Algorithms

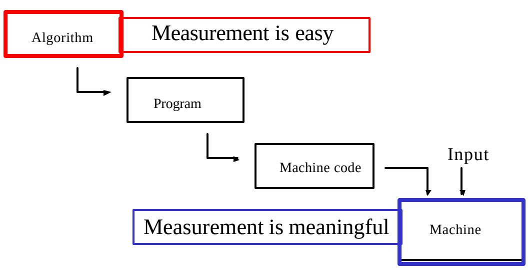
Z Sweedyk
Lecture 2
9/4/00

Last class

The two important questions we consider in CS140:

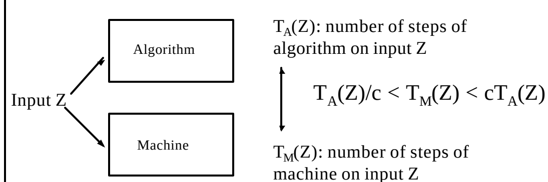
– Is the algorithm correct?
– **Is the algorithm fast?**

Running Time
Where to measure?

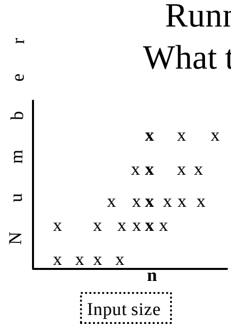


A useful assumption

T_A and T_M differ by no more than a multiplicative constant

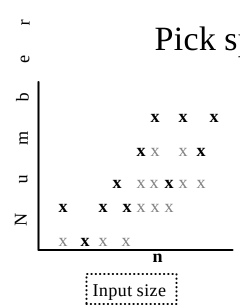


Running Time:
What to measure?



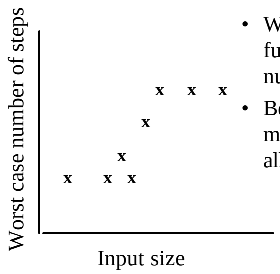
- Run time depends on input size
- Run time can vary on different inputs of size n.

Pick special case



- Run time depends on input size
- Run time can vary on different inputs of size n.
- Choose case:
  - Worst case (show in bold)
  - Best case
  - Average case
  - Etc.

Worst case performance of algorithm A



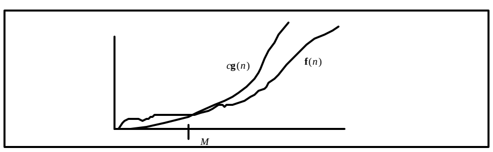
- We can compute this function at a finite number of points .
- Better yet, we can model this function for all input sizes.

A general problem …

- Question: How can we give a succinct description of an arbitrary function?
- Answer: Big-O notation.

Upper Bounds

- **f**:N→N and **g**:N→N are positive-valued, monotonically increasing functions.
- O(**g**(n)) = {**f**(n): there are constants c and M such that **f**(n) $\leq c$ **g**(n) for all $n \geq M$}



Examples:

- Is $n^2 \varepsilon O(n^3)$?
- Is $2^n \varepsilon O(n^3)$?

We will also say

f(n) = O(**g**(n)) **to mean f**(n) ε O(**g**(n))

CS140 pragmatism

What is the asymptotic behavior of the worst-case running time of the algorithm?

CS140 pragmatism

Big-O

What is the **asymptotic behavior** of the worst-case running time of the algorithm?

CS140 pragmatism

What is the asymptotic behavior of the **worst-case** running time of the algorithm?

Special case input

CS140 pragmatism

What is the asymptotic behavior of the worst-case **running time** of the algorithm?

Chosen resource

CS140 pragmatism

What is the asymptotic behavior of the worst-case running time of the **algorithm**?

Remember our assumption

Run time bounds for algorithm A

The running time of A is $O(n^3)$.

⇕

The worst case running time of A is $O(n^3)$.

⇕

A is $O(n^3)$.

