Program : **B.Tech**

Subject Name: **Data Structure**

Subject Code: **IT-303**

Semester: **3rd**

Subject Notes

IT 302- Data Structure

Unit-1

1.1 Introduction Data

1.1.1 Data and Data Item

Data are simply collection of facts and figures. Data are values or set of values. A data item refers to a single unit of value. Data items that are divided into sub items are group items; those that are not are called elementary items. For example, a student's name may be divided into three sub items – [first name, middle name and last name] but the ID of a student would normally be treated as a single item.

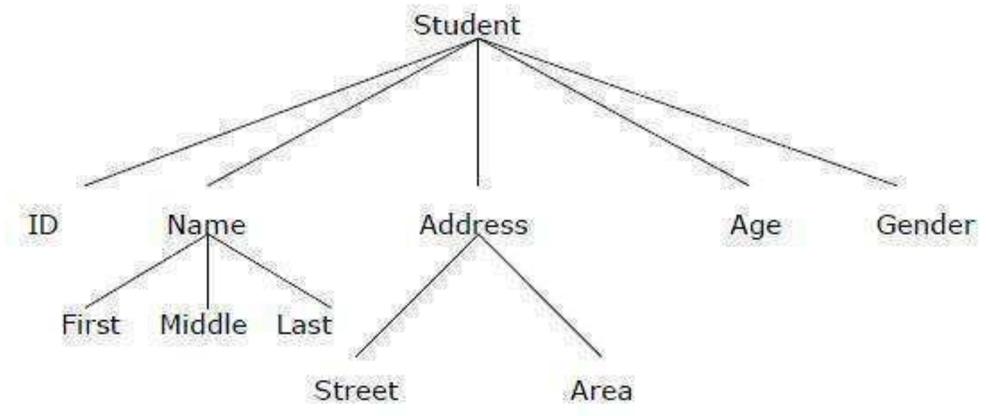


Fig 1.1 Example of Data item

In the above example (ID, Age, Gender, First, Middle, Last, Street, Area) are elementary data items, whereas (Name, Address) are group data items.

Data Definition defines a particular data with the following characteristics.

- **Atomic** − Definition should define a single concept.
- **Traceable** − Definition should be able to be mapped to some data element.
- **Accurate** − Definition should be unambiguous.
- **Clear and Concise** − Definition should be understandable.

1.2 Data Type

Data type is a classification identifying one of various types of data, such as floating-point, integer, or Boolean, that determines the possible values for that type; the operations that can be done on values of that type; and the way values of that type can be stored.

1.3 Data Object

A data object is a region of storage that contains a value or group of values. Each value can be accessed using its identifier or a more complex expression that refers to the object. In addition, each object has a unique data type. The data type of an object determines the storage allocation for that object and the interpretation of the values during subsequent access. It is also used in any type checking operations. Both the identifier and data type of an object are established in the object declaration.
An instance of a class type is commonly called a class object. The individual class members are also called objects.

1.4 Types of Data Structures:

A data structure can be broadly classified into 2 types:(i) Primitive data structure
(ii) Non-primitive data structure

1.4.1 Primitive data structure

The data structures that are directly operated upon by machine level instructions i.e. the fundamental data types such as int, float, double ,char are known as primitive data structures. It is the basic data type that is provided by the programming language with built -in support. This data type is native to the language and is supported by machine directly.

- Integers
- Boolean (true, false)
- Floating (Decimal numbers)
- Character and Strings

1.4.2 Non-primitive data structure

The data structures, which are not primitive i.e which can be derived from primitive are called non-primitive data structures. There are two types of non-primitive data structures.

1.Linear Data Structures
2.Non Linear Data Structures

1.4.3 Linear Data Structures:- In linear data structures, elements are arranged in linear fashion. The linear ordering is maintained either through consecutive memory locations or by means of pointers .Arrays, linked lists, stacks and queues are examples of linear data structures in which values are stored in a sequence. Eg: Stacks , Queues, Lists, Arrays.

There are basically two ways of representing such linear structure in memory.

a) One way is to have the linear relationships between the elements represented by means of sequential memory location. These linear structures are called arrays**.**

b) The other way is to have the linear relationship between the elements represented by means of pointers or links. These linear structures are called linked lists**.**

The common examples of linear data structure are arrays, queues, stacks and linked lists.

Array: Array is a collection of data of same data type stored in consecutive memory location and is referred by common name**.**

Linked list: Linked list is a collection of data of same data type but the data items need not be stored in consecutive memory locations.

Stack: A stack is a Last-In-First-Out linear data structure in which insertion and deletion takes place at only one end called the top of the stack.

Queue: A Queue is a First in First-Out Linear data structure in which insertions takes place one end called the rear and the deletions takes place at one end called the Front.

