CSCE 411
Design and Analysis of Algorithms

Andreas Klappenecker

Goal of this Lecture

- Recall the basic asymptotic notations such as Big Oh, Big Omega, Big Theta, and little oh.
- Recall some basic properties of these notations
- Give some motivation why these notions are defined in the way they are.
- Give some examples of their usage.

Summary

Let g: **N**->***C*** be a real or complex valued function on the natural numbers.

$O(g)$ = { f: **N**-> ***C*** | ∃u>0 ∃n_0 ∈**N**

|f(n)| <= u|g(n)| for all n>= n_0 }

$\Omega(g)$ = { f: **N**-> ***C*** | ∃d>0 ∃n_0 ∈**N**

d|g(n)| <= |f(n)| for all n>= n_0 }

$\Theta(g)$ = { f: **N**-> ***C*** | ∃u,d>0 ∃n_0 ∈**N**

d|g(n)| <= |f(n)| <= u|g(n)| for all n>= n_0 }

Time Complexity

- Estimating the time-complexity of algorithms, we simply want count the number of operations. We want to be
  - independent of the compiler used,
  - ignorant about details about the number of instructions generated per high-level instruction,
  - independent of optimization settings,
  - and architectural details.

  This means that performance should only be compared up to multiplication by a constant.
- We want to ignore details such as initial filling the pipeline. Therefore, we need to ignore the irregular

Big Oh

Big Oh Notation

Let f,g: N -> R be function from the natural numbers to the set of real numbers.

We write $f \in O(g)$ if and only if there exists some real number n_0 and a positive real constant u such that

$$|f(n)| <= u|g(n)|$$

for all n in S satisfying $n >= n_0$

Big Oh

Let g: N-> C be a function.

Then O(g) is the set of functions

O(g) = { f: N-> C | there exists a constant u and a natural number n_0 such that

$|f(n)| \le u|g(n)|$ for all $n \ge n_0$ }

Notation

We have

$O(n^2) \subseteq O(n^3)$

but it is usually written as

$O(n^2) = O(n^3)$

This does not mean that the sets are equal!!!! The equality sign should be read as 'is a subset of'.

Notation

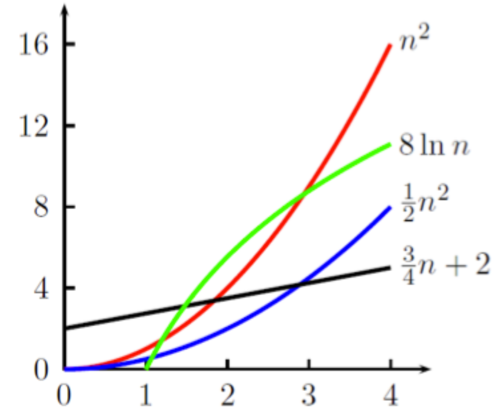
We write $n^2 = O(n^3)$,

[read as: n^2 is contained in $O(n^3)$]

But we never write

$O(n^3) = n^2$

Example $O(n^2)$



Big Oh Notation

The Big Oh notation was introduced by the number theorist Paul Bachman in 1894. It perfectly matches our requirements on measuring time complexity.

Example:

$$4n^3+3n^2+6 \text{ in } O(n^3)$$

The biggest advantage of the notation is that complicated expressions can be dramatically simplified.

Quiz

Does O(1) contain only the constant functions?

Tool 1: Limits

Limit

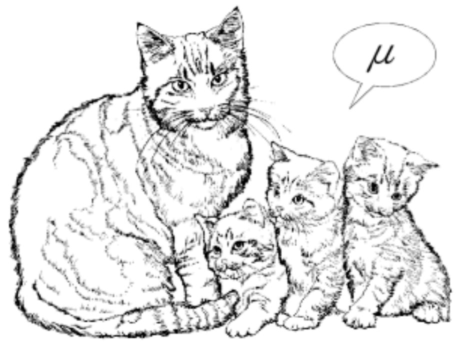
Let (x_n) be a sequence of real numbers.

