**DAY-19**

Algorithm : Minimum steps we require.

Output

Input

Condition for desired o/p

Characteristics of an Algorithm:

* Clear and unambiguous
* Well defined inputs
* Well defined output
* Finite-ness
* Feasible
* Language independent

How to write an algorithm:

🡪doesnot have any well defined standards

🡪Alg are never created to support a specific programming lang

🡪To find a sol to a problem, we create an algorithm. A problem can be solved in several different ways.

The alg may be studied on two levels.

1)prior analysis

2) post analysis

**🡪**

Time complexity: The big O notation expreses an algorithm’s time complexity.

There are different types of time complexities which can be analyzed for an algorithm:

----Best Case Time Complexity.

----Worst Case Time Complexity.

----Average Case Time Complexity

**(Best + Average )//2 = Average time Complexity**

Space complexity:

**Space complexity = Auxiliary space(Prog instruction,jump ,return statements) + input size**

**Cons of algorithms:**

Difficult to demonstrate the looping statements..

All algorithm cant be programmed

**Algorithm vs Programming**

An algorithm is more like a concept, a technique to solve a problem.

An algorithm may be run by humans

A programmer is related to executing one or more tasks by a computer.

A programing can be runned by compiler.

**Algorithm Design Algorithms:**

1. Top-Down Approach
2. Bottom – Up Approach

Big O Notation :

Big O notation is used in computer Science to describe the performance.

* O(1)- describes an algorithm that will always execute in the same time (or space)

Regardless of size of the input data set.

Ex: Push And Pop operations for a stack.

* O(N) – describes an algorithm whose performance will grow linearly and in direct proportion to the size of the input data set.

----Eg : Linear Search with unsorted data

* O(Nsqaure) – represents an algorithm whose performance is directly proportional to the square of the size of the input data set
* This is common with algorithms that involve the nested iteration over the data set
* O(log N) – Logarithmic Time

The iterative halving of data sets described in the binary search example produces a growth curve that peaks at the beginning and slowly flattens out as the size of the data sets increases

Eg : Binary Search

* An input data set containing 10 items takes one sec to complete, a data set containing 100 items takes 2 seconds

If the size is small , and data in unsorted we can use linear search.(10 items- 10 sec)

* (N log N)
* Eg : More Advanced sorting algoritms : quick sort,merge sort.

Upper Bound = Worst Case= Grows Gradually

**Merge sort**

- We’ll check whether the 1st index is less than last index or not…because if 1st index is greater than the last index then no list is present

OUICK SORT

1) Comparison

2) In place

3) Unstable

4) Recursive algorithm

- Lptr is checked with pivot for lesser values than it

- Rptr is checked with pivot for greater values than it

- If any of the above condition fails we stop the iteration/recursion

- When the left ptr and right pointer stops we swap the values, 8 and 4 are swap

Before exchange: 3 5 8 1 2 9 4 7 6

L R P

After exchange : 3 5 4 1 2 9 8 7h 6

L R P

- When lptr and rptr reach at the same position then that position value is swapped with the pivot value

3 5 4 1 2 9 8 7 6

LR P

- After swapping pivot, the pivot values find it correct position, in other words(pivot) is the only element which is sorted in the list

3 5 4 1 2 6 8 7 9

LR P

Quick sort cannot be used in phone book(contacts) because it is unstable…location varies -- it uses insertion sort and merge sort

Note : When the left pointer is crossing the right pointer (if my pivot element is greater then left as well as right)