Time Complexity

Nov 24, 2023

AGENDA

- More practice on calculating no. of iterations
- Basics of logarithm
- Asymptotic analysis & Big-O Notation
- Issues with Big-O
- Importance of constraints

Logarithms.

Q. How many times do you need to divide 15 by 2 until it reaches 1 ?

$15/2 \rightarrow 7$

$7/2 \rightarrow 3$

$3/2 \rightarrow 1$

3 times

Q. How many times do you need to divide 9 by 2 until it reaches 1 ?

$9/2 \rightarrow 4$

$4/2 \rightarrow 2$

$2/2 \rightarrow 1$

3 times

Q. 27 ?

$27/2 \rightarrow 13$

$13/2 \rightarrow 6$

$6/2 \rightarrow 3$

$3/2 \rightarrow 1$

4 times

N=		No. of times I need to divide
1		0
2	2/2 = 1	1
3	3/2 = 1	1
4	4/2 = 2 /2 = 1	2
5		2
6		2
8		3
13	13/2 = 6 /2 = 3 /2 = 1	3
16		4
⋮		⋮
27		4
30		4
31		4
32		5
100	100/2 = 50 /2 = 25 /2 = 12 /2 = 6 /2 = 3 /2 = 1	6

Powers of 2

Logarithm (log)

$$2^5 = 2*2*2*2*2 = 32$$

Q. How many times do you need to multiply 2 with itself to get 32?

or " " do you need to divide 32 by 2 to reach 1?

* log is reverse of exponentiation.

x^2

$5^2 = 25$

$\sqrt{25} = 5$

$$x^y = z$$

it means,

$$\log_x z = y$$

$$7^3 = 343$$

$$\log_7 343 = ? = 3$$

How to read this?

How many times do you need to multiply 7 by itself to reach 343?

$\log_{10} 10000 = 4$ (because $10^4 = 10000$)

$\log_2 1024 = 10$ (because $2^{10} = 1024$)

$\log_2 27 = 4.xx \approx 4$ (because $2^{4.xx} = 27$)

$2^4 = 16$

$2^5 = 32$

$\log 27$

$\log(N)$

default base is 2.

Q:

```
N > 0
i = N
while (i > 1)
{
    i = i/2
}
```

`i = i/2` → Integer division

Initial val of i	iteration no.	Final val. of i
$N = N/2^0$	1	$N/2 = N/2^1$
$N/2 = N/2^1$	2	$N/4 = N/2^2$
$N/4 = N/2^2$	3	$N/8 = N/2^3$
$N/8 = N/2^3$	4	$N/16 = N/2^4$
⋮	⋮	⋮
$N/2^{k-1}$	k	$N/2^k$
⋮	⋮	⋮
	$\log_2 N$	1

Loop breaks when final value = 1.

At what value of k, does final value become 1?

$$\frac{N}{2^k} = 1$$

$$\Rightarrow N = 2^k$$

Take log on both sides.

$$\Rightarrow \log_2(N) = \log_2(2^k)$$

$$\Rightarrow \boxed{\log_2 N = K}$$

How many times do you need to multiply 2 with itself to get 2^k?

K times

$\log_a a^x = x$ Property :)

Q.

```
for (i=1 ; i<N ; i=i*2)
{
    ==
}
```

Initial val i=	Iteration no.	final val i=
$1 = 2^0$	1	$2 = 2^1$
$2 = 2^1$	2	$4 = 2^2$
$4 = 2^2$	3	$8 = 2^3$
⋮	⋮	
2^{K-1}	K	2^K
⋮	⋮	
	$\log_2 N$	N

Loop breaks when final value becomes N.

Loop breaks.

$$2^K = N$$

$$\Rightarrow \quad K = \log_2 N$$

Q.

```
N >= 0
for (i=0 ; i<=N; i=i*2)
{
    ...
}
```

initial	Iteration	final
0	1	0
0	2	0
0	3	0
0	4	0
	⋮	

Infinite loop ☺

2, 4, 8, 16, 32, 64 ... Powers of 2. (GP)

3, 6, 12, 24, 48 ..... (GP)

Nested loops

Q.

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

	j	Iterations
i=1	{1,2,3,4....N}	N
i=2	{1,2,3,4...N}	N
i=3	"	N
i=4		N
⋮		
i=10	"	N

10 * N iterations

Q.

