CPSC 365 / ECON 365:

Algorithms

Lecture 1: Introduction

Andre Wibisono

Yale University

January 25, 2022

Plan

Logistics

Algorithms

Math Review

Course Information

Welcome to CPSC 365 / ECON 365: **Algorithms**

Yale University, Spring 2022

Lectures:

- TIME: Tuesday + Thursday, 2:30 pm – 3:45 pm
- PLACE: Dunham Lab 220 (first two weeks virtual)
- VIRTUAL: `https://yale.zoom.us/j/94057532462`
- INSTRUCTOR: Andre Wibisono

Canvas: `https://yale.instructure.com/courses/73872`

Website: `https://cpsc365.github.io`

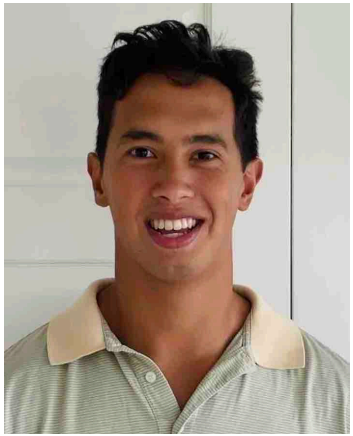
Instructor

Andre Wibisono
andre.wibisono@yale.edu

Assistant Professor
Computer Science Department
Yale University

Office Hours: Tuesday 4–5 PM
https://yale.zoom.us/my/wibisono

Course Staff

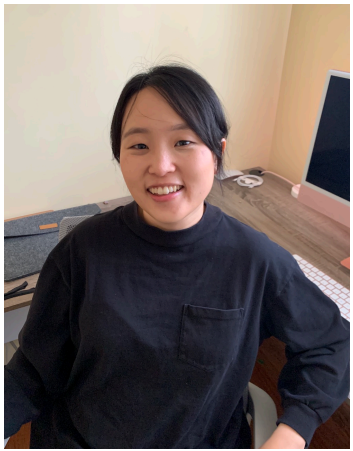


Teaching Fellow:

John Lazarsfeld

`john.lazarsfeld@yale.edu`

Office Hours: By appointment
In charge of: Content, assignments



Course Manager:

Inyoung Shin

`inyoung.shin@yale.edu`

Office Hours: By appointment
In charge of: Logistics, scheduling

ULAs

Alex Chen
Mon 8:30-10pm

Michelle Goh
Sun 1-2:30pm

Eric Xue
Sun 8:30 -10pm

Nevin George
Mon 8:30-10pm

Adit Gupta
Thur 7-8:30pm

Andrew Yuan
Mon 7-8:30pm

Michal Gerasimiuk
Fri 6-7:30pm

Alex Tan
Wed 3-4:30pm

Matthew Zhang
Sun 7-8:30pm

Rohit Giridharan
Sat 1-2:30pm

Andrew Wei
Mon 7-8:30pm

Course Format

Each week:

- **Lectures:** Tuesday and Thursday, 2:30 – 3:45 pm
- **Discussion:** Friday, 1 hour, multiple sections
- **Office Hours:** Multiple time slots

Discussion Sections

- New: Weekly 1-hour discussion sections on Friday led by ULAs
- In each section, a ULA leads a group of students on exercises and problem solving
- Is it optional? Yes.

  But highly recommended: Practice for problem solving, writing solutions, interacting with ULAs and students.
- We will assign a section time for you based on your response to the form: `https://forms.gle/bxgf5BpLpGVQq8H57`
- Discussion starts **next Friday, Feb 4, 2022**
- This Friday, Jan 28, 2022: Review session by TF (see Canvas)

Grading

- $7 \times$ Problem sets $= 60\%$
  - PS1 + PS2 $= 2 \times 5\%$
  - PS3 to PS7 $= 5 \times 10\%$
- $2 \times$ Midterms $= 40\%$
  - Midterm 1 $= 20\%$
  - Midterm 2 $= 20\%$

Midterms

- 2 midterms, in class
- Midterm 1: March 10, 2022
- Midterm 2: April 28, 2022

Problem Sets

- 7 problem sets, approximately one every two weeks
- PS 1 out, due next Tuesday, Feb 1, 2022 at 2:30 PM

  Background topics, make sure you are comfortable with them
- Submit solution via Gradescope.

  Please make sure to assign the correct page in your document corresponding to each problem.
- We recommend writing your solution in Latex.

  Handwritten solutions are accepted if they are clearly legible.
- Collaborations are allowed (see policy), but you must first try to solve the problems by yourself.
- Each student has **5 late days** for the semester.

Collaboration Policy for Problem Sets

- You must first make an honest effort to solve the PS by yourself.
- You may discuss hints and work on problems together with other students. But you may not provide nor receive explicit solutions.
- You must write your solution independently.
- You must list all your collaborators, and all resources you consult beyond the course textbook.
- You are forbidden from searching for the PS on the Internet.
- You must also follow the Yale academic integrity policy [link]

