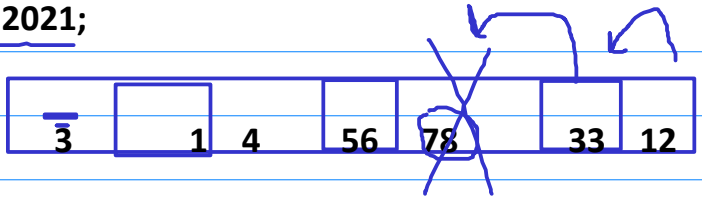
```
dd=dd+1; 32
        if(dd>noOfDays[mm]) {
            dd=1;
            mm++;
            if(mm>12) {
                mm=1;
                yyyy++;
            }
        }
```

31

```
int dd, mm, yyyy;
        dd = 31;
        mm = 12;
        yyyy = 2021;
```

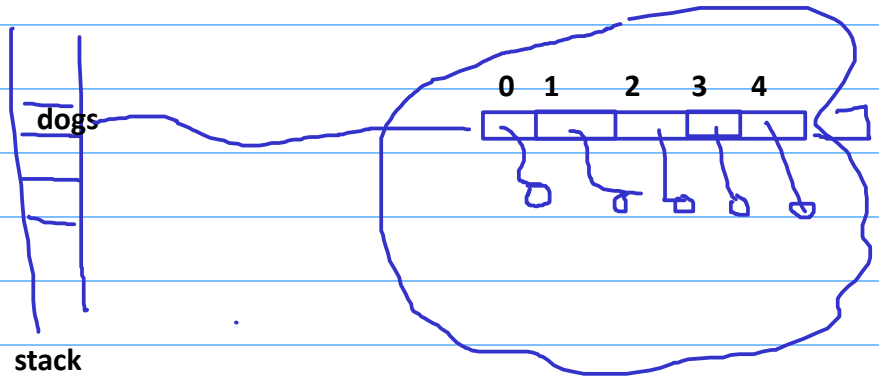


+ : random access is very fast : O(1)

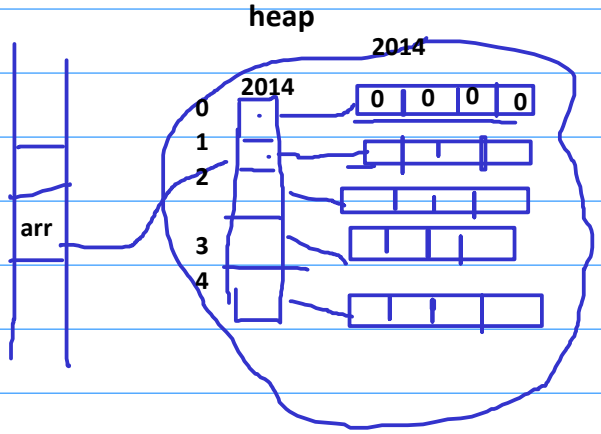
-: add/del element inside array is a slow process

O(N)

```
Dog []dogs=new Dog[5];
//how many dogs are there in this array?
```




```
int arr[][]=new int [5][4];

    //?
    for(int temp[]:arr) {
        for(int tempVal:temp) {
            System.out.print(tempVal+" ");
        }
        System.out.println();
    }
```

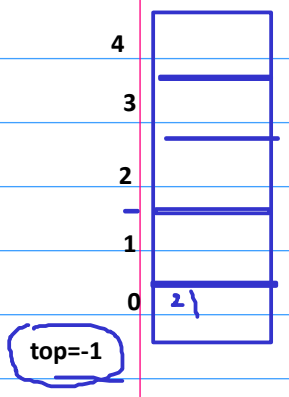


$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a00 & a01 & a02 \\ a10 & a11 & a12 \\ a20 & a21 & a22 \end{bmatrix}$$

$AI = A$

ADT? "Abstract data type"



Array
top=-1

```
int i;
float j;
String s
```

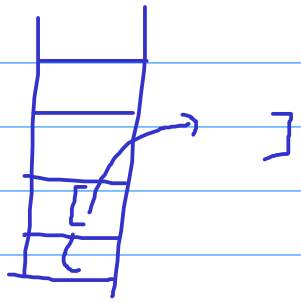
```
class Stack{
}
```

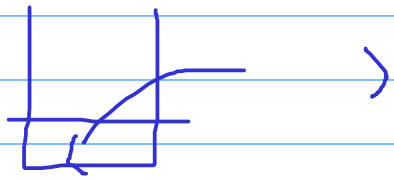
```
return top==-1;
```

stack applications:

validity of an expression

(2X3 + [5+2/7])

