Lecture_1 01.18.23

min = 10 max = 0

For every 2 you would do 3 comparisons.

5
8
3
1
2
4
9
6

8 vs 5 8 vs 0 5 vs 10

min = 5 max = 8

Assume you just have finite number of primes.

$P_1 - P_n$

$P_1 \cdot P_2 \cdot P_3 \cdot P_4 \cdots P_n = X$ → I would get a big giant number.

The number would be divisible by every prime number that was. because we're assuming there's a finite number of primes.

$X+1$

therefore you have a contradiction in the fact that some integer cannot exist.

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

$1+1+2+3+5+8+13+21$

Fibonacci sequence

In Appendix A

$n \% 7 = 5$

$(n+7) \% 7 = 5$

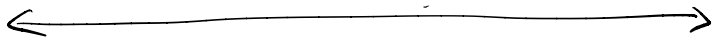
$(n+7+7) \% 7 = 5$

$(n+7k) \% 7 = 5$

$(-2+7) \% 7 = 5$

$5 \% 7 = 5$

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12 ⋯
6	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5 ⋯



$-2 \% 7 = 5$

$$\log_b n = \frac{\log_e n}{\log_e b}$$

the number of operations needed for any algorithm is not going to be less than the basic number of operations required to solve the problem.

Problem — Algorithms — Implementation

$f_p(n) \leq f_A(n) \leq f_I(n)$

$f_p(n)$: call it f problem

$f_A(n)$: the function of the Algorithms.

Why: the number of operations necessary to grow sometimes.

because Implementation might have less efficiency than this. and then Algorithms less efficiency than problem absolutely required for solving problem.

If you know your problem required $n \log n$ steps. then you can not find out an algorithm that is less than $n \log n$ steps. unless you have a bug

Problem $\leq$ Problem $\leq$ Problem

Best case Average case Worse case.

Algorithms $\leq$ Algorithms $\leq$ Algorithms

Best case Average case. Worse Case

Implementation $\leq$ Implementation $\leq$ Implementation.

Best case Average case Worse case.

Algorithms

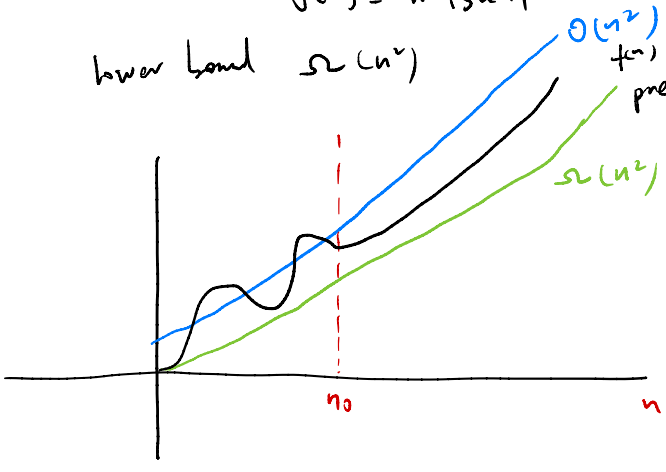
Average cost → represents the number of operations

$f(n) =$

upper bound $O(n^2)$

$f(n) = n^2 + 3n + 1$

lower bound $\Omega(n^2)$



pretend it is what the function looks like.

$\Omega(n^2)$ lower bound.

n^2+3n+1

$\Omega(n^2)$

$O(n^2)$

$\Theta(n^2)$

after n_0. $O(n^2) \in f(n) \leq \Omega(n^2)$

수학적 증명.

$$0 < c_1 n^2 \leq n^2 + 3n + 1 \leq c_2 n^2, \quad \forall n \geq n_0$$

$$0 < c_1 \frac{n^2}{n^2} \leq \frac{n^2}{n^2} + \frac{3n}{n^2} + \frac{1}{n^2} \leq c_2 \frac{n^2}{n^2}, \quad \forall n \geq n_0$$

$$0 < c_1 \leq 1 + \frac{3}{n} + \frac{1}{n^2} \leq c_2, \quad \forall n > n_0$$

when $n \to \infty$ the $f(n)$ gets smaller and it approaches 1.

$$\lim_{n \to \infty} 1 + \frac{3}{n} + \frac{1}{n^2} = 1$$

$c_1 = 1$

$c_2 = 1.31$ if $n_0 = 10$

algorithms little o can not be tight

Average case. Worst case

$f(n) =$

$O(n^n)$	$o(n^n)$	$O(n^n)$	$o(n^n)$
$O(n!)$	$o(n!)$	$O(n^4)$	$o(n^4)$
$O(n^8)$	$o(n^8)$	$O(n^3)$	~~$o(n^3)$~~
$O(n^2)$	~~$o(n^2)$~~	$\Theta(n^3)$: $\Omega(n^3)$ $f(n) = n^3 + 3n^2 + 3n + 8$	$\omega(n^3)$
$\Theta(n^2)$: $f(n) = n^2+3n+1$		$\Omega(n^2)$	$\omega(n^2)$
$\Omega(n^2)$	~~$\omega(n^2)$~~	$\Omega(n)$	$\omega(n)$
$\Omega(\sqrt{n})$	$\omega(\sqrt{n})$	$\Omega(\log n)$	$\omega(\log n)$
$\Omega(\log n)$	$\omega(\log n)$	$\Omega(1)$	$\omega(1)$
$\Omega(1)$	$\omega(1)$		

Big O
the bound can be tight but it doesn't have to be tight.