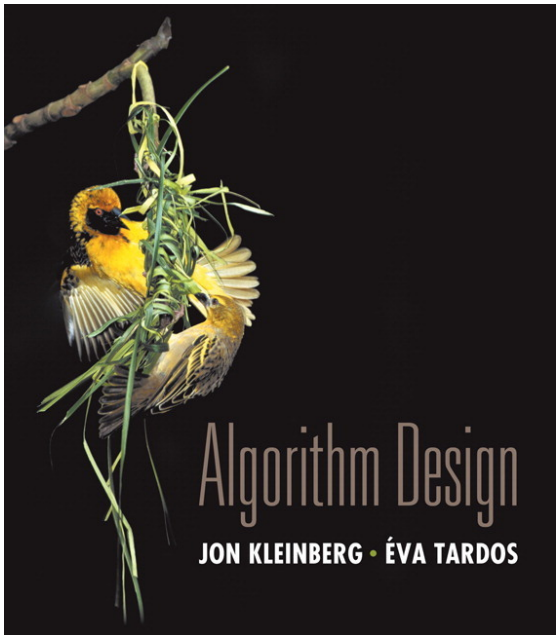
Late Policy for Problem Sets

- Each student has **5 late days** to use for the entire semester.
- Whole day use: If you miss a PS submission by N hours, you will use $\lceil \frac{N}{24} \rceil$ late days.
- Absolutely no late PS submission accepted otherwise.
- Students submitting late PS must sign honor code certifying they have not looked at solution.

Course Description

Course description: Paradigms for algorithmic problem solving: greedy algorithms, divide and conquer, dynamic programming, and network flow.
NP completeness and approximation algorithms for NP-complete problems. Algorithms for problems from economics, scheduling, network design and navigation, geometry, biology, and optimization. Provides algorithmic background essential to further study of computer science.

Textbook



Algorithm Design

Jon Kleinberg and Éva Tardos

ISBN 978-0321295354

Available via Course Reserves

All readings will be from textbook

Syllabus

Modules we will cover:

1. Review
2. Greedy methods
3. Dynamic programming
4. Divide and conquer
5. Network flow
6. Complexity theory
7. Approximation algorithms

Lecture Plan

Date	#	Lecture	Module	Reading	Problem Set
Jan 25	1	Introduction and Overview	Review 1	1, 2.1 - 2.2, 3.1	PS1 out
Jan 27	2	Stable Matching Problem	Review 2	1, 2.3 - 2.5	
Feb 1	3	Review on Graphs	Review 3	3.2 - 3.6	PS1 in / PS2 out
Feb 3	4	Interval Scheduling Problem	Greedy 1	4.1	
Feb 8	5	Scheduling Problem Variants	Greedy 2	4.2	PS2 in / PS3 out
Feb 10	6	Shortest Path Problem	Greedy 3	4.4	
Feb 15	7	Minimum Spanning Tree	Greedy 4	4.5 - 4.6	
Feb 17	8	Weighted Interval Scheduling Problem	DP 1	6.1 - 6.2	
Feb 22	9	Subset Sums and Knapsacks Problem	DP 2	6.4	PS3 in / PS4 out
Feb 24	10	Shortest Path with Negative Edges	DP 3	6.8, 6.10	
Mar 1	11	Mergesort Algorithm	Divide and Conquer 1	5.1 - 5.2	
Mar 3	12	Integer Multiplication Problem	Divide and Conquer 2	5.5	
Mar 8	13	Median Finding and Quicksort	Divide and Conquer 3	13.5	PS4 in
Mar 10	14	**Midterm 1**			
Mar 15	15	Max Flow Problem	Network Flow 1	7.1	PS5 out
Mar 17	16	Max Flow and Min Cuts	Network Flow 2	7.2 - 7.3	
Mar 29	17	Max Flow for Bipartite Matching	Network Flow 3	7.5, 7.7	PS5 in / PS6 out
Mar 31	18	Polynomial-Time Reductions	Complexity 1	8.1	
Apr 5	19	Satisfiability Problem	Complexity 2	8.2	
Apr 7	20	NP and NP-Complete Problems	Complexity 3	8.3 - 8.4	
Apr 12	21	Traveling Salesman Problem	Complexity 4	8.5	PS6 in / PS7 out
Apr 14	22	PSPACE Problems	Complexity 5	9.1 - 9.3	
Apr 19	23	Load Balancing Problem	Approximation 1	11.1	
Apr 21	24	Set Cover and Vertex Cover Problems	Approximation 2	11.3 - 11.4	
Apr 26	25	PTAS for Knapsack Problem	Approximation 3	11.8	PS7 in
Apr 28	26	**Midterm 2**			

How to succeed in this class?

- **Come to lectures**
  Lectures will be recorded and posted online, but please come to live lectures and participate if possible.
- **Go to discussion section**
  Practice solving problems, interact with ULAs and students
- **Go to office hours** and ask questions
- **Start Problem Sets early**
  Some problems require time to solve. Read PS when it is released. Read solution.
- **Form study groups**
  Form small group of students to study together (but adhere to collaboration policy).
- **Don't fall behind**
  Follow along with the class. This is a conceptual class; proper understanding requires time and practice.

365 vs 366

- CPSC 366 / ECON 366: Intensive Algorithms is a more math + theory-intensive version of 365 (this class).
- Smaller class, with more interesting and challenging problems.
- No collaborations allowed.
- Lectures same time as 365, by Prof. Dan Spielman.

Plan

Logistics

Algorithms

Math Review

What is an Algorithm?

An algorithm is:

- A process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer.
  [Oxford Dictionary]
- A finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation.
  [Wikipedia, Merriam-Webster]

Algorithms in Everyday Life – 1

How to bake an apple pie?

Classic Apple Pie

By Melissa Clark

YIELD 8 servings

TIME 1 1/2 hours, plus cooling

Ryan Liebe for The New York Times. Food Stylist: Erika Joyce.

