How to multiply big integers

Standard idea: Use polynomial with coefficients in $\{0, 1, \ldots, 9\}$ to represent integer in radix 10.

Example of representation:
$839 = 8 \cdot 10^2 + 3 \cdot 10^1 + 9 \cdot 10^0 =$ value (at $t = 10$) of polynomial $8t^2 + 3t^1 + 9t^0$.

Convenient to express polynomial inside computer as array $9, 3, 8$ (or $9, 3, 8, 0$ or $9, 3, 8, 0, 0$ or $\ldots$): "`p[0] = 9; p[1] = 3; p[2] = 8`"

Multiply two integers by multiplying polynomials that represent the integers.

Polynomial multiplication involves *small* integer coefficients. Have split one big multiplication into many small operations.

Example, squaring 839:

$(8t^2 + 3t^1 + 9t^0)^2 =$
$8t^2(8t^2 + 3t^1 + 9t^0) +$
$3t^1(8t^2 + 3t^1 + 9t^0) +$
$9t^0(8t^2 + 3t^1 + 9t^0) =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$.

multiply big integers

d idea: Use polynomial

efficients in $\{0, 1, \ldots, 9\}$

sent integer in radix 10.

e of representation:

$\cdot 10^2 + 3 \cdot 10^1 + 9 \cdot 10^0 =$

t $t = 10$) of polynomial

$^1 + 9t^0$.

ent to express polynomial

omputer as array 9, 3, 8

8, 0 or 9, 3, 8, 0, 0 or . . .):

9; p[1] = 3; p[2] = 8"

Multiply two integers by multiplying polynomials that represent the integers.

Polynomial multiplication involves *small* integer coefficients. Have split one big multiplication into many small operations.

Example, squaring 839:

$(8t^2 + 3t^1 + 9t^0)^2 =$

$8t^2(8t^2 + 3t^1 + 9t^0) +$

$3t^1(8t^2 + 3t^1 + 9t^0) +$

$9t^0(8t^2 + 3t^1 + 9t^0) =$

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$.

Oops, pr

usually h

So "carr

$ct^j \rightarrow \lfloor$

Example

$64t^4 + 4$

$64t^4 + 4$

$64t^4 + 4$

$64t^4 + 6$

$70t^4 + 3$

$7t^5 + 0t$

In other

ig integers

e polynomial

n $\{0, 1, \ldots, 9\}$

er in radix 10.

entation:

$10^1 + 9 \cdot 10^0 =$

of polynomial

ress polynomial

s array $9, 3, 8$

$3, 8, 0, 0$ or $\ldots$):

= 3; p[2] = 8"

Multiply two integers
by multiplying polynomials
that represent the integers.

Polynomial multiplication
involves *small* integer coefficients.
Have split one big multiplication
into many small operations.

Example, squaring 839:

$(8t^2 + 3t^1 + 9t^0)^2 =$

$8t^2(8t^2 + 3t^1 + 9t^0) +$

$3t^1(8t^2 + 3t^1 + 9t^0) +$

$9t^0(8t^2 + 3t^1 + 9t^0) =$

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0.$

Oops, product pol

usually has coeffic

So "carry" extra d

$ct^j \to \lfloor c/10 \rfloor t^{j+1}$

Example, squaring

$64t^4 + 48t^3 + 153t$

$64t^4 + 48t^3 + 153$

$64t^4 + 48t^3 + 159$

$64t^4 + 63t^3 + 9t^2$

$70t^4 + 3t^3 + 9t^2$

$7t^5 + 0t^4 + 3t^3 +$

In other words, 83

s

ial

, 9}

10.

$10^0 =$

mial

omial

3, 8

$\ldots$):

= 8"

Multiply two integers
by multiplying polynomials
that represent the integers.

Polynomial multiplication
involves *small* integer coefficients.
Have split one big multiplication
into many small operations.

Example, squaring 839:

$(8t^2 + 3t^1 + 9t^0)^2 =$
$8t^2(8t^2 + 3t^1 + 9t^0) +$
$3t^1(8t^2 + 3t^1 + 9t^0) +$
$9t^0(8t^2 + 3t^1 + 9t^0) =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0.$

Oops, product polynomial
usually has coefficients > 9.
So "carry" extra digits:
$ct^j \to \lfloor c/10 \rfloor\, t^{j+1} + (c$ mod

Example, squaring 839:

$64t^4 + 48t^3 + 153t^2 + 54t^1$
$64t^4 + 48t^3 + 153t^2 + 62t^1$
$64t^4 + 48t^3 + 159t^2 + 2t^1$
$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1$
$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t$
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1$

In other words, $839^2 = 7039$

Multiply two integers
by multiplying polynomials
that represent the integers.

Polynomial multiplication
involves *small* integer coefficients.
Have split one big multiplication
into many small operations.

Example, squaring 839:
$(8t^2 + 3t^1 + 9t^0)^2 =$
$8t^2(8t^2 + 3t^1 + 9t^0) +$
$3t^1(8t^2 + 3t^1 + 9t^0) +$
$9t^0(8t^2 + 3t^1 + 9t^0) =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$.

Oops, product polynomial
usually has coefficients > 9.
So "carry" extra digits:
$ct^j \to \lfloor c/10 \rfloor t^{j+1} + (c \bmod 10)t^j$.

Example, squaring 839:
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$64t^4 + 48t^3 + 153t^2 + 62t^1 + 1t^0$;
$64t^4 + 48t^3 + 159t^2 + 2t^1 + 1t^0$;
$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1t^0$;
$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

In other words, $839^2 = 703921$.

two integers
plying polynomials
resent the integers.

ial multiplication
small integer coefficients.
lit one big multiplication
ny small operations.

e, squaring 839:

$t^1 + 9t^0)^2 =$

$+ 3t^1 + 9t^0) +$

$+ 3t^1 + 9t^0) +$

$+ 3t^1 + 9t^0) =$

$8t^3 + 153t^2 + 54t^1 + 81t^0.$

Oops, product polynomial
usually has coefficients > 9.
So "carry" extra digits:

$ct^j \to \lfloor c/10 \rfloor t^{j+1} + (c \bmod 10)t^j.$

Example, squaring 839:

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0;$

$64t^4 + 48t^3 + 153t^2 + 62t^1 + 1t^0;$

$64t^4 + 48t^3 + 159t^2 + 2t^1 + 1t^0;$

$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1t^0;$

$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0;$

$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0.$

In other words, $839^2 = 703921$.

What op

divide b

gers
ynomials
integers.

lication
eger coefficients.
multiplication
perations.

839:
$^2 =$
$t^0) +$
$t^0) +$
$t^0) =$
$t^2 + 54t^1 + 81t^0.$

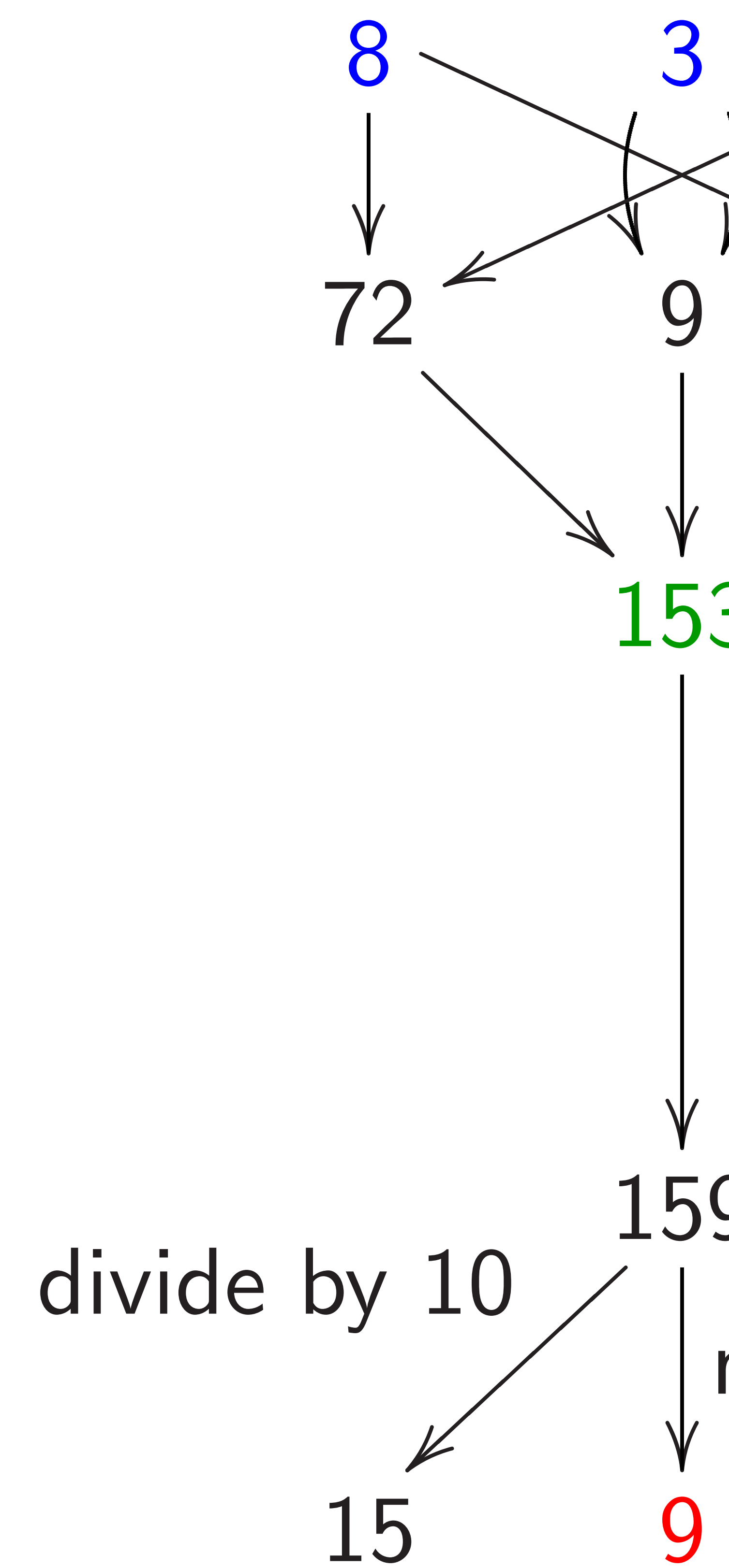
Oops, product polynomial usually has coefficients > 9.
So "carry" extra digits:
$ct^j \to \lfloor c/10 \rfloor t^{j+1} + (c \bmod 10)t^j.$

Example, squaring 839:

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0;$
$64t^4 + 48t^3 + 153t^2 + 62t^1 + 1t^0;$
$64t^4 + 48t^3 + 159t^2 + 2t^1 + 1t^0;$
$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1t^0;$
$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0;$
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0.$

In other words, $839^2 = 703921$.

What operations w



cients.

ation

$+ 81t^0$.

Oops, product polynomial usually has coefficients > 9.

So "carry" extra digits:

$ct^j \to \lfloor c/10 \rfloor t^{j+1} + (c \bmod 10)t^j$.

Example, squaring 839:

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;

$64t^4 + 48t^3 + 153t^2 + 62t^1 + 1t^0$;

$64t^4 + 48t^3 + 159t^2 + 2t^1 + 1t^0$;

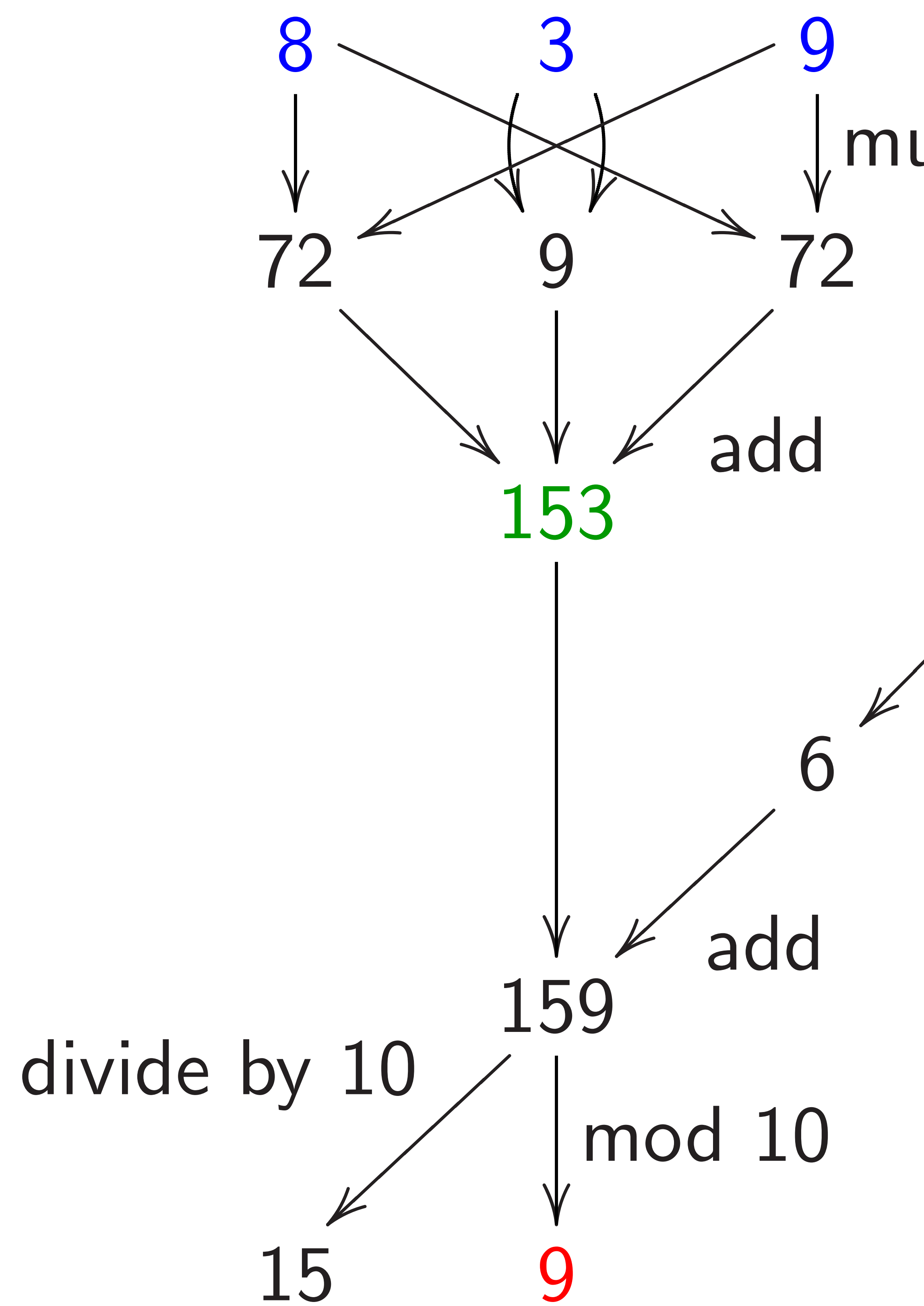
$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1t^0$;

$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$;

$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

In other words, $839^2 = 703921$.

What operations were used



Oops, product polynomial usually has coefficients > 9.

So "carry" extra digits:

$ct^j \to \lfloor c/10 \rfloor t^{j+1} + (c \bmod 10)t^j$.

Example, squaring 839:

$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;

$64t^4 + 48t^3 + 153t^2 + 62t^1 + 1t^0$;

$64t^4 + 48t^3 + 159t^2 + 2t^1 + 1t^0$;

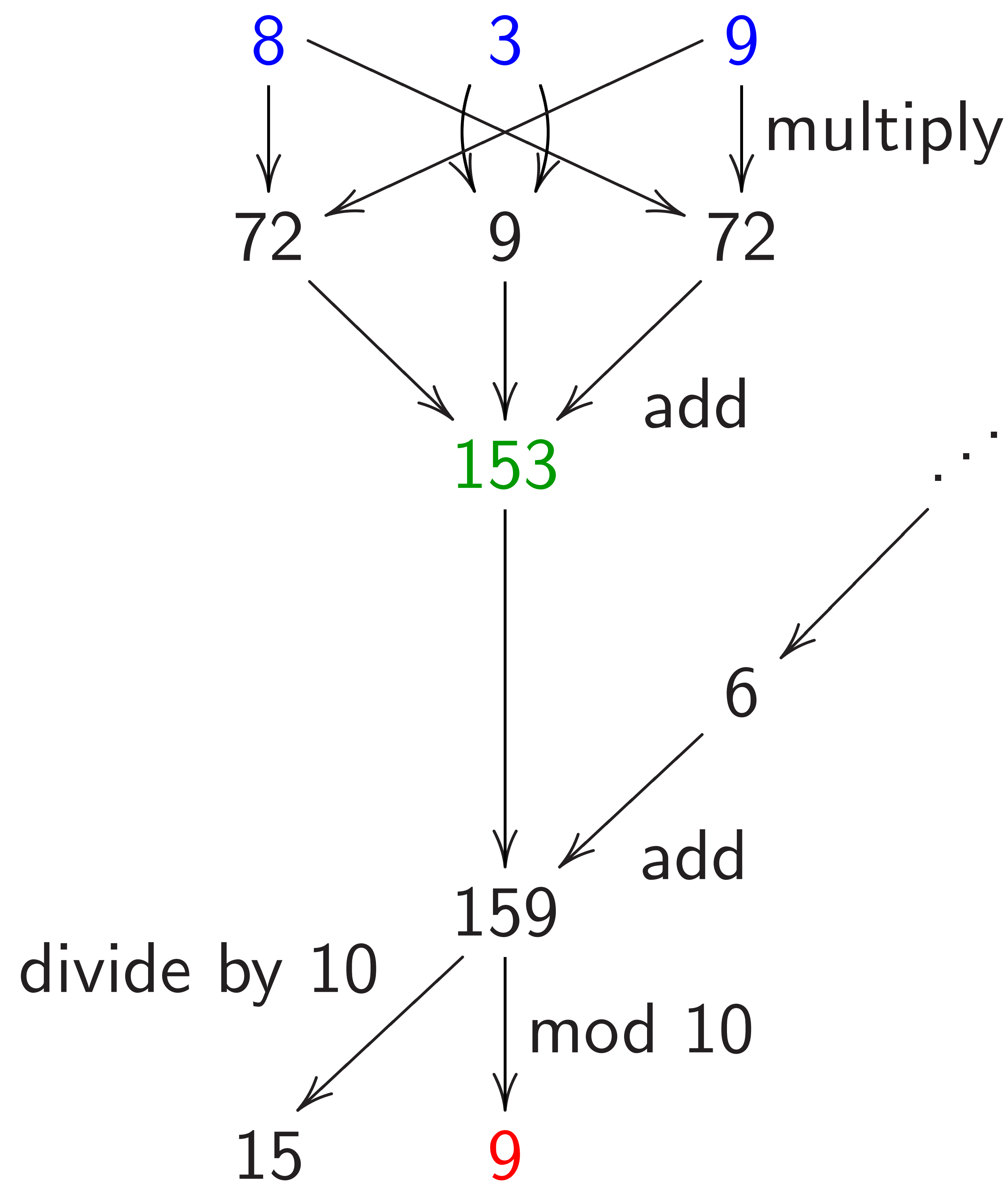
$64t^4 + 63t^3 + 9t^2 + 2t^1 + 1t^0$;

$70t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$;

$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

In other words, $839^2 = 703921$.

