

The Beauty and Joy of Computing

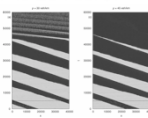
Lecture #7
Algorithmic Complexity

Sr Lecturer SOE
Dan Garcia

Quest (first exam) in 7 days!!
In this room!

IN-CAR ALGORITHM COULD DISSOLVE TRAFFIC!

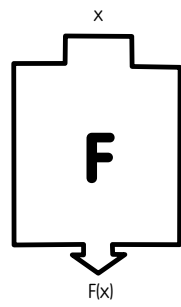
"If cars broadcast their speeds to other vehicles" ... (and the speeds of cars were automatically controlled – you could still steer) ... "a simple in-car algorithm could help dissolve traffic jams as soon as they occur!". Key idea – be optimistic leaving the jam and defensive leading into it.



www.technologyreview.com/blog/axiv/27166/

Functional Abstraction (review)

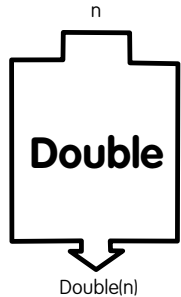
- A block, or function has inputs & outputs
 - Possibly no inputs
 - Possibly no outputs (if block is a command)
 - In this case, it would have a "side effect", i.e., what it does (e.g., move a robot)
- The contract describing what that block does is called a specification or spec



UC Berkeley "The Beauty and Joy of Computing": Algorithmic Complexity (2)

What is IN a spec? (review)


- 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: n + n



UC Berkeley "The Beauty and Joy of Computing": Algorithmic Complexity (3)

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
 - 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)!




UC Berkeley "The Beauty and Joy of Computing": Algorithmic Complexity (4)

What do YOU think?

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

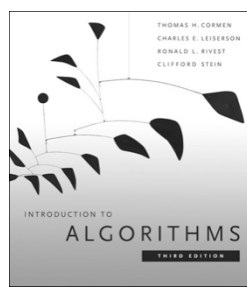
- Simplest?
- Easiest to implement?
- Takes less time?
- Uses up less space (memory)?
- Gives a more precise answer?



UC Berkeley "The Beauty and Joy of Computing": Algorithmic Complexity (5)

Reference text

- 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 vers



UC Berkeley "The Beauty and Joy of Computing": Algorithmic Complexity (6)

Algorithm analysis : the basics

- An algorithm is correct if, for every input, it reports the correct output and 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 correct algorithms

Algorithm for managing Vitamin D sterols based on serum calcium levels.
www.kidney.org/professionals/kdoqi/guidelines_bone/guidells.htm

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (7)

Algorithm analysis : running time

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

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (8)

Runtime analysis problem & solution

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

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (9)

Runtime analysis : input size & efficiency

- Definition**
 - Input size: the # of things in the input. CS10
 - E.g., # of things in a list
 - Running time as a function of input size CS61A
 - Measures efficiency
- Important!**
 - In CS10 we won't care about the efficiency of your solutions!
 - ...in CS61B we will CS61B
 - CS61C

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (10)

Runtime analysis : worst or avg case?

- Could use avg case
 - Average running time over a vast # of inputs
- Instead: use worst case
 - Consider running time as input grows
- Why?
 - Nice to know most time we'd ever spend
 - Worst case happens often
 - Avg is often ~ worst

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (11)

Runtime analysis: Final abstraction



- Instead of an exact number of operations we'll use abstraction
 - Want order of growth, or dominant term
- In CS10 we'll consider
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Cubic
 - Exponential
- E.g. $10n^2 + 4 \log n + n$
 - ...is quadratic

Graph of order of growth curves on log-log plot

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (12)

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?**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Exponential

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (13)

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?**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Exponential

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (14)

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?**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Exponential

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (15)

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
- Worst-case running time as function of the size of L?**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Exponential
- What's the worst-case running time?**

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (16)

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?**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Exponential

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (17)

Summary

- When choosing algorithm, could optimize for**
 - Simplest
 - Easiest to implement?
 - Most efficient
 - Uses up least resources
 - Gives most precision
 - ...
- In CS10 we'll consider**
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Cubic
 - Exponential



How does the goalie choose how to block ball?

UC Berkeley "The Beauty and Joy of Computing" : Algorithmic Complexity (18)