Fundamental types in C++ are divided into three categories: **integral, floating point, and void. Integral types are capable of handling whole numbers**. Floating point types are capable of specifying values that may have fractional parts. The void type describes an empty set of values.

The category **integral** includes the following datatypes -

char

bool

short

int

long

long long

wchar\_t, \_\_wchar\_t

The category f**loating** point includes the following datatypes -

float

double

long double

2.Integral Data Type

char, short, int, and long are integral types that store various sizes of numbers. They can all be signed or unsigned. (Unsigned means only whole numbers, no negatives.) bool is typically thought to evaluate to true or false, but it is actually stored as 0 or 1, so it is also considered an integral type.

The expression used in the switch must be an integral type (int, char, and enum). Any other type of expression is not allowed.

%u treats the value as unsigned and %d as signed.

* x prints an unsigned number in hexadecimal.
* o prints an unsigned number in octal.
* u prints an unsigned number in decimal.
* **d prints a signed number in decimal.**
* **i prints a signed number in decimal**.

**There is no difference between the %i and %d format specifiers for printf. Consider a following example.**

int main()

{

    int num = 9;

     printf("Value of num using %%d is = %d\n", num);

     printf("Value of num using %%i is = %i\n", num);

    return 0;

}

**Output:**

Value of num using %d is = 9

Value of num using %i is = 9

* **%d** takes integer value as **signed** decimal integer i.e. it takes negative values along with positive values but values should be in decimal otherwise it will print garbage value. Consider a following example.
* **%i** takes integer value as integer value with **decimal, hexadecimal or octal** type.  
  To enter a value in hexadecimal format – value should be provided by preceding “0x” and value in octal format – value should be provided by preceding “0”.

int main()

{

    int a, b, c;

    printf("Enter value of a in decimal format:");

    scanf("%d", &a);

    printf("Enter value of b in octal format: ");

    scanf("%i", &b);

    printf("Enter value of c in hexadecimal format: ");

    scanf("%i", &c);

    printf("a = %i, b = %i, c = %i", a, b, c);

    return 0;

}

**Output:**

Enter value of a in decimal format:12

Enter value of b in octal format: 012

Enter value of c in hexadecimal format: 0x12

a = 12, b = 10, c = 18

|  |  |
| --- | --- |
| Format Specifier | Description |
| %d | Integer Format Specifier |
| %f | Float Format Specifier |
| %c | Character Format Specifier |
| %s | String Format Specifier |
| %u | Unsigned Integer Format Specifier |
| %ld | Long Int Format Specifier |

**Modifier**=**Qualifiers:**

**A modifier is used to alter the meaning of the base type** so that it more precisely(exactly) fits the needs of various situations. C++ allows the char, int, and double data types to have modifiers preceding them.

The data type modifiers are listed here −

* signed
* unsigned
* long
* short

-The modifiers signed, unsigned, long, and short can be applied to integer base types.

- **signed and unsigned can be applied to char**, **and long can be applied to double**.

- The modifiers signed and unsigned can also be used as prefix to long or short modifiers. For example, unsigned long int.

**Note** - **Double** is another floating-point type it is none other than **long float**.

-In c++, you can simply use the word **unsigned, short, or long, without int.** It automatically implies int. For example, the following two statements both declare unsigned integer variables.

unsigned x;

unsigned int y;

**1.If no data type is given to a variable, the compiler automatically converts it to int data type.**

 signed a;

 signed b;

**2. Signed is the default modifier for char and int data types.**

int x;

    char y;

    x = -1;

    y = -2;

    printf("x is %d and y is %d", x, y);

x is -1 and y is -2.