What operations were used here?



roduct polynomial

has coefficients > 9.

y" extra digits:

$c/10\rfloor\, t^{j+1} + (c \bmod 10)t^j$.

e, squaring 839:

$8t^3 + 153t^2 + 54t^1 + 81t^0$;

$48t^3 + 153t^2 + 62t^1 + 1t^0$;

$48t^3 + 159t^2 + 2t^1 + 1t^0$;

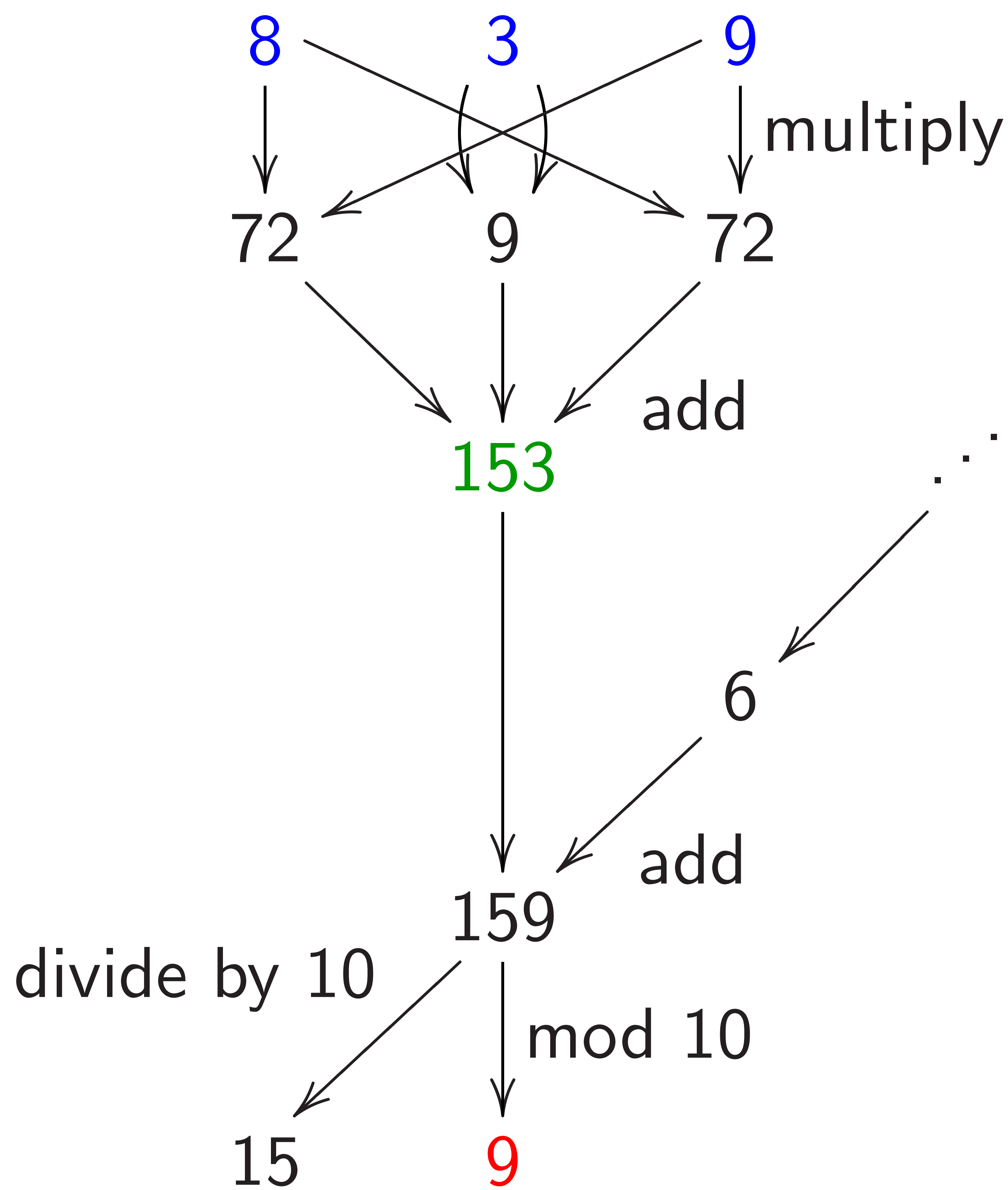
$63t^3 + 9t^2 + 2t^1 + 1t^0$;

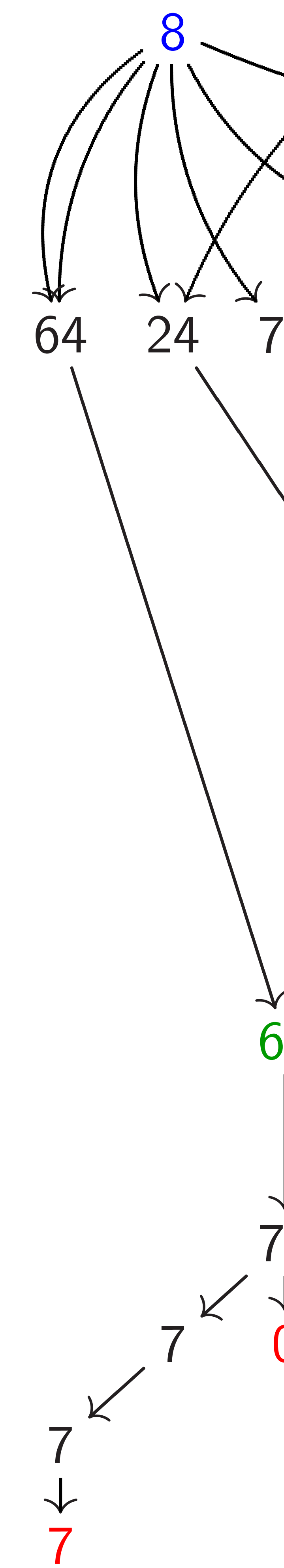
$3t^3 + 9t^2 + 2t^1 + 1t^0$;

$^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

words, $839^2 = 703921$.

What operations were used here?





ynomial

ients > 9.

ligits:

$+ (c \bmod 10) t^j$.

839:

$t^2 + 54t^1 + 81t^0$;

$t^2 + 62t^1 + 1t^0$;

$t^2 + 2t^1 + 1t^0$;

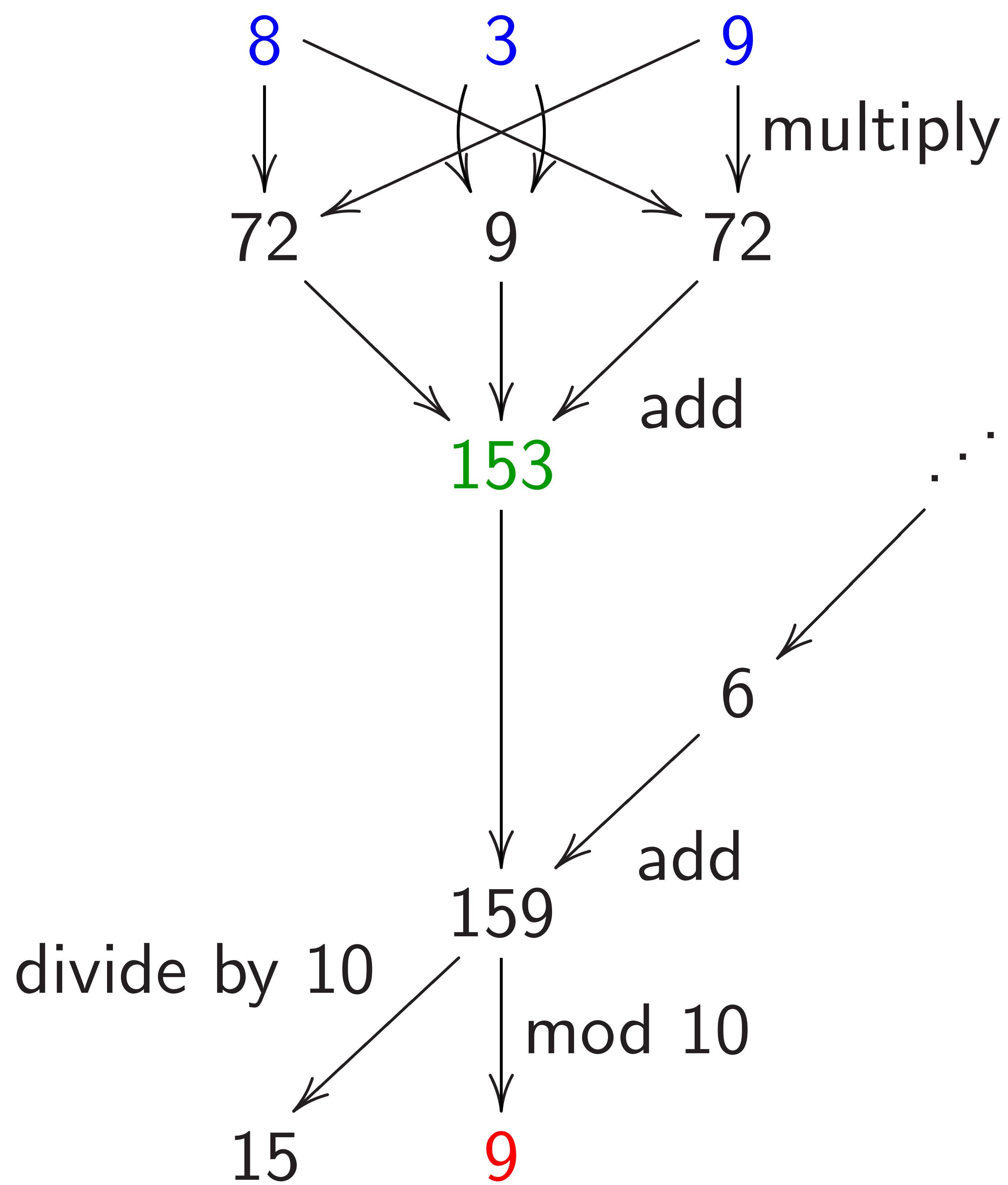
$+ 2t^1 + 1t^0$;

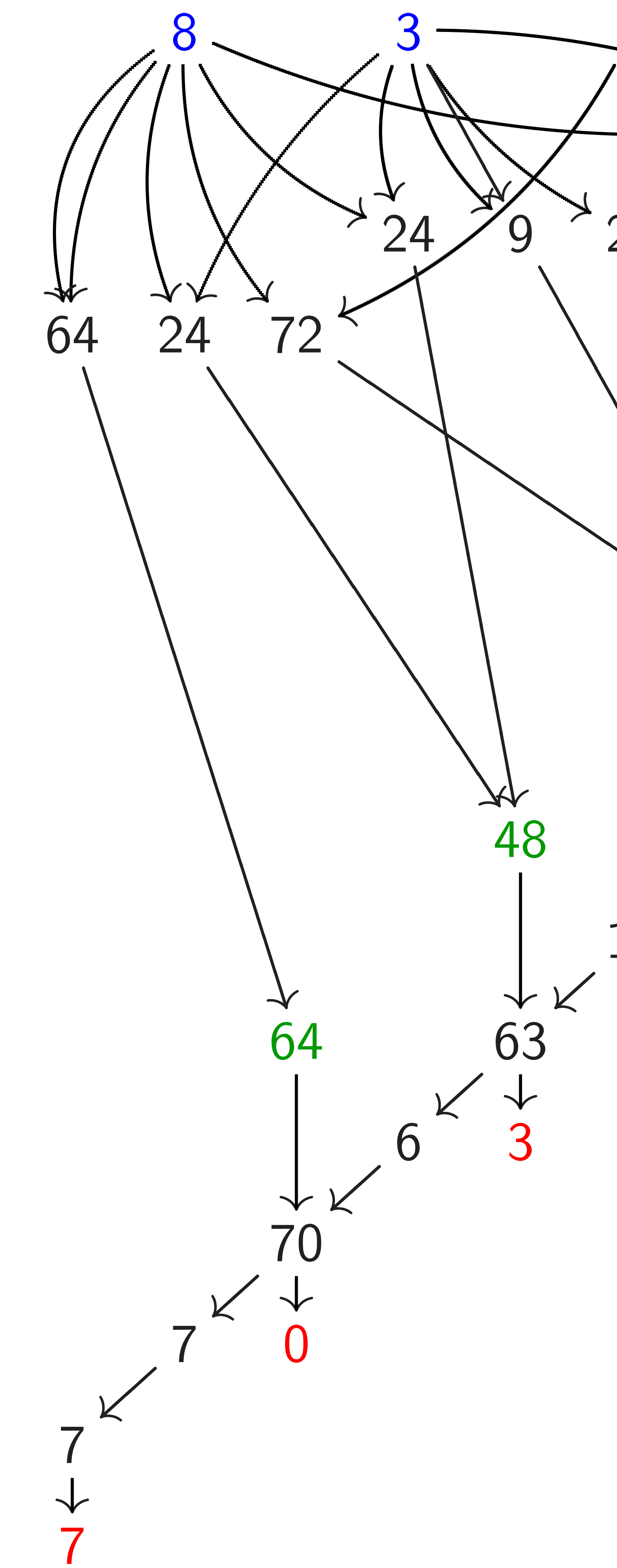
$+ 2t^1 + 1t^0$;

$9t^2 + 2t^1 + 1t^0$.

$9^2 = 703921$.

What operations were used here?





