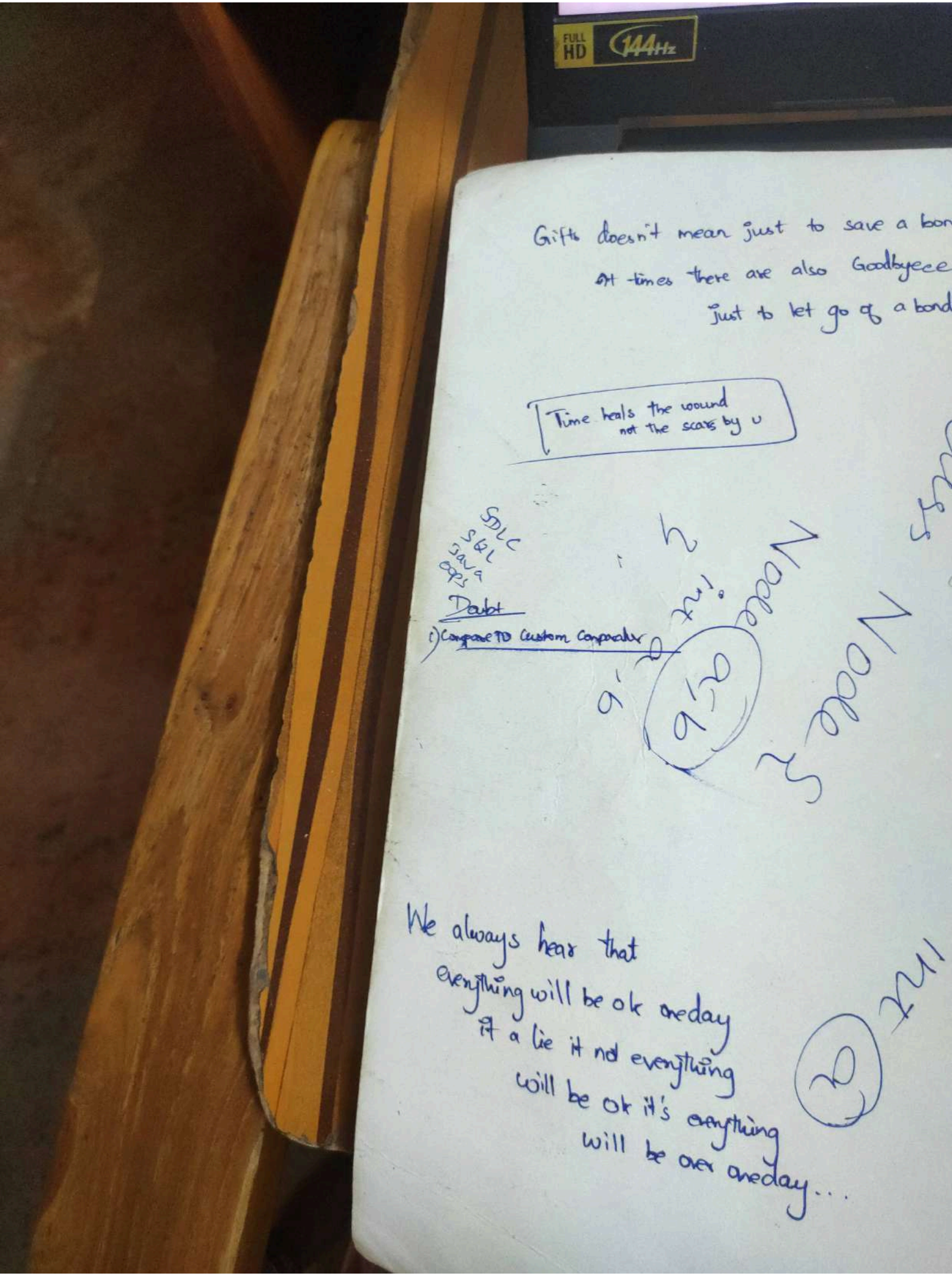

FULL HD
144Hz
Gifts doesn't mean just to save a bon
At times there are also Goodbyeee
just to let go of a bond
Time heals the wound
not the scars by u
SDLC
SQL
JAVA
OOPS
Doubt
1) Compare TO custom comparator
9:20
Node
Node
We always hear that
everything will be ok oneday
if a lie it not everything
will be ok it's everything
will be over oneday...

Gifts doesn't mean just to save a bond
At times there are also Goodbyeee!! gifts
just to let go of a bond :)

Time heals the wound
not the scars by u

SDLC
SQL
Java
oops

Doubt
1) Compare TO Custom Comparator

Node
Node 3
Sorts
int
9:50
2

Sql
CN
OS
DBMS
Optimized gr
Prime no.
rotate array
Round Robin
real time eg

We always hear that
everything will be ok oneday
it a lie it not everything
will be ok it's everything
will be over oneday...

M. Muthuram
CSE-B
2001126

If u are finally ready to say a goodbye
life gives u a new hii!...

Once someone asked "Isn't that hard? TO love someone who doesn't love you in return?"
he replied "It's the most beautiful feeling in the world.
There's nothing like the power of unrequited love...
Unlike other bonds, it isn't shared...
It's mine... and mine alone.

Nothing is permanent
Noteven the sun and moon even they leaves
us alone at night and day....

Never trust anyone and fall for someone
know that if u fall again there is noone
to lift u up...

Make yourself the priority instead of others
so noone can ever treat u like their
last of their list.... never.

Shaving your true self even to people u love put u
in a position where u will even lose your self respect

A word from heart is better
than ~~sentents~~ paragraphs from mouth...

Insertion sort
selection sort
Bubble sort
heap sort
pivot sort

Notes - tech
Notes - OS, DB
Apps
Opera - links
important folders
Round 3
whatsapp amma

Java:

- has garbage collector
- independent of platform (produce byte code)
- high level lang
- both interpreted & compiler lang

C

- low level lang used in circuit programmes
- platform specific
- Compiler lang.

---

1) We go for DP if already nama calculate panna visiyatha thirumba thirumba panura marie irundha
2) While using DP we can optimize that
3) we create DP size [m+1][n+1] → so we can handle even null pointer
4) 2 type — bottom up
   top down

Query string is everything after ? in url

hypertext - is a link that contain link to other text

markup - refers to tags & elements

```
char c = '5'; char D = 'A';   ascii value
int a = c - 0; // 5            '0' → 48
int a = D - '@'; // 1 because A is 1st
                                B is 2nd ...
                 ↓
                64 ascii value
```

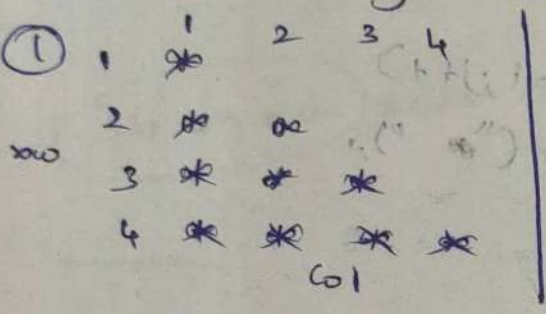
Pattern solving

(1) Run outer for loop no. of times rows.

(2) identify for every row no.
- how many col are there
- types of elements in col

(3) what do you need to print

(1)




```
Package com.kunal;
Public class Main {
    Public static void main (String[] args) {
        Pattern2 (n:4);
    }
    static void pattern2(int n) {
        for(int row=1; row<=n; row++) {
            for(int col=1; col<=row; col++){
                System.out.print("* ");
            }
            System.out.println();
        }
    }
```

(2)

```
for(int i=1; i<=n; i++)
{
    for(int j=1; j<=n; j++)
    {
        s.o.p("* ");
    }
    s.o.println();
}
```

(3)

```
for(int i=1; i<=n; i++)
{
    for(int j=i; j<=n; j++
    {
        s.o.p("* ");
    }
    s.o.println();
}
```

(4)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
for (int i = 1 ; i<= n ; i++)
{
  for(int j=1 ; j<=i ; j++)
  {
    S.O.P( j+" ");
  }
  S.O.println();
}
```

(5)

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

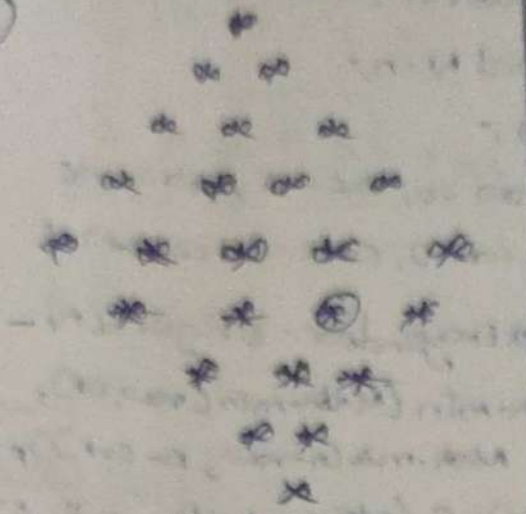
```
for(int i =1 ; i<=(2*n)-1 ; i++)
{
  if(i<=n)
  {
    for( int j=1 ; j<=i ; j++)
    {
      S.O.Print("* ");
    }
    S.O.println();
  }
  else{
    for(int j= i ; j<= (2*n)-1 ; j++)
    {
      S.O.P("* ");
    }
    S.O.println();
  }
}
```

(OR)

Any way

```
for(int row=0; row < 2* n ; row++)
{
  int totalColsInRow = row > n ? 2*n - row : row ;
  for(int col= 0 ; col <totalColsInRow; col++){
    sy.O.P("* ");
  }
  S.O.println()
}
```

(6)

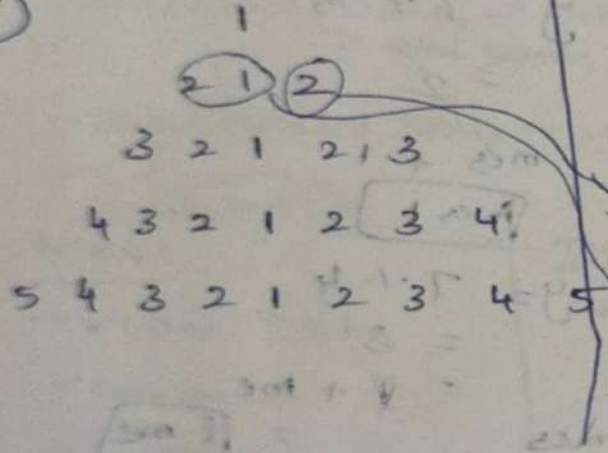


36.43 min pattern

```
for(int row=0; row<2*n; row++)
{
  int totalColsInRow = row>n ? 2*n-row : row;
  int noOfSpaces = n-totalColsInRow;
  for(int s=0; s<noOfSpaces; s++)
  { S.O.p(" "); }
  for(int col=0; col<totalColsInRow; col++)
  { Sy.O.p("*"); }
  S.O.println();
}
```

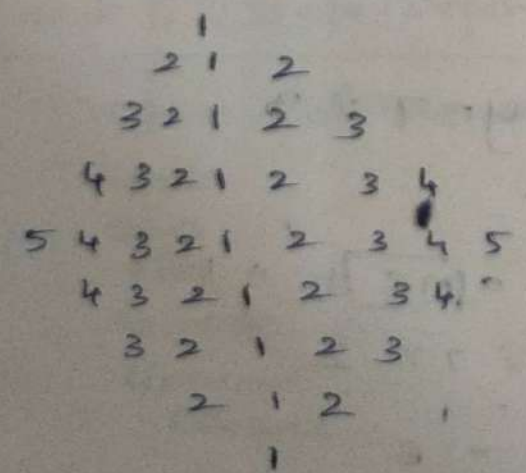
---

(7)



```
for(int row=0; row<=n; row++)
{
  for(int space=0; space<n-row; space++)
  { S.O.p(" "); }
  for(int col=row; col>=1; col--)
  { S.O.p(col+" "); }
  for(int col=2; col<=row; col++){
    S.O.p(col+" ");
  }
  S.O.println();
}
```

(8)



```
for(int row=1; row<=2*n; row++)
{
  int c = row>n ? 2*n-row : row;
  for(int space=0; space<n-c; space++)
  { S.O.p(" "); }
  for(int col=c; col>=1; col--){
    S.O.p(col+" "); }
  for(int col=2; col<=c; col++)
  { S.O.p(col+" "); }
  S.O.println();
}
```

F L A M E S
0 1 2 3 4 5

L A M E S sb3 -

7 % 5 = 2

MySQL

Database — is a collection of structured & non structured information that are stored & controlled by database management systems (DBMS)

* store data
  - hierarchical DBMS — Parent child
  - network DBMS — support many to many relation
  - Relational DBMS — tables & relation (most popural) → MySQL Database, SQL Sequel eg: microsoft DBMS
  - Object Oriented DBMS — store new data type in form of object
  - Central DB
  - Distributed DB

MySQL

→ relation DBMS
- open source
- multi

DB is a collection of structured or unstructed data stored

```
4 4 4 4 4 4 4 4 4
4 3 3 3 3 3 3 3 4
4 3 2 2 2 2 2 3 4
4 3 2 1 1 1 2 3 4
4 3 2 1 0 1 2 3 4
4 3 2 1 1 1 2 3 4
4 3 2 2 2 2 2 3 4
4 3 3 3 3 3 3 3 4
4 4 4 4 4 4 4 4 4
```

```
static void pattern21(int n){
    int originalN = n;
    n = 2*n;
    for(int row=0; row<n; row++){
        for(int col=0; col<n; col++){
            int edgeIndex = originalN -
                Math.min(Math.min(row,col),
                Math.min(n-row-1, n-col-1));
            System.out.print(edgeIndex + " ");
        }
        System.out.println();
    }
```

Flames

```
String temp;
if(y!=0)
    temp = sb3.substring(y) + sb3.substring(0,y-1);
else
    temp = sb3.substring(0, sb3.length()-1);
```

sb3 = flames

= f l a m e s

0 1 2 3 4 5

① $y = x \% sb3.length()$

$= 7 \% 6$

$y = 1$

② $y = 7 \% 5$

$= 2$

mes + 1

③ $y = 7 \% 4$

$= 3$

$= 2 + me$

$= 2 me$

④ $y = 7 \% 3$

$= 1$

$= me + 1$

⑤ $y = 7 \% 2$

$= 1$

$= e$

F I A M E S
0 1 2 3 4 5

I A M E S 563.

7 % 5 = 2

MySQL

Database – is a collection of structured & non structured information that are stored & controlled by database Mangement systems (DBMS)

* store data

- hierarchical DBMS – Parent child
- Network DBMS – support many to many relation
- Relational DBMS – tables & relation (most popular)
  → MySQL Database, ~~My SQL~~ e.g: microsoft DBMS
- Object Oriated DBMS – store new data type in form of object
- Central DB
- Distributed DB

MySQL

- relation DBMS
- open source
- multi storage engines

DB is a collection of structured or unstructed data stored in table or Json format.

SQL Commands Categories

DDL (Data Definition Language)

- Define database schema

Create
Drop
alter
Truncate
Comment
Rename

Data Manipulation language (DML)

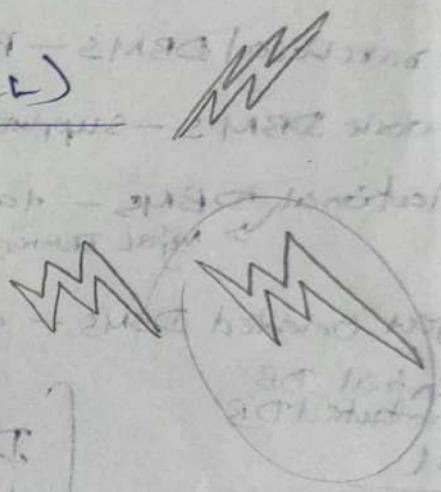
- manipulate data in database
  - select
  - insert
  - update
  - delete

Data Control language (DCL)

- Deals with rights permissions
  - Grant
  - Revoke

Transaction Control language (TCL)

- deals with transaction data
  - Commands
    - Commit
    - Rollback
    - Savepoint
    - set transaction



---

1) avg amt of interest per year which the company had to pay during this period?

a) Rs. 32.43 lakhs b) 33.72 lakhs c) Rs. 34.18 lakhs
d) 36.66 lakhs

2) total amt of bonus paid by the company during given period is approximately what percent of total amt of salary paid during period?

a) 0.1%. b) 0.5%. c) 1%. d) 1.25%.

3) total expenditure on all these items in 1998 was approxi what % of total expenditure in 2002?

a) 62%. b) 66%. c) 69%. d) 71%.

4) total expenditure of Company over these items during year 2000 is?

a) Rs 544.44 lakhs c) Rs. 501.11 lakhs
b) Rs. 446.46 lakhs d) 478.87 lakhs

5) ratio b/t total expenditure on taxes for all years & total expenditure on fuel & transport for all years respectively is approximately?

1) d) 36.66 lakhs.

2) c) 1%. $\frac{Pnr}{100} = 4016.25$

$P = \frac{4016.25 \times 100}{5 \times 9}$

$= \frac{401625}{45}$

3) d) 71%

4) a)

5)

451 : 586

$\frac{17}{1720} \times 100 \sim 1\%$

$\frac{204.4}{289.4} \times 100 = 70.63 \sim 71\%$

3.00 + 2.52 = 5.52; + 3.84 = 9.36; + 3.68 = 13.04; + 3.96 = 17.00

23.4 + 32.5 = 55.9; + 41.6 = 97.5; + 36.4 = 133.9; + 49.4 ...

288 + 342 = 640; + 324 = 964; + 336 = 1300; + 420 = 1720

98 + 1.12 = 2.10; + 1.01 = 3.11; + 1.33 = 4.44; + 1.42 = 5.86

83 + 1.08 = 1.91; + 74 = 2.65; + 88 = 3.53; + 98 = 4.51

101 + 41.6 = 142.6; + 74.0 = 216.6; + 324.0 ... 540.0; + 3.84 = 543.84

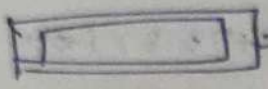
98 + 23.4 = 121.4; + 83.0 = 204.4

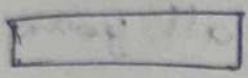
142 + 49.4 = 191.4; + 98.0 = 289.4

SQL

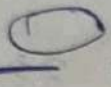
ER Diagram (Entity Relationship) → Gives database design

 = weak relationship

 = weak entity

 = strong entity } name of things like customer, product, Market ... etc.

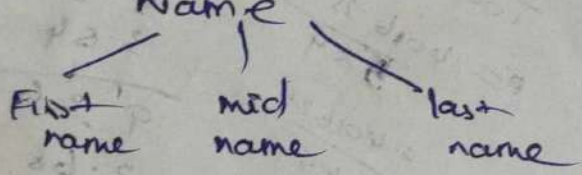
 = strong relationship

 → attribute

partial participation

total participation

Composite attribute means it can be further divided



Multivalued attribute - can have several values eg: ph & location

Stored vs Derived attribute

Birthdat → age

Complex attributes

We can group many attributes with commas,

5 keys in relational database

Primary key - unique

Super key - is a superset of candidate key.
- combination of 2 or more attributes that can identify a table uniquely.

Candidate key - subset of superkey
all candidate key are super key

Foreign key - one table oda primary key enom table oda field.

Alternate key - Contains all property to be a candidate key.

Normalization

- redundancy & dependency is reduced.

SQL Attributes data types

Numeric
Character-string
Bit-string
Boolean
Date & Time
Timestamp.

Constraints

Not Null
Unique
Check
Default
Index → Used to retrieve data from database very quickly.

DML

logical operators

AND / OR / NOT

Comparison operators

=, <, >, <=, >=, <>

Aggregate fn.

- Count
- MIN
- Max
- SUM
- AVG

Special operators

- Between
- IS NULL
- LIKE
- IN
- EXISTS
- DISTINCT

Schema – structure

```
CREATE TABLE 'Company'.
'employee' (
'fname' VARCHAR(15) NOT NULL,
'minane' CHAR NULL,
'lname' VARCHAR(15) NOT NULL,
);
```

```
INSERT INTO 'Company'.
'employee' (fname, minane, lname)
values ('John', 'B', 'Smith');
```

Varchar – ANSI standard
constant length

varchar2 – Oracle stand
can change size during execution

Relational algebra & Relational Calculus

Unary Relational operators

DDL

create

```
CREATE TABLE table_name (
  Column_Name datatype (size),
  :
  );
```

Drop

- DROP TABLE table-Name ;

Alter

ALTER TABLE Table_name ADD Column_name datatype size;

Truncate

TRUNCATE TABLE Table_Name;

RENAME

RENAME TABLE old_Table_name to New_Table_name;

—

UPDATE employee
SET salary = 27000
WHERE SSN = '333445555';

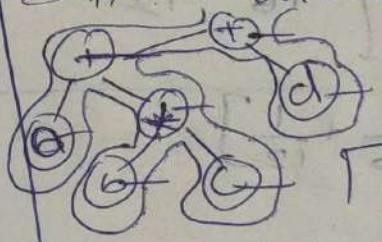
Relational Algebra & Relational Calculus

- Select
- Project
- Rename

Set operations

Union
Intersect
MINUS

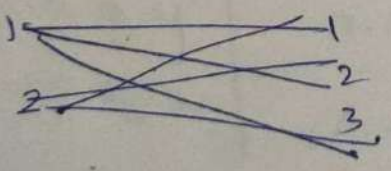
Infix → normal ah eladhuradhu
Refix, postfix
Infix = a+b*c+d → always see precedance while drawing tree



abc*+d+ → postfix

U know prefix

CARTESIAN Product (cross product)



SELECT * from emp;

Select colnames from tablename where condition;

Simple interest

$\frac{PTR}{100}$ =

1) 3 yrs = 815

4 yrs = 854

S.I for 1 year = 854 − 815 = 39

S.I for 3 yrs = 39 × 3 = 117

Principal amt = Rs (815 − 117) = **698**

$A = \frac{1}{16}$

$B = \frac{1}{12}$

$(A+B+C) = \frac{1}{4} \Rightarrow \frac{1}{16} + \frac{1}{12} + C = \frac{1}{4}$

$\Rightarrow C = \frac{1}{4} - \left[\frac{1}{16} + \frac{1}{12}\right]$

(C's 1 day work)

$= \frac{5}{48}$

C alone can do the work in $\frac{48}{5} = 9\frac{3}{5}$ days

$\frac{A}{B} = \frac{2}{3}$ | $\frac{B}{C} = \frac{5}{8}$

$B = \frac{3A}{2}$ | $B = \frac{5}{8}C$

$\frac{3A}{2} = \frac{5}{8}C$

$A = \frac{10}{24}C$

$A + B + C = 98$

2 | 2,3,4 / 1,3,2

Find second largest number

or select salary from emp order by salary limit 1 offset n-2;

select Max(salary) From emptable where salary < (select Max(salary) from emptable);

Find the number that occurs consecutively 3 times

select DISTINCT L1.marks AS consecutive
From logs L1 JOIN logs L2 ON L1.marks = L2.marks AND L1.student_id = L2.student_id-1)
& log L3 ON L1.marks = L3.marks AND L2.student_id = L3.student_id-1;

Find days when temperature was higher than its previous dates

Select weather.id As 'Id' from weather JOIN weather W On DATEDIFF(weather.recordDate, W.recordDate) = 1 AND weather.temperature > W.temperature;

select w1.id from weather as w1, weather as w2 where w1.temperature > w2.temperature and DateDiff(w1.recordDate, w2.recordDate) = 1

Delete duplicate rows

select unique * from tablename;

Delete from yourtable where id NOT IN (select MIN(id) from yourtable Group By column1, col2 ...);

To sort & retrieve based on index

Select your col from yourtable order by your_column Limit 1 Offset 1;

Display salary of each employee from Jan to July

Select empid, SUM(salary) As total_salary from emp where hire_date >= '2023-01-01' and hire_date <= '2023-07-31' Ground By emp_id;

Mean, Median & mode

1) Select AVG (salary) As mean_earning from earning table

2) Mode

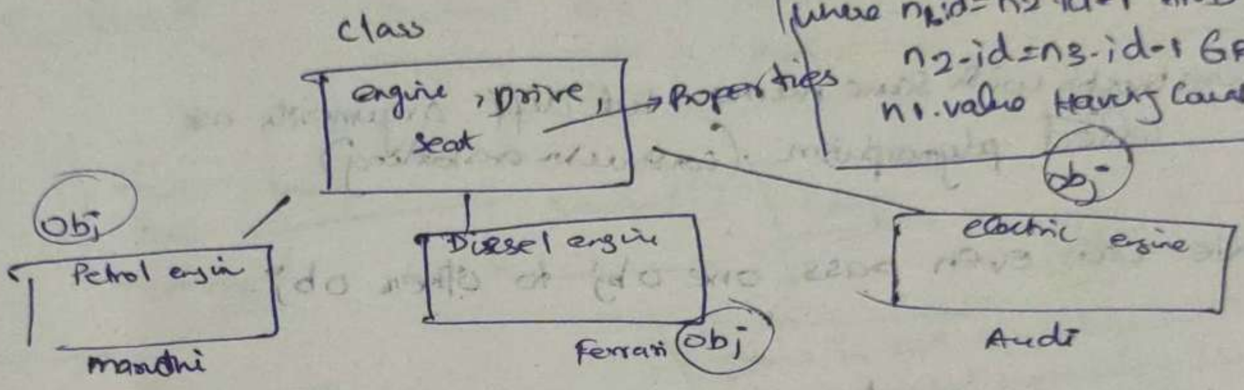
Select TOP colname from table_name GroupBy (Colname) order by Count(*) Desc;-

Class

* is a named group of properties & fns.
* U can create new data type using class.

finding 3 consecutive nums

```
select n1.value
from Number n1
JOIN Numbers n2 ON
n1.value = n2.value JOIN
Number n3 on n1.value = n3.value
where n1.id = n2.id-1 AND
n2.id = n3.id-1 GROUP BY
n1.value HAVING Count(*) = 3;
```



* Basically class is the structure
* obj is instance of class → like physical stuff

Class → logical construct
obj → phy reality (occupies memory, space)

9500884895
healthedge
Simran

9354381582

* We can access data using (.) operator
student1.rollno;

* to create obj u need to use 'new'

compile time — run time

Student student1 = new Student();
- Student → classname
- student1 → reference var
- Student() → constructor → special type of fn in class
- type
- new → allocates memory at run time & dynamically allocates memory to reference variable

Dynamic memory allocation - means memory will be allocated at run time of program

* Constructor is a special fn.
* that runs when U create an obj
* it allocates some variable
* Memory for class is created only when Constructor is called during obj creation

Constructor
- initialise obj

Destructor.
- destroys the place occupied by that obj
- There is no destructor in Java instead it has garbage collector that works similar.

this - assign value to properties of class
- access obj inside template
- this reference to obj reference variable

this.stu_name

Constructor with same name but diff arguments are called polymorphism. (constructor overloading)

We can even pass one obj to other obj.

Constructor overloading

```
Student();
Student(int a, string b)
```

This refers to what obj we are referring to.

U can call one constructor from within other constructor using this()

eg: Student() {

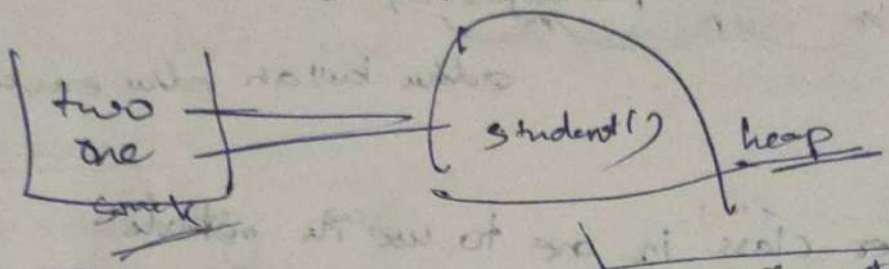
```
    this(13, "default Person", 100.0f);
}
Student(int rno, String name, float marks){
    this.rno = rno;
    this.name = name;
    this.marks = marks;
}
```

Object is stored in heap

two reference variable point to same obj in heap

In stack

```
Student one = new student();
Student two = one;
```



WrapperClass

variable
int a = 10;

object
Integer num = 45;
↳ wraper class

Java is not completely an object oriented programming lang because it has wrapper class. because we can wrap & unwrap even objects. has primitive datatype (8) eg: int char boolean

Java is pass by value

We cannot use final keyword for constructor as purpose of constructor is to create new obj but using final will make it impossible

final key word

↳ final int a = 10;
Primitive datatype
↳ this can't be modified

```
final Student kunal = new student();
kunal.name = "new name";
```

if final is applied to obj then it can be modified by it can't be assigned to other obj

```
kunal = other obj;  ✗
```

Packages

* folder like containers that hold multi files that has a seperate Class, fn in it.

eg: Package com.kunal.packages.b;

edhu kullah edhu enuku

Import other class in one to use its methods

if class is in same folder o no need to import it we can access directly

~~if it is public~~

~~to print~~

to keep common fact to all new obj in a class.

we need to declare that variable as static

for that variable allocation in construct use Class Name instead of this.