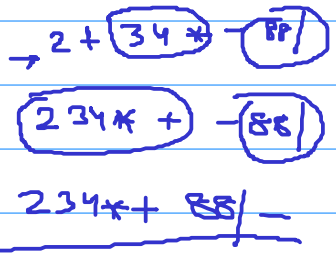




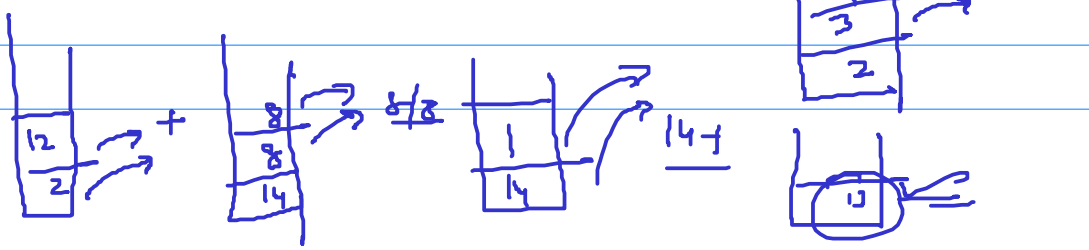
evaluation of an expression

infix ---> postfix

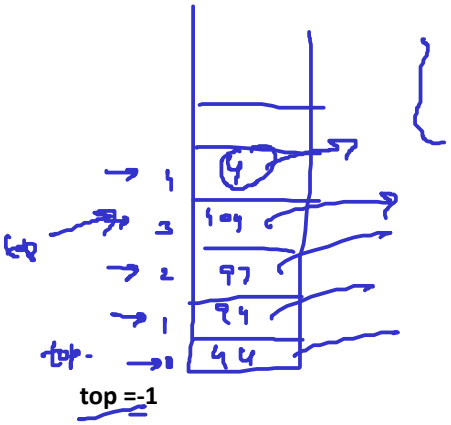
2+3*4-8/8



postfix, prefix, infix

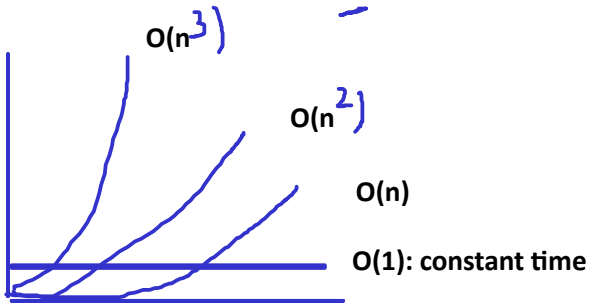


```
Stack stack=new Stack(5);
        stack.push(44);
        stack.push(94);
        stack.push(97);
        stack.push(404);
        stack.push(4);
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
        System.out.println(stack.pop());
```




```
public void push(int data) {
        if (!isFull()) {
            arr[++top] = data;
        } else
            System.out.println("stack is full");
    }

    public int pop() {
        if (!isEmpty()) {
            return arr[top--];
        } else {
            System.out.println("stack is empty");
            return -999;
        }
    }
```

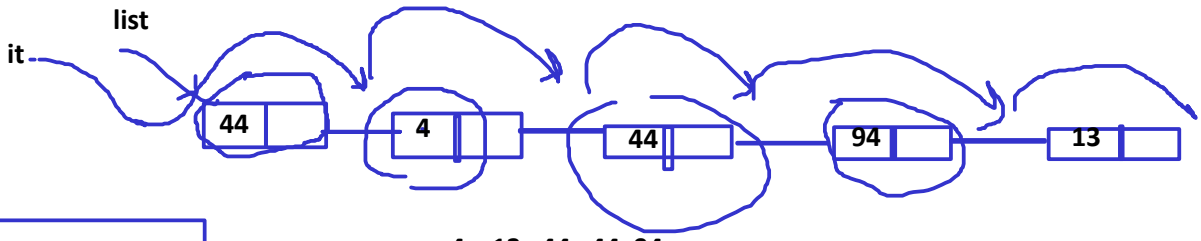
"it depends"



Arrays	LinkedList
radom access is very fast O(1)	only option of linear seach O(n)
can not grow at run time	possbile
not flexible	more flexible
add/del an element inbetween is slow	it is just pointer manipulation => fast

2 option : readymade / create ur own

```
LinkedList<Integer> list=new LinkedList<Integer>();

        list.add(44);
        list.add(4);
        list.add(44);
        list.add(94);
        list.add(13);
```



4 13 44 44 94
0 1 2 3 4

```
Iterator<Integer> it=list.iterator();
    while(it.hasNext()) {
        System.out.println(it.next());
    }
```

```
int index=Collections.binarySearch(list, 3);
```

-(-2+1)
= 0

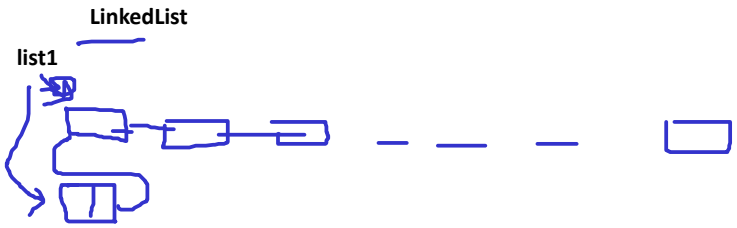
11

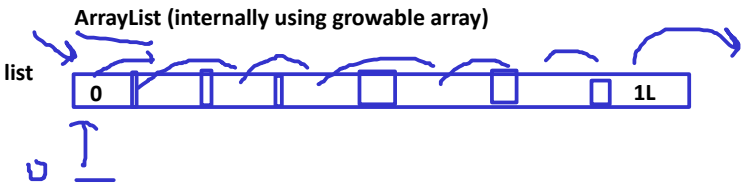
-2+1
=-1

prove the performace of LL is better then AL

DLL (Growable array)

```
private static void doBenchmark(List<Long> list1) {
        //first i will add 1L element into list
        for(long i=0;i<1E5; i++) {
            list1.add(i);
        }

        long start=System.currentTimeMillis();
        for(long i=0;i<1E5; i++) {
            list1.add(0, i);
        }

        long end=System.currentTimeMillis();

        System.out.println("time taken: "+ (end-start)+" ms");
    }
```