**3.  We can’t use any modifiers in float data type. If programmer tries to use it ,the compiler automatically gives compile time error**

 signed float a; //error

    short float b; //error

    return (0);

**const, volatile and restrict also a qualifiers in c++.**

**Data type qualifiers can be classified into 5 groups.**

**A storage class:**

A storage class **defines** the scope (visibility) and **life-time of variables** and/or functions within a C++ Program. There are following storage classes, which can be used in a C++ Program

* **auto**:- auto can only be used within functions, i.e., local variables(default.)
* **register**: The **register** storage class is used to define local variables that should be stored in a register instead of RAM like counters to quick access.
* **Static**:(can be Global or local) The **static** storage class instructs the compiler to keep a local variable in existence during the life-time of the program instead of creating and destroying it each time it comes into and goes out of scope. The static modifier may also be applied to global variables. When this is done, it causes that variable's scope to be restricted to the file in which it is declared.
* **Extern**:- The extern modifier is most commonly used when there are two or more files sharing the same global variables or functions External variables are also known as global variables. These variables are defined outside the function and are available globally throughout the function execution. The “extern” keyword is used to declare and define the external variables.
* **Mutable**:- The mutable specifier applies only to class objects,

**In computer generally Postfix expression preferred.**

Why postfix is used by computer or machine. The big advantage in prefix/postfix notation is that there never arise any questions like operator precedence.  
  
 But postfix expressions are preferred **because implementation of code for solving postfix expression is easier and more faster & efficient** then solving prefix expressions.

Infix notation is easy to read for humans , whereas pre-/postfix notation is easier to parse for a machine.

Tricks:1

I remember them as **SBI**(as in S.B.I bank: State Bank of India)

* S stands for Selection sort
* B stands for Bubble sort
* I stands for Insertion sort
  + - Example for best case is: Array is already sorted, or almost sorted.

In general:

* **Selection Sort**
  + **Worst, Average case performance is O(n Squared)**
  + **Best case is also O(n Squared)**
    - Example for best case is: Array is already sorted, or almost sorted.
  + This means fastest (best case) it can run is O(n Squared)
  + That is, if your elements are 5,000,000 (5 million) in an array, it may compare about 5,000,000 multiplied by 5,000,000, that is 25,000,000,000,000 many comparisons may happen to sort your 5,000,000 elements of your array.
  + The greater number of comparisons the slow your program will run.
  + So, never use Selection Sort
* **Bubble sort**
  + **Worst, Average case performance is O(n Squared)**
  + **Best case is O(n)**
  + That is, if your elements are 5,000,000 (5 million) in an array, it may compare about 5,000,000 only, (which is a lot better than comparing 25,000,000,000,000 elements as in Selection sort above)
  + So, it’s kind of better than Selection sort.
  + **Bubble sort is 70% faster than Selection sort.**
* **Insertion sort**
  + **Worst, Average case performance is O(n Squared)**
  + **Best case is O(n)**
  + This looks same as Bubble sort above, right?
  + However, **in practice, Insertion sort runs faster than Bubble sort.**
  + **Insertion sort is 500% faster than Bubble sort**, because there are **less swaps while sorting elements in array,** less writes done, less cache misses (possibly, works more on recently accessed items/ caches) compared to bubble sort

Hope it helped, remember SBI, and I is best.

**Trick:2**

### how to learn complexities of sorting algorithms

The sorting algorithms we usually come across are:

Bubble , Insertion, Heap, Merge, Selection and Quick Sorting and it is very difficult to learn the complexities of all these algorithms. It becomes a more tedious task when it comes to three cases of complexities(Worst, Average, Best).

Below is a method to memorize all the complexities(including average, best and worst cases) of these six algorithms.

First letter of every sorting stands for that sorting. For ex.- B stands for Bubble sort, I stands for Insertion sort and so on....

there are only types cases of complexities:

1. n
2. n2(n square)
3. nlogn

(dont worry about the complexity "n", it is not used frequently)  
  
the trick to remember is as follows:  
  
prepare a table in which first column contains all the algorithms names(in any order)  
and first row contains all three cases(worst, average and best) as shown in figure in the specified order(i.e. increasing from worst to best).

Now, the words you have to remember are

* BIn

B means bubble sort  
I means insertion sort  
  
both Bubble and insertion sort have same set of complexities but their best case is different than other. so remember "BIn": short for "Best Is n".

