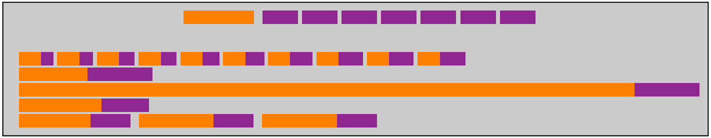
Pattern matching, $X + Y$, and Sparse Multiplication



Daniel S. Roche

Computer Science Department
United States Naval Academy
Annapolis, Maryland, USA

University of Notre Dame
October 8, 2015

This is joint work with Andrew Arnold

Currently at Fields Institute, Toronto

Three Related Problems

1. Polynomial Multiplication
   $(x - xy) \times (x^2y^2 - x^2y + y^2 - y)$
   $\mapsto \quad 2x^3y^2 - x^3y^3 - x^3y + 2xy^2 - xy^3 - xy$

2. String Matching with Wildcards (a.k.a. "don't-cares")
   `.E...T` in `PRESENTATIONS`
   $\mapsto$ `RESENT`, `SENTAT`

3. $X + Y$ a.k.a. Sumset
   $\{1, 5\} \oplus \{4, 6, 8, 10\}$
   $\mapsto \quad \{5, 7, 9, 11, 13, 15\}$

Three Related Problems

1. Polynomial Multiplication
   $(x - xy) \times (x^2y^2 - x^2y + y^2 - y)$
   $\mapsto \quad 2x^3y^2 - x^3y^3 - x^3y + 2xy^2 - xy^3 - xy$
2. String Matching with Wildcards (a.k.a. “don’t-cares”)
   `.E...T` in `PRESENTATIONS`
   $\mapsto$ `RESENT`, `SENTAT`
3. $X + Y$ a.k.a. Sumset
   $\{1, 5\} \oplus \{4, 6, 8, 10\}$
   $\mapsto \quad \{5, 7, 9, 11, 13, 15\}$

Common Feature

The output *can* have quadratic size,
but it’s frequently much smaller.

Our Result

A randomized algorithm for Polynomial Multiplication,
Sumset, and Sparse Wildcard Pattern Matching,
whose running time is nearly linear
in the size of the input and the output.

Scale of improvement

What does it look like to reduce **quadriatic running time** to **randomized nearly-linear running time**?

Analogous example: Sorting

	Insertion Sort $O(n^2)$, deterministic	QuickSort $O(n \log n)$, randomized
75KB	6 seconds	30 milliseconds
1.44MB	40 minutes	0.7 seconds
700MB	19 years?	11 minutes

What is the size of a polynomial?

Polynomials are a basic building block of mathematical and scientific computation.

They can have many variables (n):

$$x_1x_3x_5 + x_1x_2x_3x_4x_9 + x_2x_6x_7x_8x_9 + x_4x_5x_6x_7$$

. . . or large coefficients (C = largest coefficient):

$$34735667x^{12} - 86916241x^{10} - 70003088x^5 + 3786735x^3$$

. . . or very high degree (D = max degree):

$$x^{770352} - 2x^{506115} + 2x^{465975} + 9x^{422527}$$

What is the size of a polynomial?

Polynomials are a basic building block of mathematical and scientific computation.

They can have many variables (n):

$$x_1x_3x_5 + x_1x_2x_3x_4x_9 + x_2x_6x_7x_8x_9 + x_4x_5x_6x_7$$

. . . or large coefficients (C = largest coefficient):

$$34735667x^{12} - 86916241x^{10} - 70003088x^5 + 3786735x^3$$

. . . or very high degree (D = max degree):

$$x^{770352} - 2x^{506115} + 2x^{465975} + 9x^{422527}$$

How do we store these in computer memory?
What are the algorithms to perform basic arithmetic?

Step 0: Reduce to one variable

Given a *multivariate* polynomial in $x_1, x_2, x_3, \ldots$,
find a *univariate* polynomial in z that has all the same information.

Kronecker Substitution

If D is larger than any exponent in the polynomial, replace $f(x_1, x_2, \ldots, x_n)$ with $f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}})$.

The resulting degree is roughly D^n.

Example

$$f(x, y) = x^2y^2 - x^2y + y^2 - y$$

$$f(z, z^4) = z^{10} + z^8 - z^6 - z^4$$

Step 0: Reduce to one variable

Given a *multivariate* polynomial in $x_1, x_2, x_3, \ldots$,
find a *univariate* polynomial in z that has all the same information.

Kronecker Substitution

If D is larger than any exponent in the polynomial, replace $f(x_1, x_2, \ldots, x_n)$ with $f(z, z^D, z^{D^2}, \ldots, z^{D^{n-1}})$.

The resulting degree is roughly D^n.

Randomized Kronecker Substitutions [Arnold & R. 2014]

If T is the number of terms in the polynomial, replace
$f(x_1, x_2, \ldots, x_n)$ with $f(z^{s_1}, z^{s_2}, \ldots, z^{s_n})$,
where each s_i is a random integer less than T.

The resulting degree is roughly DT.
(But you have to repeat this $O(n)$ times.)