1.4.4 Non-linear Data Structure:- Elements are stored based on the hierarchical relationship among the data. The data values in this structure are not arranged in order. Tree, graph, table and sets are examples of non-linear data structures.

The following are some of the Non-Linear data structure:

Trees: Trees are used to represent data that has some hierarchical relationship among the data elements.

Graph: Graph is used to represent data that has relationship between pair of elements not necessarily hierarchical in nature. For example electrical and communication networks, airline routes, flow chart, graphs for planning projects

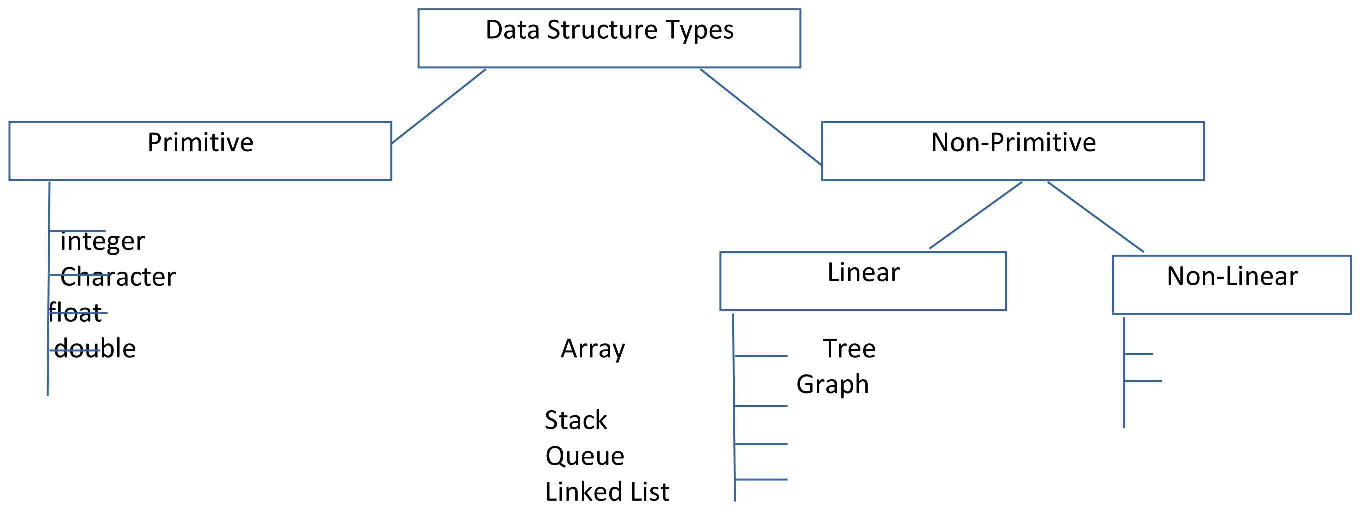


Fig 1.2 Classification of Data Structure

The data structures can also be classified on the basis of the following characteristics:

Characteristic	**Description**
Linear	In Linear data structures,the data items are arranged in a linear sequence. Example: **Array**
Non-Linear	In Non-Linear data structures,the data items are not in sequence. Example: **Tree**, **Graph**
Homogeneous	In homogeneous data structures,all the elements are of same type. Example: **Array**
Non-Homogeneous	In Non-Homogeneous data structure, the elements may or may not be of the same type. Example: **Structures**
Static	Static data structures are those whose sizes and structures associated memory locations are fixed, at compile time. Example: **Array**
Dynamic	Dynamic structures are those which expands or shrinks depending upon the program need and its execution. Also, their associated memory locations changes. Example: **Linked List created using pointers**

1.5 Operation On Data Structures: -

The four major operations performed on data structures are:

(i) Insertion: - Insertion means adding new details or new node into the data structure.

(ii)Deletion: - Deletion means removing a node from the data structure.

(iii)Traversal:- Traversing means accessing each node exactly once so that the nodes of a data structure can be processed. Traversing is also called as visiting.

(iv)Searching: -Searching means finding the location of node for a given key value.

Apart from the four operations mentioned above, there are two more operations occasionally performed on data structures. They are:

(v) Sorting: - Sorting means arranging the data in a particular order.

(vi)Merging: -Merging means joining two lists.

Algorithm
A well-defined computational procedure that takes some value, or a set of values, as input and produces some value, or a set of values, as output. It can also be defined as sequence of computational steps that transform the input into the output.

An algorithm can be expressed in three ways:-
1. in any natural language such as English.
2. in pseudo code or
3. in the form of a flowchart.