INGREDIENTS

- **2 tablespoons unsalted butter**
- **3 ½ pounds firm, crisp apples (see Tip), peeled, cored and cut into 1/4-inch wedges (about 11 cups)**
- **½ cup/110 grams light brown sugar**
- **2 tablespoons granulated sugar**
- **1 teaspoon ground cinnamon**
- **½ teaspoon ground ginger**
- **½ teaspoon grated nutmeg**
- **Pinch of ground cloves**
- **¼ teaspoon fine sea salt**
- **2 tablespoons cornstarch**
- **1 ½ teaspoons fresh lemon juice, or a little more if your apples are very sweet**
- **½ teaspoon grated lemon zest**
- **All-purpose flour, for rolling out the dough**
- **Dough for a 9-inch double crust pie**
- **Heavy cream, milk or a beaten egg**

PREPARATION

Step 1

Melt butter in a large skillet set over medium-high heat and add apples to the pan. Stir to coat with butter and cook, stirring occasionally, until the butter is evenly distributed, about 1 minute.

Step 2

In a small bowl, whisk together sugars, spices and salt. Sprinkle over the apples and toss to combine.

Step 3

Lower heat to medium and cook until apples have softened completely but still hold their shape, about 17 to 25 minutes. (Some varieties cook more quickly than others.)

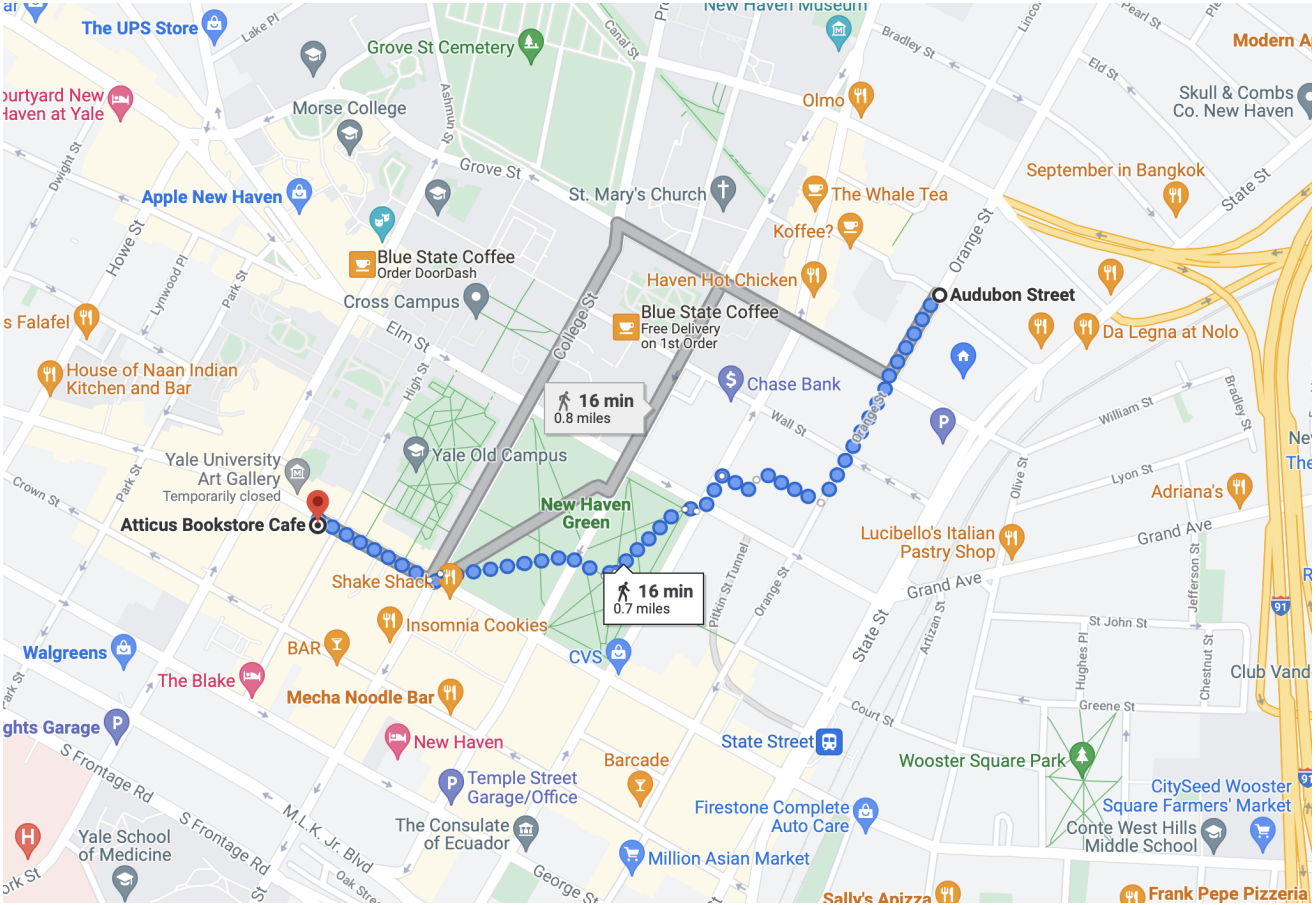
Step 4

Sprinkle cornstarch evenly over the apples and continue to cook, stirring occasionally, until the apple mixture comes to a simmer and thickens slightly, about 2 minutes. Remove pan from heat, and stir in lemon juice and zest. Allow apples to cool completely (spreading them onto a rimmed baking sheet speeds this up). Apples can be prepared up to 24 hours ahead and refrigerated.

