I always → I joined 15 min late, can you repeat everything?

Start 7:05AM

Unfortunately No! because of time. I can quickly show notes so you take a screenshot & later you can watch the recording to catch up.

Interview tip

prepare two versions of your work/experience summary

short → 3 min. For coding & system design intro is just ice breaker. Keep it brief.

long → 10–20 min for hiring manager project deep dive

Topics:
- Comparing two algos:
  - @ using execution
  - (b) using iteration
- Why big O notation
- break?
  - Why omit lower terms
  - Why omit constant coeff.
- Issues with big O
  - large constants
  - same O
- SC
- TLE

→ I'm an industry expert. we benchmark programs by runtime execution. we do not need O notation! ← Is the troll right?

Storry

Sorting Algo Competition

Algo1 & Bob

15 sec

↓

win PC 2010

↓ ✗

Macbook Pro M2

7 sec ✓

↓

c++ (Intel Compiler)

↓

7sec

↓

Hawaii, mauna

Volcano

50+ c

↓

north pole ✗

5 sec

Algo2 & Alice

10 sec ✓

Macbook Pro M2

↓

10 sec

↓

python

✗

c++ (Intel Compiler)

↓

5ces ✓

north pole

↓

5sec

=

⋮

Conclusions Execution time depends on so many params. we cannot merely use it for comparing the effeciency of algorithms.

story

Pat

Algo1 ✓ iter#: $10 \log n$

$10 \times \log 50 = 5.$

$10 \times \log_{10} 1024 = 30$

mat

Algo2 ✓ iter.#: $n/10$

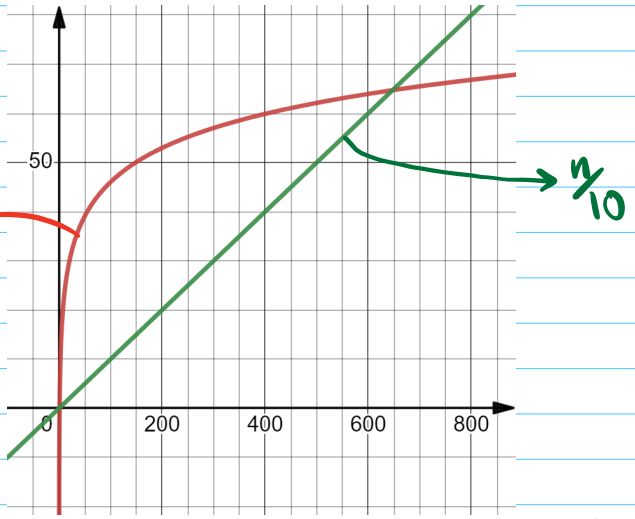
$\frac{50}{10} = 5$

$\frac{1024}{10} = 102.4$

$n = 50$

$n = 1024$

https://www.desmos.com/calculator/dfs4y5xkfw



tim sort

↓

python

<40

Algo1 Algo2

iters we focus iter.#

```
for(i=0; i<=n; i++){
    ...
}
```

why analysis for large numbers.

$n \to +\infty$

why?

we are at the age of big data

- search result in Google
- online streaming
- # of users of social network

CLRS
Intro Algo Book

Asymptotic Analysis of Algorithms

$n \to \infty$

Analysis of Algorithms for large inputs.

* https://www.scaler.com/topics/asymptotic-notations/

tight: O, Ω, Θ — loose: o, ω, θ

O → upper bound; Ω → lower bound; Θ → both

what about?
- best
- worst
- average

↓ don't be confused be aware of some web sites with wrong Info

Big O notation:

1) # iterations based on input size → why? → runtime is not enough iter is static and capable
2) omit all lower terms → why?
3) omit all constants coeff. → why?

---

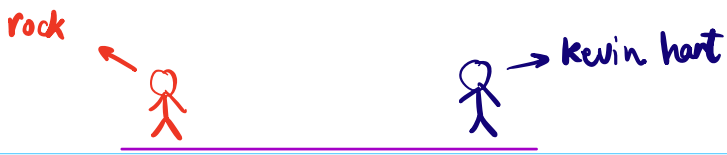
story

https://www.desmos.com/calculator/4ihihjoslh

$n^2 + 10n$

$O(n^2)$

input	iteration	% of "10n" → term
$n = 10$	$10^2 + 10 \times 10 = 200$	$\frac{10 \times 10}{200} \times 100\% = 50\%$
$n = 100 = 10^2$	$(10^2)^2 + 10(10^2) = 10^4 + 10^3$	$\frac{10^3}{10^4 + 10^3} \times 100\% \simeq 10\%$
$n = 10^4$	$10^8 + 10^5$	$\frac{10^5}{10^8 + 10^5} \times 100 \simeq 0.1\%$



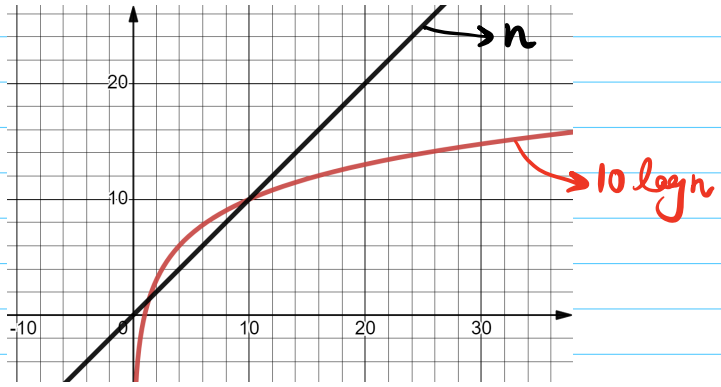
story

iterations

	$10 \log(n)$ ✓	n
$O(\log n)$	$100 \log(n)$ ✓	n
	$1000 \log(n)$ ✓	$n/10$
	$n \times \log(n)$	$100 \times n$ ✓