1.6 Complexity analysis
Algorithmic efficiency are the properties of an algorithm which relate to the amount of resources used by the algorithm. An algorithm must be analyzed to determine its resource usage. Algorithmic efficiency can be thought of as analogous to engineering productivity for a repeating or continuous process. For maximum efficiency we wish to minimize resource usage. However, the various resources (e.g. time, space) can not be compared directly, so which of two algorithms is considered to be more efficient often depends on which measure of efficiency is being considered as the most important, e.g. is the requirement for high speed, or for minimum memory usage, or for some other measure. It can be of various types:

- Worst case efficiency: It is the maximum number of steps that an algorithm can take for any collection of data values.

- Best case efficiency: It is the minimum number of steps that an algorithm can take any collection of data values.

- Average case efficiency: It can be defined as the efficiency averaged on all possible inputs.

The complexity of an algorithm describes the efficiency of the algorithm in terms of the amount of the memory required to process the data and the processing time.
Complexity of an algorithm is analyzed in two perspectives: **Time** and **Space**.

1.7 Time Space Trade-off
The best algorithm to solve a given problem is one that requires less memory space and less time to run to completion. But in practice, it is not always possible to obtain both of these objectives. One algorithm may require less memory space but may take more time to complete its execution. On the other hand, the other algorithm may require more memory space but may take less time to run to completion. Thus, we have to sacrifice one at the cost of other. In other words, there is Space-Time trade-off between algorithms.
If we need an algorithm that requires less memory space, then we choose the first algorithm at the cost of more execution time. On the other hand if we need an algorithm that requires less time for execution, then we choose the second algorithm at the cost of more memory space.

1.8 Algorithm Efficiency
Time Complexity: Time complexity of an algorithm is the amount of time it needs in order to run to completion. It's a function describing the amount of time required to run an algorithm in terms of the size of the input. "Time" can mean the number of memory accesses performed, the number of comparisons between integers, the number of times some inner loop is executed, or some other natural unit related to the amount of real time the algorithm will take.
Space Complexity: Space Complexity of an algorithm is the amount of space it needs in

order to run to completion. It’s a function describing the amount of memory an algorithm takes in terms of the size of input to the algorithm. We often speak of "extra" memory needed, not counting the memory needed to store the input itself. Again, we use natural (but fixed-length) units to measure this. Space complexity is sometimes ignored because the space used is minimal and/or obvious, however sometimes it becomes as important an issue as time.

1.9 Asymptotic Notations

1.9.1 Asymptotic

It means a line that continually approaches a given curve but does not meet it at any finite distance.

Example: x is asymptotic with x + 1 as shown in graph. Asymptotic may also be defined as a way to describe the behavior of functions in the limit or without bounds.
Let f(x) and g(x) be functions from the set of real numbers to the set of real numbers. We say that f and g are asymptotic and write $f(x) \approx g(x)$ if

$$\lim_{x \to \infty} f(x) / g(x) = c \text{ (constant)}$$

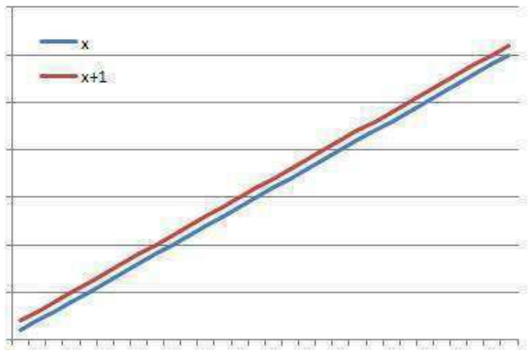


Fig 1.3 asymptotic bound

1.9.2 Big-Oh Notation (O)

The O(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or longest amount of time an algorithm can possibly take to complete.

Definition:A function f(n) is said to be in O(g(n)), denoted $f(n) \in O(g(n))$, if f(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer n0 such that $f(n) \le cg(n)$ for all $n \ge n0$

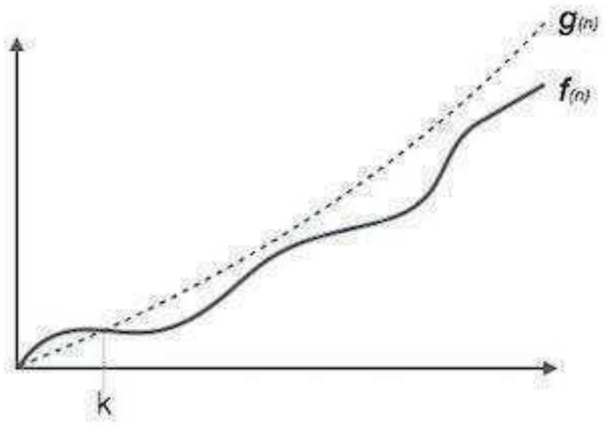


Fig 1.4 Big-Oh Notation

For example, for a function *f*(n) O(*f*(n)) = { *g*(n) : there exists c > 0 and n0 such that *g*(n) ≤ c.*f*(n) for all n > n0. }

1.9.3 Omega Notation, Ω

The Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or best amount of time an algorithm can possibly take to complete.

