Math 480A: Algebraic Complexity Theory

Jarod Alper

University of Washington
Spring 2019

April 1, 2019

Motivating Questions

- Why are some problems easy to solve and others are hard?
- How can you decide if a problem is easy or hard?

Showing that a problem is easy is easy: you need to find an efficient algorithm.

Showing that a problem is hard is hard: you need to show that there does not exist an efficient algorithm.

P vs NP

P

Define P as the set of problems that can be solved efficiently.

Example: Calculate the greatest common divisor of two integers.

NP

Define NP as the set of problems that can be verified efficiently.

Example: Determine whether a given integer n is not a prime integer.
For instance, is 21079 prime? Not sure?
Easier question: is 21079 divisible by 107? Yes, $21079 = 107 * 197$.

The holy grail of theoretical computer science.

Is $P = NP$?

Can we be more precise?

- What exactly do we mean by a problem?
- What does it mean to solve a problem efficiently?

To answer these questions we need to introduce a formalism for computation.

What is a problem?

Let $\{0,1\}^*$ be the set of all strings of 0's and 1's.

Example: 0101110 and 00010100000100

We define a problem to be a subset of $\{0,1\}^*$.

Example: Let PRIME be the set of prime integers written out in binary. That is, $\mathrm{PRIME} = \{\underbrace{10}_{2}, \underbrace{11}_{3}, \underbrace{101}_{5}, \underbrace{111}_{7}, \underbrace{1011}_{11}, \underbrace{1101}_{13}, \ldots\}$.

A solution to a problem $A \subset \{0,1\}^*$ is an algorithm to determine whether a given string of 0's and 1's belongs to the subset A.

What do we mean by an algorithm? We need to introduce a computational model.

Turing Machines: invented in 1936

Alan Turing

Image from wikipedia.org

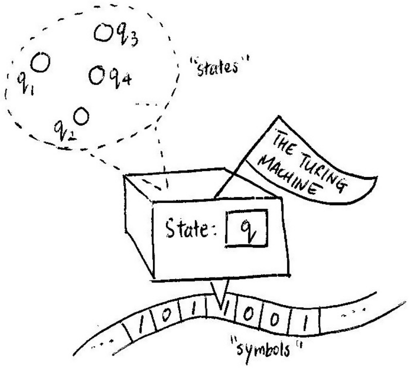


Image from plus.maths.org

P and NP again

P

We define P as the set of problems $A \subset \{0,1\}^*$ for which there exists a Turing machine that can decide whether a given string is in A in polynomial time.

By polynomial time, we mean that there is some polynomial $p(n)$ (such as $4n^3$ or n^{2019}) such that the number of steps the Turing machine takes on an input of length n is less than $p(n)$.

NP

We define NP as the set of problems $A \subset \{0,1\}^*$ for which there exists a Turing machine that can verify whether a given string is in A in polynomial time.

By verify, we mean that the Turing machine has access to a "guess" which it may use to determine if the string is in A.

The million dollar question: Is P = NP?

This is difficult. Two reasons why:

- While showing a problem is in P may be easy, it is very challenging to show that a problem is not in P.
- There is not much mathematical structure in the formulation of the P vs NP problem. Thus we can't apply many of our favorite techniques from geometry and algebra.

Instead let's try to solve a simpler question!

Algebraic complexity theory

In 1978, Leslie Valiant introduced an algebraic approach to complexity theory.

Leslie Valiant

Image from wikipedia.org

Instead of solving problems, we will try to compute the value of polynomials.

VP vs VNP

Consider a sequence of polynomials

$$f_1(x_1), f_2(x_1, x_2), \ldots, f_n(x_1, \ldots, x_n), \ldots$$

Here f_n is a polynomial in the n variables $x_1, \ldots, x_n$.
Examples:

- $f_1 = x_1, \quad f_2 = x_1 x_2, \quad \ldots, \quad f_n = x_1 \cdots x_n, \ldots.$
- $f_1 = x_1, \quad f_2 = x_1^2 + x_2^2, \quad \ldots, \quad f_n = x_1^n + \cdots + x_n^n, \ldots.$

VP

Define VP as the set of sequences of polynomials $f_1, f_2, \ldots$ whose value can be computed in polynomial time.

VNP

Define VNP as the set of sequences of polynomials $f_1, f_2, \ldots$ whose coefficients can be computed in polynomial time.

Is VP $=$ VNP?

VP

Define VP as the set of sequences of polynomials $f_1, f_2, \dots$ whose value can be computed in polynomial time.

VNP

Define VNP as the set of sequences of polynomials $f_1, f_2, \dots$ whose coefficients can be computed in polynomial time.

We have VP $\subset$ VNP.

The big question

Is VP $=$ VNP?

This is an algebraic analogue of the P vs NP question. It should be easier.

Hold on: what do we mean by computing a polynomial?

Just as Turing machines were our computation model for solving problems, we need a computational model for polynomials.

I'll give two equivalent models. Here is the first one:

Arithmetic complexity

The arithmetic complexity of a polynomial $f(x_1, \ldots, x_n)$ is the minimum number of operations (additions and multiplications) needed to compute the value.

VP

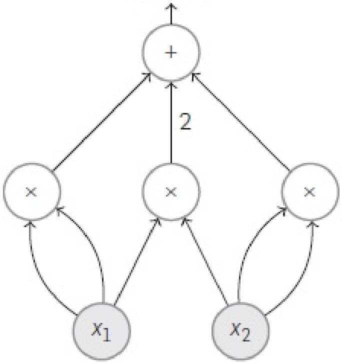
VP is the set of sequences $f_1, f_2, \ldots$ of polynomials such that there is polynomial $p(n)$ so that the arithmetic complexity of f_n is less than $p(n)$.