```
Clas Human
    int age;
    String name;
    static long populations;
    Human(int age, String name);
    this.age = age;
    this.name = name;
    Human.population += 1;
}
```

Static variable (not dependent on obj) belongs

whenever u increment use class name not reference variable

eg: Human.population ++; ✓

don't use this.population ++; ✗

Same way

while accessing use

Sy.o.p (Human.population); ✓

Sy.o.p (Rahul.population); ✗ → this works but don't use many

When a member is declared static it can be accessed before even obj is created.

Static variable, members can also be access by obj

Public class Main {

Public static void main (String[] args) {

array of string that store command line execution and to main fn

} }

declared static because it don't need obj to access it. It is the 1st to run in the Main class while executi

* Static means belong to class not to obj

* static method can access only static members & variab because they belong to obj/instance

static void greeting () {	void greeting () {
}	}
doesn't belong to obj	belong to obj

But a non static member fn can access static variable & member fn.

There is a thing called static block

```
Public class StaticBlock{
    static int a= 4;
    static int b;
    static {
        S.O.P("I am in static block");
        b=a*5;
    }
    Public static void main(String[] args){
        StaticBlock obj=new StaticBlock();
        S.O.P(StaticBlock.a+" "+StaticBlock.b);
    }
```

run only when obj is created

It only runs once throughout the program

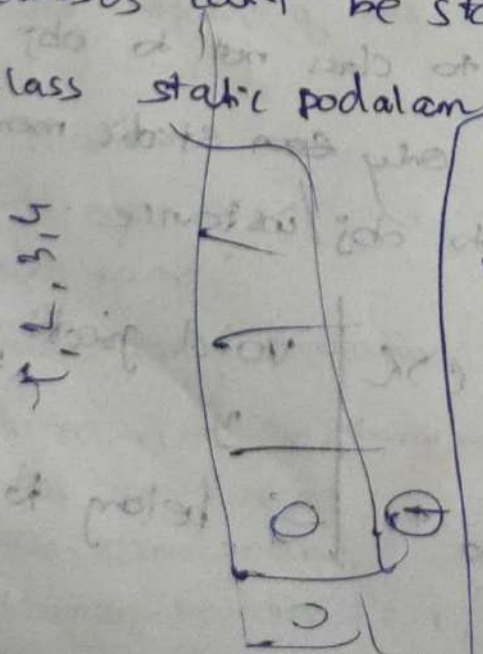
runs before main fn

3 3

1 2 3 4 5

In nested classes

- outer classes can't be static
- inner class static podalam



(class can also be declared static but outer class can't be static

If we declare class static it can't be inherited.

Inheritance → one obj can acquire all fn of other parent obj

→ base class

Parent class' members fn & variable can be used by

Child class

→ derived class

```
class child extends Base {
    int weight;
}
```

```
child child = new child();
child.Length;
child.weight;
```

---

when u use super() it calls the immediate parent class Constructor.

if u declare something private u can use only in this file.

If u create obj for parent class u can't use it to access member var & fn of child class.

eg: if

```
class Box {
    int l;
    int h;
    int w;
}
```

```
class BoxWeight extends Box {
    int weight;
}
```

if u create obj like this

Whatever u have u can access it's var & member fn.

Box boxs = new BoxWeight(l:2, h:3, w:4, weight:8);

reference is of type base class

obj created for child class

in this kind of obj creation u can access only members fns & variables of base class not child class member fn & variables

error

BoxWeight box b = new Box(2, 3, 4);

this means u can access all that child & parent member & var

but here u create a constructor of base class that don't know about child class members & variable

What u can access depend on reference var not on constructor

When u use

super(); it will call the immediate parent class constructor.

```
class object {
  Object();
}
```

this is the base class & all classes from which every thing is derived

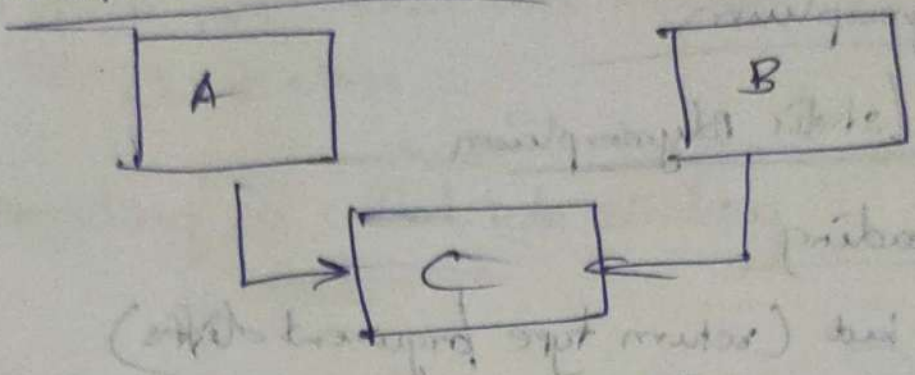
if the same variable is present in both parent class & child class to access that parent class variable u use

super

s.o.p (super.weight);

u can pass super(int a, string b)...; with paramer but only thing is the immediate parent should have constructor with that many no. of argument

multiple inheritance



C obj = new C();

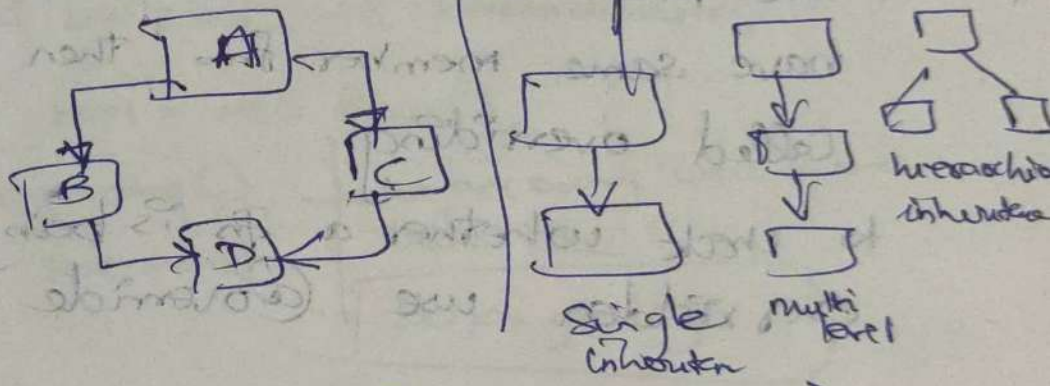
C.m(); this is not allowed in java

multiple inheritance is not allowed

we can do this in interface

hybrid inheritance

is not allowed in java



(0+(6-0))/2

Polymorphism

many ways to represent

Shape [triangle, square, circle]

is eg of polymorphism

```
Public class Shapes{
    Public void area(){
        S.O.P("I am shape");
    }}
```

```
Public class Square extends Shapes{
    Public void area(){
        S.O.P("I am a square");
    }}
```

```
Public class Circle extends Shapes{
    Public void area(){
        S.O.P("I am circle");
    }}
```

```
Shapes shapes = new Shapes();
Circle circle = new Circle();
Square square = new Square();
```

if u call the member fn with that obj it will return which is present in the obj of that class

Types of polymorphism

① Compile time / static Polymorphism

method overloading

same name but (return type / argument diffs)

Java determine which Constructor to be called during run time

overloading()

② Run time / Dynamic polymorphism

by method overriding.

if the parent class & base class have same member fn's then it is called overriding.

to check whether a fn is being overridden use @override

Shape circle = new Circle(); → upcasting.

↓ we can access fn only if it is present in base class

↓ what fn is being accessed depends on obj type

@override

kela endha fn/method eruthu nalum parent classla erukku method ah modify panna use agum like inside child class we override parent class method so nama call pannu bothu this modified override method print or being. call

$ if u put final over method u can override it in child class.

Overriding - is called late binding (runtime takes plan)

we can even put a class as final so it can't be inherited by other classes & automatically all it's method will be final

Aptitude
record
dbs is
intrays q n
call surrendhar
Draw
Aptitude
enrol
Rantly
Oops
hashmap
tree
Apple feels like bike clear

if a method is declared static inside a class we no need a obj to call it it can be be called by class name.methodname;

eg: Box box1 = new Box();
box1.greeting(); → not necessary we can even call like
↓ static method
Box.greeting();

Static methods can't be ~~overridden~~ override

(Overridding depends on obj, static don't depend on obj so static methods can't be overridden)

arr1 = new long[]{a1};

a 10
3
0

Encapsulation (external info)

Wrapping up the implementation of the data member & methods in a class Ex: class

Abstraction (external info)

hiding unnecessary details & showing valuable information.

Eg: starting a car with a key.
all u need is a key but u don't wanna know how petrol pump, to engine, Power supply to the tires etc...

Eg: Like abstract data types like arraylist, linked list, double linked list etc.
we don't actually know how it works we just use key words to do so

---

if u put private to a data member u are hiding it
u can access it by getters & setter method

return value set value

```
Eg: Public class A{
      Private int num;
      Public A(int num)
      { this.num = num; }
      Public void SETNum(int num)
      { this.num = num; }
      Public void getNum(){
          return num; }
}
```

U can get data member that is private by using
A obj = new A();
A.getNum();

but can't by
A.num; ✗

	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World (diff pkg & not subclass)
Public	+	+	+	+	+
Protected	+	+	+	+	
no modifier	+	+	+		
Private	+				

Inbuilt (Java)

- lang
- io
- util (datastructure & collection)
- applet
- awt
- net

---

hashcode

* means unique representation of something with number
* random integer value

even 34 → 1st 34 have random unique value

34 → 2nd 34 " Other " "

Interface

Difference bet interface & abstract class
→ In interface there can only be declared but it can't be defined but in abstract class we can depending on we we declared can defined or not.
→ interface don't have constructor but abstract class does
→ interface can implement 2 or more interface but abstract class can't

- blue print of class
- has static constant & abstract method.
- is a mechanism to achieve abstraction
- There can be only abstract methods in Interface
- Can achieve multiple inheritance.
- java compiler adds public & abstract keyword before interface methods & adds public, static & final keyword before data member

Class ← extends ← Class
Interface ← implements ← Class
Interface ← extends ← Interface

```
interface printable{ → small letter
    void print();
}
class A6 implements printable{
    public void print(){S.O.P("Hello");}
public static void main(String[] args)
{ A6 obj=new A6();
  obj.print();
} }
```

```
interface Printable{
  int MIN=5;
  void print();
}
```
→ compiler →
```
interface Printable{
  public static final int MIN=5;
  public abstract void print();
}
```

```
class A7 implements Printable,Showable{
} → is possible
```

marker / tagged interface is called because it don't have any method.

Hashmap

Uses hashing

* Convert large string to small string which represent the large string

* key value pair
* internally uses linked list to store key value pair
* part of java.util
* allows duplicate value but not duplicate keys
* 1 key only contains 1 value
* order may change
* In hashmap capacity is multiplied by 2
* has map Constructor

Hashmap() → create capacity 16 when instance then it create
create load 0.75

Hashmap(int initialcapacity) → create given capacity
when instance create
create load 0.75

Hashmap(int initialCapacity, float loadfactor)
↳ Create capacity instance & given load

Hashmap(Map map) → create map

```
import java.util.HashMap;
Public class Hashmap
{
  public static void main(String[] args)
{
  HashMap<String, Integer> map = new HashMap<>();
    Print(map);
    map.put ("abc", 10);
    map.put ("mno", 30);
    map.put ("xyz", 20);
    S.O.P ("Size " + map.size());
    Print (map);
    if (map.containsKey ("abc"))
    { Integer a = map.get ("abc");
      S.O.P ("value of abc " + a);
    }
    map.clear();
    Print (map);
}
Public static void print (Map<String, Integer> map)
{ if (map.isEmpty())
  {
    S.O.P ("map is empty");
  }
  else
  { S.O.P (map);
  }
}
}
```

key → value

Passing map as args.

Output

map is empty

Size of map is 3

{abc=10, xyz=20, mno=30}

value for key "abc" is :- 10

map is empty.

HashMap methods

Void clear() → remove all mapping in map

Boolean containsKey(Object key) → return true for specified key

Boolean containsValue(Object value) → return true if (or more key is mapped to same value

Object clone() – return shallow copy of method hashmap

Boolean isEmpty – check whether map is empty

Object get(Object key) – fetch the value mapped by key

Set keySet() – return set view of keys.

Int size() – return map size

Object put(Object key, value) – insert value at particular key pair in map.

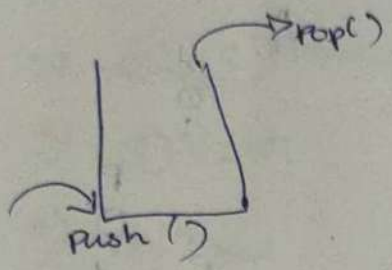
PutAll(Map M) – copies all elements of 1 map to other

Object remove(Object key) – removes values from particular key in Map.

Collection values() – return Collection view of values in hashMap().

Stack

FILO



- stack is a class
- But Queue is an interface
- Queue is not a class & don't have constructor

So

```
Queue<Integer> queue = new
                LinkedList<Integer>();
```

- Peek works in both Queue & stack
  it peeks the top but don't remove it

- queue.remove() → removes the 1st as it is the queue

- Stack & queue used to store long list of data or data in group stored as ae

Deque

priority queue used to sort elements in it by default.

- Doubly ended Queue
- we can insert & delete at both side
- is a interface

Poll() → used to implement it so when

```
Deque<Integer> deque =
        new ArrayDeque<>();
```

↓ has no capacity restriction.

- we can insert 1st & last
- " " delete 1st & last

Abstract class

```
abstract class Shape{
    abstract void draw();
}
class Rectangle extends Shape{
    void draw()
    {System.out.println("Drawing rectangle");
    }
}
class Circle extends Shape{
    void draw()
    {System.out.println("Drawing circle");
    }
}
class ex{
  public static void ...
  {Shape s = new Rectangle()
   s.draw() → And drawing rectangle
   s = new Circle();
   s.draw() → And drawing circle
  }}
```

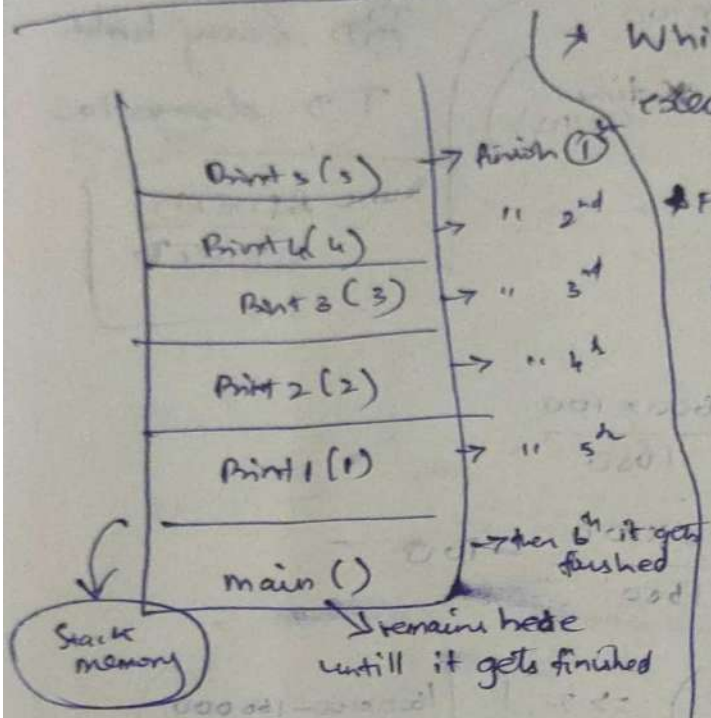
if method is abstract then it's class should also be abstract.

Dictatorial

1) Dogmatic.
2) to cat someone like a celebrity - lionize
3) Gentle - ~~benine~~ benign
4) ~~brief~~ brief and to the point → laconic
5) disguise or contradict - belie
6) person who hates others people → missanthrope
7) hostile & aggressive - beligerent
8) Unfeeling, impartial - dispassonate
9) Reason supporting an idea or cause publicly advocate
10) Lack of harmony - Dissonance.
11) Alter or debase - Adulterate
12) support or strengthen - bolster
13) Someone who attacks cherished beliefs or institutions. - iconoclise
14) to add more to idea - Corporate
15) diligent or extremely careful - scrutulous
16) no longer needed or useful - redundant
17) in existence. - extant
18) Essentially different in kind - disparate.
19) to avoid giving a clear or direct ans to a qn - equivocate
20) subject to sudden or unpredictable changes & mod. mercurial

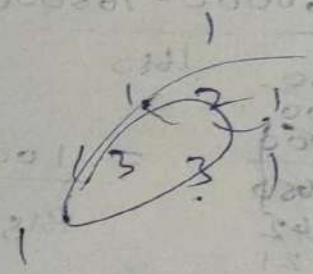
Recursion

how fn calls works in language



* While the fn is not finished executing it will remain in stack.

* For every fn, when it finishes executing then it is removed from Stack & restore it control flow to where it is called.



Self Introduction:

Good Afternoon all! My name is Muthuram. I come from Coimbatore and I'm currently persuing my ~~Bachelor degree~~ BE Computer science Engineering in Sri Rama krishna Engineering college. I have done my schooling in BFMS. During my studies, I developed a foundation in Java, C, C++, sql, HTML & CSS..

I had quiet done few project (points) one of them include soil nutrient analysis. Besides academic I'm also a good team worker..

* My areas of interest include java Programming, problem solving &

I spend my spare time drawing, playing football and kabbadi.

curval, curwt

120, 30
100, 20
60, 10

Capacity - curwt >= 0
50 - 30 >= 0
Capace = 20
total val += 120

60, 10
100, 20
120, 30

6 < 5
else -1

6
5

5
5
4

5 < 4

60, 10
120, 30
100, 20

Factional knapsack prob

totalvalue = 0
50 - 10 = 40

fraction = 20/30

$30 \times 2/3 = 20$ Capacity = 50 - 10 = 40

total value += 60

$120 \times 2/3 = 80$ Capacity = 20 - 20 = 0

40 - 20 = 20 >= 0

Capa = 40 - 20 = 20

total += 160 + (120 * 2/3) = 160 + 80 = 240

total = += 60 + 100 + 160

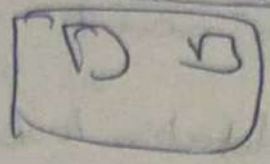
20 - 30 >= 0

React + JS

Front end

React used

JavaScript Xml
JSX



reactfunctionality

useState (update particular content)

use effect (to change particular component in screen like color, style)

Components (code redundancy like importing particular component)

importing

routing (one page to other)

Data structure

Express — Communication

Reactte node client & server

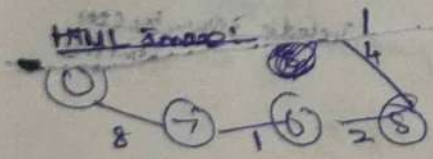
↘ front - to - back end

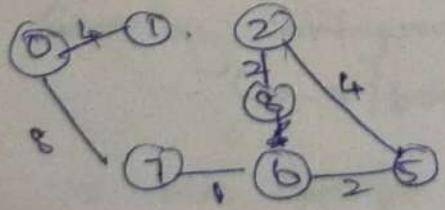
have many methods

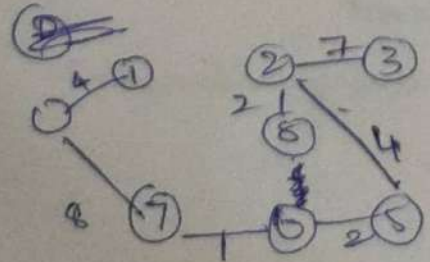
get

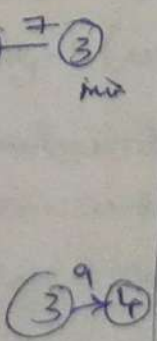
post

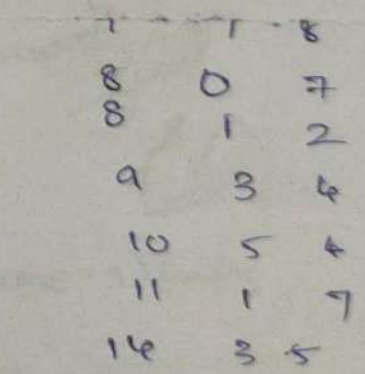
Cors - to communicate local host for security

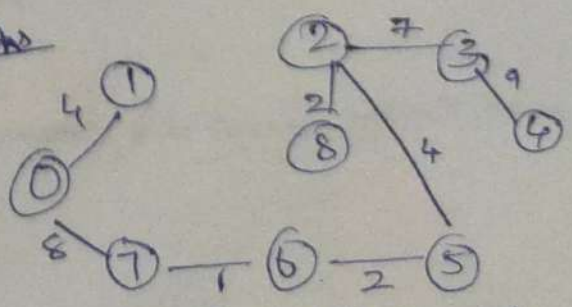




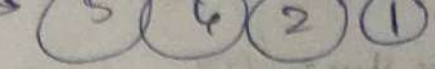








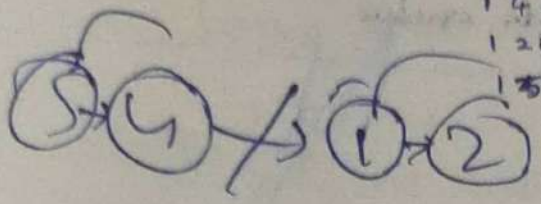
This don't form cycle



h = 5
temp = 4
temp2 = 5

4 5 2 1

2 4 5 1
1 4 2 5
1 2 4 5
1 5 4 2



Database — andha name table / relation

Primary key - unique & not null

foreign key - relation.

INF
2NF
3NF

Normalization
- used to reduce redundant value/duplicate
- section should not be

Atomic value
→ epaem ch eddhuna andha value dha

INF
- Atomic value
- ~~partio~~

2NF
- partial dependency / partial fn dependency (FD)
  - 2nf should satisfy INF (Atomicity) → endha 1-Primary depend

3NF
- transitive dependence

A → B
B → C
A → C

* Satisfy INF & 2NF
* Uses super contains

Incident Management

* Multi application. - ticket

↳ one Role - access - ticket raise

→ CI →

→ ticket tools

* JIRA
* Service IT
* service NOW
* Cherwell
* ITSM

Technical stack

SPD HR

IT

P1 - low

ticket raise
↓
form fill
↓
concern team
↓
new ticket
↓
new P1
↓
we need to give more

→ Ticket will be raised

* Incident Management

- Network team
- Cloud "

→ Incident Management will be called

→ will pass the ticket & bug to concern.

→ Backend details will be given

→ Ticket normal approach - total every ticket has a SLA - we have to assign ticket to concern team & take action immediate & we need to close the ticket before SLA Breaches.

Ticket

Priority

P1 P2 P3 P4

P1 - call or email trigger

Company ku incident manager erukaga

Cbg - ku incident management

- CSE
- ECE
- Web

→ Incident mangent

& will call

Will be in bridge / not

All detail will be & with us.

we get details

& Connect with request team.

* we get & transfer data
* we need save this form bridge
* Mediator - to connect diff teams

* epo - assign panna member ku tenla na service

* if service goes down / network not stable we need to fix.

task - set q. rule thes

* time matters -

access request

access removal

- Enhancement
- application is not working
- to build we need service
- Everything should go via Incidat

* Eg: airtel middleband.

admin - line

* Incidat →

ACT - fibernet

↓ design → developer

→ app - mobile app developer

→ wap - web developer

Healthedge

temp - ~~2018~~ 2004 - ~~Bang~~ ★ hyder | pune

* GD - [card]

- Communication
- observer - then decide to talk

So I ~~don~~ would like to say more about

& don't agree with opponent

- Continue.
- Clear & crisp
- don't interrupt (. Yes . . . then start)

(As fast as possible)

P1 - Whole company ke down   (applications)

P2 - one set of region work
& india don't work

2+3 hrs

½ work ½ don't work

1 to 2 day ← P3 - request id lost

P4 - new to company I need new
Id - takes time -

↓

make list   Data

Monday -

and

1000 employee

Revenue
owner

IT time - attendance marker

↳ incident management

→ installation

- uninstall

→

Arrays

System.arraycopy(rotated, 0, nums, 0, nums.length);

1) static

2) Arrays.toString(arr) → print entire array [1, 2, 3]

3) Arrays.sort(arr) → use Dual pivot implementation of quick sort
   → when sorting obj we used merge sort
   (we are using primitive datatype)

4) Arrays.asList(arr) → convert array of fixed size to arraylist → non fixed size.

5) Arraylist → like this will work only in wrapper class eg: Integer

6) Arrays.binarySearch(array or arraylist , key value);

   Collections.binarySearch(arraylist, keyvalue);

   ↳ to perform binary search first sort the array or arraylist

7) (arr1 == arr2) → will always return false

8) Arrays.equals(arr1, arr2) || for arraylist arr1.equals(arr2);
   ↳ return true or false depends on the

9) Arrays.compare(arr1, arr2)
   returns 0 → when both equal
   1 → when arr1 lexographically greater than arr2
   -1 → " " " lesser " "

lexographical → Dictionary / alphabetical order

10) When u initialize an array
   int[] arr = new int[5];
   all values will be 0 first

System.arraycopy(source, starting index of source, destination array, destination array start index, size)
System.arraycopy(arr, 1, subarray, 0, M);

11) Arrays.fill(arr, 0, 3, 1)

end index

start index

what should it be filled it may be 0, 1, 2, 3..

output [1, 1, 1, 0, 0]

Collections

1) ArrayList<Integer> arr = new ArrayList<Integer>();

2) list.add(value);

3) list.add(index, values);

4) list.remove(index); → list.remove(Integer.valueOf(value));

5) list.set(index, value); → change the value at a particular index to new value.

HashSet

HashSet<Integer> set = new HashSet<Integer>();

1) set.add(values)
2) set.clear()
3) set.isEmpty()

Tree set
used to sort set
LinkedHashset
→ will follow the same order of input.

4) Iterator<Integer> it = set.iterator()
while(it.hasNext())
{
Sys:out.pin(it.next());
}

5) set.size()

6) set.removeAll(value)

for(Integer i: Set)
{
}

HashMap<Integer, Integer> map = new HashMap<>();

1) map.put(key, value);
2) map.get(key);
3) map.isEmpty();
4) map.size()
5) map.clear()
6) map.remove(key) → remove value
7) map.containsKey(key) → return boolean
8) map.containsValue(value) → return boolean
9) map.keySet() → gets all keys in map
10) map.values() → get all values in map.
11) Collections.max(map.keySet());
12) " . " (map.values());

```
for(Map.Entry<Integer, Integer>
    entry : map.entrySet())
{
    entry.getKey();
    entry.getValue();
}
```

TreeMap

TreeMap is nothing but a sorted map.

TreeMap<Integer, Integer> tree_map = new TreeMap<>();

Sorting is based on key.

