Unit 7: Theory

Dave Abel

March 12th, 2016
As Promised:

Any Midterm questions?

Also: one side of a 3”x 5” index card
AI Update!

AlphaGo 3-1 against Lee Sedol

(He did it! Woohoo!)
Theory

› Revisiting Growth Rates

› Problem Classes
  - SOLVE
  - VERIFY

› The biggest unanswered question in Computer Science!
  - Implications

› Unsolvable problems

› Uncountable things

› Measuring Simplicity, Occam’s Razor
Theory

› Revisiting Growth Rates

› Problem Classes
  - SOLVE
  - VERIFY

› ***The biggest unanswered question in Computer Science!***
  - Implications

› Unsolvable problems

› Uncountable things

› Measuring Simplicity, Occam’s Razor
Some problems are unsolvable, period.
Theory: Takeaway

Some problems are unsolvable, period.
Theory: Takeaway

Some problems are unsolvable, period.

We can characterize and relate how hard each and every problem is by dividing problems into classes.
Some problems are unsolvable, period.

We can characterize and relate how hard each and every problem is by dividing problems into classes.

Q: Are problems in VERIFY, also in SOLVE?

This is the biggest unanswered question in computer science.
Growth Rate: Definition

1. **Definition:** The *growth rate* of an algorithm is the number of primitive operations an algorithm must execute, in the worst case, in order to complete its job.

2. We call it the *growth rate* because it’s how the number of operations the computer has to execute *grows* as the size of our input grows.

I.e. sort a length 2 list vs. sorting a length 203487 list
Growth Rate: Definition

1. **Definition:** The *growth rate* of an algorithm is the number of primitive operations an algorithm must execute, in the worst case, in order to complete its job.

2. We call it the *growth rate* because it’s how the number of operations the computer has to execute *grows* as the size of our input grows.

I.e. sort a length 2 list vs. sorting a length 203487 list

Reminder: *In the worst case!*
Revisiting Growth Rates

Remember Random Search? It took way longer with a longer list.
Revisiting Growth Rates

Q: What, if anything, is out here?

Remember Random Search? It took way longer with a longer list.
Remember Random Search? It took way longer with a longer list.

Q: What, if anything, is out here?

Things that can be computed, period.

Things a regular computer can compute before the sun goes supernova.

Things a domino computer could compute before the sun goes supernova.

random search, big list
Problem Specification Example

- **INPUT**: Map of solar system, description of physical laws, summary of current technology.

- **OUTPUT**: A method for colonizing Mars.
Revisiting Growth Rates

Remember Random Search? It took way longer with a longer list.

Q: What, if anything, is out here?

Things a regular computer can compute before the sun goes supernova

Things a domino computer could compute before the sun goes supernova

Things that can be computed, period.

Mars!
Revisiting Growth Rates

Remember Random Search? It took way longer with a longer list.

Q: What, if anything, is out here?

Mars!

Things a regular computer can compute before the sun goes supernova

Things that can be computed, period.

Things a domino computer could compute before the sun goes supernova
Growth Rates: The Point

Remember Random Search? It took way longer with a longer list.

The Point: we want to know how many things we have to do as our input grows, because we want to know what problems are solvable before the sun goes poof! (and which ones will take the drop of a hat)
Growth Rates

- So far, we’ve talked about the growth rates of *algorithms*, e.g. Binary Search, Selection Sort:
  - Linear Search:
Growth Rates

- So far, we’ve talked about the growth rates of *algorithms*, e.g. Binary Search, Selection Sort:
  - Linear Search: $N$
  - Binary Search:
Growth Rates

- So far, we’ve talked about the growth rates of algorithms, e.g. Binary Search, Selection Sort:
  - Linear Search: $N$
  - Binary Search: $\log(N)$
  - Selection Sort:
Growth Rates

- So far, we’ve talked about the growth rates of algorithms, e.g. Binary Search, Selection Sort:
  - Linear Search: $N$
  - Binary Search: $\log(N)$
  - Selection Sort: $N^2$
  - Build the Truth Table:
Growth Rates

- So far, we’ve talked about the growth rates of algorithms, e.g. Binary Search, Selection Sort:
  - Linear Search: $N$
  - Binary Search: $\log(N)$
  - Selection Sort: $N^2$
  - Build the Truth Table: $2^N$
Growth Rates

- So far, we’ve talked about the growth rates of *algorithms*, e.g. Binary Search, Selection Sort:
  - Linear Search: $N$
  - Binary Search: $\log(N)$
  - Selection Sort: $N^2$
  - Build the Truth Table: $2^N$

- But don’t we care about *problems*?
Growth Rates

- But don’t we care about *problems*?

- What if random search were the only way to search we had discovered so far?
Growth Rates

- But don’t we care about *problems*?
- What if random search were the only way to search we had discovered so far?
- This isn’t quite the relevant bit..
Growth Rates

- But don’t we care about *problems*?

- What if random search were the only way to search we had discovered so far?

- This isn’t quite the relevant bit..

- What we *really ought to care* about is how fast we can solve the problem *search*, period.
Growth Rates

‣ What we really ought to care about is how fast we can solve the problem search, period.

‣ What we’ll talk about in this unit is:

Q: For a given problem, what’s the fastest possible algorithm for solving that problem?
Growth Rates

‣ What we really ought to care about is how fast we can solve the problem search, period.