$10)t^j.$

$+81t^0;$

$+1t^0;$

$+1t^0;$

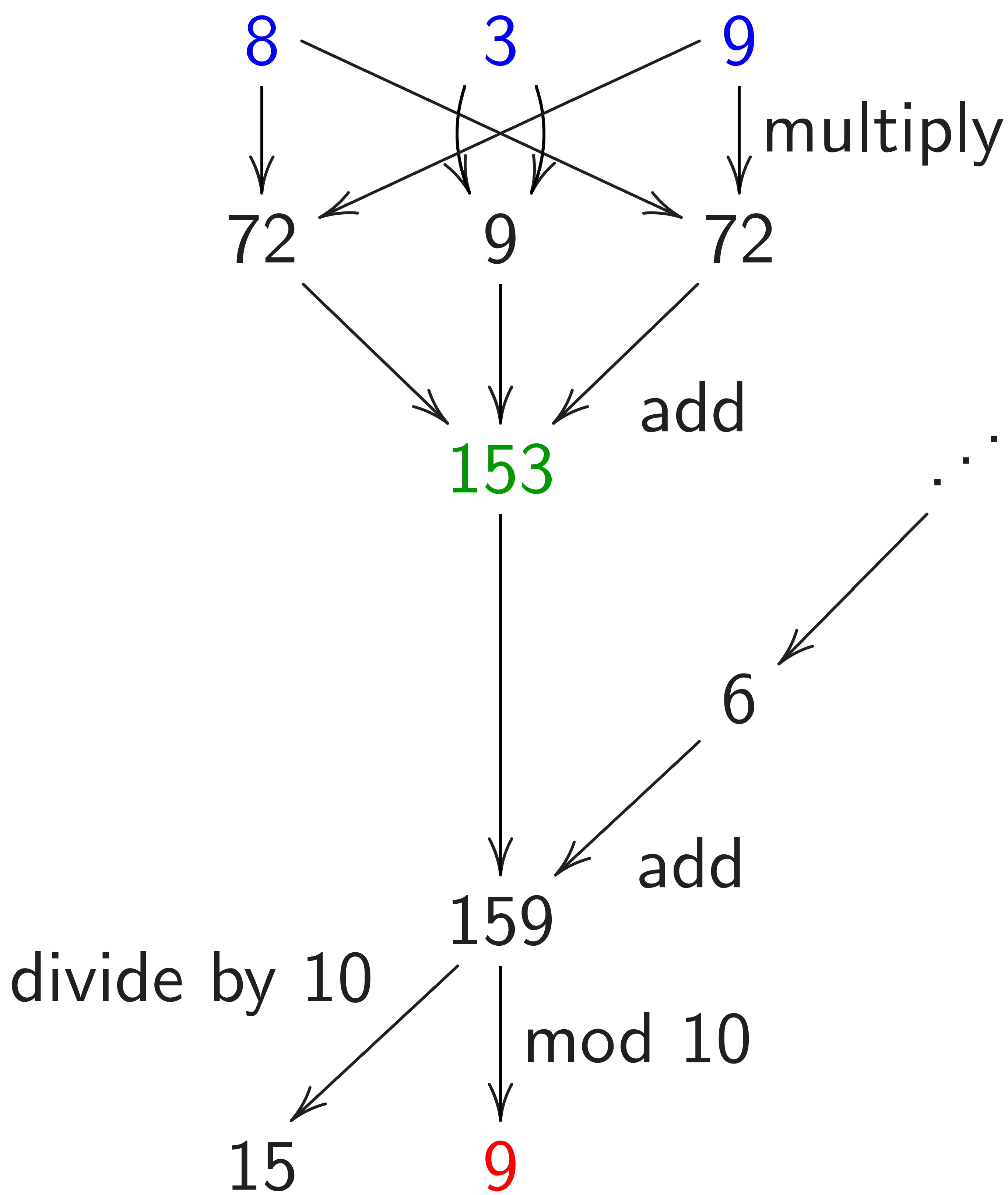
$t^0;$

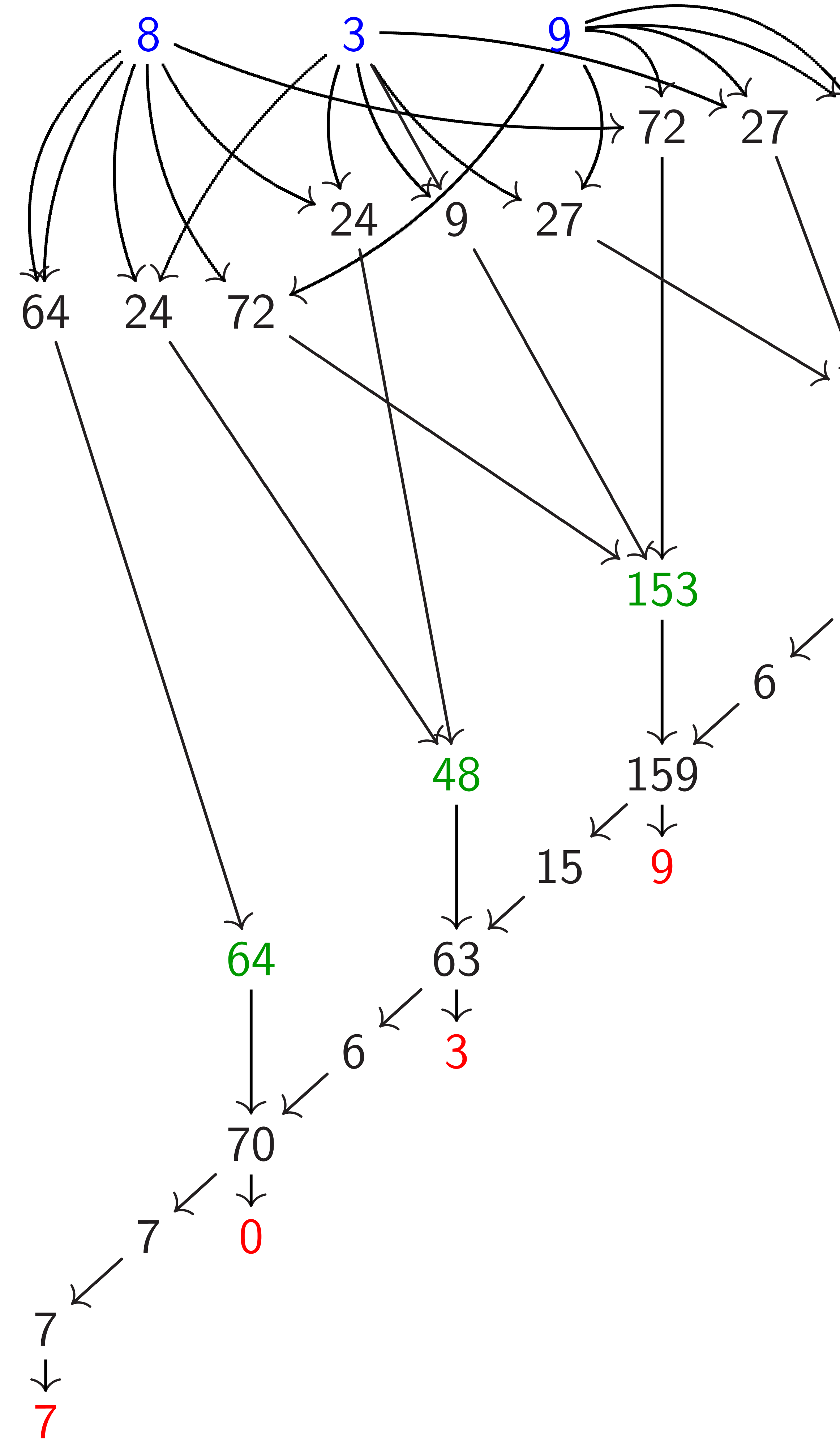
$t^0;$

$+1t^0.$

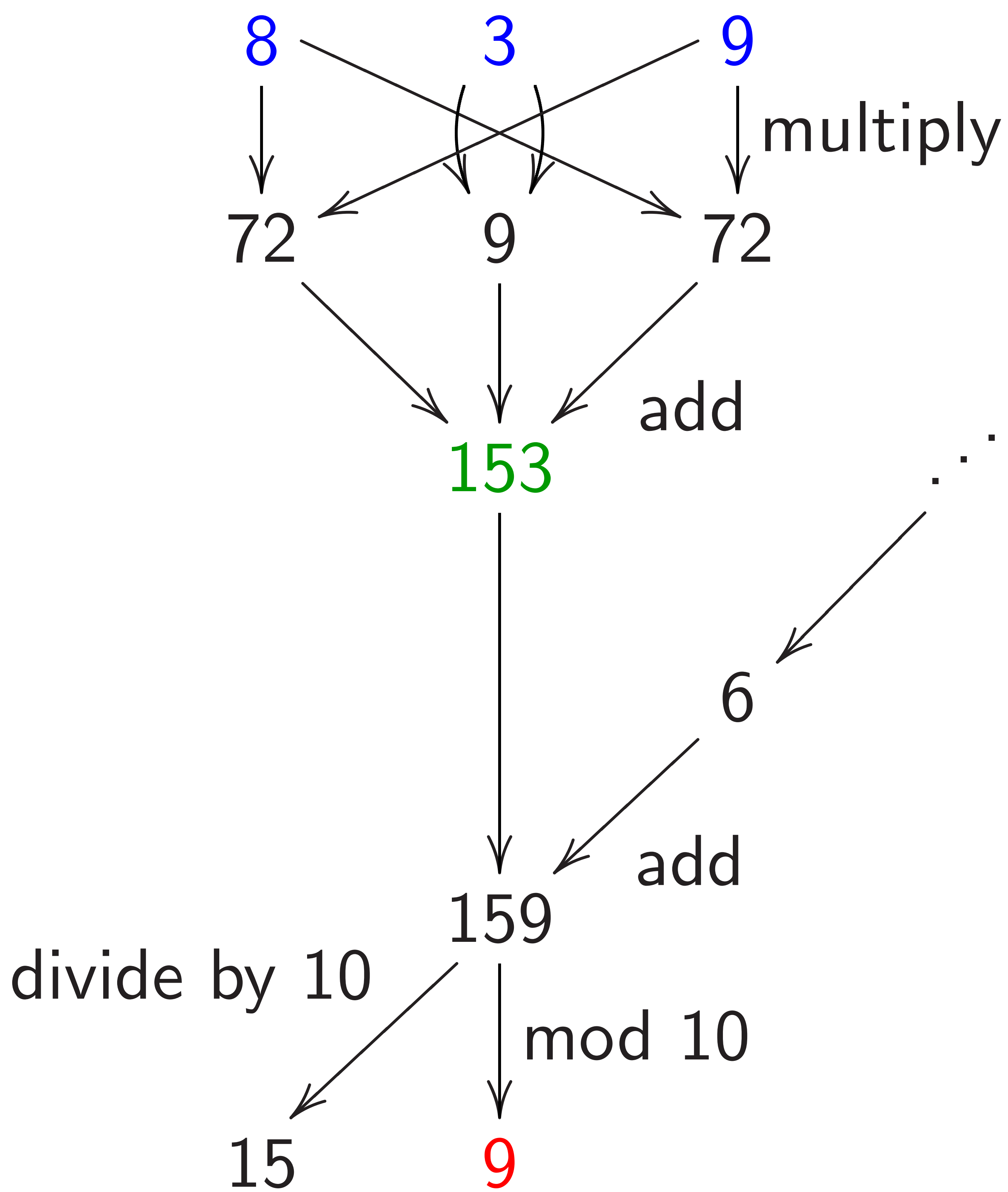
921.

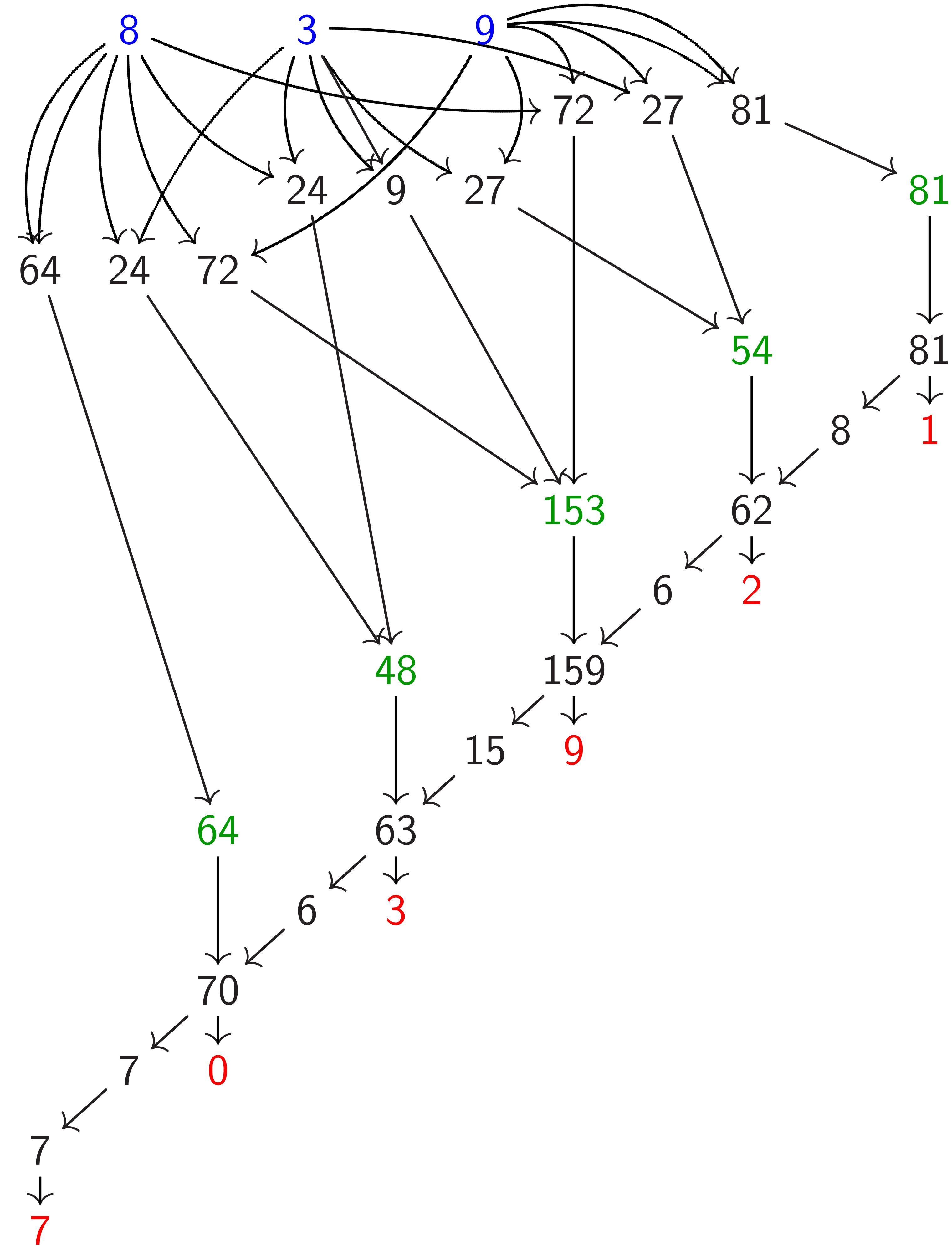
What operations were used here?



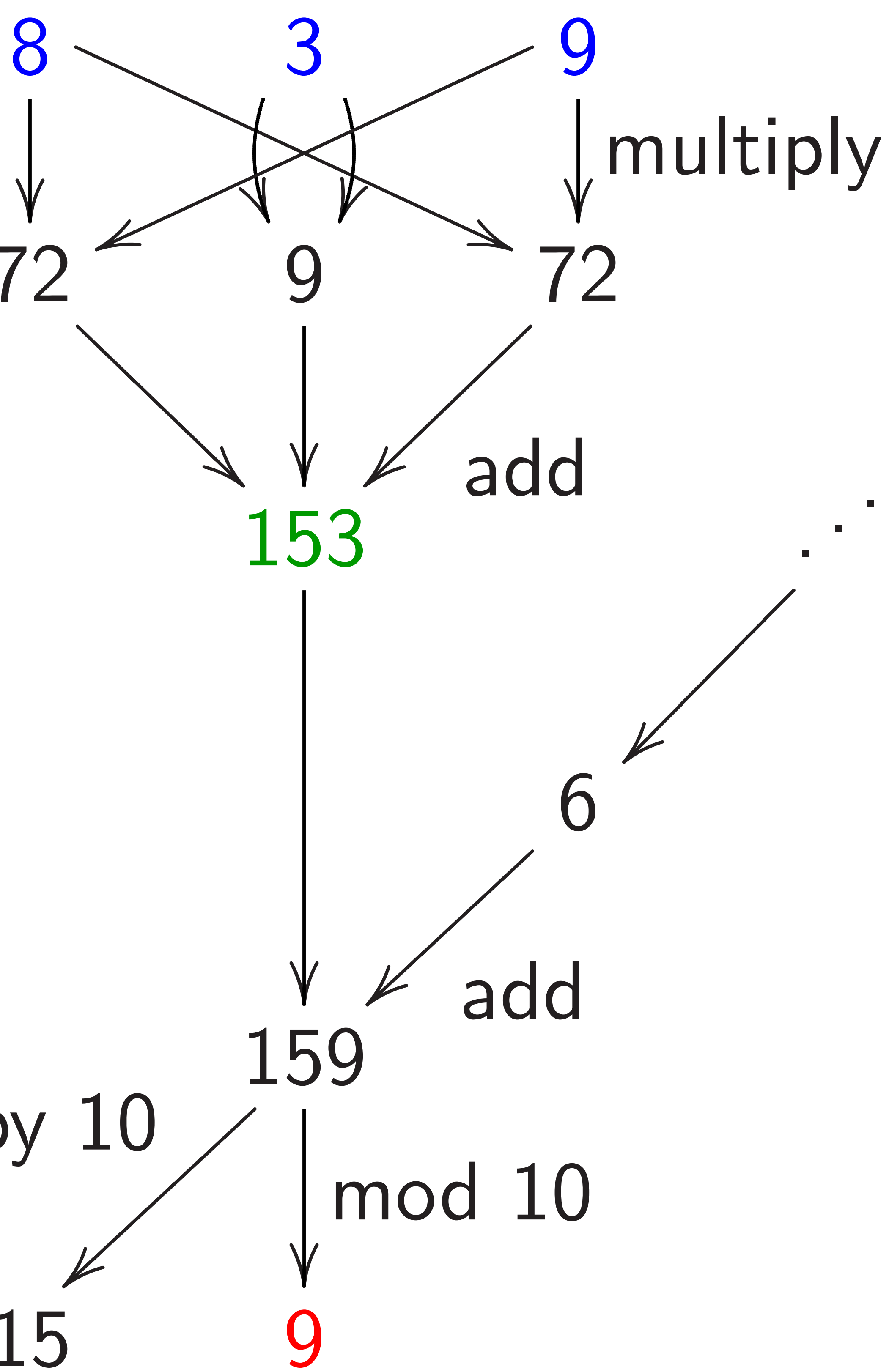


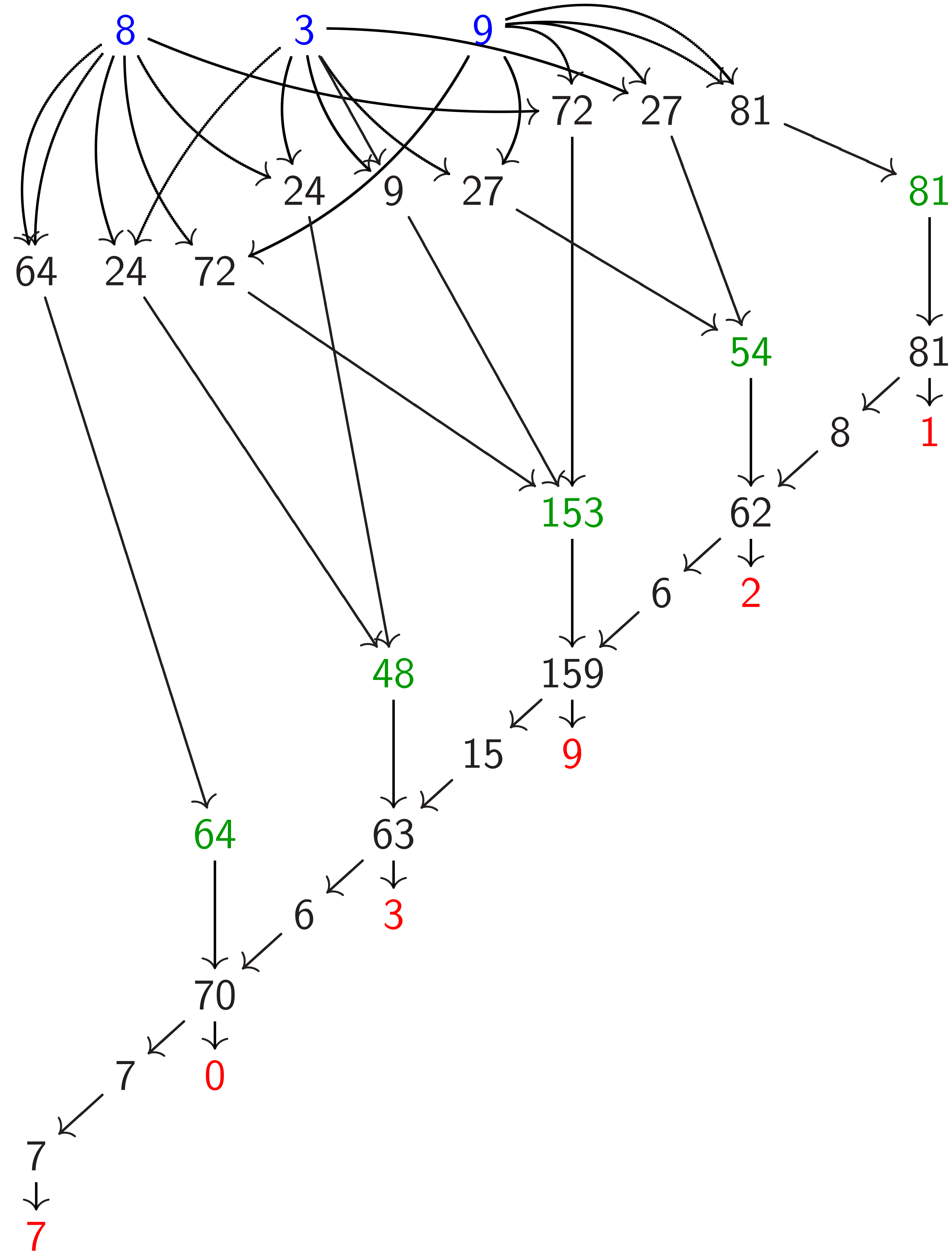
What operations were used here?





perations were used here?





