6.	A text T and patterns P and Q are in memory. Write an algorithm that replace every occurrence of P in the text T by Q.	2008,2011,2016

Answer:

The algorithm that replace every occurrence of P in the text T by Q

1. [find index of P] set K=INDEX(T,P)
2. Repeat while K≠0
   a) [replace P by Q] set T=REPLACE[T,P,Q]
   b) [update index] set K=INDEX(T,P)
3. Write T
4. Exit

7.	Let S='His Father is the Professor ' Find out the result of the following operations i) Substring (S,11,5) ii) Index (S,'ESS') iii) Delete (S,14,4) iv) Replace (S,'IS','ER')	2014

Answer: S='His Father is the Professor '

The result of the following operations

i) Substring (S,11,5)= " is t"
ii) Index (S,'ESS')= 23
iii) Delete (S,14,4)= 'His Father is Professor '
iv) Replace (S,'IS','ER')= His Father er the Professor'

8.	Let W be the string. W "abcdebecaec". I)INSERT (W, 2,"pgrs"): (i) REPLACE(W," "rsb", "mnbc").	2008,

Answer: W ="abcdebecaec".

I) INSERT (W, 2,"pgrs")= "apgrsbcdebecaec".
II) REPLACE(W, "rsb", "mnbc")= "apgmnbccdebecaec".

9.	Given, string operation DELETE ('vxtyrwq',3,3). Show the result of the operation using string operation 'SUBSRING'	2009

Answer:

DELETE(T,K,L)= SUBSRING(T,1,K-1)// SUBSRING(T,K+L, Length(T)-K-L+1)

Here, DELETE(T,K,L)=DELETE ('vxtyrwq',3,3)= 'vxwq'

SUBSRING(T,1,K-1)= SUBSRING('vxtyrwq',1,2)='vx'

SUBSRING(T,K+L, Length(T)-K-L+1)= SUBSRING('vxtyrwq',6, 2)='wq'

So, SUBSRING(T,1,K-1)// SUBSRING(T,K+L, Length(T)-K-L+1)='vxwq'

10.	Given, string operation INSERT ('prtyuwe',4,'yu'). Show the result of the operation using string operation 'SUBSRING'	2009

Answer:

INSERT (T,K,S)= SUBSRING(T,1,K-1)//S//SUBSRING(T,K, Length(T)-K+1)

Here,INSERT (T,K,S)=INSERT ('prtyuwe',4,'yu')= 'prtyuyuwe'

SUBSRING(T,1,K-1)= SUBSRING('prtyuwe',1,3)='prt'

S='yu'

SUBSRING(T,K, Length(T)-K+1)=SUBSRING('prtyuwe',4, 4)='yuwe'

So, SUBSRING(T,1,K-1)// S//SUBSRING(T,K, Length(T)-K+1)= 'prtyuyuwe'

11. Explain Variable and data type.

Each variable in any of our algorithms or programs has a death type which determines the code that is used for storing its value. For such data types follow:

1. Character. Hare data are coded using some character code such as EBCDIC or ASCII. The 8-bit EBCDIC code of some characters appears in figure 2.7.A single character is normally stored in a byte.

2. Real (or floating point).hare numerical data are coded using the exceptional form of the data.

3. Integer (or fixed point). Hare positive integers are coded using binary representat ion, and negative integers by some binary variation such as 2's complement.

4.Logical. here the variable can have only the value true or false: hence it may be coded using only one bit.1 for true and 0 for false.(sometimes the bytes 1111 1111 and 0000 000 0 may be used for true and false, respectively.

The data types of variables in our algorithms wheel not be explicitly stated as with computer programs but will usually be implied by the context.