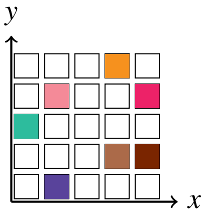
Kronecker Example

Example

$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$

(colored boxes ■ represent coefficients)

Visualization of $f(x, y)$:



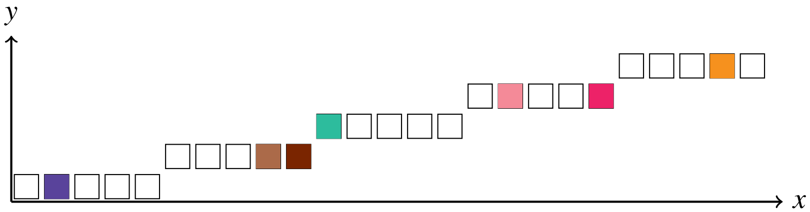
Kronecker Example

Example

$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$

(colored boxes ■ represent coefficients)

Visualization of $f(x, x^D y)$:



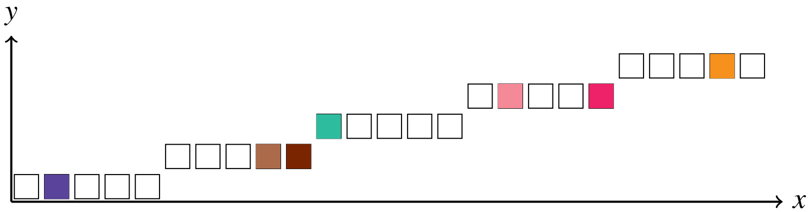
Kronecker Example

Example

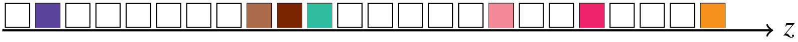
$$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare x y^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$$

(colored boxes ■ represent coefficients)

Visualization of $f(x, x^D y)$**:**



Kronecker substitution: $f(z, z^D)$, degree 23



Randomized Kronecker Example

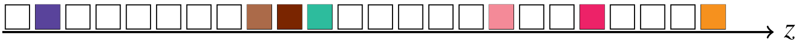
Example

$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$

(colored boxes ■ represent coefficients in R)

Visualization of $f(x, y)$**:**

Kronecker substitution: $f(z, z^D)$, degree 23



Randomized Kronecker Example

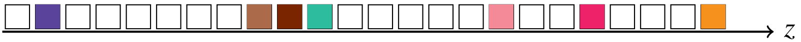
Example

$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$

(colored boxes ■ represent coefficients in R)

Visualization of $f(x^2, y)$**:**

Kronecker substitution: $f(z, z^D)$, degree 23



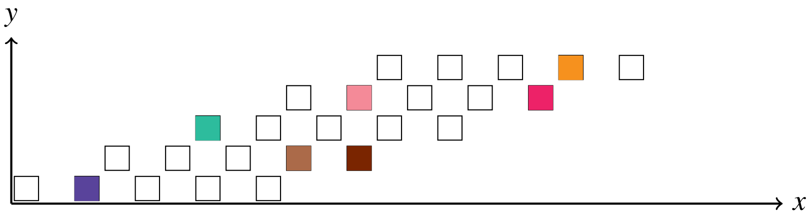
Randomized Kronecker Example

Example

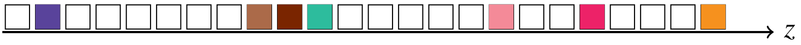
$f(x, y) = \blacksquare x + \blacksquare x^3 y + \blacksquare x^4 y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4 y^3 + \blacksquare x^3 y^4$

(colored boxes ■ represent coefficients in R)

Visualization of $f(x^2, x^3 y)$**:**



Kronecker substitution: $f(z, z^D)$, degree 23



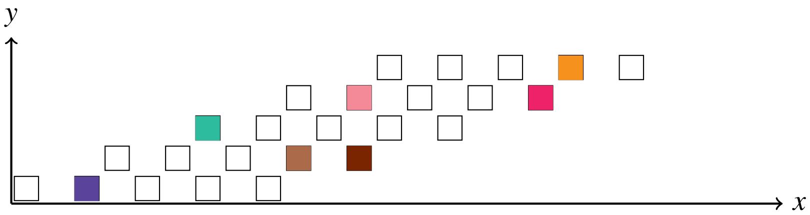
Randomized Kronecker Example

Example

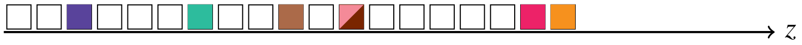
$f(x,y) = \blacksquare x + \blacksquare x^3y + \blacksquare x^4y + \blacksquare y^2 + \blacksquare xy^3 + \blacksquare x^4y^3 + \blacksquare x^3y^4$

(colored boxes ■ represent coefficients in R)

Visualization of $f(x^2, x^3y)$**:**



Randomized Kronecker substitution: $f(z^2, z^3)$, degree 18



Dense Polynomial Representation

A coefficient array indexed by exponent value
is great with just one variable and small degree:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

Dense Polynomial Representation

A coefficient array indexed by exponent value
is great with just one variable and small degree:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

Zero coefficients are stored explicitly — possibly wasteful

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$$\times$$

$$x^4 + 2x^3 + x^2 + 5$$

1	2	1	0	5

$$=$$

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$$\times$$

$$x^4 + 2x^3 + x^2 + 5$$

1	2	1	0	5

$$=$$

$$x^{15}$$

1															

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$$\times$$

$$x^4 + 2x^3 + x^2 + 5$$

1	2	1	0	5

$$=$$

$$x^{15} + 5x^{14}$$

1	5														

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$$\times$$

$$x^4 + 2x^3 + x^2 + 5$$

1	2	1	0	5

$$=$$

$$x^{15} + 7x^{14}$$

1	7														

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^8 + 4x^7 + 7x^6 + x^2 + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$\times$

