DSA IN JAVA - DAY 7

methods in java

methods are block of code which are used repetively to use the DRY principle ie d do not repeat yourself

why to use methods and how to use methods in java ?

methods are used to replicate a certain body of code
for eg question write a program to takr 2 numbers as input and return the sum

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter the number 1 here:");
int num1 = sc.nextInt();
System.out.println("Enter the number 2 here:");
int num2 = sc.nextInt();
int sum = num1 + num2;
System.out.println("the sum is "+ sum);
sc.close();
```

```
Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number 1 here:");
    int num1 = sc.nextInt();
    System.out.println("Enter the number 2 here:");
    int num2 = sc.nextInt();
    int sum = num1 + num2;
    System.out.println("the sum is "+ sum);
    sc.close();
```

this code above take two numbers as input and returns the sum

if we want to execute the same code number of times it will be not good to erite it each time so we use methods to make it convineint

syntax of methods

```
acces modifier  return_type   name() {

        // body of the code

          retrun statement;
                                }
```

syntax of methods

example :-

```
static void sum() {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the number 1 here:");
      int num1 = sc.nextInt();
      System.out.println("Enter the number 2 here:");
      int num2 = sc.nextInt();
      int sum = num1 + num2;
      System.out.println("the sum is "+ sum);
      sc.close();
  }
```

the complete code :-

control

writing the function name

calling the function

```
import java.util.Scanner;

public class feb_2_methods {

    public static void main(String[] args) {

        sum();

    }
    static void sum() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number 1 here:");
        int num1 = sc.nextInt();
        System.out.println("Enter the number 2 here:");
        int num2 = sc.nextInt();
        int sum = num1 + num2;
        System.out.println("the sum is "+ sum);
        sc.close();
    }
}
```

q. write a program using a method to print hello world ans name that function display

```
import java.util.Scanner;

public class feb_2_methods {

    public static void main(String[] args) {
  display();
        }
    static void sum() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number 1 here:");
        int num1 = sc.nextInt();
        System.out.println("Enter the number 2 here:");
        int num2 = sc.nextInt();
        int sum = num1 + num2;
        System.out.println("the sum is "+ sum);
        sc.close();
    }

    static void display(){
        System.out.println("Hello , World!");
    }
}
```

the idea of return type

what happens is after the execution of function call that function call here sum() will contain a value that value is called return type value

and that value will be whatever the return statements declares

examples of return functions

example of int return type t
-- this will return a integer
the code below shows a method with return type int

```
import java.util.Scanner;

public class feb_2_methods {

    public static void main(String[] args) {
        int answer = sum2();
        System.out.println("the answer is "+answer);
        }
    static int sum2(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number 1 here:");
        int num1 = sc.nextInt();
        System.out.println("Enter the number 2 here:");
        int num2 = sc.nextInt();
        int sum = num1 + num2;
        return sum;
    }
}
}
```

since the return type is int we can assign it into a int variable

ie if we write a block of code after the return statement they will never execute and gives error named unreachable statement

example of string example

```
import java.util.Scanner;

public class feb_2_methods {

    public static void main(String[] args) {
        String print = greet();
        System.out.println(print)

        }
static String greet () {
        String greeting = "Konnichiwa";
        return greeting;
    }
}
```

parameters

parameters :- these are used to pass the values from main function to the below methods

example in code is below
here the numbers a and b are passed in the main method but calculated in the sum method

```
public static void main(String[] args) {
    int ans = sum(19, 11);
    System.out.println(ans);
    }
static int sum(int a ,int b){
    int sum = a+b;
    return sum;
}
```

example for a string based one

```
public static void main(String[] args) {
    String personalized_string = mygreet("yo");
    System.out.println(personalized_string);
    }
static String mygreet(String a){
    return "hello" + a;
```

a program to create a method to swap two numbers
note that since here we are dealing with multiple numbers you cant retrun multiple values

```
public static void main(String[] args) {
    int a = 10;
    int b = 20;
    System.out.println( a + " " + b );
    swap(a,b);
    System.out.println( a + " " + b );
    }
static void swap(int a,int b){ // no need to retrun na baka
    int temp = a;
    b= a;
    b = temp;

}
```

shit !!!!!
it did not swap it becuase we are just passing just a copy of the arguments as parameters

one more example to go

```
public static void main(String[] args) {
String name = "Himeshkar";
    ChangeName(name);
    System.out.println(name);
    }
static void ChangeName(String name){
    name = "sonu santosh";
}
```

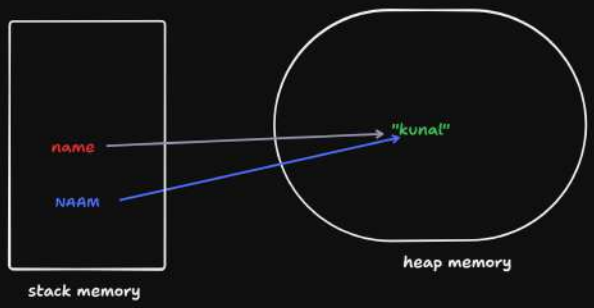
here the name will continue to be himeshkar becuase we are just passing the copy of the above value

internal working of methods

```
main () {
name = "kunal"
greet(name)
}
 void greet(naam){

  print(naam)
   }
```

here the name ie the reference variable 's copy is passed
to the function : ONLY PASS BY VALUE EXISTS
NO PASS BY REFERENCE WHICH MEANS NO POINTERS



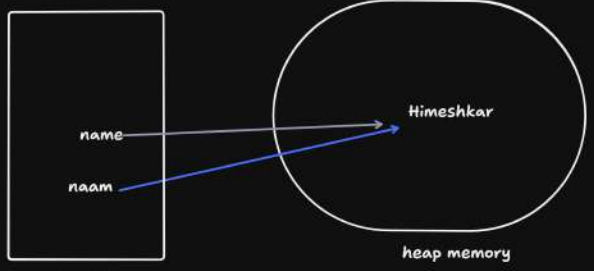
this is how we are accesing the same object even with different refernce names

but why is it not changing the values then ?????

program 1 : change name

```
psvm(){
name = "himesh"
changeName(name)
print // himesh
}
changeName(naam){
naam = "kyoronin"
}
```

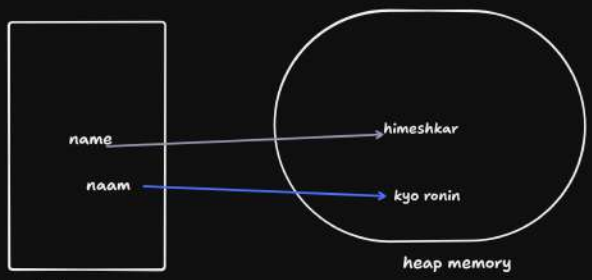
earlier:--



in this step we are not chaging the object but actually
creating a new object

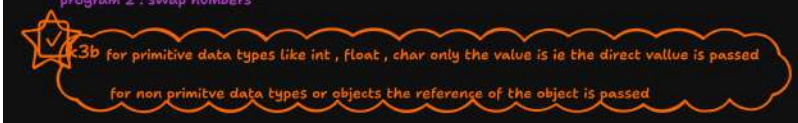
that is why the original object is not changing

now:--



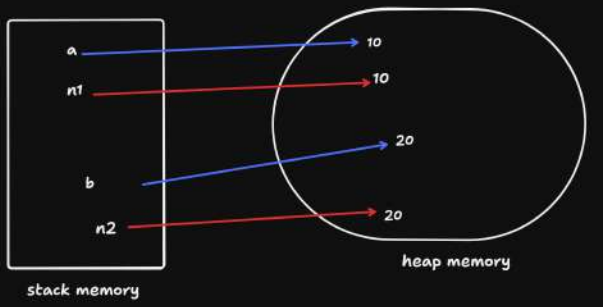
and there is no way to access the arguments of the methods outside methods
ie cannt acces naam in main class

program 2 : swap numbers




```
public static void main(String[] args) {
     int a = 10;
     int b = 20;
     System.out.println( a + " " + b );
     swap(a,b);
     System.out.println( a + " " + b );
     }
 static void swap(int n1,int n2){ // no need to retrun na baka
     int temp = n1;
     n1=n2;
     n2 = temp;

 }
```



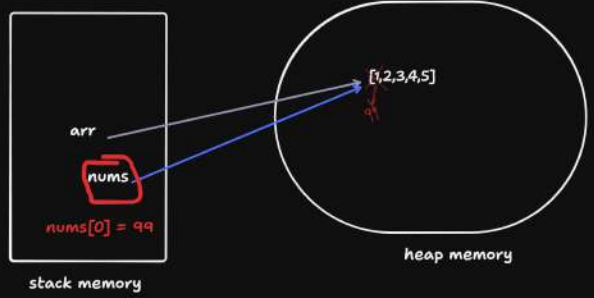
change can be mades

example program

```
public static void main(String[] args) {
  // create a array
     int arr[] = {1,2,3,4,5};
     change(arr);
     System.out.println(Arrays.toString(arr));

     }
 static void change(int nums[]){
     nums[0] = 99; // here we are modifing the string
 }
```

here since by the change function we are not creating a new object we are modifing the existing obj



DSA inJAVA day 8

scope and method scope

scope means the range where we can sees our variables

```
examples-
public class Ish_1_scope {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
    }
    static int random(){
        System.out.println(a);
    }
}
```

here the variable a cannot be accessed by random function

similarly

```
        public static void main(String[] args) {
            System.out.println(nums);
        }
        static int numm(){
            int nums = 10;
            return nums;
        }
    }
```

the variables nums which is created in the numm function cannot be accessed in main function

this type of scope is called function scope or method scope :- where variables are stored and has a scope of only their functions

example

```
    public static void main(String[] args) {
        System.out.println(marks);
    }
    static void random(int marks){
        marks +=45;
        System.out.println(marks);
    }
```

the same thing can also be said for arguments

the varibale which you declare in one function can only be used in that function this is called function or method scope

in previous swap function example

```
static void swap(int a,int b){ // no need to return na baka
    int temp = a;
    b= a;
    b = temp;
```

the changes are only limimted to the scope of the method
so only the varibales of the method have swapped no the original one

block scope

a block of code in java is created as shown below

values initialised in the block remains in the block

```
{

// block of code

}
```

this block can be created inside a function but it's scope is kind of complicated

eg for the code given below