We say that μ is the limit of this sequence of numbers and write

$\mu = \lim_{n \to \infty} x_n$

if and only if for each $\varepsilon > 0$ there exists a natural number n_0 such that $|x_n - \mu| < \varepsilon$ for all $n >= n_0$

μ? μ!



Limit - Again!

Let (x_n) be a sequence of real numbers.

We say that μ is the limit of this sequence of numbers and write

$\mu = \lim_{n\to\infty} x_n$

if and only if for each $\varepsilon > 0$ there exists a natural number n_0 such that $|x_n - \mu| < \varepsilon$

How do we prove that g = O(f)?

Lemma 1. *Let f and g be functions from the positive integers to the complex numbers such that $g(n) \neq 0$ for all $n \geq n_0$ for some positive integer n_0. If the limit $\lim_{n\to\infty} |f(n)/g(n)|$ exists and is finite then $f(n) = O(g(n))$.*

Proof. If $\lim_{n\to\infty} |f(n)/g(n)| = C$, then for each $\epsilon > 0$ there exists a positive integer $n_0(\epsilon)$ such that $C - \epsilon \leq |f(n)/g(n)| \leq C + \epsilon$ for all $n \geq n_0$; this shows that $|f(n)| \leq (C + \epsilon)|g(n)|$ for all integers $n \geq n_0(\epsilon)$. It follows that $f(n) = O(g(n))$. $\square$

Big versus Little Oh

$O(g) = \{ f: \mathbf{N} \to \mathbf{C} \mid \exists u>0\ \exists n_0 \in \mathbf{N}$

$|f(n)| \le u|g(n)|$ for all $n \ge n_0 \}$

$o(g) = \{ f: \mathbf{N} \to \mathbf{C} \mid \lim_{n\to\infty} |f(n)|/|g(n)| = 0 \}$

Quiz

It follows that $o(f)$ is a subset of $O(f)$.

Why?

Quiz

What does $f = o(1)$ mean?

Hint:

$o(g) = \{ f: \mathbf{N} \rightarrow \mathbf{C} \mid \lim_{n \rightarrow \infty} |f(n)|/|g(n)| = 0 \}$

Quiz

Some computer scientists consider little oh notations too sloppy.

For example, $1/n+1/n^2$ is $o(1)$

but they might prefer $1/n+1/n^2 = O(1/n)$.

Why is that?

Tool 2: Limit Superior

Limits? There are no Limits!

The limit of a sequence might not exist.

For example, if $f(n) = 1+(-1)^n$ then

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

does not exist.

Least Upper Bound (Supremum)

The supremum b of a set of real numbers S is the defined as the smallest real number b such that b>=x for all s in S.

We write s = sup S.

- sup {1,2,3} = 3,
- sup $\{x : x^2 < 2\}$ = sqrt(2),
- sup {(-1)^n – 1/n : n>=0 } = 1.

The Limit Superior

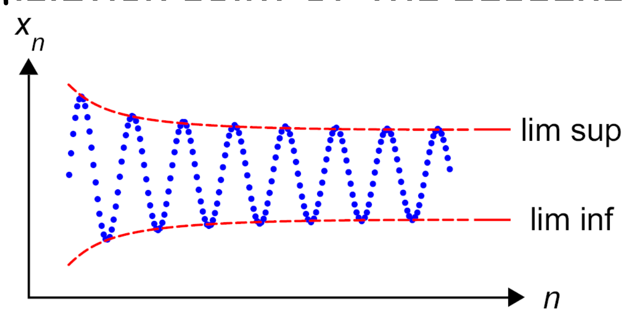
The limit superior of a sequence (x_n) of real numbers is defined as

$\limsup_{n \to \infty} x_n = \lim_{n \to \infty} (\sup \{ x_m : m >= n \})$

[Note that the limit superior always exists in the extended real line (which includes $\pm\infty$), as $\sup \{ x_m : m>=n\})$ is a monotonically decreasing function of n and is bounded below by any

The Limit Superior

The limit superior of a sequence of real numbers is equal to the greatest accumulation point of the sequence.