5	0	1	2	1

$$5 + x^2 + 2x^3 + x^4$$

$$=$$

$$x^{15}$$

1															

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^{8} + 4x^{7} + 7x^{6} + x^{2} + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$\times$

5	0	1	2	1

$$5 + x^{2} + 2x^{3} + x^{4}$$

$$=$$

$$x^{15} + 7x^{14}$$

1	7														

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^{8} + 4x^{7} + 7x^{6} + x^{2} + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$\times$

5	0	1	2	1

$$5 + x^{2} + 2x^{3} + x^{4}$$

$$=$$

$$x^{15} + 7x^{14} + 11x^{13}$$

1	7	11													

Dense multiplication

"School" multiplication algorithm:

$$x^{11} + 5x^{10} + 9x^{8} + 4x^{7} + 7x^{6} + x^{2} + 8$$

1	5	0	9	4	7	0	0	0	1	0	8

$\times$

5	0	1	2	1

$$5 + x^{2} + 2x^{3} + x^{4}$$

$=$

$$x^{15} + 7x^{14} + 11x^{13} + \cdots + 36x^{6}$$

1	7	11	14	27	49	18	52	20	36						

Running time: $O(D_1 D_2)$, quadratic in the degree

Fast Dense Multiplication

This is a powerful tool!

- Karatsuba (1962): $O(n^{1.59})$
- Toom-Cook (1966): $O(n^{1.47})$
- Schönhage-Strassen (1971): $O(n \log n \log \log n)$
- Cantor-Kaltofen (1991): $O(n \log n \log \log n)$
- Fürer (2007): $O(n \log n 2^{O(\log *n)})$
- De, Kurur, Saha, Saptharishi (2008): $O(n \log n 2^{O(\log *n)})$
- Harvey, van der Hoeven, Lecerf (2014): $O(n \log n 8^{\log *n})$

Fast Dense Multiplication

This is a powerful tool!

- Karatsuba (1962): $O(n^{1.59})$
- Toom-Cook (1966): $O(n^{1.47})$

- Schönhage-Strassen (1971): $O(n \log n \log \log n)$
- Cantor-Kaltofen (1991): $O(n \log n \log \log n)$
- Fürer (2007): $O(n \log n 2^{O(\log * n)})$
- De, Kurur, Saha, Saptharishi (2008): $O(n \log n 2^{O(\log * n)})$
- Harvey, van der Hoeven, Lecerf (2014): $O(n \log n 8^{\log * n})$

All results since Schönhage-Strassen use FFTs
and have nearly linear $\tilde{O}(n)$ complexity.

Sparse Polynomials

Frequently, polynomials have many zero coefficients:

$$x^{29} + 9x^{12} + 4x^{11} + 2x^2$$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	4	0	0	0	0	0	0	0	0	2	0	0

Sparse Polynomials

Frequently, polynomials have many zero coefficients:

$$x^{29} + 9x^{12} + 4x^{11} + 2x^2$$

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	4	0	0	0	0	0	0	0	0	2	0	0

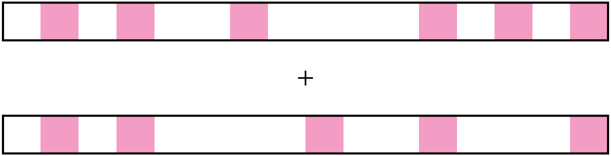
Then the sparse representation, a list of coefficient/exponent pairs, is more compact:

$$x^{29} + 9x^{12} + 4x^{11} + 2x^2$$

29	12	11	2
1	9	4	2

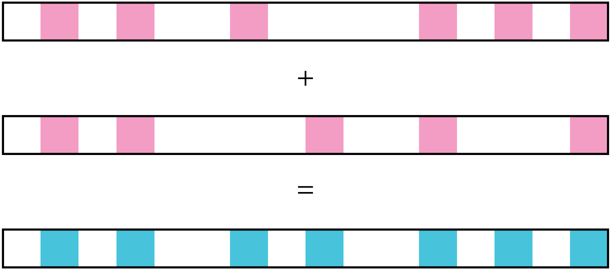
Sparse Polynomial Addition

In arithmetic operations, there are **two kinds of sparsity**:



Sparse Polynomial Addition

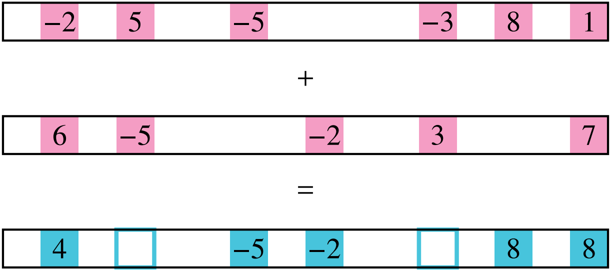
In arithmetic operations, there are **two kinds of sparsity**:



- Structural sparsity is 7.

Sparse Polynomial Addition

In arithmetic operations, there are **two kinds of sparsity**:



- Structural sparsity is 7.
- Arithmetic sparsity is 5.