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

	j	Iterations
i=1	{1,2,3,4....N}	N
i=2	{1,2,3,4...N}	N
i=3	"	N
i=4		N
⋮		
i=N	"	N

N * N = N^2

Q :

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

	j	Iterations
i=1	[1, 2, 4, 8 .... N]	$\log_2 N$
i=2	[1, 2, 4, 8, 16 ... N]	$\log_2 N$
⋮	⋮	
i=N	"	$\log_2 N$

$N * \log_2 N$

Q :

```
for (int i=1 ; i<= 4;i++)
{
    for(int j=1 ; j<=i ;j++)
    {
        ≡
    }
}
```

	j	Iterations
i=1	1	1
i=2	{1,2}	2
i=3	{1,2,3}	3
i=4	{1,2,3,4}	4

$1+2+3+4$

$= \frac{4*5}{2}$

$= 10$

Sum of natural nos. from 1 to N = $\frac{N(N+1)}{2}$

Q.

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

	j	Iterations
i=1	1	1
i=2	{1,2}	2
i=3	{1,2,3}	3
i=4	{1,2,3,4}	4
⋮		⋮
i=N	{1,2,3,... N}	N

$$\frac{N(N+1)}{2}$$

Q.

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

	j	Iterations
i=1	[1, 2]	2
i=2	[1, 2, 3, 4]	4
i=3	[1, 2, 3, ... 8]	8
i=4	[1, 2, 3, 4 .... 16]	16
⋮		
i=N	$[1, 2, 3, 4, \ldots .. 2^N]$	2^N

$$2 + 4 + 8 + 16 + \ldots 2^N$$

$a = 2 \rightarrow$ (first term)

$r = 2$

$n = N$

$$S = \frac{a(r^n - 1)}{r - 1}$$

$$= 2(2^N - 1)$$

Break till 8:20 AM

	Sriram RamX	Madhuri MadX
	$100 \log N$	$N/10$
$N = 4$	$100 * \log 4$ $= 200$	$0.4 \approx 0$
$N = 64$	$100 * \log 64$ $= 800$	$6.4 \approx 6$
$N = 1,00,000$	$100 * \log_2 10^5$ $100 * 5 * \log_2 10$ $= 500 * 3 \ldots$ $= 1500$	$100000/10$ $= 10,000$

$\log_a x^b = b \log_a x$

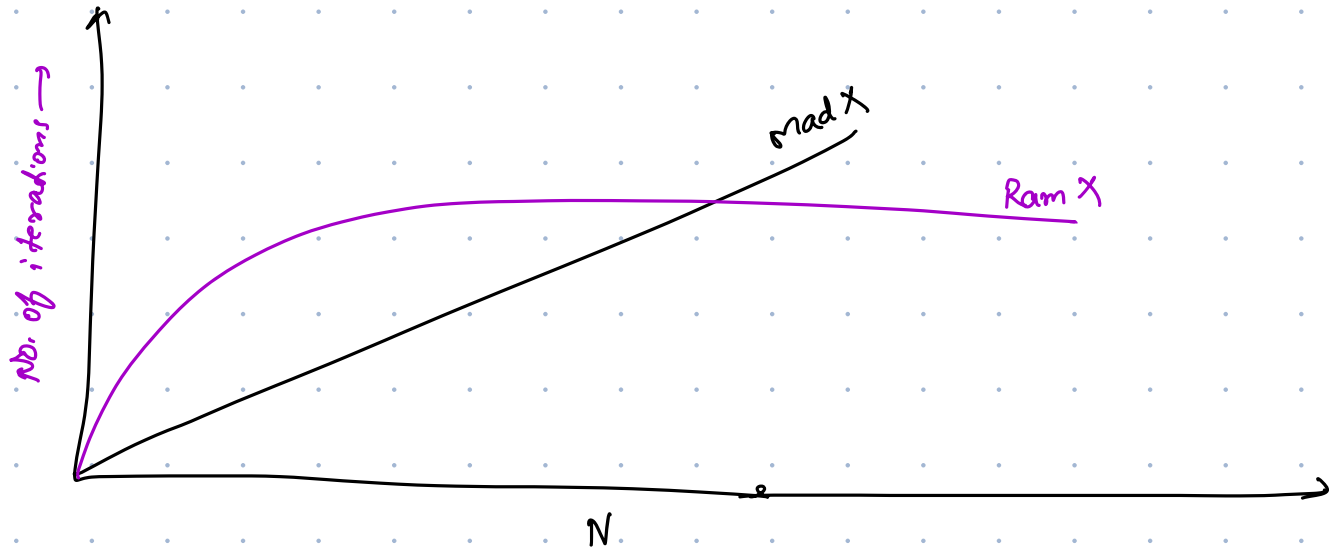
For larger N, RamX wins.
For smaller N, MadX wins.