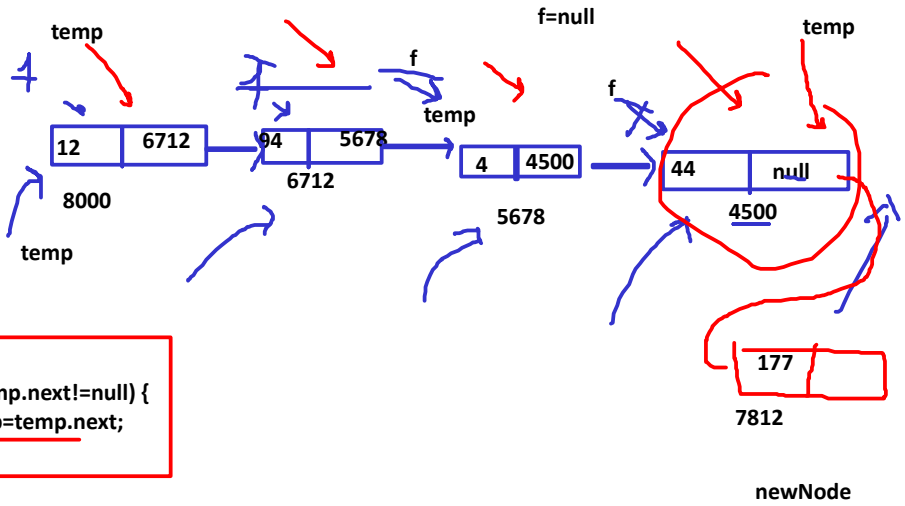
how to create my own link list?

LL is self ref structure?

```
class Node{
        int data;
        Node next;

}
```

```
class LL{
        Node f, l;

        addFirst
        addLast

        delFirst
        delLast
        print
        ...
}
```

```
void insertFirst(int data) {
        Node temp=new Node(data);
        temp.next=f;
        f=temp;
    }
    void displayList() {
        Node temp=f;
        while(temp!=null) {
            System.out.print(temp.data +"->");
            temp=temp.next;
        }
    }
```

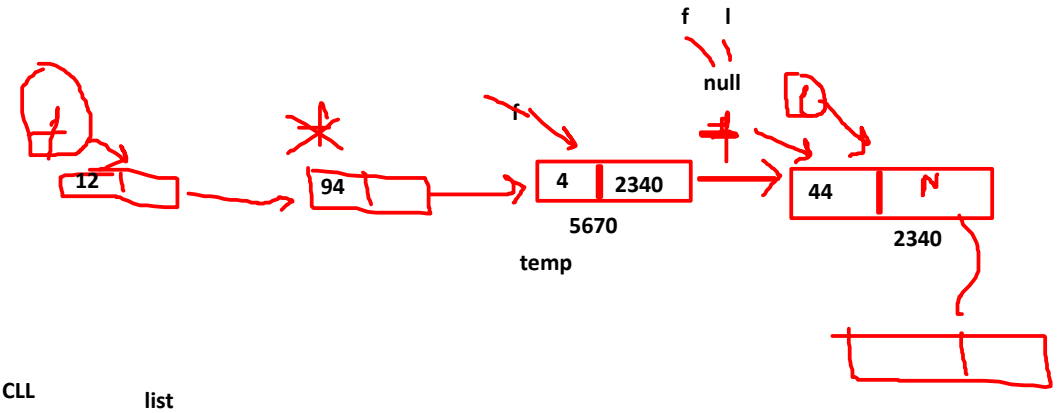



```
list.insertFirst(44);
        list.insertFirst(4);
        list.insertFirst(94);
        list.insertFirst(12);

        list.displayList();
```

```
Node temp=f;
    while(temp.next!=null) {
        temp=temp.next;
    }
```

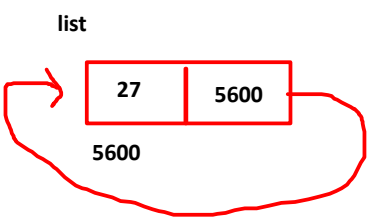
Initially f=l=null;

```
void insertFirst(int data) {
    Node temp=new Node(data);
    if(isEmpty()) {
        l=temp;
    }
    temp.next=f;
    f=temp;
}
```



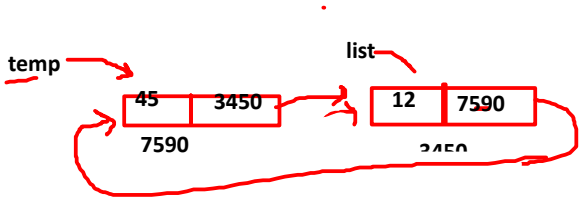

```
list.insertFirst(44);
list.insertFirst(4);
list.insertFirst(94);
list.insertFirst(12);
```

CLL




```
class CLL{
    Node list;
    public insertFront(int data){
    Node tempNode=new Node(data);

    if(list==null){
        list=tempNode;
    }else
        tempNode.next=list.next;

    list.next=tempNode;
}
```

list=null



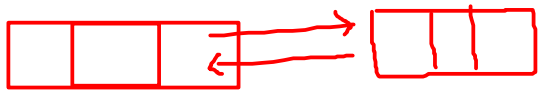
Print method

```
Node temp=list.next;
while(temp!=list){
    sysout(temp.data);
    temp=temp.next;
}
```

3 how to impl stack and queues using LL?

