Resources To learn the basics : https://www.interviewbit.com/courses/programming/topics/time-complexity/#problems

Question:Why we find it ?

Answer :To calculate how much time our program will take in seconds for the worst case scenario and see if it's within the

constraints or not.

Constraints are probably the most important part of the problem.

1. They can give you exact time complexity that is needed to pass all test cases .

2. They can tell you the space complexity that you will be needed.

3. They can tell you about data type that you should be using.

Part 1. Time Complexity

Most of the platforms uses 1 sec to perform 10^8 operations

Suppose you perform x operations

10^8 operations = 1 sec

x operation will take = x/10^8 secs

Generally you will have the time limit of 1-2 secs to pass all the test cases so ,

your number of operation must not exceed 10^8 operations.

Now how to calculate number of operations ?

Suppose constraints of n is 1<=n<=10^7

you have calculated a time complexity of your program to be N\*logN

If you want to calculate the number of operation the replace N with maximum value of N because if your maximum operation can

be performed within the time limit than lesser operation will be performed easily.

So number of operations are= 10^7\* log(10^7) = 7 \* 10^7 assuming log base 10.

Now if you want time in seconds then that will be = 10^7 \* 7 / 10^8 = 7\* 10^-1 = 10 ms .

So its easy to observe that we just need to limit our number of operations to 10^8 for passing all test cases.

How you can easily find the limit of your program's complexity according to given Constraints ?

Remember these constraints and the maximum time limit you can have for these constraints .

Constraints Time Complexity

1<=N<=10^18 O( log N )

1<=N<=10^8 O( N )

1<=N<=10^7 O( N logN )

1<=N<=10^4 O( N^2 )

1<=N<=500 O( N^3 )

1<=N<=90 O( N^4 )

1<=N<=20 O( 2^N )

1<=N<=10 O( N! )

Some Commonly used For loops and their time Complexity

For loops Their Corresponding time Complexity

1. for( i=0;i<N:i++) O( N ) // If in updation operation of summation or subtraction then complexity will always be in linear .

2. for( i=0;i<N;i\*=2) O( logN ) // If in updation operation of multiplication or division is performed the complexity will be in logarithmic.

3. for(i=0;i<N;i++) O( N\*N ) // In nested loops, multiply the Complexity of all loops to get final complexity

{

for(j=0;j<N;j++)

//nested loops

}

Few other tips.

1. Always remember the time complexity for famous algorithms like divide and conquer it always results in logarithmic complexity.

2. Whenever you learn any built in functions always remember its time Complexity eg. sort()=N\*logN .

3. Always find time complexity for the worst case.

// Happy Coding //