https://cooking.nytimes.com/recipes/1022727-classic-apple-pie

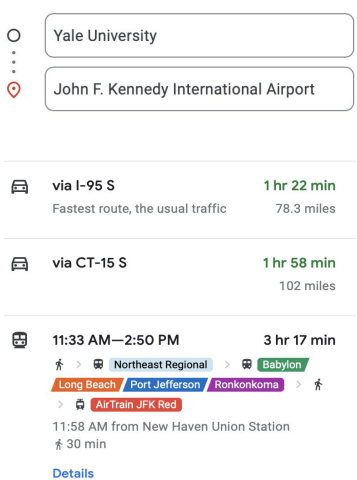
Algorithms in Everyday Life – 2

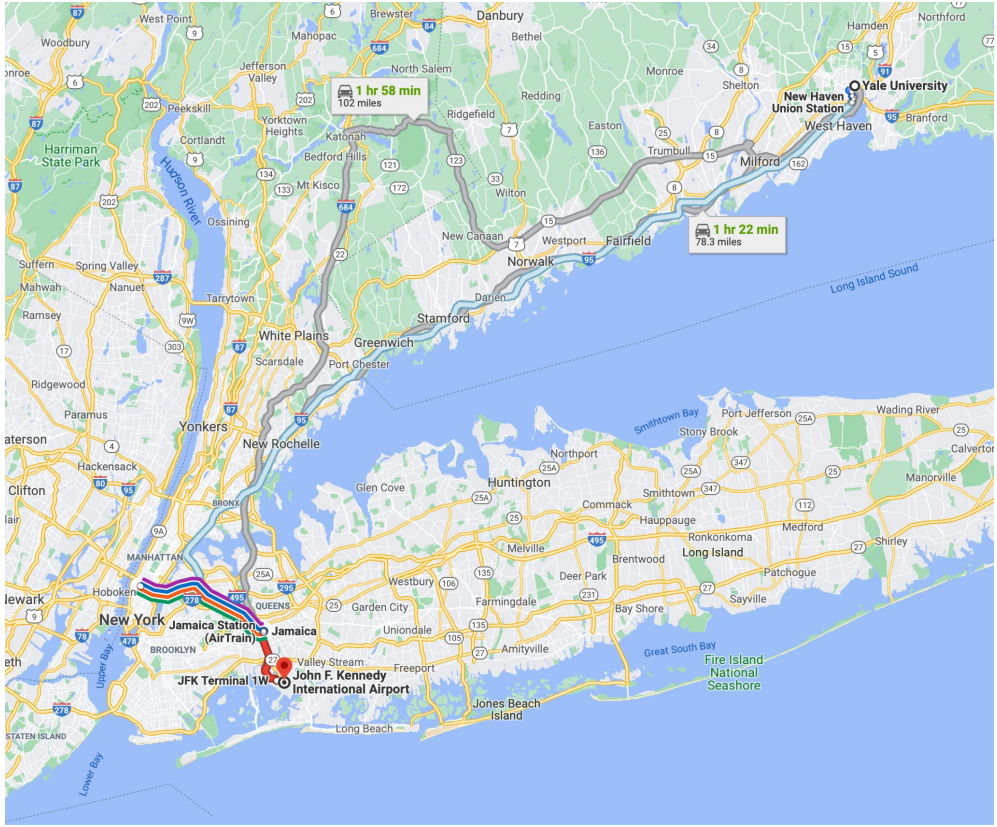
How to get coffee the fastest?



Algorithms in Everyday Life – 3

How to go to the airport the fastest / cheapest / most convenient?



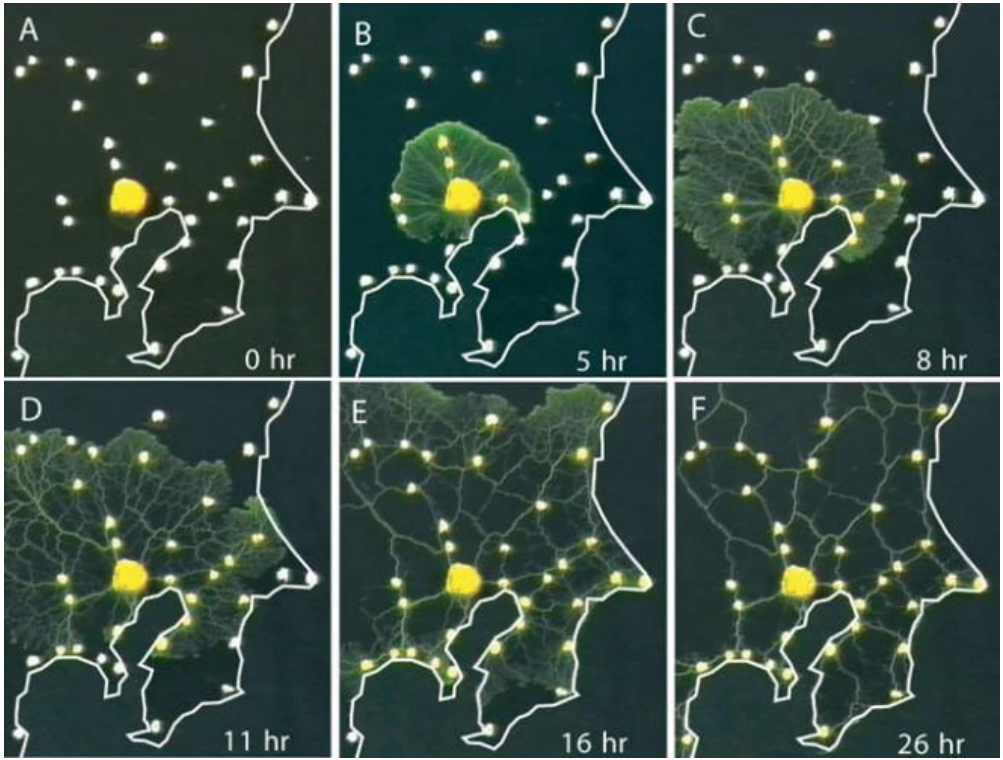


Algorithms in Everyday Life – 4

How to arrange subway lines in Tokyo?