The scal

$839 = 8$

value (a

$800t^2 +$

Squaring

$640000t$

$540t^1 +$

Carrying

$640000t$

$540t^1 +$

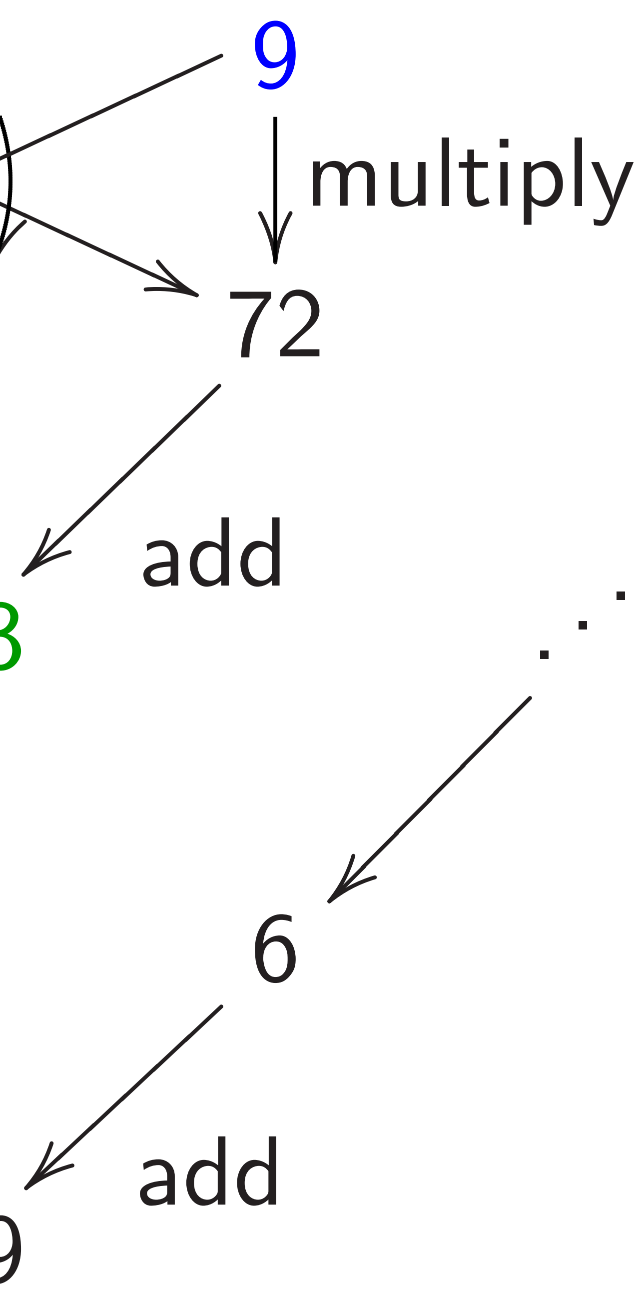
$640000t$

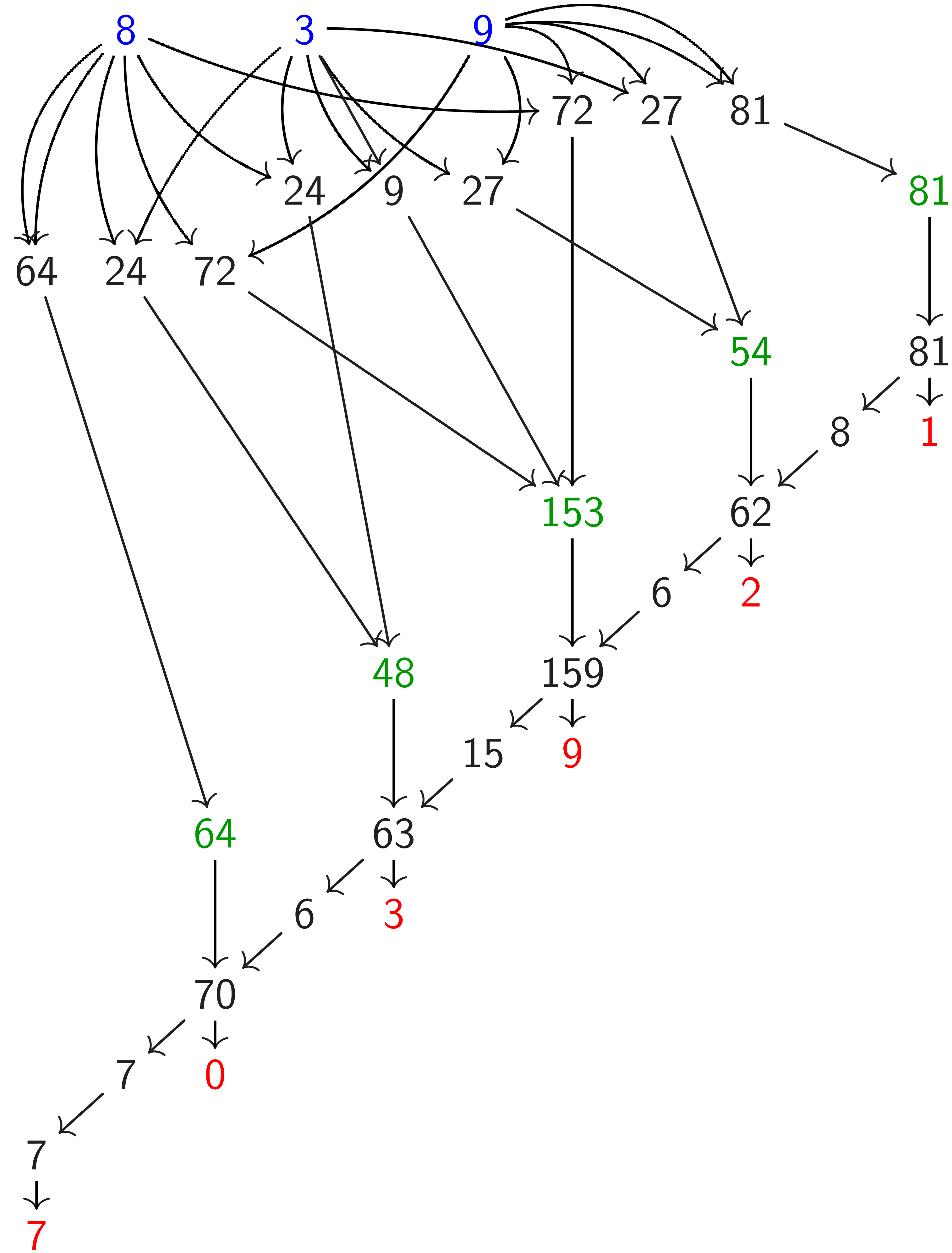
$620t^1 +$

$700000t$

$20t^1 + 1$

were used here?





The scaled variatio

$839 = 800 + 30 +$

value (at $t = 1$) o

$800t^2 + 30t^1 + 9t$

Squaring: $(800t^2$

$640000t^4 + 48000$

$540t^1 + 81t^0$.

Carrying:

$640000t^4 + 48000$

$540t^1 + 81t^0$;

$640000t^4 + 48000$

$620t^1 + 1t^0$; .

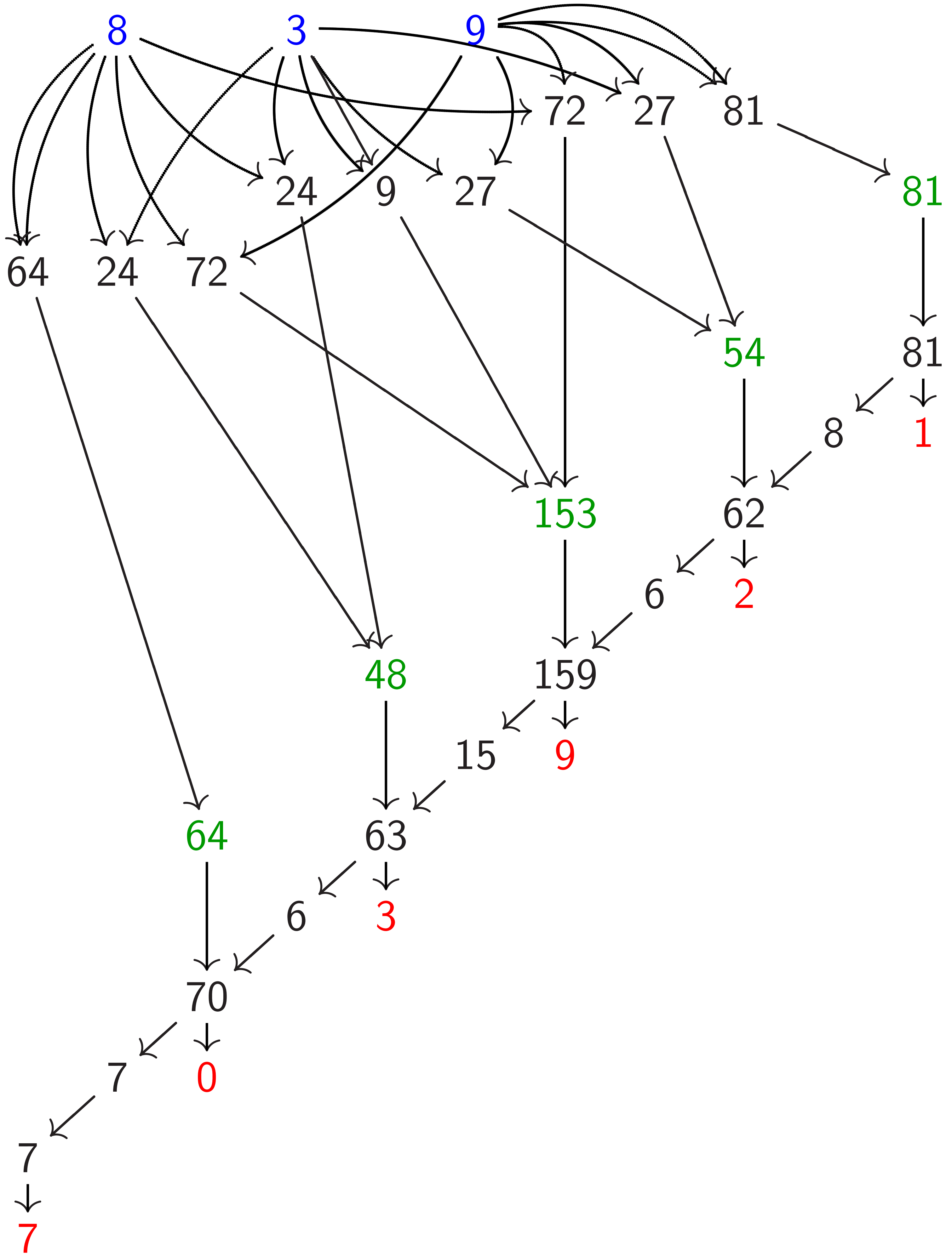
$700000t^5 + 0t^4 + 3$

$20t^1 + 1t^0$.

here?

ultiply

. . .



The scaled variation

$839 = 800 + 30 + 9 =$

value (at $t = 1$) of polynom

$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2 + 30t^1 + 9t$

$640000t^4 + 48000t^3 + 1530$

$540t^1 + 81t^0$.

Carrying:

$640000t^4 + 48000t^3 + 1530$

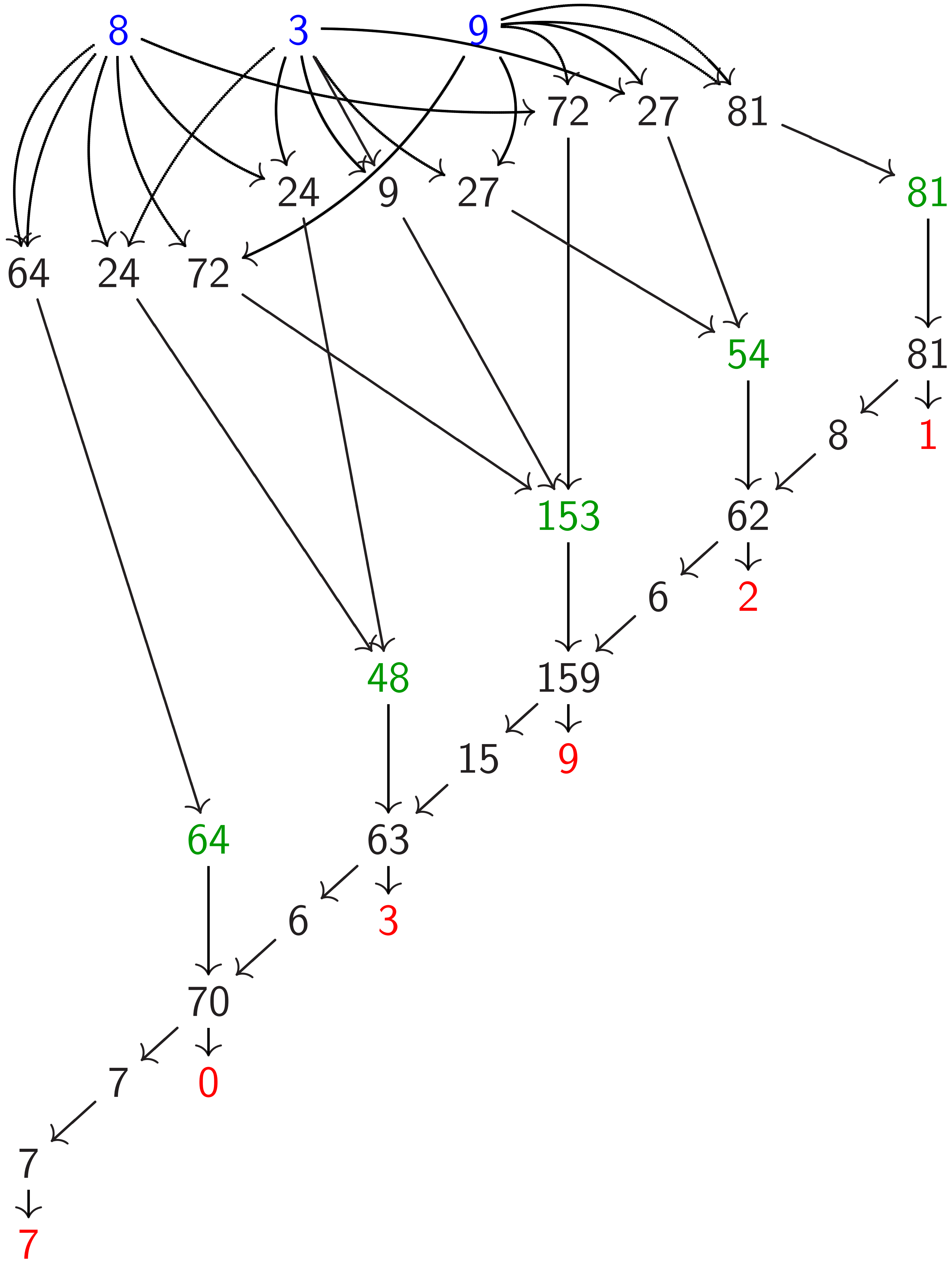
$540t^1 + 81t^0$;

$640000t^4 + 48000t^3 + 1530$

$620t^1 + 1t^0$; . . .

$700000t^5 + 0t^4 + 3000t^3 + 9$

$20t^1 + 1t^0$.

