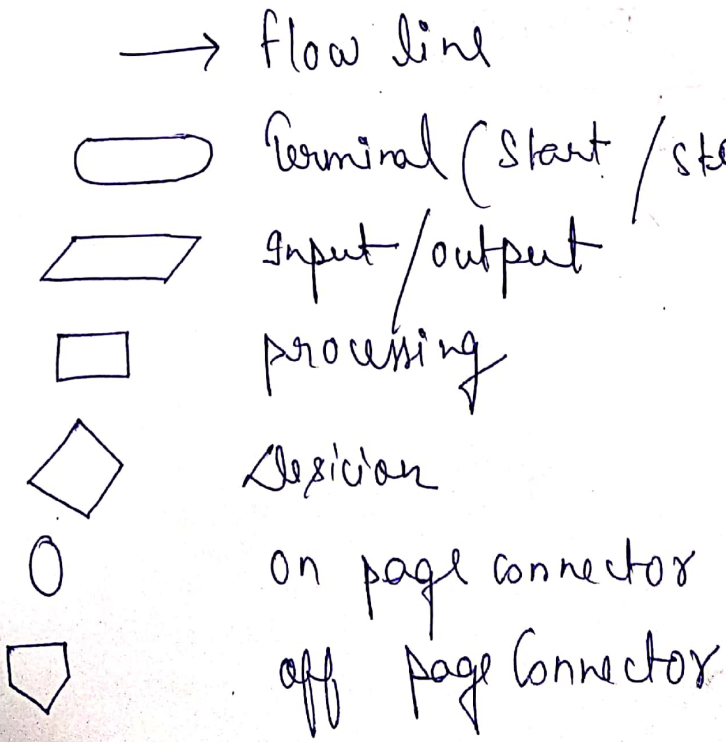
D.S.A.

What is Algorithm?

* Step by step procedure deigned to perform an operation, and which (like a map or flowchart) will lead to the sought result if followed correctly
* Algorithms have a definite beginning and a definite end.
* A finite number of steps.
* An algorithm produces the same output information given the same input information, and several short algorithms can be combined to perform complex tasks such as writing a computer program.



Write an algorithm and draw a flow chart to find whether the number is even or odd.

Step 1 :- Start

Step 2 : Input N

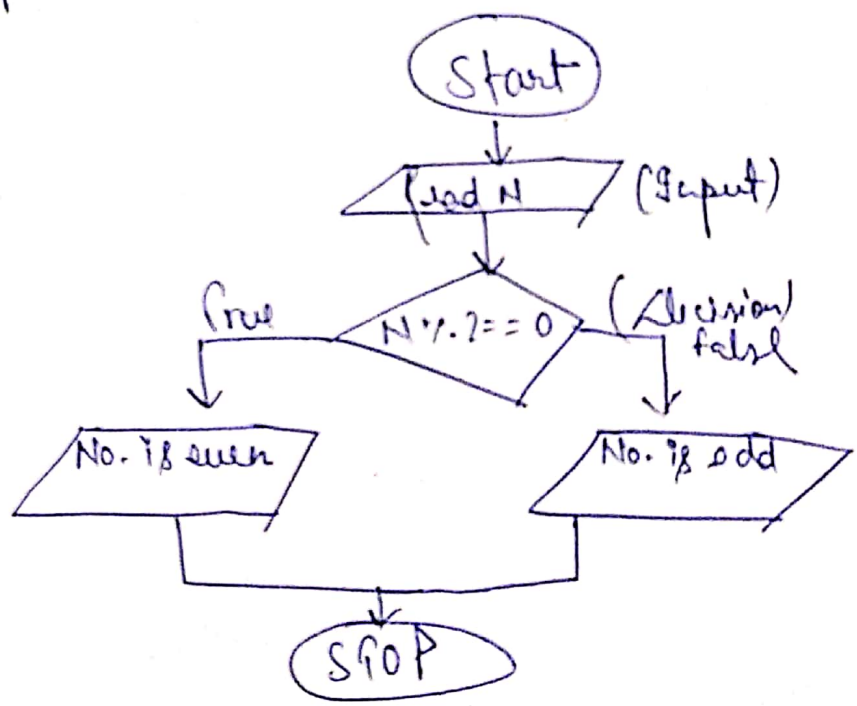
Step 3 - if (N % 2 == 0) then

print "number is even.

else

print "number is odd.