```
for(Integer i : map.keySet()){
    S.O.P(i); //key
    S.O.P(map.get(i)); //value
}
```

Collection methods

1) Collections.sort()
2) Collections.rotate(__, values) → to how much position
3) Collections.reverse()
4) Collections.binarySearch(arr, value) //return index of value or -1
5) Collections.copy(____, ____); ← this copied to this

Two pointer

arr1 = [1, 2, 3] arr2 = [2, 3, 4, 5, 6]

(Sort both arrays)

while (i < arr1.length && j < arr2.length)

Create new array it start index is k=0

if arr1[i] < arr2[j]

arr[k] = arr[i]

i++ ; k++ ;

else arr[i] > arr2[j]

arr[k] = arr2[j]

j++ ;

k++ ;

else if arr[i] == arr[j]

arr[k] = arr2[j]

j++ ;

i++ ;

k++ ;

Then add the remaining element of big array to new array use k index

---

Sorting 0's & 1's without Array.sort()

[1, 0, 1, 1, 0, 0, 0, 1, 0, 1]

then i should iterate until it finds 1

j should iterate until it find 0

then it should swap as we need 0's in start & 1's in last

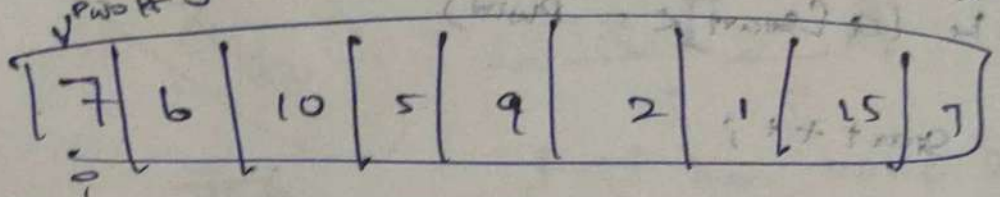
Comparator

1) If we are unable to use
   'Arrays.sort()'
   Collections.sort() → with modified merge sort
   then we can use Custom Comparator
   @override compare() method.

2) Comparator only works for wrapper class.

Quick sort

1) Based on divide & conquer technique
2) Take any no. as pivot (start or end or mid)

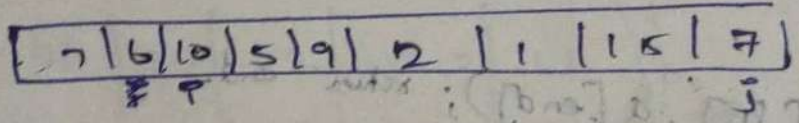


Here our pivot as edit here take a[0] = 7 as pivot

pivot = 7

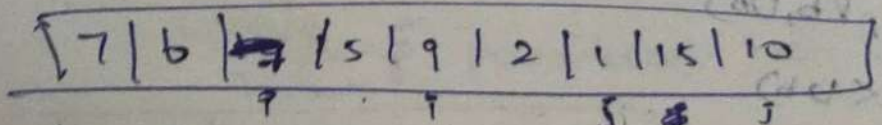
st = 0
ed = arr.length - 1

arr[i] 7 <= pivot
i++



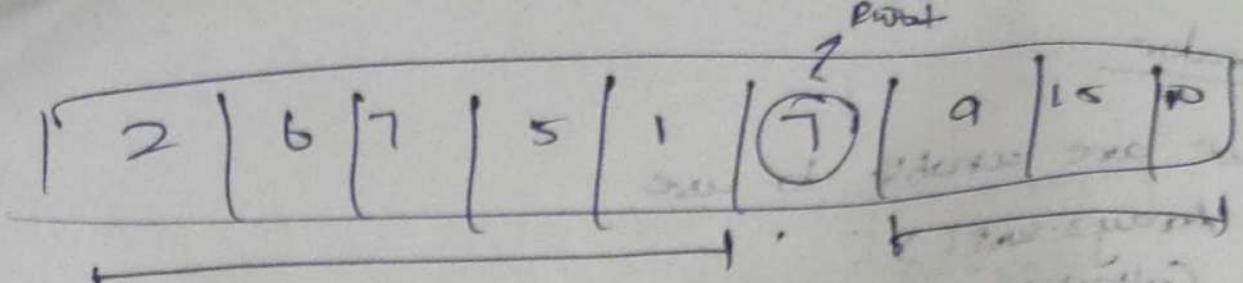
then arr[j] > pivot
so it is not

So swap arr[i] & arr[j]



7 <= 7
i++

↳ when i > j then don't swap
then swap the pivot element with the j element



partition again

then again take 2 arr[0] as pivot element & perform same operation.

Partition [A, lb, ub) → Quick sort

```
{ pivot = A[lb]
  start = lb
  end = ub;
While(start<end){
  while [A[start] <= pivot)
  { start++;
  }
  while [A[end] > pivot)
  { end--;
  }
  If [start < end)
      swap(A[start], A[end]);
  }
swap(A[lb], A[end]); return end;
}}
```

```
Quicksort [A, lb, ub)
{ if [lb < ub)
  { loc = Partition [A, lb, ub);
    Quicksort [A, lb, loc-1)
    Quicksort [A, loc+1, ub)
  }}
```

→ swap(int lb, int end)
{ temp = A[lb];
A[lb] = A[end];
A[end] = temp;
}

Pivot

i

7 6 10 5 9 2 1 15 7

j

i <= pivot } gladitiv

j > pivot

if j < i then swap pivot with arr[j]

7 6 10 5 9 2 1 15 7

i j

swap

7 6 7 5 9 2 1 15 10

i i j j

7 6 7 5 1 2 9 15 10

i j

j i

~~7 6~~ j < i then swap [j] wt with j

2 6 7 5 1 (7) 9 15 10

fixed position

Perform same for left & right

Selection sort

64	25	12	22	11

i

```
P.S.V. selection(int[] arr)
{ int n=arr.length;
  for(int i=0; i<n-1; i++)
  { int minindex = i;
    for(int j=i+1; j<n; j++)
    { if(arr[j] < arr[minindex])
      { minindex = j;
      }
    }
    int temp = arr[minindex];
    arr[minindex] = arr[i];
    arr[i] = temp; } }
```

1st element edukku, adhu la enundhu fullah traverse panni edhu least oh adhoda swap pannikkum nei (11)

i

11	25	12	22	64

11	12	25	22	64

i

11	12	22	25	64

11	12	22	25	64

Insertion Sort

[12, 11, 13, 5, 6]

1) First element um next element compare pannanu ussendung la enundha ok illana sort pannu

[11 12 13 5 6]

[11 12 13 5 6]

→ [11 12 5 13 6]

→ [11 5 12 13 6]

→ [5 11 12 13 6]

[5 11 12 6 13]

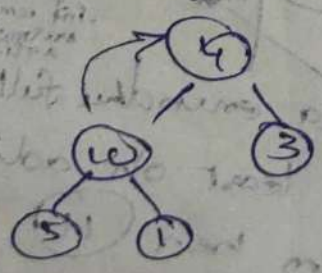
[5 11 6 12 13]

[5 6 11 12 13] → sorted.

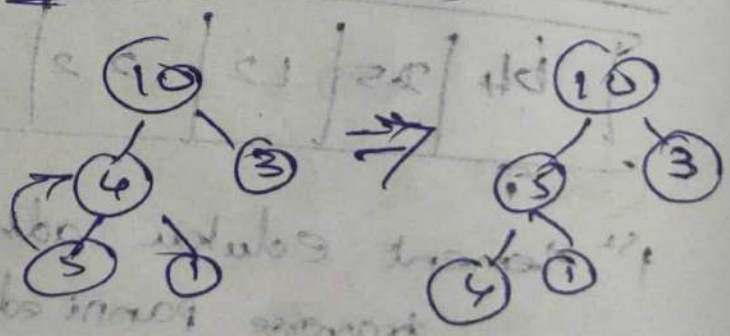
Heap sort

[4, 10, 3, 5, 1]

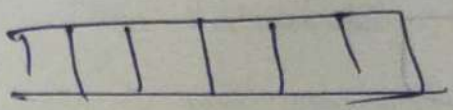
Build a binary tree



Build max heap

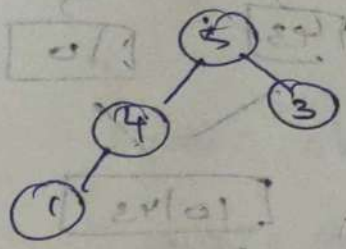


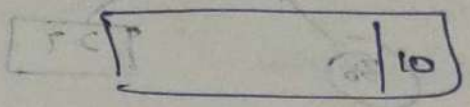
TO give sorted output use ~~queue~~ stack



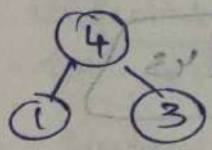
First remove (10)

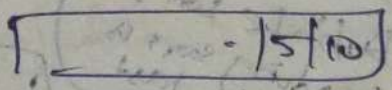
So take max of left ~~root~~ child or Least of right child



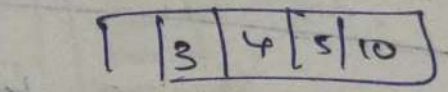


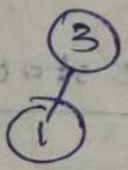
remove (5)





remove (4)



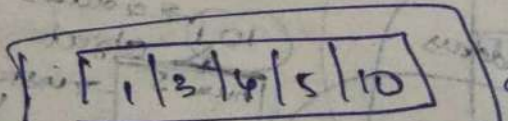


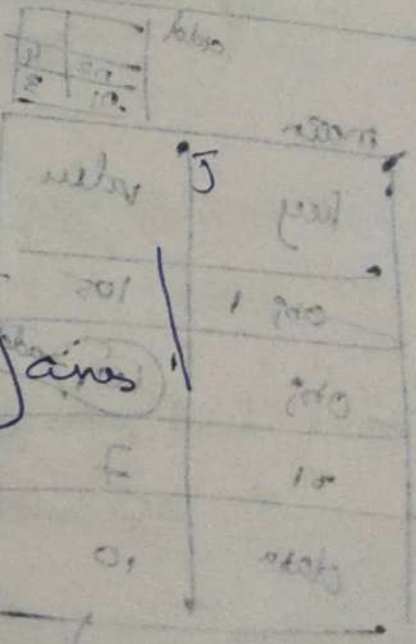
MVC

remove (3)

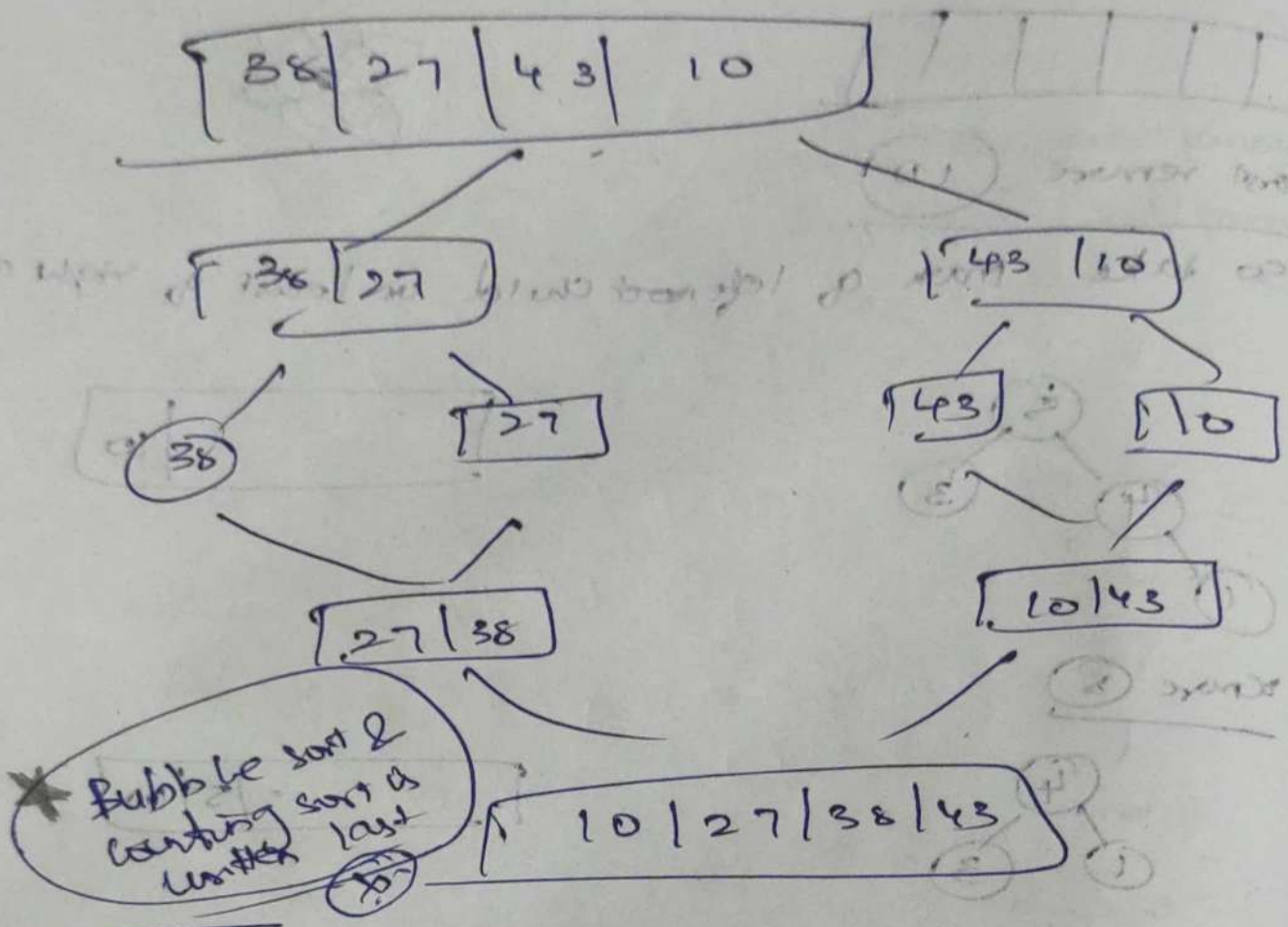
(1)

remove (1)



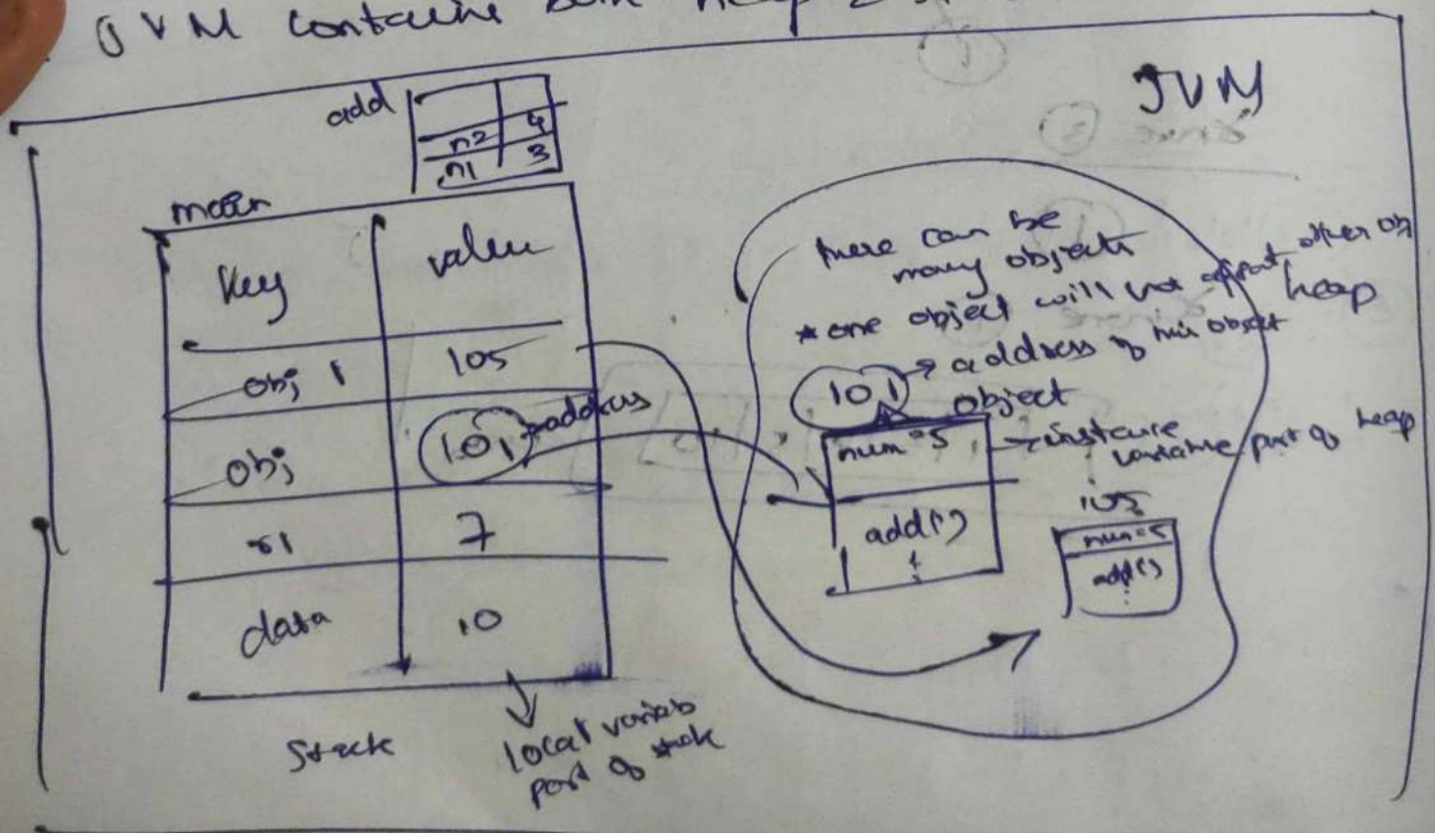


Merge sort



stack & heap (memory management)

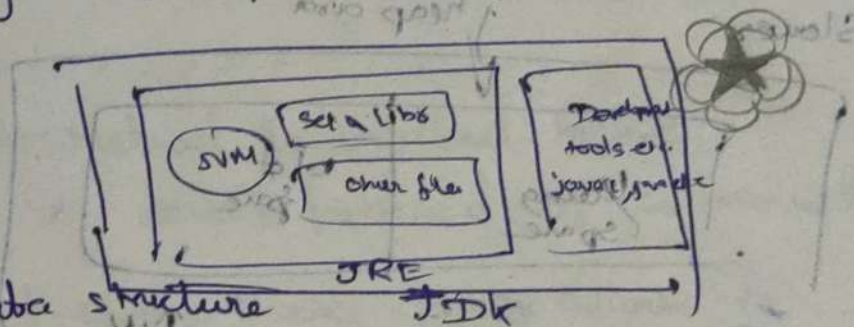
JVM contains both heap & stack



Eg: code

Instance variable - specific to particular instance of class
Can vary from obj to obj.

```
Class calculator
{
  int num=5; //instance variable

  Public int add(int n1, int n2)
  {
    return n1+n2;
  }
}
Public class Demo
{
  public static void main(String[] args)
  {
    int data=10;
    Calculator obj = new calculator(); //obj reference variable name
    Calculator obj1 = new calculator()
    int r1 = obj.add(n1:3, n2:4);
    System.out.println(r1);
  }
}
```



Stack

1) Linear data structure
2) LIFO → FILO
3) Push, Pop, Peek, TOP, isEmpty(), poll()

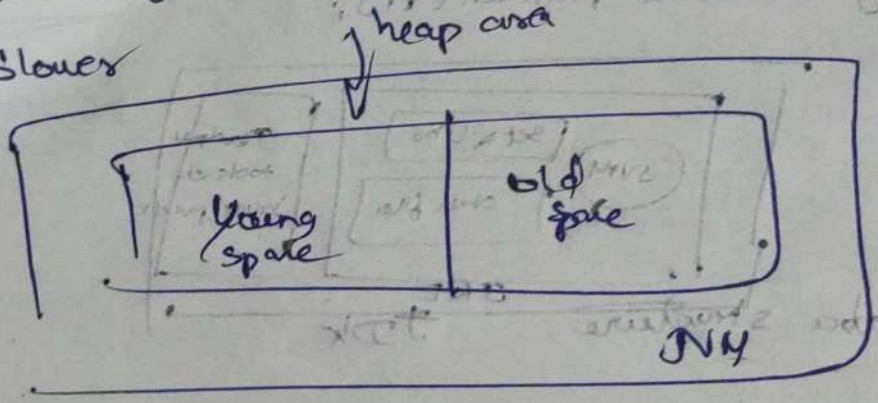
memory allocation is stack

→ memory allocated & subsequently freed without you
→ has limited size depends on OS
→ size varies
→ stack is a part of memory where temporary variables are stored.
→ has many short lived values & references of objects
→ variable in stack will exist until the fn that created them ends.

* Stack perform FILO
* This work perfect for local variable as it will get eliminated the fn created it exits
* Not suitable variable like instance variable
* Stack is fast as it can be accessed directly
* Used for static memory allocation

Heap

* Area in memory used to store objects
* Not managed automatically
* we need to free up old block/variable when they are no longer needed
* No size limit
* Slower



* Garbage Collection starts freeing object in old space following FIFO
* Used for dynamic memory allocation

Garbage Collector

* Garbage means unreferenced Object
* Garbage collection is process of destroying unreferenced objects

* In Java garbage collection done automatically.
* When heap full garbage collector starts eliminate from old space

Eg:

```
Student s = new student()
s = null;
```

then garbage collector free up s

```
Student s1 = new Student();
Student s2 = new student();
s1 = s2; // 1st obj available for Garbage collector
```

---

new student(); anonymous obj available for garbage collector

---

finalize() - method is invoked before unreferenced obj is removed

gc() - method invoke garbage collector to perform clean up.

gc() found in system & runtime class.

```
Public class Test Garbage 1{
  Public void finalize(){S.O.P.("obj is garbage collected");}
  Public static void main(String[] args){
    Test Garbage1 s1 = new Test Garbage1();
          s2 = new ;
    s1 = null;
    s2 = null;
    System.gc();
  }}
```

Searching

Linear search

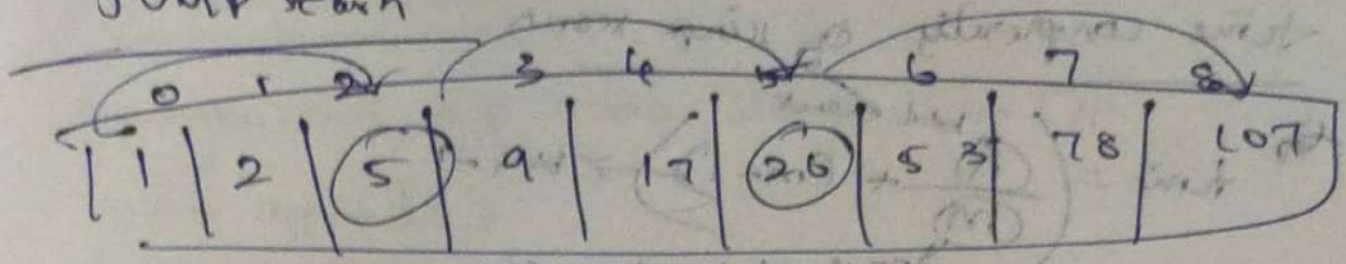
* Can be used for
  - sorted arrays
  - unsorted array
* O(n) - time complexity
* Brute force approach

Binary search

* Time complexity is O(log n)
* The array should be sorted

```
Public class FindPosition {
    Pub static vo ( ) .
{  int arr[] = {1,2,3,3,3,5,6,7,9,10,12,15}
   int pos = findPosition(arr, 0, arr.length-1, 3);
    S.O.P(pos);
}
Static int findPos (int[] arr, int start, int end, int target
while(start<end{
 if(start > end) {
     return -1.
}
int mid = start + (end - start)/2
If (arr[mid] == target {
     return mid; }
else if (arr[mid] > target){
     end = mid -1;
   return findPos(arr, start, end, target); }
else if (arr[mid] < target){
     start = mid+1;
     return findPos(arr, start, end, target); }
} }
```

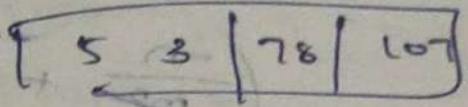
JUMP search



5 >= 78
false

26 >= 78
false

107 >= 78
true

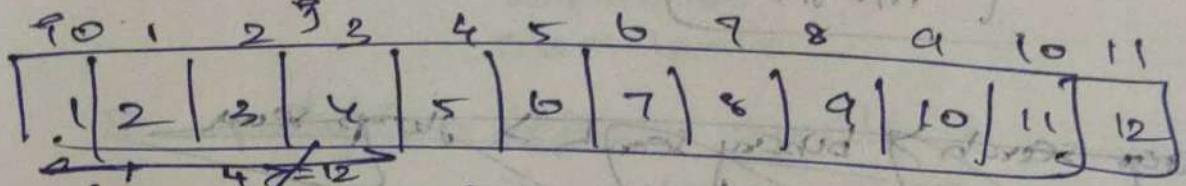


perform linear search

(78) found return index (7)

* Jump search works only when array is sorted

* Here 2 pointer
one at start & other at no of block end



block size = 3 (jump size = 3)

no. of jump = 0 + 1

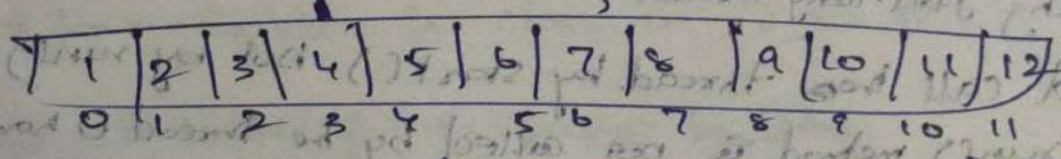
no. of element = 12

no. target = 12

Count the no of jumps

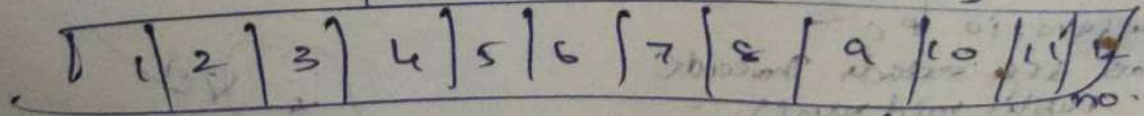
j = 3

j = j + jump size



6 >= 12

no of jump = 2



no. of jump = 3

10 >= 12

then perform linear search from (10 11 12)

if record return true
else false

time complexity of jump search

time = (n/m + (m-1)) → block-1

n → total element

m → no. of blocks jump size

$$\frac{d}{dn}\left(\frac{n}{m} + (m-1)\right) = 0$$

$$\left(\frac{-n}{m^2} \cdot + 1\right) = 0$$

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

$$n = m^2$$

$$m = \sqrt{n}$$

~~linear search > binary search > jump search~~

Linear search > Jump search > binary search

$O(n)$ > $O(\frac{n}{m} + (m-1))$ > $O(\log n)$

Thread

1) Can be implement by java.lang.Runnable interface
2) Extending java.lang.Thread class

we should call thread by start(), not by run()
because run() method is not called by the thread u have created instead it is called by thread that created your thread

