Requirements-basics python-variables,data types,conditional statements,loops and functions.

Note:

Leetcode-leetcode or hackerrank are the people who get solutions by practicing multiple problems,people ask how they get the solutions?

they get it by solving important coding problems and after solving multiple patterns we will find the approach and solution by ourselves.

DAY 1- GOAL

Day 1 Goals

-Learn Big O Analysis to find Time and Space complexity

BEFORE THIS LETS LEARN FEW DEFINITIONS:

- WHAT ARE DATASTRUCTURES?
- WHY DO WE NEED THEM TO ACE CODING INTERVIEWS?

1a)DATASTRUCTURES:

- Collection of data values.
- Relationships among them.
- Functions/operations that can be applied on data.
- Ex: Arrays are prebuilt Data structures

1. Arr=[1,5,6,7]-collections of data values
2. [1,5,6,7]
   0 1 2 3 -index values are relationship among them.

3.Arr.push(9)-functions or operations that can be applied.

2a)why do we need them to ace coding interviews

I/p -> Algorithm -> O/P

Ex:Sum of two numbers

a,b -> s=a+b ->s

same questions we can solve in multiple ways.

Some datastructures are more efficient than others to do some tasks

- Time
- Space complexity

Complexity analysis and big0:

What is need for complexity analysis:

Ex:Sum of n numbers

i/p: ->algorithm ->o/p

n n*(n-1)//2 n-1 :10=(4+3+2+1)

5 or (n-1)+(n-2)+…..1

- WHICH APPROACH IS BETTER:

WE COMPARE USING TIME AND SPACE COMPLEXITY.

- WHAT DOES BETTER MEANS?