```
    public static void main(String[] args) {
        int a = 10;
        int b = 30;

int a = 107; // this will give an error
        int c = 15;
    }
    System.out.println(c);
    }
}
```

a = 57;

here the thing is the variable a is already initialised in the function so yu cant initialise that in the block also
we can update the value if we want to

but since the variable c is declared inside the block it
also must be used inside the block only and
when tried to use outside of the block it gives us error

example :-

```
    public static void main(String[] args) {
        int a = 18;
        String name = "Himesh";

        {
            a=100;
            name = "Santhosh";
            System.out.println(a);
            System.out.println(name);
        }
        System.out.println(a);
        System.out.println(name);

        here in the block we are not creating anew reference variable at aall we are just modifiying the valluse avialbale here thats'it
    }
```

initialising

here in the block we are not creating anew reference variable at aall
we are just modifiying the valluse avialbale here thats'it

updating

loop scope

same as block scope exmaple the iterators etc

shadowing

here the variable x is intialzied bfeore psvm
so it will be same like block scope

example of code :-

```
static int x = 20; // what if i create a object here
public static void main(String[] args) {
    System.out.println(x);
    fn();
}
static void fn(){
    x = x+8;
    System.out.println(x);
}
```

one more example:-

```
static int x = 20; this is called a global variable scope
 public static void main(String[] args) {
     System.out.println(x); will print global
     int x = 40; this is the local variable scope
     System.out.println(x); will print local varibale scope
     fn(); this will make changes to the global and return it`s value
 }
 static void fn(){
     x = x+8;
     System.out.println(x);
 }
```

we say that the global varibale got shadowed in line 5

DSA IN JAVA DAY 9

variable arguments :-

this is when you create a method which take variable number of arguments

to do this while defiening the method
eg

example

```
public class feb_4_vargs {
//     public static void main(String[] args) {
//          fun(45,7,8,9,7,94,9,2,0,5,78,9,478);
//     }
//     static void fun(int ...v){
//         System.out.println(Arrays.toString(v));
//     }
```

here the method fun is saying to the compiler to take the 0 or more than zero arguments as nan array

example of zero arguments

```
public static void main(String[] args) {
    fun();
}
static void fun(int ...v){
    System.out.println(Arrays.toString(v));
}
```

will give an empty array

this concept is called VarArgs in java

one more example what if we take normal arugments and varibale length arguments both ??

```
public static void main(String[] args) {
    fun(2,3,"himesh","kyo","ronin");
}
static void fun(int a,int b,String ...v){
    System.out.println(Arrays.toString(v));
}
```

note that the order of the arguments is very important to prevent errors
also the varibale length arguments always come at the last

Method overloading

two methods of same name can exist if the parameters are different

example

```
public static void main(String[] args) {
    fun(15);
    fun("Himesh");
}
static void fun(int a ){
    System.out.println(a);
}
static void fun(String name ){
    System.out.println(name);
}
```

here the void function gives int

here the void function gives string

at compile time it decides which function to run based on the type of arguments and parameters

example with var args

```
public static void main(String[] args) {
    fun(3,4,5,7);
    fun("himesh","kyo","ronin","editing");
}
static void fun(int ...v){
    System.out.println("int wala fun");
    System.out.println(Arrays.toString(v));
}
static void fun (String ...n){
    System.out.println("String wala fun");
System.out.println(Arrays.toString(n));

}
```

here the method fun is same but its execution changes depeding upon the type of input given as arguments

what if we give no arguments and pass an emty array??

like this

```
Run | Debug
public static void main(String[] args) {
    fun();    The method fun(int[]) is ambiguous for the type feb_4_v
}
static void fun(int ...v){
    System.out.println(x: "int wala fun");
    System.out.println(Arrays.toString(v));
}
static void fun (String ...n){
    System.out.println(x: "String wala fun");
    System.out.println(Arrays.toString(n));

}
```

this will say that it is ambigous for the file

arrays in java

DSA day 11

why do we need arrays in java

if we want to store a roll number what we do is

```
int roll = 21
```

if we want to store somones`s name what we do is

```
String name = "Himesh"
```

but if we want to store roll umbers of 5 people doing that manually sucks

it would be awsome to have a data structure to store these collection of data types

and that data structure is array

what is an array?

an array is a data structure which consists of similar datatypes
it is a collection of data of similar data types

syntax of an array

```
dtype[] variable_name = new dtype[size]
```

or

```
dtype [] array_name = {items,}
```

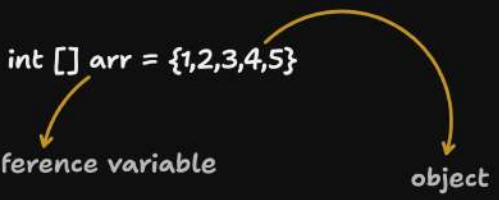
example :
store 5 integers

```
int[] roll_no = new int[5];
int [] roll_no2 = {21,22,78,98,12};
```

k3b

1. this data type is basically the data type of the array
2. all the type od data inside the array should be same

how does arrays work?



int [] arr

⟶ this is called declaration of arr

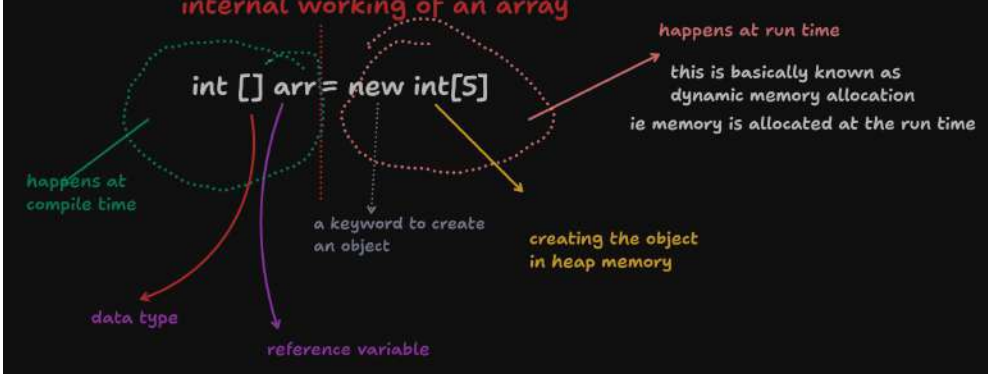
⟶ the arr is getting defined in the stack

arr = new int[5]

⟶ here the actual memory creation occurs

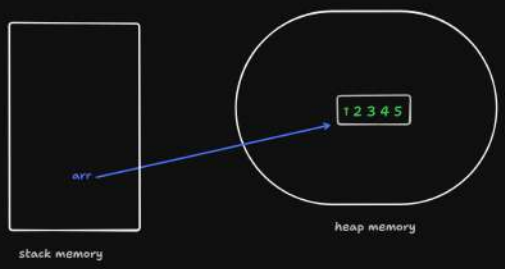
⟶ intialization of object occurs here

internal working of an array



Text

internal working of arrays



in c and c++ the array is a continous memory allocation

since java does not contain the concept of pointers accesing the meory adress and all is impossible

continuity of an array

wkt

1. all the objects including array are stored in heap
2. heap objects are not continous
3. all the memory is allocated at the runtime ie DMA

hence in java arrays may not be continous ⟶ depend on jvm

index of an array

index of an array are used to get the lements at that index position . they generally start from 0

arr[i] means the ith element of the given array

eg arr[0] = 3
arr [1] =8x

for an integer array if the elments are not mentioned all the elements will be 0

creating a string array :

string [] arr = new String[5] // creates a string array of size 5

if we try the same thing which we did with integer array it will return none

```
String[] arr = new String[5];
 System.out.println(arr); // will show null
```

what is null in java ?

it is a literal in java and similar to None in python

String str = null; ☑

int a = null; ☒

null a = ak; ☒

In Java, null is a keyword much like the other keywords public, static or final.
It is just a value that shows that the object is referring to nothing.

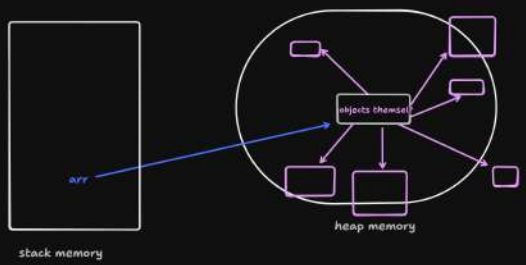
this can also be typecasted

null is used as defult

- in java all the primitives are stored in stack memory
- and all the other objects like string, data types are stored in heap memory

- when we create a String array it is basically an arrray of objects as string itself is a class

String[] arr = new String[6]



- these objects in heap memory all are objects
- and here arr[0] arr[1] etc all are the names of the ref variables
- since we didnt set any value to arr[0] it will be null by usual
- in non primitive data types the arrays are basically a colllection of reference varibales

DSA day 12
Arrays p2

Array input

consider an integer array of 5 elements

```
int[] arr = new int[5];
arr[0] = 45;
arr[1] = 36;
arr[2] = 78;
arr[3] = 96;
arr[4] = 78;
```

here the array is sotred like this [45,36,75,96,78]

```
System.out.println(arr[2]);
```

- but if we want to take input of each element of an array
- we iterate it over a for loop

eg :-

```
for (int i = 0; i < arr.length; i++) {
    arr[i] = sc.nextInt();
}
```

- to get the length of the array ⟶ array.length

// printing an array

similar to the above code instead of asking the value of array print it like this

```
System.out.println("the array is");
for (int i = 0; i < arr.length; i++) {
    System.out.println(arr[i]);
}
```

enhanced for loop

syntax is

```
for ( data_type ref_variable : array_name){
   sout( ref_variable)
}
```

here the refernce varibale refers to as element of the array
logic similar to python's

for i in array

eg

```
for (int element : arr) {
          System.out.println(element);
      }
```

to string() method :-

it is a function in the Arrays class which prints all the values of the array into a string and displays it at once

```
eg  int[] arr = new arr[5]
    // inputing the array
    for (int i = 0; i < arr.length; i++) {
       arr[i] = sc.nextInt();
    }

    // tostring method
    System.out.println(Arrays.toString(arr));
```

output :== [1, 2, 3, 4, 5]

Array of objects

it means array of string

by using the same method above we now create the array of string

```
  String[] str = new String[5];
  for (int i = 0; i < str.length; i++) {
      str[i] = sc.next();
}
System.out.println(Arrays.toString(str));
```

passing of an array in functions

already done this so easy it will be mutable the passed array

eg