```
import java.io.*;
class GFG extends Thread {
    public void run()
    { s.o.p("Welcome Method");
    }
public static void main(String[] args)
    GFG g = new GFG();
    g.start();
}
```

String Builder

StringBuilder sb = new StringBuilder();

sb.append()

sb.insert (3, "Java") → andha particular index la
change aerum
e: welcome
welJava

Sb.replace(3, 6, "Java") e: simplilearn
simJava learn

start ↑ end-1
Sb.delete (2, 5) → e: welcome
weme

Sb.reverse ( ) → simplilearn
nraelilpmis

Sb.capacity( ) → default capacity 16
if not then it will be
(n*2) + 2
(16×2)+2 ⇒ 34

Sb.length() → size

Sb.charAt()

Sb.indexOf ( ) → substring enukuna adhoda
start index point panum

Sb.setCharAt( );

★ StringBuilder is class in Java API

★ Sb.toString() → Change to String.

		8	3	3
0 3 7 1		7	9	7
		17	3	0

Bubble sort

always no. of passes = n-1

| 15 | 16 | 6 | 8 | 5 |

swap every two elements

Pass 1

| 15 | 16 | 6 | 8 | 5 |

| 15 | 16 | 6 | 8 | 5 |

| 15 | 6 | 16 | 8 | 5 |

| 15 | 6 | 8 | 16 | 5 |

| 15 | 6 | 8 | 5 | 16 |

Pass 2

| 15 | 6 | 8 | 5 | 16 |

| 6 | 15 | 8 | 5 | 16 |

| 6 | 8 | 15 | 5 | 16 |

| 6 | 8 | 5 | 15 | 16 |

| 6 | 8 | 5 | 15 | 16 |

Pass 3

| 6 | 8 | 5 | 15 | 16 |

| 6 | 8 | 5 | 15 | 16 |

| 6 | 5 | 8 | 15 | 16 |

| 6 | 5 | 8 | 15 | 16 |

| 6 | 5 | 8 | 15 | 16 |

pass 4

| 6 | 5 | 8 | 15 | 16 |

| 5 | 6 | 8 | 15 | 16 |

| 5 | 6 | 8 | 15 | 16 |

| 5 | 6 | 8 | 15 | 16 |

| 5 | 6 | 8 | 15 | 16 |

ella pass laum lasts la erundhu ovvoru element ah adho da crt possition ku bubble up etche varum

Regex in Java

other way

boolean b = pattern.matches("a.s", "abs");

edited pattern → complete sentence

```
package RegularExpression;
import java.util.regex.*;
Public class Main{
    Public static void main(String[] args)
    {
        String sentence = "God will God at the Of his God";
        Pattern p = Pattern.compile("God");
        Matcher m = p.matcher(sentence);
        int count = 0;
        while(m.find()){
            count++;
            S.O.P(m.group()+"starts at "+m.start()+"ends
                    at "+m.end());
        }
        S.O.P("This word occurs"+count);
    }
}
```

to find the particular element occur first or not

```
// Pattern p = Patten.compile("^God");
```

to find whether particular word occurs last or not

```
//Patten p = pattern.compile("God$");
```

to check whether a particular character occurs or not

```
//pattern p = pattern.compile("a|w");
```

```
pattern.compile("[a,b,c]");
Patten.compile("[^ab]"); → except a & b
```

[a-zA-Z]

[a-zA-Z0-9]

[^a-zA-Z0-9] → except

Pattern.compile("\\S"); Capital S non space character

Pattern.compile("\\s"); → space

Pattern.compile("\\D"); → non digit

Pattern.compile("\\d") digit

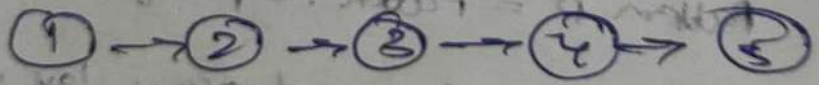
" ("\\W") → alphanumeric

("\\w") → Begumbaba

Single Linked List

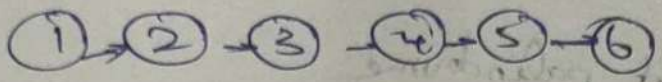
```
class Node{
    int data;
    Node next;
    Public Node(int data)
    {
        this.data = data;
        this.next = null;
    }
}
```

To reverse a linked list




```
class Solution{
    Public ListNode reverseList(ListNode head){
        ListNode prev=null;
        ListNode curr=null;
        while(head!=null){
            curr = head;
            head = head.next;
            curr.next=prev;
            prev=curr;
        }
        return curr;
    }
}
```

Return middle



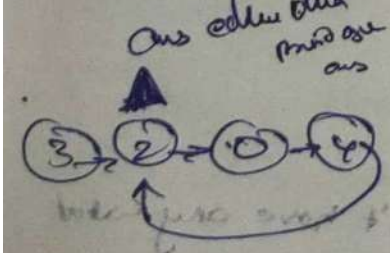
```
class solution{
    Public ListNode middleNode (ListNode head)
    {
        ListNode slow = head;
        ''    fast = '' ;
        while (fast.next != null && fast.next.next != null)
        {   slow = slow.next;
            fast = fast.next.next;
        }
        if (fast.next == null)
            { return slow;
        }
        return slow.next;
    }
}
```

fast != null & fast.next != null

Merge two sorted linked list is mobile & leetcode (easy)

Detect loop in LL (to find ending point and the loop vandhu joint ayudhu)

```
class solution{
    Public ListNode DetectNode (ListNode head)
        ListNode slow = head;
        ListNode fast = head;
        while (fast.next != null & fast.next.next != null)
        {   slow = slow.next
            fast = fast.next.next
            if (slow == fast)
                { break; }
        }
        if (fast == null || fast.next != null) return null;
        while (slow != head)
        {   slow = slow.next;
            head = head.next;
        }
        return slow;
}}
```



TO find Linked list palindrome

use StringBuilder
~~remove~~ reverse () fn
append()
toString()

HTML

<title>
We can use title tag as an attribute to any tag
- gives more info
<p title="model do"> Hi everyone </P>

* HyperText Markup language
* tell browser how to display webpage
* Describe structure of webpage
* Contains many tags & elements
* <!DOCTYPE HTML> → Describe this is an HTML5 doc

Nested html elements

* <html> → start tag
  <head> </head>
  <body </body>
  </html> → end tag

---

*
 → break tag
  → empty element, because it don't have any data
  → no closing tag

* HTML is not case sensitive <p> & <P> are same

Attribute

* Provide additional information to tags.

 click here

attribute

↓

anchor tag → contains hyperlink

<img src=" " > → <img> tag don't have closing tag

Types of URL

1) Absolute URL

→ Link that is hosted on other website
Eg:- https://www.scaler.com/image.jpg

2) Relative URL

if url begins without slash (/) → It is relative to current page

Eg:- src = "img.jpg"

if " " with " / " → It is relative to domain

src = "/img/img_girl.jpg"

<alt, width, height, use map, target are all attributes of img>

<style> is a attribute used to style elements.

Allways use lowercase for attributes

Always quote attribute values

<P style = "Property: Value;" > hi </P>

Properties

- background-color
- background-img
- color
- font-family
- font-size
- text-align

border: 2px, solid tomato;

=

Formatting element

<b>
<i>
<small>
<sub> - subscript
<sup> - superscript

==

HTML Comment

<!- - - ->

==

Color

rgb(255,255,255) rgb(0,0,0) - black
rgb(255,255,255) - white
# ff6347
hsl(9, 100%, 64%)
rgba( )
hsla( )

==

HTML CSS

Inline CSS → style attribute inside html element
Internal CSS → using <style> tag in head of html
External CSS → using <link> link the ext CSS

HTML Link

 click

unvisited link → underline blue
visited link → underline & purple
active link → underline & red

target attribute

* - self → Default, open doc in same window/tab as it was clicked
* - blank → open doc in new tab
* - parent → open doc in parent frame
* - top → open doc in full body of window.

<img src = "" alt = "" usemap = "#workmap">

<map name = "workmap">
<area shape = "rect" coords = "34,44,75,76" alt = "Emp" href = " ">
<area>
<area>
</map>

background-image: url("");

table

* <table>
<tr>
<th> </th>
<th> </th>
</tr>
</table>

html list

 - unorder • • •

 - order 1,2,3

```
<ul>
    <li>coffee</li>
    <li>tea</li>
</ul>
```

```
<dl>
    <dt> coffee </dt>
        <dd> - black hot drink</dd>
</dl>
```

Block Level element

* always starts in new line
* always takes full width available
  * eg: <p> & <div>

Inline element

* doesn't start in new line
* only takes as much space as necessary
  * <span> element inside paragraph
* central of a parala dhidirenu edho ~~hight~~ highlight
  pannanum na use pannalam
  adhuku seperate style kudukalam

<iframe>

used to display our kutti page mari enaum
website la use aye.

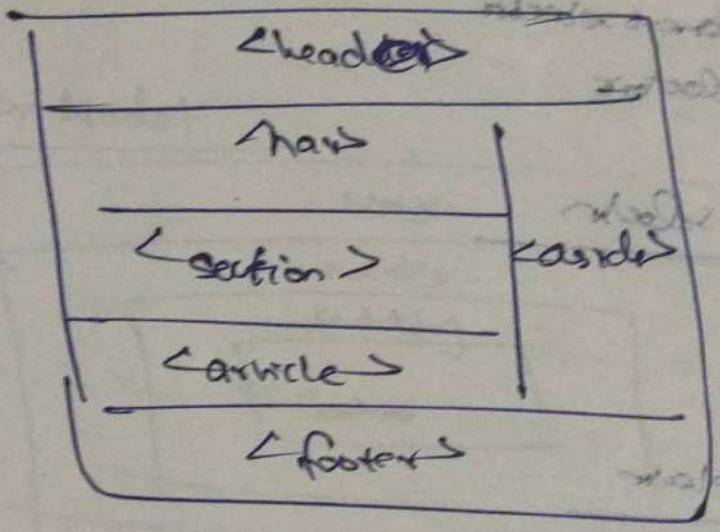
```
<iframe src=" " height=" " width="300"
        title="I an iframe"></iframe>
```

Javascript html

* to declare JS we need to declare <script> in head or footer
* Define a client side script
* Mostly uses document.getElementById() method.

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML="Hi all";
</script>
```

Layouts



#Choose

CSS flexbox
" grid
" framework

Responsive html

" will alter the design/structure of webpage according to laptop, ipad, phone automatically.

media query

```
<meta name="viewport" content="width=device-width,
                               initial-scale=1.0">
```

Css

```
h1 { color: blue;
     font-size : 10px;
}
```

Css selector

- Simple selector (id, class, name)
- Combinator selector
- Pseudo class selector
- Pseudo element selector
- Attribute selector

Css element selector

```
p{
  ...
}
```

Css id selector

* uses #

```
<style>
#para1 {
  ...
}
</style>
```

```
<p id = "para1" >hi</p>
```

only one element can have one id

Css class selector

* uses (.)

```
.center
{
}
```

```
<p class = center > hi</p>
```

Universal selector

```
* {
}
```

Css Group selector

```
h1
}
h2{
}
```

Comment in CSS

/* */

CSS Backgrounds

background-color
" - image
" - repeat
" - attachment
" - position

h1 > b -> eppadi vardha (parent selector)
h1 kullah enkuna direct child ah mattudha style pannu

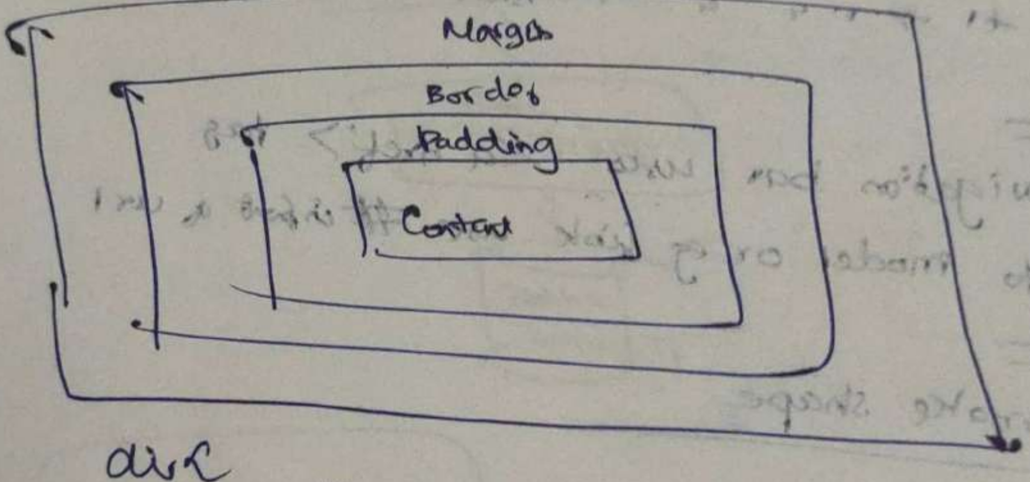
h1 + b -> h1 ku keela vara ellam'ah style pannu (adjacent selector) (takes only 1 adjacent)

h1 : hover -> adu mela kondhu vandha Color change agum.

h1 ~ b -> eppadi enundhu any b's after h1 will be styled.

Opacity - 0.3, 0.1 , 1 -> not transparent (range 0 - 1)
↓
transparent

CSS Box Model



div {
width : 320px;
padding : 10px;
border : 5px solid gray;
margin : 0;
}

Position

* Static
- not affected by top, bottom, left, right
- Default

relative:

- adjust itself according to page

fixed

* fixed to page even when page scrolled

absolute

* relative to nearest positioned ancestor

Sticky

* will behave as pinned note

z-index

overlaps element

-1 → image will be in background.

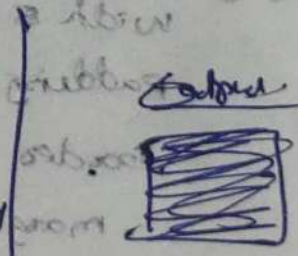
+1 → " " " front

Navigation bar uses <a href> tag

to model or eg link use ~~#~~ in front of url

---

to make shape



```
<style>
 .square {
    height : 50px;
    width : 50px;
    background-color : #555; }
 </style>
<div class="square"></div>
```

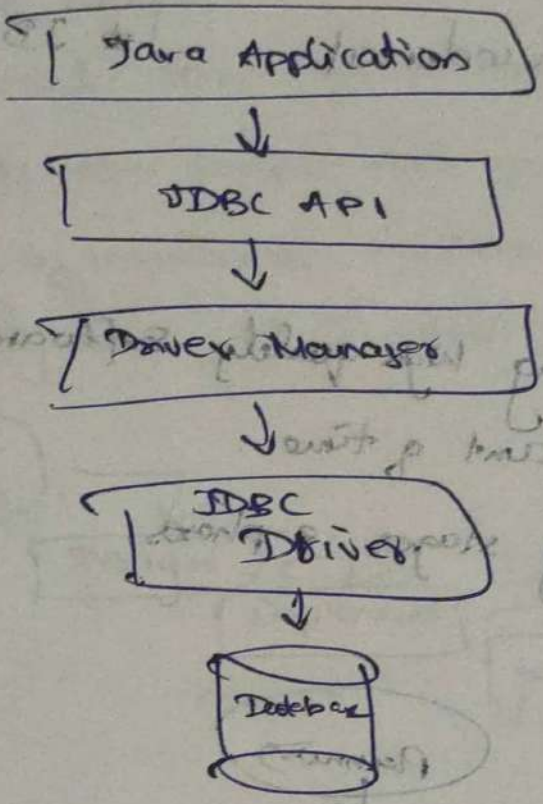
If circle called Border-radius : 50%

JDBC

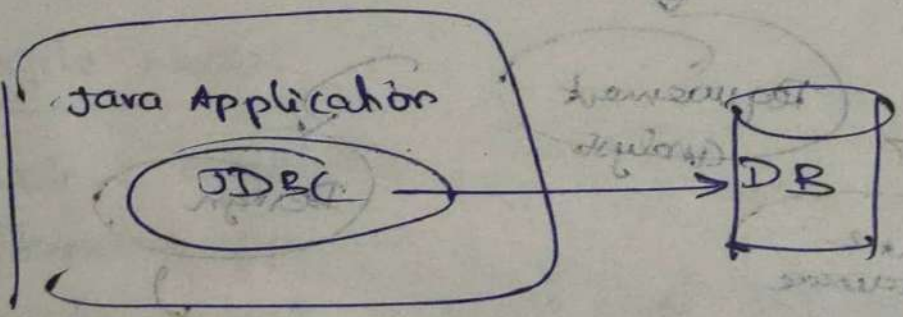
1) JDBC is an API for Java

2) It is a Java Database connectivity used to interact with database & retrieve information & execute sql queries

3) Uses JDBC Drivers to connect to DB (oracle, MySQL ...)

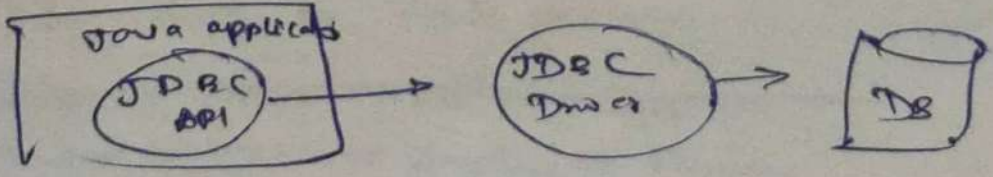


4)



5) JDBC Drivers

-> have many classes & interface to enable java application to interact with DB



ODBC / open Database connectivity	JDBC
★ used for C, C++, Java	★ only for Java
★ work only in window platform dependent.	★ platform is independent
★ ODBC drivers are developed in native lang like C, C++, Java	★ JDBC drive developed only in Java
★ ODBC is procedural	★ JDBC is object oriented.

SDLC

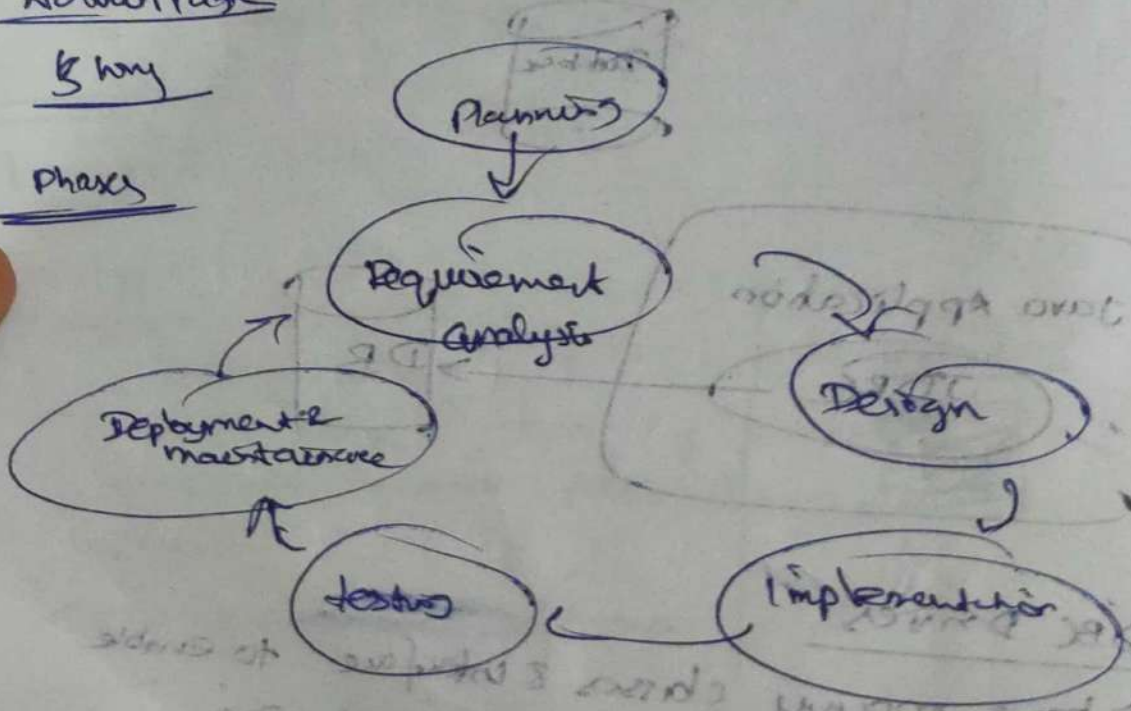
★ It is producing high quality software in low cost with least and q time.

★ It has many stages & phases.

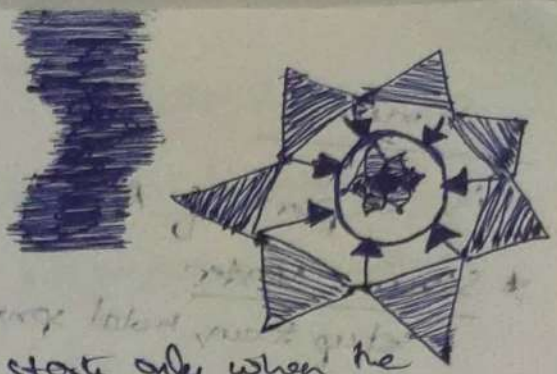
Advantage

Many

Phases

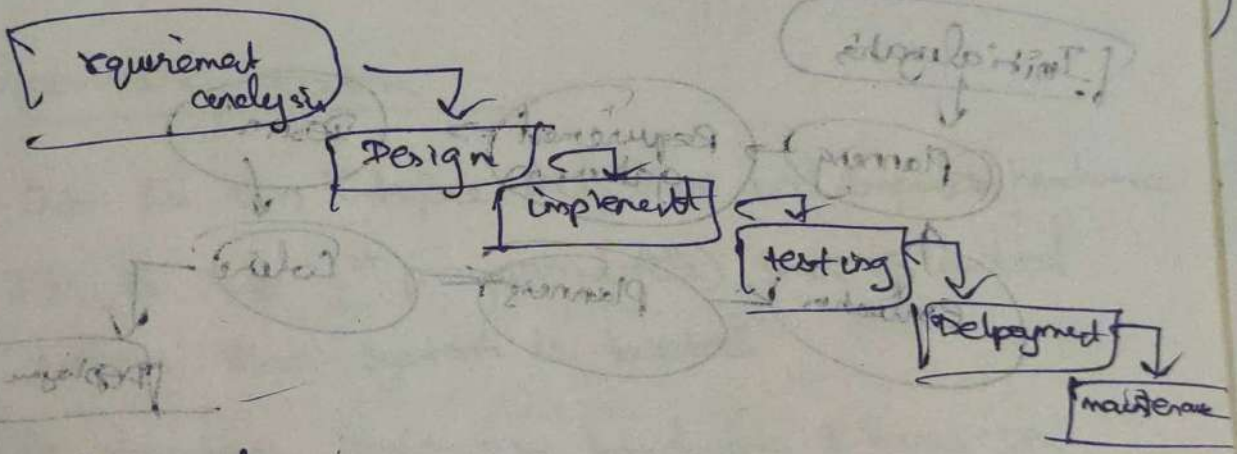


Types of SDLC Model

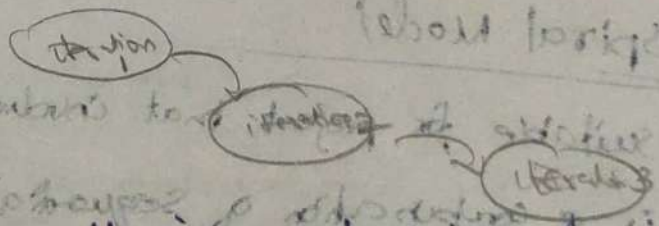


1) Waterfall model

* Linear sequential flow
* In this the next phase start only when the prev stage is finished.
* It is used where
  - requirement don't often change
  - used in non complicated project
  - Clarity of requirement provide by client
  - not dynamic it is static



2) Agile model



* Quick & adaptable
* Iterative approach
* Divide project to small piece & long term plan
* Each iteration is a small frame last for 1 to 4 weeks
* Used to interact with client & change the process & objective
* Change & adapt to raw strategy (sprint)

~~Develop → Design → Review → Deploy~~

Design → Develop → Test → Deploy → Review

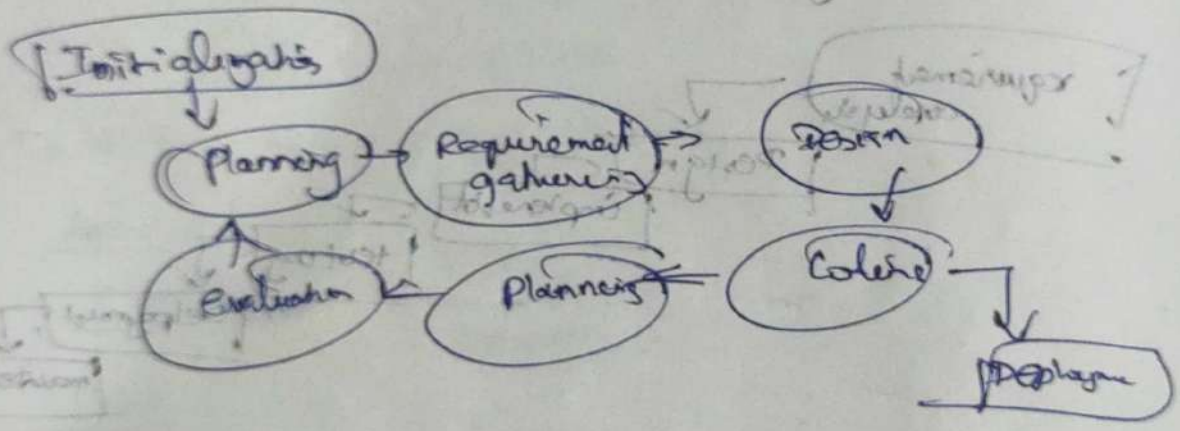
Scrum — follows shielded approach — Transparency, Inspection, Adaptation

is a part of Agile model

* Scrum Master:
  - setup team, hold sprint meetings
* Product owner
* Scrum team

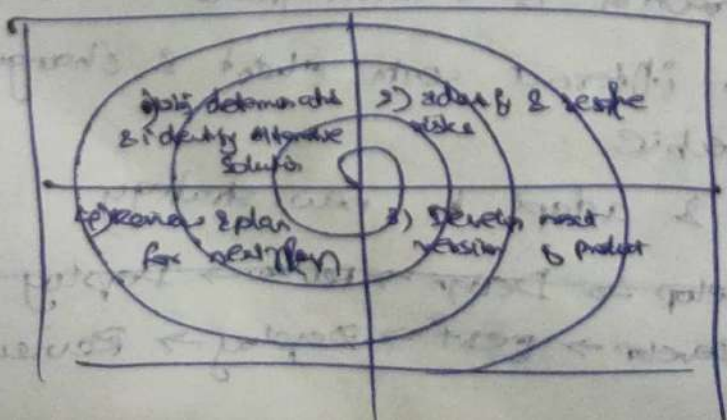
3) Iterative model

* In initial development is based on initial requirement
* we can add new features by iterative approach till the project gets completed.



4) Spiral Model

* suitable for project that involves risks
* is a combination of sequential model + iterative approach
* it uses step by step approach like waterfall model



levels of testing

Unit testing → testing individual component

Integration testing → test integrated components

System test → test entire sys

acceptance test → Test the final sys

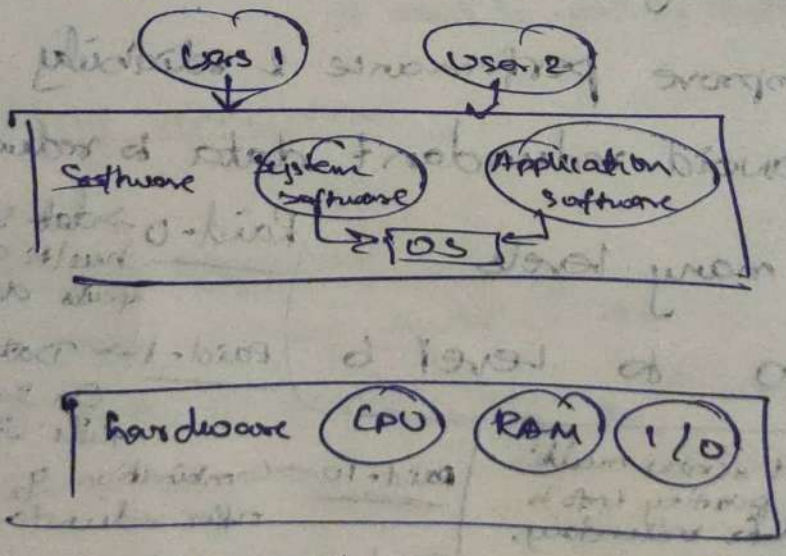
RAD model

1) Bussiness Modeling
2) Data "
3) Process "
4) Application generation
5) testing & turnover

* Rapid Application Development
* relie on prototype rather than design

Operating system

* OS is an interface bet. user and computer hardware
* It is the 1st program (file) that is loaded when the system is booted.
* It manages software, hardware & resources.



