COMP 482 / ELEC 420

Asymptotic Concepts & Useful Math

Math Background: Review & Beyond

1. Asymptotic concepts & useful math
2. Recurrences
3. Probabilistic analysis

Your To-Do List

- Read [CLRS] 3.1,A,B,3.2.

- Assignment 1.

Defining Resource Usage

Time

- Total time to elapsed program time

Space

- Maximum space required during program execution

Processors

- Number of processors used by parallel program

…

Defining Resource Usage

How does the running time depend on the input?

T(x) = running time for instance x

Problem: Impractical to use, e.g.,

“15 steps to sort [3 9 1 7], 13 steps to sort [1 2 0 3 9], …”

Need to abstract away from the individual instances.

Abstracting Resource Usage

Abstract based on **size** of input:

~~How does the running time depend on the input?~~

~~T(n) = running time for instances of size n~~

Problem: Also depends on other factors, e.g., input sortedness.

Solution: Bound over these instances:

Most common. Default.

Worst case	$T(n) = \max\{T(x) \mid x \text{ is an instance of size } n\}$
Best case	$T(n) = \min\{T(x) \mid x \text{ is an instance of size } n\}$
Average case	$T(n) = \Sigma_{\lvert x\rvert=n} \Pr\{x\} \times T(x)$

Determining the input probability distribution can be difficult.

Abstracting Resource Usage

What's confusing about this notation?

Worst case	$T(n) = \max\{T(x) \mid x \text{ is an instance of size } n\}$
Best case	$T(n) = \min\{T(x) \mid x \text{ is an instance of size } n\}$
Average case	$T(n) = \Sigma_{\lvert x\rvert=n} \Pr\{x\} \times T(x)$

Two different kinds of functions:

T(instance)   T(size of instance)

Won't use T(instance) notation again, so can ignore.

Asymptotes: Why?

Problem: $T(n) = 3n^2 + 14n + 17$

Too much detail: constant factors may reflect implementation details & lower terms are insignificant.

Solution: Ignore constant factors & low-order terms.

(Omitted details still important pragmatically.)

n	$3n^2$	14n+17
1	3	31
10	300	157
100	30,000	1,417
1000	3,000,000	14,017
10000	300,000,000	140,017

$3n^2 > 14n+17$

$\forall$ "large enough" n

Upper Bounds

Creating an algorithm proves we can solve the problem within a given bound.

But another algorithm might be faster.

E.g., sorting an array.

Insertion sort → $O(n^2)$

Lower Bounds

Sometimes can prove that we cannot compute something without a sufficient amount of time.

That doesn't necessarily mean we know how to compute it in this lower bound.

E.g., sorting an array.

comparisons needed in worst case → $\Omega(n \log n)$

Will prove this soon…

Upper & Lower Bounds: Informal Summary

Upper bounds:

$\leq$ O() $<$ o()

Lower bounds:

$\geq$ Ω() $>$ ω()

Upper & lower (“tight”) bounds:

$=$ Θ()

