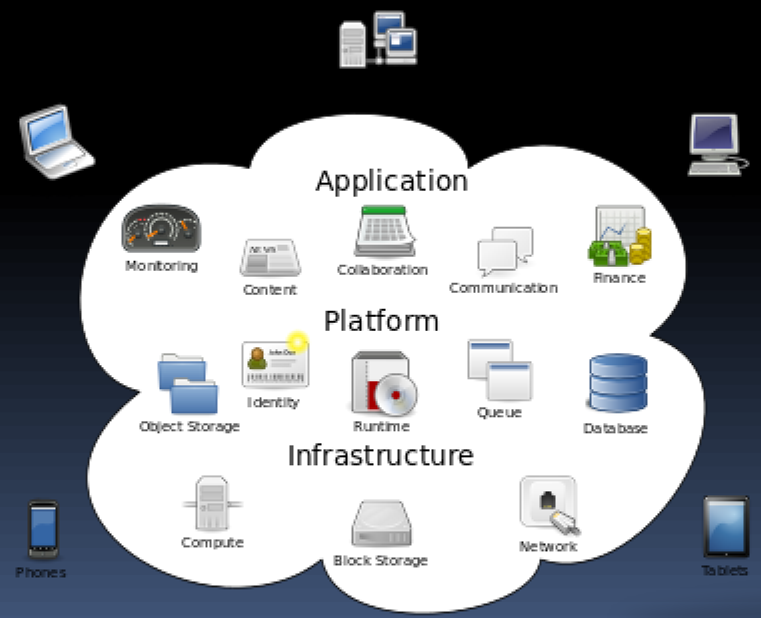
Jonathan McKinsey

The Beauty and Joy of Computing

Lecture #7
Algorithmic Complexity

What is your cloud strategy?