```
    String[] str = new String[5];
    for (int i = 0; i < str.length; i++) {
        str[i] = sc.next();
}
System.out.println(Arrays.toString(str));
changeName(str);
System.out.println(Arrays.toString(str));

 // changes the valur of first element into my name
}
static void changeName(String[] arr){
    arr[0] = "Himesh";
}
```

2 - D arrays

what are 2 -D arrays :-

we can imagine it to be like a matrix

eg :- this matrix here has 3 rows and 3 columns
1 2 3
4 5 6
7 8 9

syntax of 2 d arrays

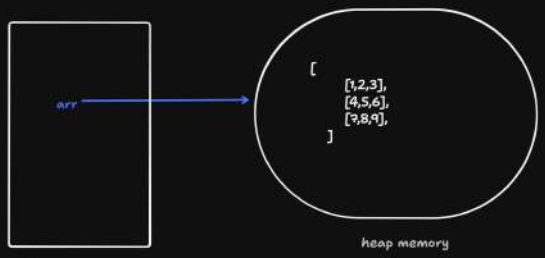
```
int[][] array_name = new int[size_of_rows][size_of_coumns];
```

k3b:
- it is not necesessary to define the size of columns
- even without defeining the size of coulmns the 2 d arrays will work
- but we have to definetly define the size of rows

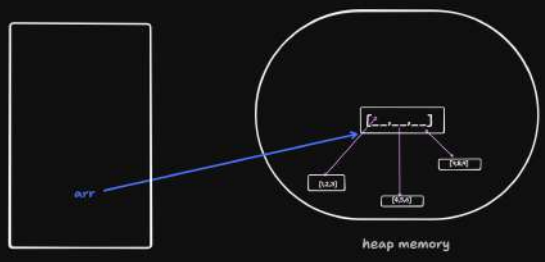
syntax of 2 d arrays in another way

```
int[][] 2d_arr = {        similar to numpys way of managing arrays
   {1,2,3},
   {4,5,6},
   {7,8,9},
};
```

internal woking of 2d arrays



think of this as an array of arrays



if we say that arr[1] this will return the array that is available at first index

so arr[1] ==> [4,5,6]

so when we do arr[1][0] this will give the 0th elemnt of arr[1]'s array ie 4 here

the reason why the size of colums does not matter is
the array of array can have any elements as it wishes

they can also be like this
{1,2,3},
{4,5},
{6,7,8,9},

input of 2d arrays

for a 2d array when we do arr.length it will give us the number of rows

so we know that in array of arrays arr[length] gives an array
and arr[length][index] gives a the elemnt at the index of that array
same concept we are going to use here :

taking input for a 2d array

```
int[][] ar = new int[3][3];
for (int row = 0; row < ar.length; row++) {
    for (int column = 0; column < ar[row].length; column++) {
        ar[row][column] = sc.nextInt();
    }
```

printing for a 2d array

here after the compleetion of the inner coloumn loop donr forget to add a new line to get a matrix type

eg

```
        for (int row = 0; row < ar.length; row++) {
        for (int column = 0; column < ar[row].length; column++) {
             System.out.print(ar[row][column]);
        }
        System.out.println();
    }
```

using Arrays.tostring()

```
      for (int i = 0; i < ar.length; i++) {
          System.out.println(Arrays.toString(ar[i]));
       }
       }
```

using enhanced for loop :-

```
for (int[] element : ar) {
   for (int element2 : element) {
      System.out.print( element2);
   }
   System.out.println();
}
```

for each integer array in ar:
  for int element in integer array:
    print element

DSA day 13
arrays part 3

ArrayList:-

consider the following array

```
int [][] arr = {
    {1,2,3,4},
    {5,6},
    {7,8,9}
};
```

to print the values of the above array the for loop below is used
as it is can be aplied to an array irresepctivtive of its row and column size

```
for (int row = 0; row < arr.length; row++) {
    for (int col = 0; col < arr[row].length; col++) {
        System.out.print(arr[row][col] + " ");
    }
    System.out.println();
}
```

if we do not know the size of array before-hsnd and want to add it at sometime then
that type of array is called dynamic array

for this we can use array lists

Array lists are the collection frmaework of java

the syntax of Array lists :-

```
ArrayList<dtype>  ar_name= new ArrayList<>(inital capacity); // creating the object of array list
```

in Array list you cannot pass primitives , you have to pass Wrapper classes

eg for integer Integer
for strings String etc

ArrayList functions:-

in array list there exists a function called ar_name.add()

it will all the value and convert into a string
it also internally uses the string output
eg :-

```
ArrayList<Integer> list  = new ArrayList<>(10);
list.add(12);
list.add(13);
list.add(12);
list.add(13);
list.add(23);
list.add(278);
list.add(2745);
list.add(2);
list.add(244);
list.add(24424);
list.add(2442465);
list.add(244246578);
list.add(24424654);
System.out.println(list);
```

we may get a doubt as if the inital capacity is just 10 then how are we able
to add more than 10 numbers

updating the array list

you can also update the values of of your list by using list.set function

list.set(index,value)

you can also checkout the different and various functions available in the array list

removing the values :-

we can also remove the value whihc is at a certain index by using list.remove(index) fucntion

input in arraylists:-

```
for (int i = 0; i < 5; i++) {
    list.add(sc.nextInt());
}
```

output or getting an index in array list :-

like how we did with arrays we cannot do the same and get the indices here

to get the values we hAVE TO uding get(index) function and pass in the index

```
for (int i = 0; i < 5; i++) {
    System.out.print(list.get(i));
}
```

Internal working of ArrayList<>

the internal concept of Array list is same as normal array of object
but the main important thing is how arraylist is able to store values beyond its size and why size doesn't matter ?

1. the size of an array listr is already fixed internally
2. say that the array list is filled upto a certain amount then what happens is.
    1. it creates a new arraylist of maybe double the size
    2. it copies the info of the first one into the other
    3. deletes the old array list

multi-dimensional array lists

```
ArrayList<ArrayList<Integer>> multi = new ArrayList<>(3);
// initialization
for (int i = 0; i < 3; i++) {
    multi.add(new ArrayList<>());
}
// add elements
for (int i = 0; i < 3; i++) {
   for (int j = 0; j < 3; j++) {
    multi.get(i).add(sc.nextInt());
   }

}
System.out.println(multi);
```

DSA day 14
linear search pI

what is searching ?

linear search is a techinque which i used to search for a certain element in a data structure

linear search :-

consider the following array

arr = {18,12,9,14,77,50} → un-sorted array

the question is to find whether the element 14 is in the array arr or not?
if no value is found return -1

my approach
loop 0: arr.length
if arr[i] = x
return true

there exists two ways to access elements in an array
1. arr[index]
2. using for each loop ie the enhanced for loop

linear search states that start searching from the start till you find the required element

time complexity:-

- for this linerar search the best time complexity is O(1) which means constant
- for this linear the worst possible time complexity is o(N) where n i sthe size of the array

what do yttttttou think is the best case when we are going to search for an element?

The best case:-

- the best case will be for the the element we are searching for is the first element

the amount of comparisions made in this case are :=

how many checks will the loop make in the best case

lets say the size of the array is 200

arr ={8,12,4,5,8,6,78,9.....}

here no of checks made is one that is arr[0] == target

arr = {18,56,45, ... 1 lakh items }

here also only one check is made

in the best case the amount of checks made is not dependent on the size of the array

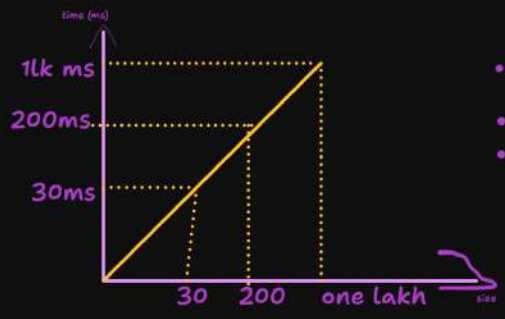
in the best case only one comparision is made and the size of the array doesn't matter

the worst case possible is

for the target element to be last element of the array

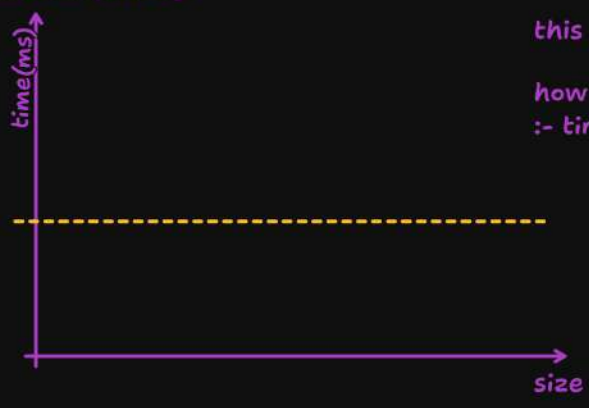
if the size of thae array is 30 ==> 30 comparisions assume that this takes 2 millisecond
is 200 => 200 comparisions this takes 25 milliseconds
is one lakh = 1 lakh comparisions tis takes 1 lkah milliseconds

graph for this worst possible case



- this is called linear time complexity
- as the input is growing how is the time
- going to take is called time complexity

constant time complexity



this is called constant time complexity

how the time grows as the input grows
:- time complexity in one line

questions

1. search in the array : return the index if item found otherwise if item not found return -1

the answer :-

```
    static int Linearsearch(int[] arr,int target){
        // this step below is one of the most
 important one
        if (arr.length==0) {
            return -1;
        }
        // running a for loop
        for (int i = 0; i < arr.length; i++) {
            if(arr[i] == target){
                return i;
            }
        }
        // if none of the return statement is
 executed ie the elemnt is not found
        return -1;
    }
```

search in strings

- because string is similar to an array of characters

by using the for loop the context of the code remians ismilar to the previous one searching a certain element

but when we try to use a for each looop a problem occurs
as we can only use for each loop over a set of array or iterables
so we convert the string into a character array like this

str.toCharArray() // this will convert the string into a character array

code: -

