The Sparsity Challenges

James Davenport & Jacques Carette

University of Bath, McMaster University
(both visiting Waterloo)

26 September 2009

Notation

For a polynomial f:

d_f is the degree of f

t_f is the number of non-zero terms in f

$|f|$ is the largest absolute value of a coefficient

Notation

For a polynomial f:

d_f is the degree of f

t_f is the number of non-zero terms in f

$|f|$ is the largest absolute value of a coefficient

$t_f/(d_f + 1)$ is a measure of the sparsity of a polynomial

Setting: Typical Textbooks

Setting: Typical Textbooks

- Introduce sparse polynomial representations

Setting: Typical Textbooks

- Introduce sparse polynomial representations

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;
- Carefully explain good algorithms for adding and multiplying sparse polynomials;

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;
- Carefully explain good algorithms for adding and multiplying sparse polynomials;
- Go on to discuss division, gcd, factorization etc.,

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;
- Carefully explain good algorithms for adding and multiplying sparse polynomials;
- Go on to discuss division, gcd, factorization etc.,

!! while silently switching to dense thinking.

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;
- Carefully explain good algorithms for adding and multiplying sparse polynomials;
- Go on to discuss division, gcd, factorization etc.,

!! while silently switching to dense thinking.

Setting: Typical Textbooks

- Introduce sparse polynomial representations, and explain how every realistic representation has to be sparse;
- Carefully explain good algorithms for adding and multiplying sparse polynomials;
- Go on to discuss division, gcd, factorization etc.,

!! while silently switching to dense thinking.

This is the sparsity challenge!

More precisely: the sparsity challenges

Each operation presents its own peculiarities

More precisely: the sparsity challenges

Each operation presents its own peculiarities

More precisely: the sparsity challenges

Each operation presents its own peculiarities

division with/without remainder,

More precisely: the sparsity challenges

Each operation presents its own peculiarities

division with/without remainder, or a divisibility test only.

More precisely: the sparsity challenges

Each operation presents its own peculiarities

division with/without remainder, or a divisibility test only.

More precisely: the sparsity challenges

Each operation presents its own peculiarities

- **division** with/without remainder, or a divisibility test only.
- **gcd** or a test for relative primality

More precisely: the sparsity challenges

Each operation presents its own peculiarities

- **division** with/without remainder, or a divisibility test only.
- **gcd** or a test for relative primality
- **square-free decomposition** or a test for square-freeness

More precisely: the sparsity challenges

Each operation presents its own peculiarities

- division: with/without remainder, or a divisibility test only.
- gcd: or a test for relative primality
- square-free decomposition: or a test for square-freeness
- factorization: or a test for irreducibility

More precisely: the sparsity challenges

Each operation presents its own peculiarities

- **division** with/without remainder, or a divisibility test only.
- **gcd** or a test for relative primality
- **square-free decomposition** or a test for square-freeness
- **factorization** or a test for irreducibility
- **others** e.g. polynomial decomposition — does $f(x) = g(h(x))$?

More precisely: the sparsity challenges

Each operation presents its own peculiarities

division: with/without remainder, or a divisibility test only.

gcd: or a test for relative primality

square-free decomposition: or a test for square-freeness

factorization: or a test for irreducibility

others: e.g. polynomial decomposition — does $f(x) = g(h(x))$?

but there are some common difficulties

The cyclotomic polynomials

The cyclotomic polynomials

$C_n = x^n - 1$

The cyclotomic polynomials

$$C_n = x^n - 1$$

$$\Phi_n = \prod_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} \left(x - e^{2\pi i k/n}\right)$$

The cyclotomic polynomials

$$C_n = x^n - 1$$

$$\Phi_n = \prod_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} \left(x - e^{2\pi ik/n}\right)$$

The cyclotomic polynomials

$$C_n = x^n - 1$$

$$\Phi_n = \prod_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} \left(x - e^{2\pi i k/n}\right)$$

$$C_n(x) = \prod_{d|n} \Phi_d(x)$$

The cyclotomic polynomials

$$C_n = x^n - 1$$

$$\Phi_n = \prod_{\substack{k=1 \\ \gcd(k,n)=1}}^{n} \left(x - e^{2\pi i k/n}\right)$$

$$C_n(x) = \prod_{d|n} \Phi_d(x)$$

$$\Phi_n(x) = \prod_{d|n} C_d(x)^{\mu(n/d)}$$

where μ is the Möbius function.

