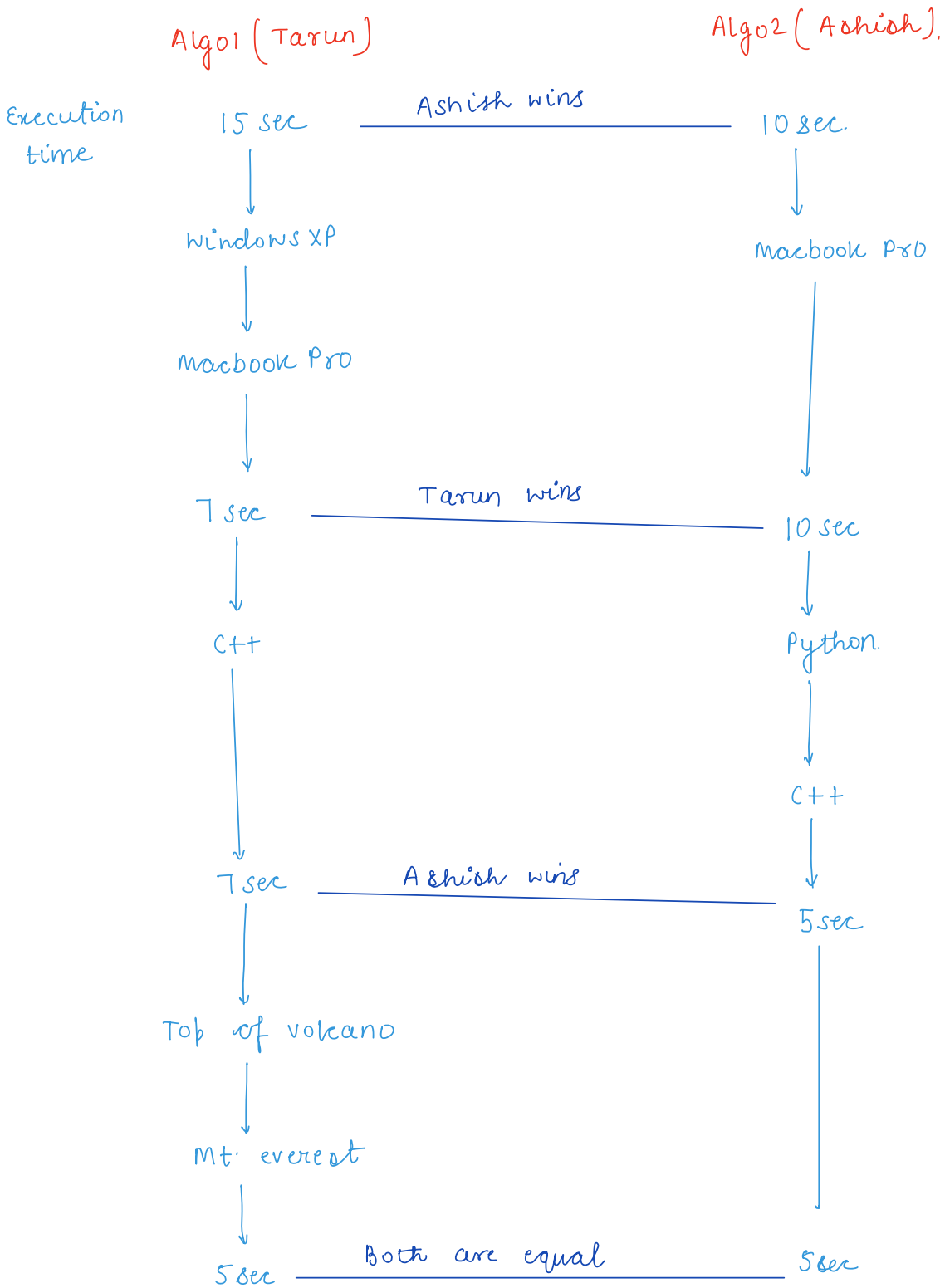
Lecture 3 : Time complexity 2.

Agenda:

- Comparing Algos
- Big O notation
- Space complexity
- TLE.

Contest :- Given a no. n, find count of factors



Execution time :- It depends on many external factors. Hence, we generally do not compare our algo using execution time.