LIFO

DLL

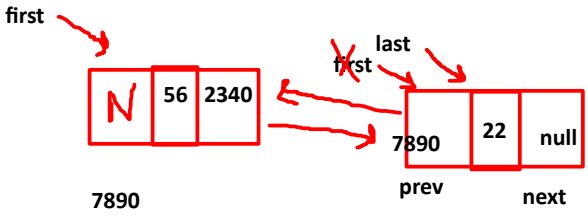


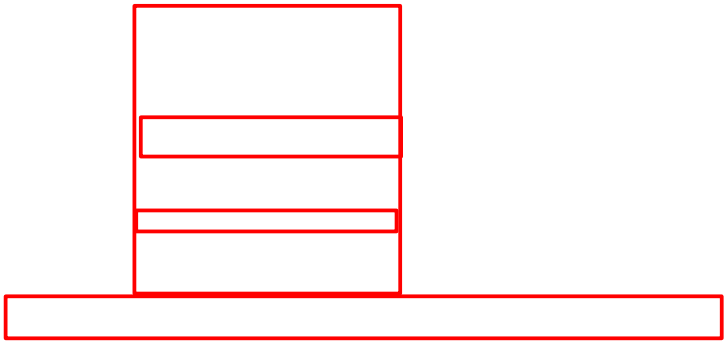

```
class Node{

    int data;
    Node next, prev;
}
```

```
class DLL{

    Node first, last;

    ....
    public boolean isEmpty(){
        return first==null;
    }

    ///....

}
```

```
public void insertFirst(int data) {
    Node tempNode=new Node(data);     //333
    if(isEmpty()) {
        last=tempNode;
    }else {
        first.prev=tempNode;
    }
    tempNode.next=first;
    first=tempNode;
}
```

first=last=null;





DS OOPs DBMS OS

Good morning all, we start in 2min

Stack?
LIFO: applicatations:rec, undo redo, expression evaluation

add/ delete:

push

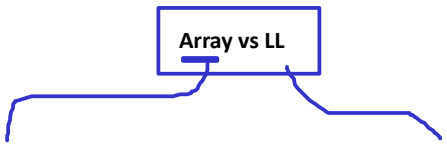
pop

isEmpty

peek

Colllections?
readymade ds

ds vs algo



it can not grow at run time
random access is very fast O(1)

add/del is slow process
as shifting of element is required

dynamic
add/del of node
is just a pointer
maniputaion

is fast

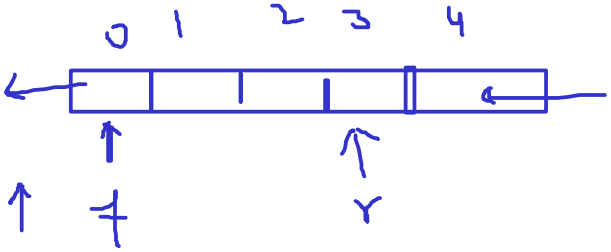
random access is not allowed

LL: singleLL
CircularLL
DLL

Agenda for day:

1. Queue : own queue, collection api
2. recursion and applications
3. searching
4. sorting
5. hashing
6. tree

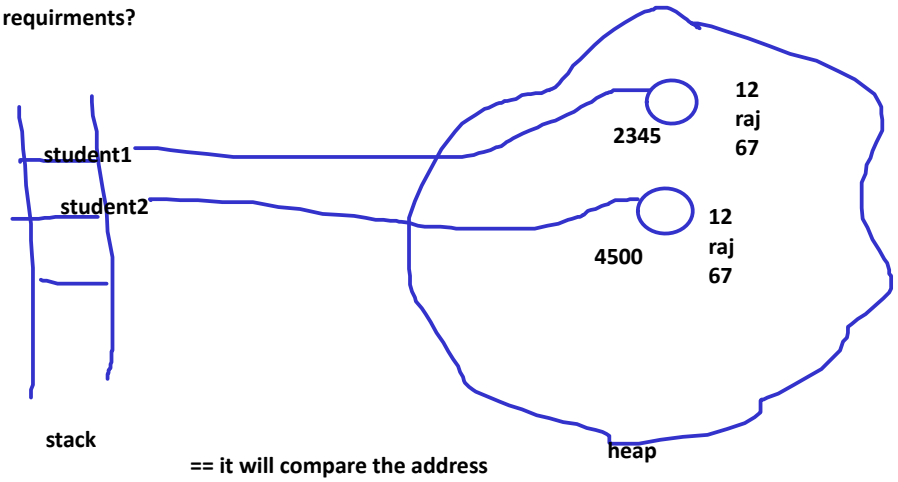
FIFO



fundamentals of java oops

```
Student student1=new Student(12, "raj", 67);
Student student2=new Student(12, "raj", 67);
```

equals method requirments?



== it will compare the address

equals method if u want to compare the contents!

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

u need to override equals method for custom object (user define objects)

Student, Account, etc

for them i need to override eqauls method

hashcode() : it is a good programing practice to override this method with equals () method

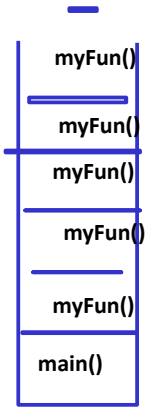
overriding ?

Assignment:

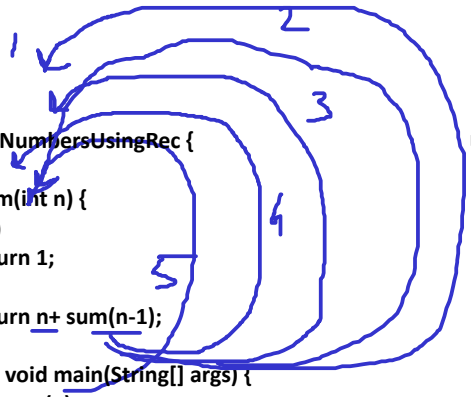
Product
---
id
name
price

We need to create 5 products and add the the priority queue and print them as per there price

```
class Foo{
    void myFun() {
        System.out.println("it is myfun");
        myFun();
    }
}
public class BadRecursion {

    public static void main(String[] args) {

        Foo f=new Foo();
        f.myFun();
    }
}
```



