Class 22: Classy Complexity Classes

Office hours note: my office hours tomorrow will be 10-11am in my office.

PS6 (the last one) is posted now and will be due Thursday, April 24.

cs302: Theory of Computation
University of Virginia Computer Science

David Evans
http://www.cs.virginia.edu/evans

Menu

- Why 2150?
- Asymptotic Analysis
- Mind vs. Turing Machine (from PS5)
- Complexity Class **P**
- Complexity Class **NP**

From last class:

Computability

Undecidable

Decidable

~1800s – 1960s

1900: Hilbert's Problems
1936: Turing's *Computable Numbers*
1957: Chomsky's *Syntactic Structures*

(Mostly) "Dead" field

Complexity

Intractable

Tractable

1960s – 2150?

1960s: Hartmanis and Stearns: Complexity class
1971: Cook/Levin, Karp: $P=NP$?
1976: Knuth's O, Ω, Θ

Very Open and Alive

Predicting Knowledge

- In golden age fields, knowledge doubles every 15 years (read Neil DeGrasse Tyson's *Science's Endless Golden Age*)
- Hence, in 2158, we should know ~1024 times (10 doublings) what we know today
- So, guessing it will end in ~2150 implies:
  - Computational Complexity is a finite field
  - What we know today is about 1/1000th what there is

I don't know if either of these is true, but they seem like reasonable guesses…

Asymptotic Notation Recap

- Big-O: $f \in O(g)$: no faster than

  if there exist $c, n_0 > 0$ such that

  $f(n) \leq cg(n)$ for all $n \geq n_0$.
- Omega: $f \in \Omega(g)$: no slower than

  if there exist $c, n_0 > 0$ such that

  $f(n) \geq cg(n)$ for all $n \geq n_0$.
- Theta: $f \in \Theta(g)$ iff $f \in O(g)$ and $f \in \Omega(g)$

Little-o: $<$ instead of $\leq$

Algorithm Analysis

- What is the asymptotic running time of the Java-ish procedure below:

```
int gaussSum (int m) {
  int sum = 0;
  for (int i = 1; i <= m; i++) {
     sum = sum + i;
  }
  return sum;
}
```

Good "cs201/cs216" answer: $\Theta(n)$

What does this mean?
What does it assume?

Algorithm Analysis

- gaussSum is order n: "A function that outputs the running time of gaussSum when the input is the *value* of the input is in $\Theta(n)$.

```
int gaussSum (int m) {
  int sum = 0;
  for (int i = 1; i <= m; i++) {
    sum = sum + i;
  }
  return sum;
}
```

Assumes:
m is unbounded (not true for real Java)
+ is constant time (not true if m is unbounded)

Note that these assumptions are mutually inconsistent so the answer is "wrong" (but useful).

"Correct"ish Answers

```
int gaussSum (int m) {
  int sum = 0;
  for (int i = 1; i <= m; i++) {
    sum = sum + i;
  }
  return sum;
}
```

Assume m is bounded (e.g., 32-bit integer as in real Java). Then, running time of gaussSum is in $O(1)$.

Assume m is unbounded. Then, the average running time of the + is in $\Theta(\log m)$, so the running time of gaussSum is in $\Theta(m \log m)$.

What are we *really* measuring?

- Input size: number of tape squares it takes to write down the input
- Running time: number of steps it takes before TM enters a final state
- Input size for gaussSum = $\log m$
  - Number of bits to represent m (not its magnitude)
  - Note: if we used unary it would be size m

Why doesn't log base matter in asymptotic notations?

Most Correct Answer

```
int gaussSum (int m) {
  int sum = 0;
  for (int i = 1; i <= m; i++) {
    sum = sum + i;
  }
  return sum;
}
```

Is $\boldsymbol{\Theta}(2^N N) = \boldsymbol{\Theta}(2^N)$?

Left as small challenge problem (everyone should be able to answer this using definition of $\boldsymbol{\Theta}$.)

Assume the size of the input N is unbounded. Then, $m \sim 2^N$. The running time of + is in $\Theta(\log m) = \Theta(N)$ so the running time of gaussSum is in $\boldsymbol{\Theta}(2^N N)$ = where N is the *size* of the input.

Algorithm Analysis

```
int gaussSum (int m) {
  int sum = 0;
  for (int i = 1; i <= m; i++) {
    sum = sum + i; }
  return sum;
}
```

cs201/cs216 answer: $\Theta(n)$ where n is the value of the input

cs302 answer: in $\boldsymbol{\Theta}(2^N N)$ where N is the *size* of the input.

cs432 answer: don't analyze Java code, analyze idealized pseudocode and state assumptions clearly.

gaussSum Problem

- So, what is the time complexity of the gaussSum *problem*?