Toda's world's data is huge.

Asymptotic analysis

↳ Analysis of an algorithm on large data.

$N \to \infty$



As $N \to \infty$, (or when N is very large),

Ram X algorithm performed better.

$100 \log N$, $N/10$, $4N^2+N$, $3N^3+N/10$, $4 * 2^N$

$100 \log N + N/200$

$(100 \log N)^2$

* Can't always plot a graph.

* Standard to help us easily figure out which algo is best?

"Big-O Notation"

Mathematical relation

$1 < \log N < \sqrt{N} < N < N \log N < N\sqrt{N} < N^2 < N^3 \ldots < 2^N < N!$

N^N

N=100

1	6	10	100	600	1000	10,000	10^6	2^{100}	100!

How to convert no. of iterations to Big-O Notation?

1. Get the no. of iterations. : $f(N)$
2. Neglect lower order terms
3. Ignore constant coefficients.

a+b : 2 terms

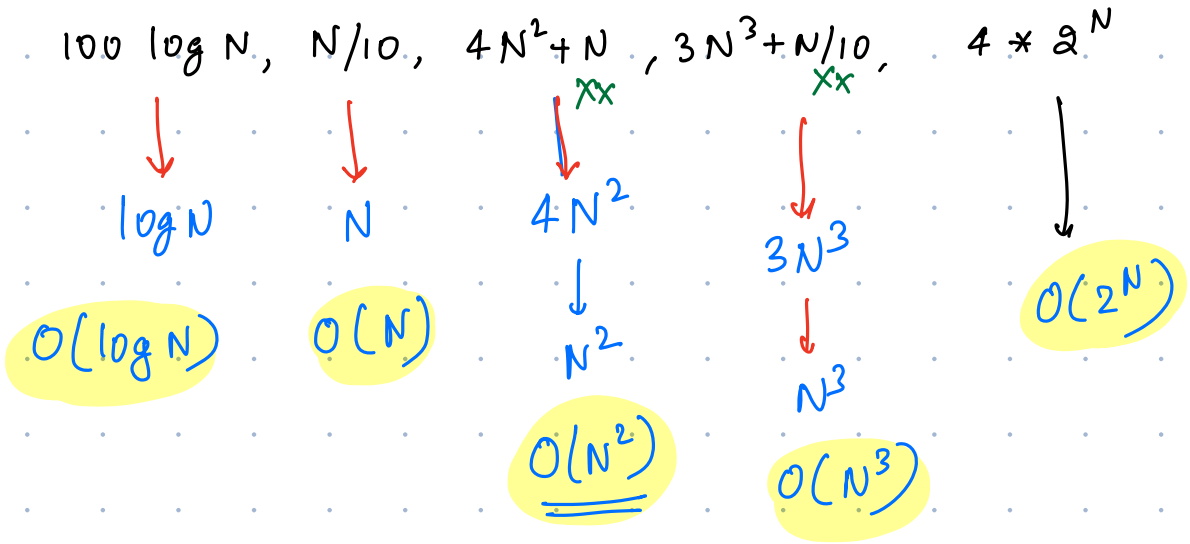
ab : 1 term

ab+cd : 2 terms

a+cd+e : 3 terms.

a-b : a+(-b) 2 terms

$100 \log N$, $N/10$, $4N^2+N$, $3N^3+N/10$, $4 * 2^N$



Standard names:

Best to Worst

$O(1)$	:	constant
$O(\log N)$	:	logarithmic
$O(N)$	:	linear
$O(N \log N)$	:	linear logarithmic
$O(N^2)$	:	Quadratic
$O(N^3)$	:	Cubic
$O(N^4)$		
$O(N^5)$	:	
$O(2^N)$	:	Exponential
$O(N!)$	:	Factorial

Polynomial

$4N+3N\log N+1$: 3 terms $O(N \log N)$

$4N\log N + 3N\sqrt{N} + 10^6$: 3 terms. $O(N\sqrt{N})$

const

* Why do we neglect lower order terms?

No. of iterations: $I(N) = N^2 + N$

$N = 1$

$I = 2$

↳ quadratic (N^2) : 1 — 50%

linear : 1 — 50%

$N = 10$ $\quad 10^2 + 10 = 110$

$I = 110$

quadr. = 100 — 91%

linear = 10 — 9%

$N = 100$

$I = 10^4 + 100$

quad. (N^2) : 10^4 — 99%

linear (N) : 100 — 1%

$$\frac{10^8}{10^{16} + 10^8} \%$$

$N = 10^8$

$I = 10^{16} + 10^8$

quad. : 10^{16} ≈ 99.99999999%

linear : 10^8 → 0.00000001%

* At larger N values,

quadratic (N^2) realizes that linear (N) is making ~0 contribution

Kick it out.