Step 4 - Stop



Data: Anything to give information is called Data.

Ex- Student Name, student Rollno.

Structure – Representation of Data is called structure.

Ex- graph, Arrays, list

Data Structure :-

Data Structure = Data + Structure

Data structure is a way to (Store) (Manage) and Organize data so that it can be used efficiently (better way).

Ex- Student

Name, Rollno. Branch,

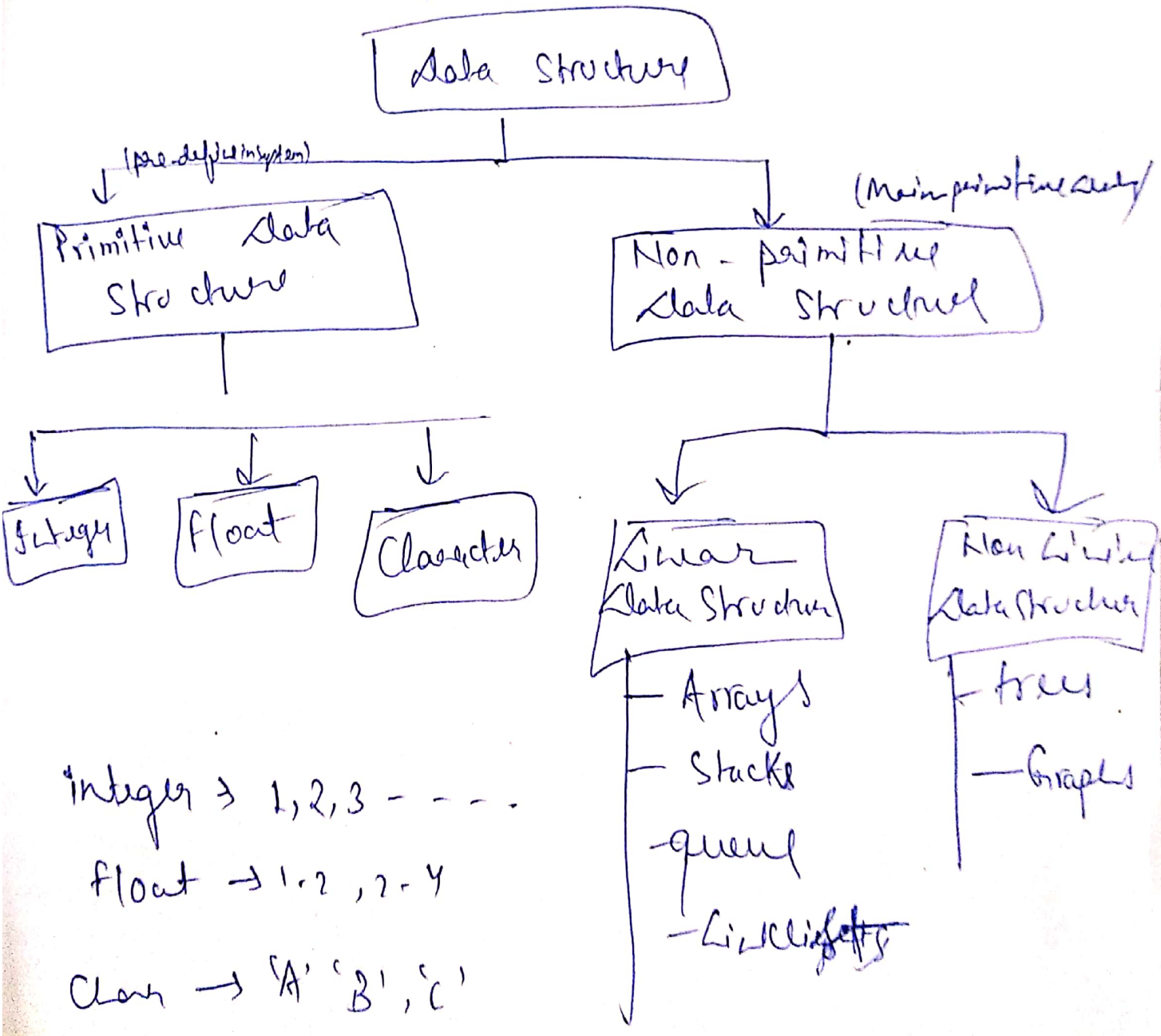
Name	Roll no.	Branch
Kunal	2	IT

(Table)

* Data Structure is a way of Organizing all data items and Relationship to each other.