Char.	Zone	Numeric	Hex	Char.	Zone	Numeric	Hex	Char.	Zone	Numeric	Hex
A	1100	0001	C1	S	1110	0010	E2	*blank*	0100	0000	40
B	↓	0010	C2	T	↓	0011	E3	.	↓	1011	4B
C	↓	0011	C3	U	↓	0100	E4	<	↓	1100	4C
D	↓	0100	C4	V	↓	0101	E5	(	↓	1101	4D
E	↓	0101	C5	W	↓	0110	E6	+	0100	1110	4E
F	↓	0110	C6	X	↓	0111	E7	&	0101	0000	50
G	↓	0111	C7	Y	↓	1000	E8	$	↓	1011	5B
H	↓	1000	C8	Z	1110	1001	E9	*	↓	1100	5C
I	1100	1001	C9					)	↓	1101	5D
J	1101	0001	D1	0	1111	0000	F0	;	0101	1110	5E
K	↓	0010	D2	1	↓	0001	F1	—	0110	0000	60
L	↓	0011	D3	2	↓	0010	F2	/	↓	0001	61
M	↓	0100	D4	3	↓	0011	F3	,	↓	1011	6B
N	↓	0101	D5	4	↓	0100	F4	%	↓	1100	6C
O	↓	0110	D6	5	↓	0101	F5	>	↓	1110	6E
P	↓	0111	D7	6	↓	0110	F6	?	0110	1111	6F
Q	↓	1000	D8	7	↓	0111	F7	:	0111	1010	7A
R	1101	1001	D9	8	↓	1000	F8	#	↓	1011	7B
				9	1111	1001	F9	@	↓	1100	7C
								=	0111	1110	7E

FIG: part of the EBCDIC code

12.	**Explain first pattern matching algorithm with example.**	**2009,2014**

Answer:

(Pattern matching) P and T are strings with length R and S. Respectively and are stored as Eris with one character per element. This algorithm find the INDEX of P in T.

1.[Initialize] Set K := 1 and MAX : = S–R+1.
2.Repeat Steps 3 to 5 while K £MAX:
3.Repeat for L=1 to R : [Test each character of P.]
If P [L]'T[K+L-1], then : Go to step 5.
[End of inner loop]
4. [Success.] Set INDEX = K, and Exit .
5. Set K:=K+1.
[End of setup 2 outer loop.]
6.[Failure.] Set INDEX= 0.
7.Exit

Example

Suppose P=aaba and T =cdcd ....cd=(cd)¹.Clearly P does not occur in T.Also, for each of the 17 cycles,Nk=1, since the first character of P does not match Wk, Hence

C=1+1+1+ ...+1=17

Suppose P= abba and T = ababaaba.... Observe that P is a substring of T.In fact , P=W and so N5=4,also, comparing P with W1 = abab,we see that N1=2, since the first letters do match; but comparing P with W2=baba,we set that N2=1, since the first lette r do not match. Similarly,N3=2 and N4=1 Accordingly,

C=2+1+2+1+4=10

CHAPTER 3
ARRAYS, RECORDS AND POINTERS

1.	Define array	2008,2010,

Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms. Following are the important terms to understand the concept of Array.

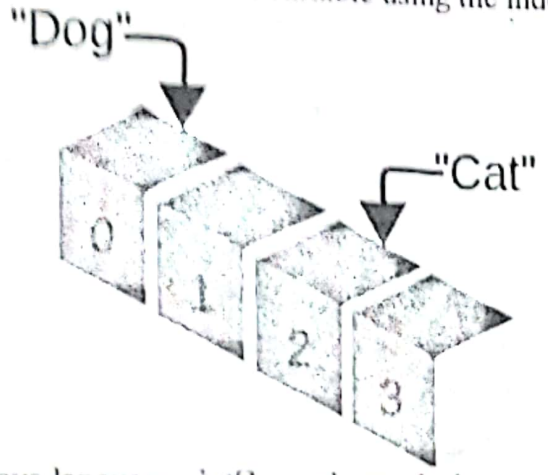
Element – Each item stored in an array is called an element.

