Believe It or Not, Another Semester

SIGma

Outline

Σ

Updates!

Weekly updates:

- We're back!
- `#research-advice` in Discord
- `#seminars` in Discord
- Come make meetings

Section 1

Admins in No Particular Order

Sam

- CS PhD
- Doing Computational Geometry with Sariel Har-Peled
- SIGPwny

Hassam

- CS Major (takes math classes for fun ???)
- SIGPwny Crypto Gang + Admin team + Infra lead
- CA for CS 341, CS 173
- Compiler research (paper accepted ASPLOS '24 !!!)
- Graduating (and subsequently selling out) this semester

Ryan

- Not selling out yet - currently a junior in CS
- CA for CS{222, 374, 461}
- Interest in algorithmic game theory and fair division, currently working on approximation algorithms for division
- Attempting to do hardware

Alex

- Stats&CS, Math Double Major
- Part of PeopleWeave Research Project
- CA for CS 225, CS 374
- Foosball Pro

Porter

- CS Major, Math Minor
- Intern at CDK Global over the summer
- CA for CS 128H
- Foosball Pro

Anakin

- Doing research in ICLUE with Alexander Yong
- Did Computational Group Theory at an REU
- Used to do Graph Theory / Optimization Research with Sam
- SIGPwny Crypto[1] Gang + Admin team
- Coffee Club
- CA for CS 173 + CS 225H, former 374 + 475

[1] Not that one, the other one

Come Make Meetings!

Brand New: *Short and Sweet Presentations*

- 3 presentations each day
- 10-15 minutes long (*Short*)
- Probably some food and drink (*Sweet*)
- March 18th & April 22nd
- Good way to show you are interested in being a future admin

Join Discord + DM any @admin if interested

Section 2

P vs NP

Complexity Classes

- *Complexity Classes* are groups of problems that are characterized by being of the same "difficulty"
- "Difficulty" refers to big-O notation: If an algorithm runs in $O(n^2)$ time, that means as the input size n tends to infinity, the algorithm takes cn^2 time for some constant c
- Most people think about two classes:
  - problems with polynomial time algorithms
  - problems requiring larger-than-polynomial time algorithms

Decision Problems

- These are problems that have *yes or no* answers
- Examples:
  - Is this list sorted?
  - Does this graph have a path visiting every node exactly once (Hamiltonian path)?
  - Is this number prime?

Decision Problems

- The typical computation model is a Turing Machine
  - For all intents and purposes, a normal computer + your favorite programming language[2]
- The input size of the problem is usually part of the problem specification
  - Is this list of length n sorted?
  - Does this graph with n nodes have a Hamiltonian path?
  - Is this n-bit binary number prime?

[2] Unless your favorite language is HTML

Solving vs Verifying

- For decision problems, the two big paradigms are *solving* the problem and *verifying* a solution
- Consider the problem of finding a Hamiltonian path in a graph with n nodes:
  - *solving*: Try all $n!$ orderings of nodes, see if any of them are a Hamiltonian path. Runs in $O(n * n!)$ time
  - *verifying:* Given a candidate path, check if it is a Hamiltonian path. Runs in $O(n)$ time

The Million Dollar Question

- In the year 2000, the Clay Mathematics Institute posed 7 Prize Problems where the people who found the solutions would get \$1,000,000
- The P vs NP problem is about whether or not the following two complexity classes are equal:
  - P: The set of decision problems with polynomial time algorithms to *solve* the problem
  - NP: The set of decision problems with polynomial time algorithms to *verify* a solution

Section 3

Circuits Like You've Never Seen Before

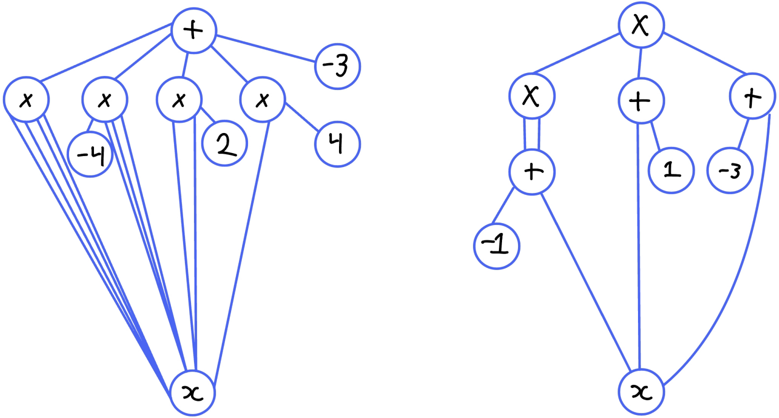
A New (Old) Approach to P vs NP

- One of the most influencial papers in complexity theory is "Completeness classes in algebra" by Valient [Val79]
- He proposed an *algebraic* approach to the P vs NP problem
- This has been furthered by Mulmuley and Sohini and is currently seen as the most viable approach to resolving P vs NP
- *Idea:* Polynomials are our computational model

Σ

Algebraic Circuits

$$x^4 - 4x^3 + 2x^2 + 4x - 3 = (x-1)^2(x+1)(x-3)$$



Complexity

- There are two main measures of complexity of an algebraic circuit:
  - *Size:* Number of nodes
  - *Depth:* Length of the longest path from an input to the output gate
- Given a polynomial f, we can ask two types of questions:
  - Can we construct a circuit for f of small {size/depth}?
  - Can we show no such small circuit for f exists?

