COL 702 : Advanced Data-Structures & Algorithms.

Data-structure
1. Running Time Analysis ← review.
2. Graphs – DFS, BFS, Strong Connectivity, Biconnectivity ...
3. Shortest path : Bellman Ford, Dijkstra ...
4. Some more data-structures : "amortized", "randomization"

Core alg. ideas
5. Greedy
6. Divide and conquer
7. Dynamic Programming

Network Flow.
8. Max flow, matching ...
9. Min-cost flow

10. NP completeness. : approx alg. other new ideas.

— × —

Textbooks ?
- Algorithms by Kleinberg - Tardos.
- Algorithms by CLRS.
- Algorithm : Sandeep Sen & Amit Kumar.

— × —

Web-site ? Maybe NO : use TEAMS.

Grading ?
- 5% class participation (attendance, interaction ..)
- 25 % assignments (once every 2 weeks) .. quizzes ?
- open notes ← 30 % minor exam
- 40 % major exam.

Audit pass : either C or better or better than 40%.

— × —

Asymptotic Analysis :

Program → What is the running time ? — Not a well-defined question

- input not specified.
- system not specified
- cpu / hardware / ...

n

+1	sum = 0;
+2n	for i = 1 ... n
+2n	sum = sum + A[i];

t1
t2
t3

+	:	1 ns
compare	:	0.5 ns
*	:	2 ns
=	:	1.5 ns
<, >	:	1.2 ns.
A [ ]	:	...

algorithm as a sequence of "basic" instructions

1 UNIT of time — 1 ns.

t1 + 5t2 + ...

- Get an "approximate" idea: ~ seconds, ~minutes, ~hours, ~days ...?
- compare algorithms?
- How many UNITS of time? counting. : $4n+100$ ✓ || factor 2-3 loss.

$6n+1$.

Only concerned with large n $f(n)$

maximum: n, A

```
max = -1          +1
for i=1..n do     +2n
  -> if max < A[i] <- 2n
  n { max = A[i]  <- ??
output max.       +1
```

ℓn.

1 4 2
10 40 20
1 2 4

$\sim 4n+2+2\log_2 n$.

5 4 3 2 1 ✓
1 2 3 4 5 ✓

$6n+2$

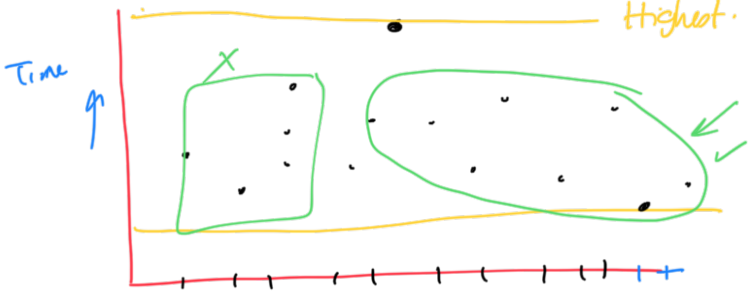
✓ 1, 5 ℓ

$4n+2+2\ell$ $4n+2$, $8n+2$.

$4n+2+\cancel{2\ell}$: running time may not be expressible in terms of n alone?

$\ell n+n$.

Conditional expressions : running time could depend critically on these.



n fixed.

different inputs of length n. ← 1, 2, ..., n

$n!$

Worst Case Analysis: Report the max. value of running time over all inputs of length n.

Best Case : Min.

→ Average Case Analysis: $\left[\dfrac{\sum_{\sigma:\, \sigma \text{ permutation}} [\text{Running time on } \sigma] *}{n!} \right]$ ← difficult quantity to calculate.

uniform ← input may come from some other distr.

σ_1, σ_2 ... $\sigma_{n!}$
1%. 2%. 0%. 0.1%. ←

✓1. # basic instructions.
✓2. Worst possible input for each n } $\boxed{\frac{1}{2}}n^2+6n$ ✓, n^2+1 ✓

$\boxed{3}$ Compare alg. or understand how running time scales as we raise n ....

$\boxed{O(n)}$ $\boxed{n}$ $\boxed{A}$ $\cancel{\boxed{100}}\, n+20$ $\boxed{B}$ ✓ ✓ n^2, n^2+6n.

$\frac{n^2}{100}+6n < 100n+20$

$n \leq 20$.

$O(n^2)$.

Care about how running time scales as we raise n.

$T(n)$: expression for running time.

only care about the largest term, and ignore constants.

$\sim n$

A	B
$4n$	$100n$.
cn	$d\, n \log n$

$\sim n$

$100n$.

