**ALGORITHMS**

Algorithm characteristics

1. Clear and unambiguous
2. Well defined input
3. Well defined output
4. Finiteness
5. Feasibility
6. Language independent

All algorithms cannot be programmed but all programs are algorithms

Approaches

1. Top Down approach
2. Bottom up approach

Time complexity

1. Best time complexity ---- if n = 1
2. Worst time complexity ---- if found at last or not found
3. Average time complexity ---- Mid of best and worst

O(1) 🡺 always execute in the same time or space regardless of the size of the input data set. Ex: PUSH and POP operation of stack

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

* Big O notation will always assume the upper limit
* Ex : Linear search with unsorted data

O(N^2) 🡺 It represents an algorithm whose performance is directly proportiomal to the square of the size of the input data set.

Ex: Two dimensional arrays

O(log N) – Logarithmic time 🡺 The iterative halving of data sets described in the binary search example produces the growth curve that peaks at the beginning and slowly flattens out as the size of the data sets increases

Ex : Binary search: an input data set containing 10 items takes one second to complete, a data set containing 100 items takes two seconds and a data set containing 100 items takes three seconds.

O(N log N) – Logarithmic time 🡺 Ex : More advanced sorting algorithms: quick sort, merge sort

Note: Upper bound means worst case

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)

**VCF file in phone check in contacts**

Develop a phone book application using the sorting techniques