Definition:A function f (n) is said to be in Ω (g(n)), denoted f(n) ∈ Ω (g (n)), if t (n) is bounded below by some constant multiple of g (n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer n0 such that f(n) ≥ cg(n) for all n ≥ n0

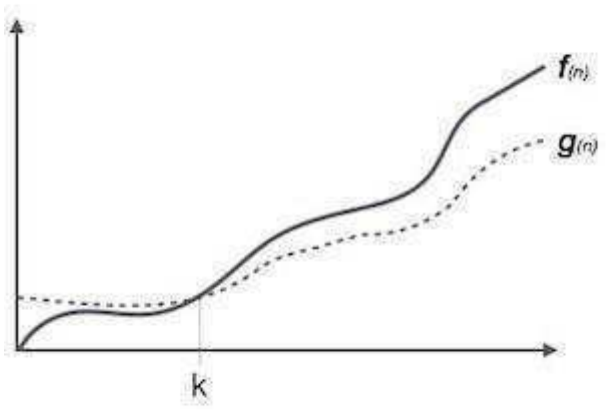


Fig 1.5 Big-Omega Notation

For example, for a function *f*(n)Ω(*f*(n)) ≥ { *g*(n) : there exists c > 0 and n0 such that *g*(n) ≤ c.*f*(n) for all n > n0. }

<u>1.9.4 Theta Notation, θ</u>

The θ(n) is the formal way to express both the lower bound and upper bound of an algorithm's running time.

Definition:A function f(n) is said to be in Θ(g (n)), denoted t(n) ∈ Θ (g (n)), if f(n) is bounded both above and below by some constant multiple of g (n) for all large n, i.e., if there exist some positive constant c1 and c2 and some nonnegative integer n0 such thatc2 g (n) ≤ t (n) ≤ c1 g (n) for all n ≥ n0. It is represented as following θ(*f*(n)) = { *g*(n) if and only if *g*(n) = O(*f*(n)) and *g*(n) = Ω(*f*(n)) for all n > n0. }

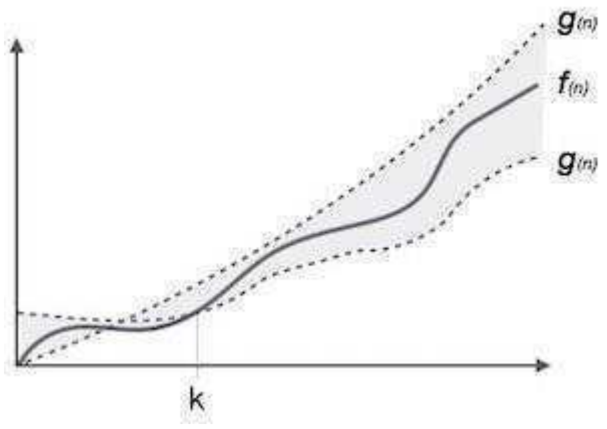


Fig 1.5 Big-Theta Notation

First Java Program - Input/Output, Debugging and Datatypes

10 February 2023 21:23

OUTPUT

```
public class Main
 */(create every class name start with Capital letter "it is a convention)"/*
{
Public static void main(String[] args){
System.out.println("HELLO WORLD");

    }
}
```

//the public class name should always be same as the file name
//we want to run this function without creting a object of "Main" that's why we need static .
//"Sting []args" means it is like a array , that whatever the arguments are given that stores in this array.
// "System OUT.println" the System is a class in which "out" is like refference variable for printstream (it connects it to the procedure 'code' written by java creators for output process)

PRIMITIVE DATA TYPE are like the data type which is not converted into Other data type after dividing it LIKE "String is not a primitive data type" Because it can be converted into CHAR data type after dividing it.

HOW INPUT WORKS

//in this program "Scanner" is a public class which is created by java creaters .
//"Scanner input "takes input from us
//newScanner() is like a constructor ;
//(System.in) is to take input from keyboard .
//we can put other file names to take input from it like (like file handling in cpp)

```
importjava.util.Scanner;
publicclassMain{

publicstaticvoidmain(String[]args){
Scannerinput=newScanner(System.in);
```

For taking input

```
Import java.util.Scanner;
Publicclass Inputs{
publicstaticvoidmain(String[]args){
Scannerinput=newScanner(System.in);
System.out.print("please enter the rollno");
introllno=input.nextInt();
System.out.println("your rollno is"+rollno);
}
}
```

```
System.out.println(input.nextLine());
}
}
```

FOR SIMPLE TYPCASTING :

```
publicclasstypecasting{

publicstaticvoidmain(String[]args){
//Scannerinput=newScanner(System.in);

intnum=(int)(74.87f);
System.out.println(num);
}
}
```

JAVA follows the Unicode principle so that you can put all the languages easily injava

Java automatically promoting from byte to integer whenever you performing expresions
Like these

```
publicstaticvoidmain(String[]args){
bytea=30;
byteb=40;
intc=30*40;
System.out.println(c);
}
```

It promotes the result of any expression into the resulting type or the which have bigger precedence over other data types.

