?



$$\frac{i(i-1)}{2} + \frac{(m-i)(m-i-1)}{2} = \frac{i(i-1) + (m-i)(m-i-1)}{2}$$

Math trick for linear time good ways

$$\text{Time complexity: } O(2n+m)$$

and then for that new grid we need to find the ways
and one step right or down we get a new grid
we move one step right or down we get a new grid as

$$\text{Total ways} = \frac{i(i-1)}{2} + \frac{(m-i)(m-i-1)}{2}$$

$$\begin{aligned} & \text{Total ways} = \\ & \quad \text{down ways} + \text{right ways} \\ & \quad \text{down ways} = f(x,y+1) \\ & \quad \text{right ways} = f(x+1,y) \end{aligned}$$

$$\text{Source} = \frac{(n-1)!}{m!}$$

if we are looking

$$f(x,y+1)$$

$$f(x,y+1) = f(x,y) + f(x,y+1)$$

	\leftarrow	\uparrow	\rightarrow	\downarrow
	A	0,1	1,0	1,1
	1	2,1	3,0	3,1
	2	3,2	4,1	4,2
	3	4,3	5,2	5,3
	4	5,4	6,3	6,4
	5	6,5	7,4	7,5
	6	7,6	8,5	8,6
	7	8,7	9,6	9,7
	8	9,8	10,7	10,8
E	\leftarrow	\uparrow	\rightarrow	\downarrow

one way that's down

one way up

one way left

Sudoku

→ We are just solving the sudoku

→ For that we need to check the following condition

- i) If that number vertical same
- ii) If that number duplicate present horizontally
- iii) Is the number inside the small grid.

↳ Duplicate present in?

If not then point that number.] From 1-9.
If yes then try another number]

- ## # ArrayList
- ↳ in C++ this topic is Vectors.
 - ↳ It is a part of java collection framework.
 - ↳ pre defined data structure in java (inbuilt)
 - ↳ Linear storing data

Advantages than array

- ① ↳ dynamic size (can be changed any time)
 - ② → Only non primitive data types can be stored
- ~~int~~ → Integer ✓
class

Syntax

```
import.util.ArrayList; or util.*;
```

```
public class Classroom {
```

```
ArrayList<type> name = new ArrayList<>();
```

ArrayList<type> name = new ArrayList<>();

in type classes are used

example → < Integer >
< String >
< Boolean >

ex - ~~ArrayList~~

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

Operations on an ArrayList

- ① Add a element.

object.add(element);

Time - $O(1)$

list.add(1);
list.add(2);
System.out.println(list);

Output
[1, 2]

System.out.println(list);

ex -

⑥. Ac

② Get Element Time: O(1)

`list.get(index)`

→ `list.get(0);`

ex- `int a = list.get(0);`
`System.out.println(a);` Output 1

③ Remove Element Time: O(n)

`list.remove(index);`

`list.remove(2);` Output

`System.out.println(list);` $\Rightarrow [1, 2, 4, 5]$

ex `list = [1, 2, 3, 4, 5]`

`index - 0 1 2 3 4`

④ Set Element at Index

Time: O(6)

`list.set(index, element)`
You want to store

Output:

$[1, 2, 10, 4, 5]$

remaining element moved forward.

`list.set(2, 10);`

`System.out.println(list);`

⑤ Contains Element Time: O(n)

↳ To check the element is present in the array or not
 If yes then it will return true if not then it will return false

`list.contains(element);`

Output:

True

ex- `System.out.println(list.contains(2));`

Time = O(n)

⑥ Add a element on specific index



`list.add(index, element)`

ex- `list.add(1, 9);`

`System.out.println(list);`

Output

$[1, 9, 2, 3, 4, 5]$

Size of Array list

~~Ex-~~

`list.size()`

~~ex-~~

`sys.out.println(list.size());`

Output:

5

Print reverse of array list

```
for(i = list.size() - 1; i >= 0; i--) {  
    System.out.println(list.get(i));  
}
```

Find Maximum

```
int max = Integer.MIN_VALUE;  
max = Math.max(max, get(i));
```

Swap 2 Members

```
swap(ArrayList<Integer> list, int idx1, int idx2) {  
    int temp = list.get(idx2);  
    list.set(idx1, list.get(idx2));  
    list.set(idx2, temp); }
```

Sorting an Array List

inbuilt method = `Collections.sort(name);`

according order sort.