Sparse Multiplication

"School" multiplication algorithm:

$$x^{10} + x^8 - x^6 - x^4$$

1		1		-1		-1				

$$\times$$

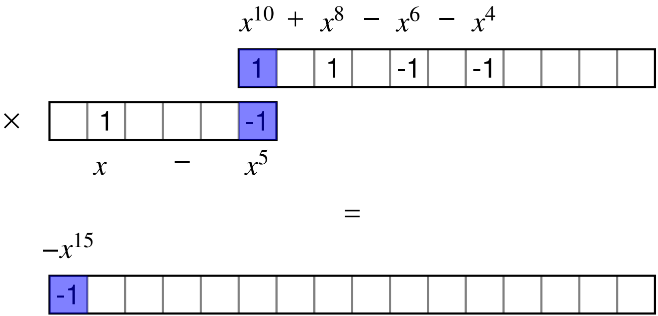
$$-x^5 + x$$

-1				1	

$$=$$

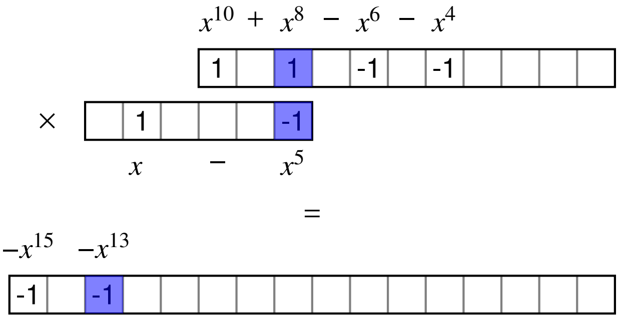
Sparse Multiplication

"School" multiplication algorithm:



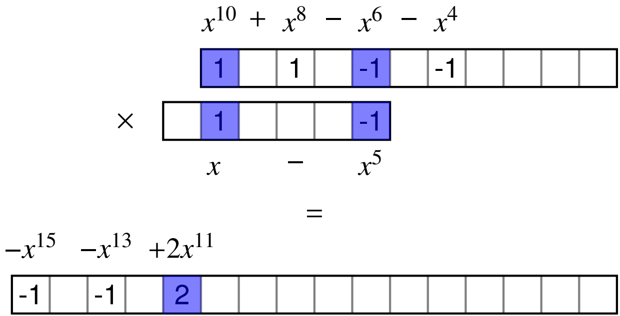
Sparse Multiplication

"School" multiplication algorithm:



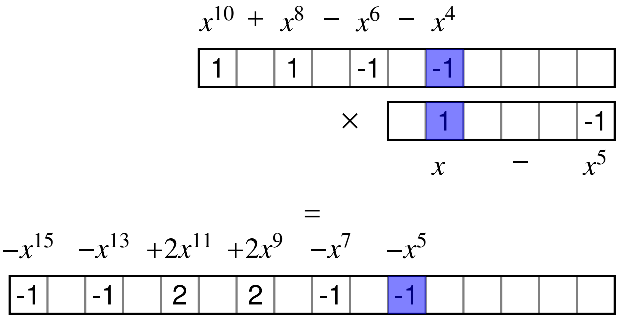
Sparse Multiplication

"School" multiplication algorithm:



Sparse Multiplication

"School" multiplication algorithm:



Running time: $\tilde{O}(T_1 T_2)$, quadratic in the number of terms

Output-Sensitive Sparse Multiplication

Quadratic-time already defeated in many cases:

- Recursive dense
- Chunky, equal spaced (R. '11)
- Blockwise dense (van der Hoeven & Lecerf '12)
- Homogeneous dense (Gastineau & Laskar '13)
- Support on a lattice (van der Hoeven, Lebreton, Schost '13)
- Support is given (van der Hoeven & Lecerf '13)

Sparse Interpolation

Another powerful tool!

Sparse Polynomial Interpolation Problem

Given a way to evaluate $f(\theta)$ at any θ,
plus bounds on degree, sparsity, and height,
determine the coefficients and exponents of f.

Reduces polynomial multiplication to scalar multiplication, because

$$h = f \cdot g \quad \Rightarrow \quad h(\theta) = f(\theta) \cdot g(\theta)$$

Sparse Interpolation Algorithms

“Big prime” algorithms

Computation is performed modulo p, $p \gg \deg(fg)$.

But one evaluation needs $\tilde{O}(T \log \deg(fg))$ ops modulo p;
hence at least $\tilde{O}(T \log^2 \deg(fg))$ bit complexity

- Prony (1795)
- Ben-Or & Tiwari (1988)
- Kaltofen & Lakshman (1989)
- Kaltofen & Lee (2003)
- Cuyt & Lee (2010)

Sparse Interpolation Algorithms

"Small primes" algorithms

Computations performed modulo small primes p.

But all algorithms still need $\tilde{O}(T \log^2 \deg(fg))$ operations.

- Grigoriev & Karpinsky (1987)
- Garg & Schost (2007)
- Giesbrecht & R. (2011)
- Arnold, Giesbrecht & R. (2014)
- Khochtali, R. & Tian (2015)

Sparse Interpolation Algorithms

“Big prime” algorithms

Computation is performed modulo p, $p \gg \deg(fg)$.

But one evaluation needs $\tilde{O}(T \log \deg(fg))$ ops modulo p;
hence at least $\tilde{O}(T \log^2 \deg(fg))$ bit complexity

“Small primes” algorithms

Computations performed modulo small primes p.

But all algorithms still need $\tilde{O}(T \log^2 \deg(fg))$ operations.

Observe: The trouble is in the degree!