Comparing Algos

1.) Iterations. :- do not depend on any external factor.

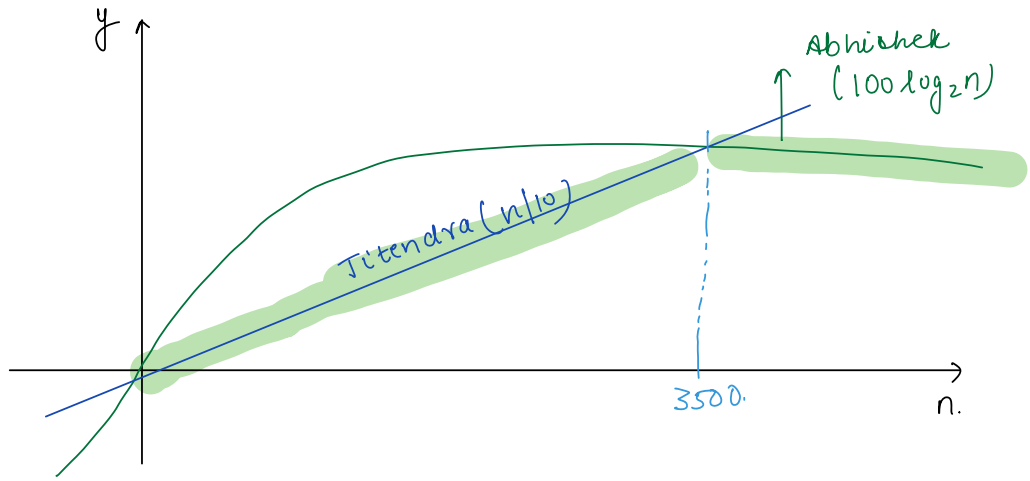
```
for(i=0; i<10; i++){
    print("Abhishek");
}
```

itr = 10. (xp).
= 10 (macbook)
= 10 C++

2.) Graph

Algo1 (Abhishek) Algo2 (Jitendra)

Itr :- $100 \log_2 n$. $n/10$.



Observation:

	itr	faster
Till N <= 3500	Jitendra < Abhishek	Jitendra
N > 3500	Abhishek > Jitendra	Abhishek

India vs Pak :- 10 million

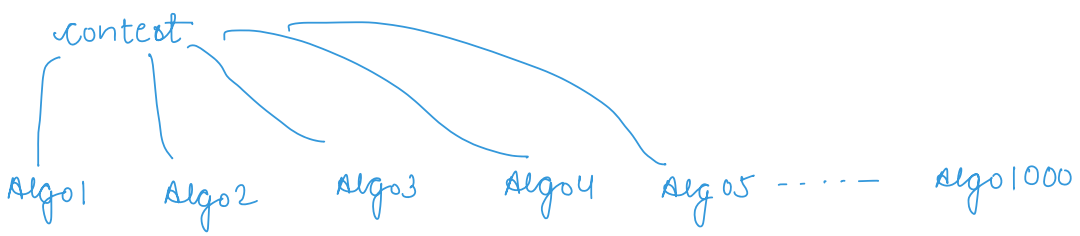
Google search :- million results | sec.

Baby shark :- 10.84 billion

Data :- inc^r for real life app^n.

Conclusion: Pick any algo on basis of larger inputs

Question:



How to compare multiple algo?

Ans: **Asymptotic analysis**

- Analysis performance of an algorithm for larger ips.
- **Big (O)** notation is used to do the asymptotic analysis.

Calculation of Big-O :-

- Calculate no of itr
- Neglect lower terms
  (or)
  Take higher terms
- Ignore constant co-efficients

Ex:

	Algo1 (Bala)	Algo2 (Proteeth)
Itr:	$100 \log_2 n$	$n/10$
Big-O	$O(\log_2 n)$	$O(n)$

Bala is better than Proteeth.

$\{O(\log_2 n)\}$ $\{O(n)\}$

Q Why do we neglect lower terms?

Algo(Priyanka) :- Itr = $n^2 + 10n$.

Big O = $O(n^2)$.

Lower term :- $10n$. { Ignored }

Higher term :- n^2.