  Input: a positive integer *m*

  Output: sum of the integers from 1 to *m*.

From the previous analysis, we know an algorithm that solves it with running time in $\boldsymbol{\Theta}(N2^N)$.

This means the time complexity of the *problem* is in $O(N2^N)$. But it does not give a tight bound.

gaussSum Problem

- Can we get a lower bound?

  Input: a positive integer *m*

  Output: sum of the integers from 1 to *m*.

At a minimum, we need to look at each symbol in the input. So, there is no algorithm asymptotically faster than $\Theta(N)$.

This means the time complexity of the *problem* is in $\Omega(N)$. But it does not give a tight bound.

gaussSum Problem

- Can we get a tight bound?

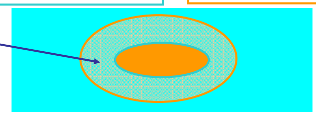
  Input: a positive integer *m*

  Output: sum of the integers from 1 to *m*.

The time complexity of the *problem* is in $\Omega(N)$.

The time complexity of the *problem* is in $O(N2^N)$.

Ring of possibilities



Is there a Θ bound?

Getting a Tighter Bound

Johann Carl Friedrich Gauss, 1777-1855

gaussSum(n) = (n + 1)(n/2)

What is the fastest known multiplication algorithm?

Until 2007: **Schönhage-Strassen algorithm** in $\Theta(N \log N \log\log N)$

Today: **Fűrer's algorithm** in $\Theta(N \log N\, 2^{O(\log^* N)})$

Tomorrow: unknown if there is a faster multiplication algorithm

Best Known Bounds

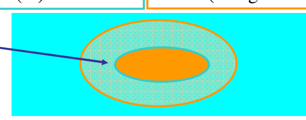
Input: a positive integer *m*

Output: sum of the integers from 1 to *m*.

The time complexity of the *problem* is in $\Omega(N)$.

The time complexity of the *problem* is in $O(N \log N\, 2^{O(\log^* N)})$.

Ring of possibilities



Getting a tight bound for a problem is **very** hard!
Need to prove you have the best possible algorithm.

Minds vs. Turing Machines

Problem Set 5, Question 6: Many people find the suggestion that a human mind is no more powerful than a Turing Machine to be disturbing, but there appear to be strong arguments supporting this position. … Write a short essay that counters this argument (although many books have been written on this question, you should limit your response to no more than one page). If you reject the premise of this question either because you do not find it disturbing to think of your mind as a Turing Machine, or you feel that the only way to counter this argument is to resort to supernatural (e.g., religious) notions, you may replace this question with Sipser's Problem 5.13.

About 1/5 chose to replace question.

Most Common Answer: Randomness

- "...human brain can create true randomness"
- "The outputs of neurons do NOT deterministically depend on the inputs because of quantum uncertainty."

Self-Modification

- "… Humans can even learn enough from the world around them to alter their own programming."
- "…a TM cannot adapt, and has no way to change its own rules or states."

Recall a Universal TM can simulate every other TM. So, it is certainly possible for a TM to simulate a TM that changes rules and states in response to the input.

Self-Awareness

Humans are more "cognizant" of their shortcomings than TMs. There are several problems that humans understand are impossible to answer, but no TM can simulate the decision that any of these problems are decidedly unsolvable.

David Horres

I can't help but quote from South Park: "You see, the basis of all reasoning is the mind's awareness of itself. What we think, the external objects we perceive, are all like actors that come on and off stage. But our consciousness, the stage itself, is always present to us." (Kyle)

Hung-in Lam

Memory Access

Our brains have the ability to recognize patterns and then use those patterns to filter new data. Since our brains store memories primarily through association rather than just memory addresses, this allows for an integrated, relational system of memories…. Our memories "fade" over time, yet can occasionally be brought back…

Eric Montgomery

Real Time/World Interactions

"ability to interact with the surrounding environment"

A mind can interact with physical inputs and outputs in real time. The brain is able to make decisions in real time; either a synapse fires or it doesn't. What would happen in a brain model that waited forever for a single binary decision? Would the brain-simulating TM ever be able to make all the decisions necessary for even the tiniest slice of time? Such a TM would probably be eaten by a hungry woof; how embarrassing for such a smart machine!