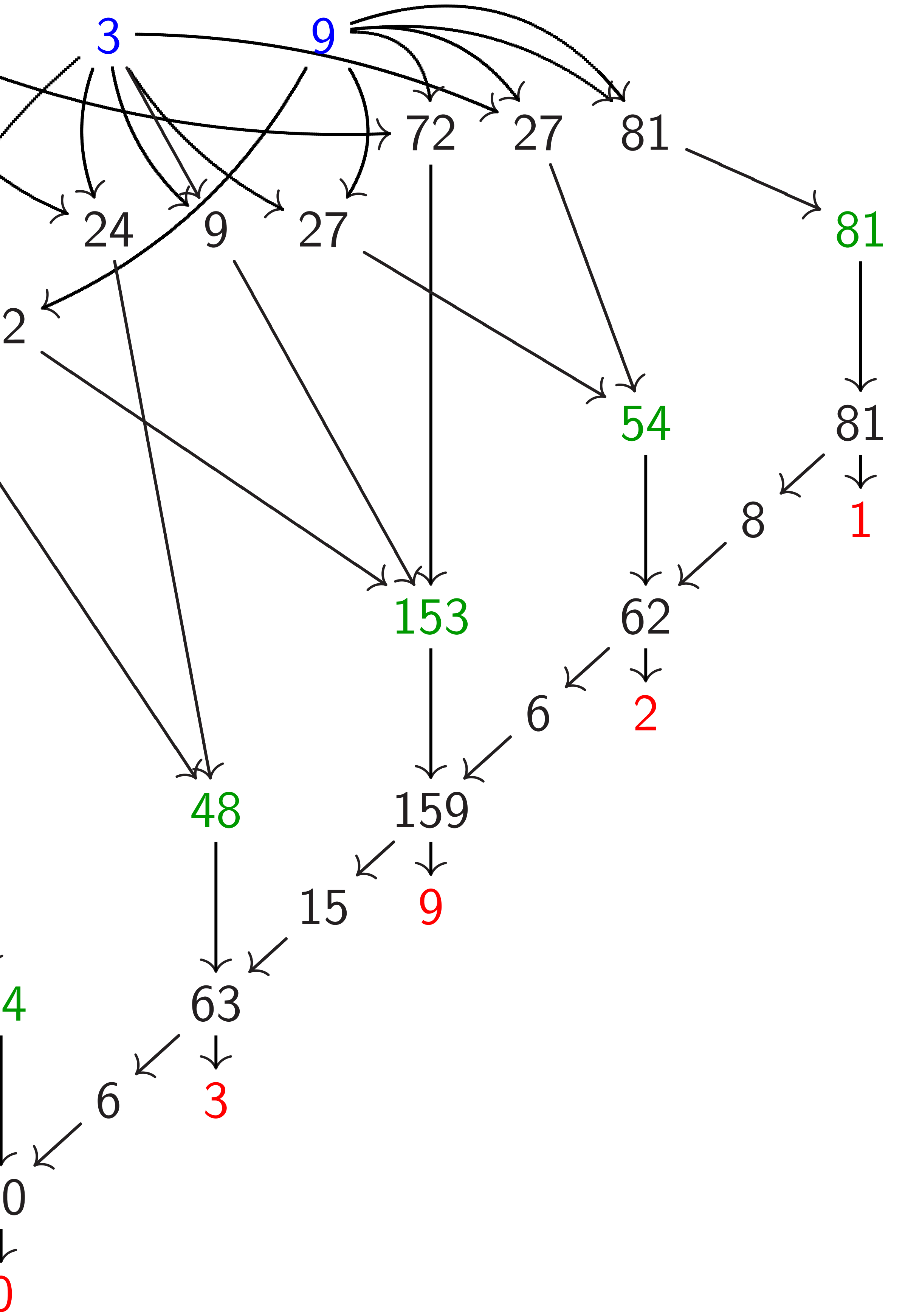


The scaled variation

$839 = 800 + 30 + 9 =$
value (at $t = 1$) of polynomial
$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2 + 30t^1 + 9t^0)^2 =$
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$.

Carrying:
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$;
$640000t^4 + 48000t^3 + 15300t^2 + 620t^1 + 1t^0$; . . .
$700000t^5 + 0t^4 + 3000t^3 + 900t^2 + 20t^1 + 1t^0$.



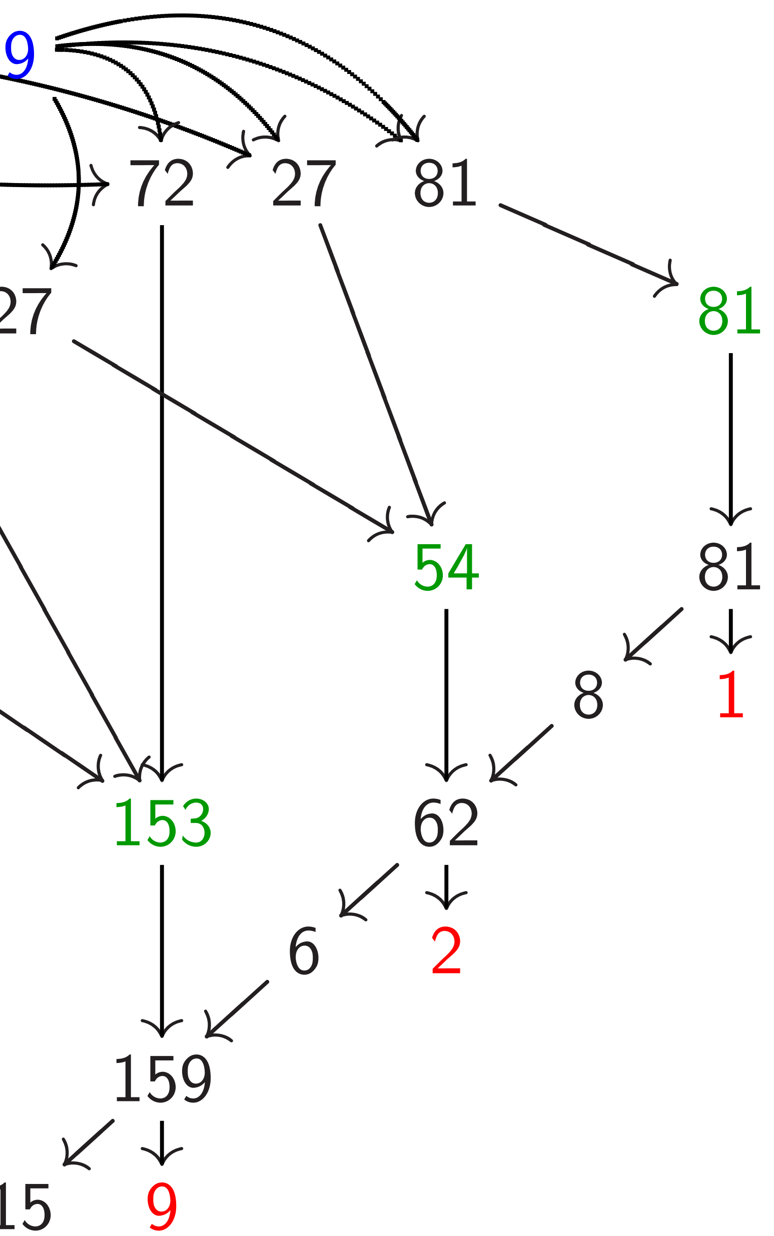
The scaled variation

$839 = 800 + 30 + 9 =$
value (at $t = 1$) of polynomial
$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2+30t^1+9t^0)^2 =$
$640000t^4 + 48000t^3 + 15300t^2 +$
$540t^1 + 81t^0$.

Carrying:
$640000t^4 + 48000t^3 + 15300t^2 +$
$540t^1 + 81t^0$;
$640000t^4 + 48000t^3 + 15300t^2 +$
$620t^1 + 1t^0$; ...
$700000t^5 + 0t^4 + 3000t^3 + 900t^2 +$
$20t^1 + 1t^0$.

What op



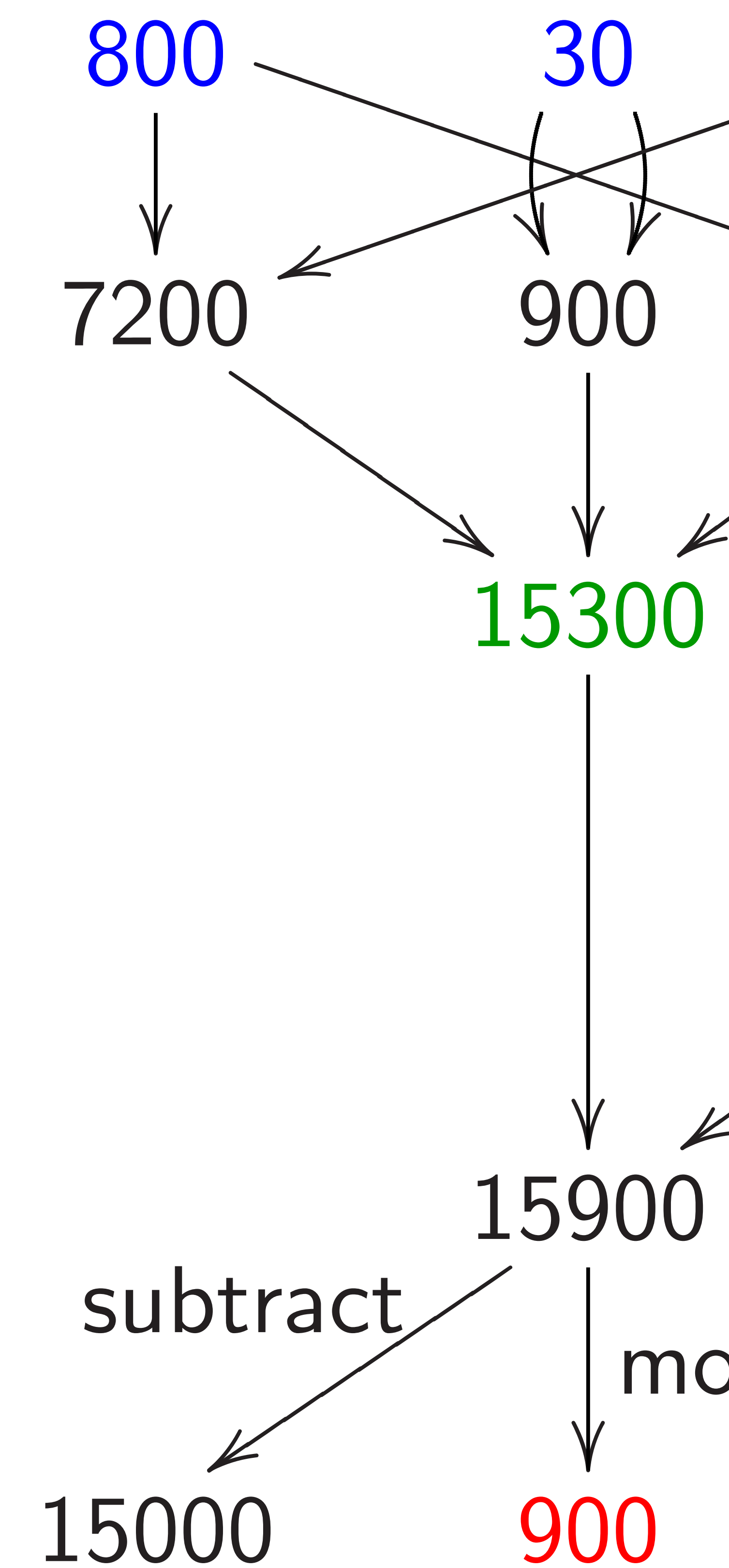
The scaled variation

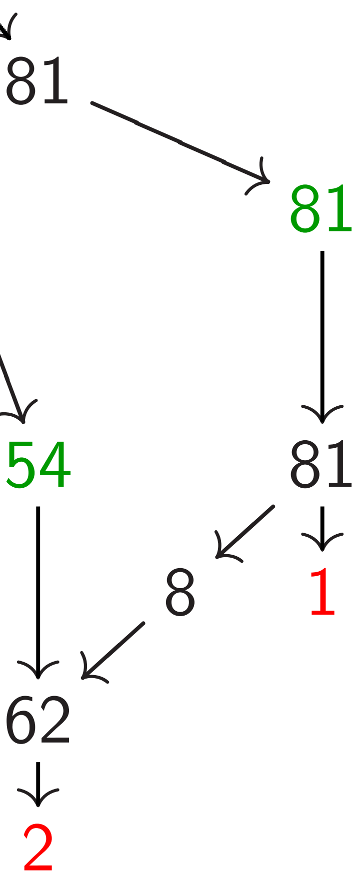
$839 = 800 + 30 + 9 =$
value (at $t = 1$) of polynomial
$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2 + 30t^1 + 9t^0)^2 =$
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$.

Carrying:
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$;
$640000t^4 + 48000t^3 + 15300t^2 + 620t^1 + 1t^0$; ...
$700000t^5 + 0t^4 + 3000t^3 + 900t^2 + 20t^1 + 1t^0$.

What operations w





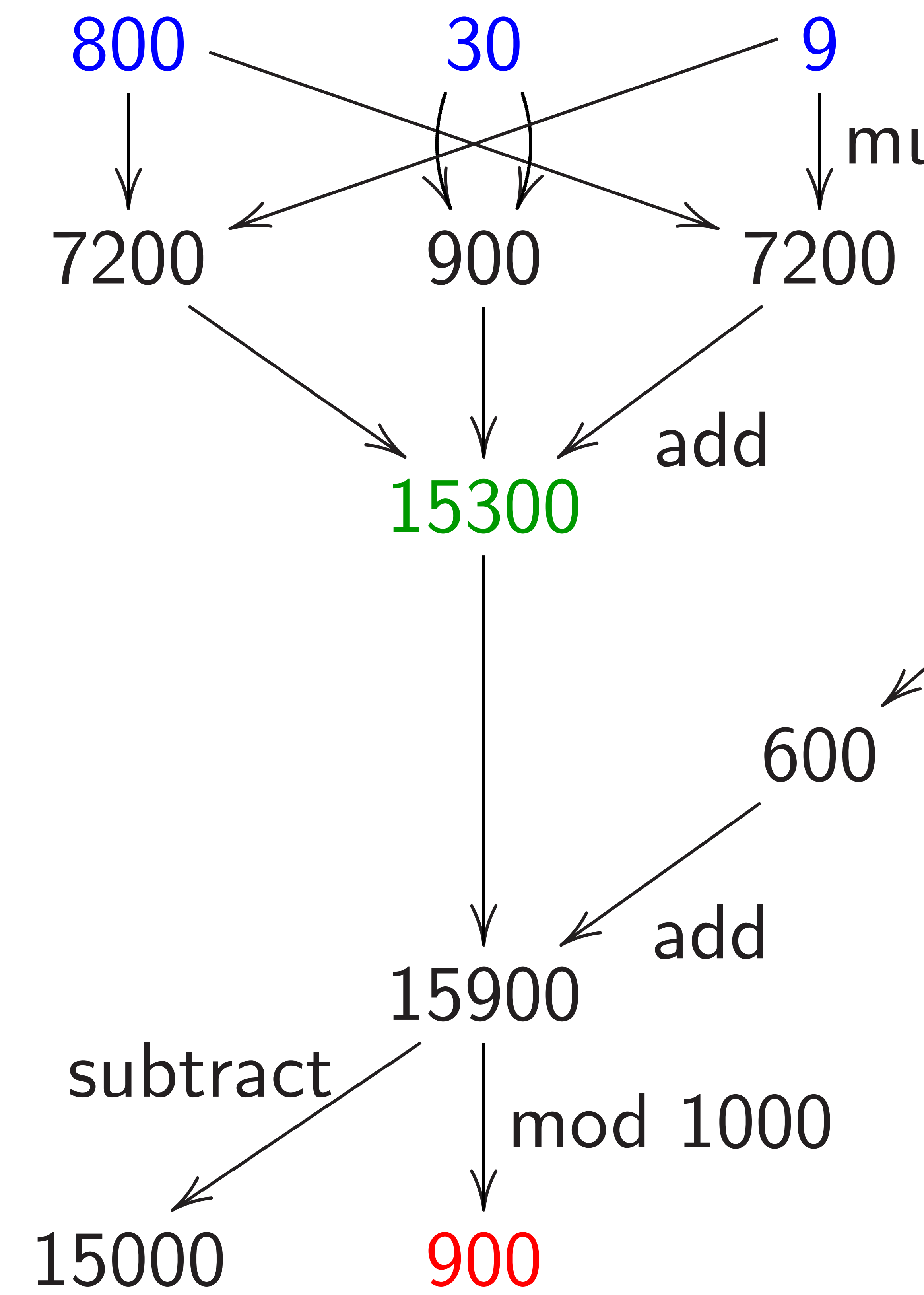
The scaled variation

$839 = 800 + 30 + 9 =$
value (at $t = 1$) of polynomial
$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2+30t^1+9t^0)^2 =$
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$.

Carrying:
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$;
$640000t^4 + 48000t^3 + 15300t^2 + 620t^1 + 1t^0$; . . .
$700000t^5 + 0t^4 + 3000t^3 + 900t^2 + 20t^1 + 1t^0$.