Algorithms in Nature – 1

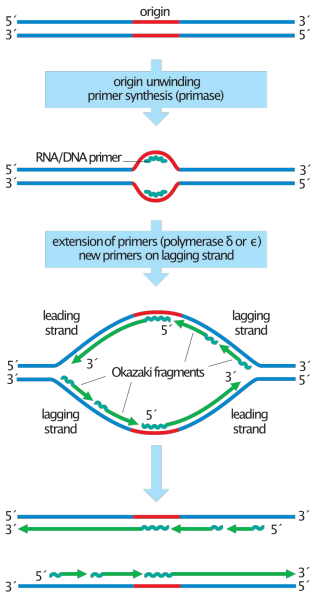
Slime Mold Grows Network Just Like Tokyo Rail System

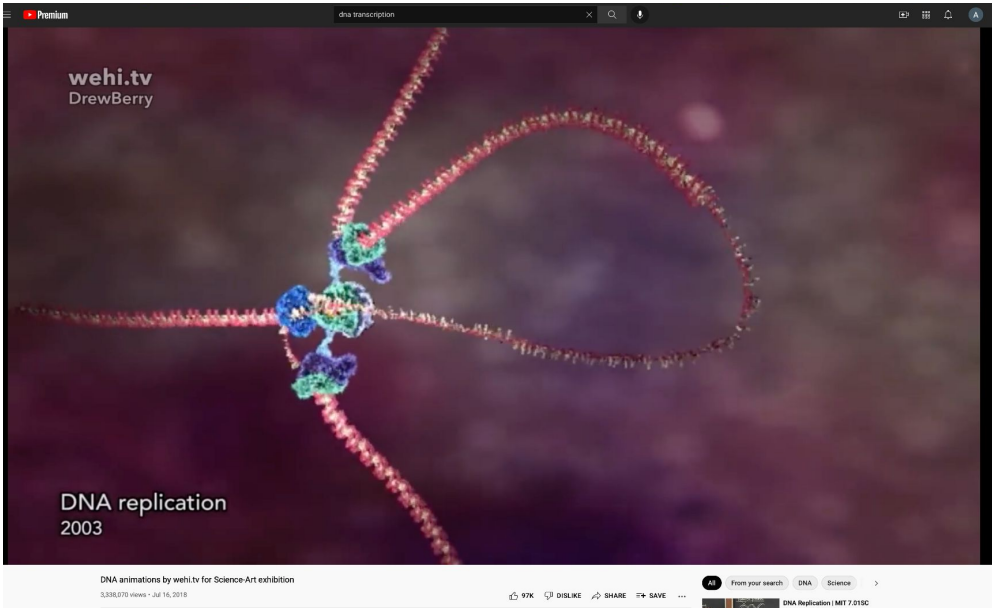


https://www.wired.com/2010/01/slime-mold-grows-network-just-like-tokyo-rail-system/

Algorithms in Nature – 2

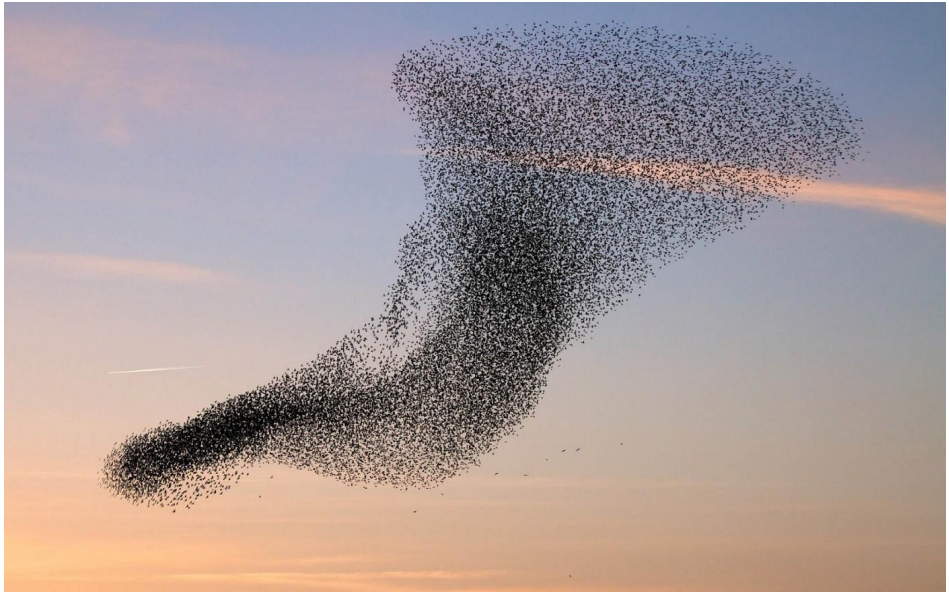
DNA Replication

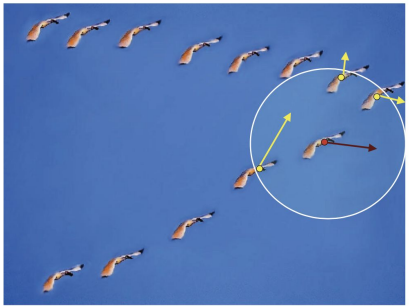


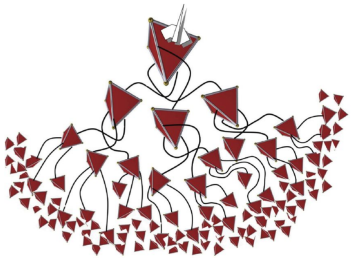


https://youtu.be/7Hk9jct2ozY?t=149

Algorithms in Nature – 3







Bernard Chazelle, *"The Convergence of Bird Flocking"*, Journal of the ACM, Vol 61, Issue 4, 2014