Φ_k is surprising

Φ_k is surprising

$$\Phi_p(x) = x^{p-1} + \cdots + x + 1$$

Φ_k is surprising

$$\Phi_p(x) = x^{p-1} + \cdots + x + 1$$

$$\Phi_6(x) = x^2 - x + 1; \qquad \Phi_{10}(x) = x^4 - x^3 + x^2 - x + 1$$

Φ_k is surprising

$$\Phi_p(x) = x^{p-1} + \cdots + x + 1$$

$$\Phi_6(x) = x^2 - x + 1; \qquad \Phi_{10}(x) = x^4 - x^3 + x^2 - x + 1$$

But $\Phi_{105}(x) = x^{48} \pm \cdots - 2x^{41} \cdots 2x^7 \cdots 1$

Φ_k is surprising

$$\Phi_p(x) = x^{p-1} + \cdots + x + 1$$

$$\Phi_6(x) = x^2 - x + 1; \qquad \Phi_{10}(x) = x^4 - x^3 + x^2 - x + 1$$

But $\Phi_{105}(x) = x^{48} \pm \cdots - 2x^{41} \cdots 2x^7 \cdots 1$

Table: Large coefficients in Φ_k

$\|a_i\|$	2	3	4	5	6	7	8=9
first Φ_k	105	385	1365	1785	2805	3135	6545
$\phi(k)$	48	240	576	768	1280	1440	3840

Φ_k is surprising

$$\Phi_p(x) = x^{p-1} + \cdots + x + 1$$

$$\Phi_6(x) = x^2 - x + 1; \qquad \Phi_{10}(x) = x^4 - x^3 + x^2 - x + 1$$

But $\Phi_{105}(x) = x^{48} \pm \cdots - 2x^{41} \cdots 2x^7 \cdots 1$

Table: Large coefficients in Φ_k

$\|a_i\|$	2	3	4	5	6	7	8=9
first Φ_k	105	385	1365	1785	2805	3135	6545
$\phi(k)$	48	240	576	768	1280	1440	3840
$\|a_i\|$	14	23	25	27	59	359	
first Φ_k	10465	11305	17225	20615	26565	40755	
$\phi(k)$	6336	6912	10752	12960	10560	17280	

Challenge 1

Find useful bounds on the number of terms in *non-cyclotomic* factors of sparse polynomials.

Challenge 1

Find useful bounds on the number of terms in *non-cyclotomic* factors of sparse polynomials.
Note that Bremner has a trinomial which factors as two dense degree 7 polynomials.

Challenge 1

Find useful bounds on the number of terms in *non-cyclotomic* factors of sparse polynomials.
Note that Bremner has a trinomial which factors as two dense degree 7 polynomials.
Is this as bad as it gets?

C_n/Φ_k is difficult

C_n/Φ_k is difficult

- Factoring C_n requires factoring n,

C_n/Φ_k is difficult

- Factoring C_n requires factoring n,

C_n/Φ_k is difficult

- Factoring C_n requires factoring n, but the output will be lengthy

C_n/Φ_k is difficult

- Factoring C_n requires factoring n, but the output will be lengthy
- Writing down just the degrees of the factors of C_n still requires factoring n

C_n/Φ_k is difficult

- Factoring C_n requires factoring n, but the output will be lengthy
- Writing down just the degrees of the factors of C_n still requires factoring n
- Various results of Plaisted

C_n/Φ_k is difficult

- Factoring C_n requires factoring n, but the output will be lengthy
- Writing down just the degrees of the factors of C_n still requires factoring n
- Various results of Plaisted

Also x^n Asking for *all* decomposition of x^n means writing down *all* factors of n

C_n/Φ_k is difficult: Plaisted

C_n/Φ_k is difficult: Plaisted

Theorem (Plaisted)

It is NP-hard to determine whether two sparse polynomials (in the standard encoding) have a non-trivial common divisor.