What operations were used



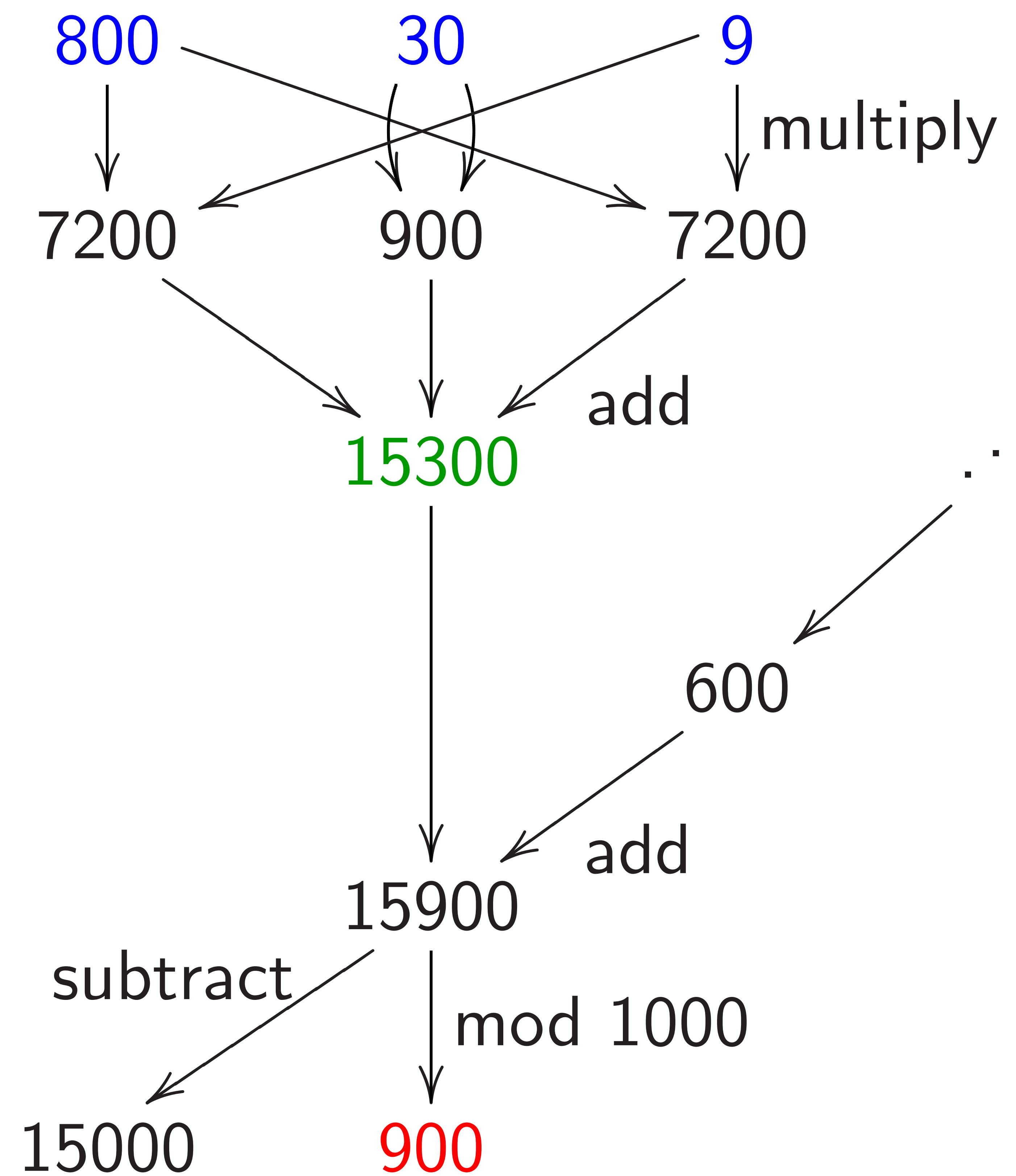
The scaled variation

$839 = 800 + 30 + 9 =$
value (at $t = 1$) of polynomial
$800t^2 + 30t^1 + 9t^0$.

Squaring: $(800t^2 + 30t^1 + 9t^0)^2 =$
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$.

Carrying:
$640000t^4 + 48000t^3 + 15300t^2 + 540t^1 + 81t^0$;
$640000t^4 + 48000t^3 + 15300t^2 + 620t^1 + 1t^0$; ...
$700000t^5 + 0t^4 + 3000t^3 + 900t^2 + 20t^1 + 1t^0$.

What operations were used here?



led variation

$00 + 30 + 9 =$

t $t = 1$) of polynomial

$30t^1 + 9t^0$.

g: $(800t^2+30t^1+9t^0)^2 =$

$^4 + 48000t^3 + 15300t^2 +$

$81t^0$.

:

$^4 + 48000t^3 + 15300t^2 +$

$81t^0$;

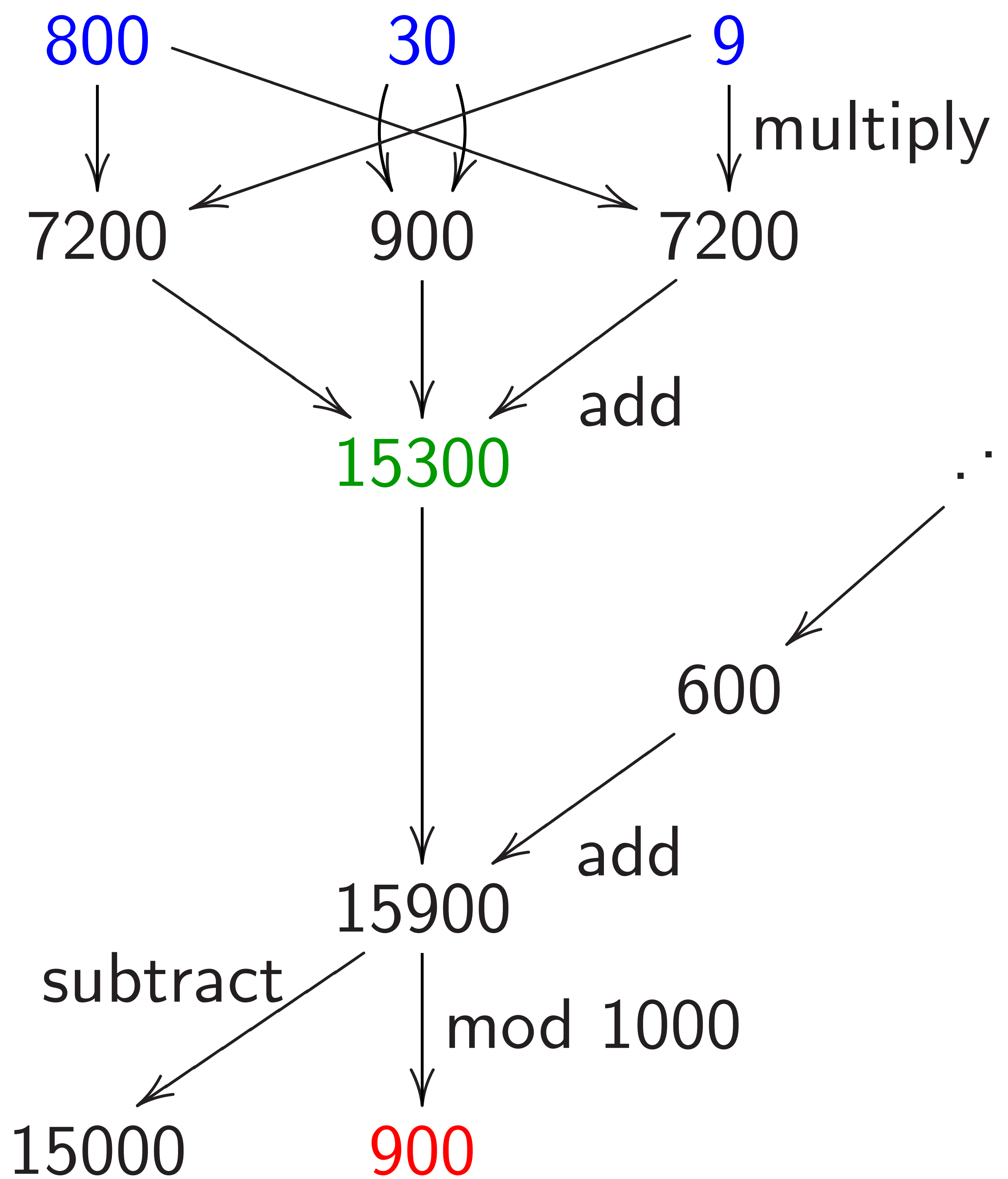
$^4 + 48000t^3 + 15300t^2 +$

$1t^0$; . . .

$^5 + 0t^4 + 3000t^3 + 900t^2 +$

t^0.

What operations were used here?



Speedup

$(\cdots + f_2$

has coef

$f_4 f_0 + f_3$

5 mults,

on

9 =

f polynomial

0.

$+30t^1+9t^0)^2 =$

$t^3 + 15300t^2 +$

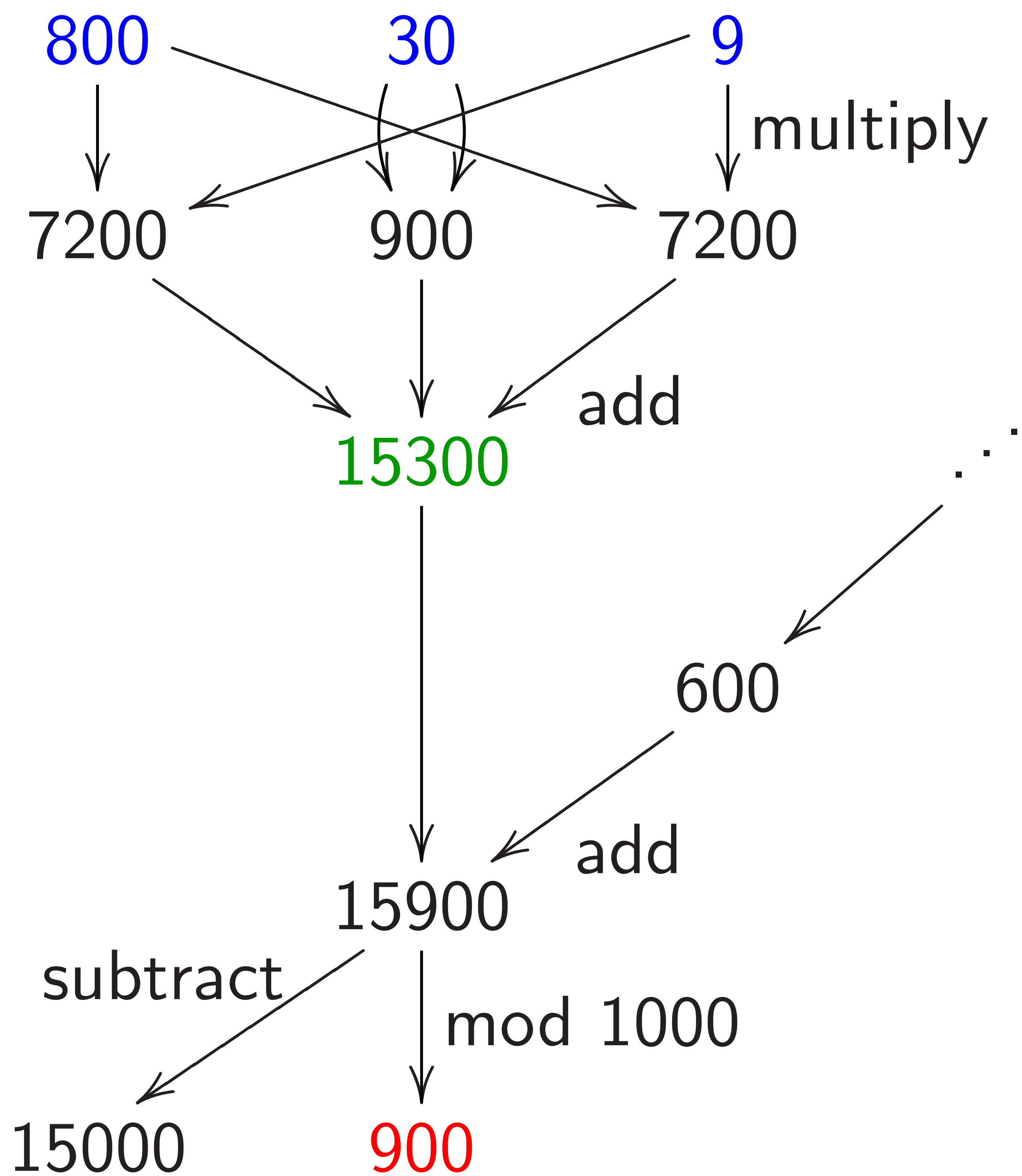
$t^3 + 15300t^2 +$

$t^3 + 15300t^2 +$

. .

$3000t^3+900t^2+$

What operations were used here?



Speedup: double i

$(\cdots + f_2 t^2 + f_1 t^1$

has coefficients su

$f_4 f_0 + f_3 f_1 + f_2 f_2 +$

5 mults, 4 adds.

ial

$^{0})^2 =$

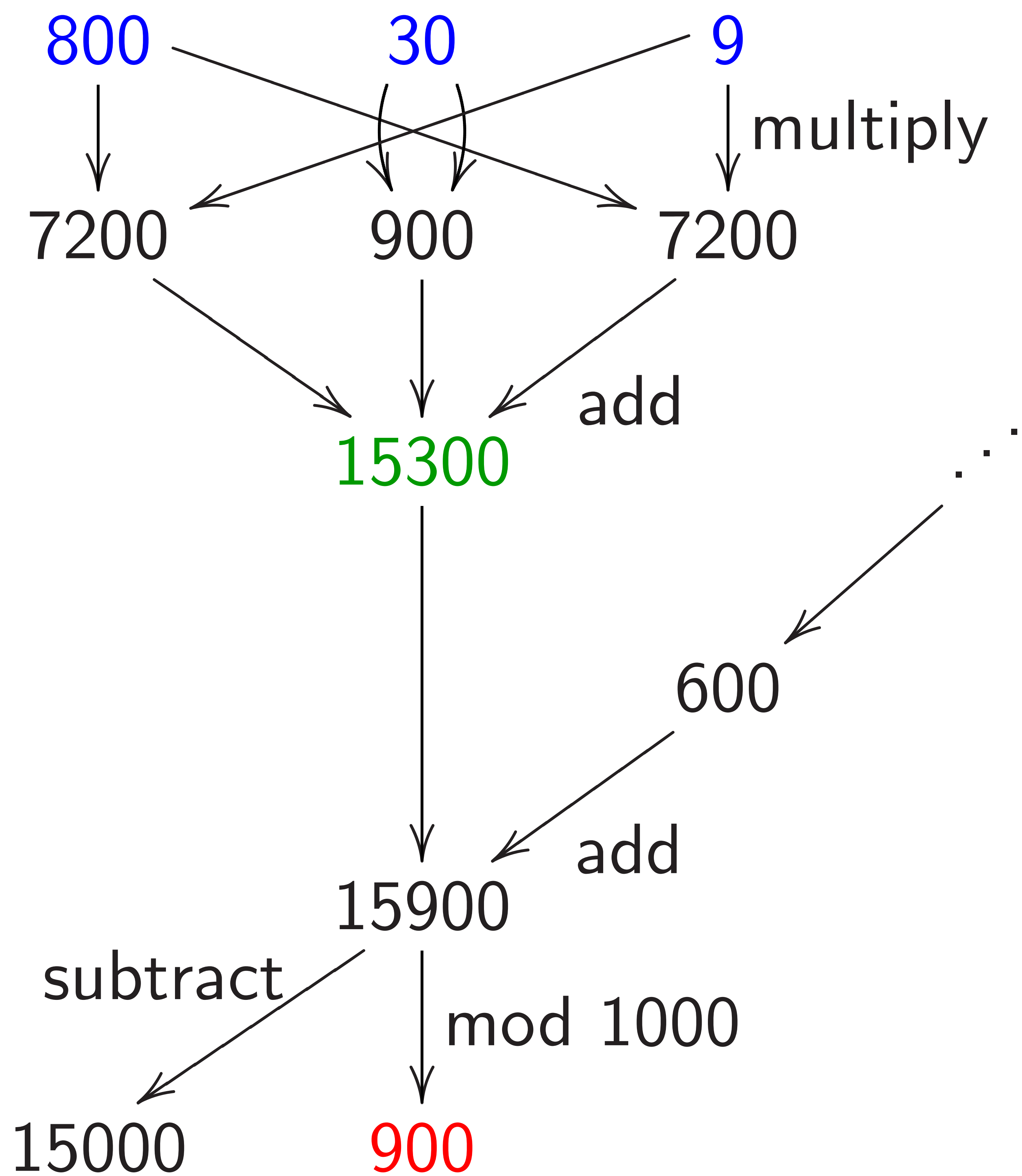
$0t^2 +$

$0t^2 +$

$0t^2 +$

$00t^2 +$

What operations were used here?