Input size	Itr. $[n^2+10n]$	% of lower order contribution in total itr
$n = 10$	200	Lower order contribution = $10n$ = 100. % = $\frac{100}{200} * 100$ = 50 %
$n = 100$	$10^4 + 10*100$ = $10^4 + 10^3$ = 11000	Lower order :- $10n = 10*100 = 1000$ % = $\frac{1000 \times 100}{11000} \simeq$ 9 %
$n = 10^4$.	$10^8 + 10*10^4$ = $10^8 + 10^5$.	Lower order :- $10n = 10*10^4 = 10^5$. % = $\frac{10^5 \times 100}{10^8 + 10^5} \simeq$ 0.1 %

Conclusion: As ip size inc^r, contribution of lower term decreases

H|w :- Replicate same ex for why do we ignore constant co-efficients.

Issues :-

	Algo1 (Sharat)	Algo2 (manjunath)
Itr :-	$10^3 N$.	N^2.
Big O =	$O(n)$.	$O(n^2)$.

Sharat wins.

Input size	Sharat $(10^3 n)$	Manjunath (n^2)	
n=10	$10^3 * 10 = 10^4$.	$10^2 = 100$.	manju.
n=100	$10^3 * 100 = 10^5$.	$(10^2)^2 = 10^4$.	manju
n=1000	$10^3 * 10^3 = 10^6$	$(10^3)^2 = 10^6$	equal.
n=1001.	1000 * 1001.	1001 * 1001.	Sharat
⋮	⋮	⋮	⋮

for ip $\geq$ 1000 , Sharat Better.

Claim : When we compare two algo using Big O.

Any algo will always be better after a threshold point { n=1000 }

Issue 2:

	Algo1 (Pratik)	Algo2 (Dinesh)
Itr.	$2n^2 + 4n$.	$3n^2$.
Bigo	$O(n^2)$ — Both are equal.	$O(n^2)$.

$2n^2 + 4n$ $\qquad$ $3n^2 = 2n^2 + n^2$.

→ Acc to Big O, both algo are equal.

→ But logically, they are not [Pratik is better]

conclusion: Big O notation is perfectly fine for larger ips. But at the end, itr will always yeild correct comparisons

final conclusion

Since we only deal with larger ips for real life appn, we should go with Big (O) notation.

Best case and worst case

Code: Search an el = k from an array

```
boolean search(int[] arr, int k){
    for(i=0; i<arr.length; i++){        // n.   # itr = n.
        if(arr[i]==k){
            return true;
        }
    }
    return false;
}
```

Ex1: arr[] = {1, 2, 3, 4, 5, 6 ..... 100} [Best case]

k = 1.

itr = 1.

Ex2: arr[] = {1, 2, 3, 4, 5, 6 ..... 100} [Worst case]

k = 100

itr = 100(n).

Conclusion: Always write the algo keeping worst case scenario in mind

Break :- 8:38 AM

Space complexity

```
fun( int n) {
4B   int x = n.;
4B   int y = x+x;
8B.  long z = x+y;
}
```

4B → n=10 [4B]; n=1000 [4B]

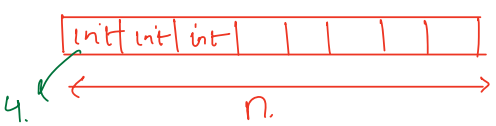
TC :- O(1).

Memory :- (4 + 4 + 4 + 8) B = 20 B.

SC :- O(1)

[n = 10 → SC: O(1).
n = 1000 → SC: O(1)
n = 10^9 → SC: O(1)]

```
fun( int n) {        // 4B
4B   int x = n.;
4B   int y = x+x;
8B.  long z = x+y;
     int[] arr = new int[n];
}
```



Memory :- 20B + 4*n.

SC :- O(n). [n = 10 → SC: O(10)
n = 100 → SC: O(100)]

TC :- O(1)

```
fun( int n) {
4B   int x = n.;
4B   int y = x+x;
8B.  long z = x+y;
4*n. int[] arr = new int[n];
8*n^2 long[][] mat = new long[n][n];
}
```

Memory :- 20B + $8n^2$ + 4*n.

SC :- $O(n^2)$

Space complexity It is amt of space additionally used by algo, other than the ip size for necessary computation.