Index – Each location of an element in an array has a numerical index, which is used to identify the element.

2.	What do you mean by linear array and two-dimensional array?	2012

What is 1D Array

1D array or **single dimensional array** stores a list of variables of the same data type. It is possible to access each variable using the index.



In Java language, int[] numbers; declares an array called numbers. Then, we can allocate memory for that array using 'new' keyword as follows.

numbers= new int[10];

This array is capable of storing 10 integer values.

We can combine the above two statements together and write as follows.

int numbers = new int[10];

Below is an example of assigning values to the array.

numbers ={1,2,3,4,5,6,7,8,9,10};

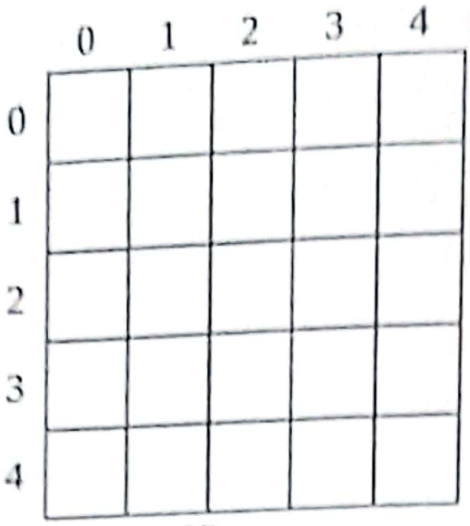
The starting index of an array is 0. Therefore, the element in the 0^{th} index is 1. The element in the 1^{st}index is 2. The element in the 2^{nd} index is 3, etc. The index of the final element is 9.

If the programmer wants to store number 50 on the 2^{nd} index, he can write the statement as follows.

numbers[2] = 50;

What is 2D Array

2D array or **multi-dimensional array** stores data in a format consisting of rows and columns.

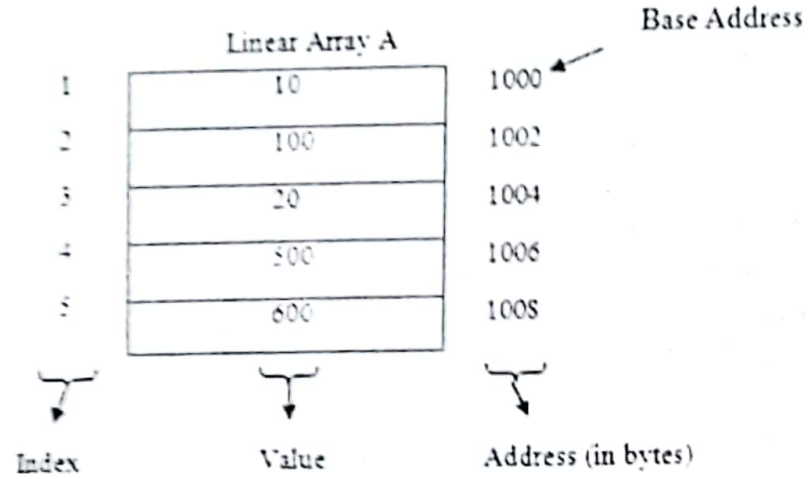


For example, int[][] numbers; declares a 2D arrays.
numbers = new int [2][3];
The above statement allocates memory for a 2D array of 2 rows and 3 columns.
We can combine the above two statements together and write the statement as follows.
int[][] numbers = new int[2][3];
Below is an example of assigning values to the 2D array.
int[][] numbers = { {10,20,30}, {50,60,70}};
Similar to a 1D array, the starting index of the 2D array is also 0. This array has two rows and three columns. The indexes of the rows are 0 and 1 while the indexes of columns are 0, 1 and 2. The element 10 is in the 0^{th} row 0^{th} column position. Number 20 is in the 0^{th} row, 1^{st} column position. Number 70 is in 1^{st} row, 2^{nd} column position.
numbers[1][2] = 50;
Above statement assigns number 50 to 1^{st} row, 2^{nd} column position.