base condition :
i want to sum from 1 to N=10




```
//how it works?
public class SumNumbersUsingRec {

    static int sum(int n) {
        if(n==1)
            return 1;
        else
            return n+ sum(n-1);
    }
    public static void main(String[] args) {
        int val= sum(5);
        System.out.println(val);
    }
}
```

n=5

return 5 + sum(4)

return 4+sum(3)

return 3+ sum(2)

return 2 + sum(1)

return 1

return 5 + 4 + 3 +2 +1

Fibonacci numbers

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

a b

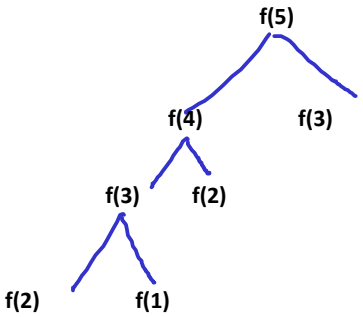
c=a+b

lopping logic

```
a=0;
b=1;

while(.............){
  c=a+b;
  print c

  then say

  a=b;
  b=c;

}
```

Recursion?

```
public class DemoFav {

    //nth term : 5
    static int fab(int n) {
        if(n==0 || n==1) {
            return n;
        }else {
            return fab(n-1)+ fab(n-2);
        }
    }
    // 1 1 2 3 5.....fibonacci series? using recursion?
    public static void main(String[] args) {
        // 1, 1, 2, 3, 5, 8, 13, 21....
        int sum= fab(5);
        System.out.println(sum);
    }
}
```



Binary search:

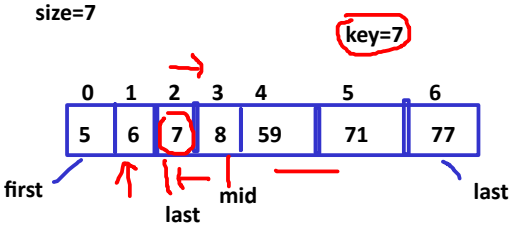
O(log n)

64

log 64 = log

6

size=7



mid=(fist+last)/2 =(0+6)/2=3

3-5PM

5:15PM
5PM

Session break 3-5PM

I have informed about your meeting extension to edureka

Sorting technique:

bubble sort:

day 3:
searching: linear, binary
sorting: bubble, selection, insertion, merge sort, quick sort

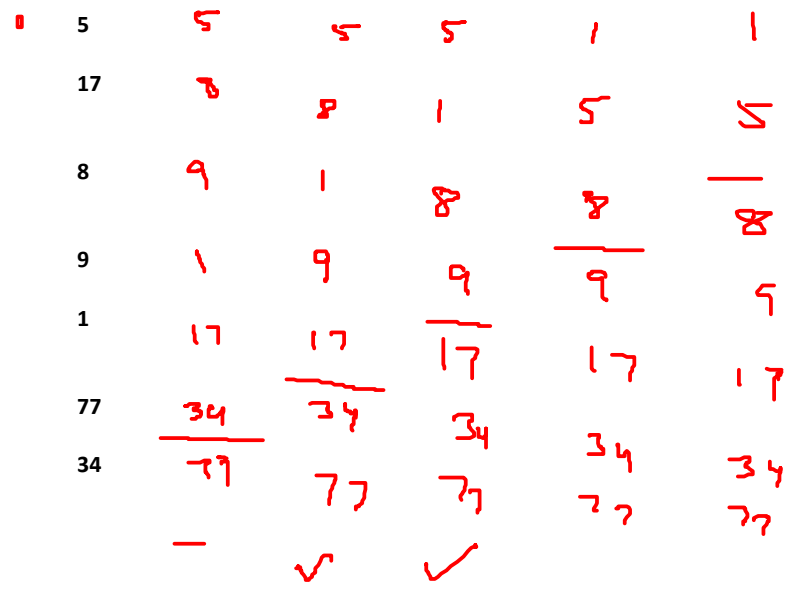
hashing:
?hashcode, ?hashSet, Hashmap, hashtable, ex application
how hashing works? collision

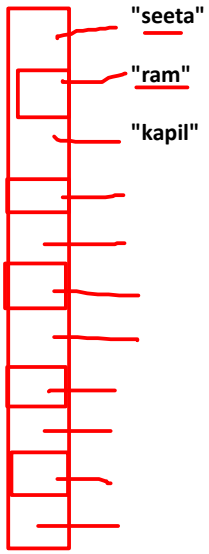
tree, BBT, traversal
TreeSet? example?
creating tree, traversal, delettion etc

graph: adjancy matrix/list
dijkstra

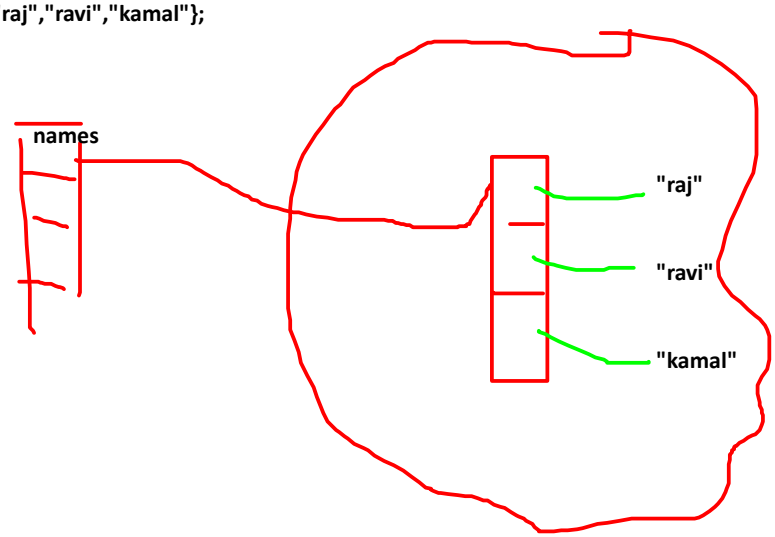
heap max,min heap example applications

Bubble sort:





