Good Evening Everyone

1. Count of Factors
2. Sum of N Natural Numbers
3. GP and Sum of GP
4. Iterations
5. Comparing 2 algorithms
6. Big O Time Complexity
7. Space Complexity.

Problem 1 Count of Factors

What is a factor ?

we can say i is a factor N when i divides N completely

$N \% i == 0$

Given N, we have to count factors of N.

Quiz1.

Number of factors of the number 24?

1 2 3 4 6 8 12 24

$\Rightarrow$ 8.

Quiz2.

Number of factors of number 10.

1 2 5 10 $\Rightarrow$ 4.

Brute Force Solution

min factor = 1
max factor = The number itself.

```
function countofFactors (N) {

    fact_count = 0
    for (i -> 1 to N) {

        if (N%i == 0)
            fact_count = fact_count+1
        }
    }
    return fact_count

}
```

Servers have capability of running ~ 10^8 iterations in 1 second.

(N iterations)

N	Iteration	Execution Time
10^8	10^8 iteration	1 sec
10^9	10^9 iteration	10 sec
10^{18}	10^{18} iteration	$\approx$ 317 years.

$10^8 \rightarrow 1$ sec

$10^8 \times 10 \rightarrow 1 \times 10$ sec

$10^9 \rightarrow 10$ sec

Optimisation for Counting factors

$i * j = N$ {i and j are factors of N}

$j = N/i$ {i and N/i are factors of N}

$N = 24$

i	N/i
1	24
2	12
3	8
4	6
6	4
8	3
12	2
24	1

$N = 100$

i	N/i
1	100
2	50
4	25
5	20
10	10
20	5
25	4
50	2
100	1

$i <= N/i$

$\boxed{i * i <= N}$

$N = 10$.

1 2 5 10

```
function countFactors (N) {
    fac = 0;
    for (i -> 1 to i*i <= N)
        if (N%i == 0)
            if (i == (N/i) {
                fac = fac+1;
            }
            else {
                fac = fac+2
            }
        }
    }
    return fac
}
```

fac = ~~2~~ 4

$i <= N/i$

$i*i <= N$

$i <= \sqrt{N}$

$\sqrt{N}$ iteraton

Iterations

$N = 10^{18}$

$\sqrt{10^{18}} \Rightarrow 10^9$ iteraton $\rightarrow$ 10 seconds

follow up: given N, check if it's prime or not?

Quiz 3.

How many primes are there?

10 11 23 2 25 27 31

Maths Properties

$[a, b]$ → This range means that a and b are inclusive.

$[1, 10]$

(a, b) → This means a and b are excluded.

$(1, 10)$ →

Quizy.

How many numbers b/w $[3, 10]$?

3 4 5 6 7 8 9 10

lower

upper bound

[3, 10]

Quiz 5.

How many numbers are there in range [a, b]?

upper − lower + 1

10 − 3 + 1 = 8.

[2 5]

→ 2 3 4 5

5 − 2 + 1 = 4

3 + 1

Quiz 6.

S = 1 + 2 + 3 + 4 + 5 + 6 + ... 99 + 100.

S = 100 + 99 + 98 + 97 + 96 + 95 .... 2 + 1

2S = 101 + 101 + 101 + 101 + 101 + 101 ... 101 + 101

2S = 101 * 100

$S = \frac{101 * \cancel{100}\,50}{\cancel{2}} = 5050$.

Sum of 1st N natural number $\underline{N-1+2}$

$$S = 1 + 2 + 3 \; \cdots\cdots\cdots \; (N-1) + N$$
$$\underline{S = N + N-1 + (N-2) \; \cdots\cdots\cdots \; 2 + 1}$$
$$\underline{2S} \quad (N+1)+(N+1) + (N+1) \; \cdots\cdots\cdots \; (N+1) + (N+1)$$

$$2S = N * (N+1)$$

$$S = \frac{N * (N+1)}{2}$$

Geometric Progression

5 10 20 40 80 → The type of series in which the common ratio is same.

$10/5 \Rightarrow 2$
$20/10 \Rightarrow 2$
$40/20 \Rightarrow 2$
$80/40 \Rightarrow 2$

r.

$a \quad ar \quad ar^2 \quad ar^3 \quad ar^4 \ldots \ldots \quad ar^{n-1}$

$a=5$ 5 10 20 40 80

1 1 1 1 1 1 1

$r = 1$

$a * n$.