```
static boolean search(String str, char target){
    // in strings the length is a function becuase we used arrays ka varibale to create this string function
  if (str.length() ==0) {
    return false;
  }
  for (int ch = 0; ch<str.length(); ch++) {
    if (str.charAt(0)==target) {
        return true;
    }
  }
  return false;
}
static boolean foreachsearch(String str,char target){
          if (str.length() ==0) {
    return false;
  }
  // when trying to apply a for each loop in java
  it will say can only iterate over an array or an instance
//   for (char element : str) {
// therefore use tochararray
for (char ch : str.toCharArray()) {
    if (ch==target) {
        return true;
    }
  }
  return false;
}
```

searching in range

consider the given array here

arr = {18,12,-7,3,,14,28}
0 1 2 3 4 5

search for 3 in the range of index [1,4]

only the in and out conditions of the for loop is changed
nothing new here

Day 16 in DSA

Binary search

- it is an optimized way of search
- and is usually used in sorted arrays

in this topic as of now lets assume that the aray given is sorted

eg

arr = {2,4,9,10,12,14,18,19]

→ ascending order

or

arr ={19,18,14,12,9,4,2}

→ descending order

what happened in Linear search ?'';:

here also we can do linear search and the maximum comparisions in the worst case is N where N ===> no of elements

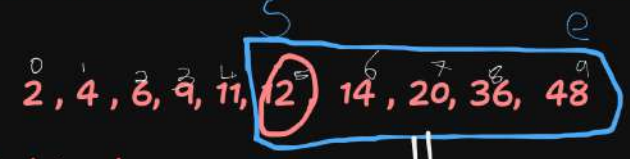
arr = [2,4,6,9,11,12,14,20,36,42]

middle element

consider the above array and find whether 36 exists or not

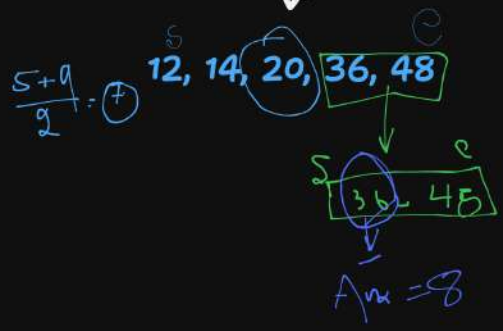
binary search is simple which states that

1. find the middle element
2. check if the target > middle element ==> search in right otherwise search =>left side
3. if middle element == target element ==> super mitsuketa

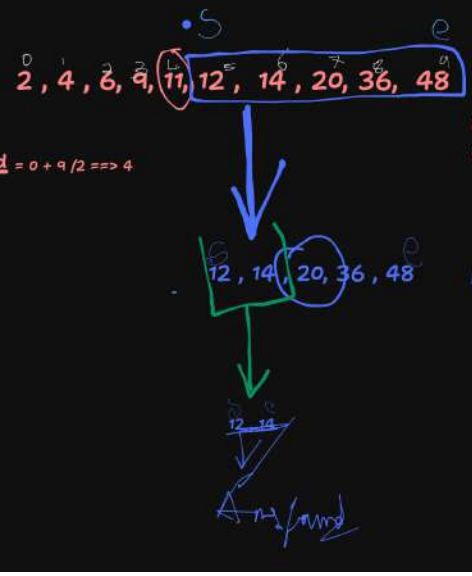


here the mid element is $\frac{\text{start} + \text{end}}{2}$ = 0 + 9 /2 ==> 4

note that we are not creating a new array
we are just changing the start and end points to check



suppose that we are searching for the element 12



here the mid element is $\frac{start + end}{2}$ = 0 + 9 /2 ==> 4

imp assumption if target < mid the start is not changing
if target > mid the end is not changing

if start > end : element is not found

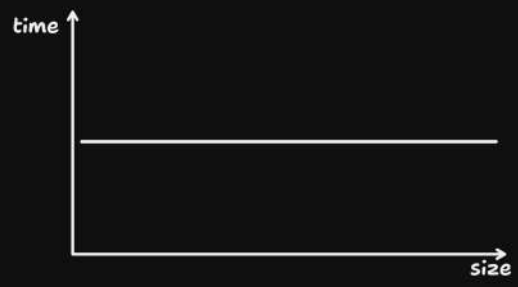
the first time we will check the first middle lemetn ie 0+9/2 == 4
if target > mid which happens here

here the mid element will get updated again ie 5 + 9/2 ==>7

if target is < mid which is hapenning here

why binary search ?

- if the first middle value is the value we are searching then it is the best case



for the best case O(n)

the worst case :-

q. find the maximum number of such comparisions in the worst case

consider the following array of size N

$$\frac{N}{2^K} = 1$$

$$\Rightarrow N = 2^K$$

$$\Rightarrow \log N = K \cdot \log 2$$

$$\Rightarrow K = \frac{\log N}{\log 2} \Rightarrow \log_2 N$$

the total comparisions in the worst case here is log N

assume that there is an array with 1 million element s

if the same question is done with linear search it will taken 1 million comparisions but by using binary search the amount of comparisions reduce to 20 now

this is the main reason to use binary search

the complexity is O(logn)

day 17
DSA

binary search

code for binary search

```
// return the index and return -1 if not exists
static int binarysearch(int[]arr,int target){
    int start = 0;
    int end = arr.length -1;
    // there is a problem that it might be possible that start + end excceds the range of int in java

}
```

so to resolve the size error we write int mid = start + (end-start)/2

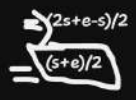
lets see how and why :???

the better way to find the mid

mid = $\frac{(start + end)}{2}$ — the problem is that there is a possibility for this value to exceed the range of int

so what we did is

$$int\ mid = start + \frac{end-start}{2}$$



the code for the binary code is below :-

```
static int binarysearch(int[]arr,int target){
    int start = 0;
    int end = arr.length -1;
    // there is a problem that it might be possible that start + end excceds the range of int in java
     while (start<=end) {
    // (int mid = start + end /2) no not feasible for our calcauations now
    int mid = start + (end-start)/2;

    if (target < arr[mid]) {
        end = mid-1;
    }
    else if(target > arr[mid]){
        start = mid +1;
    }
    else{
        // answer found
        return mid;
    }
    }
return -1;
}
```

order agnostic binary search

in the previous binary search we assumed that array is sorted and is in ascending order
but what if that is not the case

here also assume that the array is sorted but you dont know whether the array is sorted
in ascending order or descending order

before this lets try to ananlyze the changes that will be made to the code if
the given array is in descending order

lets assume the array given is

arr = [90 , 75 , 18, 12 , 6, 4, 3, 1]

mid = 0 + 7/2 ==>3

if the target> middle element ==> search space will be on left => end = mid-1

if the target < middle element ==> search space check for the right hand side => start = mid +1

now comes the main part how to determine whether the array is sorted in ascending order or descending order

take any two elements and compare

the best two numbers to search are the first and last ones right

if start < end : Ascending order
if start > end : Descending order

```
static int orderAgnosticBS(int[] arr, int target){
    int start = 0;
    int end = arr.length -1;
    boolean order = arr[start]<arr[end];
        while (start<=end) {
        // (int mid = start + end /2) no not feasible for our calcauations now
            int mid = start + (end-start)/2;
            if (target == arr[mid]) {
                return mid;
            }
            if (order) {
                if (target<arr[mid]) {
                    end = mid -1;
                }
                else {
                    start = mid+1;
                }
            }
            else{
                if (target>arr[mid]) {
                    end = mid -1;
                }
                else {
                    start = mid+1;
                }
            }
        }