Algorithms

- Procedures to solve computational tasks.
- Usually sequential, involves a repetition of steps.
- Ubiquitous in everyday life and in Nature.
- Computational thinking: Formulate problems as computational tasks, and solve via Algorithms.
- Deep connections to the nature of computation and Mathematics.

This Class

A study of **Algorithms**:

0. How to formulate problems as computational tasks
1. How to design algorithms to solve computational tasks
2. How to analyze correctness, running time of algorithms
3. How to find more efficient algorithms, or understand the computational difficulties of problems

What You Need

A good understanding of background topics:

1. Mathematics: Logic and proofs
2. Discrete mathematics: Graphs, counting, basic probability
3. Data structures
4. Running times and asymptotics

Algorithms through History – 1

1. Arithmetic [Babylonian c. 2500 BC, Egyptian c. 1550 BC]
2. Prime finding (sieve of Eratosthenes) [Greek c. 240 BC]
3. Euclid's algorithm for finding greatest common divisor [Elements, c. 300 BC]
4. Cryptography via frequency analysis [Al Kindi c. 850 AD]
5. Ada Lovelace: First computer program for the Analytical Engine [1842]

Algorithm: Etymology

- Muḥammad ibn Mūsā al-Khwārizmī (of/from Khwarazm in Central Asia)
- al-Khwārizmī ↠ Algorithm
- Also known for treatise on algebra, solving quadratic equation
- Al-jabr (“completion”) ↠ Algebra
- Treatise on Hindu-Arabic numerals

https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi

Numbers and Bases

We can use any base b to represent numbers

* Arabic: $b = 10$

$$365 = 3 \times 10^2 + 6 \times 10 + 5 \times 1$$

* Mayan: $b = 20$
* Babylonian: $b = 60$
* Computer: $b = 2$

This makes addition and arithmetic easy

- c.f. in Roman numeral: MCDXLVIII + DLXXIV = ?

Arithmetic Algorithms: Addition

In base 10:

$$53 + 35 = 88$$

In base 2:

Carry:	1			1	1	1		
		1	1	0	1	0	1	(53)
		1	0	0	0	1	1	(35)
	1	0	1	1	0	0	0	(88)

Arithmetic Algorithms: Multiplication

In base 10:

$$13 \times 11 = 143$$

In base 2:

$$\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
 & & & \times & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}$$

(1101 times 1)
(1101 times 1, shifted once)
(1101 times 0, shifted twice)
(1101 times 1, shifted thrice)
(binary 143)

Plan

Logistics

Algorithms

Math Review

Review Topics

1. **Logic**
2. Proofs
3. Asymptotics

Notation – Sets

1. $\mathbb{N} = \{1, 2, 3, \dots\}$ set of natural numbers
2. $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$ set of integers
3. $\mathbb{Z}_0 = \{0, 1, 2, \dots\} = \mathbb{N} \cup \{0\}$ set of non-negative integers
4. $\mathbb{Q} = \{\frac{a}{b} : a, b \in \mathbb{Z}, b \neq 0\}$ set of rational numbers
5. $\mathbb{R}$ set of real numbers
6. $\mathbb{R}_+ = (0, +\infty)$ set of positive real numbers
7. $\mathbb{R}_0 = [0, +\infty) = \mathbb{R}_+ \cup \{0\}$ set of non-negative real numbers

$$\mathbb{N} \subset \mathbb{Z}_0 \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R}$$
$$\mathbb{R}_+ \subset \mathbb{R}_0 \subset \mathbb{R}$$

Notation – Operators

1. $\forall$: for all
2. $\exists$: there exists
3. $\neg$: not (negation)
4. $\cup$: union
5. $\cap$: intersection
6. $\in$: element of
7. $\subseteq$: subset (can be equal)
8. $\subset$: strict subset (also denoted $\subsetneq$)
9. $\sum$: summation
10. $\prod$: product

Propositional Logic

A **proposition** is a statement that is either True or False

Examples:

1. $1 + 1 = 2$
2. $\mathbb{Z} \subset \mathbb{R}_+$
3. $\sqrt{2} \in \mathbb{Q}$
4. $\forall\ n \in \mathbb{N}, n \geq 3,\ \neg\big(\exists\ x, y, z \in \mathbb{Z} \colon x^n + y^n = z^n\big)$

Propositional Logic

A **proposition** is a statement that is either True or False

Examples:

1. $1 + 1 = 2$ True
2. $\mathbb{Z} \subset \mathbb{R}_+$ False
3. $\sqrt{2} \in \mathbb{Q}$ False
4. $\forall\ n \in \mathbb{N}, n \geq 3,\ \neg\big(\exists\ x, y, z \in \mathbb{Z}\colon x^n + y^n = z^n\big)$ True
   (Fermat's Last Theorem)

Non-examples:

- $2 + 2$ (does not evaluate)
- $x^2 + 4x = 5$ (truth value depends on what x is)

Combining Propositions

Let P, Q be propositions.