String Matching

Problem Definition

Given a text t and a pattern p,
find all occurrences of p in t.

- Big, “classical” problem in computer science
- Applications to bioinformatics, information retrieval, databases,. . .
- Live demo?

Matching Example

"School" string matching algorithm:

Text

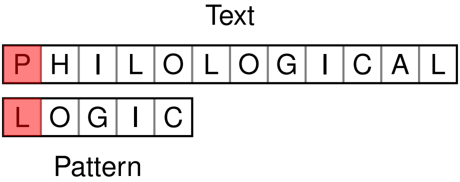
P	H	I	L	O	L	O	G	I	C	A	L

Pattern

L	O	G	I	C

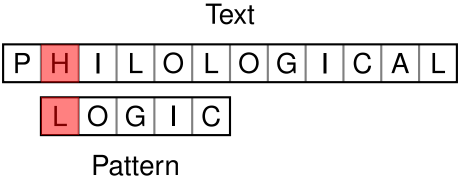
Matching Example

"School" string matching algorithm:



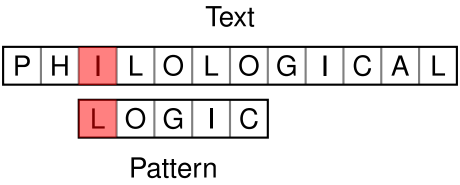
Matching Example

"School" string matching algorithm:



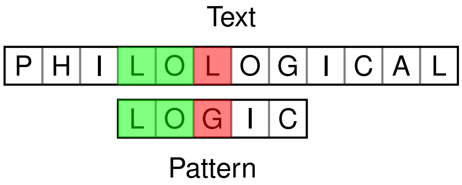
Matching Example

"School" string matching algorithm:



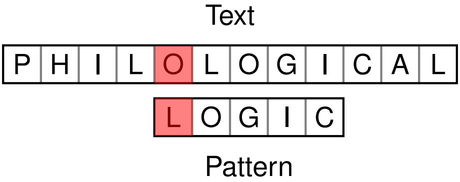
Matching Example

"School" string matching algorithm:



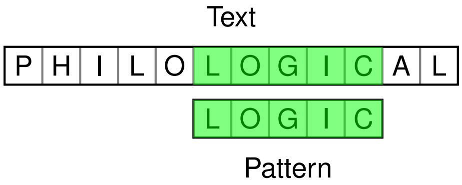
Matching Example

"School" string matching algorithm:



Matching Example

"School" string matching algorithm:



Running time: $O(nm)$, quadratic in the sizes

String Matching Algorithms

Several great solutions are available:

- Use a DFA (problem: slow to create)
- Use a suffix tree (problem: uses $O(n)$ space)

- Knuth-Morris-Pratt — $O(m + n)$ worst case
- Boyer-Moore — $O(n + m + |\Sigma|)$ and practical

It looks like there's nothing left here to do!

Pattern Matching with Wildcards

What if the pattern has don't-care characters?

And what if the pattern and text are multi-dimensional?

Pattern Matching with Wildcards

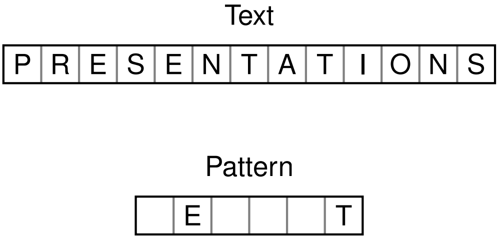
What if the pattern has don't-care characters?

And what if the pattern and text are multi-dimensional?

Applications

- Object recognition (computer vision)
- Computational biology (drug design)
- Structured text search
- Music retrieval

Wildcard Pattern Matching



Wildcard Pattern Matching

Preprocessing: Clear extraneous characters from text

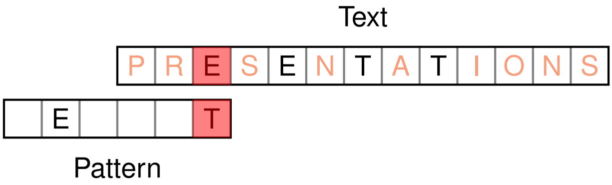
Text

P	R	E	S	E	N	T	A	T	I	O	N	S

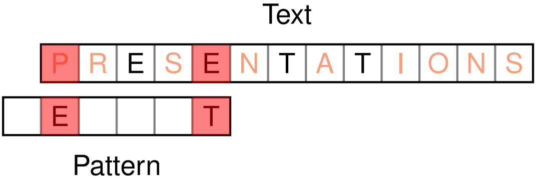
Pattern

	E				T

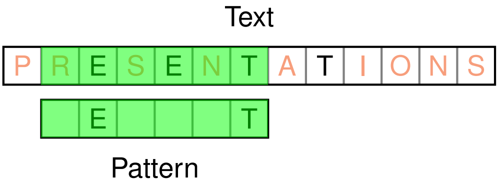
Wildcard Pattern Matching



Wildcard Pattern Matching



Wildcard Pattern Matching



Running time: $\tilde{O}(T_1 T_2)$, quadratic in the number of non-wildcards

Sparse Wildcard Pattern Matching Algorithms

- Fischer & Paterson (1974)
- Cole & Hariharan (2002)
- Clifford & Clifford (2007)
- Amir, Kapah & Porat (2007)

The fastest algorithms use (sort of)
randomized Kronecker substitution
and dense multiplication
to get $\tilde{O}(T \log^2 D)$ complexity

Sumset

Problem Statement

Given two sets X, Y, the **sumset** $X \oplus Y$ equals $\{x + y \mid x \in X \text{ and } y \in Y\}$

Related problems:

- 3SUM (Given a set, do three numbers sum to 0?)
- $X + Y$ Sorting (Given two sets, sort their sumset)

What is the connection to polynomial multiplication and string matching?