Types of Data Structures

There are mainly two types of Data Structure.



integer → 1, 2, 3 - - - -

float → 1.2 , 2.4

Char → 'A' 'B', 'C'

Primitive Data Structure =

These are basic structure and are directly operated by machine instruction.

Ex- Integer, float, character.

Non- primitive data structure :-

These are derived from the primitive data structure, it is a collection of same type or different type primitive data structure

Ex- Arrays, Stacks, Linked list, trees

What is Data Structure?

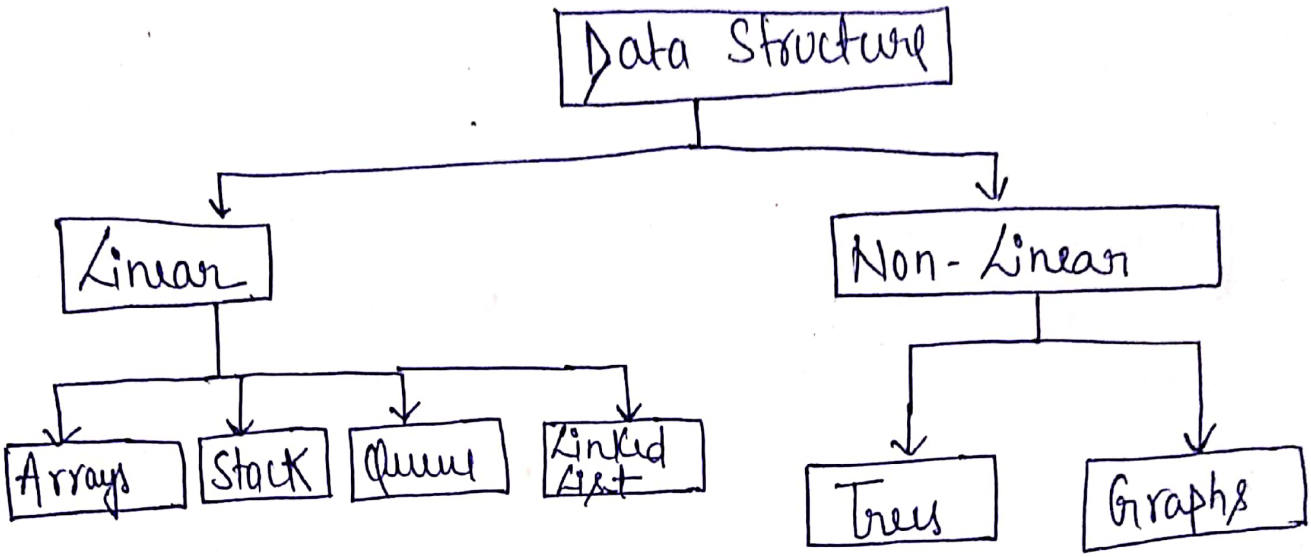
* Data structure are the programmatic way of storing data so that data can be used efficiently.

Interface and Implementation

Interface – Interface represents the set of Operations that a data structure supports. An Interface only provides the list of Supported Operations, types of parameters they can accept and return type of these operations.

Implementation – Implementation provides the Internal Representation of Data structure, Implementation also provides the definition of the algorithms used in the Operations of the Data Structures.

Data Structure is classified into two categories:-



1). Linear Data Structure:-

Data structure where data elements are arranged sequentially or linearly where the elements are attached to its previous and next adjacent in what is called a Linear data structure.