1. **Negation (NOT):** $\neg P$
   True when P is False.
2. **Disjunction (OR):** $P \vee Q$
   True when at least one of P and Q is True.
3. **Conjunction (AND):** $P \wedge Q$
   True only when both P and Q are True.
4. **Implication (IMPLIES, $\Rightarrow$):** $P \Rightarrow Q$ (if P, then Q)
   False only when P is True but Q is False.

Combining Propositions – Truth Table

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$
True	True	False	True	True	
True	False	False	True	False	
False	True	True	True	False	
False	False	True	False	False	

Combining Propositions – Truth Table

P	Q	$\neg P$	$P \vee Q$	$P \wedge Q$	$P \Rightarrow Q$
True	True	False	True	True	True
True	False	False	True	False	False
False	True	True	True	False	True
False	False	True	False	False	True

Note on implication:

- $P \Rightarrow Q$ is a *promise* that if P happens, then Q also happens
- The promise is only broken (False) when P happens but Q does not
- If P does not happen, then the promise is (vacuously) kept

Quiz 1

Which of the following are equivalent to $P \Rightarrow Q$?
Select all that apply.

(a) $\neg P \Rightarrow \neg Q$

(b) $\neg Q \Rightarrow \neg P$

(c) $Q \Rightarrow P$

(d) $\neg P \vee Q$

Quiz 1

Which of the following are equivalent to $P \Rightarrow Q$?
Select all that apply.

(a) $\neg P \Rightarrow \neg Q$
No. This is the converse.

(b) $\neg Q \Rightarrow \neg P$
Yes. This is the contrapositive.

(c) $Q \Rightarrow P$
No. This is the converse (also contrapositive of (a)).

(d) $\neg P \vee Q$
Yes. This is equivalent to the definition.

Contrapositive and Converse

Consider the implication $P \Rightarrow Q$

- This is equivalent to the **contrapositive**: $\neg Q \Rightarrow \neg P$
- This is *not* equivalent to the **converse**: $Q \Rightarrow P$

Example

Suppose we are walking outside without an umbrella. Let:

$$P = \text{“it is raining”}$$
$$Q = \text{“my clothes are wet”}$$

- $P \Rightarrow Q$: If it is raining, then my clothes are wet.
  True
- $\neg Q \Rightarrow \neg P$: If my clothes are dry, then it is not raining.
  True
- $Q \Rightarrow P$: If my clothes are wet, then it is raining.
- $\neg P \Rightarrow \neg Q$: If it is not raining, then my clothes are not wet.
- $\neg P \vee Q$: Either it is not raining, or my clothes are wet.

Example

Suppose we are walking outside without an umbrella. Let:

$$P = \text{“it is raining”}$$
$$Q = \text{“my clothes are wet”}$$

- $P \Rightarrow Q$: If it is raining, then my clothes are wet.
  Suppose this is True.
- $\neg Q \Rightarrow \neg P$: If my clothes are dry, then it is not raining.
  This is also True.
- $Q \Rightarrow P$: If my clothes are wet, then it is raining.
  This can be False (e.g. car splash).
- $\neg P \Rightarrow \neg Q$: If it is not raining, then my clothes are not wet.
  This can be False (e.g. car splash).
- $\neg P \vee Q$: Either it is not raining, or my clothes are wet.
  This is True.

Negation and Distributive Law

1. $\neg(\neg P) \equiv P$
2. $\neg(P \vee Q) \equiv \neg P \wedge \neg Q$
3. $\neg(P \wedge Q) \equiv \neg P \vee \neg Q$
4. $\neg(\forall x\ P(x)) \equiv \exists x\ \neg P(x)$
5. $\neg(\exists x\ P(x)) \equiv \forall x\ \neg P(x)$

Equivalence

Definition: P is **equivalent** to Q, denoted $P \Leftrightarrow Q$, if both $P \Rightarrow Q$ and $Q \Rightarrow P$ hold.

$$P \Leftrightarrow Q \quad \equiv \quad (P \Rightarrow Q) \wedge (Q \Rightarrow P)$$

Example:

$$P = \text{“}G \text{ is a bipartite graph”}$$
$$Q = \text{“}G \text{ has no cycles of odd length”}$$

PS1: Prove that $P \Leftrightarrow Q$

Review Topics

1. Logic
2. **Proofs**
3. Asymptotics

Proofs

Suppose we have a statement we want to show, e.g.:

$$\forall x \in \mathcal{X} : P(x) \Rightarrow Q(x)$$

- To prove the statement, give a sequence of logical deductions to go from assuming $P(x)$ is True to showing $Q(x)$ is True.
- To disprove the statement, give a counterexample (an example $x \in \mathcal{X}$ such that $P(x)$ is True but $Q(x)$ is False).

Proof Techniques

1. Direct proof
2. Proof by contrapositive
3. Proof by contradiction
4. Proof by induction
5. Proof by cases

⋮

1 - Direct Proof

Goal: To prove $P \Rightarrow Q$

Steps:

- Assume P
- ...
- Therefore Q

2 - Proof by Contrapositive

Goal: To prove $P \Rightarrow Q$

Steps:

- Assume $\neg Q$
- ...
- Therefore $\neg P$

Conclusion: $\neg Q \Rightarrow \neg P$, which is equivalent to $P \Rightarrow Q$

2 - Proof by Contrapositive: Example

Problem: Let $d, n \in \mathbb{N}$ and suppose d divides n.
Show that if n is odd, then d is odd.