Complexity

Recall the normal computational model for a problem with input size n:

- P: Decision problems we can *solve* in poly(n) time.
- NP: Decision problems whose solution we can *verify* in poly(n) time.
- P $\subseteq$ NP
- P $\subsetneq$ NP: Million Dollar Question

Now consider a polynomial f of degree d. We define *Valient's P / NP*:

- VP: f has a poly(d) size circuit to *compute it*
- VNP: Given some monomial, we can *find the coefficient* of the monomial in f with a poly(d) size circuit
- VP $\subseteq$ VNP

A Tale of Two Polynomials

Consider an $n \times n$ matrix X with entries $x_{i,j}$

$$\det(X) = \sum_{i=1}^{n} (-1)^i \cdot x_{1,i} \cdot \det(X_{-1,-i}) \quad \mathrm{perm}(X) = \sum_{i=1}^{n} x_{1,i} \cdot \mathrm{perm}(X_{-1,-i})$$

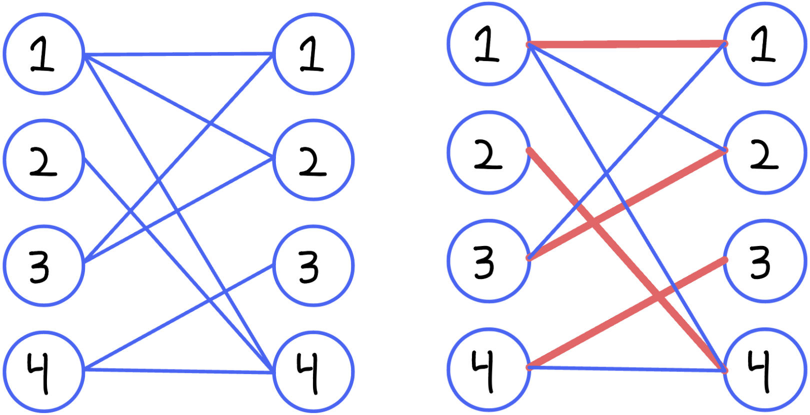
- $\det(X) \in \mathrm{VP}$: Gaussian Elimination gives $O(n^3)$ size
- $\det(X) \in \mathrm{VNP}$: $\mathrm{VP} \subseteq \mathrm{VNP}$
- $\mathrm{perm}(X) \in \mathrm{VNP}$: Just trust me
- $\mathrm{perm}(X) \notin \mathrm{VP}$: Algebraic Million Dollar Question

If VP = VNP and the (generalized) Riemann Hypothesis holds, then "P = NP" [Bö0]

Section 4

Matchings in Parallel

A Match Made in Heaven



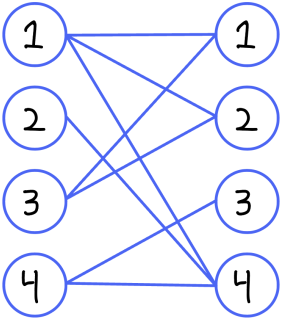
Algebraic Computation

- Determining if there is a perfect matching in a bipartite graph can be determined in polynomial time
- We will see how to do this with an algorithm that can be *parallelized*

Turning a Graph into a Polynomial

Given a graph $G = (V, E)$ with n vertices labeled $1, \ldots, n$, let X be a matrix such that

$$X_{i,j} = \begin{cases} x_{i,j} & \text{if } i \leftrightarrow j \in E \\ 0 & \text{if } i \leftrightarrow j \notin E \end{cases}$$



$$\begin{pmatrix} x_{1,1} & x_{1,2} & 0 & x_{1,4} \\ 0 & 0 & 0 & x_{2,4} \\ x_{3,1} & x_{3,2} & 0 & 0 \\ 0 & 0 & x_{4,3} & x_{4,4} \end{pmatrix}$$

Alternate Formulation

Consider an $n \times n$ matrix X with entries $x_{i,j}$

$$\det(X) = \sum_{i=1}^{n} (-1)^i \cdot x_{1,i} \cdot \det(X_{-1,-i}) \quad \operatorname{perm}(X) = \sum_{i=1}^{n} x_{1,i} \cdot \operatorname{perm}(X_{-1,-i})$$

$$\det(X) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) x_{1,\sigma(1)} \cdots x_{n,\sigma(n)} \quad \operatorname{perm}(X) = \sum_{\sigma \in S_n} x_{1,\sigma(1)} \cdots x_{n,\sigma(n)}$$

- S_n = set of all orderings of integers $1, \ldots, n$
- An *inversion* in σ is if $i < j$ and $\sigma(i) > \sigma(j)$
- $$\operatorname{sgn}(\sigma) = \begin{cases} 1 & \text{if number of inversions in } \sigma \text{ is even} \\ -1 & \text{if number of inversions in } \sigma \text{ is odd} \end{cases}$$

The Final Algorithm

- Let X be the matrix from before constructed from G
- *Claim:* $\det(X) \neq 0 \iff G$ has a perfect matching
  - *Proof:* Think about which orderings σ correspond to matchings
- Recall that a non-zero degree d polynomial has at most d zeroes
- $\det(X)$ has degree $\leq n^2$

$\underline{\textsc{Match}(G)}$
$X \leftarrow$ matrix constructed from G
$f(\overline{x}) \leftarrow \det(X)$
$P \xleftarrow{\$} n^2 + 1$ distinct random points from $\mathbb{R}^{n^2}$
for $\overline{p} \in P$:
    if $f(\overline{p}) \neq 0$:
        return $\textsc{True}$
return $\textsc{False}$

Questions?

Algorithms are for people who don't know how to buy RAM

— Clay Shirky

Bibliography I

Peter Bürgisser.

Cook's versus valiant's hypothesis.

Theoretical Computer Science, 235(1):71–88, March 2000.

L. G. Valiant.

Completeness classes in algebra.

In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, page 249–261, New York, NY, USA, 1979. Association for Computing Machinery.