* Memory management, Process management
* provide user interface
* scheduling

Types of OS

* Batched OS (eg: Payroll, sys, transaction process etc.)
* Multi programming OS (eg: window O/s, UNIX etc)
* Time sharing OS (Multics)
* Distributed OS (LOCUS)
* Real time OS

=

Multiprocessor

* has many CPU, perform multi operation that share single resource.
* improve performance, run concurrently
* increase throughput
* reduce latency

==

RAID structure

* RAID - Redundant Array of Independent Disk
* used to store multi harddisk hence use data storage virtualization
* used to improve performance & reliability
* used to avoid redundant data to reduce data loss

It has many levels

Level 0 to Level 6

Raid-0 → data split across multi drives, 1 drive fails data lost

Raid-1 → Data is duplicate on 2 drives, if 1 drive fails other has data

Raid-5 → Data split across multi drives & parity info is distributed for redundancy. improve performance

Raid-6 → with additional parity data with fault tolerance

Raid-10 → Combination of Raid 1 & 0 offer redundancy & performance

Raid-50 → combination of Raid 5 & 0 Balanced performance

Raid-60 → Provide high fault tolerance & protection

Pipe

* Pipe is a connection among 2 or more processes that are related to eachother.
* Used to interprocess communication using message passing.
* It can send output of one process to other process by IPC.

---

semaphore

* is a variable to for synchronization
* Used to avoid critical section problem. & deadlock
* uses two variable
  - signal()
  - wait()
* 2 types

1) Binary semaphore
   * only two value 0 & (1) → it can enter critical section
     ↓
     it need to wait to enter critical section

2) Counting semaphore

Bootstrap program

1) first runs when power is on.
2) Cold boot → starts from 1st when power off
   warm boot → starts from restart stage

Demand paging

* It is a method that loads a page into memory only when needed based on demand.
* A new page brought to execution only when the reference of that page is called in execution.
* if any bug in page. It leads to page fault/trap

=

RTOS

* Real time OS
* That needs to perform a certain task within a specific time or else it leads to crash/prob.

Hard RTOS → The sys need to provide result within the time or lead to crash for sure. Eg: Air bag in cars, missile tracking

Soft RTOS → It needs to be perform in time but can has delay. Eg: skype call.

Firm RTOS

=

Synchronization

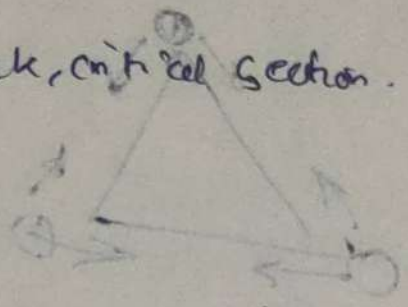
* It will coordinate process that share a resource or data.
* without process synchronization it leads to Deadlock, critical section, race condition

* To avoid use mutual exclusion.
* two types → Independent process
  → Cooperative process

mutual exclusion
hold and wait
semaphore

IPC

* Inter process communication used to communicate bet threads.
* This help to know which resource is occupied by which thread.
* helps to avoid Deadlock, critical section.



Techniques

1) Pipes
2) message passing
3) Semaphore
4) shared memory.

Overlay in Os

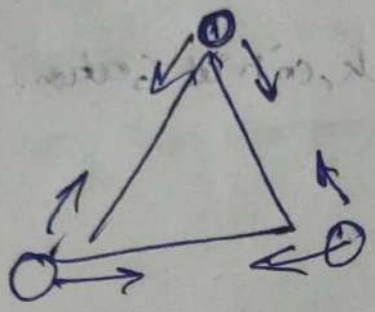
* is a programming method the divide large program & store only necessary info.

Os Eg:

1) Ms-Windows
2) Ubuntu
3) Mac Os
4) Fedora
5) Redhat
6) Android
7) Debian

3. Ant & Triangle puzzle

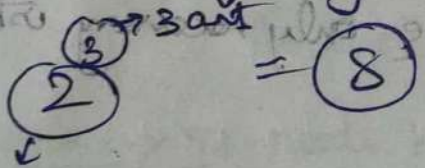
* what is probability 2 ants collide when all the ant have 2 choice



⇒ Collision don't occur only in 2 condition

1) when all ant move in clockwise direction
2) " " " anticlowise "

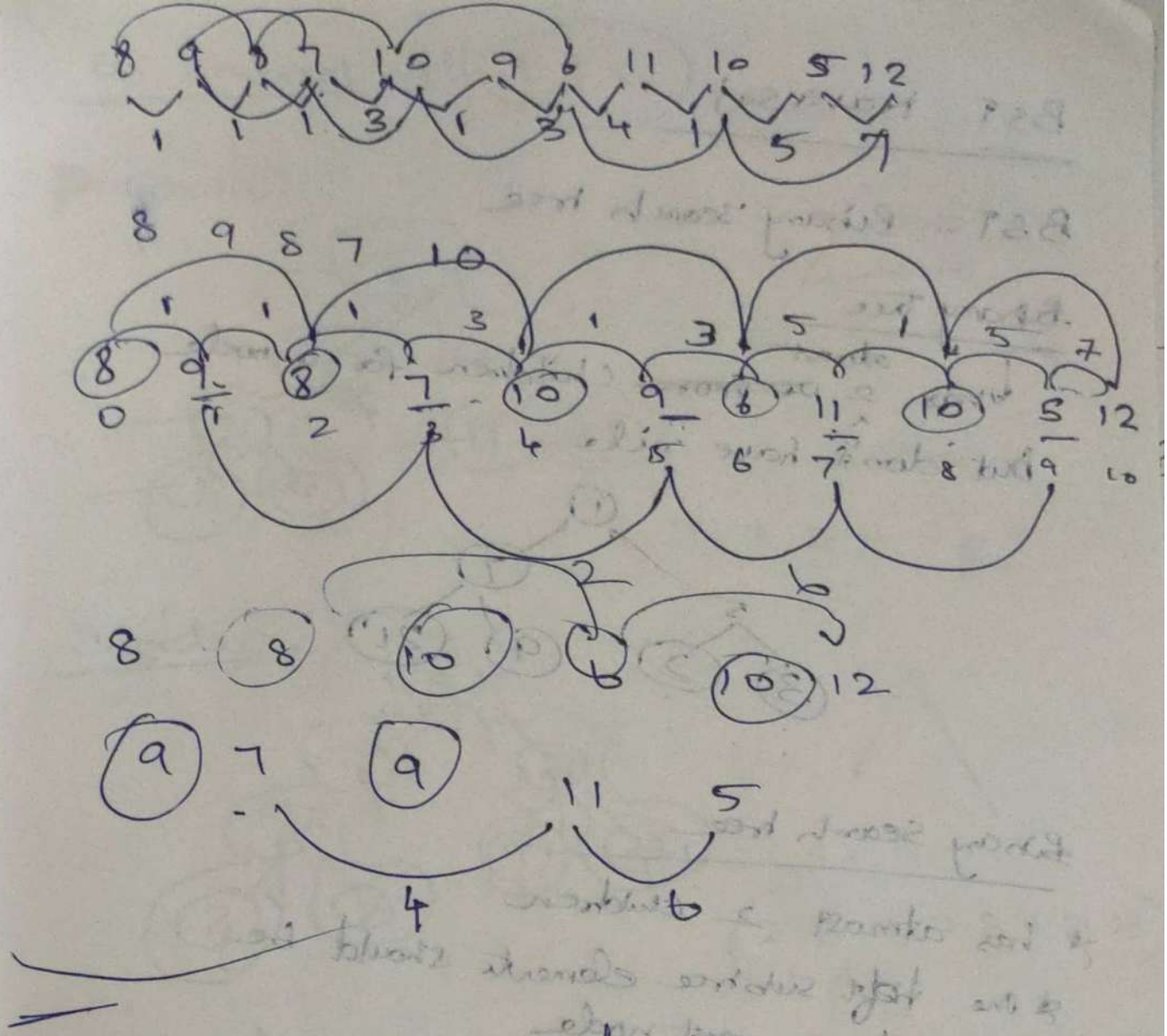
so no. of way all ant can move is



⇒ no. of way they can collide = 8 - 2 = 6 way

Probability that they will collide = 6/8 = 3/4

" " will not " = 2/8 = 1/4



Stack	Queue
★ FILO	★ FIFO
★ Only can insert in top	★ Can insert in rear side
★ Top() Peek() pop() push() isEmpty()	★ Can delete from front side ★ Insertion in queue is called enqueue ★ deletion " " " dequeue
★ is a class	★ is a interface ★ Types - Simple queue - Circular " - priority Queue - Dequeue

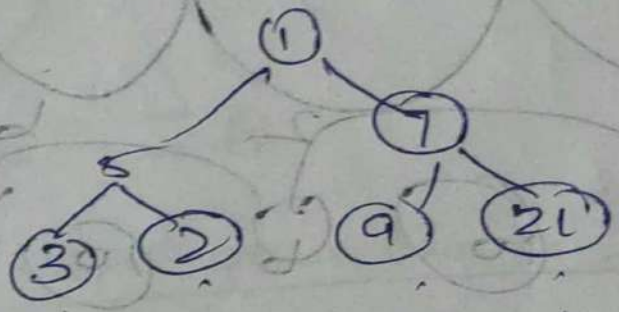
BST traversal

BST - Binary search tree

Binary tree

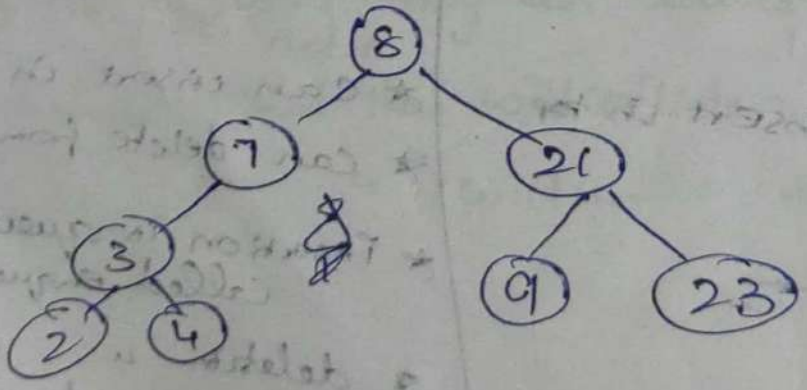
↳ has atmost 2 ~~or more~~ children for a node

But don't have rule



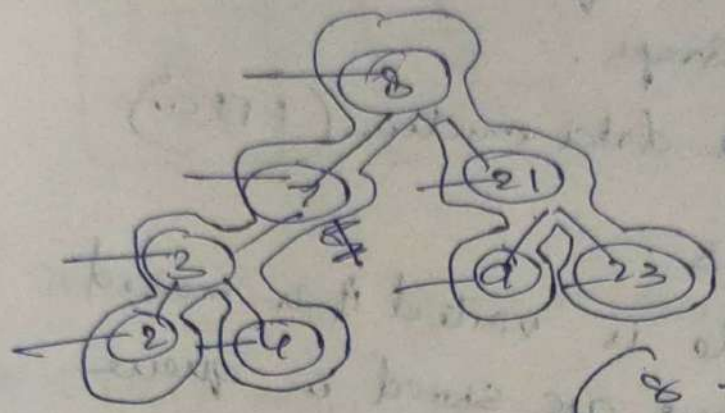
Binary Search tree

* has atmost 2 children
* the left subtree elements should be less than root node
* right subtree elements should be greater than root node



BST traversal all (DFS)

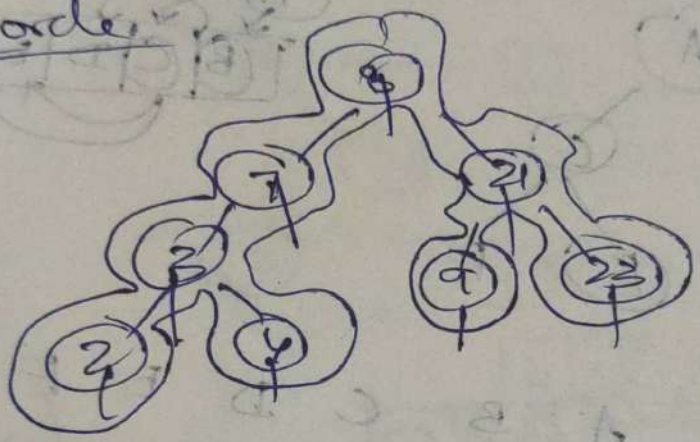
Pre order



Pre order

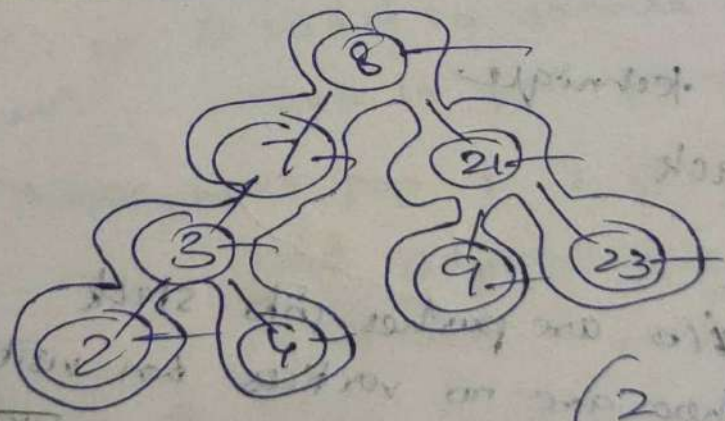
(8 7 3 2 4 21 9 23)

In order



(2 3 4 7 8 9 21 23)

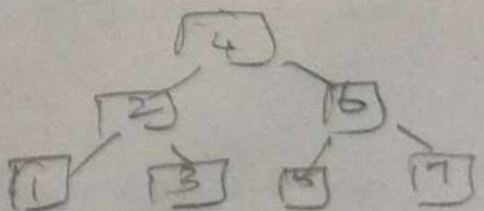
Post order



(2 4 3 7 9 23 21 8)

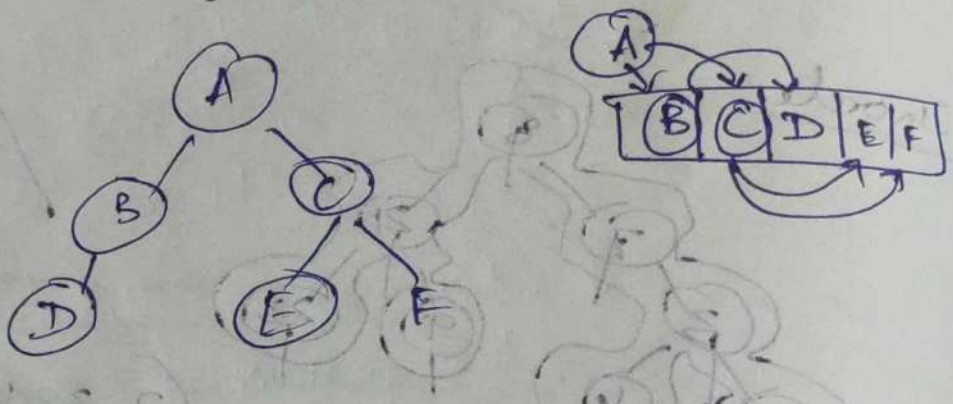
B tree

1) Self balancing tree
2) nodes can have more the 2 children
3) same like BST
4) all leaf node should be in same level
5) no empty subtree
6) Its height should be as low as possible



BFS (Graph)

1) vertex based technique used to find shortest path in Graph.
2) It uses Queue data structure (FIFO)
3) Slower than DFS
4) Once one node is visited it is marked and its adjacent are stored in queue

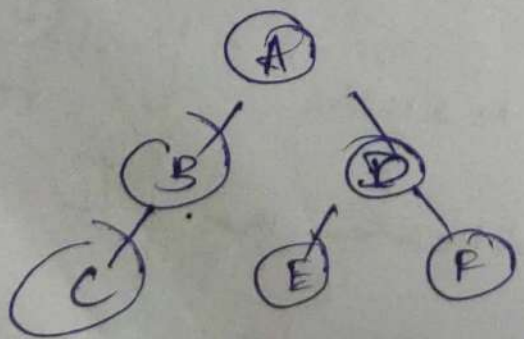


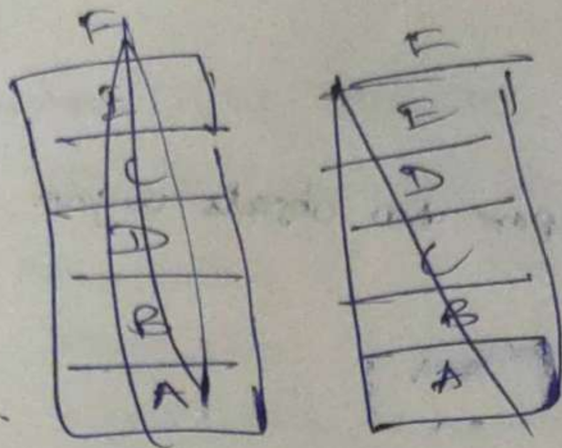
Output A B C D E F

DFS

1) Edge based technique
2) It uses stack
3) 2 stages

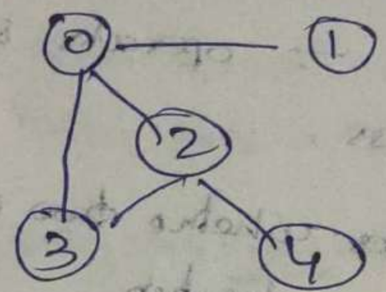
1st vertices are pushed into stack
2nd If there are no vertices then visited vertices are poped.

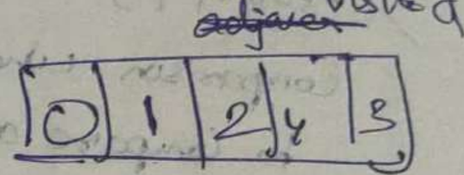


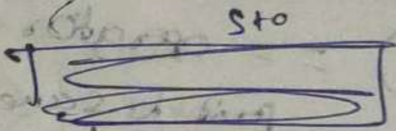


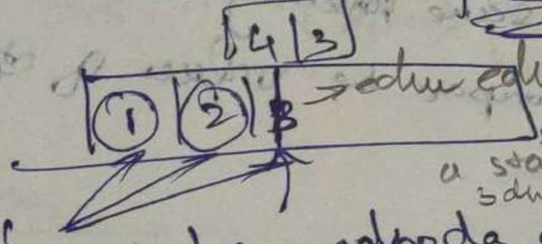
* Unlike tree graph may have cycles

* a node may be visited twice









edhu eghu ku enga irunthu pop aguthuno ith stack

a stack la last in 1st out

3 dhu last so 3 dhu 1st poposu

1st 0 visted so adhoda adjacent node Stak la push pannu

* next pop 1 & put is visited add adjacent of 1 to Stack as there is nothing leave

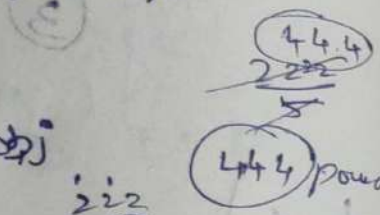
* take 2 & pop 2 oda adjacent are 3 & 4. 3 and 4 mattum push

* 4 ku ah pop then 3 ah pop

so) 0, 1, 2, 4, 3

== & .equal() :

1) Both are used to compare two objects in java
2) == is a operator
3) .equals() is a method.
4) == operator compares reference or memory location of object in a heap. whether they point to same location or not
5) .equal() method is used for content comparison whereas == operator is used to compare address.
6) == operator checks whether two obj's point to same memory location
.equals compare the values of obj



222
50
000
110
1000
444
10656 Pounds

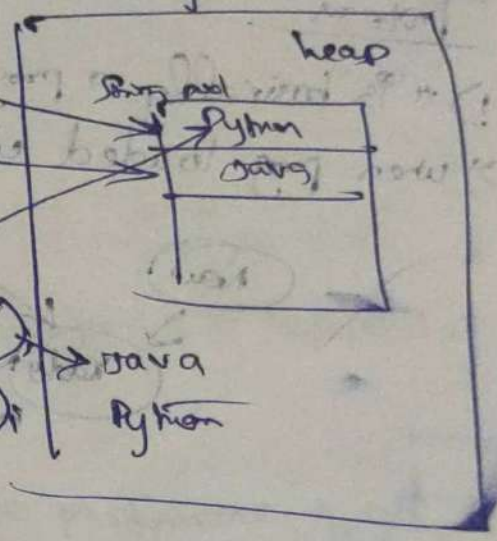
String pool

1) String pool is located in heap where String literal is stored.
2) whenever we create new string it create lives space in heap, more string creation leads to less performance

3) ~~Python~~ JVM reduces the space used by String by checking whenever we create a new string "" & check for that string in String pool if already present it returns the reference of that string, if not present new string created a string pool.

Difference between creating a string

like String str1 = "Python"
String str2 = "Java"
String str3 = "Python



~~X~~ V.I

String str4 = new String("Java")
, str5 = " " ("Python")

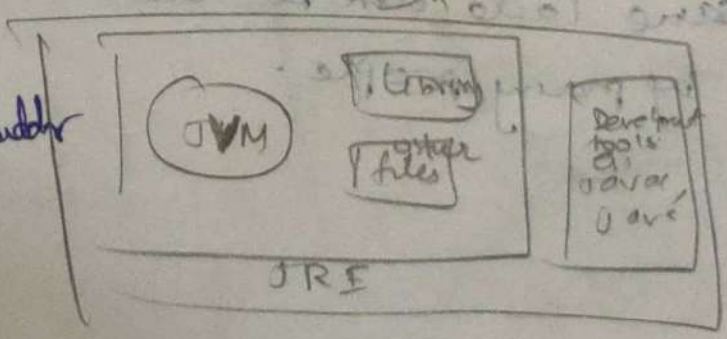
Java
Python

OS Continuation

Virtual memory

1) Gives illustration having a large main memory,
2) If the main memory is full it borrows some memory from the secondary memory.
3) paging & segmentation → not fixed size
   ↓ fixed size
   used page table
   store phy & logical addr

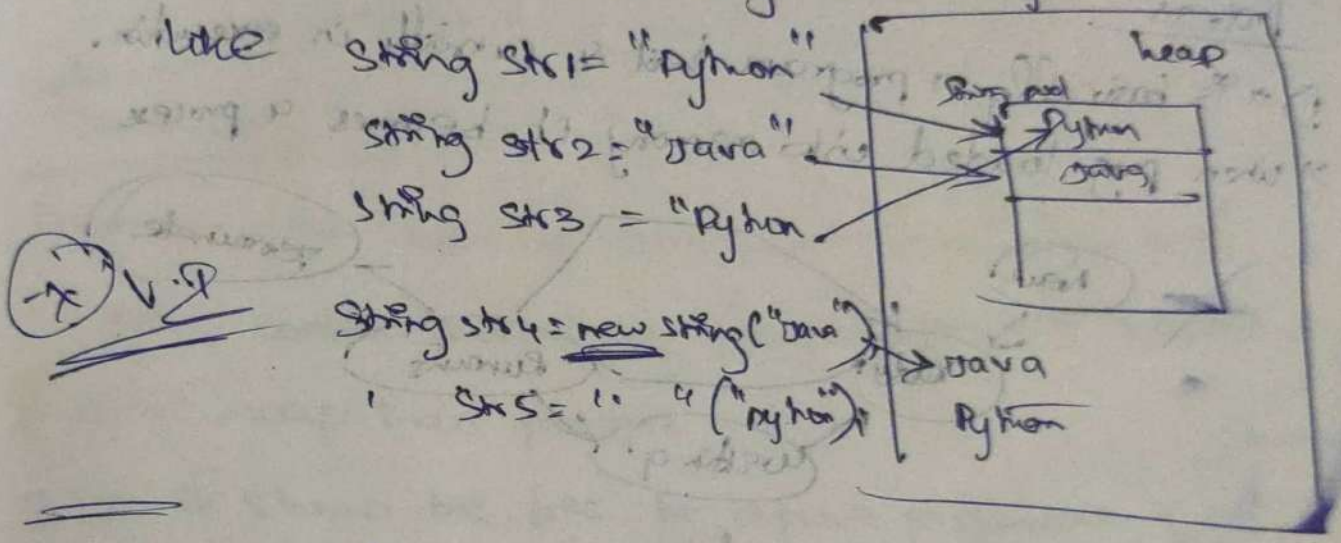
secure



3) ~~When~~ JVM reduces the space used by String by checking whenever we create a new string it checks for that string in String pool if already present it returns the reference of that string, if not present new String created in String pool.

Difference between creating a String

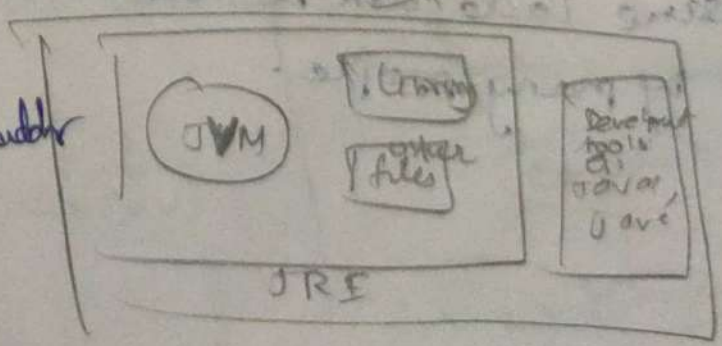
like String str1 = "Python"
String str2 = "Java"
String str3 = "Python"



OS Continuation

Virtual memory

1) Gives illustration having a large main memory.
2) If the main memory is full it borrows some memory from the secondary memory.
3) paging & segmentation → not fixed size
   ↓ fixed size
   used page table
   store phy & log ical addr

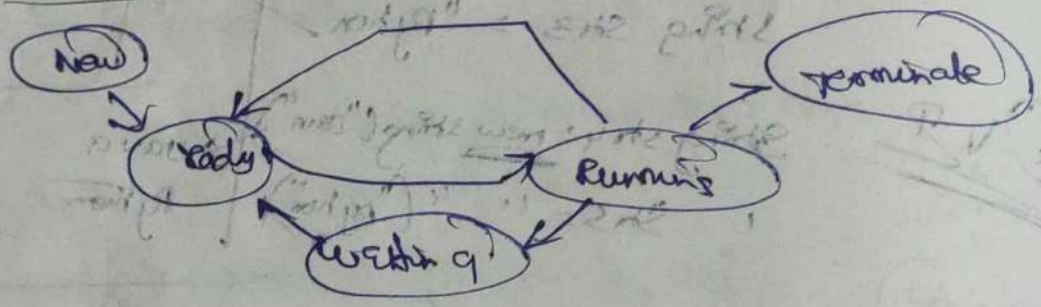


Threads in Os

1) It is a lightweight process has its own stack & make use of shared resources.
2) Many threads form a process
3) a thread is composed of PC, thread id, stack & registers
4) Reduce time in context switching & Improve parallelism
5) Threading allow to perform multitask simultaneously to increase performance.