Like // float + int - double = result show in double because of higher precedence.

Conditionals and Loops + Calculator Program

16 February 2023 21:14

We use for loop when we know the condition (like n<10)
But we use while loop when dont know about condition.

QUE) **Take INDEX as a input from the user and print there fibonaaci number?**

```
Import java.util.Scanner;

Public class fib_num{
publicstaticvoidmain(String[]args){
Scannerinput = newScanner(System.in);
intfib=input.nextInt();
In ta=0;
Int b=1;
Int count=2;
while(count<=fib) {
Int temp=b;
b = a+b;
a = temp;
count++;        }
System.out.println(b);
}
}
```

JRE is java runtime environment which buids the environment in the computer machine to run the java program .
Or we can say it invokes the machine to run java file.

JDK is a java Development tool kit that develops applications in JAVA

JVM is a java virtual machine which converts the byte code into machine code .

QUE) print how many four present in this number 844234 .

```
                                        {
Public static void main(String [] args){
Intn = 844234;
intcount=0;

while(n>0){
Int remainder = n%10;
  if(remainder == 4){
count++;
}
n=n/10;
}
System.out.println(count);

}
```

Que) print the given number into its reverse order.

```
Public class reverse{
Public static void main(String [ ] args){
Int n = 945894;
Int ans = 0;

While (n>0) {
Int rem=n%10;
n = n/10;

ans = ans*10 + rem;

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

Logic behind n=n/10;is

n/10 = 83,99,893.9

And th .9 is eliminted because int only prints without decimal values.

.charat(0) means charater at 0 th index of the string .
.trim() means remove the extra spaces around the string or the value .

Functions or Methods in Java

19 February 2023 16:10

# FUNCTIONS ARE KNOWN AS METHODS IN JAVA
#FUNCTION IS CALL BY () BRACKETS
#IN JAVA THERE IS NO PASS BY REFERENCE THERE IS ONLY PASS BY VALUE

#Any thing initialized out side of the scope can be use inside the block { } or scope but anything initialised into the block cannot be initialised outside of the block or also cannot be used.

RETURN METHOD :

```
importjava.util.Scanner;
publicclassMethods{
publicstaticvoidmain(String[]args){
Int  ok= Sum ();
System.out.println(ok);

}
staticintSum(){
Scannerinput=newScanner(System.in);
System.out.println("enternumberone");
intnum1=input.nextInt();
System.out.println("enternumbertwo");
intnum2=input.nextInt();
intsum;
sum=num1+num2;
returnsum;
}
}
```

METHODS USING PARAMETERS:

```
Import java.util.Scanner;
Public class Methods{
Public static void main(String[]args){
Scanner input = new Scanner(System.in);
String Movie = input.nextLine();


System.out.println(movie(Movie));

}
static String movie(Stringname){
String statement = name+" is a good movie";
Return statement;
}
```

Public class Scope {
Public static void main(String []args){
int a = 10;
Int b = 20;
  {   a = 34; //already initialized in same method you cannot initialized it again.

Pvsm(){
String naam = " priyanshu";
Name(naam);
System.out.println(name)
System.out.println(nom) // this will give us an error because we cannot access nom (variable or object).

This is known as scopping !
}

```
  {    a = 34; //already initialized in same method you
cannot initialized it again.
      b = 26;   // the values in this block is only remain in
this block cant be accesed outside of it
}
```

This is known as scopping !

```
}
```

```
Static void name (string nom){
Nom = priyank;

}
```

Any thing initialized out side of the scope can be use inside the block { } or scope but anything initialised into the block cannot be initialised outside of the block.

SHADOWING

```
Publicclass  shadowing{
Static in tx =36;
publicstaticvoidmain(String[]args){
System.out.println(x);     //output 36
Int x =45;
System.out.println(x);     //output 45
shadow();
}
Static int shadow(){
System.out.println(x);    // output 36
return 0;
}

}
```

Shadowing variable is accesed in all main function and also in outer function (like in shadow)

VARIABLE ARGUMENTS (VARARGS)

Variable arguments (varargs) is used for taking multiple output from the function in a string array we can it is INVOKED BY TYPING (...v) in the parameter column we can also give input in strings type array by typing (string ...v) and the (..v) variable th arguments should always come at the end

```
importjava.util.Arrays;

publicclassVar_args{
publicstaticvoidmain(String[]args){
ok(34,4,5,6,4,3,4,5);

}
staticvoidok(int...v){
System.out.println(Arrays.toString(v));

}
}
```

String and arguments

```
importjava.util.Arrays;

publicclassVar_args{
publicstaticvoidmain(String[]args){
ok(4 ,5 , "rahul" , "vicky" , "kaushal");

}
staticvoidok(int a , int b , string...v){
System.out.println(Arrays.toString(v));

}
}
```

Q) Find the richest wealth of the the account, there are three accounts given in the for of 2D array .