Rachel Miller

…ability of the human mind to process analog inputs. The possible ranges for sound or light are infinite, and are not stored digitally in the brain. …

Chris Dodge

Some physicists think space-time can be quantized at about 10^{-23} meters and 10^{-32} seconds. So, in theory a TM could process analog inputs, but in practice all the atoms in the universe would not be enough to make the tape for the TM…

Resilience

"The brain can function without some of its components, but a TM cannot…."

Jalysa Conway

"The human mind is also capable of breaking out of an infinite loop that a TM would be stuck in forever… a person gets bored, something that no TM can emulate. "

Timothy Kang

Neurons tend to fire in a synchronized way. A group of neurons in one part of the brain, for example, may light up at the same time and cause another group to activate in another region. Finally, neurons are capable of neurogenesis, the creation of new brain cells. A TM, no matter how much use it gets, will always remain a TM. The brain, however, is a muscle that is influenced by many factors, including usage. In fact, even the eldest of living humans can avoid mental breakdown by simply exercising their brains frequently…

Christopher Andersen

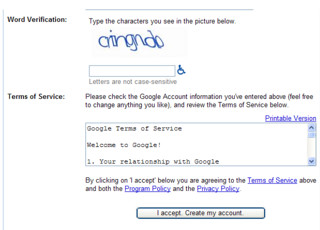
Note: exercising your brain is a good idea for young humans also!

Would it be useful to have a computational problem that humans can solve but computers cannot solve?

link

link

Thwarting Spammers, Annoying Humans



CAPTCHA: *Completely Automated Public Turing Test to Tell Computers and Humans Apart*

Luis von Ahn, Manuel Blum and John Langford. Telling Humans and Computers Apart Automatically.

Complexity Class **P**

Non-Robustness of TM Complexity

- Computability: all variations on TMs have the same computing power
  - If there is a multi-tape TM that can decide L, there is a regular TM that can decide L.
  - If there is a nondeterministic TM that can decide L, there is a deterministic TM that can decide L.
- Complexity: variations on TM can solve problems in different times
  - Is a multi-tape TM *faster* than a regular TM?
  - Is a nondeterministic TM *faster* than a regular TM?

Multi-Tape vs. One-Tape TM

Are there problems that are in $\mathbf{TIME}(t(n))$ for a multi-tape TM, but not in $\mathbf{TIME}(t(n))$ for a one-tape TM?

Copy Input Problem

Input: w, a string of N bits
Output: ww

Obvious multi-tape algorithm that involves $2N$ steps:
N steps: walk over the input, copying it to the second tape
N steps: continue to move right, copying the second tape contents onto the input tape after the input

Best (?) single-tape algorithm that involves ~$2N^2$ steps:
N iterations: move over the input, marking each symbol
N steps: move to the first non-blank square, write that symbol
N steps: move back to the rightmost marked input symbol

Intuitively is seems impossible to do much better, but hard to prove!

Theory is about Big Questions

If little tweaks to our model change the answers, we might as well focus on answering the practical questions for a real system and specific problem instance instead.

Making things Robustier?

- Find a more robust computing model than a TM
  - Church-Turing thesis says all mechanical models are equivalent (computing power) to a TM
  - But, this doesn't mean there might not be better models for complexity
- Make the complexity classes bigger
  - Define a complexity class big enough so the little tweaks to TMs do not change the answers

Complexity Class **P**

$$\mathbf{P} = \bigcup_k \mathrm{TIME}(N^k)$$

P is the class of languages that can be decided in **Polynomial Time** on a deterministic, single-tape Turing machine.

Classes in **P**

a) $\mathrm{TIME}(N^2)$
b) $\mathrm{TIME}(O(N^7))$
c) ~~$\mathrm{TIME}(O(2^N))$~~
d) Class of languages that can be decided in Polynomial Time by a 2-tape TM
e) Class of languages that can be decided in Polynomial Time by a nondeterministic TM

Yes! We can simulate each step of a 2-tape TM by making 2 passes over the whole tape ~ $2(N+t(n))$ (See Theorem 7.8)

Unknown! This is the P = NP question. Focus of next class...

Charge

- PS6 is now posted, due Thursday, April 24
- Office hours tomorrow are in my office, 10-11am
- Read Sipser Chapter 7
  - It is not expected to understand the proof of the Cook-Levin Theorem (pages 277-282)
- Thursday (Isabelle Stanton):
  - Restating the P = NP question
  - How do we make progress in answering it?