02. Time Complexity

14:08

Agenda:

1. Log Basics + Iteration Problems
2. Comparing Iterations using Graph
3. Time Complexity - Definition and Notations (Asymptotic Analysis - Big O)
4. TLE
5. Importance of Constraints

1. Log Basics + Iterations

1. $\log_b a = c \Rightarrow a = c^b$
2. $\log_b a^n \Rightarrow n \log_b a$
3. $\log x*y \Rightarrow \log x + \log y$
4. $\log x/y \Rightarrow \log x - \log y$
5. $\log 1/x \Rightarrow -\log x$

Iteration Problems :

1.
```
For(int i=1; i<N; i=i*2){
        .....
}
```

$1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow 16 \ldots \; i*2^K = N \Rightarrow \boxed{K = \log(N)}$

2.
```
For(int i=0; i<N; i=i*2){
        ....
}
```

$0 \rightarrow 0 \rightarrow \ldots \infty$

3.
```
for(i=1; i<=10; i++){
   for(j=1; j<=N; j++){
   /    ......./
   }
}
```

$10*1 \rightarrow 10*2 \ldots 10*N \Rightarrow O(N)$

4.
```
for(i=1; i<=N; i++){
   for(j=1; j<=N; j++){
      ...
   }
}
```

$N*1 \rightarrow N*2 \rightarrow N*3 \ldots N*N \Rightarrow O(N^2)$

5.
```
for(i=1; i <= N; i++){
   for(j=1; j <= N; j = j*2){
      ...
   }
}
```

$1*\log N \rightarrow 2*\log N \ldots N*\log N \Rightarrow O(N \log N)$

6.
```
for(i = 1; i <= 4; i++) {
   for(j = 1; j <= i ; j++) {
      //print(i+j)
   }
}
```

$1 \rightarrow 1*2 \rightarrow 1*3 \rightarrow 1*4 \Rightarrow O(1)$

7.
```
for(i=1; i<=N; i++){
   for(j=1; j<=(2^i); j++)
   {
      ...
   }
}
```

$[1\ 2] \rightarrow [1\ 4] \rightarrow [1\ 8] \rightarrow \ldots [1\ 2^n] \Rightarrow$ This is GP $\{2, 4, 8, \ldots 2^n\}$

Using GP $\Rightarrow 2*(2^N - 1)$

Comparing Algorithm using Graph

Consider two algorithms Algo1 and Algo2 given by Kushal and Ishani respectively.
 Comment

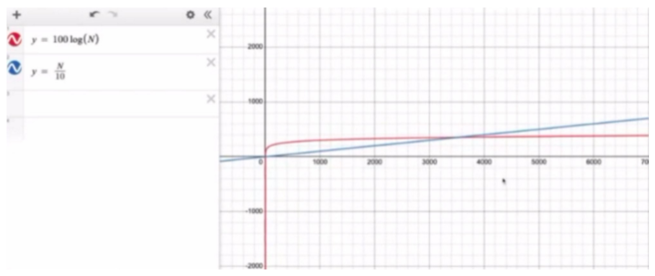
Considering N to be the size of the input:

Algo	Number of Iterations
Algo1	100 * log(N)
Algo2	N / 10

Now, see the graph of the two algorithms based on N.

Graphs info:

X-axis plots N (input size)
Red line (Algo 1): 100 * log(N)
Blue line (Algo 2): N/10



Observations:
Assuming both graphs intersect at N = 3500, let's draw some observations.

For small input (N <= 3500), Ishan's algorithm performed better.
For large input (N > 3500), Kushal's algorithm performed better.

In today's world data is huge

IndiaVSPak match viewership was 18M.
Baby Shark video has 2.8B views.
Therefore, Kushal's algorithm won since it has lesser iterations for huge data value.

We use Asymptotic Analysis to estimate the performance of an Algorithm when Input is huge.

Asymptotic Analysis OR Big(O) simply means analysing performance of algorithms for larger inputs.

Calculation of Big(O)

Steps for Big O calculation are as follows:

- Calculate Iterations based on Input Size
- Ignore Lower Order Terms
- Ignore Constant Coefficients

Comparison Order:

log(N) < sqrt(N) < N < N log(N) < N sqrt(N) < N^2 < N^3 < 2^(N) < N! < N^N

Why do we neglect Lower Order Terms

Let's say the number of Iterations of an Algorithm are: N2+10N

N	Total Iterations = N2+10N	Lower Order Term = 10N	% of 10N in total iterations = 10N/(N2+10N)*100
10	200	100	50%
100	104+103	103	Close to 9%
10000	108+105	105	0.1%

We can say that, as the Input Size increases, the contribution of Lower Order Terms decreases.

Issues with Big(O)

- We cannot always say that one algorithm will always be better than the other algorithm.
- If 2 algorithms have same higher order terms, then Big O is not capable to identify algorithm with higher iterations.

Time Limit Exceeded Order

- Codes are executed on online servers of various platforms such as Codechef, Codeforces, etc.
- The processing speed of their server machines is 1 GHz which means they can perform 10^9 instructions per second.
- Generally, codes should be executed in 1 second.

Using this information, we can say at max our code should have at most 10^9 instructions.

Importance of Constraints

Question
If 1 <= N <= 105,
then which of the following Big O will work ?

Complexity	Iterations	Works ?
O(N3)	(105)3	No
O(N2) log N	(1010)*log 105	No
O(N2)	(105)2	No
O(N * log N)	(105)*log 105	Yes