C_n/Φ_k is difficult: Plaisted

Theorem (Plaisted)

It is NP-hard to determine whether two sparse polynomials (in the standard encoding) have a non-trivial common divisor.

> The basic device of the proofs is to encode the NP-complete problem of 3-satisfiability so that a formula W in n Boolean variables goes to a sparse polynomial $p_M(W)$ which vanishes exactly at certain Mth roots of unity corresponding to the satisfiable assignments to the formula W, where M is the product of the first n primes. [MR 85j:68043]

Challenge 2

Either

Challenge 2

Either

- find a class of problems for which the gcd problem is still NP-complete even when cyclotomic factors are encoded as C_n (or Φ_k); or

Challenge 2

Either

- find a class of problems for which the gcd problem is still NP-complete even when cyclotomic factors are encoded as C_n (or Φ_k); or
- find an algorithm for the gcd of polynomials with *no* cyclotomic factors, which is polynomial-time in the standard encoding.

C_n/Φ_k can be disguised

C_n/Φ_k can be disguised

There are "scaled cyclotomics" such as

$$x^{105} - 2^{105} = 2^{105} C_{105}(x/2)$$

C_n/Φ_k can be disguised

There are "scaled cyclotomics" such as

$$x^{105} - 2^{105} = 2^{105} C_{105}(x/2)$$

A partial answer to the cyclotomics problem is to admit C_n (or Φ_k) as elements in our *output* vocabulary.

Types of challenges

Types of challenges

- The output may not be sparse

Types of challenges

- The output may not be sparse
  - 'dumb', e.g. quotient with remainder

Types of challenges

- The output may not be sparse
  - 'dumb', e.g. quotient with remainder
  - 'degenerate', where we have encoded a different problem

Types of challenges

- The output may not be sparse
  - 'dumb', e.g. quotient with remainder
  - 'degenerate', where we have encoded a different problem
  - 'unknown', where we expect sparsity *most of the time*

Types of challenges

- The output may not be sparse
  - 'dumb', e.g. quotient with remainder
  - 'degenerate', where we have encoded a different problem
  - 'unknown', where we expect sparsity *most of the time*
- The problem may be intrinsically hard — e.g. Plaisted

Types of challenges

- The output may not be sparse
  - ‘dumb’, e.g. quotient with remainder
  - ‘degenerate’, where we have encoded a different problem
  - ‘unknown’, where we expect sparsity *most of the time*
- The problem may be intrinsically hard — e.g. Plaisted
- We may just not know a good algorithm

Types of challenges

- The output may not be sparse
  - ‘dumb’, e.g. quotient with remainder
  - ‘degenerate’, where we have encoded a different problem
  - ‘unknown’, where we expect sparsity *most of the time*
- The problem may be intrinsically hard — e.g. Plaisted
- We may just not know a good algorithm

Types of challenges

- The output may not be sparse
  - ‘dumb’, e.g. quotient with remainder
  - ‘degenerate’, where we have encoded a different problem
  - ‘unknown’, where we expect sparsity *most of the time*
- The problem may be intrinsically hard — e.g. Plaisted
- We may just not know a good algorithm as in the case of gcd of polynomials with no cyclotomic factors

Division: f/g

With remainder:

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth!

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth!

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth! Consider $x^{1000}/(x-10)$

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth! Consider $x^{1000}/(x-10)$

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth! Consider $x^{1000}/(x-10)$

Exact use "early abort": solves coefficient growth

Division: f/g

With remainder: very bad

- Naïvely $O(d_f^2 t_g)$ exponent comparisons
- Better $O(d_f t_g \log d_f)$ exponent comparisons
- Coefficient growth! Consider $x^{1000}/(x-10)$

Exact use "early abort": solves coefficient growth and in practice is very effective

- In the standard model, dependence on d_f is inevitable: $(x^n - 1)/(x-1)$.