    return -1;
}
```

Binary Search Questions

1. Ceiling of a number

Q Given an array arr = [2, 3, 5, 9, 14, 16, 18] → Sorted arr ⇒ B.Search

int target = 15

what is ceiling?

ceiling → Find the smallest number element in the array which is greater or = target.

if (target = 14) ⇒ ceiling = 14

if (target = 15) ⇒ 16

target = 4 ⇒ 5

1. get target
2. Find all the elements greater than the target
3. Take the shortest of that element

Say target = 9

(2) ⇒ [9, 14, 16, 18] ⇒ ceiling = 9

→ We can apply Binary Search

My try

→ if element not there in array → don't return -1

→ Take an arraylist, add all elements greater than or equal to the target } wrong kadhu! but time complexity O(n)

→ get the element

Soln

arr = [2, 3, 5, (9), 14, 16, 18] (target = 15)

target > mid

target < mid

target < mid

target > mid

return start

18 ⇒ loop breaks

Start & end pointer

e s
14 16 18

→ they are two pointers they are used to reduce the search space

→ after the loop breakdown

→ Start is becoming the answers which we need

→ as it is the next number greater than target it is the answer

⇒ if we need the floor then return ~~ans~~ end//

Start → answers ← end

• this is the point imp

• answers is always b/w start & end

Our answers what we want

end target Start // → Condition is while loop violated

§2 floor

floor ⇒ Greatest no in the array that is smaller or equal to target number

→ What if not any answer will be found ie the target is greater than the greatest number

ie if(target > arr[arr.length-1]) {

return -1;
}

e e e
0 1 2
[c f g]

§3 Smallest number

'eetCode 744

Given

→ char[] letters = [ ]

→ ascending order

→ target char = "a"

→ letters wrap around

→ return the smallest char in the letters Alphabetically

≃ similar to floor

→ ~~Ceil~~

Sol^n

letters wrap around

if letters = ['a','b'] target = 2

the return statement will change

if no element ⇒ return this

eg

arr = ['c', 'd', 'f', 'j'], target = j

(0, 1, 2, 3)

→ No char is larger than j ⇒ return arr[0] = 'c'

We can do this via modulo

Condition is violated :→ start = end + 1 ⇒ length of array = N

return S % N

or

if S == N:
    return 0

To wrap around use %

→ modulo converts overflow into circular indexing

⇒ Same concept of % when → next greater, wrap around, circular array

Q) Find first and last position of element in sorted array

Leet Code 34

Given

int[] nums in Ascending order ⇒ B.S

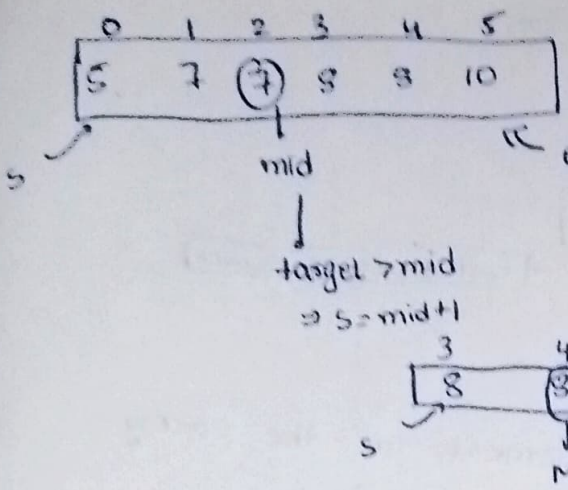
To find starting & ending position of a given target value

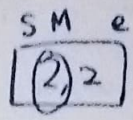
algo with O log(n)

eg [5, 7, 7, 8, 8, 10] target = 8

if not found → [-1, -1]

output : [3, 4]





target > m
Start = mid + 1

target < m
end = mid - 1

target = = m

[S, M]

target = = m

end = mid - 1

8	8

Sol^n

eg arr = [0 1 2 3 4 5 6 7
5, 7, 7, 7, 7, 8, 8, 10] target = 7

First - I = (1)
L. I. 7 = (4)

First Brute force → [5, 7, 7, 7, 7, 8, 8, 10]

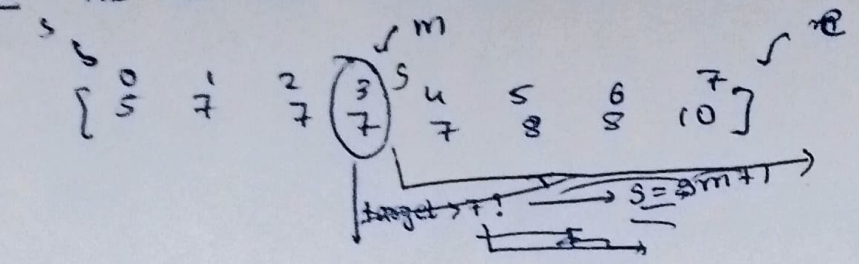
Start Comparing from both sides ⇒ ~~O(n)~~ O(n)

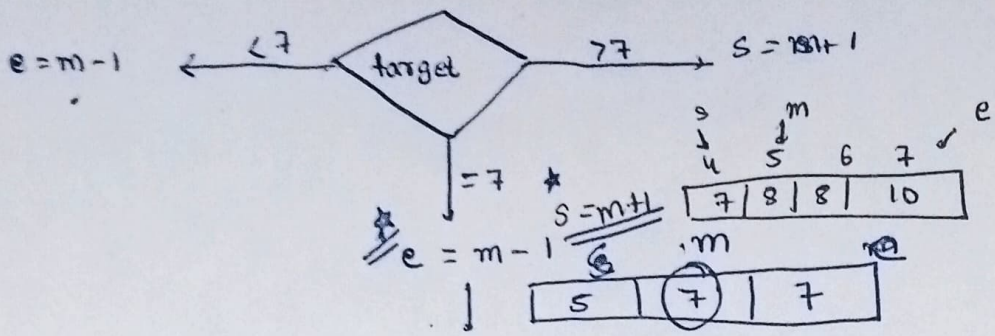
Better one:- → use Binary Search

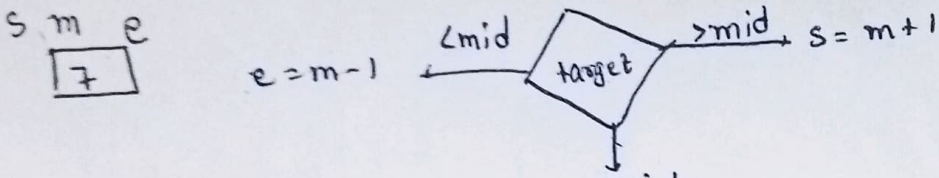
whenever we find the target →

→ run the binary search twice → first time for first index
Second time for last index

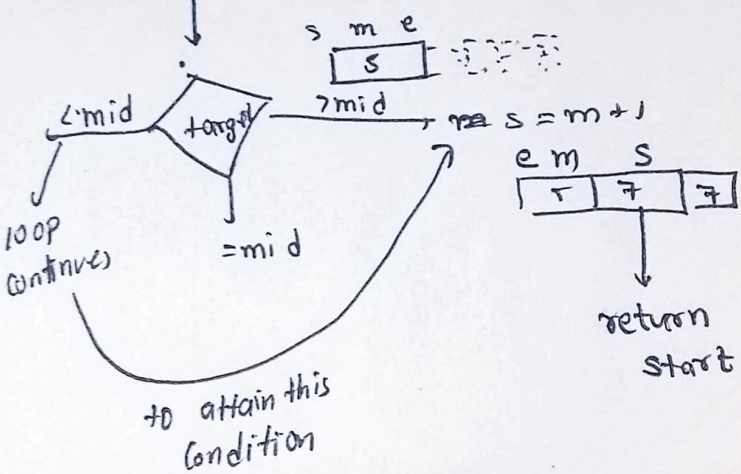
PI of the q :- Find the first occurance of 7 here :-







Wrong

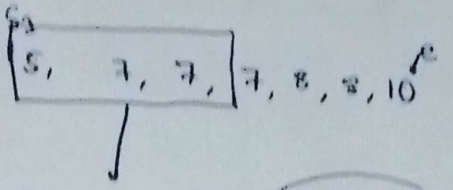


Solution :-

Find first occurence of 7

Sol)

First occurence of (7) → (mn - 1 = e)



Last occurance of (7) → (mn + 1 = s)

---

leetcode

Q Find the position of an element in a sorted array of infinite numbers

Doubt! ⇒ How to code even for an ∞ array!

① mimic :→ imagine that the given array is ∞ ⇒ don't use array.length function

or

② Take an array class

Eg

Say the array is

int[] arr = [2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 23, 30]

int target = 15

First :→ Brute Force

```
i = 0
while [arr[i] != 15]:
    i = i + 1
```

→ TC ⇒ O(N)

Second :→ use Binary Search (sorted)

↳ problem ⇒ we dont know the (start) & (end) ?
KYa?

⇒ question is reduced to get start & end indices

↳ check by chunks

⇒ Take a chunk of elements you want

like if i take a chunk of 6 element ⇒ ie K = 6

s2						e1	11	12	15	20	23	30
2	3	5	6	7	8	10						
0	1	2	3	4	5	6						

start = 0

end = 6 = K

if element in the chunk (by BS) → answer found

else ↓

move to next chunk

ie $s = 0 + K \Rightarrow$

$e = K + K - 2K$]

If element found ⇒ answer found

else ↓

move to the next chunk

ie $s = K + K \Rightarrow 2K$

$e = K + K + K \Rightarrow 3K$

Code:-

```
int K = 6;
int start = 0;
int end = k;

while (start <= end) {
    int mid = start + (end - start) / 2;

    If (target > arr [mid]) {
        start = mid + 1
    }
    else if (target < arr[mid]) {
        end = mid - 1;
    }
    else {
        start += K;
        end += k;
    }
}
return start
```

wrong

will

Shift the window here

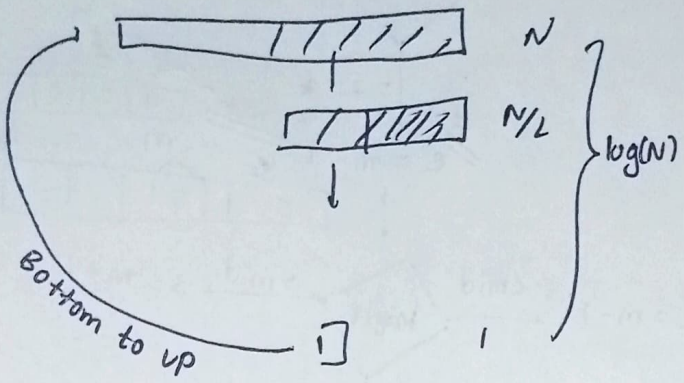
wrong

wrongs in this

- Fixed chunk size
  - ⇒ Cannot assume this
- The else block runs only if arr[mid] == target which is also wrong

Kunal's Solution -

- Use the chunks but dont keep the chunks constant
- The side difference b/w start & end call it window
- we can do this by expanding the window exponentially
- In Binary Search we do



→ Start from one element and double it each time to get the full array → it will also take logn steps

→ after expanding the array we check if the target element is lying there or not

s_1 e_1 s_2 e_2
2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 23, 30

→ since the target does not lie in the s_1 e_2, they will double the window size

→ The same process will continue & get's the range at which the element is there

→ Time complexity → same as B.S logn

Code

→ we will pass the start & end using the previous box

```
Static int ans finding range (int[] arr, int target) {
    // First Find the range
    // First start with a box of size 2
    int start = 0;
    int end = 1;
    // Condition for the target to be in the range
    // target is less than n, since it is a sorted array no
       need to check if taget is < start because
    // Keep doubling the size till the target > end
    while( target > ~~end~~ arr[end]) {
        ~~int newS = end+1;~~ int temp = end + 1
        ~~int end =~~
        // double the box value ⇒ end = previous end + Size of Box ⇒ 2
                                                        ↓
                                                    end-start+1
        end = end + (end - start + 1) * 2;
        start = temp;
    }
    return binary search (arr, target, start, end)
```

---

leetcode 852 Peak index in a mantain array

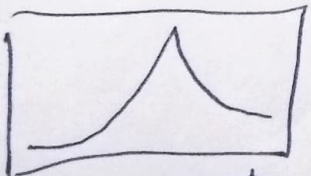
Given

→ int[] arr

length n

return the peak value ka index

O(log(n)) → Binary search



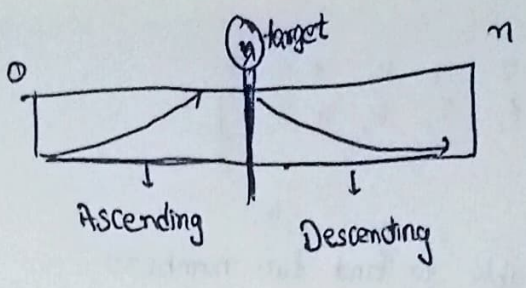
array elements

eg arr = [0, 1, 0] // 1

[0, 2, 1, 0] // 1

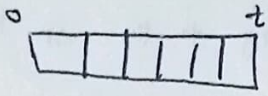
[0, 10, 5, 2] // 1

max element of array



A : s = 0
e = t

D: s = t
e = n