Problem Connections

- Consider multiplying $(-x^5 + x) \cdot (x^{10} + x^8 - x^6 - x^4)$.

  The sumset $\{1, 5\} \oplus \{4, 6, 8, 10\}$ encodes the exponents of the sparse product.

Problem Connections

- Consider multiplying $(-x^5 + x) \cdot (x^{10} + x^8 - x^6 - x^4)$.

  The sumset $\{1, 5\} \oplus \{4, 6, 8, 10\}$ encodes the exponents of the sparse product.

- Consider searching for `.E...T` in `PRESENTATIONS`.

  The sumset $\{1, 5\} \oplus \{4, 6, 8, 10\}$ encodes the positions that need to be checked for potential matches.

  Also note, we can encode each character as a number so the product of matching encodings equals 1.

A fast sumset algorithm is critical to both applications!

Sumset Algorithm Overview

The randomized sumset computation works in two phases.

Phase 1: Estimate the size

Reduce the entries modulo
random small primes, increasing in size,
and use dense multiplication
until the result becomes sparse.

Phase 2: Get the sumset

Construct a sparse polynomial whose
coefficients encode sumset inputs,
then use sparse interpolation
to compute the product.

Running Example

The Problem

$X = \{1238, 2520, 3631, 4913\}$

$Y = \{641, 1923, 4316\}$

We want to find $X \oplus Y$.

Running Example

The Problem

$X = \{1238, 2520, 3631, 4913\}$

$Y = \{641, 1923, 4316\}$

We want to find $X \oplus Y$.

Step 0: Form sparse polynomials from the exponent sets.

- $f = z^{4913} + z^{3630} + z^{2520} + z^{1238}$
- $g = z^{4316} + z^{1923} + z^{641}$

The exponents in the product fg form the sumset.

Step 1: Estimate structural sparsity

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$

$g = z^{4316} + z^{1923} + z^{641}$

How sparse is the product $h = f \cdot g$?

1. Choose primes $p = 211, \quad p' = 5$
2. Compute $\left((f \cdot g) \bmod p\right) \bmod p'$
   $= 2z^4 + 3z^3 + 3z^2 + 2z + 2$
3. Less than half-dense? No

Step 1: Estimate structural sparsity

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$

$g = z^{4316} + z^{1923} + z^{641}$

How sparse is the product $h = f \cdot g$?

1. Choose primes $p = 211, \quad p' = 11$
2. Compute $\left((f \cdot g) \bmod p\right) \bmod p'$
   $= 3z^9 + 2z^8 + z^7 + 2z^4 + z^3 + 3z^2$
3. Less than half-dense? No

Step 1: Estimate structural sparsity

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$
$g = z^{4316} + z^{1923} + z^{641}$

How sparse is the product $h = f \cdot g$?

1. Choose primes $p = 211, \quad p' = 17$
2. Compute $\left((f \cdot g) \bmod p\right)^{\bmod p'}$
   $= z^{16} + z^{7} + z^{6} + 2z^{4} + 3z^{3} + z^{2} + z + 2$
3. Less than half-dense? Yes
   Means structural sparsity is close to 8.

First technique: Multiple Reduction and Relaxation

$$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$$

$$f^{\bmod 211} = z^{199} + z^{183} + z^{60} + z^{44}$$

$$\left(f^{\bmod 211}\right)^{\bmod 17} = z^{13} + z^{12} + z^{10} + z^{9}$$

What's going on?

- First reduce exponents modulo p
- Now treat that as an ordinary polynomial
- Then reduce further!
- Each reduction introduces a factor-2 in the error estimation.

First Tool

How to compute $\left((f \cdot g) \bmod p\right) \bmod p'$?

- This polynomial never gets very sparse
- Its degree is *linear* in the actual structural sparsity
- So we can use dense polynomial arithmetic!

Step 2: Compute structural support

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$
$g = z^{4316} + z^{1923} + z^{641}$

$\#(f \cdot g) \approx 8$

What are the exponents of $h = f \cdot g$?

- Use the same prime $p = 211$ as before.
- Compute $h_1 = \left(f^{\bmod p} \cdot g^{\bmod p}\right)^{\bmod p}$
  $= 2z^{207} + z^{191} + z^{156} + z^{140} + 2z^{84} + 3z^{68} + z^{52} + z^{12}$

Step 2: Compute structural support

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$
$g = z^{4316} + z^{1923} + z^{641}$

$\#(f \cdot g) \approx 8$

What are the exponents of $h = f \cdot g$?