Sort in descending

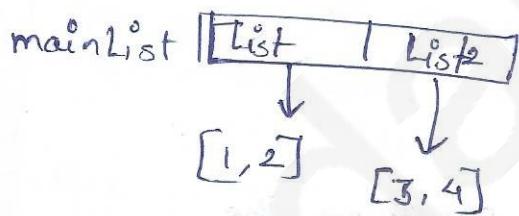
`Collections.sort(list, Collections.reverseOrder());`

comparator for defining the logic of sort

Multidimensional ArrayList

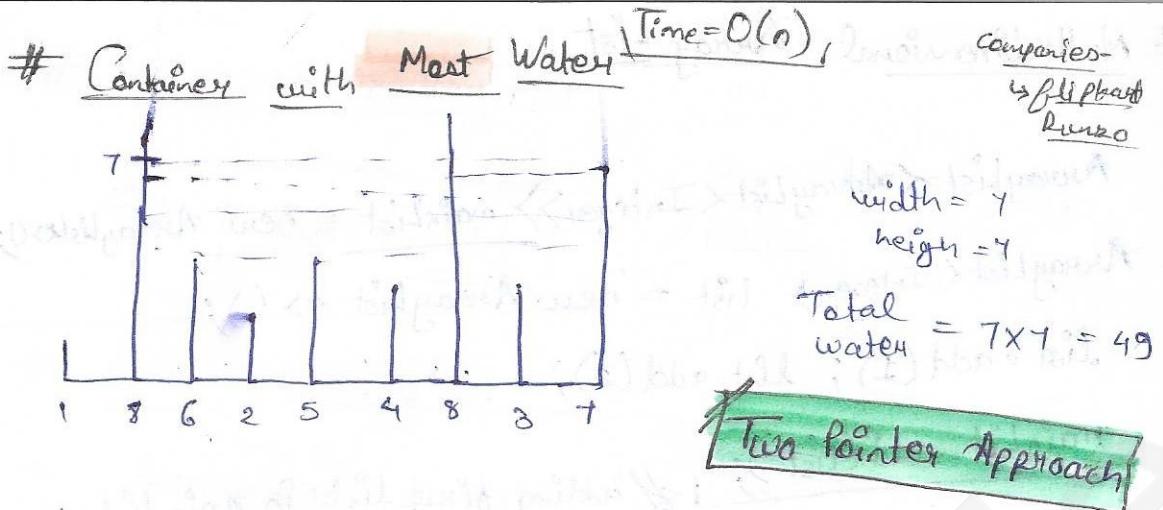
ArrayList<ArrayList<Integer>> mainList = new ArrayList<>();

- ① ArrayList<Integer> list = new ArrayList<>();
list.add(1); list.add(2);
mainList.add(list); // adding first list in main list
- ② ArrayList<Integer> list2 = new ArrayList<>();
list2.add(3);
list2.add(4);
mainList.add(list2); // adding second list in main list.



Printing them

```
→ for (int i; i < mainList.size(); i++) {  
    ArrayList<Integer> curr = mainList(i);  
    for (j=0; j < curr.size(); j++) {  
        System.out.println(curr.get(j));  
    }  
    System.out.println();
```



• Brute Force - Time - $O(n^2)$

↳ make a pair of every two lines and see which two lines can store max water

↳ water \rightarrow container \times width
height

$\text{container} = \text{height of small side}$ $\text{width} = \frac{\text{last} - \text{first}}{2}$

• Optimized Solution - Time : $O(n)$

2 pointer Approach.

↳ Water capacity of the contained depends on the

- ① Width
- ② Height.

↳ if we move the bar that have long height then capacity of the container will only decrease.
so we move the bar that have smaller height.

approach • $i = \text{index of Left Pointer}$
while ($i < j$) $j = \text{index of Right Pointer}$

$$\text{curr water} = ht \times \text{width}$$

$$ht = \min(L.P \text{ height}, R.P \text{ height})$$

$$\text{width} = j - i$$

if ($L.P \text{ height} < R.P \text{ height}$)

{ L.P++ } else {

R.P++ }

Optim
approac

T.C = O(n)

→

↳ we
we

- # How trapping rain water and container is different?
- here there are many boundaries
 - Water we want that total stored in all containers.
 - Here it has only one container.
 - Whatever we want that container has maximum capacity.
 - here we have asked the capacity of max container.

Pair Sum → we have given, Sorted ArrayList.

↳ find if any pair in the arraylist have targeted sum?

Optimized approach:

[1, 2, 3, 4, 5, 6]

↑ ↑
L.P R.P

Two Pointer Approach

Case 1 = if ($(L.P + number \text{ on } (R.P)) == target$) { return true; }

Case 2 = if ($(number \text{ on } (L.P) + number \text{ on } (R.P)) < target$) { L.P++ }

Case 3 = if ($(number \text{ on } (L.P) + number \text{ on } (R.P)) > target$) { R.P-- }

→ Because we have sorted list.