Process

1) It is basically a program that is currently in execution.
2) when prog loaded into memory it becomes a process



New → process created
running → CPU executing process
waiting → process can't run because it is waiting for an event to occur
ready → resource available ready
terminate → process finished.

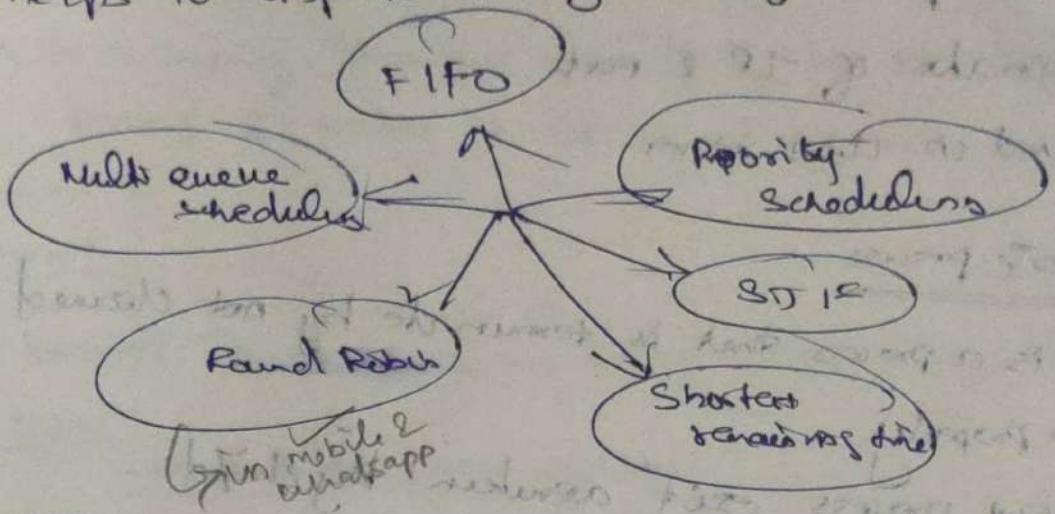
Reentrancy

↳ same code can be used multiple times by many people.

Preemptive scheduling → will interrupt process in mid
Non-preemptive → will not interrupt process.

⇒ Scheduling

* helps to improve utilization of CPU resources



Thrashing in OS

1) when CPU performs less work / performance and more context switching / paging switching happens. more swap than process

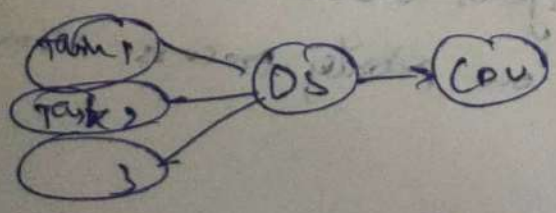
2) so it should be free to improve performance of OS.

Asymmetric clusters

* " In that only 1 node is hot stand by
  Other nodes perform diff application
* increas performance

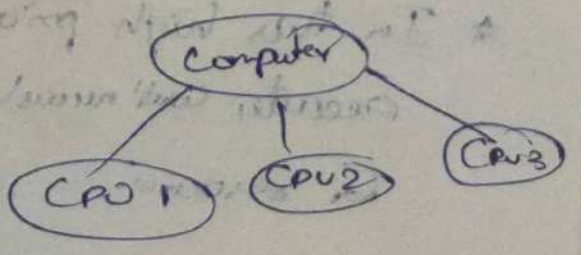
Multi tasking

* CPU perform any tasks simultaneously by context switch



Multi processing

* allows CPUs to work together in computer



Sockets

1) are end point for IPC
2) Combination of IP & port
3) used in client server

Zombie process

1) It is a process that is terminate but not cleaned up properly
2) Parent process exit when by child without when.
3) It will not hold any memory & it is dead but yet exist.

Cascading termination

1) an solution for zombie process
2) if the parent class/process terminate its child should also terminate.

Starvation

* When we use priority scheduling / SJF starvation can happen.
* In this a process waits/starves for resource for long period of time
* In this high priority gets resource & executes continuously & low priority process starves for resource.

Aging

1) solution for stravation. Why priority inversion - stravation priority inheritance - aging

2) It simply increase priority of low priority process & makes it use of resources.

Kernel

1) kernel is a interface bet. software & hardware

2) types

- monolithic kernel
  - is an Os architecture supp all comp
- micro kernel (small size)
  - supports only minimal res & compon
  - large size

Time sharing sys

→ allows many user to access the same resource at same time

Context switching

1) It holds the running of one process & saves it state

2) Then execute more priority process once it is done the holded process will start executing

3) Makes use of same resource

Deadlock

Explain → is a situation where no process executed
are blocked & not
- infinite wait
- starvation

Prevention

Mutual exclusion
Hold & wait
NO Pre-emption
Circular wait or Resource wait.
semaphore

Belady's anomaly

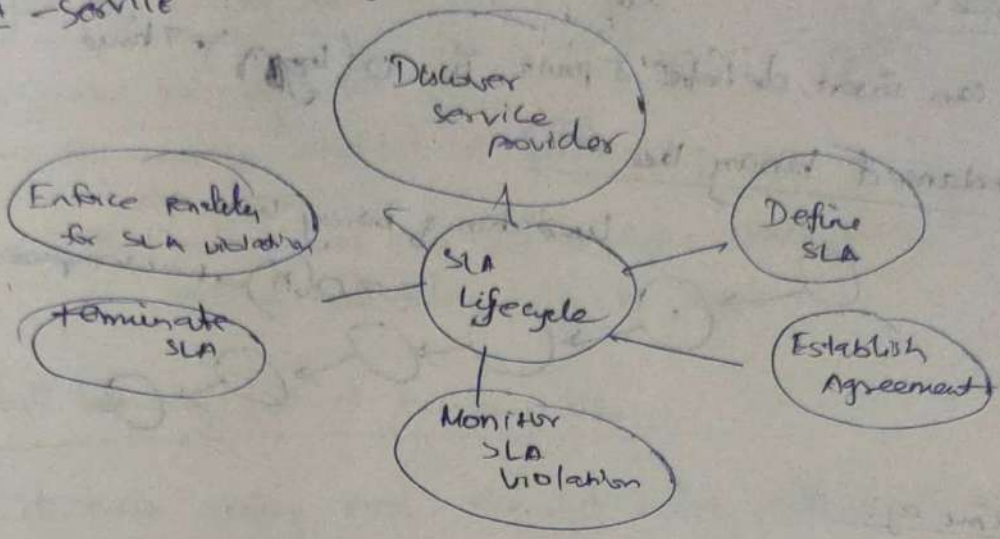
* Data loaded as chunks → refered as pages
* each pages loaded to fixed frame
* as frames ↑ increase the page faults also increase.
* FIFO

AppView X

1) loadbalancing
2) Blockchains, cloud ADC
3) Certification generation
4) Public key generation
5) $30 funds → 2019 fundraised from Brighton Park Capital it is started.
6) Rabobank - Certificate generator
7) Cybersecurity oriented.
8) 2 → products (CERT + AUTOMATION) (certificate management lifecycle)
9) 2009 → started

ADC → automate your Application Lifecycle Management. (ALM)

SLA - savile level Agreement



Newyork - headquarters

4 Countries

Integer.parseInt()	Integer.valueOf()
* only take string as arg	* Take both integer, & string as args
* ~~and~~ is a method belongs to Integer class java.lang.package	* Is a static method belong to java.lang. package
* Take only valid string as parameter & parse it to primitive data type it	* if string is valid throw error
* other than string if any passed as arguments it gives error	* it can even take Char as parameter but gives Unicode value -128 to 127

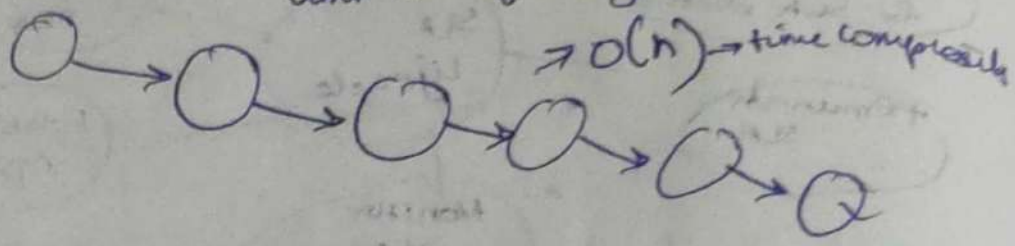
V.D

Trees

1) we can insert, delete, & print in $O(\log n)$ → time

Unbalanced binary tree

limitation of binary tree



→ $O(n)$ → time complexity

- Real time eg:

1) File systems
   Node → parent child relationship file system
2) Database → fast retrieval data
3) Routing / Algorithm / networking
4) Solve complex mathematical prob
5) Compressing files

Trees are Directed acyclic graphs

Binary tree

```
Class Node {
    int value;
    Node left;
    Node right;
}
```

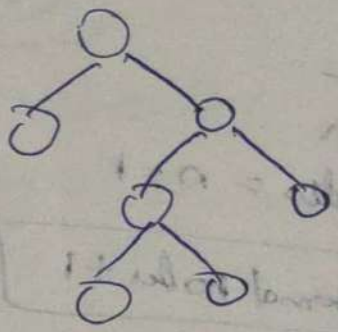
Properties

1) Size = total no. of nodes
2) child & parent
3) siblings
4) Edge → line joining node
5) height → max no. of edge bet node we want to find & leaf node
6) last node is called left -
7) level → subtract height of root node - height of node
8) root node level always 0

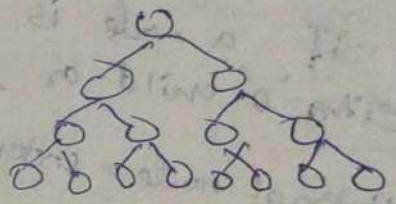
Types of Binary tree

1) complete binary tree
   - → All levels should be full
   - → The last level can be incomplete but should be filled from left to right.

2) full binary / strict binary tree
   - In this every not should have either 0 child or 2 children but no one child

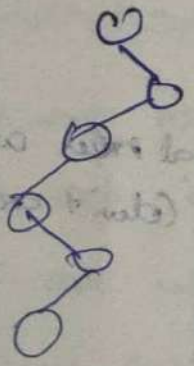


3) Perfect binary tree
   - All levels are full



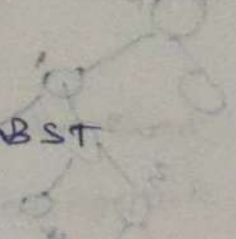
4) height balanced → Avg height o(log n)

5) Skewed binary tree → every node has only 1 child



height (o(n))
like linked list

6) Ordered binary tree
   - Every node follow a property : eg : BST



1) Perfect B.T, height = h

total nodes = $2^{h+1} - 1$

2) leaf nodes in perfect binary tree = 2^h

3) NO. of internal nodes = $(2^{h+1} - 1) - 2^h$

total no of nodes — no. of leaf nodes

$= \boxed{2^h - 1}$

~~4) N — no. of leaves~~

In strict binary tree

No of leaf nodes = n

NO. of internal nodes = n - 1

$\boxed{\text{NO. of leaf nodes} = \text{internal nodes} + 1}$

in this if a node is add it will have either 0 child or 2 child so

internal node = internal node + 1

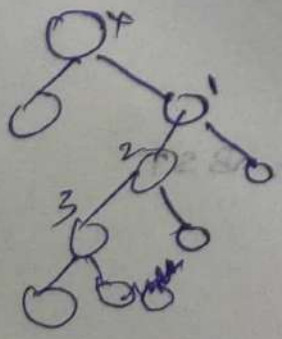
leaf " = leaf node + 1

Degree

" is either 0, 1 or 2

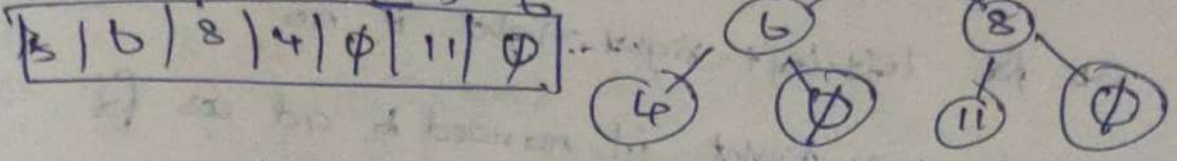
No. of leaf nodes = 1 + NO. of internal nodes with 2 children (don't exclude root node)

$\Rightarrow \boxed{4 = 1 + 3}$



Implementation

1) Linked representation (more efficient)
2) sequential → using array.



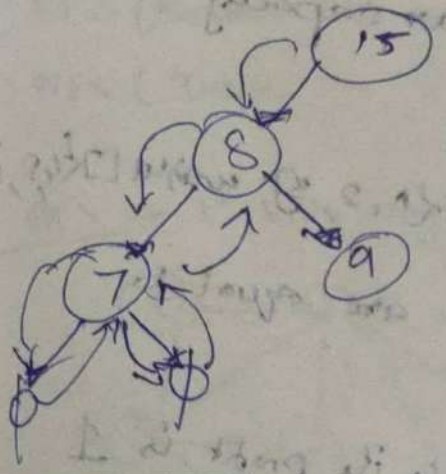
$N = 2^h$

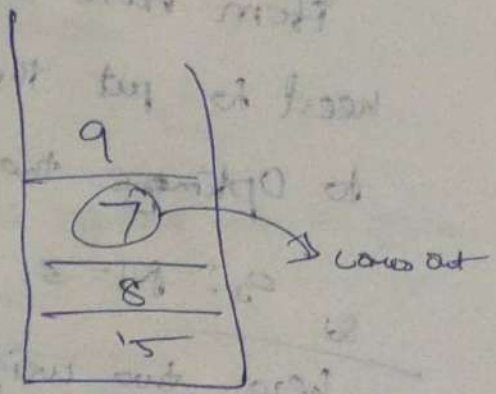
* log ⇒ $\log N = h \log 2$

$$h = \log(N)$$

---

recursion in BT





---

for insertion

1) 1st insert root node
2) Then ask it user want to insert left if yes insert
3) Then ask whether it want to insert left.left
4) If yes insert e then ask again it user to insert left.left.left
5) if yes insert then ask it want to insert left.left.left.left if no go backtrack

- if u want to add left.left.left.right yes add
  Or go back to left.lef.right if yes add.
- Then as u want to ad left.left.right.left if no
  as for left.left.right.right, if no
- Go back left.right if no need to ad as for
  just root-right if no just leave.

---

knapsack problem

In this we are given N items where each item have their weight & profit & we need to put that in a bag of max capacity W to Optimize the profit

eg: N=3, W=4, Profit[] = {1, 2, 3} weight[]={4,5,1}

Sol

here two weight of items 4 & 1 are equal to or less than bag capacity W

but if u put item of weight 4 its profit is 1

but " " " " 1 " " 3

so max profit is (3)

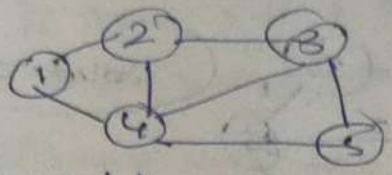
if u can't put anything in bag then profit 0

weight only 1 dha poda mudium its profit (3) namma vera edhavum add panna mudila engaa because it any value weight will be so not possible.

P07MC420DS

NEWSD

Graph representation



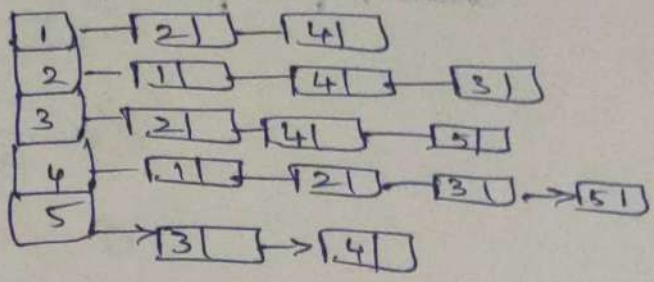
undirected graph

Matrix representation

	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	1	0
3	0	1	0	1	1
4	1	1	1	0	1
5	0	0	1	1	0

(5×5) space $O(n^2)$

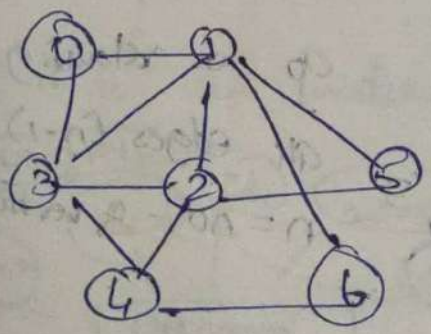
Adjacency list



Graph Traversal

1) BFS (Breadth First search) → level order → Queue
2) DFS (Depth " ") → stack

BFS → 1st visit any Or starting node & push adjacent node in queue



start from 0

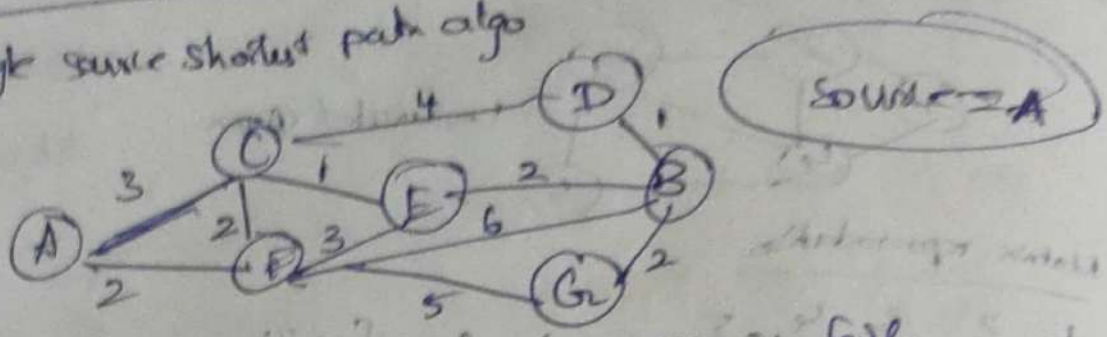
0
~~0~~ 1 3
~~1~~ 3 5 6 2
~~3~~ 5 6 2 4
~~5~~ 6 2 4
~~6~~ 2 4
2 4
~~4~~

0 1 3 2 6 5 4

~~1~~ ~~3~~ ~~2~~ ~~6~~ ~~5~~ ~~4~~

Dijkstra Algorithm

1) Single source shortest path algo



Source = A

2) If graph has -ve cycle weight then it fails

Shortest path from A to C = 3

F = 2

E = 4 (3+1)

D = 7 (3+4)

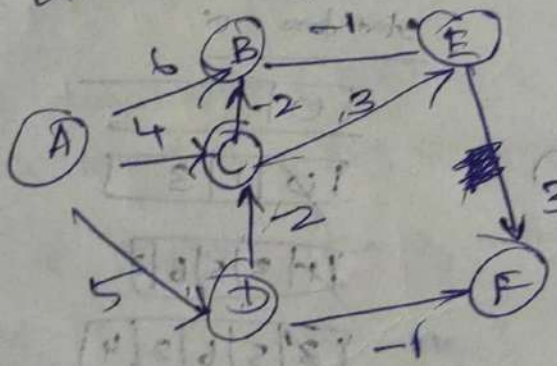
B = 6 (3+1+2)

G = 7 (2+5)

Bellman-ford algo

1) It is slower than Dijkstra's algo

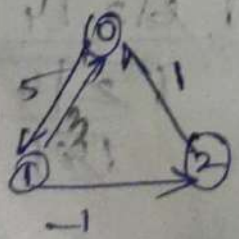
2) but can use negative values



go on relaxing all edges (n-1) times

n = no. of vertices

3) It can start from any vertices



source 2

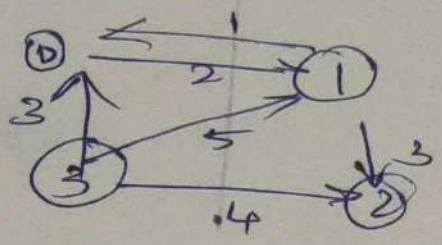
V = 3

E = [[0,1,5][1,0,3][1,2,1][2,0,

Floyd Warshall algo..

1) Used for both +ve & -ve weights
2) Used to find shortest path for every pair of nodes
3) Multisource shortest path.

* Nothing but using Dijkstras' algo for each node
* Floyds Way to DP

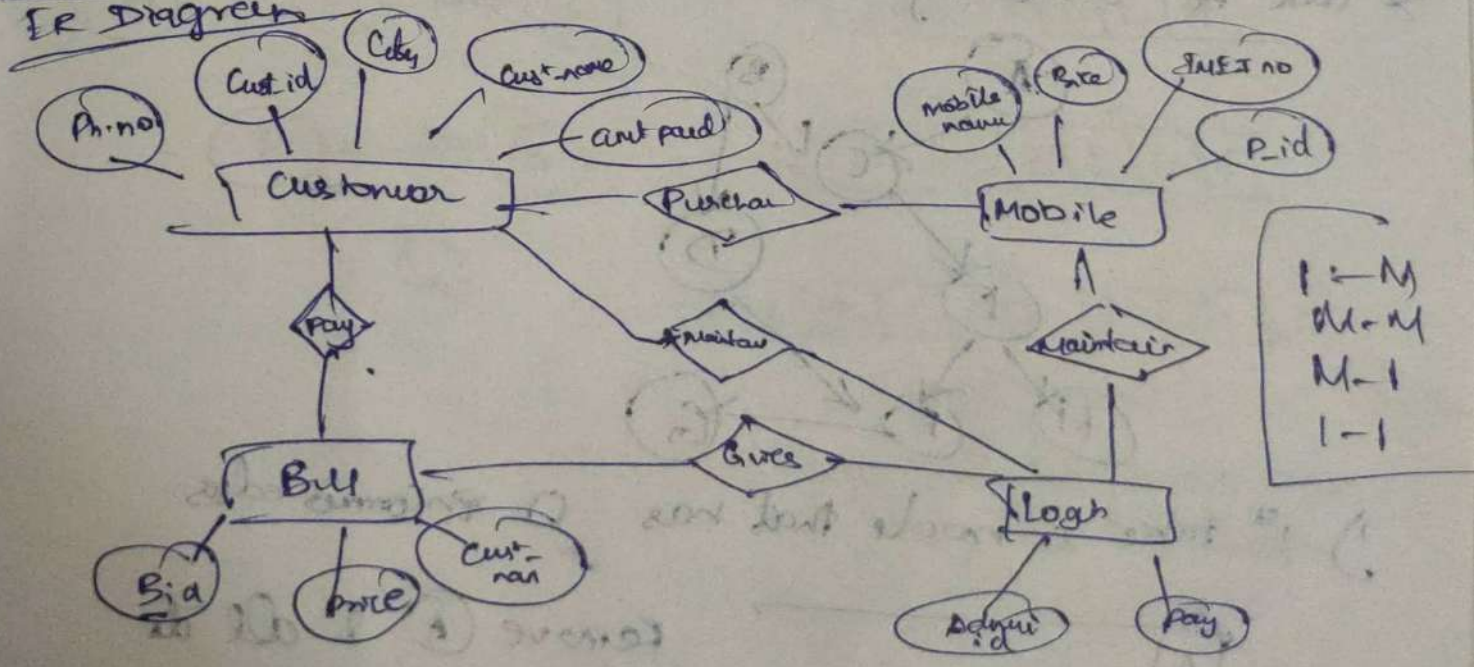


Same node cost always 0

	0	1	2	3
0	0	2	∞	∞
1	1	0	3	∞
2	∞	∞	0	6
3	3	5	4	0

If it is an undirected graph makes it dual of

ER Diagram



Entity attribute relation

Aptitude

1) Link - login
   hire - platform, , practice
   enter data, resume
   20 qn - 30 min → give score

2) tech
   20 qn - 20 min
   give mark

3) Coding
   Program - 30 min

4) GD

5) Final interview

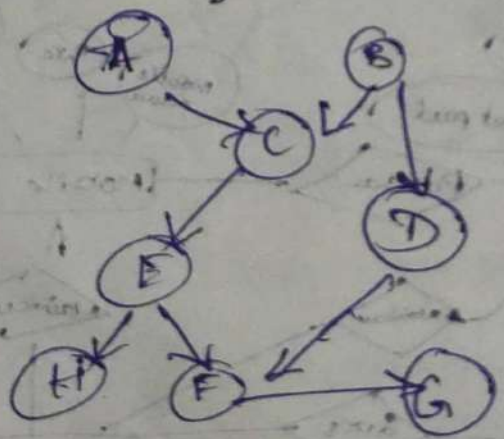
6) Final meet → offer letter

Skillset assessment application or own

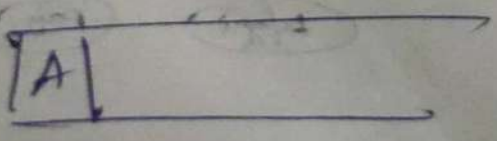
1) Share link to placement officer
2) technical
3) final

Topological sorting

* Can be performed in graph that contains DAG (Directed Acyclic Graph)
* uses stack & boolean array
* can be done by both DFS & BFS (kahn's algo)

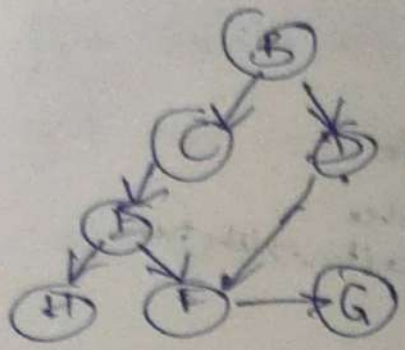


1) 1st take a node that has 0 incoming edges

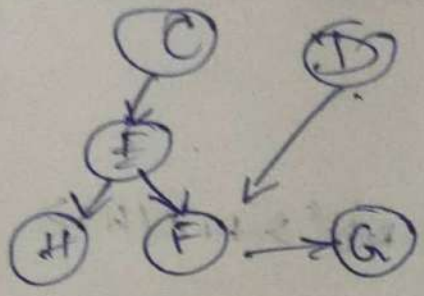


Remove (A) & all its outgoing edges

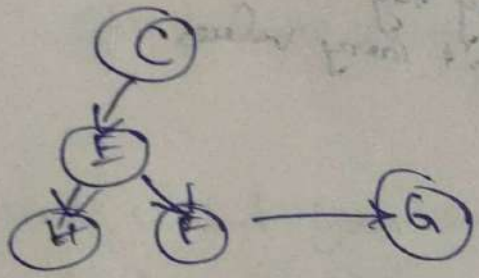
Chose (B)



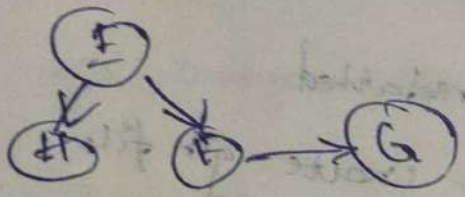
| A | B |



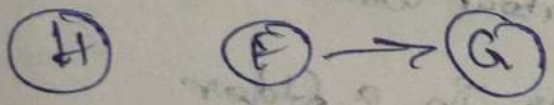
| A | B | D |



| A | B | D | C |



| A | B | D | C | E |



| A | B | D | C | E | H |

| A | B | D | C | E | H | F |

| A | B | D | C | E | H | F | G |

Final ans

There can be more ans for this Graph

HTML

<mark> → highlight things

<pre> → give the same format as we type

<del> → write what is inside & strike — Do it → like this

<ins> → inserted → with underline

<sub>

<sup>

<h1 style = "color : blue;"> hi </h1>

<h1 style = "color : blue; background-color : Red;"> hi </h1>

<img> <link> don't have closing tag because they don't carry values:
<area>

Git

Git → checks whether it is installed

Git init → initiate the git & create .git file

.git file → contains all info about commit, logs etc...

git add . → add all files in a folder

git status → helps to check what all files has been added & what all are not added

if u modify any file then u need to give git add.

ls → list the content & files in folder after u upload using git add.

ls - for linux
dir - in windows

u need to commit

git commit -m "first commit"

git remote add origin ". . ".git

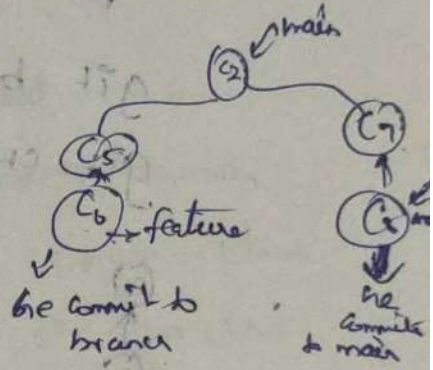
git push -u origin [branch name] / git push origin master
Master

git stash → saves dd commits

git commit main → ethula commit panna adhu ;*
correct ah erukanum illana edhu
vhde project ah affect pannu

so we need to use branch eg:

git commit branch
feature



once u find your code is correct & can be added to main then we

git merge feature

git log → displays all commits with time & date

Using fork u can clone to yourself

eg: if u get a code of others u should not be able to alter their original code because it will cause the author prob so we can clone it to our repository & change/alter it

after fork go to CMD give

git clone then then that link which is in code

git clone code

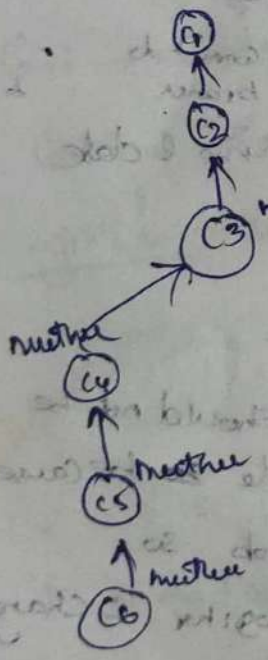
Where we added others folder is called upstream url

Using vi filename.extention
↳ open that file
to insert press i
write anything save

Cat filename.extension
↳ display content in that file

Always when u are creating or altering some
don't commit to main branch instead create
new branch later u can merge with main

git branch mathu
git checkout mathu → this means the head will now point to mathu. each time nama commit panura apo edhu dha more agum



Epaium new seperate developer avagaloda seperate branch la ellam change/modify create panuvaga

After complete the main branch owner kita na edhu puthusa add/update paniruka edha unoda main oda merge paniko nu sollaga

Pull request

Pull request → pannathuku Owner will review the individual contributions than add it to master branch

`git push origin muthu` → adhu edhukuna nama namalodathu master ku anupa use agum

than click `Compare & pull request` from your git

this means namaloda project author ku message anupum to give authorization to merge.