3.	How Linear arrays are represented in memory?	2008,2010, 2012,2013

Represent a Linear Array in memory

T he elements of linear array are stored in consecutive memory locations. It is shown below:



Arrays are allocated as a contiguous block of memory divided into one or more elements of equal size and type. Knowing the start address of the array makes it possible to reference any element within the array through a zero-based index. Multiplying the index by the size of an element determines the offset from the start of the array.

4.	Define record	

A record (also called a structure, struct, or compound data) is a basic data structure. A record is a collection of fields, possibly of different data types, typically in fixed number and sequence. The fields of a record may also be called members, particularly in object-oriented programming.

5.	Difference between Array and record	2012, 2016

Array	Record
1. Array is A data structure that stores a list of values of the same data type.	1. Record is A data structure that stores bits of data of multiple data types in fields.
2. It is possible to input and output values or process an entire array using a For Next Loop. Each element has an index to specify its position.	2. It is possible to input or output data into the fields of an entity to create a record using record sets.
3. An array can be used to store data of the same data type.	3. A record can be used to store a collection of data of different data types.
4. Arrays have locations and these are numbered.	4. A records have fields and these are named.
5. Every location in an array is of the same data type	5. Every field in a record can be of different data types.

6.	Define pointer	

A pointer is a programming language object, whose value refers to (or "points to") another value stored elsewhere in the computer memory using its memory address. A pointer references a location in memory, and obtaining the value stored at that location is known as dereferencing the pointer.

7.	What is searching?	2008

Searching is a process of locating a particular element present in a given set of elements. The element may be a record, a table, or a file.

A search algorithm is an algorithm that accepts an argument 'a' and tries to find an element whose value is 'a'. It is possible that the search for a particular element in a set is unsuccessful if that element does not exist. There are number of techniques available for searching.

8.	Define linear search	

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.

9.	Write the algorithm of linear search in an array	2009,

Algorithm

Linear Search (Array A, Value x)
Step 1: Set i to 1
Step 2: if i > n then go to step 7
Step 3: if A[i] = x then go to step 6
Step 4: Set i to i + 1
Step 5: Go to Step 2
Step 6: Print Element x Found at index i and go to step 8
Step 7: Print element not found
Step 8: Exit

10.	Write the linear search algorithm for finding largest elements in the array.	2012

Algorithm (Largest element in Array)

1. K:=1, LOC:=1, MAX:=DATA[1]
2. K=K+1.
3. IF (K>N), then :

Write: LOC,MAX, and Exit.

4. IF MAX < DATA [K], then

LOC:= K and MAX:= DATA [k]

5. Go to step 2.

11. What is linear array? Write an algorithm to insert an element into a linear array. [2020]

Linear array: A linear array, is a list of finite numbers of elements stored in the memory. In a linear array, we can store only homogeneous data elements. Elements of the array form a sequence or linear list, that can have the same type of data. Each element of the array, is referred by an index set.

Algorithm:

State Algorithm for Inserting an Element in a linear Array-

1. [Initialize Counter] Set J: = N.
2. Repeat steps 3 and 4 while J ≥ K.
3. [Move Jth element downward] Set LA [J+1]: = LA [J]
4. [Decrease Counter] Set J: = J-1. [End of step 2 loop]
5. [Insert element] Set LA [K]: = ITEM.
6. [Reset N] Set N: = N+1.
7. Exit.

12.	Define binary search	2011,

Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item.

Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the sub array reduces to zero

13.	Write an algorithm for binary search	2008,2009,2013

Algorithm

As it is an improvisation of the existing BST algorithm, we are mentioning the steps to search the 'target' data value index, using position probing –

Step 1 – Start searching data from middle of the list.
Step 2 – If it is a match, return the index of the item, and exit.
Step 3 – If it is not a match, probe position.
Step 4 – Divide the list using probing formula and find the new midle.
Step 5 – If data is greater than middle, search in higher sub-list.
Step 6 – If data is smaller than middle, search in lower sub-list.
Step 7 – Repeat until match.