```
end = 1    ? max
while (target > end){
    end = end + ~~step~~(end - start+1)~~/~~2;
}
return end
```

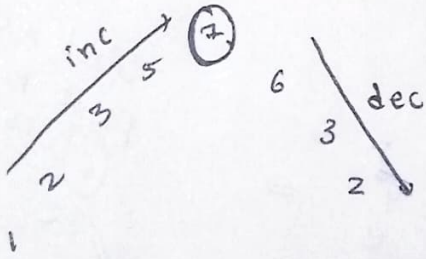
```
max -> target
max = arr[0]
if arr[i] > max
    max = ~~max~~ arr[i]
return max
```

nlogn x logn

Soln Bitonic array :- increases first and decrease later

eg

arr = [1, 2, 3, 5, 7, 6, 3, 2]



bitonic array

Find the peak in the mountain array

→ in short Find the max element

→ does not Contain duplicate element

arr = [0 1, 1 2, 2 3, 3 5, 4 6, 5 4, 6 3, 7 2]

s m e

→ If we were able to find two numbers

lets look at possibilities

element @ mid > e @ mid+1 → we are at Dec part of the array

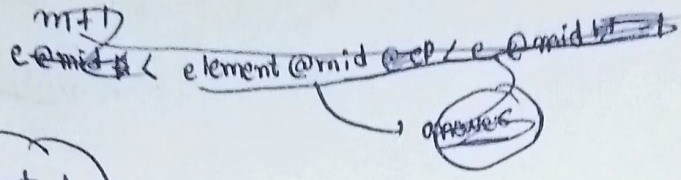
s m e then the answer may is going to lie on the left hand side

e = mid

element @ mid < e @ mid+1 → we are at inc part of the array

s = mid+1

s m e then the answer may is going to lie on the right hand side

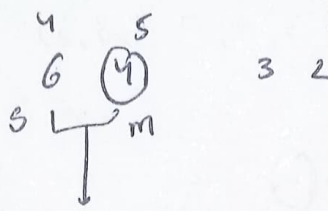


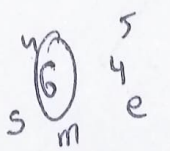
No target !

→ when will the loop break?

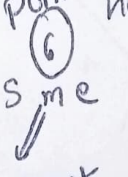
0 1, 1 2, 2 3, 3 5, 4 6, 5 4, 6 3, 7 2

s m e





⇒



Start & end will point to the largest num

loop will break at s == e

17 Find in mantain array

Leetcode 1095

Given

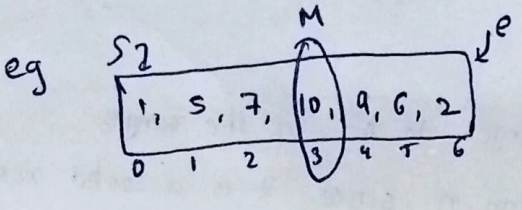
mantain array :→ mantainArr

if D.N.E

⇒ -1

mantainArr.get(index) == target

mantainArray.length() → returns the length



target =

target > Mid → s = M+1

target < mid → e = m-1

target = Mid

get the peak element

target = peak

int peak = peak();

if target < peak-1

target < peak

target < peak-1 ⇒ Asc

target > peak+1 ⇒ Desc

BSASO

s:0 e:M

BSDSO

s:mid+1 e→len

what i did:-

get the peak → checked target @ peak+1

→ < As → AS.BS

→ > Des → De.Bs

what's wrong ↘ interface used not array how will this affect my answer

arr = [0, 1, 2, 3, 4, 5, 6 / 1, 2, 3, 4, 5, 3, 1] ⇒ same approach as me

→ Find the peak

→ order agnostic BS ie B-S in as array ⇒ (0,4)

if not found Bs in dec array

leetcode 33 Search in Rotated Sorted Array

#Distinct values

- Earlier

[1,2,3,4,5,6,7] Ascending array → Rotated by K here eg K=3 → [4,5,6,7,0,1,2,3] → Return the index of target eg here target = 0 // 4

A Sorted array eg

arr = [2, 4, 5, 7, 8, 9, 10, 12]

↳ 1-rotation

After 1 rotation:-

arr = [12, 2, 4, 5, 7, 8, 9, 10]

↳ 2-rotation

After 2 rotation:-

arr = [10, 12, 2, 4, 5, 7, 8, 9]

Pivot

~~After~~

Approach no 1:- Find the Pivot in the array.

aka → the largest number in the array

Pivot ⇒ from where your nex no's are ascending

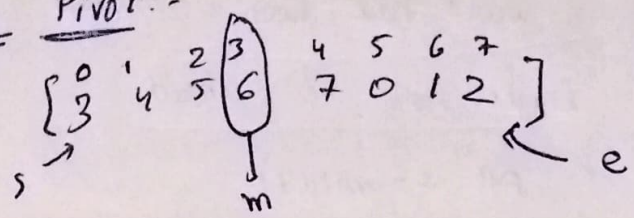
eg [3, 4, 5, 6, 7, 0, 1, 2, 3]

a > c | Pivot | a > c

* Find Pivot
* apply B.S → 0 : Pivot
→ Pivot+1 : end

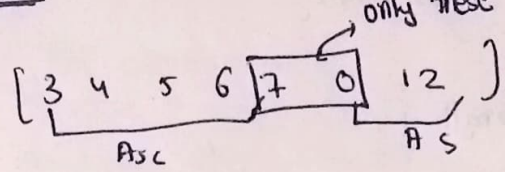
if Pivot is found ⇒ Bs.

Q⇒ Find Pivot:-



Compare: mid & mid+1

Ans when?



When we find that mid > mid+1 element ⇒ pivot. // Case 1

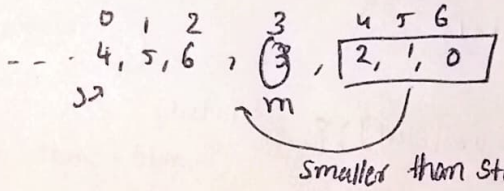
~~mid < mid+1 element~~

⇒ ~~start = mid+1;~~

If mid element < (mid-1) element // Case 2

⇒ also the answer ⇒ answer = (mid-1)

Start elemet => mid - element



// Case 3

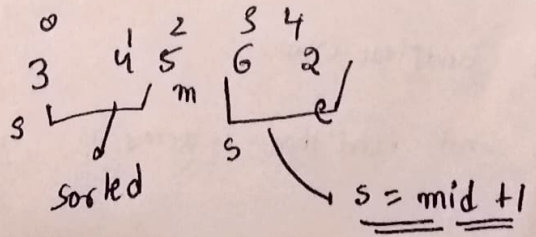
→ In this Case, all elements from mid will be < start hence we can ignore this since we are looking for peak ie largest element

→ we can ignore this by changing

(e = m-1)

// Case 4

Start element < mid element



→ If mid was pivot it would have been returned in case 2

→ Hence proved, that bigger no's lie ahead

Hence, ignore mid & put s = mid+1

Code

```
Static int findPivot(int[] arr){
    int start = 0;
    int end = arr.length-1;
    while (start <= end){
        int mid = start + (end-start)/2
        // 4 Case
        if (arr[mid] > arr[mid+1]){
            return mid;
        }
        if (arr[mid] < arr[mid-1]){
            return mid-1;
        }
        if (arr[mid] <= arr[start]){
            end = mid-1;
        }
        else{
            start = mid+1;
        }
    }
    return -1;
```

if any chance mid = end
then mid+1 will praide a
index out of bounds err

⇒ @ mid < end (&&) use and
(arr[mid] > arr[mid+1]

Similarly
mid > start && condition

```
Static int Search (int[] nums, int target){
    int pivot = findPivot(nums);
    // if pivot not found then ⇒ array is not rotated
    if (pivot == -1){
        // Binary search → returns binarySearch(nums, target, 0, nums.length-1)
```

if pivot is found, 2 asc sorted arrays are found

```
if (nums[pivot] == target){
    return pivot;
}
if( target > nums[0]) {
    ~~end~~ return Bs(nums, target, 0, pivot-1)
}
else {
    return Bs( nums, target, p+1, nums.length-1)
}
```

the ~~four~~ 4th cases when finding the answer

Case 1:- pivot element = target ||ths

Case 2:- target > start element

eg target = 6 ⇒ search space = (s, p-1)

why? → cus all the nos after Pivot are < start

Case 3:- target < start element

↳ we know that all elements from s to pivot are going to be bigger than target

⇒ search space = (p+1, e)

Rotated Binary Search:-

→ An array which was initially sorted was rotated

→ we have to find the target element

Brute force

→ Linear search O(n)

eg [3, 4, 5, 6, 7, 0, 1, 2] tar = 0

→ using Binary search because the array is sorted

if array was sorted

0, 1, 2, 3, 4, 5, 6, 7 ⇒ Basic Binary Search!

st ↑ mid ↑ end ↖

#Logic is to PICK search space

Rotated Sorted array:- one half is always sorted

[3, 4, 5, 6, (7), 0, 1, 2]

sorted sorted

↳ Pivot element

Have to find this pivot element:-

→ Since always in a rotated sorted array one half is sorted

arr[Start < mid] ⇐ left-Sorted → L→B.S

Right-Sorted → Start > mid arr → R→B.S

→ if the target lies between start & left ⇒ use B.S (st, mid-1) } left sorted

else Right (mid+1, end) → normal search

eg | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 |

→ if target lies between end & Right ⇒ use Bs (mid+1, end) } Right sorted

else

Left(start, mid-1)

Pseudo Code

```
Start = 0
end = n-1

while (start <= end){
   mid = st + (end-st)/2
  if (A[mid] = target) → mid
  if ( A[st] <= A[mid]) ⇒ //Left sorted
        Binary Search (st, mid-1)
```

else // ~~Right sorted~~

Right → st = mid+1

else // Right sorted

if (A[mid] <= target <= A[end]) → Right

start = mid+1

Binary Search (mid+1, end)

else

end = mid - 1

→ But Kunal's Answer is better than this if You can understand

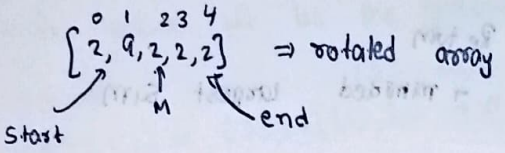
→ what if there are duplicate elements in the above question?

The above code will not work for duplicate values

Consider an array

The problem will be when

arr = [2, 2, 2, 2, 9] ⇒ non rotated

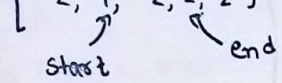


⇒ rotated array

→ Here all the start, mid & end are same

→ we can trim the array ie we can skip the same number from both the end

ie



#we can ignore as many duplicate elements as possible because pivot is the largest element

→ But Note what if these elements at start and end were the pivot?

→ check if start is pivot