Necessary and Sufficient Condition

Lemma 2. *Let f and g be functions from the positive integers to the complex numbers such that $g(n) \neq 0$ for all $n \geq n_0$ for some positive integer n_0. We have $\limsup_{n\to\infty} |f(n)/g(n)| < \infty$ if and only if $f(n) = O(g(n))$.*

Proof. If $\limsup_{n\to\infty} |f(n)/g(n)| = C$, then for each $\epsilon > 0$ we have

$$|f(n)|/|g(n)| > C + \epsilon$$

for at most finitely many positive integers; so $|f(n)| \leq (C+\epsilon)|g(n)|$ holds for all integers $n \geq n_0(\epsilon)$ for some positive integer $n_0(\epsilon)$, and this proves that $f(n) = O(g(n))$.

Conversely, if $f(n) = O(g(n))$, then there exists a positive integer n_0 and a constant C such that $g(n) \neq 0$ and $|f(n)|/|g(n)| \leq C$ for all $n \geq n_0$. This implies that $\limsup_{n\to\infty} |f(n)/g(n)| \leq C$. □

Big Omega

Big Omega Notation

Let f, g: N-> R be functions from the set of natural numbers to the set of real numbers.

We write $g \in \Omega(f)$ if and only if there exists some real number n_0 and a positive real constant C such that

$$|g(n)| >= C|f(n)|$$

for all n in N satisfying $n >= n_0$.

Big Omega

Theorem: $f \in \Omega(g)$ iff $\liminf_{n\to\infty} |f(n)/g(n)| > 0$.

Proof: If $\liminf |f(n)/g(n)| = C > 0$, then we have for each $\varepsilon > 0$ at most finitely many positive integers satisfying $|f(n)/g(n)| < C - \varepsilon$. Thus, there exists an n_0 such that

$$|f(n)| \geq (C-\varepsilon)|g(n)|$$

holds for all $n \geq n_0$, proving that $f \in \Omega(g)$.

The converse follows from the definitions.

Big Theta

Big Theta Notation

Let S be a subset of the real numbers (for instance, we can choose S to be the set of natural numbers).

If f and g are functions from S to the real numbers, then we write $g \in \Theta(f)$ if and only if

there exists some real number n_0 and positive real constants C and C' such that

$$C|f(n)| <= |g(n)| <= C'|f(n)|$$

for all n in S satisfying $n >= n_0$.

Thus, $\Theta(f) = O(f) \cap \Omega(f)$

Examples

Sums

- $1+2+3+\ldots+n = n(n+1)/2$
- $1^2 + 2^2 + 3^2 + \ldots + n^2 = n(n+1)(2n+1)/6$

We might prefer some simpler formula, especially when looking at sum of cubes, etc.

The first sum is approximately equal to $n^2/2$, as $n/2$ is much smaller compared to $n^2/2$ for large n. The second sum is approximately equal to $n^3/3$ plus smaller terms.

Approximate Formulas

(complicated function of n)

= (simple function of n)

+ (bound for the size of the error in terms of n)

Approximate Formulas

Instead of

$1^2 + 2^2 + 3^2 + \ldots + n^2 = n^3/3 + n^2/2 + n/6$

we might write

$1^2 + 2^2 + 3^2 + \ldots + n^2 = n^3/3 + O(n^2)$

Approximate Formulas

If we write $f(n) = g(n)+O(h(n))$, then this means that there exists a constant $u>0$ and a natural number n_0 such that

$$|f(n)-g(n)| <= u|h(n)|$$

for all $n>=n_0$.