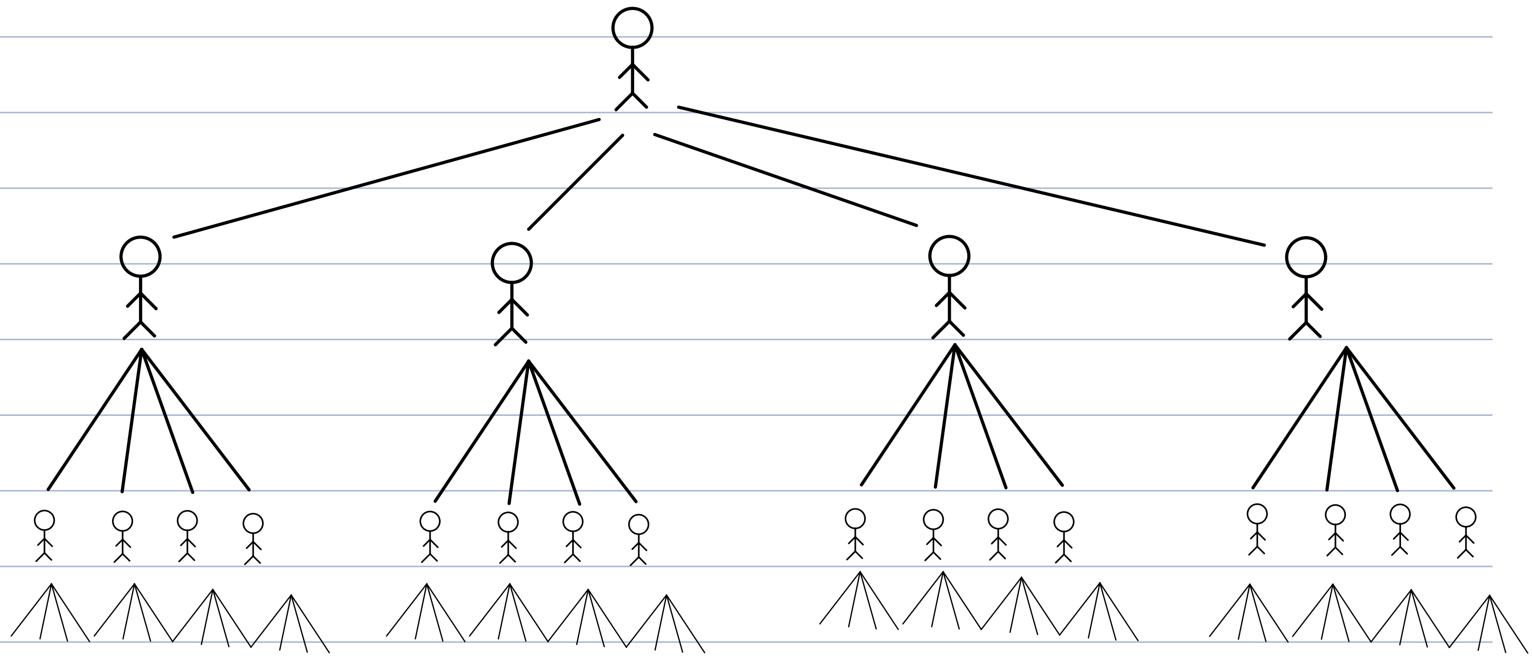
Sum of first n terms of GP.

$$\frac{a(r^n - 1)}{r - 1} \quad ; \quad r \neq 1.$$

$$\frac{5(2^5 - 1)}{2 - 1} \Rightarrow \frac{5 \times (32 - 1)}{1}$$

$\Rightarrow 31 \times 5 \Rightarrow 155$

Real Life Examples of GP



1   4   16   64

$a \quad ar^1 \quad ar^2 \quad ar^3 \quad\quad ar^{(7)}$

1st   2nd   3rd   4th   8th

$a = 1, \quad r = 4, \quad n = 8$

$1 * 4^{(7)} \Rightarrow \underline{16384}$

Iteration ?

The number of Times a loop runs is known as iteration.

Quiz 7.

```
for (i -> 1 to N) {
    if (i==N)
        break;
}
```

→ N iteration

Quiz 8:

```
for (i-> 0 to 100) {
    s = s + i + i^2 ;
}
```

$\Rightarrow \boxed{101}$

[0 - 100]

100 - 0 + 1

= 101 .

Quiz 9 .

```
func() {
    for (i-> 1 to N) {
        if (i%2 == 0) {
            print(i);
        }
    }
    for (j-> 1 to M) {
        if (j%2 == 0) {
            print(j);
        }
    }
}
```

$\boxed{N+M}$

How to compare two algorithms ?