Cloud Computing

https://en.wikipedia.org/wiki/Cloud_computing

Algorithms: Specifications

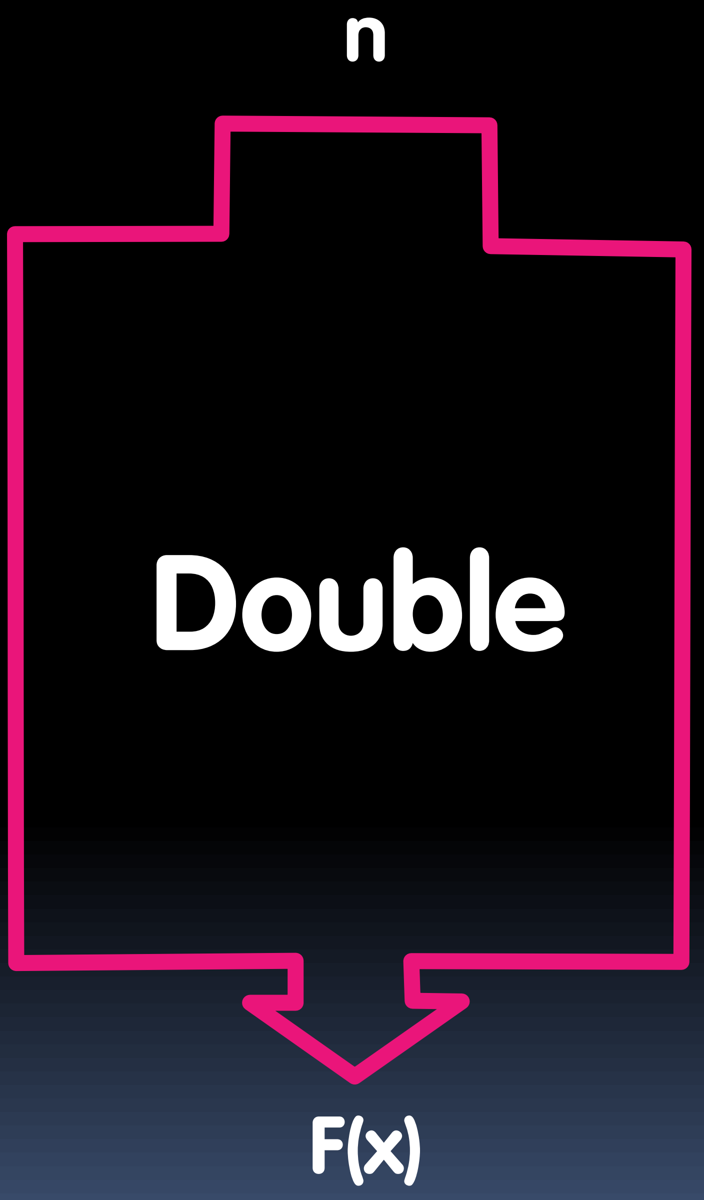
Functional Abstraction

- A **block**, or **function** has inputs & outputs
  - Possibly no inputs
  - Possibly no outputs (if block is a command)
    - In this case, it would have a "side effect", i.e., what it does (e.g., move a robot)
- The contract describing what block does is called a **specification** or **spec**

McKinsey

What is IN a spec?

- Typically they all have
  - NAME
  - INPUT(s)
    - (and types, if appropriate)
    - Requirements
  - OUTPUT
    - Can write "none"
  - (SIDE-EFFECTS)
  - EXAMPLE CALLS
- Example
  - NAME : Double
  - INPUT : n (a number)
  - OUTPUT: Twice input
  - SAMPLE: 



What is NOT in a spec?

- How!
  - That's the beauty of a functional abstraction; it doesn't say **how** it will do its job.
- Example: Double(n)
  - Could be n * 2
  - Could be n + n
  - Could be n+1 (n times)
    - if n is a positive integer
- This gives great freedom to author!
  - You choose algorithm(s)!

McKinsey

What do YOU think?

Which factor below is the most important in choosing the algorithm to use?

A. Simplest?
B. Easiest to implement?
C. Takes less time?
D. Uses up less space (memory)?
E. Gives a more precise answer?

Algorithm Analysis

Algorithm Analysis: Running Time

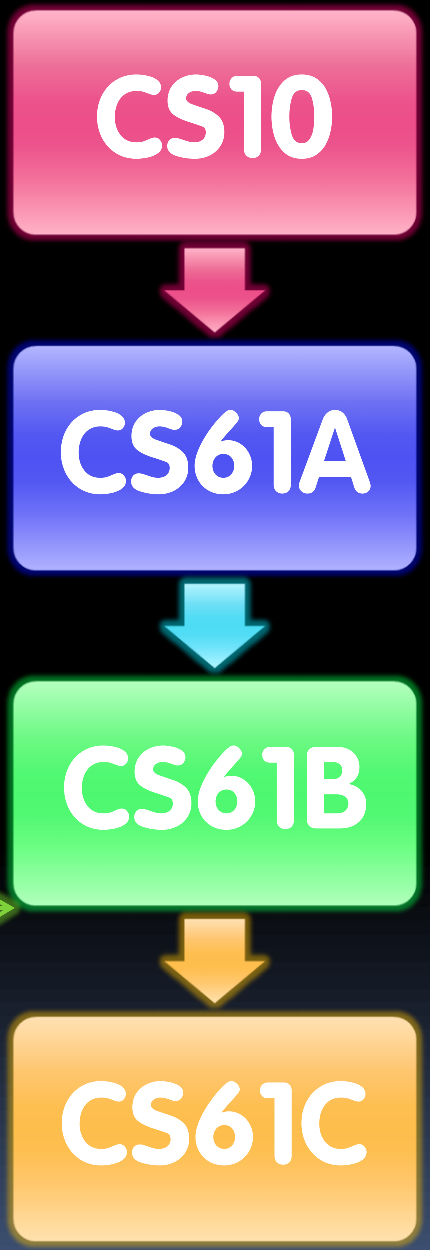
- One commonly used criterion in algorithm analysis is **running time**
  - how long does the algorithm take to run and finish its task?
- How do we measure it?