14.	What is the complexity of binary search?	2008,2011

The complexity is measured by the number $f(n)$ of comparisons to locate ITEM in DATA where DATA contains n elements. Observe that each comparison reduces the sample size in half. Hence we require at most $f(n)$ comparisons to locate ITEM where

$$2^{f(n)} > n \quad \text{or equivalently} \quad f(n) = \lfloor \log_2 n \rfloor + 1$$

That is, the running time for the worst case is approximately equal to $\log_2 n$. One can also show that the running time for the average case is approximately equal to the running time for the worst case.

15.	What are the limitation of binary search.	2011

The major limitation of binary search is that there is a need of sorted array to perform binary search operation.

If array is not sorted the output is either not correct or may be after a long number of steps and according to data structure the output should come in minimum number of steps.

SO the limitation are:

1. Array need to be sorted
2. Can not apply on linked list (elements need to be contiguous in memory)
3. Cache performance is also not good (because we are accessing n/2th position of array every time).

16.	Write down the algorithms for Inserting and Deleting Data from a Linear array.	2011

(Inserting into a Linear Array) INSERT(LA,N,K,ITEM)

Here LA is a linear array with N elements and K is a positive integer such that $K \leq N$. This algorithm inserts an element ITEM into the Kth position in LA.

1. [Initialize counter.]Set J:=N.
2. Repeat Steps 3 and 4 while J≥ K.
3. [Move Jth element downward.] Set LA [J + 1]:=LA[J].
4. [Decrease counter.] Set J:=J-1.

   [End of Step 2 loop.]
5. [Insert element.] Set LA[K]:=ITEM.

6. [Reset N.]Set N:=N+1.
7. Exit.

(Deleting from a Linear Array)DELETE(LA,N.K.ITEM)
Here LA is a linear array with N elements and K is a positive integer such that K≤ N. This algorithm deletes the Kth element from LA.
1. Set ITEM:=LA[K].
2. Repeat for J=K to N-1:
   [Move J + 1st element upward.] SetLA[J]:=LA[J+1].
   [End of loop.]
3. [Reset the number N of elements in LA.] Set N:=N-1.
4. Exit.

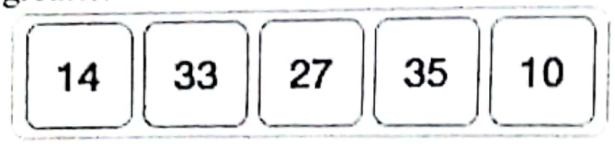
17.	**Write down the algorithm to delete an item from an array.**	**2010**

(Deleting from a Linear Array) DELETE(LA, N.K,ITEM)
Here LA is a linear array with N elements and K is a positive integer such that K≤N. This algorithm deletes the Kth element from LA.
1. Set ITEM:=LA[K].
2. Repeat for J=Kto N-1:
   [Move J+ 1st element upward.] Set LA[J] :=LA[J+1].
   [End of loop.]
3. [Reset the number N of elements in LA.] Set N:=N-1.
4.Exit.

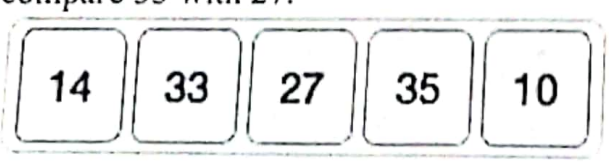
18.	**Suppose the following numbers are stored in an array: A= (14,33,27,35,10). Using bubble sort algorithm ,sort these numbers.**	

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.