* HtMl

H: Heap  
M: Merge  
t: just to remeber the whole word. it is of no use otherwise.  
l means complexity "nlogn".  
  
both Heap and Merge have same set of complexities and also all of them are nlong(l).

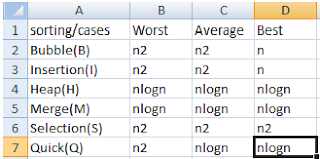
* S stands for selection sort and "square". so all three of selection sort are n2.
* Qsl(Queen of  scott-land): first column of quick sort is n2(as "s" in scott) and rest are nlong (as l in land).

Note: nlogn is used whenever it is specified otherwise use n2 as default

like: in BIn, use n2 in average and worst cases.

as I said earlier that complexity "n" is not much of use as it is used only twice that too can easily be memorize as "best is n".

Practice this two or three times and you will remember it as good as anything.

[](http://2.bp.blogspot.com/-Vcf5RP9oCuA/T9OxQUlvbgI/AAAAAAAAAEc/lnUfRMj-F7c/s1600/Sorting_Table.png)

Efficiency of any sorting algorithm depends on two component:

1. Time complexity: how much time an algorithm take with respect to input size.
2. Space complexity : how much space your program is taken

**HeapSort**: It is the slowest of the **sorting** algorithms than Merge and Quick Sort

# **Quick Sort vs Merge Sort**

1.In worst case quick sort have O(2^n) complexity and merge sort have O(Nlon N).so in worst case merge sort gives better output.

2. The quick sort is internal sorting method where the data is sorted in main memory ,quick sort there are no need of another array.  
whereas the merge sort is external sorting method  In merge sort we required one more array to merge and sort .

 By space wise quick sort gives better result.

**Conclusion** : ***Quick sort*** is best algorithm for sorting an array.

Bubble Sort vs Insertion Sort

* + Insertion sort is faster than Bubble sort, because there are less **swaps while sorting elements in array**, less writes done, less cache misses (possibly, do more works on recently accessed items/ caches) compared to bubble sort

**Binary Tree**

Binary Tree is a hierarchical data structure in which  
  
There is no particular order to how the nodes are to be organized.

each node has zero, one, or at the most, two children.  
  
**Binary Search**

A Binary Search Tree is a type of binary tree data structure in which the nodes are arranged in order,

there is a relative order to how the nodes should be organized.

The value of the nodes in the left sub tree are less than or equal to the value of the root node, and the nodes to the right sub tree have values greater than or equal to the value of the root node.

**AVL/ Height Balanced Tree –**

AVL tree is binary search tree with additional property that difference between height of left sub-tree and right sub-tree of any node can’t be more than 1

AVL(**A**delson-**V**elsky and **L**andis)This guarantees that lookup, insertion, and deletion all take O(log N) time in both the average and worst cases.

See: <https://www.gatevidyalay.com/linear-search-searching-algorithms/>

<https://www.studytonight.com/data-structures/introduction-to-sorting>

**The time complexity of linear search algorithm over an array of n elements is** O(n)

**For a linear search in an array of n elements the time complexity for best, worst and average case are ......., ....... and ........ respectively** O(1),O(n) and O(n)

**The time required to search an element in a linked list of length n is** O(n)

**The worst case time required to search a given element in a sorted linked list of length n is** O(n)

**Non-comparison based sorting –**  
In non-comparison based sorting, elements of array are not compared with each other to find the sorted array.

* **Radix sort –**  
  Best, average and worst case time complexity: nk where k is the maximum number of digits in elements of array.
* **Count sort –**  
  Best, average and worst case time complexity: n+k where k is the size of count array.
* **Bucket sort –**  
  Best and average time complexity: n+k where k is the number of buckets.  
  Worst case time complexity: n^2 if all elements belong to same bucket.

**In-place/Outplace technique –**  
A sorting technique is in-place if it does not use any extra memory to sort the array.  
Among the comparison based techniques discussed**, only merge sort is outplaced technique** as it requires an extra array to merge the sorted subarrays.  
Among the non-comparison based techniques discussed, all are outplaced techniques.