Runtime Analysis: Problem and Solution

- Time with stopwatch, but…
  - Different computers may have different runtimes. ☹
  - Same computer may have different runtimes on same input. ☹
  - Need to implement the algorithm first to run it. ☹
- *Solution*: Count the #of "steps" involved, not time!
  - Each operation = 1 step
  - *If we say "running time", we mean # of steps, not time!*

Runtime Analysis: Input Size & Efficiency

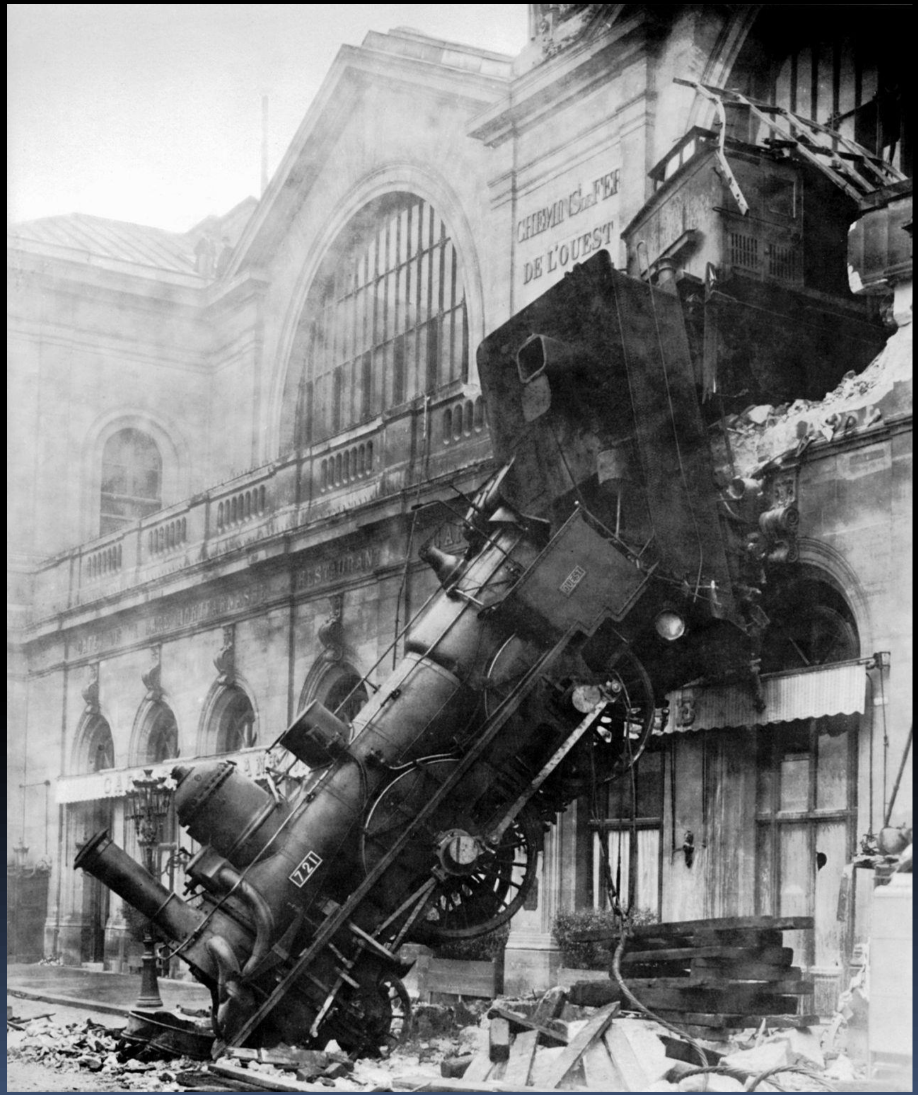
- Given # of input things
  - E.g., # of list elements
  - E.g., # of sentence characters
- We define efficiency as a function of the input size
  - Running time (# of steps)
  - Memory Usage
- We determine efficiency by reasoning formally or mathematically
- Remember!
  - In CS10 we won't care about the efficiency of your solutions!
  - Usually they care in CS61B