$O()$: notation,

$\Omega(), \Theta()$

$100n^2\log n + \frac{n}{1000} + 1000000n^2$

$O(n^2 \log n)$

$2n$ is $O(n)$

$\boxed{100}n^2 + 50n = O(n^2)$.

— x —

Definitions: $O()$ notation ←

$f(n)$, $g(n)$ are two functions of n, where n is non-neg. integer.

running time / some known function

$\exists$: there exists

$\forall$: forall

We say that $f(n) = O(g(n))$ if

$\exists n_0 > 0, \exists c > 0$ such that $f(n) \leq c\, g(n)$ for all $n \geq n_0$

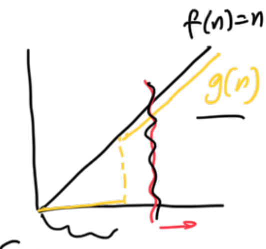
there exists

$f(n) = n^2$, $g(n) = n$.

$f(n) \neq O(g(n))$.

$f(n) = 2n$, $g(n) = n$ $\quad \underbrace{2n}_{f} \leq \underbrace{2}_{c} \cdot \underbrace{(n)}_{g}$

→ $f(n) = \underbrace{100n^2 + 50n}_{\leq 150n^2}$, $g(n) = n^2$ $\quad f(n) \leq \underbrace{150}_{c}\, g(n)$



$f(n) = 2n \quad g(n) = n^2 \quad f(n) \leq \underset{c}{2}\, g(n)$

$f(n) = \underbrace{2n+3}_{\leq 5n} \quad g(n) = n \quad f(n) \leq \underset{c}{5}\, g(n)$

$\begin{cases} f(n) = n, & n \geq 1 \\ g(n) = n, & n \geq 100 \\ 0 & \text{otherwise} \end{cases}$

$f(n) = n \quad g(n) = n^3 \quad f(n) = O(g(n))$

$n = O(n^2) \quad 10n\log n = O(n^2)$ ...

$f(n) = O(g(n))$

$f(n) \leq \boxed{c}\, g(n)$ for all n

$n = 10$: $10 \quad 0$

— x —

$f(n) = O(g(n)) \qquad f(n) = n$

$f(n) = O(g(n))$

if $\exists c>0$, $\exists n_0>0$ such that $\forall n \geq n_0$ ✓

$f(n) \leq c\, g(n)$.

$g(n) = n - 50$

$n_0 = 1.$

$c =$ __

$c = 2,\ n_0 = 100$: $n \leq \underset{\uparrow c}{2}(n-50)$ if $n \geq \underset{\uparrow n_0}{100}$.

Ex: $f(n) = n^2$, $g(n) = n$

$f(n) \neq O(g(n))$. ← by contradiction.

→ Suppose there is a c, n_0 such that $f(n) \leq c\, g(n)$ $\forall n \geq n_0$

$n^2 \leq cn$ $\forall n \geq n_0$

$\boxed{n \leq c \quad \forall n \geq n_0}$ ✗

Ex:

$g(n) = n$ if n is even, 1 if n is odd. $\|$ $f(n) = n$. $\|$

Is $f(n) = O(g(n))$?

Suppose $\exists\ n_0, c$ such that $f(n) \leq c\, g(n)$ $\forall n \geq n_0$

pick n which is odd, $n \geq n_0$:

$n \leq c$. ✗

—— x ——

$f(n) = \Omega(g(n))$ if

$\exists c, \exists n_0$ such that $\forall n \geq n_0$

$f(n) \geq c \cdot g(n)$.

$n^3 = \Omega(n)$, $\frac{n}{100} - 50$ is $\Omega(n)$., $n^2 - 18000n - 70$ is $\Omega(n)$.

—x—

$T(n)$ is the worst case running time.

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

inputs of length n → $I_1\ I_2 \ldots$

✓ $T(I_1)$: running time on I_1 $\quad T(n) = \max\{T(I_1), T(I_2) \ldots\}$

$T(I_2)$: . . . on I_2

all of them are $\leq c \cdot n^2$

$T(n) = \Omega(n^2)$.

$n \quad n^2 \quad \boxed{10n^2} \quad 5n$ ←

$\|\ \boxed{T(n)} = \max\{T(I_1), T(I_2) \ldots\} \geq cn^2$ for some c.

$\leftrightarrow$ there is an input I such that $T(I) \geq cn^2$.

Ex: Max-finding A n

```
max = -1;
for i = 1 ... n
    if A[i] > max
(*)     max = A[i].
```

n 5 4 3 2 1

$\|$ $T(n) = \Theta(n)$. $\Theta(n^2)$

n

worst case.