One branch only one pull request

This means epo marubadiyum nama namaloda local project la edha add panitu, adha upload panna adhu comit 2, 3 .. etc va dha author ku pogum so it will be difficult to review code.

So new feature add panna add it in new branch apo dha new pull request kudukamudiyum author.

If we want to delete above commit copy the Commit code 1 below it and

git reset 4ae3c5cef38 2

start temporary saving

epo namakitta 2 branch esuthu total ah na 4 changes pannanum 1st branch la 2 change panitu na 2nd branch la nedhuthuvo panitu commit pannanum but 1st branch la enoda thirst ana adhu u have uncommited change sollu so name stash use pannala

Pull request → pannubathu Create / owner will review the individual contribution than add it to master branch

`git push origin muthu` → adhu edhukuna nama namalodathu master ku anupanum agum

than click `Compare & pull request` from your git

this means, namaloda project author ku message anupum to give authorization to merge.

One branch only one pull request

This means epo marubadiyum nama namaloda local project la edha add pannitu, adha upload panna adhu commit 2, 3 .. etc va dha author ku pogum. So it will be difficult to review code.

So new feature add panna add it in new branch apo dha new pull request kudukanumdiyin author.

If we want to delete above commit copy the Commit code 1 below it and

`git reset 4ae3c5cef382`

---

start temporary saving

epo namakitta 2 branch exists total ah na 4 changes pannirukom 1st branch la 2 change panitu na 2nd branch la nedhituvo panitu commit pannama but 1st branch la enoda shirt ana adhu u have uncommitted change so name stash use

/ - library
- Package
- Drivers

(C#)
eg: Winforms
WPF
MAUI

(input)
(output)

Future apps
- Flutter
- Ionic
- Xamarin

- Sqlserver
- Postman
- Git
- IDE - (Vs
- Net

after changes pannadhu we want to force push this

git push origin [illegible] -f → to push to our main branch

---

author oda dha nama permission vagidha pull request author alaum nama files kudukuro adhe mari author alaum nama files alter panna mudiyadhu for this 2 way

[fetch upstream] → button

Or manually

git checkout main — [upstream na author odadhu]

git status

git log

git fetch --all --prune ①

git reset --hard upstream/main ②

git log

git push origin main → ③ → apo dha push agum

} to make local system even with author

---

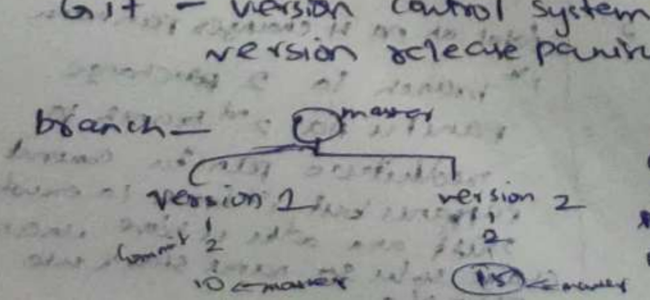
git pull upstream main

[fetch & merge panuradhunala nama loda Branch ah author oda update kuu etha mari vechukalam]

1st create paruva apo master branch matudhaenuku

Git – version control system – edhukuna epo oru app la mox version release panirupage like 1.2.7 but namaku epo 1.1 oda code venumna nama git la enk

branch –



epo namaloda developers version 1/2 la work panuvaga if version1 oda 10th commit ah merge pannanumna adhu master aagum / or version 2 oda 15 commit ... master aagum

Create another new branch

Epadi panni badhu again, adhu head la onndhu
Adha branch agum

Pull – server la irukura code da nama edutthu use panna

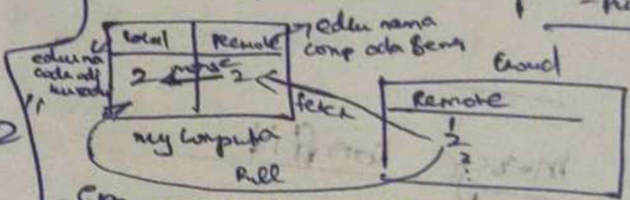
git branch temp
git checkout temp → adhu kullan pogg like cd

touch 1
git add .
git commit -m "1"
touch 2
git add .
git commit -m "2"
touch 3
git add .
git commit -m "3"
touch 4
git add .
git commit -m "4"

Differnce between Pull (request) & fetch

Pull (request) panna adhu master ku poirum as review adhu ok na avanu merge panniruvanu with main branch

actuall epadidhu ondhu | Fetch + merge = Pull



Epo pull kudutha adhu mothama sync aagum like epo cloud la irukura new files an change ellan remote & local ku vandhu & if change in same file of local apo conflict varum.
Fetch kudutha cloud oda remote la irund mycomputer oda remote ku mathu varum.

→ edhu ellathaum single commit pannanum na last commit ah copy pani

git reset 1235675001
git stash
then git commit

(Or)

pick & squash

git rebase -i 5e576a237

Esc : x → veliya varum

it display

pick d9da124 1
pick da7f20 2
pick - - 3
pick - - 4

Which ever is pick then other s will be merged with previous.

Edha ellam s num mathuna pick la irukura 1 ah commit panna ellame ore commit la commit aagum

git log kudutha terium ellame commit aenaikum

if namaku 1st 3 pothu oru commit:la na,

pick 5768 32f. 1
s . 2
s 3
pick 4

fash ah

git reset --hard E5768549238FCB
↓
commit no.

merge conflict

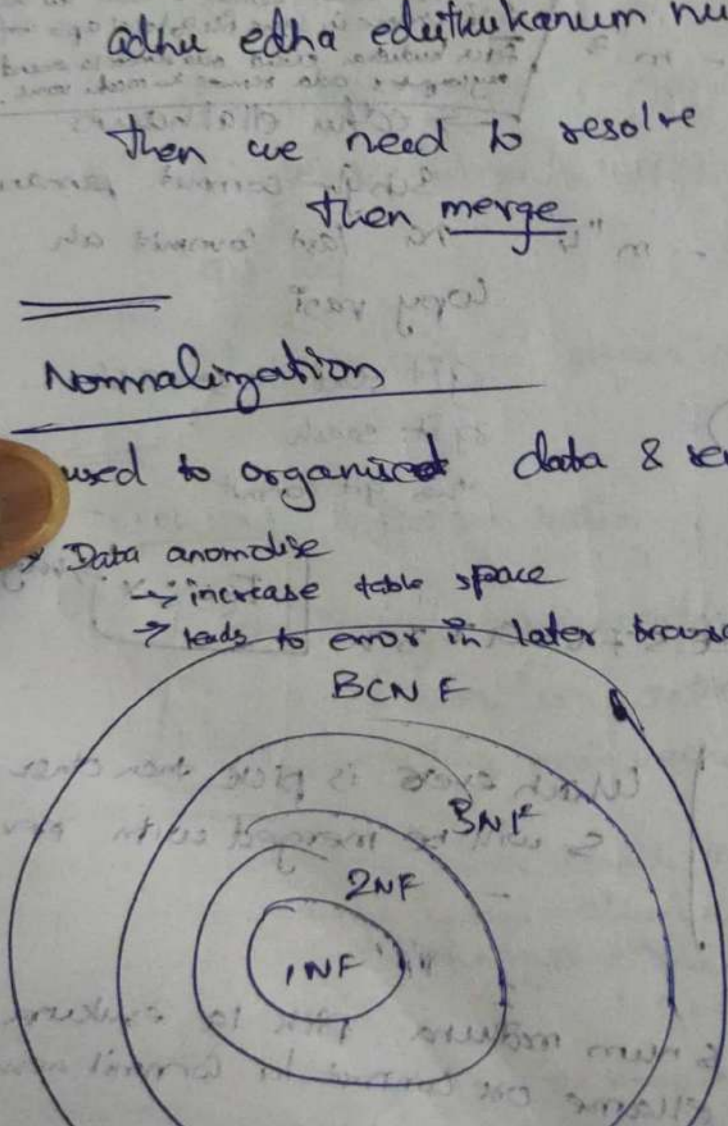
go rendu peru ore line number la change panna adhu edha edithukanum nu kekum

then we need to resolve manually & then commit then merge.

Normalization

used to organised data & remove data redundency

→ Data anomolise
 → increase table space
 → leads to error in later transaction



1NF

* remove repeating groups from table
* Create a seperate table for each set of related data
* Identify each set of related data with primary key.
* remove multiple values.

Employee ID	Employe name	ph	Salary
IEDU001	Alex	94430 .. 8500 ...	60,131
IEDU002	Barry	8760...	48,302
IEDU003	Clair	9159...	22900
IEDU004	David	6362.. 87500	81538

1) No shell should have multi values & it should be atomic

Employ ID	Emp Name	ph	Salary
IEDU001	Alex	94430 ..	60,131
IEDU001	Alex	8500..	60,131
IEDU002	Barry	8760..	48,302
IEDU003	Clair	9159...	22900
IEDU004	David	6362..	81538
IEDU004	David	87500..	81538

achieved atomicity

2NF

* should be 1NF
* should not contain partial dependency

Emp Id	Dep I.d	Office loca
1EDU001	EDT1	Pune
1EDU002	ED-S2	Bengaluru
1EDU003	ED-M1	Delhi
1EDU004	EDT.T3	Mumbai

both primary key always by office · 2 unique ids · 1 primary key dha irukum · this is composite key

both Emp id & Dep. Id are primary key
but office loca depends only on Dept. id
so this is partially dependent on primary key

Emp Id	Dep Id
1EDU001	ED-T1
1EDU002	ED-S2
1EDU003	ED-M1
1EDU004	ED-T3

Dep Id	office loca
ED-T1	Pune
ED-S2	Bengalum
ED-U1	Delhi,
ED-T3	Mumbai

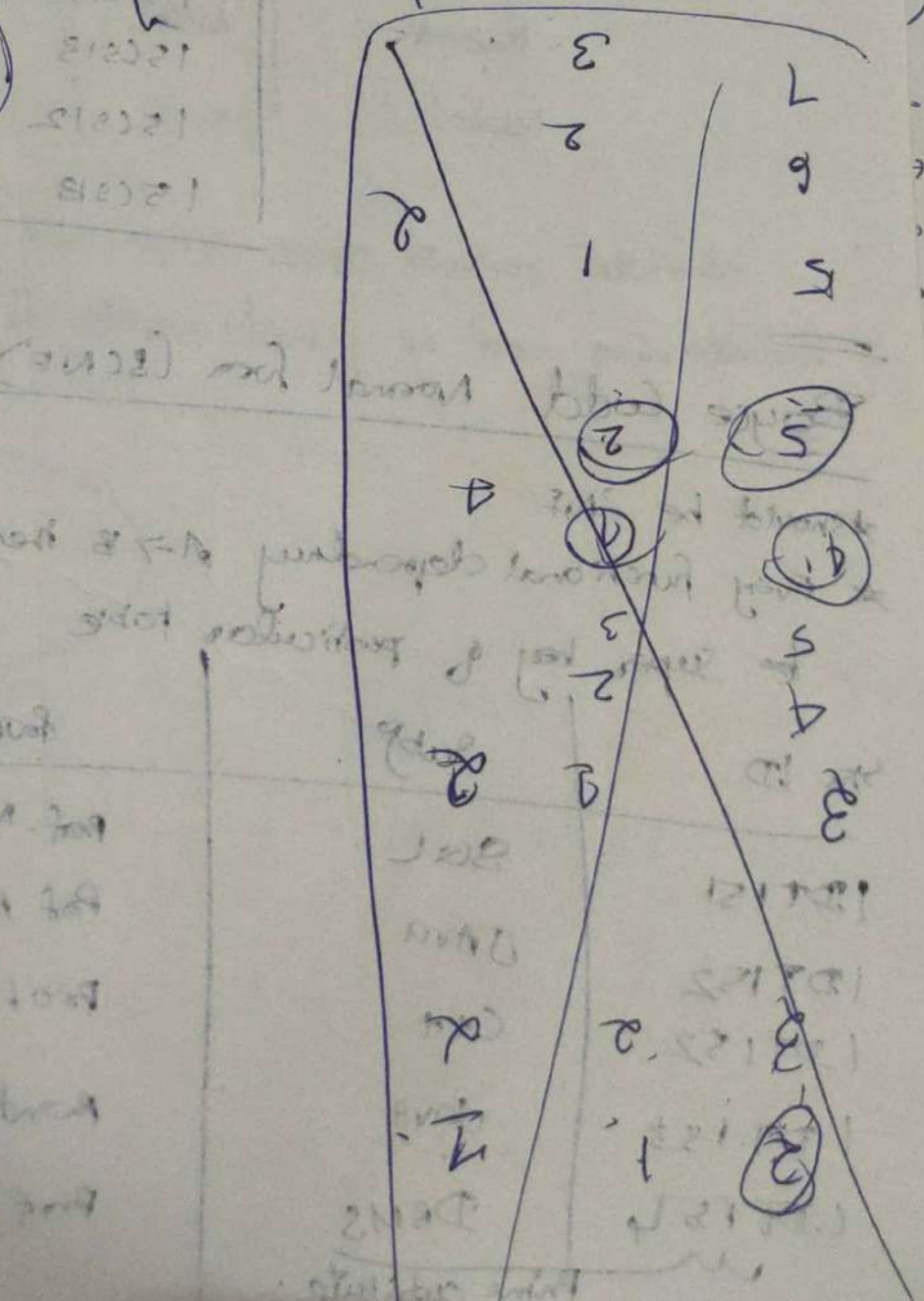
Enga enna solurgana 2 primary key values (composite key) erundhu ennoud colum adhula one id ah mattu depend agierundha adha mattu nama seperate pannanum

3 NF

* should obey 2NF, 1NF
* should be no transitive dependency for non-prime attribute

Student Id	Stu Name	Subj Id	Subj	Addre
1DT151	Alex	15CS11	SQL	Goa
1DT152	Barry	15CS13	JAVA	Bengalum
1DT153	Clara	15CS12	C++	Delhi
1DT154	David	15CS13	JAVA	Kochi

PK



We need to make table in which all columns are only dependent on primary key

Stud id	Stud Name	Subj Id	Address
1DT151	Alex	15CS11	Goa
1DT152	Bonny	15CS13	Bangalore
1DT153	Clair	15CS12	Delhi
1DT154	David	15CS15	Kochi

Subj Id	Subj
15CS11	SQL
15CS13	JAVA
15CS12	C++
15CS13	JAVA

Boyce Codd Normal form (BCNF) (B.C.N.F)

*should be 3NF

Every functional dependency A → B then A has to be super key of particular table

Stu ID	Subj	Professor
1DT151	SQL	Prof. Nisha
1DT152	JAVA	Prof Anand
1DT152	C++	Prof. kavitha
1DT153	Java	Prof. Anand
1DT154	DBMS	Prof. Lokesh

Prime attribute.

Stud ID	Prof. Id
IDTIS1	IDTPF01
IDTIS2	IDTPF02
IDTIS2	IDTPF03
IDTIS3	IDTPF02
IDTIS4	IDTPF04

Prof Id	Subj	Prof
IDTPF01	SQL	Prof. Mishra
IDTPF02	DBMS	Prof. Anand
IDTPF03	C++	" Kanthi
IDTPF04	JAVA	" Anand
IDTPF05	DBMS	" Lokesh

IN BCNF we will create another attribute to avoid fn dependency on non prime attribute

PTO

Counting Sort

enga nama sort panna poradhu kedayadhu kudutha array la each element ethana time repeat aagukuthu enanu array la store panna num

a =

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

First range kandupudi 0 - 9
edhula negative element no allowed

Create another array of sige range+1

~~count~~ int[] count = new ~~int~~ a+1] ← count[]

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

```
for(i=0; i<n; i++)
{ count[a[i]]++ ;
}
```

(x) Easy but takes times

```
for(int i=1 ; i<=k; i++)
{ count[i]+= count[i-1]
}
```

```
for (int i=n-1 , i>=0; i--)
{ b[count[a[i]]-1] = a[i] ;
}
```

```
for(i=0; i<n; i++)
{ a[i] = b[i];
}
```

or System.arraycopy(source, start, Destination, start, size)
(b[i], 0, a[i], 0, 1

sort original array   k=9 , n=17

0	0	0	1	1	1	2	2	2	2	4	5	7	7	8	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

then Count[] array la update pannanum like Count[i] += Count[i-1]

3	6	10	10	11	12	12	14	15	17
0	1	2	3	4	5	6	7	8	9

values is due position not index

then start from original array oda last to maintain stability

b = (array name)

0	0	0	1	1	1	2	2	2	2	4	5	7	7	8	9	9
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

time complexity
O(n+k)

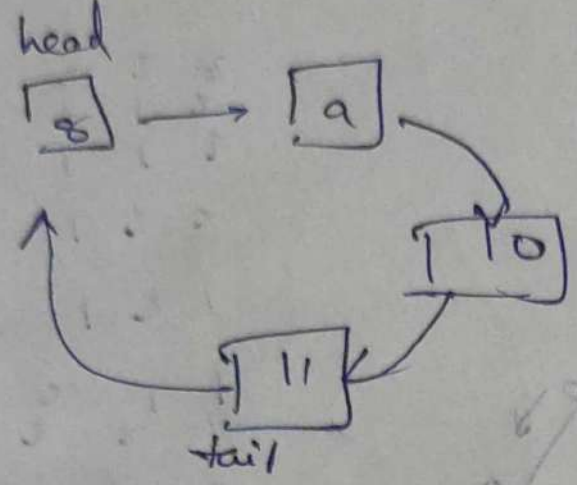
appo last element 9 apo Count[] las 9th index la 17 ouku 17-1 la nama 9 podanum en new array Create pannitu adhu

last la irundhu ovvoru value ah check pannu epo (9) na Count la 9th index poyi andhu value -1 oda index la andha pudhu array la andhu 9 ah podu same way all.

jenny lechoo

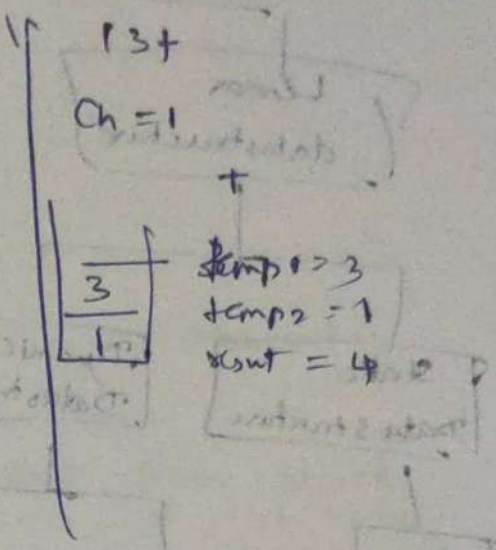
Circular Linked list

No one points to null until the list is empty.

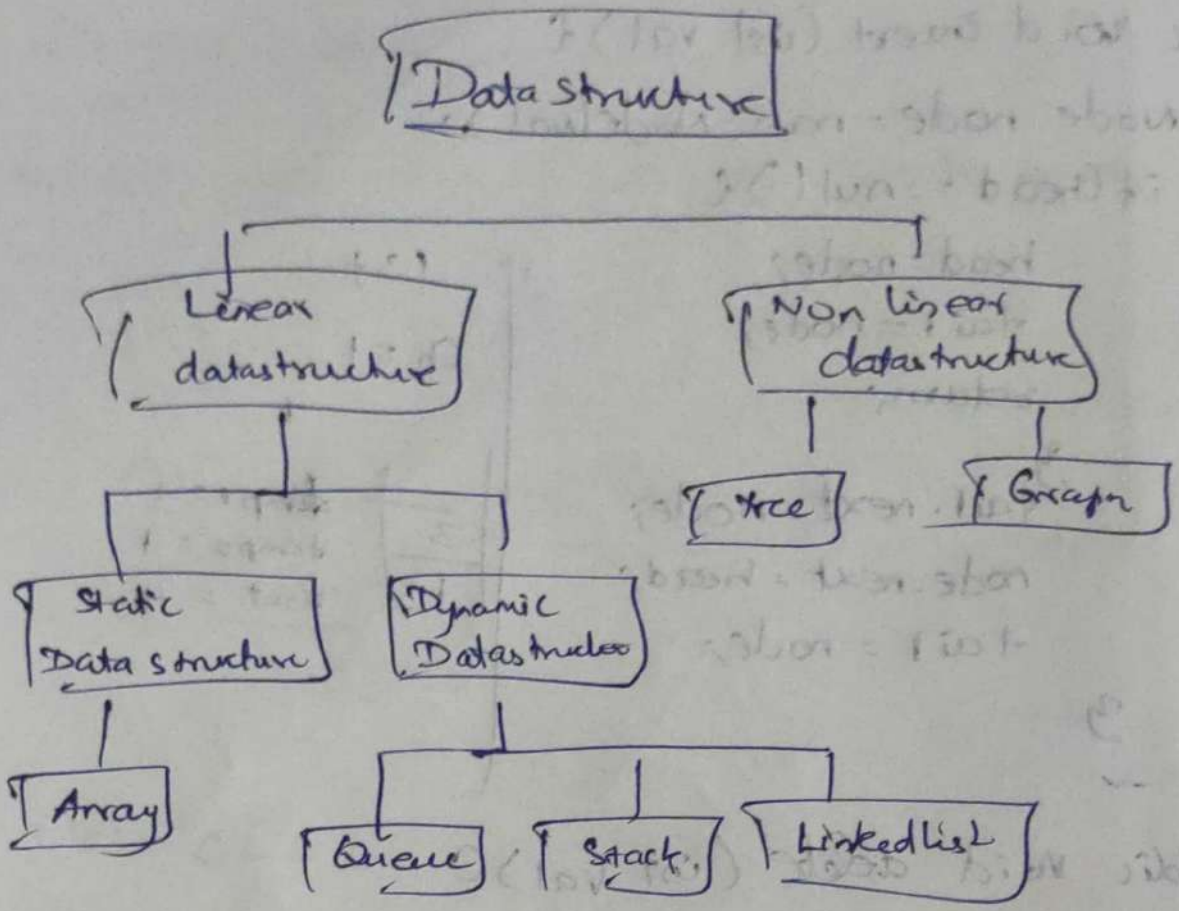



```
Class CLL {

    Private Node Head;
    Private Node tail;

    Private class Node {
        int val;
        Node next;
    }

    Public Void display() {
        Node node = head;
        if(head != null) {
            do{
                System.out.print(node.val + "->");
                node = node.next;
            } while(node != head);
        }
        System.out.println("HEAD");
    }
```

```
Public void insert (int val){
    Node node = new Node(val);
    if(head == null){
        head = node;
        tail = node;
        return;
    }
    tail.next = node;
    node.next = head;
    tail = node;
}
```




```
Public void delete (int val){
    Node node = head;
    if(node == null){
        return;
    }
    if (node.val == val){
        head = head.next;
        tail.next = head;
        return;
    }
    do{
        Node n = node.next;
        if (n.val == val){
            node.next = n.next;
            break;
        }
        node = node.next;
    } while (node != head);
}
```



Linear datastructure

* elements arranged in sequential manner. present element is connected to previous & adjacent elements.

eg: Array
Stack
queue
linked list

Static data structure

* has fixed size
* easy to access
* eg: array

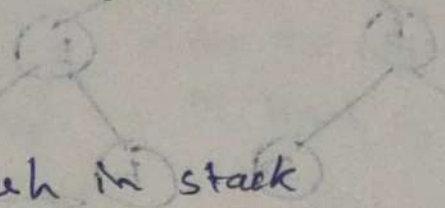
Dynamic data structure

* has no fixed size
* eg: queue, stack

Non-linear data structure

* elements are not placed sequentially.
* we can't traverse all element in one run time.

eg: trees & graphs



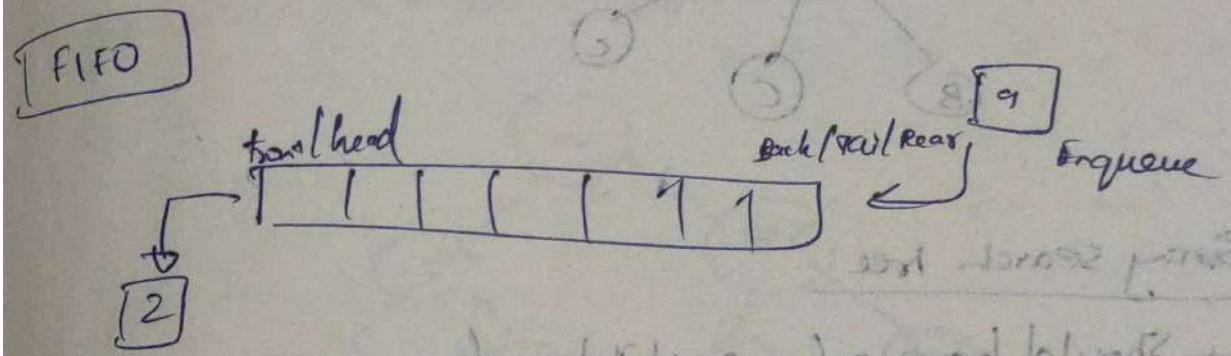
Stack

* Push( ) → push in stack
* peek( ) → retrieve or access top element in stack.
* Pop( ) → delete the top element from stack.