```
staticintmax(int[][]wealth){
```

```
intans=Integer.MIN_VALUE;
for(inti=0;i<wealth.length;i++){
intsum=0;
for(intj=0;j<wealth.length;j++){
sum+=wealth[i][j];

}
if(sum>ans){
ans=sum;
}

}
returnans;
}
}
```

Binary search

25 March 2023 22:34

Why Binary search :

The total no of comparisons in binary search is logN and in linear searchis n,
For example
If the size of the array is 1 millon , linear search will take 1million comparisons in the worst case and the binary search will take log2(100000) = 16.60964047443681 , 16 - 20 comparisons.

```
packagebinary_search;

publicclassbinaryS{
publicstaticvoidmain(String[]args){
int[]arr={-18,-12,-4,0,2,3,4,15,16,18,22,45,89};
inttarget=15;
intans=binarysearch(arr,target);
System.out.println(ans);


}
staticintbinarysearch(int[]arr,inttarget){
intstart=0;
intend=arr.length-1;
while(start<=end){
intmid=start+(end-start)/2;
if(arr[mid]>target){
end=mid-1;
}
elseif(target>arr[mid]){
start=mid+1;
```

- Q)To print the ceiling of the number.
  Ceiling = smallest element in array greater or = to the target element .
- And for floor just write return end in the end of the code.

```
package com.kunal;

public class Ceiling {

  public static void main(String[] args) {
    int[] arr = {2, 3, 5, 9, 14, 16, 18};
    int target = 15;
    int ans = ceiling(arr, target);
    System.out.println(ans);
  }

  // return the index of smallest no >= target
  static int ceiling(int[] arr, int target) {

    // but what if the target is greater than the greatest number in
the array
    if (target > arr[arr.length - 1]) {
      return -1;
    }
    int start = 0;
    int end = arr.length - 1;
elseif(target>arr[mid]){
start=mid+1;
}
else{
returnmid;
}

}
return-1;
}
}
```

```
        }
        int start = 0;
        int end = arr.length - 1;

        while(start <= end) {
            // find the middle element
//          int mid = (start + end) / 2; // might be possible that (start +
end) exceeds the range of int in java
            int mid = start + (end - start) / 2;

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

```
packagebinary_search;
```

- FIND THE SMALLEST NUMBER IN THE ARRAY WHICH IS GREATER THAN THE TARGET

```
publicclassSmallest_lettter{
publicstaticvoidmain(String[]args){
intarr[]={0,2,3,4,15,16,18,22,45,89};
inttarget=16;
intans=Binarysearch(arr,target);
System.out.println(ans);


}
staticintBinarysearch(intarr[],inttarget){
intstart=0;
intend=arr.length-1;
while(start<=end){
```

```
packagebinary_search;
```

FIND THE FIRST AND LAST POSITION OF THE GIVEN TARGET VALUE IN SORTED ARRAY

```
publicclassfirstNlastdigit{
publicstaticvoidmain(String[]args){
int[]arr={3,4,4,4,4,7,6,};
inttarget=4;
System.out.println(range(arr,target));

}
staticint[]range(int[]arr,inttarget){
int[]ans={-1,-1};
ans[0]=Searchindex(arr,target,true);
intend=arr.length-1;
while(start<=end){
intmid=start+(end-start)/2;
if(arr[mid]>target){
end=mid-1;
}else{
start=mid+1;
}


}
returnarr[start%arr.length];

}
}
```

```
package com.kunal;

public class Mountain {
    public static void main(String[] args) {

    }
    // https://leetcode.com/problems/peak-index-in-a-mountain-array/
    // https://leetcode.com/problems/find-peak-element/
    public int peakIndexInMountainArray(int[] arr) {
        int start = 0;
        int end = arr.length - 1;

        while (start < end) {
            int mid = start + (end - start) / 2;
            if (arr[mid] > arr[mid+1]) {
                // you are in dec part of array
                // this may be the ans, but look at left
                // this is why end != mid - 1
                end = mid;
            } else {
```

```
int[]ans={-1,-1};
ans[0]=Searchindex(arr,target,true);
ans[1]=Searchindex(arr,target,false);
returnans;
}
staticintSearchindex(intarr[],inttarget,booleanfindstartInde
x){
intstart=0;
intend=arr.length-1;
intans= - 1;
while(start<=end){
intmid=start+(end-start)/2;
if(target>arr[mid]){
Start = mid+1;
}
Else if(arr[mid]>target){
end=mid-1;
}
else{
ans=mid;
if(findstartIndex){
end=mid-1;
}
else{
start=mid+1;
}
}
}
returnans;
}
}
```

```
                // you are in asc part of array
                start = mid + 1; // because we know that mid+1 element > mid element
            }
        }
        // in the end, start == end and pointing to the largest number because of the 2 checks above
        // start and end are always trying to find max element in the above 2 checks
        // hence, when they are pointing to just one element, that is the max one because that is what the
    checks say
        // more elaboration: at every point of time for start and end, they have the best possible answer till
    that time
        // and if we are saying that only one item is remaining, hence cuz of above line that is the best
    possible ans
        return start; // or return end as both are =
      }
    }
```