Ex1

```
int maxArray(int[] arr, int n){        // not part of S.C.
    int max = arr[0];                  // 4B
    for(i=1; i<arr.length; i++){       // 4B
        max = Math.max(max, arr[i]);
    }
    return max;
}
```

SC: O(1)

TC: O(n)

Ex2:

```
void fun(int[] arr){
    int n = arr.length;                // 4
    int[] temp = new int[n];           // 4*n
    ...
}
```

SC: $4 + 4*n = O(n)$

Ex3

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

SC: O(1)

int i=1; — &101 [?] 2

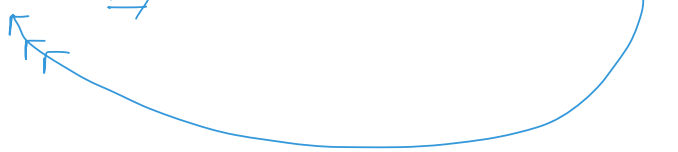
i++ → i = i+1

Prefer TC over SC [TC is closely associated **user experience**]

TLE :- Time Limit Exceeded

Dinesh ⟶ Amazon ⟶ hiring challenge ⟶ 2Q ⟶ 1hr.

Q1 ⟶ idea ⟹ code ⟹ submit ⟹ TLE



Remarkable idea

Without writing a single line of code, can we say whether we can get TLE or not?

Online editors

code servers ⟶ P.S. ⟶ 1 Ghz ⟶ 10^9 instructions / sec

constraint :- code should be executed in 1 sec.

Observation 1: At max, our code can have 10^9 instructions

variables, operators

```
int countFactors(int n){
    int c = 0;
    for(int i=1; i<=n; i++){
        if(n%i==0){
            c++;
        }
    }
    return c;
}
```

1 2 3 4 7 6

itr = n.

1 itr ≃ 7 instructions.

Total instructions = 7n

Approximation 1:

1 itr ≃ 10 instructions.

Our code can contain = 10^9 instructions.

= 10^8 * 10 instructions

= 10^8 * 1 itr

Our code can contain = 10^8 itr.

Approximation 2:

1 itr = 100 instructions

Our code can contain = 10^9 instructions.

= 10^7 * 100 instructions

= 10^7 * 1 itr

Our code can contain = 10^7 itr.

In general ⟶ code itr → $[10^7 - 10^8]$ itr.

Q Read the question.

constraints — $1 <= n <= 10^5$

Ideal:- $O(n^2) \simeq 10^{10}$ [TLE]

Idea2:- $O(n) \simeq 10^5$. [Works].

$O(n \log n) \simeq$ Works $= 10^5 * \log_2 10^5 < 10^8$.

Q constraint $n <= 10^3$

Ideal: $O(n^2) = 10^6$ [Works]

Thankyou :)

Doubts (Sieve of eratosthenes)

boolean checkPrime (int n) {

$n = 10^9$.

for (i = 2; i <= n/2; i++) {

}

}

iter = n/2.

iter = $10^9/2$ = $0.5 * 10^9$.

= $5 * 10^8$.

$$n + \left[\frac{n}{2} + \frac{n}{4} + \frac{n}{8} - \cdots \cdots \frac{n}{2^{\log_2 n}} \right]$$

$$n + n\left[\frac{1}{2^1} + \frac{1}{4} + \frac{1}{8} + \cdots \cdots \frac{1}{2^{\log_2 n}} \right]$$

2^1 2^2 2^3 ... $2^{\log_2 n}$

$a = \frac{1}{2}$.

$r = \frac{1}{2}$.

$n = \log_2 n$.

$$\text{sum} = \frac{a(r^n - 1)}{r - 1} = \frac{\frac{1}{2}\left[\left(\frac{1}{2}\right)^{\log_2 n} - 1 \right]}{\frac{1}{2} - 1}$$

$$= \frac{\frac{1}{2}\left[\frac{1^{\log_2 n}}{2^{\log_2 n}} - 1 \right]}{-\frac{1}{2}}$$

$$= 1 - \frac{1}{2^{\log_2 n}}$$

$$= \boxed{1 - \frac{1}{n}}$$

$$n + n\left[1 - \frac{1}{n}\right]$$

$$n + n - 1$$

$$\boxed{2n - 1}$$ Ans