Runtime Analysis: Worst Case Scenario

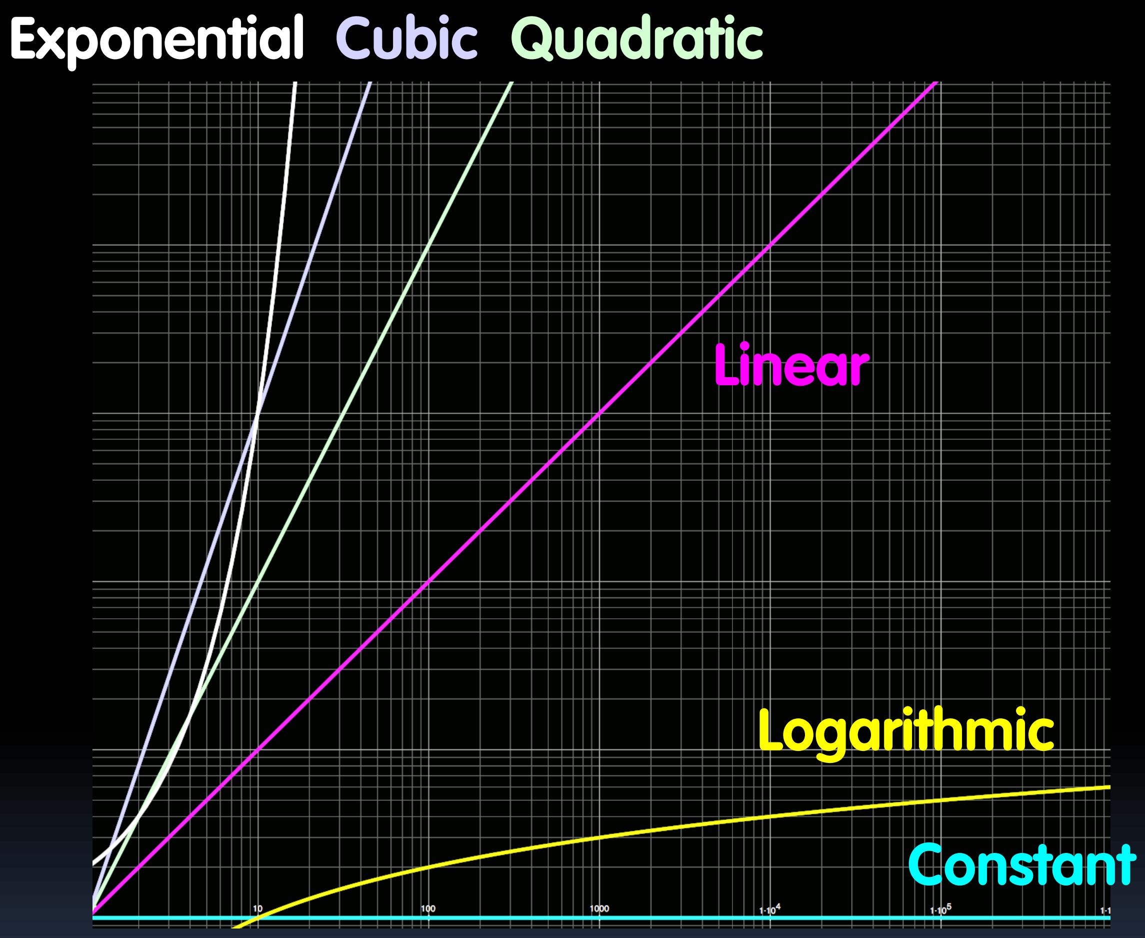
- Use worst case
  - Consider running time as input grows towards infinity
- Why?
  - Nice to know most time we'd <u>ever</u> spend
  - Worst case happens more often than you think

Montparnasse Derailment
(*Wikipedia*, Public Domain)



Runtime Analysis: Order of Growth

- Instead of an exact number of operations we'll use abstraction!
  - Want **order of growth**. As n grows larger and larger, dominant term will eclipse the other terms.
- In CS10 we'll consider
  - Constant
  - Logarithmic
  - Linear
  - Quadratic
  - Cubic
  - Exponential
- E.g. $10\ n^2 + 4 \log n + n$
  - …is quadratic



Graph of order of growth curves on log-log plot

McKinsey

Example: Finding a student (by ID)

- Input
  - Unsorted list of students L
  - Particular student S
- Output
  - True if S is in L, else False
- Pseudocode Algorithm
  - Go through one by one, checking for match.
  - If match, true
  - If exhausted L and didn't find S, false

- Worst-case running time as function of the size of L?