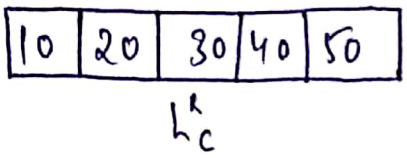
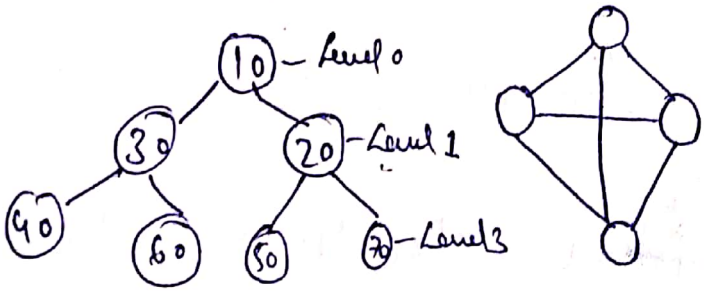
E.g.:- Array, stack, queue, Linked List, etc.

Operation on Linear

- Add an element
- Delete an element
- Traverse / Display
- Sort the List of elements
- Search for a data elements.

Data Structure

Difference between Linear and Non-Linear Data Structure.

Linear	Non-Linear
1)- Every item is related to its previous and next item.	Every item is attached with many other items.
2)- Data is arranged in Linear sequence. 	Data is not arranged in sequence. 
3)- Data items can be traversed in a single run.	Data cannot be traversed in a single run
4)- Application Software Development.	Artificial Intelligence, Image processing.

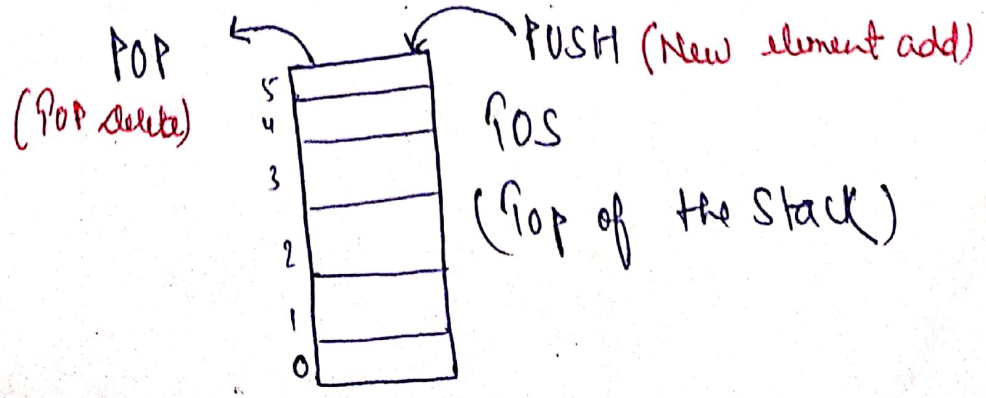
Types of Linear Data Structure

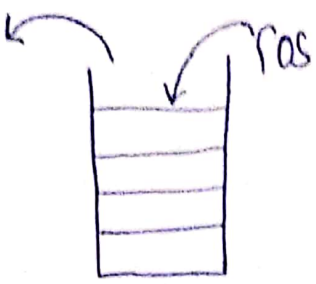
- **Array :-** An array is the collection of the Variables of the same data type that are referenced by the common name.

Arrays

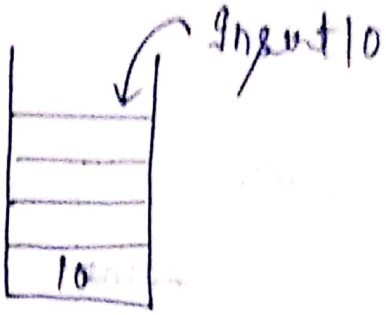
```
int. a[10] = {1,2,3,4 ---- 10}
     a[0] = 1;
     a[1] = 2;
      :
     a[9] = 10;
```

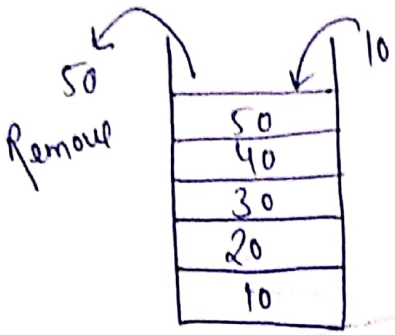
- **Stack:** A Stack is a LIFO (Last in first Out) data structure where element that added last will be deleted first. All operations on a stack are performed from on end called TOP. A TOP is a pointer which points to the top element - the element which entered last in the stack.