Speedup: double inside squa

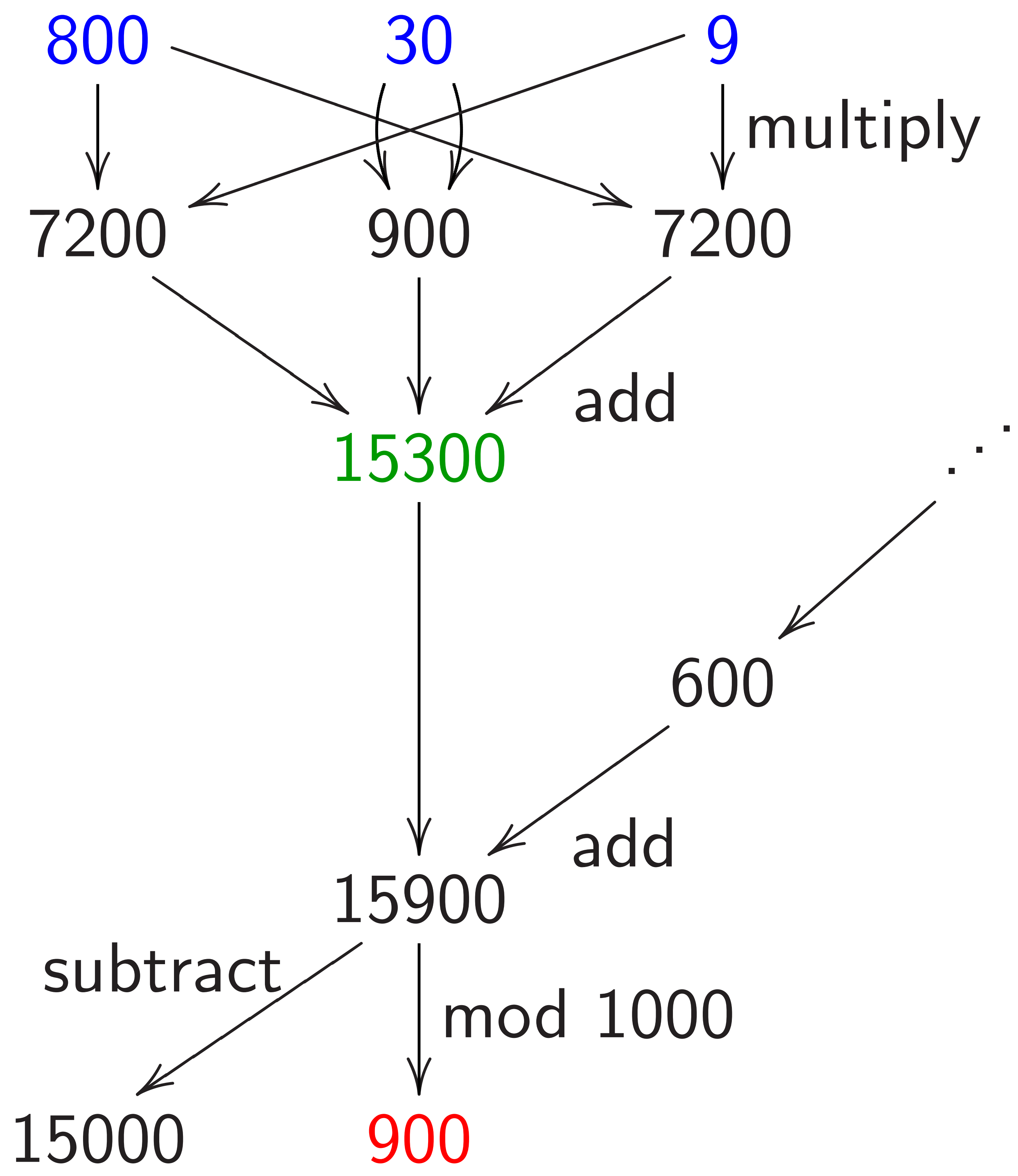
$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$

has coefficients such as

$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0$

5 mults, 4 adds.

What operations were used here?



Speedup: double inside squaring

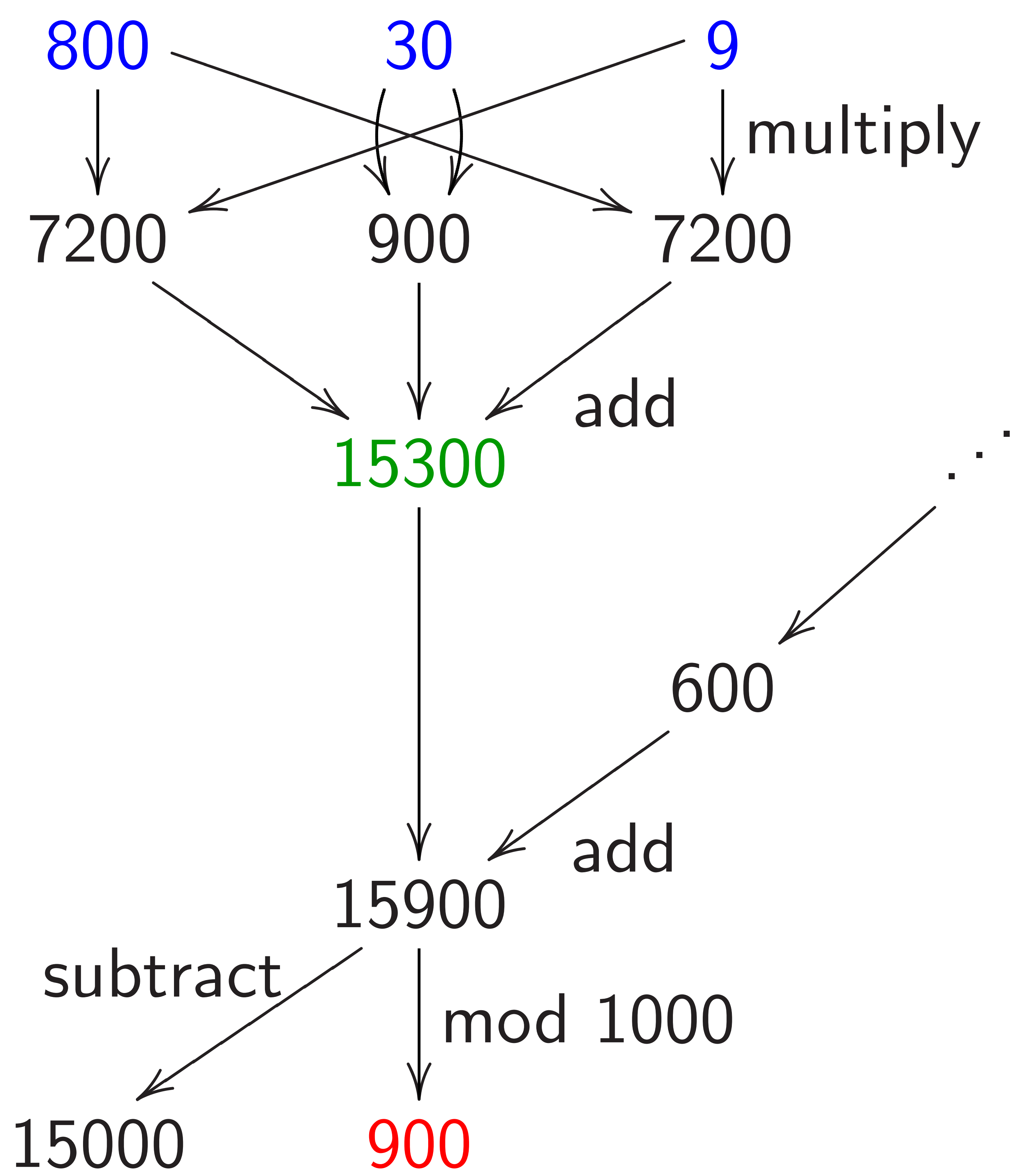
$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$

has coefficients such as

$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.

5 mults, 4 adds.

What operations were used here?



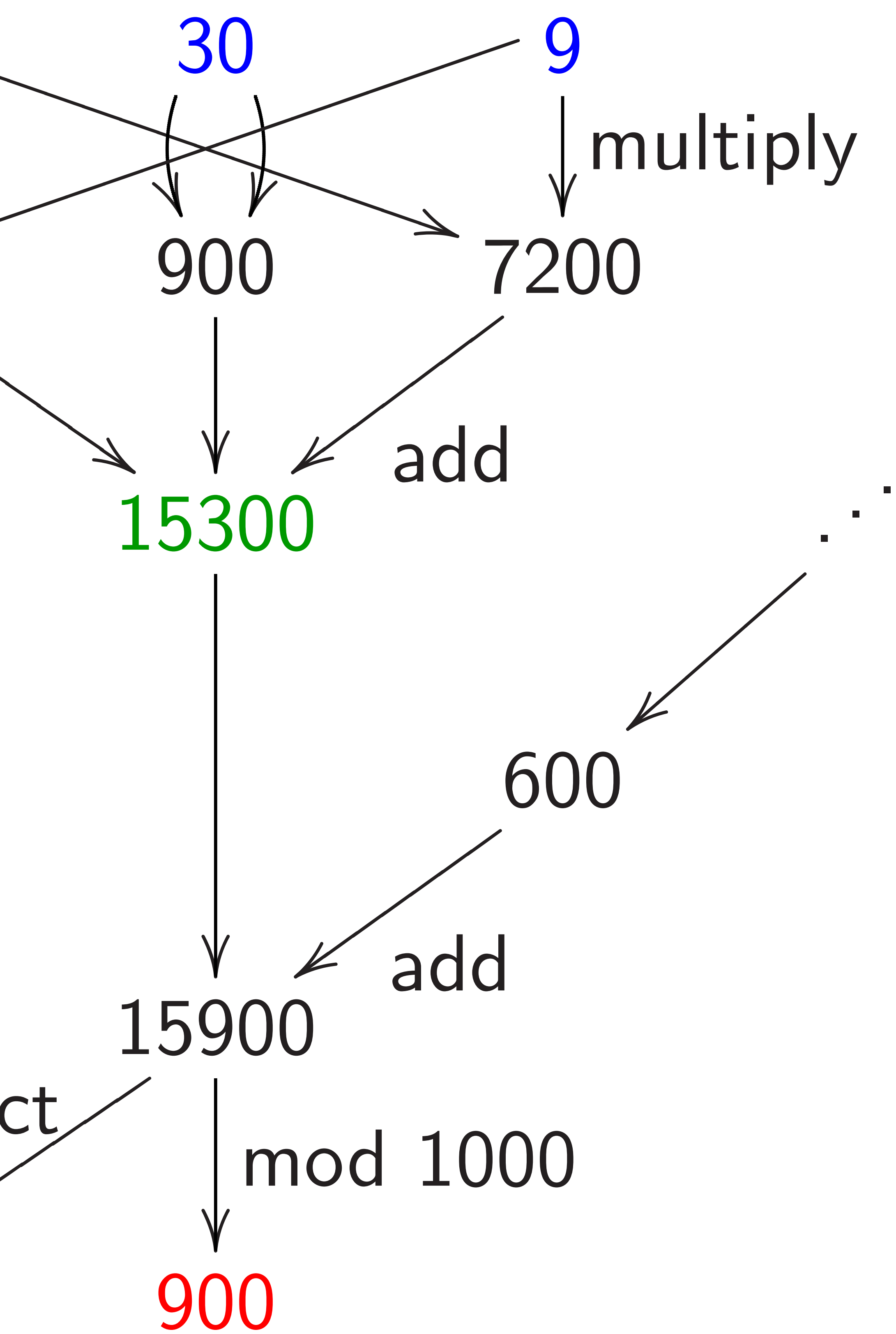
Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$
has coefficients such as
$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.
5 mults, 4 adds.

Compute more efficiently as
$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.
3 mults, 2 adds, 2 doublings.

Save $\approx 1/2$ of the mults
if there are many coefficients.

...erations were used here?



Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$
has coefficients such as
$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.
5 mults, 4 adds.

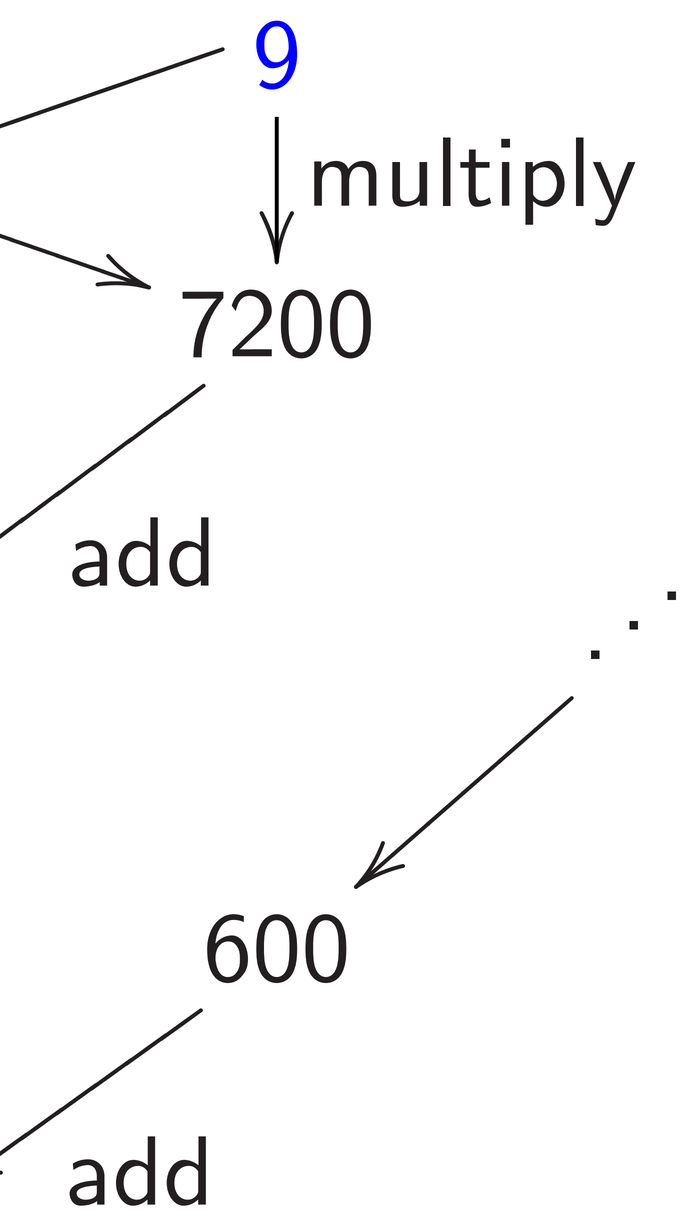
Compute more efficiently as
$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.
3 mults, 2 adds, 2 doublings.

Save $\approx 1/2$ of the mults
if there are many coefficients.

Faster a...
$2(f_4 f_0 + \ldots$
3 mults, ...

Save $\approx$ ...
if there a...

were used here?



od 1000

Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$
has coefficients such as
$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.
5 mults, 4 adds.

Compute more efficiently as
$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.
3 mults, 2 adds, 2 doublings.

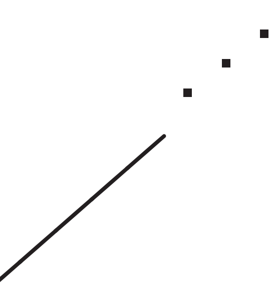
Save $\approx 1/2$ of the mults
if there are many coefficients.

Faster alternative:
$2(f_4 f_0 + f_3 f_1) + f_2$
3 mults, 2 adds, 1

Save $\approx 1/2$ of the
if there are many

here?

ultiply



Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$
has coefficients such as
$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.
5 mults, 4 adds.

Compute more efficiently as
$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.
3 mults, 2 adds, 2 doublings.

Save $\approx 1/2$ of the mults
if there are many coefficients.

Faster alternative:
$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.
3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds
if there are many coefficient

Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$

has coefficients such as

$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.

5 mults, 4 adds.

Compute more efficiently as

$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.

3 mults, 2 adds, 2 doublings.

Save $\approx 1/2$ of the mults

if there are many coefficients.

Faster alternative:

$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.

3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds

if there are many coefficients.

Speedup: double inside squaring

$(\cdots + f_2 t^2 + f_1 t^1 + f_0 t^0)^2$
has coefficients such as
$f_4 f_0 + f_3 f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.
5 mults, 4 adds.

Compute more efficiently as
$2 f_4 f_0 + 2 f_3 f_1 + f_2 f_2$.
3 mults, 2 adds, 2 doublings.

Save $\approx 1/2$ of the mults
if there are many coefficients.

Faster alternative:
$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.
3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds
if there are many coefficients.

Even faster alternative:
$(2 f_0) f_4 + (2 f_1) f_3 + f_2 f_2$,
after precomputing $2 f_0, 2 f_1, \ldots$.

3 mults, 2 adds, 0 doublings.
Precomputation ≈ 0.5 doublings.

p: double inside squaring

$_2 t^2 + f_1 t^1 + f_0 t^0)^2$

fficients such as

$f_1 + f_2 f_2 + f_1 f_3 + f_0 f_4$.

4 adds.

e more efficiently as

$2 f_3 f_1 + f_2 f_2$.

2 adds, 2 doublings.

$1/2$ of the mults

are many coefficients.

Faster alternative:

$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.

3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds

if there are many coefficients.

Even faster alternative:

$(2f_0)f_4 + (2f_1)f_3 + f_2 f_2$,

after precomputing $2f_0, 2f_1, \ldots$.

3 mults, 2 adds, 0 doublings.

Precomputation ≈ 0.5 doublings.

Speedup

Recall 1

Scaled:

Alternat

Scaled:

Use digi

instead

Small di

Several

easily ha

easily ha

reduce p

nside squaring

$+ f_0 t^0)^2$

ch as

$+ f_1 f_3 + f_0 f_4$.

iciently as

$_2$.

doublings.

mults

coefficients.

Faster alternative:

$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.

3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds if there are many coefficients.

Even faster alternative:

$(2f_0)f_4 + (2f_1)f_3 + f_2 f_2$,

after precomputing $2f_0, 2f_1, \ldots$.

3 mults, 2 adds, 0 doublings.

Precomputation ≈ 0.5 doublings.

Speedup: allow ne

Recall $159 \mapsto 15, 9$

Scaled: $15900 \mapsto$

Alternative: $159 \mapsto$

Scaled: $15900 \mapsto$

Use digits $\{-5, -4$

instead of $\{0, 1, \ldots$

Small disadvantag

Several small adva

easily handle nega

easily handle subtr

reduce products a

aring

f_4.

s.

s.

Faster alternative:

$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.

3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds
if there are many coefficients.

Even faster alternative:

$(2f_0)f_4 + (2f_1)f_3 + f_2 f_2$,
after precomputing $2f_0, 2f_1, \ldots$.

3 mults, 2 adds, 0 doublings.
Precomputation ≈ 0.5 doublings.

Speedup: allow negative coe

Recall $159 \mapsto 15, 9$.
Scaled: $15900 \mapsto 15000, 900$

Alternative: $159 \mapsto 16, -1$.
Scaled: $15900 \mapsto 16000, -1$

Use digits $\{-5, -4, \ldots, 4, 5\}$
instead of $\{0, 1, \ldots, 9\}$.
Small disadvantage: need $-$
Several small advantages:
easily handle negative intege
easily handle subtraction;
reduce products a bit.