Time Complexity: $O(n)$

Pair Sum - 2

→ we have sorted and Rotated list

T.C = $O(n)$

Q. Find pair sum?

target = 16

→ [11, 15, 6, 8, 9, 10], (breaking point)

↳ We will solve this with two pointer approach

We will take $L.P = \text{breaking point} + 1$] → then same approach.
 $R.P = \text{breaking point}$]

Module A arithmetic Properties

↳ This have many arithmetic rules or formulas which make our algorithm or logic so simple.

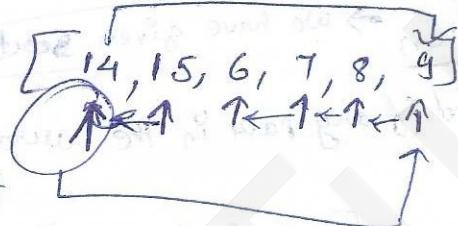
ex- if we have a rotated sorted array.

And we want to make pointers

and we want

that when the pointer hits the last ~~element~~

iterator it should go to the or ~~or~~ jump to the last iterator.



for moving forward.

→ Property →

$$\text{left} = (\text{left} + 1) \% n$$

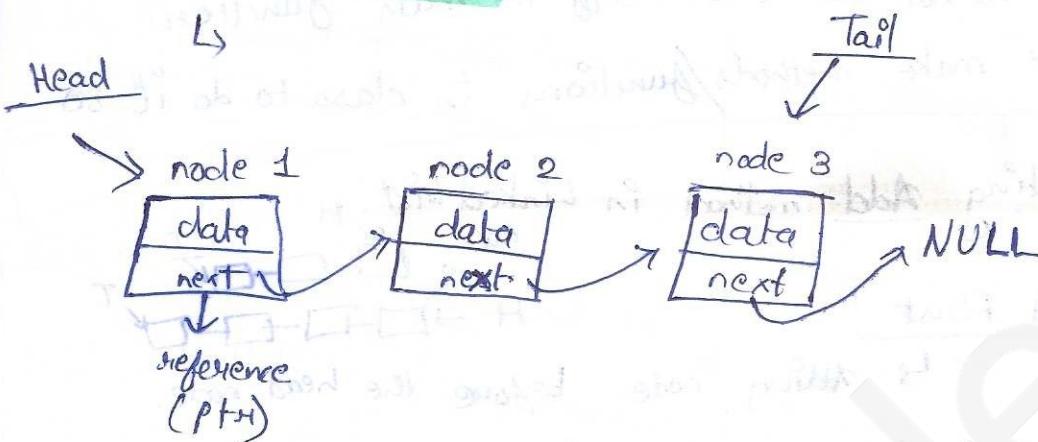
$$\text{right} = (\text{right} + \text{right} - 1) \% n$$

for moving backwards

* used this

④ Pair Sum in Sorted Rotated Array Question

Linked List



As each **node** stores same type of **data** means we know the **blueprint** of **node**, so we will make a class and each node will be an object.

Making node class.

```
Class Node {  
    int data;  
    Node next;  
}  
public Node (int data) { //constructor  
    this.data = data;  
    this.next = null;  
}
```

Head Node → First node

Tail Node → Last valid node
(Not the null one)

- Head Node = Tail Node if there is only one node in list.

- To perform all the operations in a linked list we do not do it ourself in main function.

→ We make methods/functions in class to do it so

Creating Add method in Linked List

① Add First

↳ Adding node before the head node.

- Step 1 - Create new node (Creating a node named Newnode)
- Step 2 - new node's next = head (Making it as head)
- Step 3 - head = new Node. (Transferring its data from head to Newnode)

Code -

Inside the Node class

```
public void addFirst(int data){
```

T.C = O(1)

// step1 = create new node

Node newNode = new Node(data)

if (head == null) {

head = tail = newNode;

return;

}

because
we have
a ~~single~~ node
in list.
empty list

// step2 - newNode.next = head

~~newNode~~.next = head;

// step3 - head = newNode

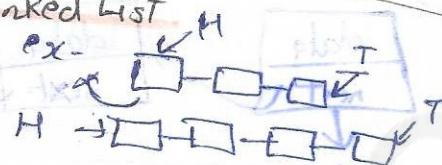
head = newNode;

}

(solution of part 1)

problem no 3 with first solution

Code



Creating Add Last method.