3 - Proof by Contradiction

Goal: To prove P

Steps:

- Assume $\neg P$
- ...
- Show Q
- ...
- Also show $\neg Q$, a contradiction.

Conclusion: $\neg P \Rightarrow Q \wedge \neg Q = \text{False}$. This is equivalent to $\text{True} \Rightarrow P$, hence P.

3 - Proof by Contradiction: Example

Problem: Show that $\sqrt{2} \notin \mathbb{Q}$.

4 - Proof by Induction

Goal: To prove $\forall\, n \in \mathbb{N}\colon P(n)$

Steps:

- Base case: Show $P(1)$
- Induction hypothesis: Assume $P(n-1)$ for some $n \geq 2$
- Induction step: Show $P(n)$

$$P(1) \;\Rightarrow\; P(2) \;\Rightarrow\; P(3) \;\Rightarrow\; P(4) \;\Rightarrow\; \ldots$$

5 - Proof by Cases

Goal: To prove $\forall x \in \mathcal{X} : P(x)$

Steps:

- Partition the domain into subsets (cases): $\mathcal{X} = A_1 \cup \cdots \cup A_n$
- For each case $i = 1, \ldots, n$, prove $\forall x \in A_i : P(x)$

Proof Fallacies – 1

Claim: $-2 = 2$

Proof(?): Assume $-2 = 2$.
Squaring both sides, we have $(-2)^2 = 2^2$, or $4 = 4$, which is True.
We conclude that $-2 = 2$, as desired. □

Proof Fallacies – 2

Claim: $1 = 2$

Proof(?): Assume that $x = y$ for some $x, y \in \mathbb{Z}$. Then,

$$\begin{aligned} x^2 - xy &= x^2 - y^2 && \text{(since } x = y\text{)} \\ x(x-y) &= (x+y)(x-y) \\ x &= x + y && \text{(divide both sides by } x - y\text{)} \\ x &= 2x. \end{aligned}$$

Setting $x = y = 1$ yields the claim. □

Review Topics

1. Logic
2. Proofs
3. **Asymptotics**

Asymptotic Notation

Let $f(n)$, $g(n)$ be functions of $n \in \mathbb{N}$
(for example, running times of two algorithms on inputs of size n)

- We say $f = O(g)$ if there exist $C > 0$, $N \in \mathbb{N}$ such that for all $n \geq N$, we have $f(n) \leq C \cdot g(n)$.

  Equivalently, $\frac{f(n)}{g(n)}$ is bounded above by a constant over $n \in \mathbb{N}$.

- We say $f = \Omega(g)$ if $g = \Omega(f)$.
- We say $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

Orders of Growth

- **Polynomial:** $n,\ n^2,\ n^3,\ n^k\ (k > 0)$

  If input doubles, output is multiplied by a constant:

$$(2n)^k = 2^k \cdot n^k$$

- **Exponential:** $2^n,\ 3^n,\ e^n,\ b^n\ (b > 0)$

  If input doubles, output is squared:

$$b^{2n} = (b^n)^2$$

- **Logarithm:** $\log_2 n,\ \log n,\ \log_b n\ (b > 0)$

  If input doubles, output is added by a constant:

$$\log(2n) = \log n + \log 2$$

Exponential

Euler's number:

$$e = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = 2.71828...$$

For all $b > 0$, $m, n > 0$:

- $b^{m+n} = b^m \cdot b^n$
- $(b^m)^n = b^{mn}$
- $b^n = e^{n \log b}$

Logarithm

Logarithm is inverse function of exponential

$$\log_b n = x \quad \Leftrightarrow \quad b^x = n$$

- $b > 0$ is *base*
- By default, $\log = \log_e$. Will also use $\log_2$.
- For any $a, b > 0$: $\log_b n = \dfrac{\log_a n}{\log_a b}$. So $\log_b n = O(\log n)$.
- $\log(mn) = \log m + \log n$

Asymptotic Notation: Rules

Rules to help simplify calculations.

- Multiplicative constants can be omitted: $365n^2$ becomes n^2
- n^a dominates n^b if $a > b$: for example, n^2 dominates n
- Any exponential dominates any polynomial: 2^n dominates n^{10}
- Polynomial dominates logarithm: $n^{0.1}$ dominates $(\log n)^{10}$

Exercise: Prove the above.

Basic Recursion

Let $f(n)$ be a function of $n \in \mathbb{N}$. Verify that:

- If $f(n) = f(n-1) + 1$, then $f(n) = \Theta(n)$
- If $f(n) = f(n-1) + n$, then $f(n) = \Theta(n^2)$
- If $f(n) = 2f(n-1) + 1$, then $f(n) = \Theta(2^n)$
- If $f(n) = f(\frac{n}{2}) + 1$, then $f(n) = \Theta(\log n)$

Basic Counting

Let S be a set with n elements.

- Factorial: $n! = n \cdot (n-1) \cdots 2 \cdot 1$
  This is the number of permutations of the elements of S
- 2^n = the number of subsets (of any size) of S
- n^2 = the number of ordered pairs of elements of S
- $\binom{n}{2} = \frac{n(n-1)}{2}$ = the number of unordered pairs of distinct elements of S
- $\lceil \log_2 n \rceil$ = the number of times to halve n to reach 1

Next Lecture

- Stable Matching Problem
- Reading: 1, 2.3 - 2.5