$O(n)$

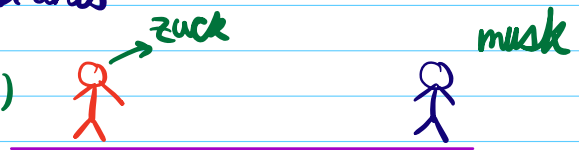
https://www.desmos.com/calculator/tjqkvuak17



Issues with big O's

1- large constants

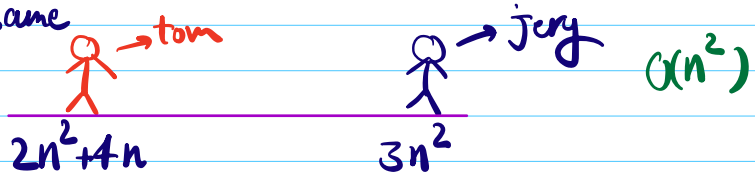
story



iter#.	1000 n	n^2
n=10	10^4	10^2 ✓
n=100	10^5	10^4 ✓
$n=10^3$	10^6 ~	10^6
$n=10^3+1$	$10^3(10^3+1)$ ✓	$(10^3+1)\times(10^3+1)$
	↓ ✓	

2- when O is same

story



iter#.

↓

we cannot clearly compare two algos only by O notation

n →

a[3 7 2 -10 12 11 17 ... -12 12 13]

```
bool find(int[] a, int k){
    for(i=0; i<a.Len; i++){
        if(a[i]==k){
            ret true;
        }
    }
    ret false
}
```

k = 13 | -3 | -13

- best → 1 iteration
- worst → n iterations
- average → n/2

TC: Time Complexity SC: Space Complexity

ex1

```
int func1(int n){
    int x = n        4B
    int y = x+n      4B
    long z = 2xy+x   8B
    ret z
}
```

4+4+8 = 16 B

O(16) = O(1) → Constant SC

16 x n^0 = O(n^0) = O(1)

ex2

```
int func1(int n){
    int x = n        4B
    int y = x+n      4B
    long z = 2xy+x   8B
    int a[] = new array[n]   4xn
    ret z
}
```

n = 10 → 16 B + 4x10 B = 56 B

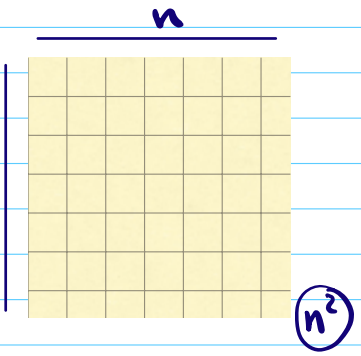
n = 10^3 → 16 B + 4x10^3 B = 4016 B

SC: O(16+4n)

SC: O(n) ✓

ex3
```
int func1(int n){
    int x = n          // +4B
    int y = x+n        // +4B
    long z = 2xy+x     // 8B
    int a[][] = new array[n][n]
    ret z
}
```

SC: $O(4+4+8+4n^2)$
SC: $O(n^2)$



ex4
```
bool find(int[] a, int k){      // a.Len is n
    for(i=0; i<a.Len; i++){
        if(a[i]==k){
            ret true;
        }
    }
    ret false
}
```

$O(4B) = O(1)$

SC: $O(1)$?
$O(n)$?

input data → algorithm → output data

SC

FAQ

- why not input data + output data — auxuliry space

- I saw a reputable book/website consider them.

- what about interview, SC analysis?

- I am still not convinced. let's fight!! JK

- Can I ask this doubt 1000 times repeated till the end of advance DSA? yes! no problem.

a.Len = n

ex5
```
int func5(int a[]){
    int PS[] = new int[a.Len]
    PS[0] = a[0]
    for(i=1; i<a.Len; i++){
        PS[i] = PS[i-1] + a[i]
    }
    ret PS[a.Len-1]
}
```

SC: O(n)

4 × 1000

$O(n^2)$

4×10^6 Bytes

4×10^3 KB

4 MB

MLE

TLE : Time Limited Exceeded

story → shubham Google interview

TLE

How to estimate runtime?

Assumption1: 1GHz CPU → 10^9 $\frac{\text{operation}}{\text{sec}}$ | 1 sec / 2 sec

Assumption2: → 10 — 100 x iteration

GByte

```
for(i ... ){
    int x = i + 1 → 3
    = = = → 5
    = 2
}
```

10

$O(n^2)$ $10^5 \leftarrow n$

$(10^5)^2 \times 100 = 10^{12}$

Constraints

$O(n^2) \longleftrightarrow 10^4$

$O(n^2)$ $10^2 \leftarrow n$

$10^4 \times 100 = 10^6$

How to approach a problems check the Constraints :

- Read the Question and Constraints carefully.
- Formulate an Idea or Logic.
- Verify the Correctness of the Logic.
- Mentally develop a Pseudocode or rough Idea of Loops.
- Determine the Time Complexity based on the Pseudocode.
- Assess if the time complexity is feasible and won't result in Time Limit Exceeded (TLE) errors.
- Re-evaluate the Idea/Logic if the time constraints are not met; otherwise, proceed.
- Code the idea if it is deemed feasible.

$O(1)$ → Contant TC

$O(n)$ → Linear TC

$O(\log n)$ → logarithmic

$O(n^2), O(n^3), O(n^4) \ldots$ → Polynomial TC

$O(2^n), O(3^n), \ldots O(10^n), \ldots$ → exponential TC

$O(n!)$ → factorial TC

TC vs SP

iteration

memory consumption (usually the largest array/matrix I define part of solution)

both are defind based on input size n.
it is the same n.