```
if (arr[start] > arr[start+1]) {
    return start;
}
start ++;
```

→ // check if end is pivot

```
if (arr[end] > arr[end-1]) {
    return end;
}
```

// left side is sorted, so pivot should be in right

```
else if (arr[start] < arr[mid] || arr[start] == arr[mid] && arr[mid] > arr[end]) {
    start = mid+1;
```

```
}else{
    end = mid - 1;
    }
return -1;
```

Q Find the Rotation Count in Rotated sorted array

My approach

```
Start from Pivot, count = 0
while ( Pivot >= 0);
    Pivot --;
    count ++;
```

→ Yes! Got the answer by this

but more good way → the array was rotated by Pivot + 1 times

→ the rotation Count is always Pivot + 1 times

leetcode 410 - Split Array Largest Sum

Given

int[] nums

int k

Return → minimized largest sum

→ Split nums into k non-empty subarrays such that sum of any subarray is minimized

→ eg nums = [7,2,5,10,8] (0 1 2 3 4) k = 2

→ subarrays are continous part of the array

Completely wrong

0 - 1	2 - 4	
[7,2]	[5,10,8]	:way 1 ⇒ 9 , 23
[2,5]	[7,10,8]	: way 2 ⇒ 7 , 25
[5,10]	[7,2,8]	: way 3 ⇒ 15, 17 ⇒ not a sub array
[8,10]	[7,2,5]	:way 4 ⇒ 18 , 14

0 : k+1 | k-1 end

Solution:-

It is Binary search under the hood

The given array arr = [7, 2, 5, 10, 8] , m = 2

Can do the Splitting largest return the minimum of the answer

7, 2, 5, 10 & 8 ⇒ 24
Sum = 24

7, 2, 5 & 10, 8 ⇒ (18)
Sum = 14 Sum = 18

7, 2 & 5, 10, 8 ⇒ 23
Sum = 9 Sum = 23

- we have to first divide the sub-arrays ⇒ find sum of each sub array → get the max of each division → get the min of all

(1) what is the minimum no of partitions i can make?
⇒ min no of partitions = (1)

(2) what is the maximum no of partions i can make? ⇒ index of array
⇒ max no of partitions = N

eg arr = [3, 4, 1, 2] ⇒ [3], [4], [1], [2]

- what will be the answer in case (1) ⇒ ans = Sum of ~~all~~ the ~~only~~ entire array

in case 2 ⇒ The answer will be the greatest element of the array!

eg if m = 4
[3, 4, 1, 2] ⇒ [3], [4], [1], [2] ⇒ # 4 is the answer

Max value of ans of Question ⇒ Case (1)
Min value of ans of Question ⇒ Case (2)

minAns = max value in array
maxAns = sum of all values in array

for the question
arr = [7, 2, 5, 10, 8]
The range is [10, 32]

Here it clicked that BS will be used

Start = 10 end = 32

mid = (Start + end)/2 = 42/2 = (21) → Try to see if you can split the array with 21 as the max sum

7, 2, 5, 8, 10

[7, 2, 5, ~~8~~]

[8, 10]

Pieces
1 → 2

if the Total no of (pieces ≤ m) {
// Try to reduce 21
end = mid

S = 10, end = 21

mid = 15

(10)

[7, 2, 5, 8, 10]

[7, 2, 5], [8], [10]

pieces = 3

Here the pieces = 3

// Since pieces should be equal to the og given m

start = mid + 1

s = 16, end = 21

mid = (16+21)/2 ⇒ 37/2 = (18)

newmid = (18)

[7, 2, 5, 8, 10]

[7, 2, 5]

[8, 10]

pieces = = m

end = mid

s = 16 end = 18

mid = 17

[7, 2, 5, 8, 10]

[7, 2, 5]

[8]

[10]

pieces = 3

Check 2: pieces > m → # start = mid + 1

Check 1: pieces ≤ m → end = mid

start = 18, end = 18

mid = 18 ⟹ answer

when (start = mid = end) // answer

Solut → The answer exists definetly hence by the above 2 checks we will reach that answer

Code:-

```
public int splitArray(int[] nums, int m) {

    int start = 0;
    int end = 0;

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

        start = Math.max(start, nums[i]);
        // in the end of the loop the start will have the max element
          of the given array

        end += nums[i]
        // The end will have the sum of all the values of array
    }
    // Binary search
    while(start < end){
        // try for the middle as potential ans
        int mid = start + (end - start)/2;
        // calculate how many pieces we can divide this with max sum
        int sum = 0;
        int pieces = 1;

        for(int num: nums){
            if(sum + num > mid){
                // You cannot add this in sub array, make, new one
                // say you add this num in the new subarray, then sum = num
                    sum = num
                    pieces++
                } else {
                    sum += num;
                }
            }
        }
        if(pieces > m){
            start = mid+1;
        } else {
            end = mid;
        }

}
return start;
```

Search in matrices and Binary Search in 2D Array:-

Searching in matrices

Consider a 3×3 array

	0	1	2
0	18	9	12
1	36	-4	91
2	44	33	16

and the target element is 91

Logic → Simple Linear search

Pseudo code

```
for row{
    for col{
        if (arr[row][col] == target)
            return [i,j]
        }
}
```

The worst case will take N×N calculations

∴ Time complexity is $O(N^2)$

if the no. of rows = n and no. of columns = m then Time Complexity is $O(n \times m)$

matrix is sorted in row ~~and~~ wise and Column wise manner

↳ ie

	0	1	2	3
0	10	20	30	40
1	15	25	35	45
2	28	29	37	49
3	33	34	38	50

→ All rows are sorted

→ All the columns are also sorted

target = 37

No info in a question is waste
→ use them fully

⇒ First thought ⇒ Brute force;

⇒ They provided that the array is sorted ⇒ use this

when a large search space is provided try to minimize the search space

Case (1) if the element == Target ↝ ans found

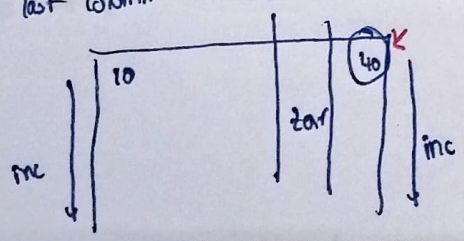
Case (2) if the element < target
row++;

Case (3) if the element > target
col--;

think about eliminating rows and Columns

the lower bound is element at 0,0 ie 10

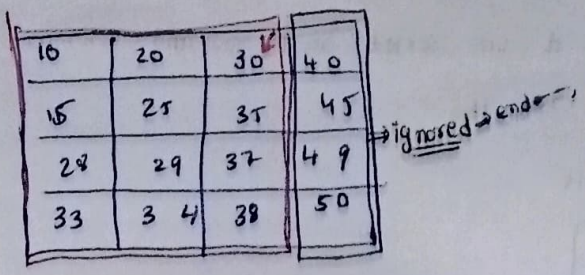
and the last column is the upper bound ie 40



if the target ≤ 40 and 40 is always the least in last col

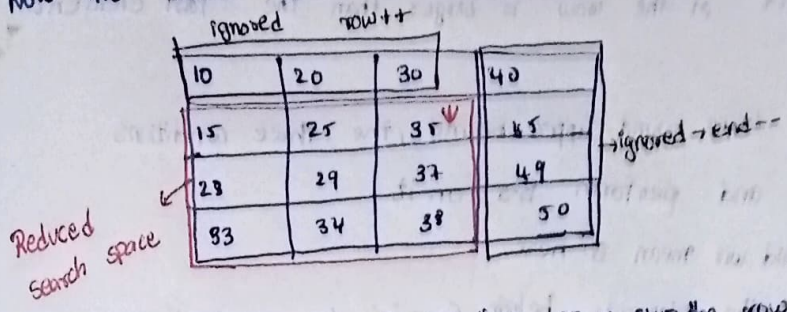
⇒ all the elements of the last column will be less greater than target ∴ ignore it

after this the remaining search space is



Here the next element 30 which is < target
⇒ all the elements left to 30 are obviously will be less than target
∴ skip that row = row++;

Now the search space is reduced to



the next element 35 is less than 37 ⇒ skip the row ⇒ row++;

∴ The answer = target ∴ ans found

→ The time complexity of this code ⇒ O(n)
col is decreasing ⇒ 2n times
row is increasing

Code

```
Static int[] Search (int[][] matrix, int target){

    int r = 0;
    int c = matrix.length - 1;

    while ( r < matrix.length && c >0){

        if(matrix[r][c] == target) {
            return new int[] {i,j};
        }
        if(matrix[r][c] < target){
            r++;
        }
        else{ col--; }        → if nothing → r++ c--
```

True Sorted matrix

- In the earlier model the thing was it was sorted acc to row and columns
- But a true sorted matrix looks like this.

eg Search in a true sorted matrix

arr =

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

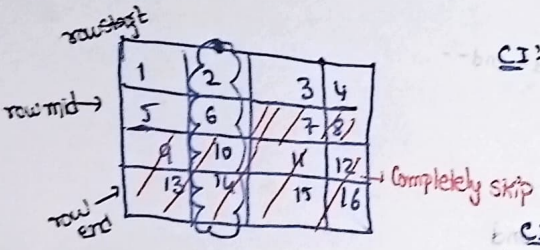
How to approach?

1. Brute Force
2. Convert into 1D array & apply BS

- observe that the first element of the row is larger than the last element of the previous row
- In Binary Search we need lower bound, upper bound, the three conditions
- lets take the middle column and perform B.S on it

Say the target = 2 so the mid will mean 6 here

Since the target < 6 ⇒ all the elements below 6 are also greater than 2
⇒ exclude them



CI: If element == target
ans found

CII: If element > target
// ignore rows after it

CIII: If the element < target
// ignore rows above it

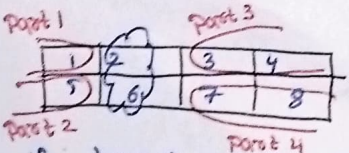
- In the end only two rows will be remaining

Bounds

row = 0 ↑ lower bound

row end ↑ upper bound

- In the end 2 rows are remaining:-



If target = 3 then not found

① Check whether the mid column you are at: Contains the ans

② Considers the four parts
⇒ ~~search~~
⇒ simple Binary Search on

Time complexity:

If there are n rows we will have log(n) to get the row no.
& we will take log(m) to get the column.

∴ Total Time complexity := O(log n + log m)

Time Complexity :-

if there are n rows ⇒ will take log(n) to get the rows
⇒ will take log(m) to get the columns

∴ Total Time Complexity :- $O(\log N + \log M)$

CODE