Best case

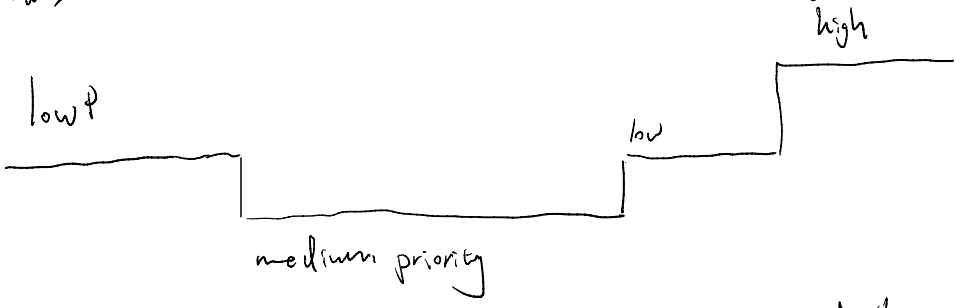
generally we don't care about the best case. you can hack the best case

I can define my input to say pass me an array of integers and a boolean. and if the boolean is true it tells me that the integers are already sorted and all I have to do is return it.

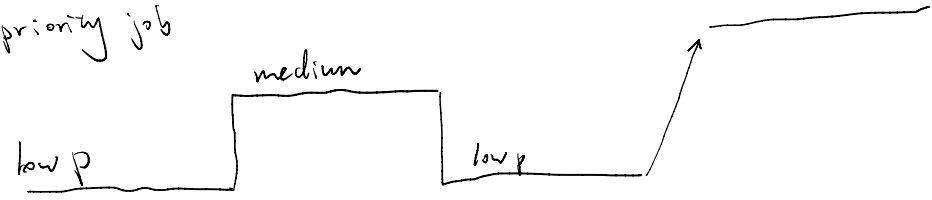
Sometimes you worry about best case in real time systems because in real-time systems particularly if you're doing fixed priority systems. Jobs that runs too fast can cause you to miss a deadline and that happens because other things get started or maybe that job runs too fast which kicks off another job

of high priority and it's in a middle priority job didn't get to run because the look that finished. so in fixed priority I'll describe that real quickly.

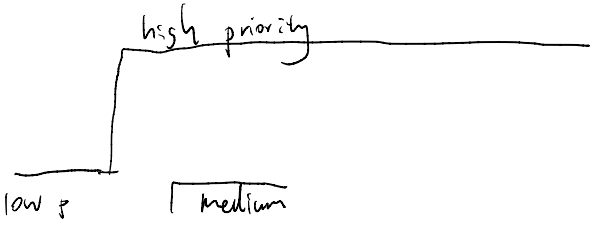
Let's say I have a low priority job and normally it runs like this and then they get suspended with a medium priority and the medium priority runs and then the low finishes and the low triggers a high



the medium came in and actually low priority and then low priority job gets pended by the medium finishes and then the low priority runs. and the low priority finishes the low priority then triggers a high priority job



if low priority job runs too fast what happens is your low priority job runs fast and finishes too fast it's going to trigger my high priority job here. when my medium priority job is ready to run, it doesn't get to run, it starved because the high priority is running instead.



So the fact that my low priority job finished too fast it caused my high priority job to start running and it starts with the medium priority job that normally work.

In some case you care about your best case because you don't want your best case to be good. In fact a lot of times for real-time systems you want all three of these cases to be as close to the same as possible so that you have a deterministic running time. But I'm going to say in this particular case there our best case will can be described as $\sqrt{n}+3$

Best case

$O(n^n)$

$O(n!)$

$O(n^4)$

$O(n^3)$

$O(n^2)$

$O(\sqrt{n})$

$\sqrt{n}+3$

$\Omega(\sqrt{n})$

$\Omega(\log n)$

$\Omega(1)$

$\Theta(\sqrt{n})$

Average case.