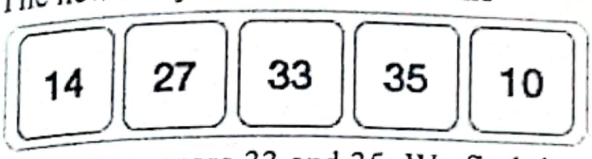


In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

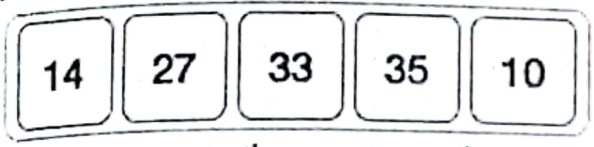


We find that 27 is smaller than 33 and these two values must be swapped.

The new array should look like this –

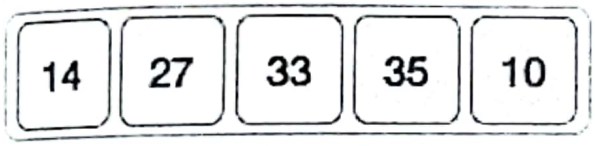


Next we compare 33 and 35. We find that both are in already sorted positions.

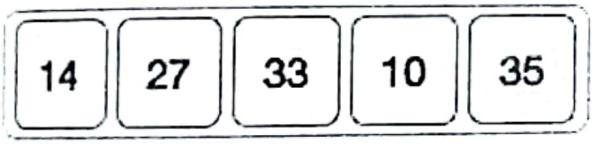


Then we move to the next two values, 35 and 10.

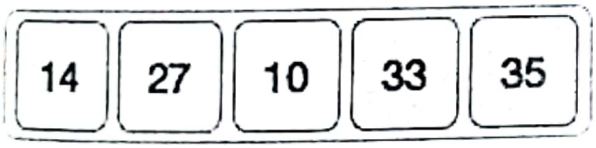
We know then that 10 is smaller 35. Hence they are not sorted.



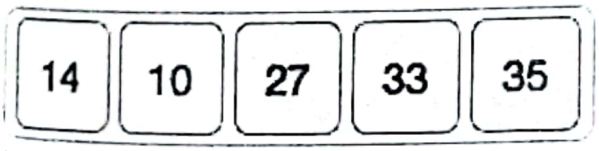
We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this –



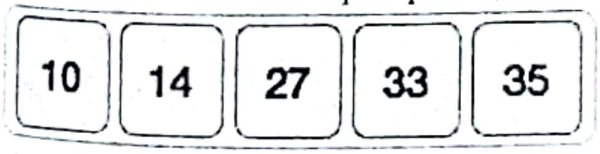
To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this –



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



And when there's no swap required, bubble sorts learns that an array is completely sorted.



19.	**Suppose the following numbers are stored in an array: DATA= (5, 2, 15, 8, 33, 26, 6). Using bubble sort algorithm, sort DATA in ascending order.**	2009

Same way as previous one

20.	**Suppose the following numbers are stored in an array: DATA= (5, 2, 15, 8, 33, 26, 6). Using bubble sort algorithm, sort DATA in ascending order.**	2009

Same way as previous one

21.	**Suppose the following numbers are stored in an array: DATA= (7, 18, 25, 2, 6, 12, 9). Using bubble sort algorithm, sort DATA in descending order.**	2010

Same way as previous one

22.	**Suppose the following numbers are stored in an array: A= (23, 43, 17, 95, 76, 21, 14, 57). Using bubble sort algorithm, sort these numbers step by step in ascending order.**	2013

Same way as previous one

23.	**Difference between linear and binary search**	2007,2014,2016

LINEAR SEARCH	BINARY SEARCH
1. Time Complexity O(N)	1. Time Complexity $O(\log_2 N)$
2. Best case time First Element O(1)	2. Best case time Center Element O(1)
3. Prerequisite for an array No required	3. Array must be in sorted order
4. N comparisons are required for worst case	4. Can conclude after only $\log_2 n$ comparisons
5. Can be implemented on Array and Linked list	5. Cannot be directly implemented on linked list
6. Easily inserted at the end of list	6. Require processing to insert at its proper place to maintain a sorted list.
7. Algorithm type is Iterative in nature	7. Divide and conquer in nature
8. Easy to use and no need for any ordered elements	8. Somehow tricky algorithm and elements must be arranged in order
9. Lines of Code Less	9. Lines of Code More