Step ①

create a node (newNode)

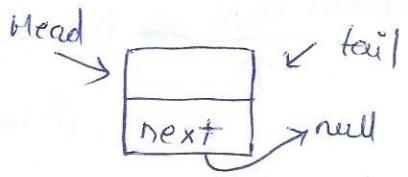
Step ②

tail.next = newNode

③

tail = newNode

If there was a null
list then



Code

```
public void addLast(int data) {
```

```
    Node newNode = new Node(data);
```

```
    if (head == null) {
```

```
        head = tail = newNode;
```

return;

```
    tail.next = newNode; // step 2
```

```
    tail = newNode; // step 3
```

Printing Linked List:

→

```
while (temp != null) {
```

→ If (head == null) {
 set(Empty);
 return;

```
    print(temp.data);
```

```
    temp = next temp.next;
```

Add a node in Middle

```
Node temp = head
```

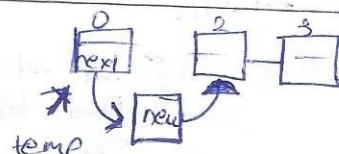
```
while (i < idx - 1) {
```

```
    temp = temp.next;
```

i++

}

For finding the $idx - 1$
position



Step 1: newNode.next = temp.next;

Step 2: temp.next = newNode.

Exception

```
if (index == 0) {  
    addFirst(data);  
    return;  
}
```

Size of linked List

```
class LinkedList {  
    public static int size;
```

↑ because it will change everywhere.

→ add size ++ in addFirst

addLast

addMiddle

time =

Remove First Node

↳ We just change the head and make the head to the second node. And when the garbage collector will come as the code runs it will remove the first node thinking it as garbage.

```
public int removeFirst() {
```

```
    int val = head.data;  
    size--;
```

```
    head = head.next;
```

```
    return val;
```

Exception

① When our LL is empty

if (size == 0) {

throw (LinkedListEmpty);

}

② If (size == 0) {

int val = head.data;

~~head = null;~~

~~tail = null;~~

head = tail = null;

return val;

size = 0

}

Remove Last Node

```
public int removeLast() {
```

```
    if (size == 0) {  
        throw ("LL is empty");  
    return;
```

```
    if (size == 1) {  
        int val = head.data;  
        head = tail = null;  
        size = 0; return val;
```

```
    Node prev = head;
```

```
    for (int i=0; i<size-1; i++) {
```

```
        prev = prev.next;
```

```
    prev.next = null;
```

```
    tail = prev;
```

```
    size--;
```

```
    return val;
```

Iterative Search in a Linked List

↳ search a key and return its index; if not found return -1;
int i=0;
Node temp = head
while (temp != null) {

time = O(n)

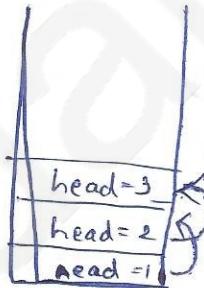
if (temp.data == key) {
 temp = temp.next;
 i++;
} else

return -1;

Recursive Search

↳ Find the index of key? use recursion?

Base case



we will ask in each time
is our key == head
if not ~~not~~ the put
~~value again~~ value again in the
function.

And if key == head return
return i+1;

public int recSearch(int key) {
 return helper(head, key);

public int helper(Node head, int key) {
 if (head == null) {

sys("It is empty")
 return -1;

if (head.data == key) { // base case
 return 0;

int idx = helper(head.next, key);
 if (idx == -1) { return -1; }

return idx + 1;

variables 4 steps

Reverse a Linked List

(Iterative approach)

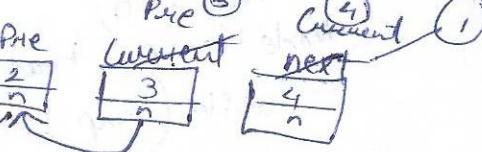
Approach

Time = $O(n)$

- ① $\text{next} = \text{current.next}$
- ② $\text{current.next} = \text{prev}$
- ③ ~~$\text{prev} \leftarrow \text{current}$~~
- ④ $\text{next} = \text{current}$



while ($\text{current} \neq \text{null}$)



(a) $O = 3^{\text{rd}}$

Code: public void reverse() {

Node prev = null; // because there is no node before head.