Queue

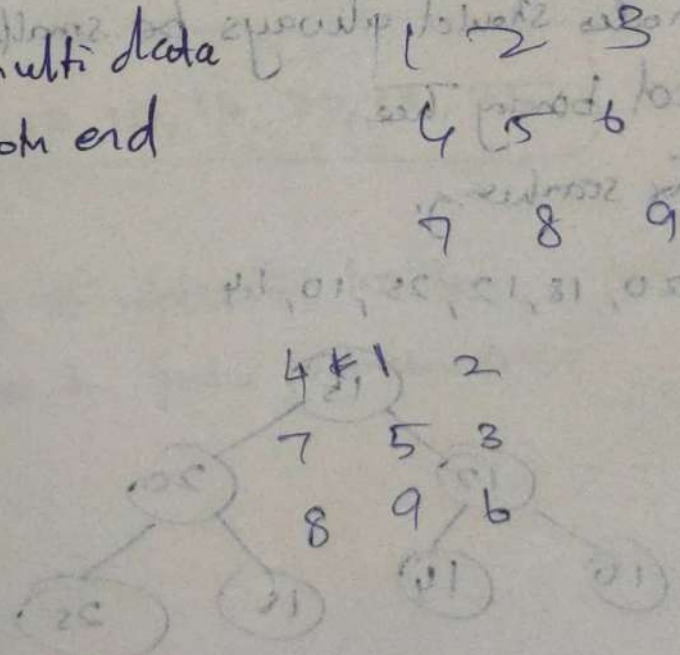
Open at both end

FIFO



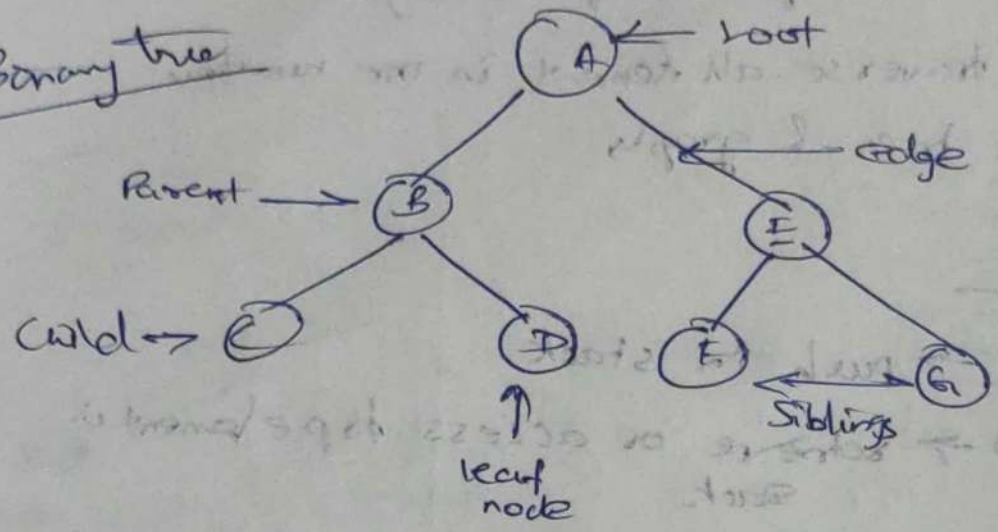
* Can handle multi data
* Can access both end

4 1 2
7 5 3
8 9 6

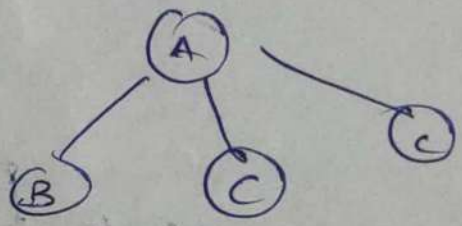


Intro to tree

Binary tree



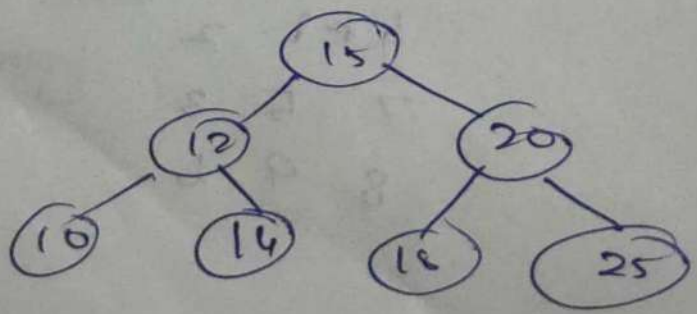
Normal tree there is no restriction that should have only 2 children



Binary search tree

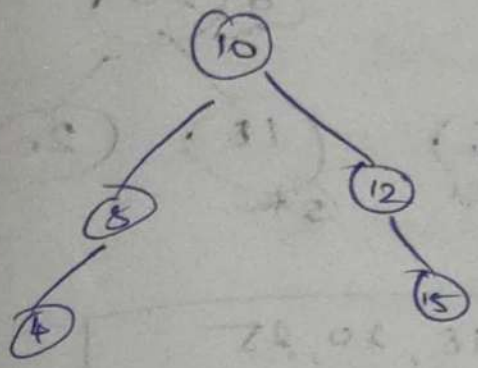
- Should have only 2 child node
- left nodes should always be smaller than right
- ordered binary tree
- used for searching.

eg: 15, 20, 18, 12, 25, 10, 14



Binary search tree

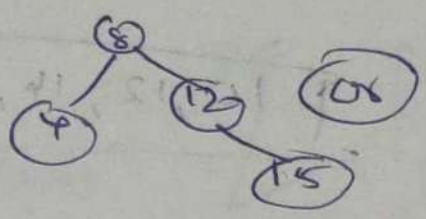
for deleting

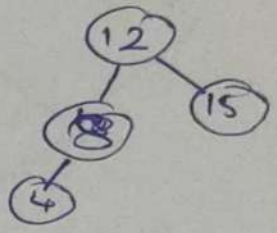


if U want to del (10) then

take the max of left sub tree (8)

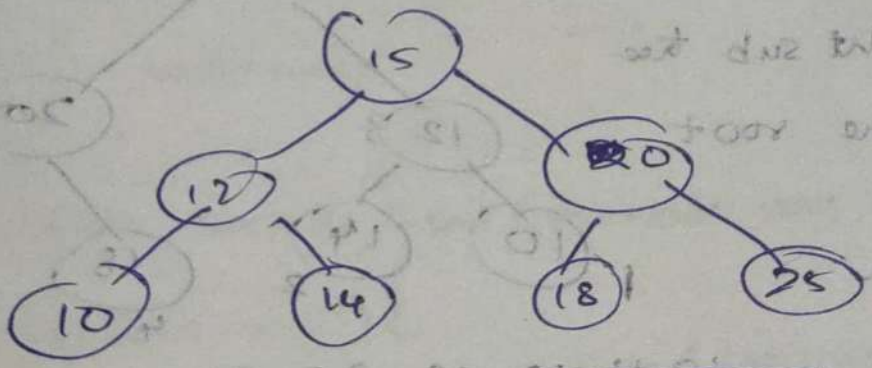
or least of right sub tree (12)





Reorder traversal

1) visit root node 1st
2) visit all nodes on left side
3) visit all nodes on right side



15, 12, 10, 14, 20, 18, 25

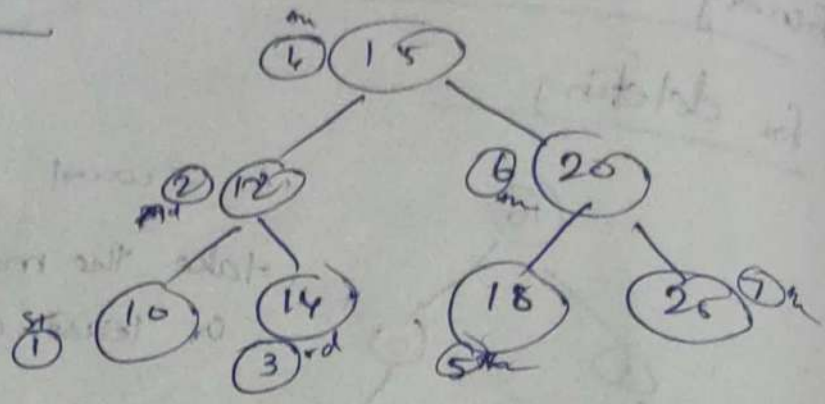
Uses

when U want to replicate the tree

expression tree to postfix understanding

Inorder traversal

- 1st visit left sub
- then root
- then right sub tree



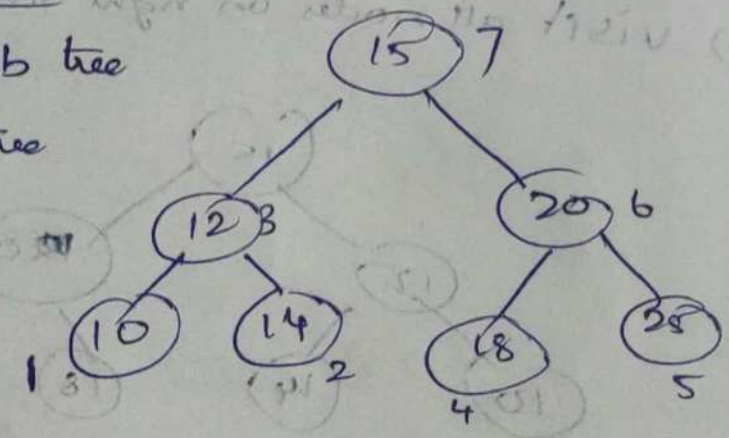
10, 12, 14, 15, 18, 20, 25

ascending order always

- when U person inorder in BST then U get sorted ans

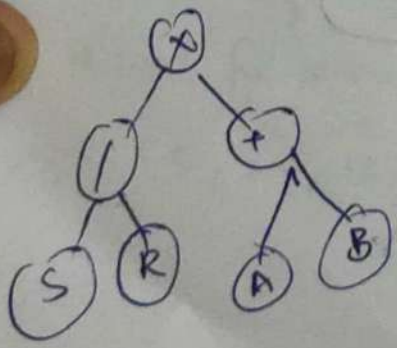
Post order traversal

- 1st visit left sub tree
- then right sub tree
- last the root



10, 14, 12, 18, 25, 20, 15

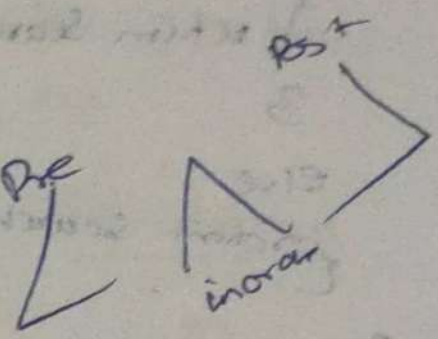
Post order traversal U get postfix notation.



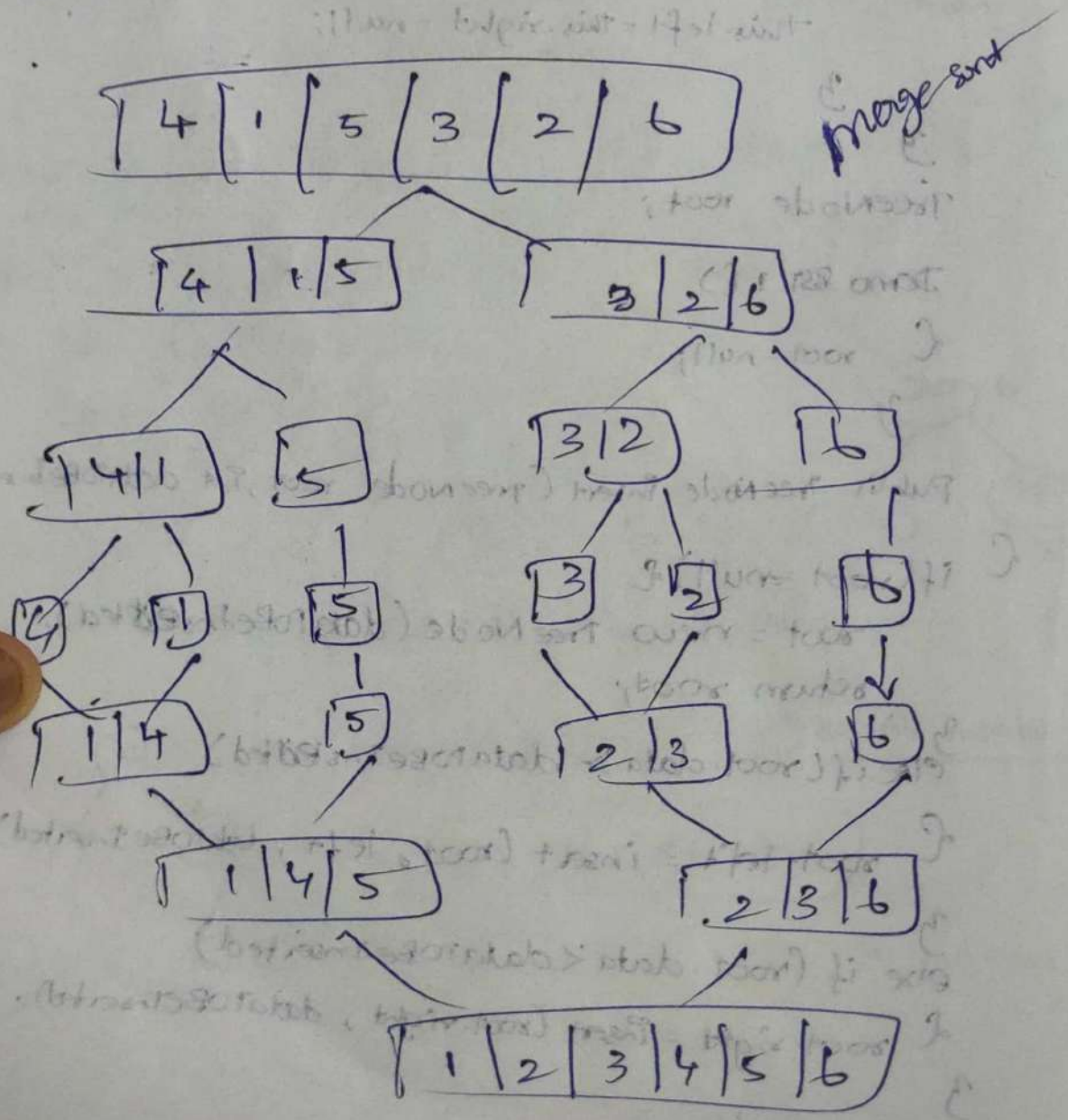
S R / A B + *

BST (Binary search tree)

```
package test1;
import java.util.Scanner;
Public class DemoBST1 {
    Public class TreeNode {
        int data;
        TreeNode left;
        TreeNode right;
        TreeNode(int data)
        { this.data=data;
          this.left=this.right=null;
        }
    }
    TreeNode root;
    DemoBST1(){
    {   root=null;
    }
    Public TreeNode insert(TreeNode root, int dataTOBeInserted)
{   if(root==null){
        root = new TreeNode(dataTOBeInserted);
        return root;
    }else if(root.data > dataTOBeInserted)
    {   root.left = insert(root.left, dataTOBeInserted);
    }
    else if(root.data < dataTOBeInserted)
    {   root.right = Insert(root.right, dataTOBeInserted);
    }
    return root;
    }
```



```
Public TreeNode search(TreeNode root, int dataToBeSearched)
{ if(root==null || root.data == dataToBeSearched)
    {return root};
else if(root.data > dataToBeSearched)
   { return search(root.left, dataToBeSearched);
   }
   else
   {return search(root.right, dataToBeSearched);}
}
```



NPTEL

Week 2

2.1

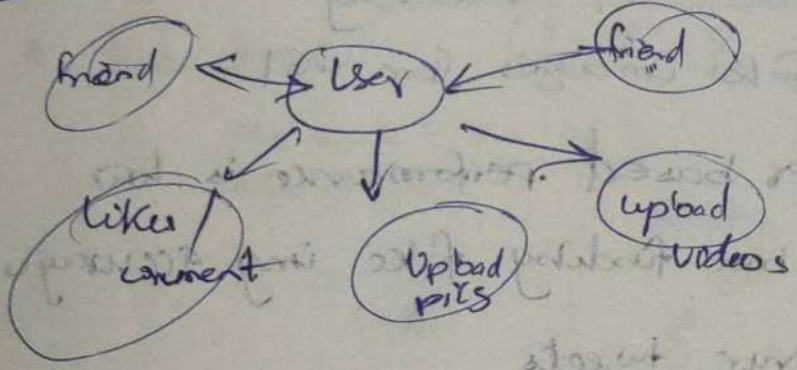
4 / 5 V's of OSM

volume
velocity
veracity
~~validity~~ variety
values

JSON - JavaScript Object Notation

PhpMyAdmin - access MySQL database & query using browser

* In facebook data is stored in graph format (Graph API)



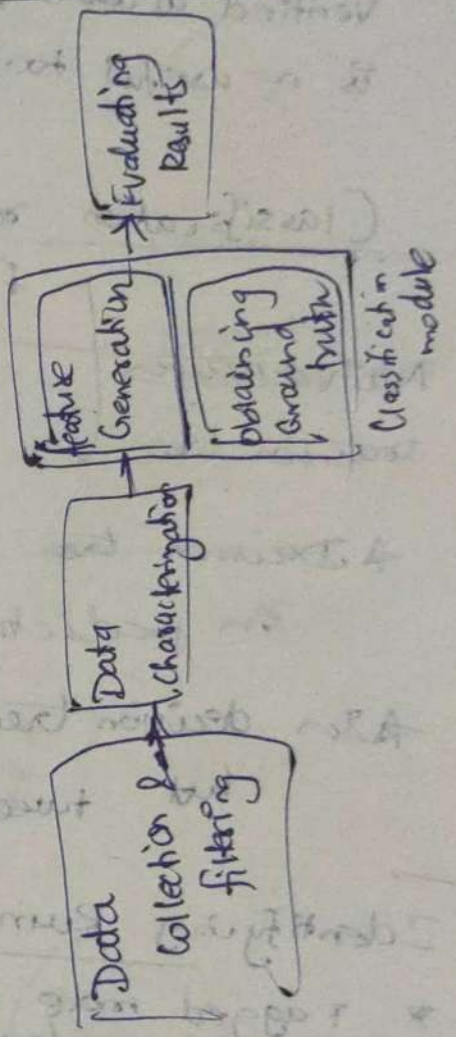
* Called as graph API because all obj are stored as nodes

Week 2.2

Trust & credibility on OSM

* Boston blast
* Hurrican Sandy
  date: Oct 22-31, 2012
  Damage Cost: $75 billion
  Cost of NA

Methodology



During Hurricane Sandy

* Tweets with fake images - 10,350
* Users with fake images - 10215
* tweets with real images - 5767
* Users with real images - 5678

We do analysis like

Who
When
Where
what
why
How

---

* to find user is legitimate verified account from facebook is a useful tool.

Classification results

	F1 (users)	F2 (tweets)	F1 + F2
Naive Bayes	56.32%	91.97%	91.52%
Decision tree	53.24%	97.65%	96.65%

* Decision tree is best 97% accuracy in predicting fake images from real

* In decision tree user based performance is low but tweets base finding fake img accuracy is high

Identifying Rumor/True tweets

* Tagged more than 20 tweet contain
  - Rumor/fake
  - true
  - General (NA)

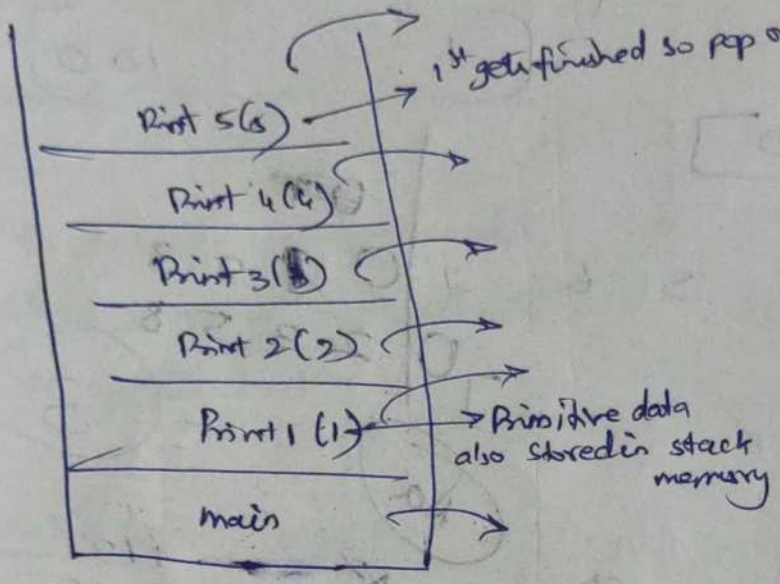
* Six Rumors:
  - 130,690 tweets / Retweets (29%)
    RIP 8 year old boy running for Sandy hook kids

* Seven true news:
  116,54 tweets / retweets (20%)
  - Doctor bombs contained pellets, shrapnel & nail that kid alive

Recursion (fn. that calls itself)

how function call get in stack

the fn reside in stack until it is finished



* when a fn. finishes executing it is removed from stack & the flow of fn is restored to where it is called.

Recursion

if u count to print 1 2 3 4 5

```
Public static void main (String[] args)
{
    Print(1);
}
Public static void print (int n)
{
    System.out.println(n);
    if(n<5)
    {
        print(n+1);
    }
}
```

Output

1
2
3
4
5

* every fn. call even the safe fn. it takes some
* if there is not base condition the recursion fails the stack gets filled. (stack overflow)

need of recursion

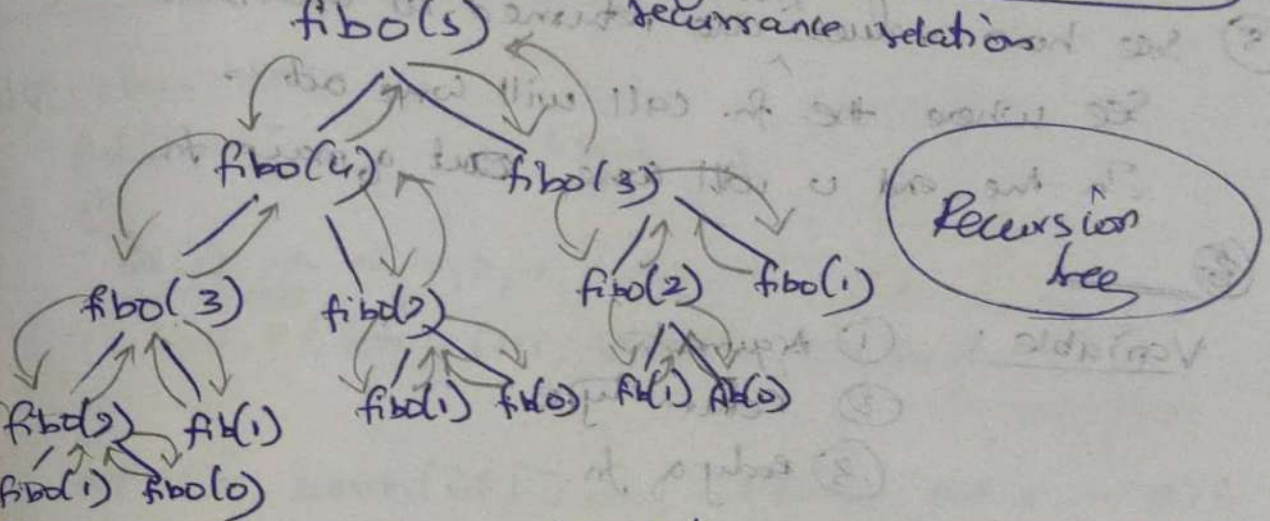
* helps to solve big & complex prob in simple way.
* U can convert recursion sol. into iteration & vice versa.
* space complexity is not constant because of recursive calls.
* It helps breaking big prob to small prob

kudutrano oda muradi no

kudutan no. ku twice muradi na

fibonacci no. $fibo(n) = fib(n-1) + fib(n-2)$

recurrance relation



* break it down to smaller prob-
* bas condition is represented by ans we have already.
* in this case we know $F(0) = 0$, $F(1) = 1$

```
Public class Fibbo{
    Public static void main(String[] args)
    {   s.o.p(fibo(7));
    }
    Public static int fibo(int n){
        if (n<2)
            return n;
    } return fibo(n-1) + fibo(n-2);
}
```

output: 8

Rules to identify & approach a problem

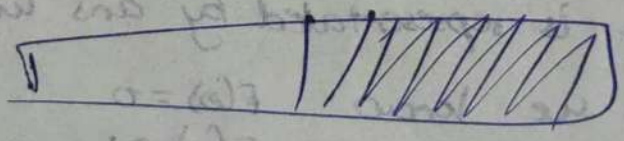
(1) Identify if u can break down prob into smaller prob
(2) write the recurrence relation if needed
(3) Draw recursive tree
(4) about the tree
- ★ see flow & fn, how they are getting in stack.
- ★ Identify the flow of left tree & right tree
- ★ Draw the tree & pointer again & again using pen & paper
- ★ use a debugger to see the result

(5) see how values & what type of value are returned at each step
See where the fn. call will come out.
In the end u will come out of main fn.

(6) ———

Variable: (1) Arguments
(2) Return type
(3) Body of fn.

Q: Binary search with recursion



(1) Comparing → O(1)
(2) Divide into half

recurrance relation

Comparison array search in constant time

Divide ½

$$T(N) = O(1) + T\left(\frac{N}{2}\right)$$

N → no of element in array

Types of recurrance relation

① linear recursion relation → fibo
② Divide & conquer recurrance relation → Binary search
recurrance by factor

Dynamic programming

if two or more fn perform same operation & result use dynamic programming to neglect it.

* future recursion calls la theva padura args' ah fn. call la podu.
* theva elladhadha body of fn. la podu

Binary search using Recursion

```
Public class BS {
    Public st...          (str[]...)
    {
      int[] arr = {1,2,3,4,55,66,78};
      int target = 4;
       S.O.P(search(arr, target, 0, arr.length-1));
    }
    static int search(int[] arr, int target; int s, int e){
      if(s>e){
          return -1;
      }
      int m = s+(e-s)/2; // Illya m=(s+e)/2
      if(arr[m]==target){
          return m;
      }
      if(target<arr[m]){
          return search(arr, target, s, m-1)
      }
      return search(arr, target, m+1, e)
    }
```

→ to avoid overflow errors
to avoid no. to overflow more than given datatype.

Print pattern using recursion

```
* * * *
* * *
* *
*
```

```
Public class main{
  P.S.V.M(String[] args)
  {
    triangle(r:4, c:0);
  }
Static void triangle(int r, int c)
{  if(r==0)
   { return ;}
   if(c<r)
   { s.o.p("*");
     triangle( r, c+1);
   }
   else
   {
     sy.o.println();
     triangle(r+1, 0);
   }
}
```

do print

```
*
* *
* * *
* * * *
```

Print statement before calls.

---

bubble sort fn call bubble(arr, 4, 0)

```
Static void bubble(int[] arr, int r, int c){
  if(r==0){
    return;
  }
  if(c<r){
    if(arr[c] > arr[c+1]){
      int temp = arr[c];
      arr[c] = arr[c+1];
      arr[c+1] = temp;
    }
    bubble(arr, r, c+1);
  }
  else {
    bubble(arr, r-1 ; 0);
  }
}
```

selection sort recursion

```
static void selection(int [] arr, int r, int c, int max){
    if(r==0){
        return;
    }
    if(c<r){
        if(arr[c] >arr[max])
        {
            selection(arr, r, c+1, c);
        }
        else{
            selection(arr, r, c+1, max);
        }
    }
    else{
        int temp = arr[max];
        arr[max] = arr[r-1];
        arr[r-1] = temp;
        selection(arr, r-1, 0, max:0);
    }
}
```