Rate of growth of common functions

- Review of properties/notation
- See CLR pp 32-37 for details

KNOW THIS STUFF

Some useful observations about Big-O

- Transitivity:
  f(n)=O(g(n)) and g(n)=O(h(n)) $\Rightarrow$ f(n)=O(h(n))
- If $\lim_{n\to\infty}$ f(n)/g(n) is constant then f(n)=O(g(n)).
- If $\lim_{n\to\infty}$ f(n)/g(n) is unbounded then f(n)$\neq$O(g(n)).

9/4/00 CS140 - Fall 00 19

Logarithms

Compare the rates of growth of the following functions:

- lg n
- lg n^2
- lg^2 n
- lg 10000n

9/4/00 CS140 - Fall 00 20

Polynomials

Compare the rate of growth of the following functions:

- n
- n^2
- $1000n^2 + n$

9/4/00 CS140 - Fall 00 21

Polynomially bounded functions

f(n) is polynomially bounded if there is a constant k such that f(n) = $O(n^k)$

9/4/00 CS140 - Fall 00 22

Exponentials

- Compare the rate of growth of the following functions:
- 2^n
- 3^n
- $2^{(n^2)}$
- $(2^n)^2$

9/4/00 CS140 - Fall 00 23

Some rules of thumb

- Logs are slower growing than polynomials
  log(n) = $O(n^k)$ for any k>0
- Polynomials are slower growing than exponentials
  $n^k = O(r^n)$ for any k>0, r>1

9/4/00 CS140 - Fall 00 24

Logs, Polys, and Exps

- Compare the rate of growth of the following functions:
  log n
  n^3
  2^n
- Which are polynomially bounded?

Other functions

- Factorial: $n! = n\,(n-1)!$, $0!=1$
- Tower of 2: $T^*(n) = 2^{T^*(n-1)}$, $T^*(0)=1$
- Iterated log: $\text{Log}^*(n) = m$ such that $T^*(m-1) < n \le T^*(m)$
- Ceil-ceil: $\lceil\lceil n \rceil\rceil = 2^m$ such that $m-1 < \lg n \le m$

Logs, polys, exps, and others

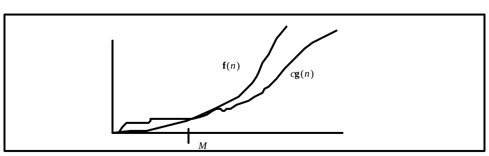
- Compare the rate of growth of the following functions:
  $\lg n$, n^3, 2^n, $n!$, $T^*(n)$, $\log^*(n)$, $\lceil\lceil n \rceil\rceil$
- Which of these functions are polynomially bounded?

Beyond O

real numbers	functions
• $\le$	• O
• $\ge$	• Ω
• =	• Θ
• <	• o
• >	• ω

Lower Bounds

- **f**:N→N and **g**:N→N are positive-valued, monotonically increasing functions.
- $\Omega(\mathbf{g}(n)) = \{\mathbf{f}(n)$: there are constants c and M such that $\mathbf{f}(n) \ge c\,\mathbf{g}(n)$ for all $n \ge M\}$



We will also say

$\mathbf{f}(n) = \Omega(\mathbf{g}(n))$ **to mean** $\mathbf{f}(n) \in \Omega(\mathbf{g}(n))$

Definition: Θ

$f(n) = \Theta(g(n))$ if the following hold:

1. $f(n) = O(g(n))$, and
2. $f(n) = \Omega(g(n))$

Definition: little-o, little-ω

- $f(n) = o(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = 0$
- $f(n) = \omega(g(n))$ if $\lim_{n\to\infty} f(n)/g(n) = \infty$

Logs, polys, exps, and others

Compare the following functions. Which of O, Ω, Θ, o, and ω apply?

$\lg n$, n^3, 2^n, $n!$, $T^*(n)$, $\log^*(n)$, $\lceil\lceil n \rceil\rceil$

A slight twist…

Is $f(2n) = O(f(n))$?

1. $f(n) = 1$: Is $2n=O(n)$?
2. $f(n) = 3n$: Is $6n=O(3n)$?
3. $f(n) = n^2$: Is $4n^2=O(n^2)$?
4. $f(n) = 2^n$: Is $4^n=O(2^n)$?
5. $f(n) = n!$: Is $(2n)!=O(n1)$?