```
Public class answer{
Public static void main (String[] args){

static int[] Search (int[][], matrix, int target){

// It may be possible that the matrix is of one-dimension

    int rows = matrix.length
    int cols = matrix[0].length // be Cautious matrix may be empty

    if (rows == 1){
        return binarySearch(matrix, 0, 0, cols-1, target);
    }

    // run the loop till 2 rows are remaining
    while(

    int rStart = 0
    int rEnd = rows-1;
    int cMid = cols/2;

    while (rstart < (rEnd-1)){ // while this is true will have more than
    int mid = rstart + (rEnd-rstart)/2 ;

        if (matrix[mid][cMid] < target){
            rstart = mid;
        } else {
            rEnd = mid;
        }

    // now we have two rows
    //chek whether the target is in the column of two rows
    if (matrix[rstart][cMid] == target){
        return new int[] {rStart, cMid};
    }
```

matrix, CStart, target, row, cend

rows

since the value of Start gets updated to mid after previous step

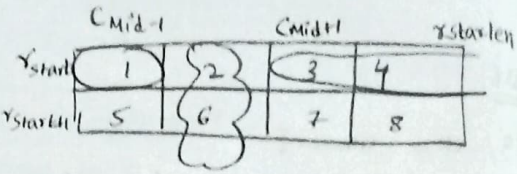
~~if (matrix[rstart][rend][cMid] == target){~~

~~return new int[] {rstart, cMid};~~

~~}~~

after updation the value of rstart will be mid
if the prev condition fails
we will check for its next value

```
if (matrix [rstart + 1][cMid] == target) {
    return new int[] {rstart+1, cMid}
}
```




```
// check Search in 1st half
if (target <= matrix[rstart][cMid-1] {
    return binary Search(matrix, rstart, 0, cMid-1, target);
}
if (target >= matrix[rstart][cMid+1]) {
    return binarySearch(matrix, rstart, cMid+1, col-1, target);
}
if (target <= matrix[rstart+1][cMid-1] {
    return binarySearch(matrix, rstart+1, 0, cMid-1, target);
} else {
    return binarySearch(matrix, rstart+1, cMid+1, col-1, target);
}
```

```
// Search in first half
if (target <= matrix [rstart][cMid-1]{
    return BinarySearch (matrix, rstart, 0, cMid-1, target);
}
// Search in Second half
if ( target >= matrix [rstart][cMid+1]  && target <= matrix[rstart][cols-1]) {
    return BinarySearch(matrix, rstart, cMid+1, cols-1, target)
}
```

The target should be b/w 3 and 4

```
// Search in third half
~~return Bina~~
if(target <= matrix[rstart+1][cMid-1]) {
    return BinarySearch (matrix, rstart+1, 0, cMid-1, target);
} else {
    return binarySearch (matrix, rstart+1, cMid+1, cols-1, target);
}
```

```
static int[] binarySearch(int[][] matrix, int row, int cStart, int cEnd, int target){

while (cStart <= cEnd){
    int mid = cStart + (cEnd - cStart)/2;
    if (matrix[row][mid] == target){
        return new int[] {row, mid};
    }
    if (matrix[row][mid] < target){
        cStart = mid+1;
    }
    else{
        cEnd = mid-1;
    }
}
return new int[] {-1,-1};
}
}
```

more simple way

Search in a 2D Matrix

1	3	5	7
10	11	16	20
23	30	34	60

target = 34

true → GI case

false → if does not exists

Brute force → linear search

- Each row is sorted in a non-decreasing order
- the first integer of each row is greater than the last integer of the previous row.

→ if we conside the range of row

row : range [start, end] → only compare the target with start and end value.

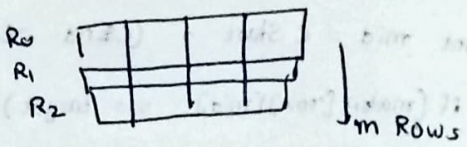
unique

non-overlapping

1, 3, 5, 7, 10, 11, 16, 20, 23, 30, 34, 60 → Flat karke single array liya

① Search the Correct row

→ Since we will search the row this can also be done by B-S



⇒ if element > target

Calculate mid

⇒ ~~R~~ end = mid - 1

⇒ if element < target

start = mid + 1

SRow = 0

ERow = m - 1

→ ~~return ans~~

SR = 0 eR = m - 1

mid = s + (e-s)/2;

① mid Row ⇒ target exist

mat[midR][0] =< tar <= mat[midR][n-1]

② tar > mat[midR][n-1]

SR = mid + 1

else

eR = mid - 1

② After getting the row ⇒ use BS again

Search in 2D Matrix II

Consider this array matrix

(0,0)

1	4	7	11	15	(arr.length, 0) end arr[0].length
2	5	8	12	19	
3	6	9	16	22	
10	13	14	17	24	
18	21	23	26	30	

=end

target < end

c--;

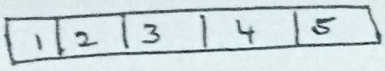
target > end

r++;

Kadane's Algorithm :- (LeetCode 53)

* subarray is a continous part of an array

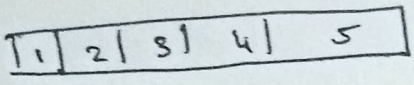
Maximum subarray sum

consider the arr = 

1, 2, 3, 4, 5 ⟶ sub array of size 1
12, 34, 23, 45 → sub array of size 2
123, 234, 345 → Sub-array of size 3
1234, 2345 ⟶ sub-array of size 4
12345 ⟶ sub-array of size 5

The number of subarrays for an array of index n is $\frac{n*(n-1)}{2}$

→ To get the subarray we would want all the possible start & end values

eg 

Start	end (st to n-1)
0	0, 1, 2, 3, 4
1	1, 2, 3, 4
2	2, 3, 4
3	3, 4
4	4

```
for(st=0 ; st<n ; st++){
    for(end=st, end<n, end++{
        ⇒(st to end) loop
    }
}
```

↘ Since we are using 3 loops the T.C of this has become $O(n^3)$

The code to print these sub arrays

```
Psvm{
    int [] arr ={1,2,3,4,5};
    for(int st=0, st<=arr.length; st++){
        for(int end=st, end<=arr.length; end++){
            for(int i=st; i<end; i++){
                System.out.println(arr[i]);
            }
            System.out.println("\t");
        }
        System.out.println();
    }
```

output

```
1  12  123  1234  12345
2  23  234  2345
3  34  345
4  45
5
```

Maximum subarray Sum :-

Brute force

↳ add all the elements of a subarray

↳ get the max out of it

{ [3, -4] [5, 4, -1] 7, -8 }

-1 8

→ Slight optimization:

3, -4, 5, 4, -1, 7, -8

st → sum = 3

end ↓

end++

3, -4, 5, 4, -1, 7, 8

st → sum ⇒ (3) (-4)

already got in previous iteration

⇒ sum = sum - 4

⇒ sum += -4

Algorithm

```
maxSum = 0;
for(st = 0; st < n; st++){
    CurrSum = 0 // Current Sum
    for(end = st; end < n; end++){
        cs += arr[end]   ← store the start value already present in previous iteration
        maxSum = max(cs, ms);
    }
}
return maxSum;
```

CODE :-

```
static int maxSubArraySum(int[] arr){
    int maxSum = 0;
    for(int st = 0; st < arr.length; st++){
        int CurrentSum = 0;
        for(int end = st; end < arr.length; end++){
            CurrentSum += arr[end];
            maxSum = Math.max(CurrentSum, maxSum);
        }
    }
}
```

Brute force approach

→ T.C ⇒ ~~O(log)~~

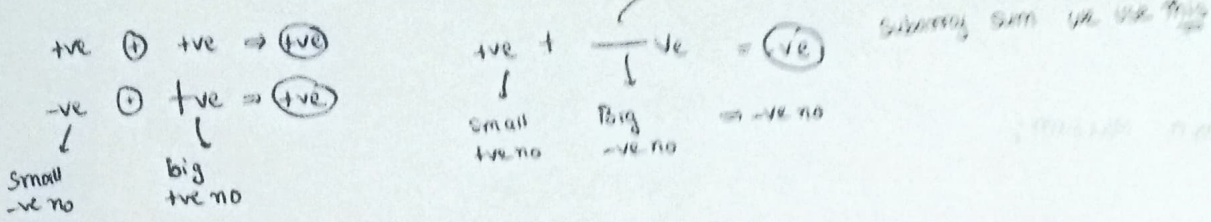
$O(n^2)$

Kadane's Algorithm

→ Most optimised

arr = { 3, -4, 5, 4, -1, 7, -8 }

Intution of Kadane's Algorithm:-



→ The approach is run a for loop on the arr, take 2 variables

CurrSum, MaxSum

add

→ update the values of the array to the CurrSum

if CurrSum < 0 ⟶ CurrSum = 0;

eg 3, -4, 5, 4, -1, 7, -8

CurrSum = 3

CurrSum = 3 + (-4) = -1 < 0 ⇒ CurrSum = 0

CurrSum = 0 + 5 ⇒ (5)

Pseudocode

```
CurrSum = 0 , maxSum = int-min
for(i = 0; i < n; i++){
    CurrSum += arr[i]
    maxSum = max(CS, MS)
    if(CS < 0)
        CS = 0
}
```

Dry run

CS = 0 3 -1 0 5 9 2 7

MS = - 0 3 3 5 (9)

Q Should the reset Current Sum be before maxSum or after maxSum line?

an edge case for answering this ?

arr = {-1, -2, -3, -4, -5}

already the maxSubSum ⇒ -ve

so if add the condition if (CS < 0) before the maxSum it will always return a 0

∴ upar wala kaam @ last

Code

```
static int kadane_algorithm(int[] nums) {
    int MaxSum = Integer.MIN_VALUE;
    int CurrentSum = 0;
    for(int i = 0; i < nums.length; i++){
```

```
MaxSum = Math.max(MaxSum, CurrentSum);

  if (CurrentSum < 0) {

     CurrentSum = 0;
    }

}
return MaxSum;

}
```