‣ What we’ll talk about in this unit is:

Q: For a given problem, what’s the fastest possible algorithm for solving that problem?

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

Things that can be computed, period.

Things a regular computer can compute before the sun goes supernova

Things a domino computer could compute before the sun goes supernova

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose *fastest* (correct) algorithm have growth rates of at most $N$

Q: Moreover, what’s the *growth rate* of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose fastest (correct) algorithm have growth rates of at most $N$

Q: Is Search in this?

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose *fastest* (correct) algorithm have growth rates of at most $N$

Q: Is Search in this?

A: Yes! If this list is sorted, we get $\log(N)$ from Binary Search, if unsorted, $N$ from Linear Search

Q: Moreover, what’s the *growth rate* of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose *fastest* (correct) algorithm have growth rates of at most $N$

Q: What other problems are in here?

Q: Moreover, what’s the *growth rate* of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose fastest (correct) algorithm have growth rates of at most $N$

Computing the median?

Q: What other problems are in here?

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose *fastest* (correct) algorithm have growth rates of at most $N$

Computing the median!

Q: What other problems are in here?

Sorting?

Q: Moreover, what’s the *growth rate* of the fastest (correct) algorithm for solving each problem?
Growth Rates

Problems whose fastest (correct) algorithm have growth rates of at most $N$

Computing the median!

Q: What other problems are in here?

Sorting...

Q: Moreover, what’s the growth rate of the fastest (correct) algorithm for solving each problem?
Growth Rates

We can do this for *all relevant growth rates.*
Growth Rates

We can do this for all relevant growth rates:

1. Linear or faster \((N, \log(N), \text{etc.})\)
Growth Rates

We can do this for all relevant growth rates:

1. Linear or faster \((N, \log(N), \text{etc.})\)

2. Polynomial or faster \((N^5, N^{20}, N, \text{etc.})\)
Growth Rates

We can do this for all relevant growth rates:

1. Linear or faster \((N, \log(N), \text{ etc.})\)

2. Polynomial or faster \((N^5, N^{20}, N, \text{ etc.})\)

Example: \(5N^3 + 2N^2 + N\)

Example: \(8N^{27} + 9N^5\)
Growth Rates

We can do this for all relevant growth rates:

1. Linear or faster ($N$, $\log(N)$, etc.)

2. Polynomial or faster ($N^5$, $N^{20}$, $N$, etc.)

Example: $N^3$

Example: $N^{27}$
Growth Rates

We can do this for all relevant growth rates:

1. Linear or faster ($N, \log(N), \text{ etc.}$)
2. Polynomial or faster ($N^5, N^{20}, N, \text{ etc.}$)
3. Exponential or faster ($2^N, 3^N, \text{ etc.}$)
Growth Rates

- Linear
- Polynomial
- Exponential
Growth Rates

Reminder: these are the problems that have algorithms whose growth rates are \textit{at most} Linear, Polynomial, etc.
Growth Rates

Important: polynomial is green because that’s the class of problems we consider solvable.
Class: SOLVE

Important: polynomial is green because that’s the class of problems we consider solvable.
Reminder: Exponentials are BIG

\[2^{100} = 1,267,650,600,228,229,401,496,703,205,376\]
Clicker Question:

Q: Is the *Sorting* problem in SOLVE?
Clicker Question:

Q: Is the *Sorting* problem in SOLVE?

[A] Yes!

[B] No!

[C] I’m confused :/
Q: Is the *Sorting* problem in SOLVE?

[A] Yes!

[B] No!
Clicker Answer:

Q: Is the *Sorting* problem in SOLVE?

[A] Yes!

[B] No!

Reminder: SOLVE are the problems that have algorithms whose growth rates are *at most* Polynomial.
Q: Is the *Sorting* problem in SOLVE?

[A] Yes!  

[B] No!

Reminder: SOLVE are the problems that have algorithms whose growth rates are *at most* Polynomial.
Q: Can we solve a problem efficiently?

A: Is it in SOLVE?
Reminder:

- Problem Specification:
  - INPUT: some things
  - OUTPUT: some true stuff about the things
Reminder:

› Problem Specification:
  - INPUT: some things
  - OUTPUT: some true stuff about the things

› Example:
  - INPUT: A Sudoku board
  - OUTPUT: Solution to the Sudoku board
Another View: Verification

- Verification Example:
  - INPUT: An empty Sudoku board, a proposed solution to that Sudoku board
  - OUTPUT: True if the Sudoku board is a correct solution

```
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```
Another View: Verification

- Another Verification Example:
  - INPUT: An empty Crossword, a proposed solution to that Crossword
  - OUTPUT: True if the filled out Crossword board is a correct solution
Another View: Verification

- Another Verification Example:
  - INPUT: A list, a proposed sorting of that list
  - OUTPUT: True if the proposed sorting is actually in sorted order.
Discuss!

- In light of recent events consider how *making the perfect single move* in the Game Go can be pitched as a verification problem!
Discuss!

In light of recent events consider how making the perfect single move in the Game Go can be pitched as a verification problem!

Talk with your neighbors for a minute or two
Discuss!

- In light of recent events consider how making the perfect single move in the Game Go can be pitched as a verification problem!

**INPUT:** A configuration of the Go board, a Go move

**OUTPUT:** True if the move is the perfect move.
Another Class: VERIFY

The class of problems VERIFY is the set of problems where we can verify solutions.