**Online/Offline technique –**  
A sorting technique is considered Online if it can accept new data while the procedure is ongoing i.e. complete data is not required to start the sorting operation.  
Among the comparison based techniques discussed, **only Insertion Sort** qualifies for this because of the underlying algorithm it uses i.e. it processes the array (not just elements) from left to right and if new elements are added to the right, it doesn’t impact the ongoing operation.

**Stable/Unstable technique –**  
A sorting technique is stable if it does not change the order of elements with the same value.  
Out of comparison based techniques**, bubble sort, insertion sort and merge sort and also straight insertion sort are stable techniques**. Selection sort is unstable as it may change the order of elements with the same value. For example, consider the array 4, 4, 1, 3.

In the first iteration, the minimum element found is 1 and it is swapped with 4 at 0th position. Therefore, the order of 4 with respect to 4 at the 1st position will change. Similarly, quick sort and heap sort are also unstable.

Out of non-comparison based techniques, Counting sort and Bucket sort are stable sorting techniques whereas radix sort stability depends on the underlying algorithm used for sorting.

Adaptive and Non-Adaptive

Adaptive sorting algorithms: If order of the elements to be sorted of an input array matters (or) affects the time complexity  
1. Bubble Sort  
2. Insertion Sort  
3. Quick Sort

Non-adaptive sorting algorithms: order of the elements in the input array doesn’t matter, complexity remains same  
1. Selection Sort  
2. Merge Sort  
3. Heap Sort

### Comparison sorts

[Quicksort](https://en.wikipedia.org/wiki/Quicksort), [Merge sort](https://en.wikipedia.org/wiki/Merge_sort), [Heapsort](https://en.wikipedia.org/wiki/Heapsort), [Insertion sort](https://en.wikipedia.org/wiki/Insertion_sort), [Selection sort](https://en.wikipedia.org/wiki/Selection_sort). [Binary tree sort](https://en.wikipedia.org/wiki/Binary_tree_sort), [Bubble sort](https://en.wikipedia.org/wiki/Bubble_sort), [Shell sort](https://en.wikipedia.org/wiki/Shell_sort)

[Timsort](https://en.wikipedia.org/wiki/Timsort), [Cubesort](https://en.wikipedia.org/wiki/Cubesort), [Block sort](https://en.wikipedia.org/wiki/Block_sort), [Odd–even sort](https://en.wikipedia.org/wiki/Odd%E2%80%93even_sort)

A comparison sort cannot perform better than *O*(*n* log *n*).

**Analysis of sorting techniques :**

* When the array is almost sorted, insertion sort can be preferred.
* When order of input is not known, merge sort is preferred as it has worst case time complexity of nlogn and it is stable as well.
* When the array is sorted, insertion and bubble sort gives complexity of n but quick sort gives complexity of n^2.
* **Merge sort is often the best choice for sorting a**[**linked list**](https://en.wikipedia.org/wiki/Linked_list)**:** merge operation of merge sort can be implemented without extra space for linked lists.
* For array quicksort is much efficient than merge sort because,

quicksort is much faster because, it is in-place sort which means without needing to create any auxiliary arrays to hold temporary values.

If n=2, then the value of O(n log n) is = 2

##### Which sorting algorithm is the slowest algorithm for large number of data?

1. Quick sort
2. Heap sort
3. Bubble sort
4. Shell sort

Explanation: Quick sort, Heap sort and Shell sort all have best case time complexity as O(nlogn) and Bubble sort has time complexity of O(n2).  
So, Bubble sort is slowest.

##### Retrieval operation is fastest in which data structure

1. Heap
2. Stack
3. Linked list
4. None

View Answer

Answer:A

Refer for MCQ

<https://www.interviewsansar.com/2017/02/15/mcq-tree-binary-tree-binary-search-tree-avl-tree/>

<https://mympsc.com/Share.aspx?ArticleID=56765571-b098-4d82-b255-e68baee6ff24>

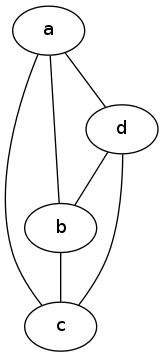
**Dope Vectors** is a data structure that is **used** by compilers to store some metadata about the array like its total size, the size of one unit

**Graph**:

Graph can be represented as array or linked list

In an undirected graph, the sum of degrees of all the nodes

is twice the number of edges or must be even



The degree sequence is 3,3,3,3.