~~Node curr = tail; tail = null;~~

Node curr = tail = head; // As our new tail will be our head

while ($\text{curr} \neq \text{null}$) {

 next = current.next;

 current.next = prev;

 prev = curr;

 curr = next;

}

head = prev;

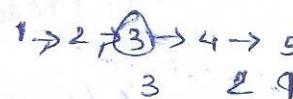
}

amazon
adobe
outlook

Q. # Find & Remove N^{th} node from End

(Iterative Approach)

Ex - $n = 3$



So we cannot go backward but there is a simplest approach that

n^{th} node from backward is

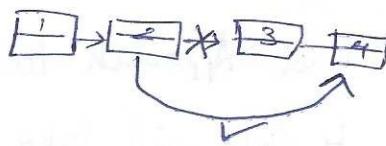
$(\text{Size} - n + 1)^{th}$ node

Ex $n = 3$

$= 5 - 3 + 1 = 3^{\text{rd}}$ node.

null

And to remove a node just change the pointer of the previous node by pointing it towards the next node of n^{th} node.



Code :

```
public void removeFromnth() {
```

```
// to calculate node size  
int sz = 0;  
Node temp = head;  
while (temp != null) {  
    temp = temp.next;  
    sz++;  
}
```

~~// if we have reached the node that we want to remove~~

if (n == sz) {
 head = head.next;
 return;
}

Corner Case

```
// sz - n
```

```
int i = 1;  
int iToFind = sz - n;  
Node prev = head;  
while (i < iToFind) {  
    prev = prev.next;  
    i++;  
}
```

```
prev.next = prev.next.next;  
return;
```

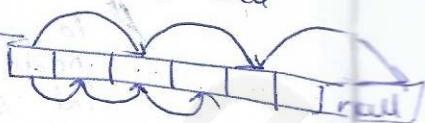
```
}
```

test

Check if L.L is a palindrome

Slow-Fast Approach to find mid]

↳ slow will take one step at a time and fast will take two step at a time
the fast reaches at the end the slow will be at middle.



(1) for even case when

fast will be at null

slow will be at mid

(2) for odd case when

fast will be at fast.next=null

Slow = mid,

Palindrome

- ① find middle
- ② 2nd half reverse
- ③ check if 1st half == 2nd half

* Corner Case/Exception

↳ if size of L.L is odd then mid will be

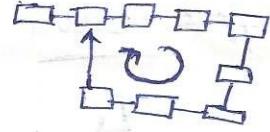
slow.next

$\{ \text{if } (\text{fast} != \text{null}) \}$ //odd case

$\text{mid} = \text{slow.next}; \}$

App

Detect a Loop/Cycle in a L.L



So we use Floyd's Cycle Finding Algorithm

Slow Fast Approach

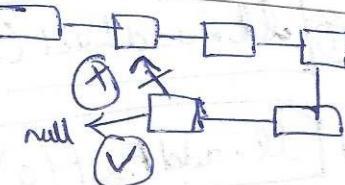
→ We will run slow and fast and if the

$\{ \text{if } (\text{slow} = \text{fast}) \}$ // If it is loop then it will happen once
{ then the loop exists};

$\text{slow} = \text{head}$
 $\text{fast} = \text{head}$
 $\text{while } (\text{fast} \neq \text{null} \& \text{fast.next} \neq \text{null})$
{
 $\{ \text{slow} = \text{slow.next},$
 $\text{fast} = \text{fast.next};$
}
 $\{ \text{if } (\text{slow} = \text{fast})$
 $\{ \text{data} = \text{slow.data},$
 $\text{return } \text{t};$
}
}
}
}
}

Remove the Cycle code

- ① find the last node
- ② point it towards null



Approach

- ① detect cycle $\text{Node prev} = \text{null}$

- ② after detecting make again

$\text{slow} = \text{head}$
then $\text{slow}++;$
 $\text{fast}++;$

while ($\text{slow} == \text{fast}$)
 $\text{prev} = \text{fast};$
 $\text{slow}++;$
 $\text{fast}++;$
③ the make the end point towards null
④ and point prev towards the fast.next by slow.

And the node where the slow and fast meet again is the point of end.

Java Collection Framework

- ↳ ArrayList
- ↳ LinkedList
- ↳ Stack
- ↳ Queue
- ↳ HashSet
- ↳ HashMap

→ We learn to create a data structure from scratch because interviewer can ask us to make data structure from scratch.
 → In C.P. ~~we will use~~ we will use inbuilt data structures.

Linked List in Java frame work

- ↳ Linked list is same as array list ~~As it is a class we have to make objects in it like~~ with using <Integer>, <Float>, <Boolean> classes as my data types.

Creating Linked List

```
datatype      name
LinkedList < Integer > ll = new LinkedList<>();
```

① `ll.addLast(1)` ← Adding elements in last.

② `ll.addFirst(0)` ← Adding elements in starting

③ `ll.removeFirst();` ← removes element from first

④ `ll.removeLast();` ← removes element from last.