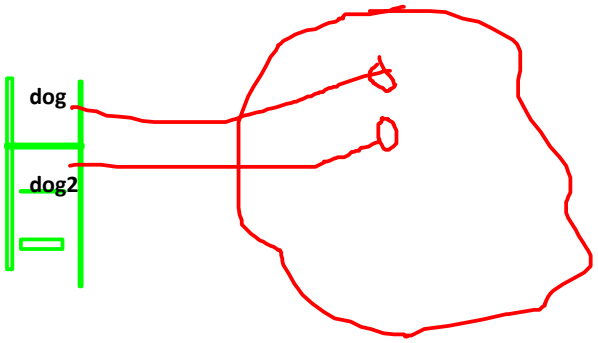
String []names= {"raj","ravi","kamal"};



```
//String vs StringBuffer vs StringBuilder
//String are immutable and thread safe=> String pool*
Dog d=new Dog();
String name="raja";
String name2=new String("raja");
String name3="raja"
```


```
class Dog{
 static int counter;
 public void bark(){

}
}
```

```
Dog dog=new Dog();
Dog dog2=new Dog();
```



Selection sort

25	47	3	19	8	18
3 |	47	25	19	8	18
3	8 |	25	19	47	18
3	8	18 |	19	47	25
3	8	18	19 |	47	25

25 | 47 3 19 8 18

25 47 | 3 19 8 18

3 25 47 | 19 8 18

3 19 25 47 | 8 18

3 8 19 25 47 | 18

i = 1 el = 3

j = 0

```
int arr[]= {25,47,3,19,8,18};

    for(int i=1;i<arr.length; i++) {
        int ele=arr[i];

        int j=i-1;
        while(j>=0 && arr[j]> ele) {
            arr[j+1]=arr[j];
            j--;
        }
        arr[j+1]=ele;
    }

    for(int temp: arr) {
        System.out.print(temp+" ");
    }
```

el = 47 j = 0

0	1	2	3	4	5
25	47	3	19	8	18

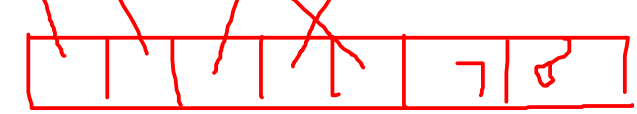
25 47 | 3 19 8 18

3 25 47 | 19 8 18

2 4 1 6 8 5 3 7

i → → j

1 2 4 6 3 5 7 8



k

Lunch break 1-2PM

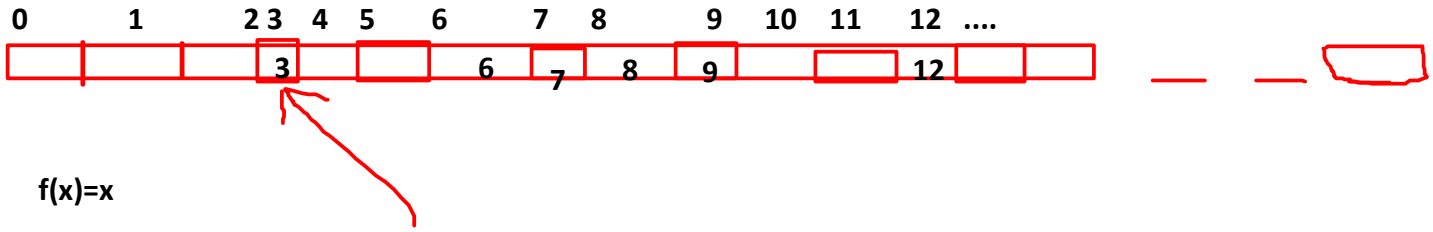
Hashing ? hashcode.
HashSet
Set
Map examples applications
how it works?

O(1)