Definitions: O, Ω

$T(n) \in O(g(n))$

$\leftrightarrow$

$\exists$ constants $c, n_0 > 0$

such that

$\forall\ n \geq n_0,\ 0 \leq T(n) \leq c \cdot g(n)$

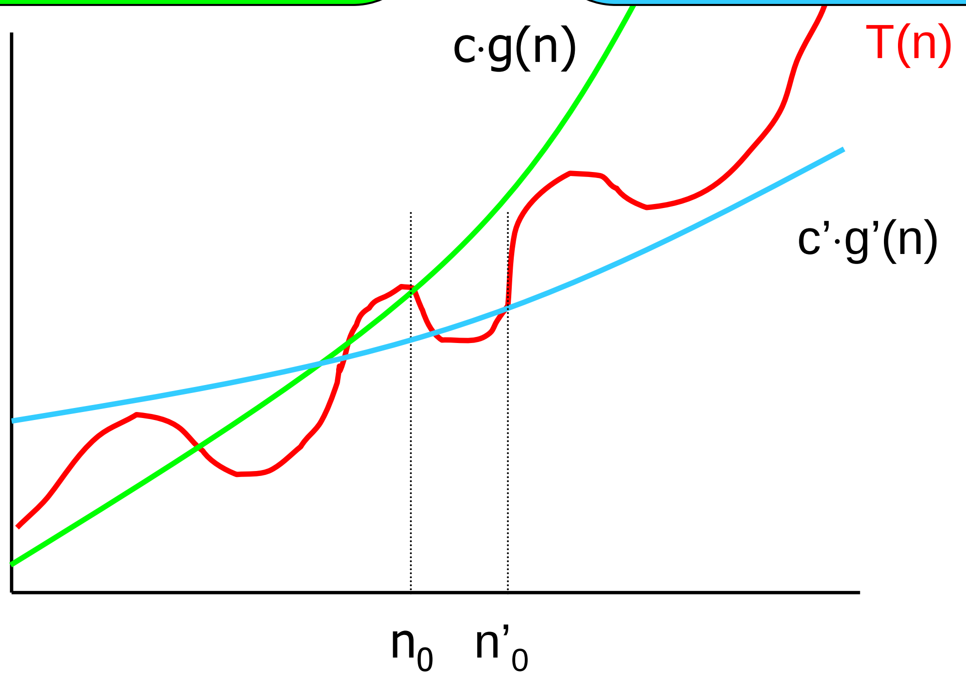
$T(n) \in \Omega(g'(n))$

$\leftrightarrow$

$\exists$ constants $c', n'_0 > 0$

such that

$\forall\ n \geq n'_0,\ T(n) \geq c' \cdot g'(n) \geq 0$



Examples: O, Ω

$2n+13 \in O($? $)$	$O(n)$ Also, $O(n^2)$, $O(5n)$, … Can always weaken the bound.
$2n+13 \in \Omega($? $)$	$\Omega(n)$, also $\Omega(\log n)$, $\Omega(1)$, …
$2^n \in O(n)$? $\Omega(n)$?	Given a c, $2^n \geq c \cdot n$, for all but small n. $\Omega(n)$, not $O(n)$.
$n^{\log n} \in O(n^5)$?	No. Given a c, $\log n \geq c \cdot 5$, for all large enough n. Thus, $\Omega(n^5)$.
$-n \in O(n)$?	No. -n not asymptotically positive.

Definitions: o, ω

Might know that our upper & lower bounds aren't tight.

$T(n) \in o(g(n))$

$\leftrightarrow$

$\forall$ constants $c>0$, $\exists$ constant $n_0>0$,
such that
$\forall\ n \geq n_0,\ 0 \leq T(n) < c \cdot g(n)$

Also, $T(n) \in o(g(n))$

$\leftrightarrow$

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = 0$$

Not tight…

…for any constant.

$T(n) \in \omega(g'(n))$

$\leftrightarrow$

$\forall$ constants $c'>0$, $\exists$ constant $n'_0>0$,
such that
$\forall\ n \geq n'_0,\ T(n) > c \cdot g'(n) \geq 0$

Examples: o, ω

$2n+13 \in o(n)$? $\omega(n)$?

No to both!

$2n+13 < c \cdot n$ fails for many c.

$2n+13 > c \cdot n$ fails for many c.

$2n+13 \in o(\ ?\ )\quad \omega(\ ?\ )$

$o(n \log n)$, $o(n^2)$, …

$\omega(\log n)$, $\omega(1)$, …

$1 / \log n \in o(1)$?

Yes.

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{1/\log n}{1} = \lim_{n\to\infty} \frac{1}{\log n} = 0$$

Definitions: Θ

$$T(n) \in \Theta(g(n))$$

$$\leftrightarrow$$

$$T(n) \in O(g(n)) \text{ and } T(n) \in \Omega(g(n))$$

Ideally, find algorithms that are asymptotically as good as possible.

Alternate Definitions

Some authors use:

$$T(n) \in O_1(g(n))$$

$$\leftrightarrow$$

$$\exists \text{ constants } c, n_0 > 0$$

such that

$$\forall\ n \geq n_0,\ |T(n)| \leq c \cdot |g(n)|$$

What is an example of when O() and O_1() disagree?

Some students try:

$$T(n) \in O_2(g(n))$$

$$\leftrightarrow$$

$$\exists \text{ constants } c, n_0 > 0$$

such that

$$\forall n \geq n_0,\ \cancel{\lim_{n\to\infty}} \frac{T(n)}{g(n)} < \infty$$

What is an example of when O() and O_2() disagree?

Fix: $\limsup_{n\to\infty}$

Notation

O(), Ω(), … are sets of functions.