Given an undirected graph G with V vertices and E edges, the sum of the degrees of all vertices is:2E,E is Edges

A path in a graph is a walk that does not repeat any vertices.

The length of a path is the number of edges traversed by the path

a rightarrow d rightarrow b rightarrow cit is a path of length 3)

 a  rightarrow d rightarrow a rightarrow bThis is a walk but not a path since it repeats the vertex a.

Digraph: directed edges

#### Self-loops and Multigraphs

one of which makes it a multigraph.

* Node A has three self-loops.
* Nodes A and B have five edges between them.
* Nodes A and C have two edges between them

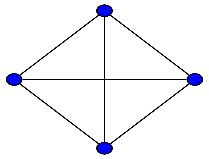
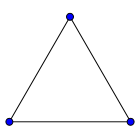
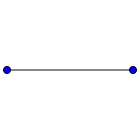
A multigraph may or may not contain self-loops.

**Complete Graph**:A graph that has an edge between every single node in the graph.

1.**Complete Undirected Graph:-** Complete digraphs are a graphs in which every pair of nodes is connected by a unique edge

A graph with exactly n\*(n-1)/2 edges is said to be Complete Undirected graph Or

The maximum no. of edges in undirected graph with n vertices is: n\*(n-1)/2

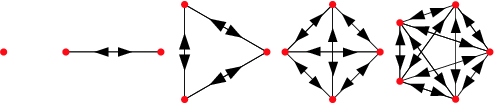
How many undirected graphs (not necessarily connected) can be constructed out of a given set V = {v1, v2, ... vn} of n vertices?is

|  |
| --- |
| 2n(n-1)/2 |

2.**Complete digraphs(Directed grapg):**Complete digraphs are digraphs in which every pair of nodes is connected by a bidirectional edge.

A graph with exactly n\*(n-1) edges is said to be Complete directed graph OR

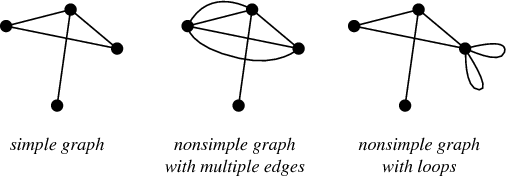
The maximum no. of edges in directed graph with n vertices is: n\*(n-1)



**Simple Graph:**

A simple graph, also called a strict graph containing no [graph loops](http://mathworld.wolfram.com/GraphLoop.html) or [multiple edges](http://mathworld.wolfram.com/MultipleEdge.html) A simple graph may be either [connected](http://mathworld.wolfram.com/ConnectedGraph.html) or [disconnected](http://mathworld.wolfram.com/DisconnectedGraph.html).

The maximum degree of any vertex in a simple graph with n vertices is (n-1)



**Cyclic Graph:**

A cyclic graph is a graph containing at least one [graph cycle](http://mathworld.wolfram.com/GraphCycle.html).

The number of **vertices** in C**n** equals the number of edges, and every **vertex** has degree 2;

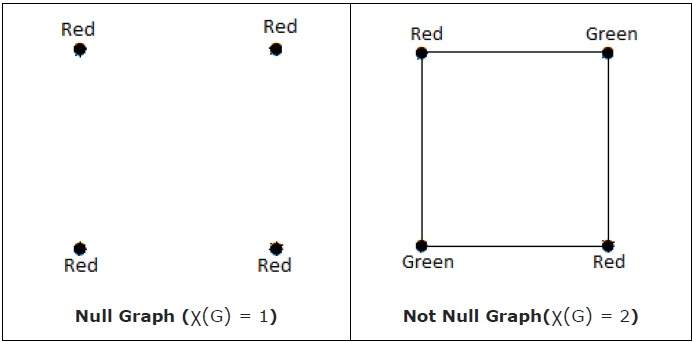
**Graph Coloring:**

## Vertex Coloring

Vertex coloring is an assignment of colors to the vertices of a graph ‘G’ such that no two adjacent vertices have the same color. Simply put, no two vertices of an edge should be of the same color.