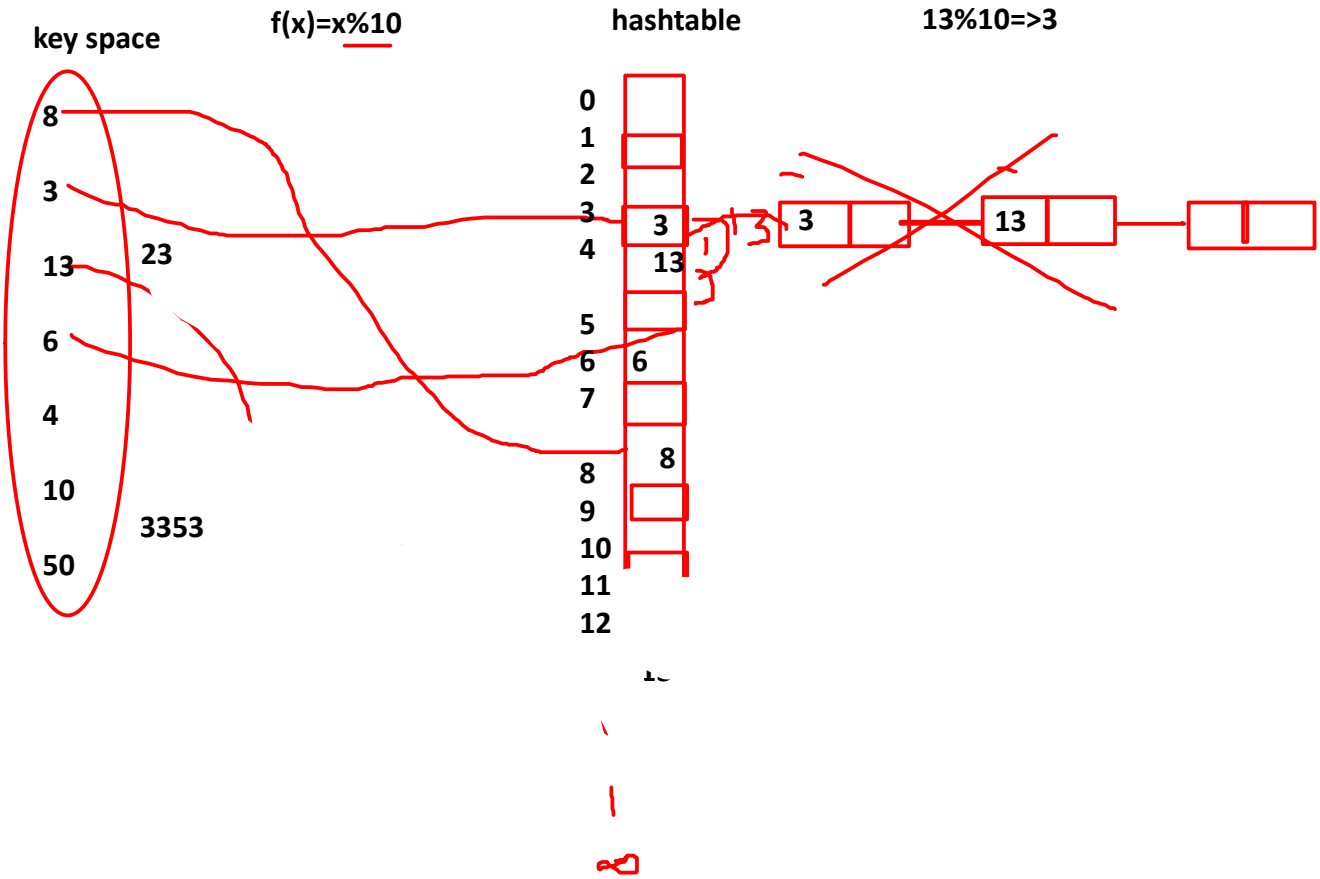
linear : O(n)
binary : O(log n)
hashing O(1)

What is hasing :

8 3 6 7 12 9 6 599999999



f(x)=x



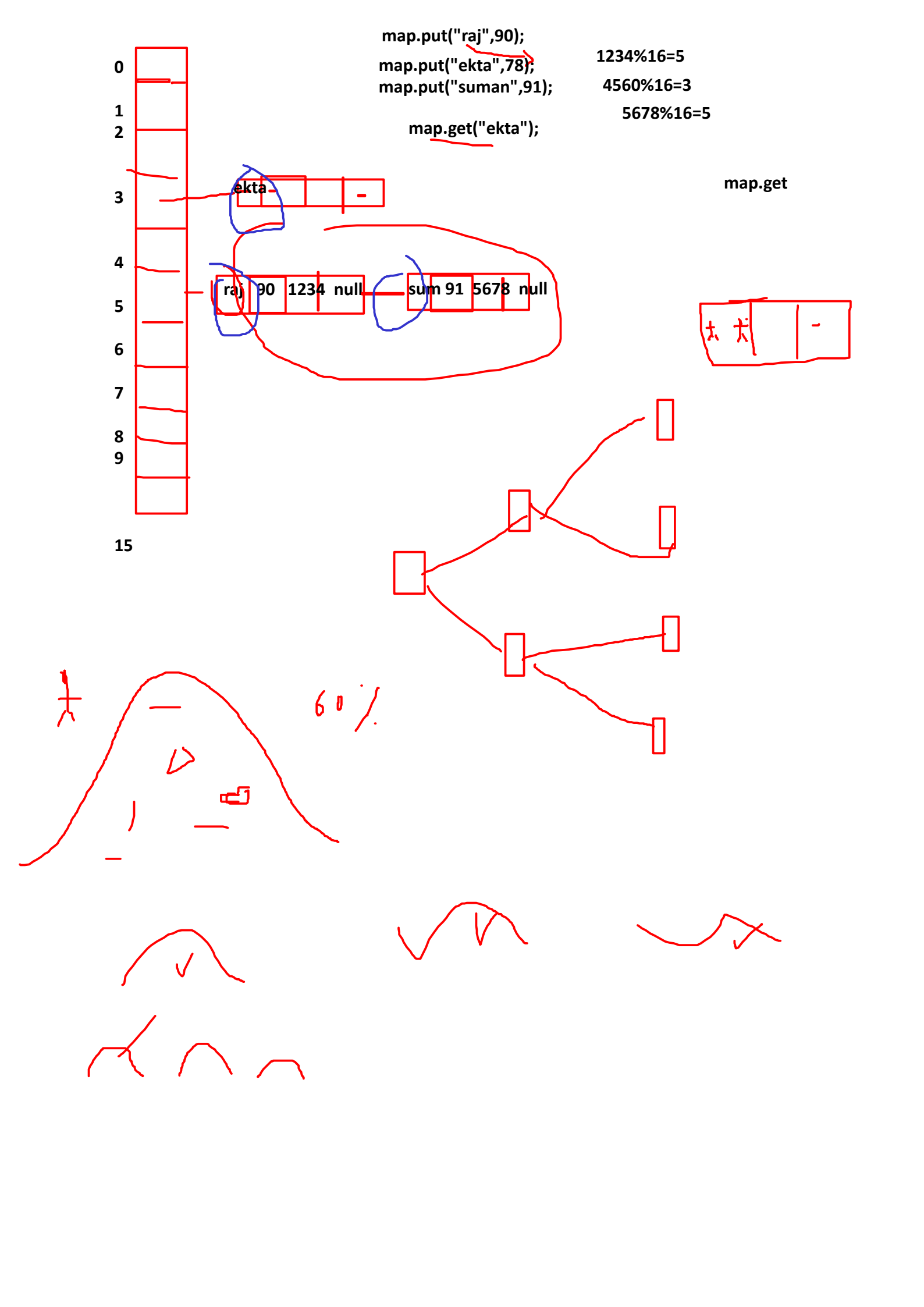
what to do:
i have run some code ..
putting in chat run it...