- Use the same prime $p = 211$ as before.
- Set $\ell \gg \deg(h) = 16000$
- Compute $f_2 = \sum (e\ell + 1) z^{e \bmod p}$
  $= (4913 \cdot 16000 + 1) z^{4913 \bmod 211} + (3631 \cdot 16000 + 1) z^{3631 \bmod 211} + \cdots$
  $= 40320001 z^{199} + 19808001 z^{183} + 78608001 z^{60} + 58096001 z^{44}$
- Compute g_2 similarly.
- Compute $h_2 = (f_2 \cdot g_2)^{\bmod p} \bmod \ell^2$

$= 101152002 z^{207} + 30064001 z^{191} + 147664001 z^{156} + 127152001 z^{140} + 218752002 z^{84} + 266592003 z^{68} + 68352001 z^{52} + 71088001 z^{12}$

Step 2: Compute structural support

Given

$f = z^{4913} + z^{3631} + z^{2520} + z^{1238}$
$g = z^{4316} + z^{1923} + z^{641}$

$\#(f \cdot g) \approx 8$

What are the exponents of $h = f \cdot g$?

- $p = 211, \quad , \ell = 16000$
- $h1 = 2z^{207} + z^{191} + z^{156} + z^{140} + 2z^{84} + 3z^{68} + z^{52} + z^{12}$
- $h2 = 101152002z^{207} + \cdots + 68352001z^{52} + \cdots$
- Take coefficient ratios: $\dfrac{\frac{c_2}{c_1} - 1}{\ell}$
- **And the sumset is**:
  $1879, \ 3161, \ 4272, \ 4443, \ 5554, \ 6836, \ 7947, \ 9229$

Second technique: Coefficient ratios

The polynomials f_2, g_2, h_2 have their exponents *encoded in the coefficients*.

The encoding is *additive* modulo ℓ^2:
$(a\ell + 1)(b\ell + 1) \bmod \ell^2 = (a + b)\ell + 1$

Allows recovering the *actual exponents* from the coefficients of the degree-reduced product.

Big idea: turning scalar multiplication into addition

Second Tool

How to compute $h_2 = f_2 \cdot g_2$?

- This polynomial is *kind of* sparse.
- It has huge coefficients!
- We can use sparse polynomial interpolation!
- **Requirement**: Linear-time in the sparsity bound, poly-logarithmic in the degree.

What just happened?

We have a randomized algorithm to compute sumset in nearly linear time, using the tools of dense multiplication and sparse interpolation.

Completely glossed over:

- How big do those primes really need to be?
- What is the failure probability?
- Which version of sparse interpolation can be used?

Now let's apply this to sparse polynomial multiplication.

Running Example

The Problem

$$f = 65x^{31}y^{36} + 20x^{13}y^{49} + 26x^{38}y^{12} + 16x^{20}y^{25}$$
$$g = 60x^{16}y^{43} + 78x^{41}y^{6} - 48x^{23}y^{19}$$

What is the product $h = fg$?

Running Example

The Problem

$$f = 65x^{31}y^{36} + 20x^{13}y^{49} + 26x^{38}y^{12} + 16x^{20}y^{25}$$
$$g = 60x^{16}y^{43} + 78x^{41}y^{6} - 48x^{23}y^{19}$$

What is the product $h = fg$?

Overview of approach

0. Reduce to univariate
1. Compute the structural support
2. Compute arithmetic support (i.e., the actual exponents)
3. Compute the coefficients

Step 0: Substitutions

Given

$f = 65x^{31}y^{36} + 20x^{13}y^{49} + 26x^{38}y^{12} + 16x^{20}y^{25}$
$g = 60x^{16}y^{43} + 78x^{41}y^{6} - 48x^{23}y^{19}$

Kronecker Substitution

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$

Note: h completely determined from $f_K g_K$.

Step 1: Compute structural support

Given

$f_S = z^{4913} + z^{3631} + z^{2520} + z^{1238}$
$g_S = z^{4316} + z^{1923} + z^{641}$

$\#(f_S \cdot g_S) \approx 8$

What are the exponents of $h_S = f_S \cdot g_S$?

- Just compute the sumset $\{1238, 2520, 3631, 4913\} \oplus \{641, 1923, 4316\}$
- (We already did it!)
- $= \{1879,\ 3161,\ 4272,\ 4443,\ 5554,\ 6836,\ 7947,\ 9229\}$

Step 2: Trim down to the arithmetic support

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$
$\mathrm{supp}(f_K \cdot g_K) \subseteq S =$
$\{1879,\ 3161,\ 4272,\ 4443,\ 5554,\ 6836,\ 7947,\ 9229\}$

What are the *actual* exponents of $f_K \cdot g_K$?

1. Choose $p = 23, \quad q = 47$ (note $p|(q-1)$)
2. Compute $S \bmod p = \{16, 10, 17, 4, 11, 5, 12, 6\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$
   $= 41z^{17} + 7z^{16} + 46z^{12} + 25z^{6} + 31z^{4}$

Step 2: Trim down to the arithmetic support

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$

$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$

$\mathrm{supp}(f_K \cdot g_K) \subseteq S =$

$\{\mathbf{1879},\ 3161,\ \mathbf{4272},\ \mathbf{4443},\ 5554,\ 6836,\ \mathbf{7947},\ \mathbf{9229}\}$

What are the *actual* exponents of $f_K \cdot g_K$?

1. Choose $p = 23, \quad q = 47$ (note $p | (q-1)$)
2. Compute $S \bmod p = \{\mathbf{16}, 10, \mathbf{17}, \mathbf{4}, 11, 5, \mathbf{12}, \mathbf{6}\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$ $= 41z^{17} + 7z^{16} + 46z^{12} + 25z^{6} + 31z^{4}$
4. Identify support from nonzero terms

Twist on second tool

How to compute $(f_K \cdot g_K)^{\bmod p} \bmod q$?