$f(n) =$

$O(n^n)$ $\quad \theta(n^n)$

$O(n!)$ $\quad o(n!)$

$O(n^8)$ $\quad o(n^8)$

$O(n^2)$ $\quad \cancel{o(n^2)}$

$f(n) = n^2+3n+1$

$\Omega(n^2)$ $\quad \cancel{\omega(n^2)}$

$\Omega(\sqrt{n})$ $\quad \omega(\sqrt{n})$

$\Omega(\log n)$ $\quad \omega(\log n)$

$\Omega(1)$ $\quad \omega(1)$

$\Theta(n^2)$

Worst case

$O(n^n)$ $\quad o(n^n)$

$O(n^4)$ $\quad o(n^4)$

$O(n^2)$ $\quad \cancel{o(n^2)}$

$w(n)\ n^3+9n^2+3n+8$

$\Omega(n^3)$ $\quad \omega(n^3)$

$\Omega(n^2)$ $\quad \omega(n^2)$

$\Omega(n)$ $\quad \omega(n)$

$\Omega(\sqrt{n})$ $\quad \omega(\sqrt{n})$

$\Omega(\log n)$ $\quad \omega(\log n)$

$\Omega(1)$ $\quad \omega(1)$

$\Theta(n^3)$

$f(n) = n^2$

$g(n) = n^3$

$$\lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{\frac{n^2}{n^2}}{\frac{n^3}{n^2}} = \lim_{n\to\infty} \frac{1}{n} = 0$$

if
- $\to \infty$: n^2 larger
- $\to c$: same
- $\to 0$: n^3 larger

$$\lim_{n\to\infty} \frac{n^2+3n+1}{n^2} = \lim_{n\to\infty} \frac{1+\frac{3}{n}+\frac{1}{n^2}}{1} = 1$$

$$\lim_{n\to\infty} \frac{1}{n} = 0$$

$$\lim_{n\to\infty} \frac{4n^2}{6n^2} = \lim_{n\to\infty} \frac{4}{6} = \frac{2}{3}$$ → constant

: same =

$f(n) = 4n^2 \quad g(n) = 6n^2$

$\Theta(n^2)$

$g(n) = 2^n \quad f(n) = 2^{n+1}$

$$\lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{2^n}{2^n \cdot 2} = \frac{1}{2}$$

$$\lim_{n\to\infty} \frac{g(n)}{f(n)}$$

- $\to \infty$: numerator is larger (분자)
- $\to c$: equivalent.
- $\to 0$: denominator is larger

$h(n) = n^2$

$p(n) = n^3$

$$\lim_{n\to\infty} \frac{h(n)}{p(n)} = \lim_{n\to\infty} \frac{n^2}{n^3} = \lim_{n\to\infty} \frac{\frac{n^2}{n^2}}{\frac{n^3}{n^2}} = \lim_{n\to\infty} \frac{1}{n} = 0$$

$$\text{or} = \lim_{n\to\infty} \frac{\frac{n^2}{n^3}}{\frac{n^3}{n^3}} = \lim_{n\to\infty} \frac{\frac{1}{n}}{1} = 0$$

$\therefore p(n)$ larger.

$h(n) = \log(n^2) = \log(n) + \log(n) = 2\log(n)$

$p(n) = \log(n^3) = 3\log(n)$

$$\lim_{n\to\infty} \frac{h(n)}{p(n)} = \lim_{n\to\infty} \frac{\log(n^2)}{\log(n^3)} = \lim_{n\to\infty} \frac{2\log(n)}{3\log(n)} = \frac{2}{3}$$

$\log(n!) = (\log(n^n)) = n\log n$

$w(n) = n!$

$g(n) = (n+1)!$

$$\lim_{n\to\infty} \frac{w(n)}{g(n)} = \lim_{n\to\infty} \frac{n!}{(n+1)!} = \lim_{n\to\infty} \frac{n!}{n!(n+1)} = \lim_{n\to\infty} \frac{1}{n+1} = 0$$

$\frac{f(n)}{m(n)} = \frac{2^n}{5^n}$ $\quad 5^n$ larger

$$\lim_{n\to\infty} \frac{2^n}{5^n} = \lim_{n\to\infty} \left(\frac{2}{5}\right)^n = 0$$

Powers of 2

$2^0 = 1$

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16$

$2^5 = 32$

$2^6 = 64$

$2^7 = 128$

$2^8 = 256$

$2^9 = 512$

$2^{10} = 1024 = 1k \approx 1000$

$2^{11} = 2048 = 2k \approx 2000$

$2^{12} = 4096 = 4k \approx 4000$

⋮

$2^{18} = \text{——} = 256k \approx 256,000$

$2^{20} = \text{——} = 1M \approx 1000,000$

$2^{21} = \text{——} = 2M \approx 2000,000$

$2^{28} = \text{——} = 256M \approx 256,000,000$

$2^{26} = \text{——} = 64M \approx 64,000,000$

$2^{30} = \text{——} = 1B \approx 1,000,000,000$

$2^{40} = \text{——} = 1T \approx 1,000,000,000,000$

$2^{50} = \text{——} = 1P$

$2^{45} = 32T$