Challenge 3

Find an algorithm for exact division of f by g which is polynomial-time in t_f, t_g *and* $t_{f/g}$.

Challenge 3

Find an algorithm for exact division of f by g which is polynomial-time in t_f, t_g *and* $t_{f/g}$.
This plus challenge 1 (bounds on term count) would be a real breakthrough

Exact Divisibility

Theorem (Plaisted)

The following problem is NP-hard: given an integer N and a set $\{p_1(x), \ldots, p_k(x)\}$ of sparse polynomials with integer coefficients, to determine whether $x^N - 1$ divides $\prod_{j=1}^{k} p_j(x)$.

Exact Divisibility

Theorem (Plaisted)

The following problem is NP-hard: given an integer N and a set $\{p_1(x), \ldots, p_k(x)\}$ of sparse polynomials with integer coefficients, to determine whether $x^N - 1$ divides $\prod_{j=1}^{k} p_j(x)$.

Again, the proof is based on 3-SAT.

Exact Divisibility

Theorem (Plaisted)

The following problem is NP-hard: given an integer N and a set $\{p_1(x), \ldots, p_k(x)\}$ of sparse polynomials with integer coefficients, to determine whether $x^N - 1$ divides $\prod_{j=1}^{k} p_j(x)$.

Again, the proof is based on 3-SAT. Note, however, that the product may be dense, so we shouldn't quite give up hope here.

Challenge 4

Either

Challenge 4

Either

- find a class of problems for which the simple problem "does g divide f?" is still NP-complete; or

Challenge 4

Either

- find a class of problems for which the simple problem "does g divide f?" is still NP-complete; or
- find an algorithm for the divisibility of polynomials which is polynomial-time.

Challenge 4

Either

- find a class of problems for which the simple problem "does g divide f?" is still NP-complete; or
- find an algorithm for the divisibility of polynomials which is polynomial-time.

Failing this

Challenge 4

Either

- find a class of problems for which the simple problem "does g divide f?" is still NP-complete; or
- find an algorithm for the divisibility of polynomials which is polynomial-time.

Failing this

- find an algorithm for the divisibility of cyclotomic-free polynomials which is polynomial-time.

Challenge 4

Either

- find a class of problems for which the simple problem "does g divide f?" is still NP-complete; or
- find an algorithm for the divisibility of polynomials which is polynomial-time.

Failing this

- find an algorithm for the divisibility of cyclotomic-free polynomials which is polynomial-time.

Again, there is scope for a major breakthrough here.

Greatest Common Divisor

Greatest Common Divisor

Plaisted's theorem shows that there are hard cases here.

Challenge 5

As a special case of Challenge 1 we can ask the following.

Challenge 5

As a special case of Challenge 1 we can ask the following. Find useful bounds on the number of terms in the greatest common divisor of sparse polynomials.

Challenge 5

As a special case of Challenge 1 we can ask the following.
Find useful bounds on the number of terms in the greatest common divisor of sparse polynomials.
Failing this, one might ask for such a bound for non-cyclotomic factors.

Challenge 6

By analogy with Challenge 3, we can also pose the following.

Challenge 6

By analogy with Challenge 3, we can also pose the following. Find an algorithm for computing $\gcd(f, g)$ which is polynomial-time in t_f, t_g *and* $t_{\gcd(f,g)}$.

Challenge 6

By analogy with Challenge 3, we can also pose the following.
Find an algorithm for computing $\gcd(f, g)$ which is polynomial-time in t_f, t_g *and* $t_{\gcd(f,g)}$.
Again, we might restrict ourselves to the non-cyclotomic case.