But common to abuse notation:

Shorthand	Meaning
$f(n) = O(g(n))$	$f(n) \in O(g(n))$
$f(n) = g(n) + O(h(n))$	$f(n) \in \{g(n) + h'(n) \mid h'(n) \in O(h(n))\}$
$f(n) + O(g(n)) = O(h(n))$	$\{f(n) + g'(n) \mid g'(n) \in O(g(n))\} \subseteq O(h(n))$

Review (mostly) some math useful for asymptotics…

Bounds

Some useful algebra: $a<b$, $c<d$ implies…

- $-b < -a$
- $1/b < 1/a$ – Unless a,b have different signs!
- $a+c < b+c$
- $ac < bc$ – If $c>0$
- $a+c < b+d$ – But not necessarily $a-c < b-d$ or $a-c > b-d$!
- $a^n < b^n$ – If $a,b>0$

Floor & Ceilings

n objects in m groups → some group has $\geq \lceil n/m \rceil$ objects

$$a-1 < \lfloor a \rfloor \leq a \leq \lceil a \rceil < a+1$$

$\forall$ a>0

$$\frac{a-(b-1)}{b} \leq \left\lfloor \frac{a}{b} \right\rfloor \leq \frac{a}{b} \leq \left\lceil \frac{a}{b} \right\rceil \leq \frac{a+(b-1)}{b}$$

$\forall$ integer a>0,

$\forall$ b≥1

$$\left\lfloor \frac{a}{2} \right\rfloor + \left\lceil \frac{a}{2} \right\rceil = a$$

$\forall$ integer a

Logarithm Notation

$\lg n = \log_2 n$ $\ln n = \log_e n$ $\log n = \log_{10} n$

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

$\log \log n = \log (\log n)$

$\log n + k = (\log n) + k$

$\log nk = \log (nk)$

$\log^{(0)} n = n$ $\log^{(i+1)} n = \log \log^{(i)} n$

$\log^* n = \min\{i \geq 0 \mid \log^{(i)} n \leq 1\}$

How many times can you take the logarithm before it’s ≤ 1?

Exponentials & Logarithms

$$a^{\log_b c} = c^{\log_b a} \xrightarrow{c=b} a = b^{\log_b a}$$

To change base, just multiply
by appropriate constant.

$$\log_a x = \left(\log_b x\right) \cdot \left(\log_a b\right)$$

$$\xrightarrow{x=a} 1 = \left(\log_b a\right) \cdot \left(\log_a b\right)$$

Asymptotically, the log base doesn’t matter:

O(lg n) = O(ln n) = O(log n)

Polylogarithmic is smaller than polynomial:

$$\log_b^c n = o\left(n^a\right)$$

For a>0

Factorials

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^{n}\left(1+\Theta\left(\frac{1}{n}\right)\right)$$

Factorial vs. exponential:

Bigger than some: $n! = \omega(2^n)$

Smaller than some: $n! = o(n^n)$

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

Choice Function

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

“n choose k” = # of k-combinations of an n-set

There are many useful identities, but
most are easily derived from def’n.

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

Summations

Find closed forms of summations.

A quick review of some techniques…

Summations

Know some common ones:

Arithmetic $$\sum_{i=0}^{n} i = n + (n-1) + (n-2) + ... + 2 + 1 + 0 = \frac{n(n+1)}{2}$$

Geometric $$\sum_{i=0}^{n} x^i = \frac{x^{n+1} - 1}{x - 1}$$ if $x \neq 1$

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1 - x}$$ if $|x| < 1$

Exponential $$\sum_{i=0}^{\infty} \frac{x^i}{i!} = e^x$$

Summations

Know some common ones:

Harmonic $$\sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$$

Telescoping $$\sum_{i=0}^{n-1} (a_i - a_{i+1}) = a_0 - a_1 + a_1 - a_2 + \ldots + a_{n-1} - a_n = a_0 - a_n$$

How to solve following as telescoping sum?

$$\sum_{i=1}^{n-1} \frac{1}{i \times (i+1)} = \ ?$$

$$\sum_{i=1}^{n-1} \left(\frac{i+1}{i \times (i+1)} - \frac{i}{i \times (i+1)} \right) = \sum_{i=1}^{n-1} \left(\frac{1}{i} - \frac{1}{i+1} \right) = 1 - \frac{1}{n}$$

Summations

Combine known summations with differentiation, e.g.,

Have: $$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$$ if $|x| < 1$

Differentiate: $$\sum_{i=0}^{\infty} i \cdot x^{i-1} = \frac{1}{(1-x)^2}$$ if $|x| < 1$

Algebra: $$\sum_{i=0}^{\infty} i \cdot x^i = \frac{x}{(1-x)^2}$$ if $|x| < 1$

Similarly, can combine with integration.