Q8)Search in rotated sorted array

Bubble sort

12 April 2023 20:15

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

From <https://www.geeksforgeeks.org/bubble-sort/>

Time Complexity: O(N2)

Auxiliary Space: O(1)

Optimized Implementation of Bubble Sort:

The above function always runs **O(N2)** time even if the array is sorted. It can be optimized by stopping the algorithm if the inner loop didn't cause any swap.

From <https://www.geeksforgeeks.org/bubble-sort/>

Arrays

21 March 2023 22:40

- In java we don’t have pointers In array
- # array objects or any other objects are stored in heap memory in java.
- Array is not stored continous in heap memory, beacause in heap memory is not continous.

- #in java the array is not continous.
- #NULL a special value which is by default value of reference variable.

- IN java primitives are stored in stack memory only and the reference variables , classes, objects, and your own varaiables , array type , hash type are stored in heap memory

```
importjava.util.Arrays;
importjava.util.Scanner;
publicclassarray{
publicstaticvoidmain(String[]args){
Scannerinput=newScanner(System.in);
intarr[]=newint[5];
for(inti=0;i<5;i++){
arr[i]=input.nextInt();
}
//for(inti=0;i<arr.length;i++){
//System.out.print(arr[i]+"");
//}
System.out.println(Arrays.toString(arr));

}
}
```

- Method To Declare an array in java and take input of Values.

```
intarr[]=newint[5];
for(inti=0;i<5;i++){
arr[i]=input.nextInt();
}
```

Easy Method to Print an array in java

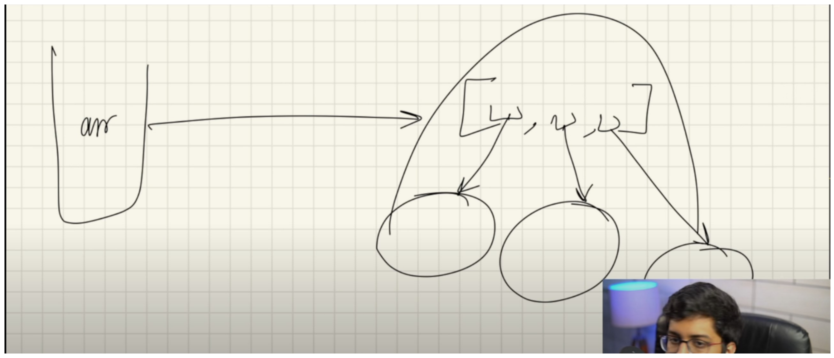
```
System.out.println(Arrays.toString(arr));
```

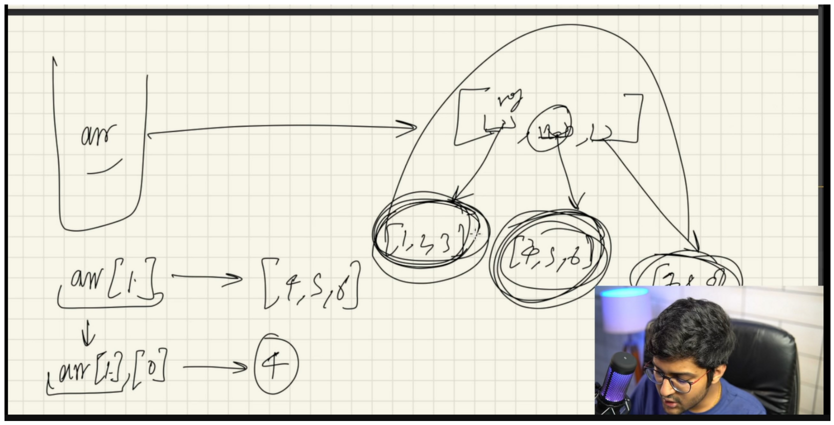
- In this "Arrays" is a class which has feature of "toString" to print the array values ina string form .