← $T(n)$: # times this instruction is

$T(n) = O(n^2),\ O(n\log n),\ O(n^3)$ ... = executed.

$T(n) = \Omega(n),$ $\boxed{T(n) = O(n)}$ Best Case: 1

$\Omega(\sqrt{n})$ ... $\boxed{T(n) = \Omega(n)}$ consider the input $1, 2, 3, \dots, n$

Def: We say that $f(n) = \Theta(g(n))$ if
$f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

$10n$ $5n+7$

- Worst Case
- Average Case: $T(n) :=$ Average running time ⊛ is executed. $A[i] \geq \max(A[1], \dots, A[i])$.

$a_1, \dots, a_n$ ← A all permutations

$$\frac{\sum_\sigma [\#\text{ times ⊛ is executed on } \sigma]}{n!}$$

$A[i] ,\ \max$

For how many of the $n!$ inputs will we update max when we compare $A[i]$, max. max ← $A[i]$

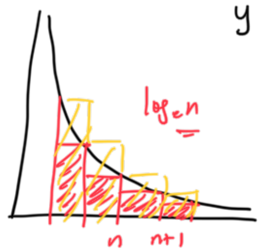
$$\binom{n}{i} \cdot \frac{(n-i)! \cdot (i-1)!}{n!} = \frac{1}{i}$$

$$\sum_\sigma [\#\text{ times ⊛ is executed}] = \sum_{i=1}^{n} \frac{[\#\text{ permutation } \sigma \text{ for which ⊛ is executed in the } i\text{th iteration}]}{n!}$$

$i: 3$ 5 2 3 ...

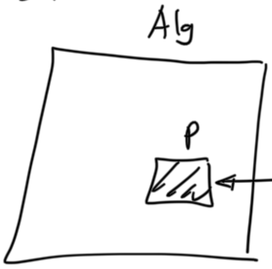
5 2 7 ... $= \sum_{i=1}^{n} \frac{1}{i} = \Theta(\log_e n).$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{n} = \log_e n \pm 1$$



Amortized Case Analysis: ← worst case for a sequence of calls.

Alg. calls P k times.
$kT(n)$



$T(n)$: worst case.

Increment: n bit #, add 1 to it

```
  1 0 1 1 0 0 1 1
                1
  ---------------
  1 0 1 1 0 1 0 0
```

$T(n)$: # bit operations.

Worst case running time: $T(n)$ $\quad T(n) = O(n)$, $T(n) = \Omega(n)$ ✓ $\Big\}$ $T(n) = \Theta(n)$.

0 1 1 1 1 1

1

Average Case? all n-bit #s

$$\frac{\sum_{x:\, n\text{ bit } \#} \text{Running time on } x}{2^n} \leq 2.$$

— × —

Suppose we start Increment with $x=0$ and keep adding 1 for n steps. What is the total running time? $O(n \log n)$.

$O(n)/n$. $O(1)$.

How often will $T(n)$ be just 1? $n/2$ ✓

2? $n/4$ ✓ 000

3 $n/8$ ✓ 01

i : $n/2^i$ ✓ 011

$\log_2 n$

$$\frac{n}{2}\cdot 1 + \frac{n}{4}\cdot 2 + \frac{n}{8}\cdot 3 + \dots + \frac{n}{2^i}\cdot i + \dots$$

$$\leq \sum_{i=1}^{\log n} \frac{n}{2^i}\, i \leq 3n$$

$$\sum_{i=1}^{\infty} \frac{i}{2^i} \leq 3.$$

$\frac{1}{2^i}$ $\quad$ 0 1 1 1 1 1 1 $\quad$ 0* $\quad$ $n/2 \cdot \frac{1}{2}$.

Amortize time Complexity: n operations $\quad \frac{\text{Total running time on } n \text{ op'ns}}{n}$

$O(\,)$, $\Omega(\,)$, $\Theta(\,)$

worst case (amortized case), average case, best case. ✗

$\leq n^2 \quad n \log n$.

$\Omega(n \log n.)$ ✓ $\quad \geq n^2$ $\quad \Omega(n \log n)$ $\quad \geq n \log n$.

$O(n^2)$. ✓ $\quad \Omega(n^2)$. ✓

$\Theta(n^2)$.

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

$O(n^3)$ ✓ $\longleftrightarrow \Omega(n^2)$ ✓

$O(n^{2.5})$ $\quad \Theta(n^{2.5})$

$\Theta(n \log n)$

$\Theta(n^2)$ $\quad \Theta(n^2)$.

1, 10, 100, 50, 30 $\quad$ 100

→ 1, n, $10n$, n^2

$\Omega(n^2)$ ✓ $\quad \Omega(n)$