Remember calculus?

Summations

Bounding is often sufficient, e.g.,

Could solve as arithmetic sum:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

To prove: $\sum_{i=0}^{n} i = O(n^2)$

$$\sum_{i=0}^{n} i \le \sum_{i=0}^{n} n = n(n+1) = O(n^2)$$

Anything bounded above by $O(n^2)$ is also $O(n^2)$.

Summations

Bounding is often sufficient.

<u>Similar ideas:</u>

- Rounding up with $\lceil \; \rceil$
- Rounding up to a multiple of k
- Rounding up to a power of k

- Rounding down for $\Omega()$

Summations

Bounding is often sufficient, e.g.,

Could solve as geometric sum:

$$\sum_{i=0}^{n} x^{i} = \frac{x^{n+1} - 1}{x - 1}$$

To prove: $\sum_{i=0}^{n} 7^{i} = O\left(7^{n}\right)$

Show by induction: $\sum_{i=0}^{n} 7^{i} \leq c \cdot 7^{n}$ For some c

Summations

Show by induction: $\sum_{i=0}^{n} 7^{i} \le c \cdot 7^{n}$ For some c

? What is the base case? ?

Base, n<0: $0 \le c/7$

Zero terms in sum.

Summations

Show by induction: $$\sum_{i=0}^{n} 7^{i} \leq c \cdot 7^{n}$$ For some c

Base, n<0: $$0 \leq c/7$$

Induction, n≥0: $$\sum_{i=0}^{n} 7^{i} = \left(\sum_{i=0}^{n\text{-}1} 7^{i} \right) + 7^{n}$$

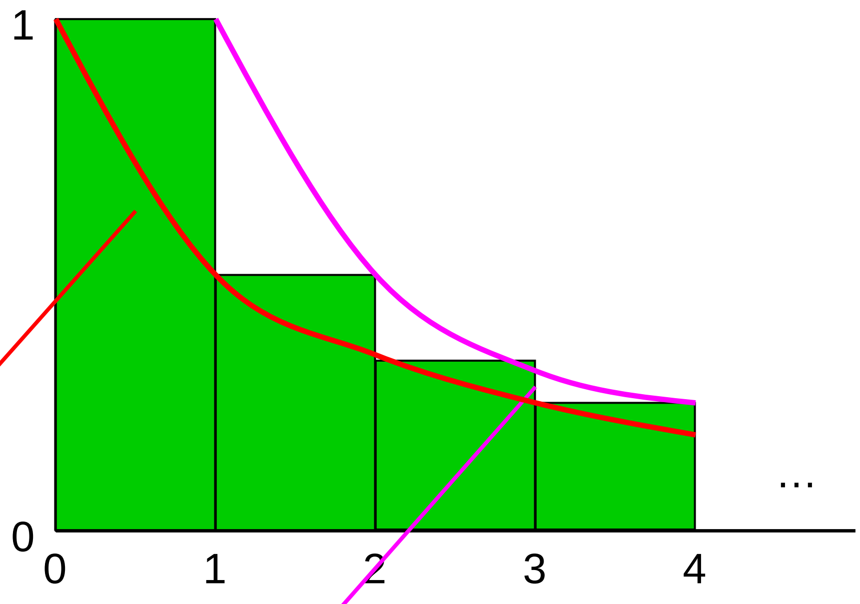
$$\leq c \cdot 7^{n-1} + 7^{n} = \left(\frac{1}{7} + \frac{1}{c} \right) \cdot c \cdot 7^{n} \leq c \cdot 7^{n}$$

if $$\frac{1}{7} + \frac{1}{c} \leq 1 \qquad c + 7 \leq 7c \qquad \frac{7}{6} \leq c$$

Summations

Bound/approximate by integrals, e.g.,

To prove: $$\sum_{i=1}^{n} \frac{1}{i} = \ln n + O(1)$$



$$\ln(n+1) = \int_{1}^{n+1} \frac{1}{x} dx \le \sum_{i=1}^{n} \frac{1}{i} \le 1 + \int_{1}^{n} \frac{1}{x} dx = (\ln n) + 1$$

Recall: ln 1 = 0