Data structure

When we want to solve any problem, for every problem there may be multiple ways of solving it.

So we need to find out all different approaches, and find the optimum solution. Is called as problem solving technique.

One of the way of finding multiple solutions is brain storming, group discussion, etc

In group discussion usually different people think about different aspects of the problem, Like one will think about look and feel, one will think about storage structure, one may think about performance, one may think about space requirement, and all together will find the optimum solution.

One of the famous technique people use is called as six hat thinking technique

Question--- find 6 colours of hats and the speciality of each colour hat.

Time Complexity

It is defined as time required by algorithm. And if input size changes then in what proportion it changes

Common time complexity values are

$O(n), O(n^2), O(n^3), O(\log n), o(n\log n)$

For every algorithm we may calculate 3 different time complexity

1. Best ------ it is represented by symbol Omega(Ω)
2. Average----It is represented by symbol Theta(θ)
3. Worst-----It is represented by symbol big-O (O)

To find time complexity

For(int i=0;i<n;i++){ }	O(n)

For(int i=0;i<n;i++){
For(int j=0;j<n;j++){
}
}
	$O(n^2)$

For(int i=0; i<n;i++){
If(i==5)
break;
}
	O(1)

Space complexity

To store data how much space requirements are there is called as space complexity.

The space requirements are in proportion to what is the space complexity.

To find space complexity

Int[] arr=new int(n)
For(int i=0;i<n;i++){
}
	O(n)

Int[][] arr=new int[4][5]
For(int i=0;i<n;i++){
For(int j=0;j<n;j++){
}
}
	$O(n^2)$

Int n=10
Int s=0;
For(int i=0; i<n;i++){
int x=sc.nextInt();
S=s+i
}
	O(1)

Searching algorithm

Two types

1. Linear search(sequential search)
    a. If in your application search functionality is used very rarely, then do not spend time in sorting the data, and use linear search.
2. Binary search
    a. If in your application search functionality is used very often, then do spend time in sorting the data, and use binary search.

Step by step procedure to solve the problem is called as algorithm.
Pictorial representation of algorithm is called as flow chart.

Arr=[1,2,10,5,6,7]

What is time complexity to search 7.

a. O(n)
b. O(logn)
c. Given info is insufficient.
d. None of the above

1. Accept numbers form user and store it in the array.
2. Accept a number to search in the array.
3. Set j to 1;
4. Check number matches with arr[j],
5. if found then stop else go to step 9
6. increase j by 1
7. and go to step 4
8. repeat step 4 to 7 till array finishes. If array is finished go to step 10
9. Print data found.
10. Print not found.

Binary search

1. Accept numbers form user and store it in the array.
2. Accept a number to search in the array.
3. Set first to 0, last to length-1
4. Calculate mid=(first+last)/2
5. If arr[mid]==num then goto step 9
6. Otherwise if arr[mid]<num then set first to mid+1
7. If arr[mid]>num then set last =mid-1
8. Repeat steps 4 to 7 till first<=last, if false then goto step 10
9. Display “number found at mid position
10. Display “number not found”

Find number of comparisons to search 10, If the given array is
3, 4, 12, 10, 4, 3, 2, 7, 9
Comparisons: 4

2, 3, 3, 4, 4, 7, 9, 10, 12
Comparisons: 3

First=0, last=8, mid=4	4==10
First=5, last=8, mid=6	9==10
First=7, last=8, mid=7	10==10

Recursive binary search

[5, 7, 8, 10, 12, 13, 15, 17, 18, 20] search=15

binarySearch(0,9,15,arr)	F=0, l=9, search=15, mid=4	Return 6
binarySearch(5,9,15,arr)	F=5, l=9 search=15 mid=7	Return 6
binarySearch(5,6,15,arr)	F=5, l=6,search=15,mid=5	Return 6
binarySearch(6,6,15,arr)	F=6, l=6,search=15, mid=6	Return 6

Write a recursive add function to find addition of n numbers

1+2+3+4+5+…n

N+add(n-1)

```
int add(int n){
If(n<1)
   return 0;
If(n==1){
    Return 1;
}
return n+add(n-1);
}
```