1. Constant
2. Logarithmic
3. Linear
4. Quadratic
5. Exponential

Example: Finding a student (by ID)

- Input
  - Sorted list of students L
  - Particular student S
- Output : same
- Pseudocode Algorithm
  - Start in middle
  - If match, report true
  - If exhausted, throw away half of L and check again in the middle of remaining part of L
  - If nobody left, report false

- Worst-case running time as function of the size of L?

1. Constant
2. Logarithmic
3. Linear
4. Quadratic
5. Exponential

McKinsey

Example: Finding a student (by ID)

- What if L were given to you in advance and you had infinite storage?
  - Could you do any better than logarithmic?

- Worst-case running time as function of the size of L?
  1. Constant
  2. Logarithmic
  3. Linear
  4. Quadratic
  5. Exponential

Example: Finding a Shared Birthday

- Input
  - Unsorted list L (of size n) of birthdays of team
- Output
  - True if any two people shared birthday, else False
- What's the worst-case running time?

- Worst-case running time as function of the size of L?
  1. Constant
  2. Logarithmic
  3. Linear
  4. Quadratic
  5. Exponential

McKinsey

Example: Finding Subsets

- Input:
  - Unsorted list L (of size n) of people
- Output
  - All the subsets
- Worst-case running time? (as function of n)
- E.g., for 3 people (a,b,c):
  - 1 empty: { }
  - 3 1-person: {a, b, c}
  - 3 2-person: {ab, bc, ac}
  - 1 3-person: {abc}

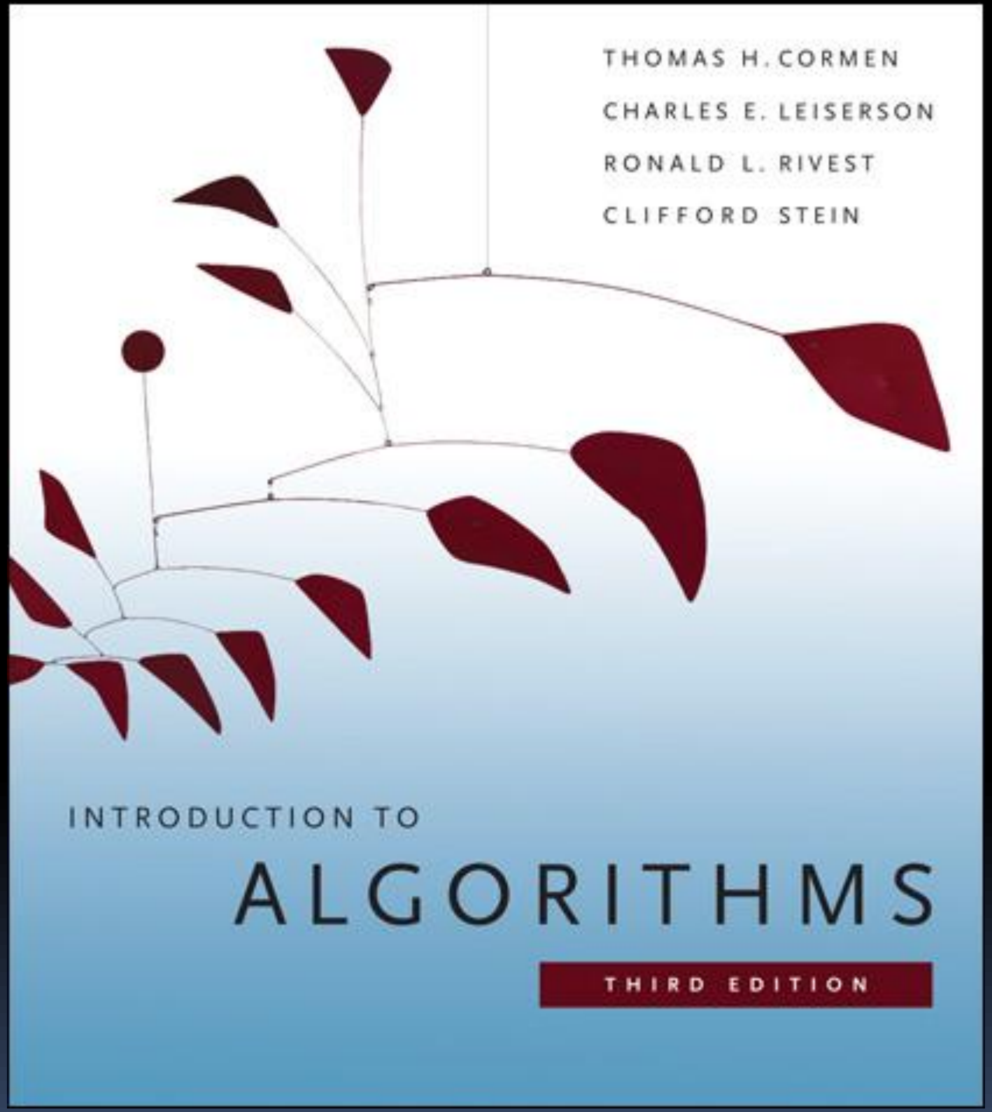
- Worst-case running time as function of the size of L?