Issues with Big-O notation

Pranav	Faisal	
$100 \log N$	$50 \log N$	: According to Big-O Notation both are same. In reality, Faisal is better.

* Big-O Notation cannot help you compare very similar algorithms.

Pranav	faisal	
$2 N^2$	$10^{20} N$	: According to Big-O Notation, Faisal is better. In reality, Pranav might be better because of a large constant.

* While ignoring constant efficients, constants can be very big which might give inaccurate results.

9:20 AM

9:15 AM

* Time Limit Exceeded

Online compilers

fact : 1 GHz machines.

↓

process 10^9 instructions in 1 second.

```
int cnt(N){
    int c=0          +1
    for(i=1 ; i<=N ; i++)
    {                 1   1   1
        if(N%i==0)    +1
            c++       +1
    }
    return cnt.
}
```

4 instruction running in every iteration

↓

In a large code.

10 instructions run in every iteration.

Unitary method.

10^9 instructions in 1 second.

10 instruction in every iteration

$\Rightarrow$ 10^8 iterations run in every second.

* Online compilers run your code for 1 second.

* Goal:

Not more than 10^8 iterations should happen in your code.

or 10^7

Constraints

$1 <= N <= 10^3$

Your code has T.C: $O(N)$ → 1000 iterations. ✓

$O(N^2)$ → 10^6 iterations ✓

$O(N^3)$ → 10^9 iterations

XX TLE error.

$1 <= N <= 10^6$

$10^{6*2} = 10^{12}$ iterations

is $O(N^2)$ ok? XX

is $O(N)$ ok? ✓✓

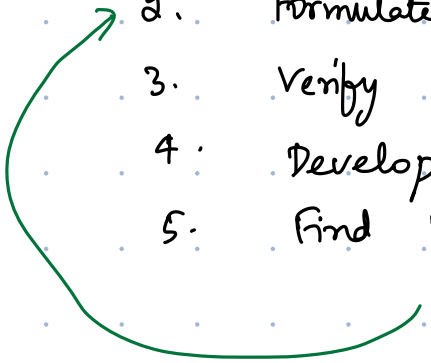
$1 <= N <= 10^{10}$

is $O(N)$ ok? XX

is $O(\log N)$ ok? ✓✓

How to approach a problem?

1. Read the question, understand the constraints.
   ↳ Find out what T.C. is OK!
2. Formulate the idea/logic.
3. Verify the correctness of logic. / Do a Dry Run.
4. Develop a pseudo-code.
5. Find T.C. of above.
   is your T.C. OK for your constraints?



6. Start writing code.

Properties of logs

$$\log_a a^x = x$$

$$\log_a a = 1$$

$$\log_a b^n = n * \log_a b$$

$$\log_a (xy) = \log_a x + \log_a y$$

$$\log_a \left(\frac{x}{y}\right) = \log_a x - \log_a y$$

Take some examples and practice.

$\log_2 10^N = N \log_2 10$

$= (3..) N$

$\log_e 10$

↓

xxx

You won't see this in DSA.

Byju's
RS Aggrawal
cengage.
Khan Academy
RD sharma
NCERT

```
for (int i=1; i<=n; i++)
{
    for (j=1; j<=3^i; j++)
    {
        i+j ==
    }
}
```

i=1 → j: [1,2,3] → 3

i=2 → j: 1 .... 3^2 [1 .... 9] → 9

i=3 → j: [1,2,1 .... 3^3] → 27

*3

$3+9+27+\dots 3^n$

$a=3$

$r=3$

$n=n$

$$S=\frac{a(r^n-1)}{r-1}$$

```
S=0
for (inf i=0; i<n=i=i*2)
    S=S+i
```

i=0

0*2=0

0*2=0

infinite loop

NO T.C.

Amazon

Sort on basis of price.

Search for Bose speakers.

* Every algo that you build should be fast.

Analyse your algo?

using a tool → Time complexity

$<= \sqrt{6}$

2 ✓

```
(i=1; i<=100; i++)
{
    s=s+i
}
```

100 iterations

$f(N) = 100$ → O(1)

H.W → Find the T.C. (in Big O Notation) of each question discussed today.

$2^5 = 32$

(exponential)

(i=1 ; i<N ; i=i*2)