FASTER ->TIME

LESS MEMORY->SPACE COMPLEXITY

Ex:(N-1)+(N-2)+(N-3)+…1

N-1+N-2=2N-3 OPPERATIONS HERE WHAT IF 2000 THEN 2000N-3

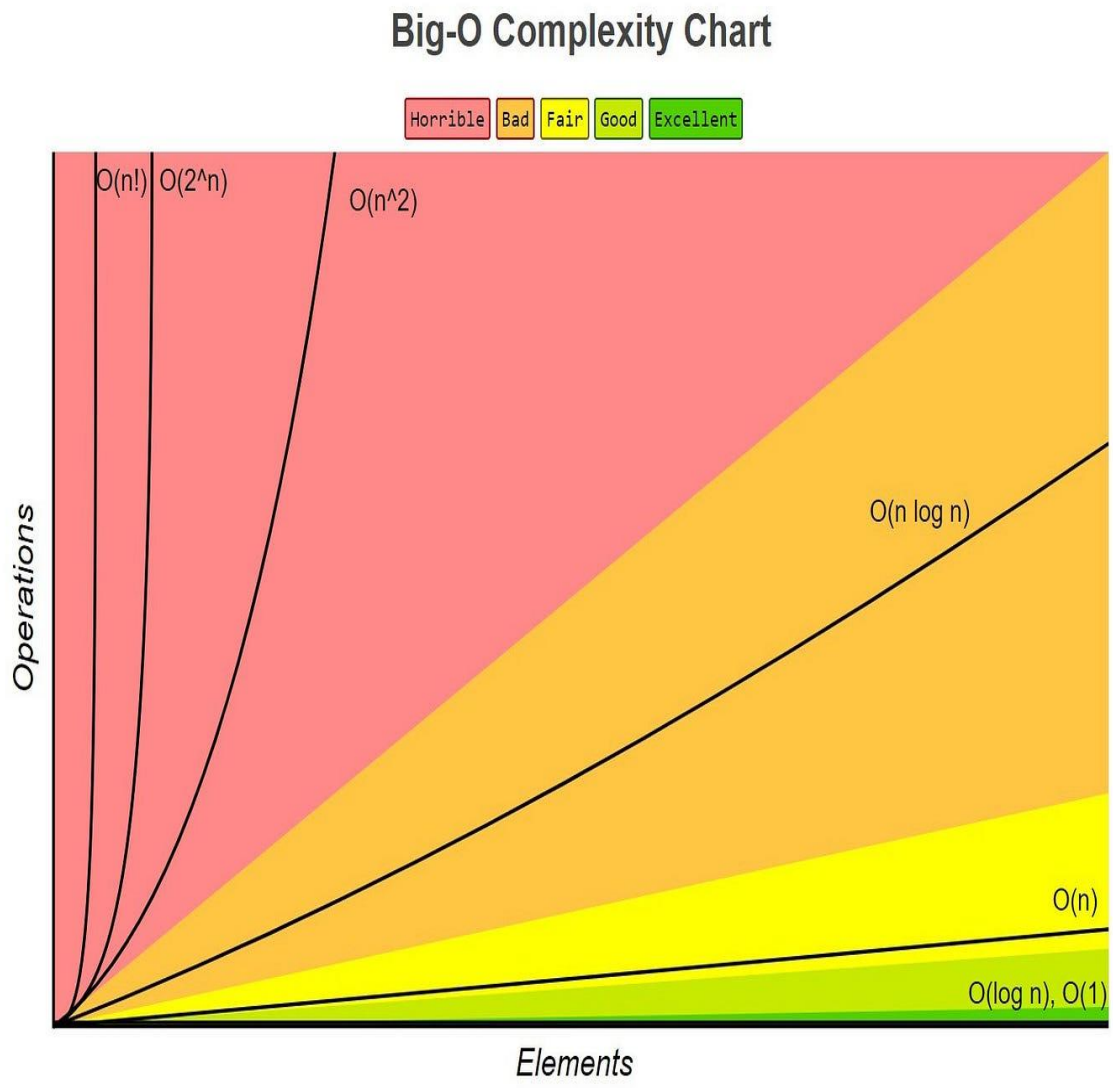
SO,BEST APPROACH IS

N*(N-1)//2 BY USING THIS FOEMULA WE CAN SOLVE EASILY AND BEST APPROACH.

BIG 0-HOW RUNTIME OF ALGORITHM GOES AS I/P GROWS.

- DEF: IS A MATHEMATICAL NOTATION THAT DESCIBES LIMITING BEHAVIOUR OF FUNCTION.
- IS USED TO CLASSIFY ALGORITHM ACCORDING TO RUNTIME OR SPACE REQUIREMENTS GROW AS INPUT SIZE GROWS.

BIG O – COMPLEXITY CHART



- 0(1)-constant
- 0(logn)-binary search algorithm

- 0(n)-traverse elements of array and add them
- Ex:[1,2,3,4]-loop traverse
- 0(nlogn)-merge sort-[4,1,2,3]-[1,2,3,4]
- 0(n^2)-given array and form all the possibility pairs
- Ex:[1,2,3]-(1,1) (1,2) (1,3) (2,1) (2,2) (2,3) (3,1) (3,2) (3,3)
- 0(2^n)-fibonacci
- 0(n!)-5!=5*4*3*2*1 input increases number of operations increases.

0(n) is better than 0(n^2).

Space complexity: Is how much Auxiliary memory needed to run the algorithm.

- Some solutions make a new array,string or hasmap those case some extra space taken.

- Numbers,Boolean,null,undefined-constant space.

Techniques to simplify big 0 expressions:

1.Drop constant:

- **0(25 N^2)- 25 is constant drop it 0(N^2)**
- **Drop insignificant terms – as N increases**
  Ex:
  0(N^3+100N)-100N is insignificant term 0(N^3)
- **Different input parameters:**
- **0(N^3+M)-you cannot drop M**
- **N^3 where there can be a chance N=1 and M=10,000 in that case M is bigger than N.**

Logarithms:

- Log n in coding it is always log base 2 n

Log 16=?

In coding Log base 2 16=4

2^?=16 2^4=16

- Log 8=?

Log base 2 8=3 because 2^?=8 2^3=8

- Log 1024 =log base 2 1024 =10 2^10=1024.