Square-free decomposition

We know this can be done by gcd, but in fact they are equivalent

Square-free decomposition

We know this can be done by gcd, but in fact they are equivalent

Theorem (KarpinskiShparlinski1999)

Over $\mathbf{Z}$ *and in the standard encoding, the two problems*

1. *deciding if a polynomial is square-free*
2. *deciding if two polynomials have a non-trivial g.c.d.*

are equivalent under randomized polynomial-time reduction.

Square-free decomposition

We know this can be done by gcd, but in fact they are equivalent

Theorem (KarpinskiShparlinski1999)

Over $\mathbf{Z}$ and in the standard encoding, the two problems

1. *deciding if a polynomial is square-free*
2. *deciding if two polynomials have a non-trivial g.c.d.*

are equivalent under randomized polynomial-time reduction.

Hence, in the light of Theorem 1, determining square-freeness is hard, at least when polynomials with cyclotomic factors are involved.

Square-free decomposition

We know this can be done by gcd, but in fact they are equivalent

Theorem (KarpinskiShparlinski1999)

Over $\mathbf{Z}$ *and in the standard encoding, the two problems*

1. *deciding if a polynomial is square-free*
2. *deciding if two polynomials have a non-trivial g.c.d.*

are equivalent under randomized polynomial-time reduction.

Hence, in the light of Theorem 1, determining square-freeness is hard, at least when polynomials with cyclotomic factors are involved.

A fortiori, computing the square-free decomposition is hard, at least when cyclotomics are involved. This is certainly the case if we want a full decomposition in the standard model, as the trivial example of

$$x^{p+1} - x^p - x + 1 = (x-1)^2(x^{p-1} + \cdots + 1) \qquad (1)$$

shows.

Challenge 6a

Find a polynomial-time algorithm for the *shape* of the square-free decomposition of a sparse polynomial.

Challenge 6a

Find a polynomial-time algorithm for the *shape* of the square-free decomposition of a sparse polynomial.
We might also ask about the square-free decomposition of cyclotomic-free polynomials.

Challenge 6a

Find a polynomial-time algorithm for the *shape* of the square-free decomposition of a sparse polynomial.
We might also ask about the square-free decomposition of cyclotomic-free polynomials.
Note, however, various results about polynomials which get sparser when we square them

Perfect Powers

However, a positive result for the standard representation in this area is provided by Giesbrecht & Roche, who give a Las Vegas polynomial-time algorithm for determining *whether* a given sparse f (not of the form x^n, else the number of possibilities is potentially vast) is h^r, and r itself.

Perfect Powers

However, a positive result for the standard representation in this area is provided by Giesbrecht & Roche, who give a Las Vegas polynomial-time algorithm for determining *whether* a given sparse f (not of the form x^n, else the number of possibilities is potentially vast) is h^r, and r itself.
One obvious question is whether h has to be sparse if f is. They conjecture that it does: more precisely the following.

Conjecture (GiesbrechtRoche2008a)

For $r, s \in \mathbf{N}$ *and* $h \in \mathbf{Z}[z]$ *with* $d_h = s$*, then* $\hat{t}_{h^i} < \hat{t}_{h^r} + r$ *for* $1 \leq i < n$*, where* $\hat{t}_f = t_{f \pmod{x^{2s}}}$*.*

Perfect Powers

However, a positive result for the standard representation in this area is provided by Giesbrecht & Roche, who give a Las Vegas polynomial-time algorithm for determining *whether* a given sparse f (not of the form x^n, else the number of possibilities is potentially vast) is h^r, and r itself.
One obvious question is whether h has to be sparse if f is. They conjecture that it does: more precisely the following.

Conjecture (GiesbrechtRoche2008a)

For $r, s \in \mathbf{N}$ and $h \in \mathbf{Z}[z]$ with $d_h = s$, then $\hat{t}_{h^i} < \hat{t}_{h^r} + r$ for $1 \leq i < n$, where $\hat{t}_f = t_{f \pmod{x^{2s}}}$.

Assuming this conjecture, they can recover h in polynomial time.