Arithmetic circuits

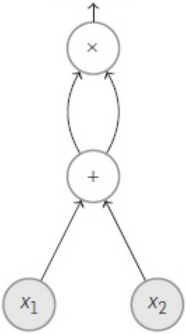
Arithmetic complexity can be visualized using arithmetic circuits

Example: Consider $f(x_1, x_2) = x_1^2 + 2x_1x_2 + x_2^2$.

$x_1^2 + 2x_1x_2 + x_2^2$



$(x_1 + x_2)^2$



Images from Communications of the ACM

A tale of two polynomials: determinants and permanents

Consider a matrix

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{pmatrix}.$$

where each $x_{i,j}$ is a variable. Define the determinant $\det(X)$ as follows:

- $n = 1 : \det \begin{pmatrix} x_{1,1} \end{pmatrix} = x_{1,1}$
- $n = 2 : \det \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix} = x_{1,1}x_{2,2} - x_{1,2}x_{2,1}$
- $n = 3 :$

$$\det \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix} = x_{1,1}x_{2,2}x_{3,3} - x_{1,1}x_{2,3}x_{3,2} + x_{1,2}x_{2,3}x_{3,1} - x_{1,2}x_{2,1}x_{3,3} + x_{1,3}x_{2,1}x_{3,2} - x_{1,3}x_{2,2}x_{3,1}$$

Determinant

As before, let

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{pmatrix}.$$

The determinant is one of the most natural operations in mathematics. Using the group S_n of permutations, we can write

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

The permanent

The permanent $\operatorname{perm}(X)$ is defined just like the determinant $\det(X)$ but without the pesky '-' signs.

Permanent

The permanent

The permanent perm(X) is defined just like the determinant det(X) but without the pesky '-' signs.

Example:

$$\operatorname{perm}\begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix} = x_{1,1}x_{2,2}x_{3,3} + x_{1,1}x_{2,3}x_{3,2} + x_{1,2}x_{2,3}x_{3,1} + x_{1,2}x_{2,1}x_{3,3} + x_{1,3}x_{2,1}x_{3,2} + x_{1,3}x_{2,2}x_{3,1}$$

What's easier to compute: the determinant or permanent?

The determinant is easier!

The determinant has magical properties:

- $\det(XY) = \det(X)\det(Y)$
- In particular, the determinant $\det(X)$ does not change if you perform elementary row and column operations to X.

These facts are not true for the permanent!

Gaussian elimination: After performing row and column operations to X, we may arrange X to have the following form:

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ 0 & x_{2,2} & \cdots & x_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & x_{n,n} \end{pmatrix}.$$

and in this case $\det(X) = x_{1,1}x_{2,2}\cdots x_{n,n}$.

Theorem

The sequence of polynomials $\det_1, \det_2, \det_3, \ldots$ *is in VP.*

Even more is true

Not only is the determinant easy to compute, but any other polynomial can be in fact computed as a determinant.

Example: $x^2 + y^2 = \det \begin{pmatrix} x & -y \\ y & x \end{pmatrix}$

This leads us to our next computation model for polynomials:

Determinantal complexity

The determinantal complexity of a polynomial $f(x_1, \ldots, x_n)$ is the size of the smallest matrix

$$L = \begin{pmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,m} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,m} \\ \vdots & & \ddots & \vdots \\ L_{m,1} & L_{m,2} & \cdots & L_{m,m} \end{pmatrix}$$

where each $L_{i,j}$ is linear in the $x_1, \ldots, x_n$, such that $f = \det L$.

The determinantal complexity of the permanent

Valiant's conjecture

The determinant complexity $\mathrm{dc}(\mathrm{perm}_n)$ of the $n \times n$ permanent grows faster than any polynomial.

In other words, there is no polynomial $p(n)$ such that $\mathrm{dc}(\mathrm{perm}_n) < p(n)$ for all n.

Important Fact

This conjecture is equivalent to showing that $VP \neq VNP$.

Example: Since

$$\mathrm{perm}\begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix} = \det\begin{pmatrix} x_{1,1} & -x_{1,2} \\ x_{2,1} & x_{2,2} \end{pmatrix}$$

the determinantal complexity of the 2×2 permanent is 2.

The best known bounds for $\mathrm{dc}(\mathrm{perm}_n)$

Theorem (Mignon–Ressayre and Grenet)

For $n > 2$,
$$n^2/2 \leq \mathrm{dc}(\mathrm{perm}_n) \leq 2^n - 1$$

For $n = 3$, this implies that $5 \leq \mathrm{dc}(\mathrm{perm}_3) \leq 7$. Grenet's formula gives

$$\mathrm{perm}_3 \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{pmatrix} = \det_7 \begin{pmatrix} 0 & x_{1,1} & x_{2,1} & x_{3,1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & x_{3,3} & x_{2,3} & 0 \\ 0 & 0 & 1 & 0 & 0 & x_{1,3} & x_{3,3} \\ 0 & 0 & 0 & 1 & x_{1,3} & 0 & x_{2,3} \\ x_{2,2} & 0 & 0 & 0 & 1 & x_{1,3} & 0 \\ x_{3,2} & 0 & 0 & 0 & 0 & 1 & 0 \\ x_{1,2} & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Can we do better?

What actually is $\mathrm{dc}(\mathrm{perm}_3)$? Is it 5, 6, or 7?

No, 7 is optimal.

Theorem (Alper–Bogart–Velasco)

The determantal complexity $\mathrm{dc}(\mathrm{perm}_3)$ *of the* 3×3 *permanent is* 7.

Tristram Bogart

Mauricio Velasco