Contest - Sort the given array (Ankit Nikita)

arr → {3 2 6 8 1} → {1 2 3 6 8}

Ankit	Nikita.
Algo 1	Algo 2
15 sec	10 sec.
↓	↓
Windows XP	MAC.

Time is not suitable for comparing algos.

↳ Number of iteration to compare algorithms

Asymptotic Analysis of Algorithms

OR

Big(O) simply means analyzing performance of algorithms on larger inputs.

Calculation of Big(O).

- Calculate Iterations based on Input Size.
- Ignore lower Order Terms.
- Ignore Constant Cofficients.

Iterations → $100 * \log_2 N + \cancel{2}$

$100 \log_2 N$

$O(\text{Iteration}) \Rightarrow O(\log_2 N)$

$N/10$

$\frac{1}{10} \times N$

$\Rightarrow O(N)$

$N/10$

$\boxed{\cancel{\frac{1}{10}}} \times N$

$\cancel{100} \log_2 N + \cancel{2}$

$O(\log_2 N)$

Iterations : $4N^2 + 3N + 1$.

1. Neglect lower order terms $\Rightarrow 4N^2$
2. Neglect cofficient $\Rightarrow N^2$

$$O(N^2)$$

$$\cancel{5N^2} + \cancel{3N^2} + 2N^3 + \cancel{1}$$

$$\Rightarrow 2N^3$$

$$N^3 \Rightarrow O(N^3)$$

Comparison Order

$\log_2(N) < sqrt(N) < N < N\log(N) < N\,sqrt(N) < N^2 < N^3 < 2^N < N! < N^N$

$N = 36$

$5 < 6 < 36 < 36*5 < 36*6 < 36^2 < 36^3 < 2^{36}$

$< 36! < 36^{36}$

$$4N^2 + \cancel{3N + 6\,sqrt(N) + 9\log_2(N) + 10}$$

$$\cancel{4}N^2$$

$$\Rightarrow O(N^2).$$

Quiz 10:

$$F(N) = \cancel{4N} + 3N * \log(N) \cancel{+ 1} \Rightarrow 3N \log N$$

$$O(F(N)) = O(N \log_2 N)$$

Quiz 11:

$$F(N) = \cancel{4N \log(N)} + 3N * \text{sqrt}(N) \cancel{+ 10^6}$$

$$3N\,\text{sqrt}(N)$$

$$O(N * \underline{\text{sqrt}(N)})$$

Why do we neglect Lower Order Terms?

Iteration : $N^2 + 10N$

N	Total Iterations (N^2+10N)	Lower Order Term ($10N$)	% of Lower Order Term in Total $(10N/(N^2+10N)) * 100$
10	200	100	50%
100	$10^4 + 10^3$	10^3	$\approx$ 10 %
10000	$10^8 + 10^5$	10^5	$\approx$ 0.1 %

As input increases, impact of lower order term decreases.

Why do we neglect Constant Cofficient

Algo 1 (Nikhil)	Algo 2 (Pooja)	Winner for larger I/P
$10 * \log_2 N$	N	Nikhil
$100 * \log_2 N$	N	Nikhil
$9N$	N^2	Nikhil
$10N$	$N^2/10$	Nikhil
$N * \log_2 N$	$100 * N$	Pooja

→

Issues with Big (O)

1. We cannot always say that one algorithm will always be better than other algorithm.

↳ Algo 1 → $(10^3 N)$ → $O(N)$

↳ Algo 2 → (N^2) → $O(N^2)$

Input Size (N)	Algo 1 10^3N	Algo 2 N^2	Optimised
$N=10$	10^4	10^2	Algo 2
$N=100$	10^5	10^4	Algo 2
$N=10^3$	10^6	10^6	Both are same
$N=10^3+1$	$(10^3)(10^3+1)$	$(10^3+1)(10^3+1)$	Algo 1.
$N=10^4$	10^7	10^8	Algo 1

for all the inputs post 1000 Algo 1 better but before that Algo 2 performs better.

2.

If 2 algorithms have same higher order terms, then Big O is not capable to identify algorithm with higher iterations

<u>Example</u>

① for (i → 1 to N) {

if (i % 2 != 0) {

c = c + 1 ;

}

}

(N)

<u>O (N)</u>.

② for (i goes from 1 to N , and i gets incremented by 2 every iteration) {

c = c+1 ;

}

<u>N/2</u>.

$\cancel{\frac{1}{2}} \times N$

=> O(N)