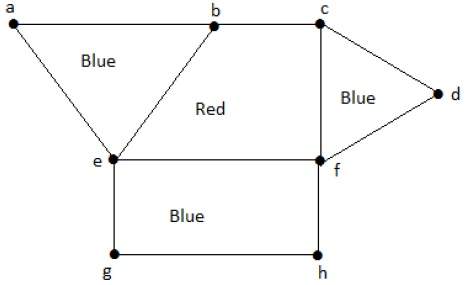
## Chromatic Number

The minimum number of colors required for vertex coloring of graph ‘G’ is called as the chromatic number of G, denoted by X(G).



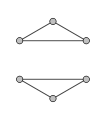
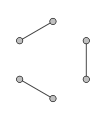
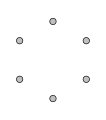
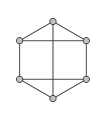
## Region Coloring

Region coloring is an assignment of colors to the regions of a planar graph such that no two adjacent regions have the same color. Two regions are said to be adjacent if they have a common edge.



**Regular Graph** :

In [graph theory](https://en.wikipedia.org/wiki/Graph_theory), a **regular graph** is a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) where each vertex has the same degree;

**Important MCQ**

1.How many edges are there in a graph with 10 vertices each of degree 6?

Ans:30

**No. of edges=nd/2**

2. What is the maximum number of edges in a simple graph with n vertices?

Therefore, the **maximum number of edges** is **N** \* (**N** - 1) .

In an undirected **graph** (excluding multigraphs), the answer is **n**\*(**n**-1)/2. In a directed **graph** an **edge** may occur in both directions between two **nodes**, then the answer is **n**\*(**n**-1)

3.what is no. of Edges in complete graph with ‘n’ vertices is equals to:n(n-1)/2

4.In Binary tree.the number of internal node of degree 1is 5 and the number of internal node of degree 2 is 10 .The number of leaf node in binary tree is : 11

Ans: In binary tree,the number of leaf node is always 1 more than number of internal node with 2 children

5.isophormic Graph: Minimum Colors

6.Cromatic Number: Graph have same degree.

7. **The number of possible ordered trees with 3 nodes A, B, C is**

**Ans:**is 12. The tree may be of depth 2 or 1. if the depth is 2, we have 6 possible trees. This is because one of the three nodes A, B, C may be the root and the next level may be one of the remaining two nodes. If the depth is 1, the root may be one of the 3 nodes A, B, C. Corresponding to a root, say A, two trees as possible as this.

8**. The number of possible binary trees with 4 nodes is: 14**

**9. he number of possible binary trees with 3 nodes is:5**

**Ans:** Total number of possible Binary Search Trees with n different keys (countBST(n)) = [Catalan number Cn](https://www.geeksforgeeks.org/program-nth-catalan-number)= (2n)! / ((n + 1)! \* n!)

9.b The order of the binary search algorithm is:   n(because it is in sorted order)

10. **The depth of a complete binary tree with 'n nodes is (log is to be base two)**

**Answer:**log (n+1)-1

11-a.A binary tree with n leaf nodes contains:2n-1 nodes

11.b A binary tree that has n leaf nodes. The no. of nodes of degree 2 in this tree is ?

Ans:  n-1.

It can be proved by induction that a binary tree with n leaf nodes will have total 2n-1 nodes. So number of non – leaf nodes is (2n-1) – n = n-1.