To make a String array

```
String[]str=newString[5];
Scannerin=newScanner(System.in);

for(into=0;o<5;o++){                    //just replaced 'i' with 'o'.
str[o]=in.nextLine();
}
System.out.println(Arrays.toString(str));
change(str);
System.out.println(Arrays.toString(str));

}

//making a change function that replace the value of the entered index no .
staticStringchange(String[]str){
str[1]="ab";
returnnull;
}
}
```

THE VALUES IN ARRAY IS STORED LIKE OBJECTS IN HEAP MEMORY





2D ARRAYS

```
importjava.lang.reflect.Array;
importjava.util.Arrays;
importjava.util.Scanner;
publicclassMULarray{
publicstaticvoidmain(String[]args){
Scannerinput=newScanner(System.in);
intok[][]=newint[3][3];
for(introw=0;row<3;row++){
```

If we want to create a array in which we can store any no. Of value orrr which can store a large no. Of values without mentioning the size we use ARRAY LISTS IN JAVA.

```
ArrayList<Integer>list=newArrayList<>(enter
initialising size)
```

```
Scannerinput=newScanner(System.in);
intok[][]=newint[3][3];
for(introw=0;row<3;row++){
for(intcol=0;col<ok.length;col++){
ok[row][col]=input.nextInt();
}
}
for(introw=0;row<3;row++){
System.out.println(Arrays.toString(ok[row]));
}

}
```

Print maximum value of the array.

```
staticint[]max(int[]arr){
intmax=arr[0];
for(inti=1;i<arr.length;i++){
if(arr[i]>max){
max=arr[i];
}
}


returnnewint[]{max};
}
}
```

Q) print the array in reverse order.

```
staticvoidreverse(int[]arr){
intstart=0;
```

```
ArrayList<Integer>list=newArrayList<>(enter
initializing size);

ArrayList<put >


importjava.util.ArrayList;
importjava.util.Arrays;

Public class arraylist{
publicstaticvoidmain(String[]args){
ArrayList<Integer>list  =  newArrayList<>(10);
list.add(5);
list.add(353);
list.add(454);
list.add(54);
list.remove(3);
list.set(2,67);


System.out.println(list);


}
}
```

```
intend=arr.length-1;
while(start<end){
swap(arr,start,end);
start++;
end--;
}
}

staticvoidswap(int[]arr,intindex1,intindex2){
inttemp=arr[index1];
arr[index1]=arr[index2];
arr[index2]=temp;
}
```

Linear Search

24 March 2023 16:27

#Time complexity is how the time grows as the input grows.

➢ Linear Search in an Array

```
Static int linearSearch(int arr[ ] ,int element){
if(arr.length==0){
System.out.println("array is empty");
}
for(inti=0;i<arr.length;i++){
int num = arr[i];
If (num == element){
System.out.println(i);
}

}
return -1;
}
```

➢ Linear Search in a String.

```
staticbooleanstringSearch(String str,char chr){
if(str.length()==0){
returnfalse;
}
for(inti=1;i<4;i++){
if(chr==str.charAt(i)){
System.out.println(i);
returntrue;
}
}
returnfalse;
}
```

➢ Search in 2D arrays

```
importjava.util.Arrays;

publicclasssearchIn2Darr{
publicstaticvoidmain(String[]args){
int[][]arr={{1,2,3,4},
{54,33,23,43},
{45,48,77}
```

➢ Searching for Maximum element in 2D array

```
staticintmax(intarr[][]){
intmax=arr[0][0];
for(introw=0;row<arr.length;row++){
for(intcol=0;col<arr[row].length;col++){
```

```
int[][]arr={{1,2,3,4},
{54,33,23,43},
{45,48,77}
};
inttarget=33;
int[]ans=search(arr,target); //creating a new array ans and calling
the function into it
System.out.println(Arrays.toString(ans));

}

staticint[]search(int[][]arr,inttarget){
for(introw=0;row<arr.length;row++){
for(intcol=0;col<arr[row].length;col++){
if(arr[row][col]==target){
return new int[] {row,col};             // this new int for creating a
new array so we can show the answer in array like [1,1].

}
}
}
return new  int[]{-1,-1};
}
}
```

Q) Given an array nums of integers, return how many
of them contain even number of digit.

```
publicclassEvenDigitcount{
publicstaticvoidmain(String[]args){
int[]nums={12,345,2,6,7896};
System.out.println(search(nums));
```

```
for(introw=0;row<arr.length;row++){
for(intcol=0;col<arr[row].length;col++){
if(max<arr[row][col]){
max=arr[row][col];
}

}
}
returnmax;
}
}
```

```
}
staticintsearch(int[]arr){
intcount=0;
for(intnum:arr){
if(even(num)){
count++;       }
}
returncount;
}
staticbooleaneven(intnum){
intvalue=digits(num);
if(value%2==0){
returntrue;
}
returnfalse;
}
staticintdigits(intnum){
intcount=0;
while(num>0){
count++;
num=num/10;

}
returncount;
}
}
```