24.	**Write down the algorithm of bubble sort.**	2010,2012,2014

Algorithm

We assume **list** is an array of **n** elements. We further assume that **swap**function swaps the values of the given array elements.

```
beginBubbleSort(list)

for all elements of list
if list[i]> list[i+1]
      swap(list[i], list[i+1])
endif
endfor

return list
endBubbleSort
```

```
procedure bubbleSort( list : array of items )

   loop = list.count;

for i =0 to loop-1do:
     swapped =false

for j =0 to loop-1do:

/* compare the adjacent elements */
if list[j]> list[j+1]then
/* swap them */
          swap( list[j], list[j+1])
          swapped =true
endif

endfor

/*if no number was swapped that means
     array is sorted now, break the loop.*/

if(not swapped)then
break
endif

endfor

end procedure return list
```

25.	Find out the complexity of bubble sort algorithm	2014

Complexity Analysis of Bubble Sort

In Bubble Sort, n-1 comparisons will be done in the 1st pass, n-2 in 2nd pass, n-3 in 3rd pass and so on. So the total number of comparisons will be,

$(n-1) + (n-2) + (n-3) + ... + 3 + 2 + 1$

Sum = $n(n-1)/2$

i.e $O(n^2)$

Hence the **time complexity** of Bubble Sort is $\mathbf{O(n^2)}$.

The main advantage of Bubble Sort is the simplicity of the algorithm.

The **space complexity** for Bubble Sort is **O(1)**, because only a single additional memory space is required i.e. for temp variable.

Also, the **best case time complexity** will be **O(n)**, it is when the list is already sorted.

Following are the Time and Space complexity for the Bubble Sort algorithm.

- ✓ Worst Case Time Complexity [Big-O]: $\mathbf{O(n^2)}$
- ✓ Best Case Time Complexity [Big-omega]: $\mathbf{O(n)}$
- ✓ Average Time Complexity [Big-theta]: $\mathbf{O(n^2)}$
- ✓ Space Complexity: $\mathbf{O(1)}$

25.	**Apply the binary search algorithm for searching a data item 85 from the following data array:-** **Data: 11,22,30,44,50,55,65,70,72,82,88,90**	**2014**

Suppose ITEM=85. The binary search for ITEM is pictured in Fig. 4.8. Here BEG,END and MID will have the following successive values:

1. Again initially, BEG = 1, END = 13, MID-7 and DATA[MID] -55.
2. Since 85>55, BEG has its value changed by BEG-MID+1=8. Hence
   MID =INT[(8+13)/2] =10 and so DATA[MID]=77
3. Since 85>77, BEG has its value changed by BEG=MID+1-11.Hence
   MID=INT[(11+13)/2]=12 and so DATA[MID]=88
4. Since 85 <88, END has its value changed by END=MID-1=11.Hence
   MID=INT[(11 +11)/2] =11 and so DATA[MID]=80
   (Observe that now BEG=END=MID=11.)

Since 85 >80, BEG has its value changed by BEG =MID +1 =12.But now BEG>END. Hence ITEM does not belong to DATA.

(1) ⑪ 22,30,33,40,44,55,60,66,77,80,88, 99

(2) 11, 22.30.33, 40, 44, 55, 60 66, 77, 80, 88, 99

(3) 11, 22, 30.33, 40, 44, 55,60,66,77. 80 . 88. 99

(4) 11,22,30,33,40,44,55,60,66,77, 80 88 .99 [Unsuccessful]