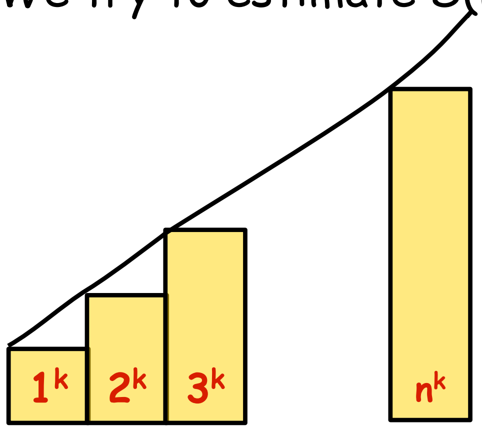
Bold Conjecture

$1^k + 2^k + 3^k + \ldots + n^k = n^{k+1}/(k+1) + O(n^k)$

Proof

Write $S(n) = 1^k + 2^k + 3^k + \ldots + n^k$

We try to estimate $S(n)$.



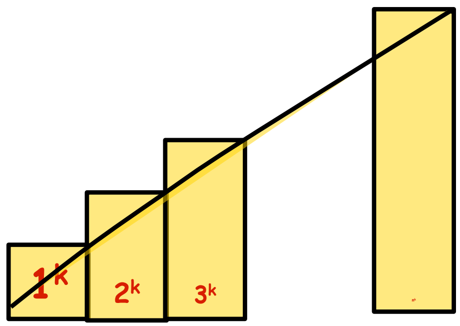
$S(n) < \int_1^{n+1} x^k \, dx$

$S(n) < (n+1)^k/(k+1)$

Proof

Write $S(n) = 1^k + 2^k + 3^k + \ldots + n^k$

We try to estimate $S(n)$.



$S(n) > \int_0^n x^k \, dx$

$S(n) > n^{k+1}/(k+1)$

Proof

We have shown that

$n^{k+1}/(k+1) < 1^k + 2^k + 3^k + \ldots + n^k < (n+1)^{k+1}/(k+1)$.

Let's subtract $n^{k+1}/(k+1)$ to get

$0 < 1^k + 2^k + 3^k + \ldots + n^k - n^{k+1}/(k+1)$

$< ((n+1)^{k+1} - n^{k+1})/(k+1)$

Proof

$((n+1)^{k+1} - n^{k+1})/(k+1) = (k+1)^{-1} \Sigma_{i=0}^{k+1} C(k+1,i)n^i - n^{k+1}$

$<= \Sigma_{i=0}^{k} \; C(k+1,i)n^k$

// simplify and enlarge terms

$<=$ (some mess involving k) n^k

It follows that

$S(n) - n^{k+1}/(k+1) <$ (mess involving k) n^k

End of Proof

Since the mess involving k does not depend on n, we have proven that

$$1^k + 2^k + 3^k + \ldots + n^k = n^{k+1}/(k+1) + O(n^k)$$

holds!

Harmonic Number

The Harmonic number H_n is defined as

$H_n = 1+1/2+1/3+\ldots+1/n$.

We have

$H_n = \ln n + \gamma + O(1/n)$

where γ is the Euler-Mascheroni constant

$$\gamma = \lim_{n\to\infty}\left[\left(\sum_{k=1}^{n}\frac{1}{k}\right) - \ln(n)\right] = \int_1^\infty \left(\frac{1}{\lfloor x\rfloor} - \frac{1}{x}\right)\,dx. = 0.577\ldots$$

log n!

Recall that 1! = 1 and n! = (n-1)! n.

Theorem: log n! = Θ(n log n)

Proof:

log n! = log 1 + log 2 + ... + log n

<= log n + log n + ... + log n = n log n

Hence, log n! = O(n log n).

log n!

On the other hand,

log n! = log 1 + log 2 + … + log n

$\geq$ log ($\lfloor (n+1)/2 \rfloor$) + … + log n

$\geq$ ($\lfloor (n+1)/2 \rfloor$) log ($\lfloor (n+1)/2 \rfloor$)

$\geq$ n/2 log(n/2)

= Ω(n log n)

For the last step, note that

$\liminf_{n \to \infty} (n/2 \log(n/2))/(n \log n) = \frac{1}{2}$.

Reading Assignment

- Read Chapter 1-3 in [CLRS]
- Chapter 1 introduces the notion of an algorithm
- Chapter 2 analyzes some sorting algorithms
- Chapter 3 introduces Big Oh notation