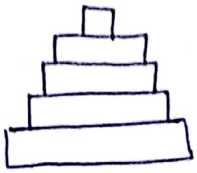




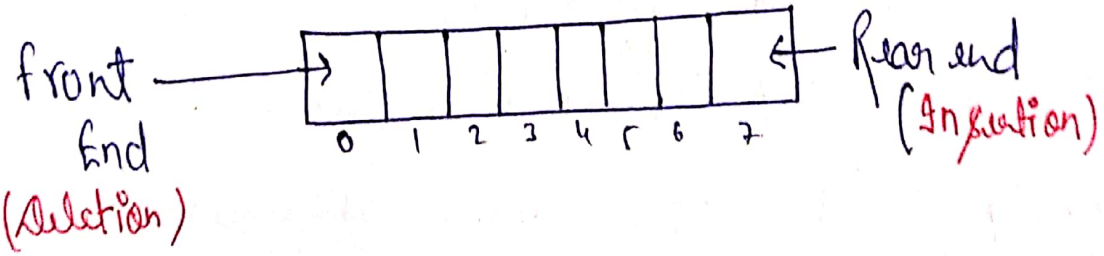
Eg. 10

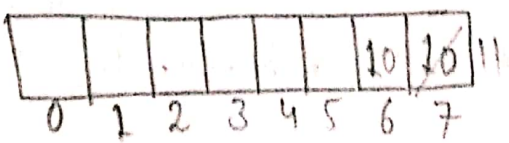




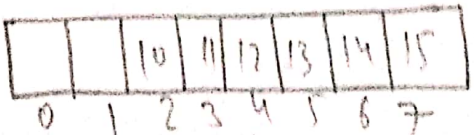


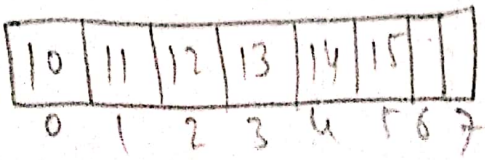
- **Queue :-** A queue is a FIFO (first in first out) data structure where element that added first will be deleted first. In queue, Insertion is performed from one end called REAR and deletion is performed from another end called FRONT.





10, 11, 12, 13, 14, 15



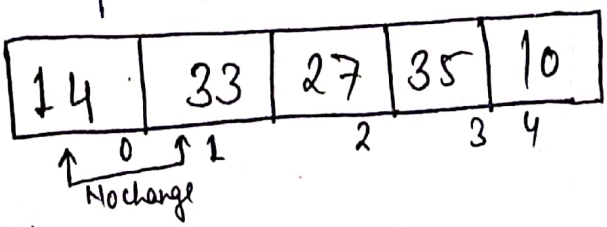


Bubble Sort

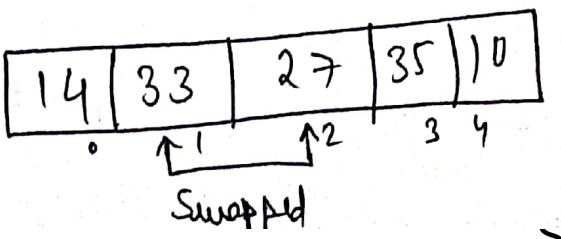
Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and Worst Case Complexity are of $O(n^2)$ where n is the number of items.

How Bubble sort works?

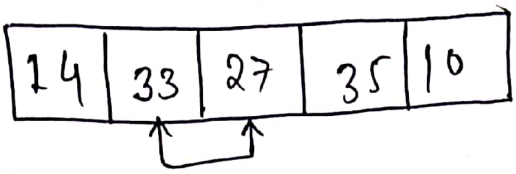
We take an unsorted array for our example. Bubble sort takes $O(n^2)$ time so we're keeping it short and precise.