1. Constant
2. Logarithmic
3. Linear
4. Quadratic
5. Exponential

Algorithms: Correctness, Summary

Reference Text for Algorithms

- This book launched a generation of CS students into Algorithm Analysis
  - It's on everyone's shelf
  - It might be hard to grok at this point, but if you go on in CS, remember it & own it!
  - …but get the most recent edition

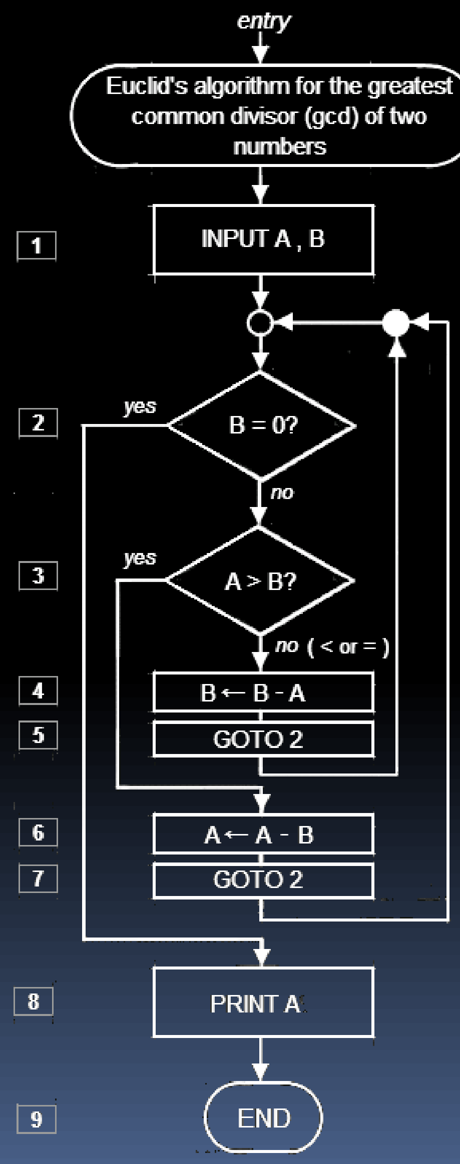


McKinsey

Algorithm Analysis: Is an Algorithm Correct?

- An algorithm is **correct** if, for every input, it reports the correct output <u>and</u> doesn't run forever or cause an error.
  - Incorrect algorithms may run forever, or may crash, or may not return the correct answer.
    - They could still be useful!
    - Consider an approximation…
  - For now, we'll only consider <u>correct</u> algorithms

Euclid's GCD Algorithm
(*Wikimedia*)



McKinsey