11.c. A binary tree with 10 leaves has exactly :-19 nodes(2\*10-1=19)

11.d A binary tree T has 20 leaves. The number of nodes in T having two children is:19

12.A 3-ary tree in which every internal node has exactly 3 children. The number of leaf nodes in such a tree with 6 internal nodes will be:

Ans: 13

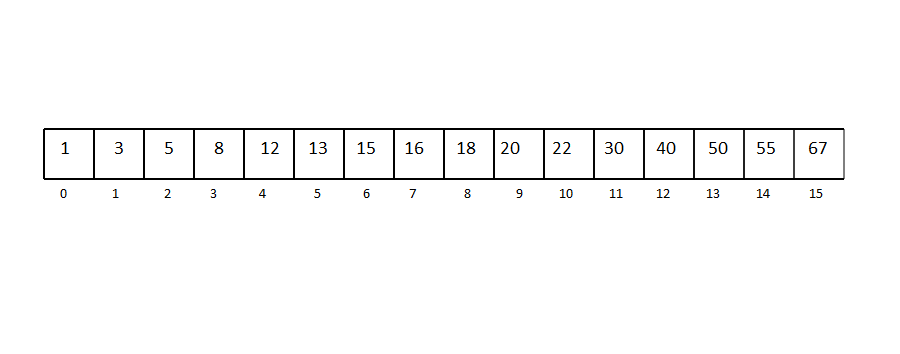
Tip:In a complete k-ary tree, every internal node has exactly k children. The number of leaves in such a tree with n internal nodes is: n( k – 1) + 1

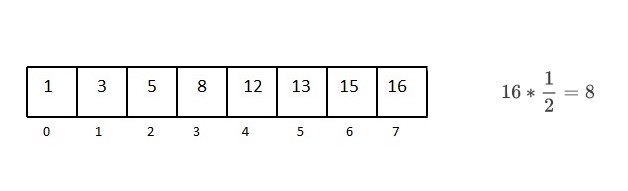
# 13.What is the maximum number of comparisons that a binary search function will make when searching for a value in a 1,000

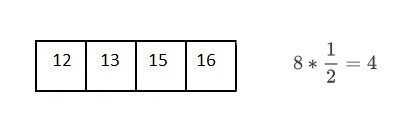
Ans: **10 Guesses**

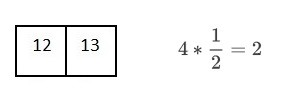
Since binary search has a best case efficiency of O(1) and worst case (average case) efficiency of O(log n), we will look at an example of the worst case. Consider a sorted array of 16 elements.

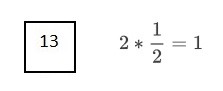
For the worst case, let us say we want to search for the the number 13











Total 4 comparision Final result is n=2^k

14.Number of Comparision in Merge Sort

For just five distinct numbers to sort, the maximum number of comparisons you can have is 8 and minimum number of comparisons is 7. Here's why:-

Suppose the array is a,b,c,d,e

divide recursively: a,b,c and d,e

divide recursively: a,b&c and d&e

divide recursively: a&b & c and d&e

Now, merging which will require comparison-

a & b : one comparison to form a,b

a,b & c : two comparisons to form a,b,c

d & e : one comparison to form d,e

a,b,c and d,e : four comparison in worst case or three comparisons id d is the largest element of array to form a,b,c,d,e

So, the total number of comparisons will be eight in worst case and seven in the best case

Suppose there are logn sorted lists of n logn elements each. The time complexity of producing a sorted list of all these elements is (use heap data structure):- O(n log logn)

**Skolemization is the process of replacing existentially quantified variables in a formula with new constants (called Skolem constants) or functions (called Skolem functions)**

Leaves of which of the following trees are at the same level? B-tree

Number of binary trees formed with 5 nodes are: 42

For a B-tree of height h and degree t, the total CPU time used to insert a node is: O(th)

The time complexity to build a heap with a list of n numbers is: O(n)

An undirected graph is Eulerian if and only if all vertices of G are of the sum of the degrees of all nodes is: ODD degree

An undirected graph G has n vertices and n-1 edges then G is: Is a Tree

Graph having every pair of vertices connected is called: Complete graph

To be continue……..<http://entrance4cdac.blogspot.com/p/data-structure-mcq.html>

From Q-125 in reverse order last Sunday