- This polynomial is *kind of* sparse.
- An advantage: this time we know the support!
- Use the coefficient-finding step of sparse interpolation!
- Because $p|(q-1)$, we can evaluate at pth roots of unity and solve a transposed Vandermonde system.

Step 3: Compute the coefficients

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$
$\mathrm{supp}(f_K \cdot g_K) = S' = \{1879,\ 4272,\ 4443,\ 7947,\ 9229\}$

What are the coefficients of $f_K \cdot g_K$?

1. Choose $p = 11, \quad q = 23$ (note $p|(q-1)$)
2. Compute $S' \bmod p = \{9, 4, 10, 5, 0\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$
   $= 14z^{10} + 4z^9 + 13z^5 + 10z^4 + 4$
4. Group like terms for Chinese Remaindering

Step 3: Compute the coefficients

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$
$\mathrm{supp}(f_K \cdot g_K) = S' = \{1879, 4272, 4443, 7947, 9229\}$

What are the coefficients of $f_K \cdot g_K$?

1. Choose $p = 11, \quad q = 67$ (note $p | (q-1)$)
2. Compute $S' \bmod p = \{9, 4, 10, 5, 0\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$
   $= 36z^{10} + 18z^{9} + 14z^{5} + 45z^{4} + 61$
4. Group like terms for Chinese Remaindering

Step 3: Compute the coefficients

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$
$\text{supp}(f_K \cdot g_K) = S' = \{1879,\ 4272,\ 4443,\ 7947,\ 9229\}$

What are the coefficients of $f_K \cdot g_K$?

1. Choose $p = 11, \quad q = 89$ (note $p|(q-1)$)
2. Compute $S' \bmod p = \{9, 4, 10, 5, 0\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$
   $= 33z^{10} + 70z^9 + 73z^5 + 86z^4 + 43$
4. Group like terms for Chinese Remaindering

Step 3: Compute the coefficients

Given

$f_K = f(z, z^{100}) = 20z^{4913} + 65z^{3631} + 16z^{2520} + 26z^{1238}$
$g_K = g(z, z^{100}) = 60z^{4316} - 48z^{1923} + 78z^{641}$
$\mathrm{supp}(f_K \cdot g_K) = S' = \{1879,\ 4272,\ 4443,\ 7947,\ 9229\}$

What are the coefficients of $f_K \cdot g_K$?

1. Choose $p = 11$, $q =$ 23, 67, 89
2. Compute $S' \bmod p = \{9, 4, 10, 5, 0\}$
3. Compute $h_{p,q} = (f_K \cdot g_K)^{\bmod p} \bmod q$
5. Apply CRT and undo the Kronecker map:
$h = 3900x^{47}y^{79} + 1200x^{29}y^{92} + 5070x^{72}y^{42} + 2028x^{79}y^{18} - 768x^{43}y^{44}$

Complexity Overview

Non-toy example

1000 terms, 8 variables, 64-bit coefficients, 32-bit exponents

Structural sparsity 10000, arithmetic sparsity 1000

Complexity Overview

Non-toy example

1000 terms, 8 variables, 64-bit coefficients, 32-bit exponents

Structural sparsity 10000, arithmetic sparsity 1000

Steps of the algorithm

1. Estimate structural sparsity (Sumset part 1)

Complexity Overview

Non-toy example

1000 terms, 8 variables, 64-bit coefficients, 32-bit exponents

Structural sparsity 10000, arithmetic sparsity 1000

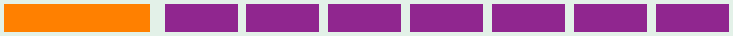
Steps of the algorithm

1. Estimate structural sparsity (Sumset part 1)
2. Compute structural support (Sumset part 2)

Complexity Overview

Non-toy example

1000 terms, 8 variables, 64-bit coefficients, 32-bit exponents



Structural sparsity 10000, arithmetic sparsity 1000

Steps of the algorithm

1. Estimate structural sparsity (Sumset part 1)

2. Compute structural support (Sumset part 2)

3. Trim to arithmetic support

Complexity Overview

Non-toy example

1000 terms, 8 variables, 64-bit coefficients, 32-bit exponents

Structural sparsity 10000, arithmetic sparsity 1000

Steps of the algorithm

1. Estimate structural sparsity (Sumset part 1)
2. Compute structural support (Sumset part 2)
3. Trim to arithmetic support
4. Compute coefficients

Multiplication Algorithm Complexity

C = |largest coefficient|
D = max degree
n = # of variables

S = structural sparsity
T = arithmetic sparsity

Theorem

Given $f, g \in \mathbb{Z}[x]$, our Monte Carlo algorithm computes $h = fg$ with $\tilde{O}(nS \log C + nT \log D)$ bit complexity.

Extends to softly-linear time algorithms for

- Multivariate polynomials
- Laurent polynomials
- Modular rings, finite fields, exact rationals

What about pattern matching?

Sparse Wildcard Pattern Matching can be solved with fast sparse polynomial multiplication.

Open research questions:

- Can we solve any *practical* matching problems faster?
- Can the approach be made sensitive to the *actual* number of matches?
- Can we work directly on some application such as music identification?

Summary

Three Problems

1. $(x - xy) \times (x^2y^2 - x^2y + y^2 - y)$
2. `.E...T` in `PRESENTATIONS`
3. $\{1, 5\} \oplus \{4, 6, 8, 10\}$

Two Tools

Dense multiplication

Sparse interpolation

And one algorithm to do it all!