Factorization

In the light of

Factorization

In the light of

- Cyclotomics

Factorization

In the light of

- Cyclotomics
- Bremner's polynomials and the absence of an answer to Challenge 1

Factorization

In the light of

- Cyclotomics
- Bremner's polynomials and the absence of an answer to Challenge 1

Factorization

In the light of

- Cyclotomics
- Bremner's polynomials and the absence of an answer to Challenge 1

we might be inclined to give up.

Factorization

In the light of

- Cyclotomics
- Bremner's polynomials and the absence of an answer to Challenge 1

we might be inclined to give up.
But there is some good news.

Lenstra's Theorem

There is a deterministic algorithm that, for some positive real number c, has the following property: given an algebraic number field K, a sparsely represented non-zero polunomial $f \in K[x]$ and a positive integer d, the algorithm finds all monic irreducible factors of f in $K[x]$ of degree at most d, as well as their multiplicities, and it spends time at most $(l + d)^c$, where l denotes the length of the input data (i.e. $t_f \log(d_f|f|)$)

Challenge 7

Understand the complexity of this result in practice.

Challenge 7

Understand the complexity of this result in practice.
In particular, we would like to know the value of c in the special case when K is $\mathbf{Q}$.

Challenge 7

Understand the complexity of this result in practice.
In particular, we would like to know the value of c in the special case when K is $\mathbf{Q}$.
Also, $(l+d)^c$ is a very neat formulation, but the dependencies on d and l are probably different in reality.

Polynomial Decomposition

i.e. is $f(x) = g(h(x))$? The case $g(x) = x^d$ is that of perfect powers. In general, we have two recent results

Polynomial Decomposition

i.e. is $f(x) = g(h(x))$? The case $g(x) = x^d$ is that of perfect powers. In general, we have two recent results

Theorem (Zannier2007)

If h is not of the form $ax^n + b$, then $d_g \leq 2t_f(t_f - 1)$

Polynomial Decomposition

i.e. is $f(x) = g(h(x))$? The case $g(x) = x^d$ is that of perfect powers. In general, we have two recent results

Theorem (Zannier2007)

If h is not of the form $ax^n + b$, then $d_g \leq 2t_f(t_f - 1)$

Theorem (Zannier2008)

There exists a computable function $\mathcal{B}$ such that if $g, h \in \mathbf{C}[x]$ are non-constant polynomials with $f(x) = g(h(x))$, then $t_h \leq \mathcal{B}(t_f)$.

Polynomial Decomposition

i.e. is $f(x) = g(h(x))$? The case $g(x) = x^d$ is that of perfect powers. In general, we have two recent results

Theorem (Zannier2007)

If h is not of the form $ax^n + b$, then $d_g \leq 2t_f(t_f - 1)$

Theorem (Zannier2008)

There exists a computable function $\mathcal{B}$ such that if $g, h \in \mathbf{C}[x]$ are non-constant polynomials with $f(x) = g(h(x))$, then $t_h \leq \mathcal{B}(t_f)$.

In other words, if f is of high degree, but has few terms, then g cannot be of high degree (and therefore implicitly has comparatively few terms) and h has few terms. However, these bounds still allow for a surprising degree of cancellation in $f(x) = g(h(x))$.

Polynomial Decomposition

i.e. is $f(x) = g(h(x))$? The case $g(x) = x^d$ is that of perfect powers. In general, we have two recent results

Theorem (Zannier2007)

If h is not of the form $ax^n + b$, then $d_g \leq 2t_f(t_f - 1)$

Theorem (Zannier2008)

There exists a computable function $\mathcal{B}$ such that if $g, h \in \mathbf{C}[x]$ are non-constant polynomials with $f(x) = g(h(x))$, then $t_h \leq \mathcal{B}(t_f)$.

In other words, if f is of high degree, but has few terms, then g cannot be of high degree (and therefore implicitly has comparatively few terms) and h has few terms. However, these bounds still allow for a surprising degree of cancellation in $f(x) = g(h(x))$.

Some cancellation is certainly possible, though

Challenge 8

Understand the complexity of this result in practice.