Q. u have a file

20min

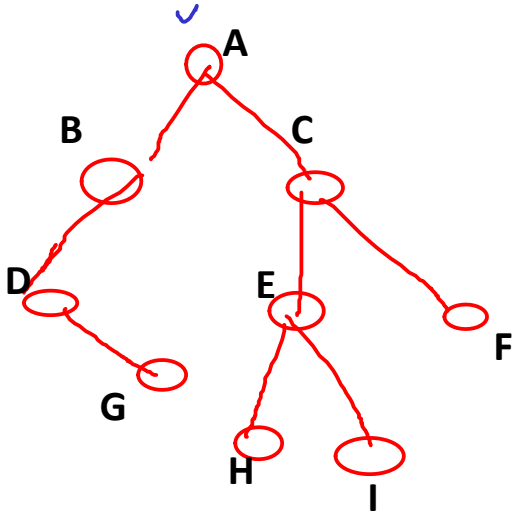
apple 4
apple 5
bannana 7
bannana 10

apple 9
bannana 17


map.put("raj",90);
map.put("ekta",78);
map.put("suman",91);
map.get("ekta");
1234%16=5
4560%16=3
5678%16=5
map.get
0
1
2
3
4
5
6
7
8
9
15
ekta
raj 90 1234 null
sum 91 5678 null

Tree ds:

root



traversal: breath first (level order)

depth first(height wise)

preorder NLR

inorder LNR

postorder LRN

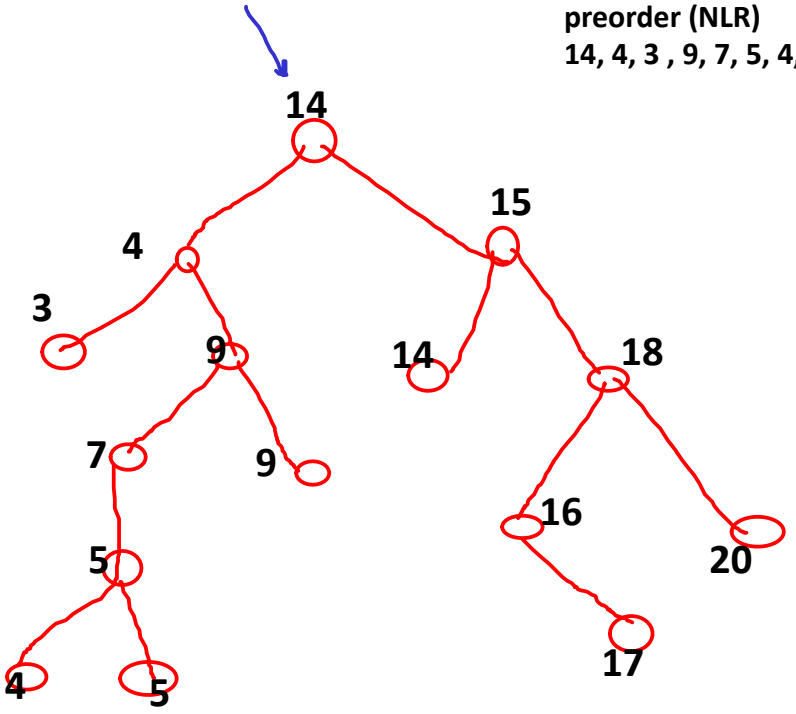
pre order: ABDGCEHIF

inorder : DGBAHEICF

postorder: GDBHIEFCA

preorder (NLR)

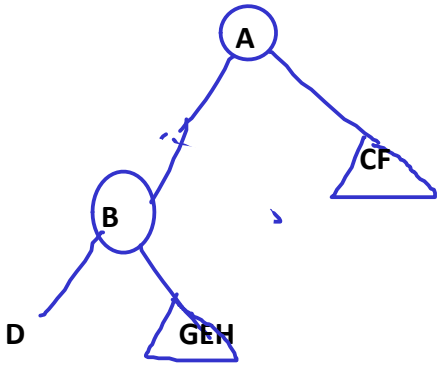
14, 4, 3 , 9, 7, 5, 4, 5, 9, 15, 14....



Construct BT

preorder : A B D E G H C F

INORDER :D B G E H A C F



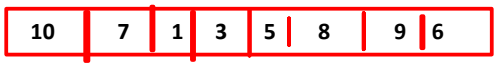
pre BDEGH

inorder DBGEH

TreeSet

```
class StudentNameSorter implements Comparator<Student> {
    @Override
    public int compare(Student o1, Student o2) {
        int val = o1.name.compareTo(o2.name);
        if (val == 0) {
            return Integer.compare(o2.marks, o2.marks);
        }
        return val;
    }

}
```

compareTo
0: both are same
+
-


10
7
1
3
5
8
9
6