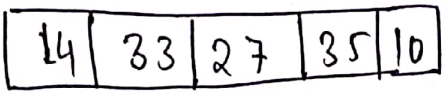
Bubble sort starts with very first two elements, comparing them to check which one is greater.

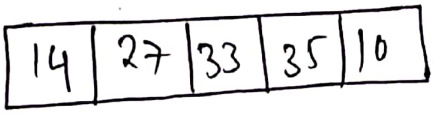


In this Case, Value 33 is greater than 14, so it is already in sorted locations. Next we Compare 33 with 27.

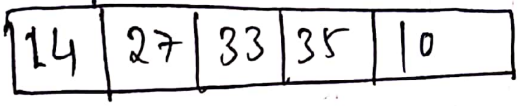


We find that 27 is smaller than 33 and these two values must be swapped.

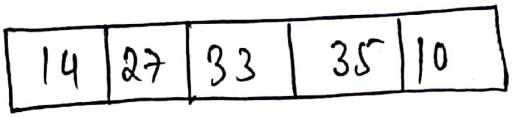




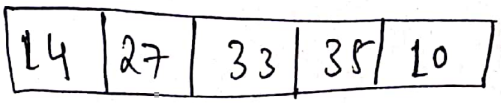
The new array should look like this.



Next we Compare 33 and 35. we find that both are in already sorted positions.



Then we move to the next two values 35 and 10.



We know then that 10 is smaller 35. Hence they are not sorted.

14	27	33	35	10

We swap these values. we find that we have reached the end of the array. after one iteration, the array should look like this -

14	27	33	10	35

To, be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this -

14	27	10	33	35

Notice that after each iteration, at least one value moves at the end.

14	10	27	33	35

And we there's no swaps required, bubble sorts learns that an array is Completely sorted.

10	14	27	33	35

Now, we should look into some practical aspects of bubble sort.

SELECTION SORT

Selection sort is a simple sorting algorithm. This sorting algorithm is an in place comparison based algorithm in which the list is divided into parts, the sorted part at the left end and and unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where 'n' is the number of items.

How Selection sort works?

Eg:1.

42	16	84	12	77	26	53

Eg:2. Consider the following depicted array as an example.

14	33	27	10	35	19	42	44

for the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

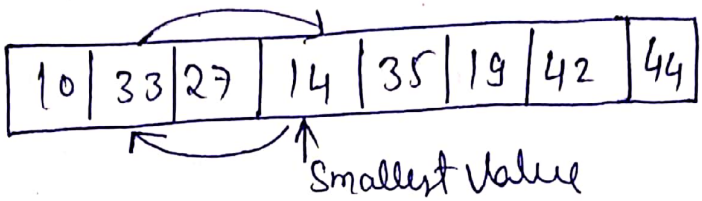
14	33	27	10	35	19	42	44

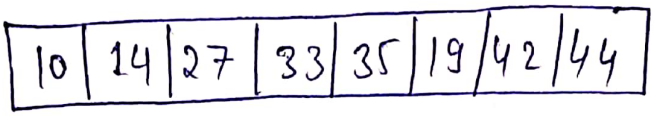
So we replace 14 with 10. After one iteration 10, which happens to be the minimum value is the list, appears in the first position of the sorted List.

10	33	27	14	35	19	42	44

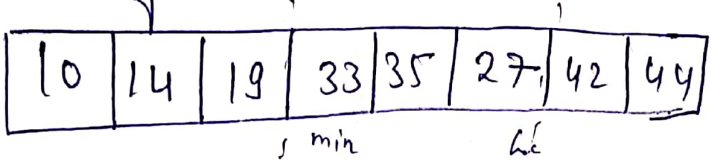
for the second position, where 33 is residing, we start scanning the rest of the list in a linear manner.

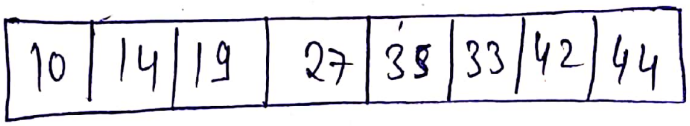
We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.

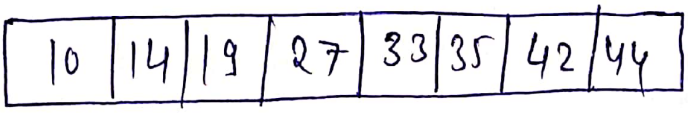


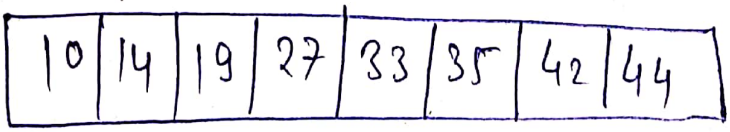


The same process is applied to the rest of the items in the array.









Now, lets get a sorted List in Selection sort.

Insertion Sort

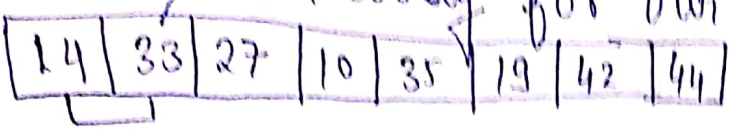
This is an in place comparison based sorting algorithm. Here a sub list is maintained which is always sorted.

For example:- The lower part of an array is maintained to be sorted. An element which is to be 'inserted' in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, Insertion sort.

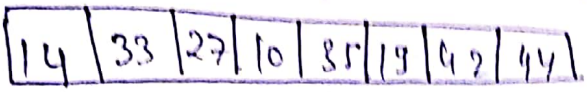
The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are $O(n^2)$ where 'n' is the number of items.

How Insertion sort works

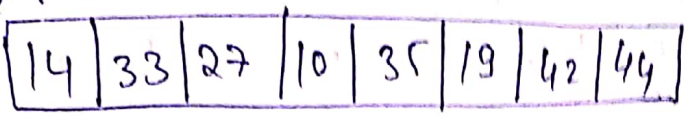
We take an unsorted array for our examples.



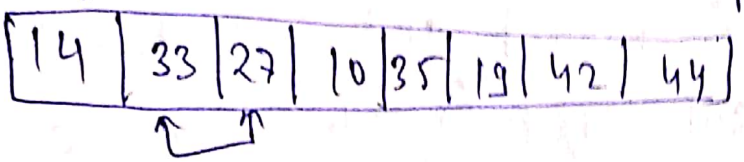
Insertion sort Compares the first two elements.



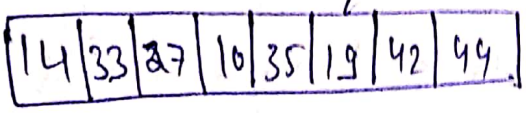
it finds that both 14 and 33 are already in ascending order. for Now, 14 is in sorted sub-list.

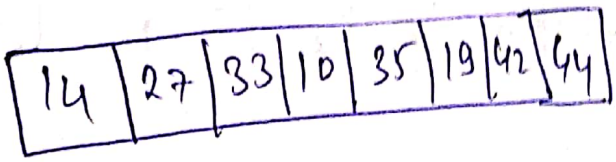


Insertion sort moves ahead and Compares 33 with 27.



And finds that 33 is not in the correct position





it swaps 33 with 27. it also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14

and 27 greater than 14, Hence, the sorted sub-list remains sorted after swapping.

14	27	33	10	35	19	42	44

By now we have 14 and 27 in the sorted sub-list. Next, it Compares 33 with 10.

14	27	33	10	35	19	42	44

These values are not in sorted order.

So we swap them

14	27	10	33	35	19	42	44

However swapping makes 27 and 10 unsorted

Swap them too

14	10	27	33	35	19	42	44

Again we find 14 and 10 in an unsorted order.

14	10	27	33	35	19	42	44

10	14	27	33	35	19	42	44

This process goes on until all the unsorted values are covered in a sorted sub-list. Now we shall see some programming aspects of Insertion sort.