Faster alternative:
$2(f_4 f_0 + f_3 f_1) + f_2 f_2$.
3 mults, 2 adds, 1 doubling.

Save $\approx 1/2$ of the adds
if there are many coefficients.

Even faster alternative:
$(2f_0)f_4 + (2f_1)f_3 + f_2 f_2$,
after precomputing $2f_0, 2f_1, \ldots$.

3 mults, 2 adds, 0 doublings.
Precomputation ≈ 0.5 doublings.

Speedup: allow negative coeffs

Recall $159 \mapsto 15, 9$.
Scaled: $15900 \mapsto 15000, 900$.

Alternative: $159 \mapsto 16, -1$.
Scaled: $15900 \mapsto 16000, -100$.

Use digits $\{-5, -4, \ldots, 4, 5\}$
instead of $\{0, 1, \ldots, 9\}$.
Small disadvantage: need $-$.
Several small advantages:
easily handle negative integers;
easily handle subtraction;
reduce products a bit.

lternative:

$f_3 f_1) + f_2 f_2$.

2 adds, 1 doubling.

1/2 of the adds

are many coefficients.

ster alternative:

$- (2f_1)f_3 + f_2 f_2$,

ecomputing $2f_0, 2f_1, \ldots$.

2 adds, 0 doublings.

putation ≈ 0.5 doublings.

Speedup: allow negative coeffs

Recall $159 \mapsto 15, 9$.

Scaled: $15900 \mapsto 15000, 900$.

Alternative: $159 \mapsto 16, -1$.

Scaled: $15900 \mapsto 16000, -100$.

Use digits $\{-5, -4, \ldots, 4, 5\}$

instead of $\{0, 1, \ldots, 9\}$.

Small disadvantage: need $-$.

Several small advantages:

easily handle negative integers;

easily handle subtraction;

reduce products a bit.

Speedup

Computi

multiply

square c

e.g. $a =$

$(3t^2+1t$

$6t^4 + 23$

carry: 8

As befor

$64t^4 + 4$

$7t^5 + 0t$

$+$: $7t^5+$

$7t^5 + 8t$

f_2.
doubling.

adds
coefficients.

ative:
$f_2 f_2$,
g $2f_0, 2f_1, \ldots$.

doublings.
0.5 doublings.

Speedup: allow negative coeffs

Recall $159 \mapsto 15, 9$.
Scaled: $15900 \mapsto 15000, 900$.

Alternative: $159 \mapsto 16, -1$.
Scaled: $15900 \mapsto 16000, -100$.

Use digits $\{-5, -4, \ldots, 4, 5\}$
instead of $\{0, 1, \ldots, 9\}$.
Small disadvantage: need $-$.
Several small advantages:
easily handle negative integers;
easily handle subtraction;
reduce products a bit.

Speedup: delay ca

Computing (e.g.)
multiply a, b polyn
square c poly, carr

e.g. $a = 314$, $b =$
$(3t^2+1t^1+4t^0)(2$
$6t^4 + 23t^3 + 18t^2$
carry: $8t^4 + 5t^3 +$

As before $(8t^2 + 3$
$64t^4 + 48t^3 + 153t$
$7t^5 + 0t^4 + 3t^3 +$

$+$: $7t^5+8t^4+8t^3$
$7t^5 + 8t^4 + 9t^3 +$

s.

. . . .

s.

lings.

Speedup: allow negative coeffs

Recall $159 \mapsto 15, 9$.

Scaled: $15900 \mapsto 15000, 900$.

Alternative: $159 \mapsto 16, -1$.

Scaled: $15900 \mapsto 16000, -100$.

Use digits $\{-5, -4, \ldots, 4, 5\}$ instead of $\{0, 1, \ldots, 9\}$.

Small disadvantage: need $-$.

Several small advantages: easily handle negative integers; easily handle subtraction; reduce products a bit.

Speedup: delay carries

Computing (e.g.) big $ab + c$

multiply a, b polynomials, ca

square c poly, carry, add, ca

e.g. $a = 314$, $b = 271$, $c =$

$(3t^2+1t^1+4t^0)(2t^2+7t^1+1$

$6t^4 + 23t^3 + 18t^2 + 29t^1 +$

carry: $8t^4 + 5t^3 + 0t^2 + 9t^1$

As before $(8t^2 + 3t^1 + 9t^0)$

$64t^4 + 48t^3 + 153t^2 + 54t^1$

$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1$

$+$: $7t^5+8t^4+8t^3+9t^2+11t$

$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1$

Speedup: allow negative coeffs

Recall $159 \mapsto 15, 9$.

Scaled: $15900 \mapsto 15000, 900$.

Alternative: $159 \mapsto 16, -1$.

Scaled: $15900 \mapsto 16000, -100$.

Use digits $\{-5, -4, \ldots, 4, 5\}$
instead of $\{0, 1, \ldots, 9\}$.
Small disadvantage: need $-$.
Several small advantages:
easily handle negative integers;
easily handle subtraction;
reduce products a bit.

Speedup: delay carries

Computing (e.g.) big $ab + c^2$:
multiply a, b polynomials, carry,
square c poly, carry, add, carry.

e.g. $a = 314$, $b = 271$, $c = 839$:
$(3t^2+1t^1+4t^0)(2t^2+7t^1+1t^0) =$
$6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0$;
carry: $8t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

As before $(8t^2 + 3t^1 + 9t^0)^2 =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

$+$: $7t^5+8t^4+8t^3+9t^2+11t^1+5t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

: allow negative coeffs

59 $\mapsto$ 15, 9.

15900 $\mapsto$ 15000, 900.

ive: 159 $\mapsto$ 16, -1.

15900 $\mapsto$ 16000, -100.

ts $\{-5, -4, \ldots, 4, 5\}$

of $\{0, 1, \ldots, 9\}$.

sadvantage: need $-$.

small advantages:

ndle negative integers;

ndle subtraction;

roducts a bit.

Speedup: delay carries

Computing (e.g.) big $ab + c^2$:
multiply a, b polynomials, carry,
square c poly, carry, add, carry.

e.g. $a = 314$, $b = 271$, $c = 839$:
$(3t^2+1t^1+4t^0)(2t^2+7t^1+1t^0) =$
$6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0$;
carry: $8t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

As before $(8t^2 + 3t^1 + 9t^0)^2 =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

$+$: $7t^5+8t^4+8t^3+9t^2+11t^1+5t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Faster: r

square c

$(6t^4 + 23$

$(64t^4 + 4$

$= 70t^4 +$

$7t^5 + 8t$

Eliminat

Outweig

slightly l

Importar

multiplic

to reduc

but carri

before a

egative coeffs

9.

15000, 900.

→ 16, −1.

16000, −100.

4, . . . , 4, 5}

. . , 9}.

e: need −.

ntages:

tive integers;

raction;

bit.

Speedup: delay carries

Computing (e.g.) big $ab + c^2$: multiply a, b polynomials, carry, square c poly, carry, add, carry.

e.g. $a = 314$, $b = 271$, $c = 839$:
$(3t^2+1t^1+4t^0)(2t^2+7t^1+1t^0) =$
$6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0$;
carry: $8t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

As before $(8t^2 + 3t^1 + 9t^0)^2 =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

+: $7t^5+8t^4+8t^3+9t^2+11t^1+5t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Faster: multiply a

square c polynomi

(6t^4 + 23t^3 + 18t^2

(64t^4+48t^3+153

= 70t^4+71t^3+171

7t^5 + 8t^4 + 9t^3 +

Eliminate intermed

Outweighs cost of

slightly larger coef

Important to carry

multiplications (ar

to reduce coefficie

but carries are usu

before additions, s

effs

0.

00.

}

.

ers;

Speedup: delay carries

Computing (e.g.) big $ab + c^2$: multiply a, b polynomials, carry, square c poly, carry, add, carry.

e.g. $a = 314$, $b = 271$, $c = 839$:
$(3t^2+1t^1+4t^0)(2t^2+7t^1+1t^0) =$
$6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0$;
carry: $8t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

As before $(8t^2 + 3t^1 + 9t^0)^2 =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

+: $7t^5+8t^4+8t^3+9t^2+11t^1+5t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Faster: multiply a, b polynon
square c polynomial, add, ca

$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4$
$(64t^4 + 48t^3 + 153t^2 + 54t^1 +$
$= 70t^4 + 71t^3 + 171t^2 + 83t^1 +$
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1$

Eliminate intermediate carrie
Outweighs cost of handling
slightly larger coefficients.

Important to carry between
multiplications (and squarin
to reduce coefficient size;
but carries are usually a bad
before additions, subtraction

Speedup: delay carries

Computing (e.g.) big $ab + c^2$: multiply a, b polynomials, carry, square c poly, carry, add, carry.

e.g. $a = 314$, $b = 271$, $c = 839$:
$(3t^2+1t^1+4t^0)(2t^2+7t^1+1t^0) =$
$6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0$;
carry: $8t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

As before $(8t^2 + 3t^1 + 9t^0)^2 =$
$64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0$;
$7t^5 + 0t^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

$+$: $7t^5+8t^4+8t^3+9t^2+11t^1+5t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Faster: multiply a, b polynomials, square c polynomial, add, carry.

$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0) +$
$(64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0)$
$= 70t^4+71t^3+171t^2+83t^1+85t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Eliminate intermediate carries. Outweighs cost of handling slightly larger coefficients.

Important to carry between multiplications (and squarings) to reduce coefficient size; but carries are usually a bad idea before additions, subtractions, etc.

o: delay carries

ing (e.g.) big $ab + c^2$:

a, b polynomials, carry,

r poly, carry, add, carry.

= 314, $b = 271$, $c = 839$:

$^1 + 4t^0)(2t^2 + 7t^1 + 1t^0) =$

$3t^3 + 18t^2 + 29t^1 + 4t^0$;

$t^4 + 5t^3 + 0t^2 + 9t^1 + 4t^0$.

re $(8t^2 + 3t^1 + 9t^0)^2 =$

$8t^3 + 153t^2 + 54t^1 + 81t^0$;

$^4 + 3t^3 + 9t^2 + 2t^1 + 1t^0$.

$8t^4 + 8t^3 + 9t^2 + 11t^1 + 5t^0$;

$^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Faster: multiply a, b polynomials, square c polynomial, add, carry.

$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0) +$
$(64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0)$
$= 70t^4 + 71t^3 + 171t^2 + 83t^1 + 85t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Eliminate intermediate carries. Outweighs cost of handling slightly larger coefficients.

Important to carry between multiplications (and squarings) to reduce coefficient size; but carries are usually a bad idea before additions, subtractions, etc.

Speedup

How mu

$f = f_0 +$

$g = g_0$

Using th

400 coef

Faster:

$F_0 = f_0$

$F_1 = f_{10}$

Similarly

Then $f g$

$+ (F_0 G_0$

rries

big $ab + c^2$:

nomials, carry,

ry, add, carry.

271, $c = 839$:

$t^2 + 7t^1 + 1t^0) =$

$+ 29t^1 + 4t^0;$

$0t^2 + 9t^1 + 4t^0.$

$3t^1 + 9t^0)^2 =$

$t^2 + 54t^1 + 81t^0;$

$9t^2 + 2t^1 + 1t^0.$

$+9t^2 + 11t^1 + 5t^0;$

$0t^2 + 1t^1 + 5t^0.$

Faster: multiply a, b polynomials, square c polynomial, add, carry.

$$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0) + (64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0) = 70t^4 + 71t^3 + 171t^2 + 83t^1 + 85t^0;$$

$$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0.$$

Eliminate intermediate carries. Outweighs cost of handling slightly larger coefficients.

Important to carry between multiplications (and squarings) to reduce coefficient size; but carries are usually a bad idea before additions, subtractions, etc.

Speedup: polynom

How much work t

$f = f_0 + f_1 t + \cdots$

$g = g_0 + g_1 t + \cdots$

Using the obvious

400 coeff mults, 3

Faster: Write f as

$F_0 = f_0 + f_1 t + \cdots$

$F_1 = f_{10} + f_{11} t +$

Similarly write g a

Then $fg = (F_0 +$

$+ (F_0 G_0 - F_1 G_1 t^1$

c^2:

arry,

rry.

839:

$t^0) =$

$4t^0$;

$+ 4t^0$.

$^2 =$

$+81t^0$;

$+ 1t^0$.

$^1 + 5t^0$;

$+ 5t^0$.

Faster: multiply a, b polynomials, square c polynomial, add, carry.

$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0) +$
$(64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0)$
$= 70t^4 + 71t^3 + 171t^2 + 83t^1 + 85t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Eliminate intermediate carries. Outweighs cost of handling slightly larger coefficients.

Important to carry between multiplications (and squarings) to reduce coefficient size; but carries are usually a bad idea before additions, subtractions, etc.

Speedup: polynomial Karats

How much work to multiply
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$

Using the obvious method:
400 coeff mults, 361 coeff a

Faster: Write f as $F_0 + F_1 t$
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$
Similarly write g as $G_0 + G_1$

Then $fg = (F_0 + F_1)(G_0 +$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10}$

Faster: multiply a, b polynomials, square c polynomial, add, carry.

$(6t^4 + 23t^3 + 18t^2 + 29t^1 + 4t^0) +$
$(64t^4 + 48t^3 + 153t^2 + 54t^1 + 81t^0)$
$= 70t^4 + 71t^3 + 171t^2 + 83t^1 + 85t^0$;
$7t^5 + 8t^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0$.

Eliminate intermediate carries. Outweighs cost of handling slightly larger coefficients.

Important to carry between multiplications (and squarings) to reduce coefficient size; but carries are usually a bad idea before additions, subtractions, etc.

Speedup: polynomial Karatsuba

How much work to multiply polys
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$?

Using the obvious method: 400 coeff mults, 361 coeff adds.

Faster: Write f as $F_0 + F_1 t^{10}$;
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$.
Similarly write g as $G_0 + G_1 t^{10}$.

Then $fg = (F_0 + F_1)(G_0 + G_1) t^{10}$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10})$.

multiply a, b polynomials,
polynomial, add, carry.

$3t^3 + 18t^2 + 29t^1 + 4t^0) +$
$48t^3 + 153t^2 + 54t^1 + 81t^0)$
$71t^3 + 171t^2 + 83t^1 + 85t^0;$
$^4 + 9t^3 + 0t^2 + 1t^1 + 5t^0.$

e intermediate carries.
hs cost of handling
arger coefficients.

nt to carry between
cations (and squarings)
e coefficient size;
ies are usually a bad idea
dditions, subtractions, etc.

Speedup: polynomial Karatsuba

How much work to multiply polys
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$?

Using the obvious method:
400 coeff mults, 361 coeff adds.

Faster: Write f as $F_0 + F_1 t^{10}$;
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$.
Similarly write g as $G_0 + G_1 t^{10}$.

Then $fg = (F_0 + F_1)(G_0 + G_1)t^{10}$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10})$.

20 adds
300 mul
$F_0 G_0$, F
243 adds
9 adds f
with sub
and with
19 adds
19 adds

Total 30
Larger c
still save

Can app
as poly

, b polynomials,
al, add, carry.

$+ 29t^1 + 4t^0) +$
$t^2 + 54t^1 + 81t^0)$
$1t^2 + 83t^1 + 85t^0;$
$0t^2 + 1t^1 + 5t^0.$

diate carries.
handling
fficients.

between
nd squarings)
nt size;
ually a bad idea
subtractions, etc.

Speedup: polynomial Karatsuba

How much work to multiply polys
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$?

Using the obvious method:
400 coeff mults, 361 coeff adds.

Faster: Write f as $F_0 + F_1 t^{10}$;
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$.
Similarly write g as $G_0 + G_1 t^{10}$.

Then $fg = (F_0 + F_1)(G_0 + G_1) t^{10}$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10})$.

20 adds for $F_0 + F$
300 mults for thre
$F_0 G_0$, $F_1 G_1$, $(F_0 +$
243 adds for those
9 adds for $F_0 G_0 -$
with subs counted
and with delayed r
19 adds for $\cdots(1$
19 adds to finish.

Total 300 mults, 3
Larger coefficients
still saves time.

Can apply idea rec
as poly degree gro

mials,
arry.
$4t^0)+$
$+81t^0)$
$+85t^0;$
$+5t^0.$
es.
gs)
idea
ns, etc.

Speedup: polynomial Karatsuba

How much work to multiply polys
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$?

Using the obvious method:
400 coeff mults, 361 coeff adds.

Faster: Write f as $F_0 + F_1 t^{10}$;
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$.
Similarly write g as $G_0 + G_1 t^{10}$.

Then $fg = (F_0 + F_1)(G_0 + G_1) t^{10}$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10})$.

20 adds for $F_0 + F_1$, $G_0 + G$
300 mults for three products
$F_0 G_0$, $F_1 G_1$, $(F_0 + F_1)(G_0 +$
243 adds for those products
9 adds for $F_0 G_0 - F_1 G_1 t^{10}$
with subs counted as adds
and with delayed negations.
19 adds for $\cdots (1 - t^{10})$.
19 adds to finish.

Total 300 mults, 310 adds.
Larger coefficients, slight ex
still saves time.

Can apply idea recursively
as poly degree grows.

Speedup: polynomial Karatsuba

How much work to multiply polys
$f = f_0 + f_1 t + \cdots + f_{19} t^{19}$,
$g = g_0 + g_1 t + \cdots + g_{19} t^{19}$?

Using the obvious method:
400 coeff mults, 361 coeff adds.

Faster: Write f as $F_0 + F_1 t^{10}$;
$F_0 = f_0 + f_1 t + \cdots + f_9 t^9$;
$F_1 = f_{10} + f_{11} t + \cdots + f_{19} t^9$.
Similarly write g as $G_0 + G_1 t^{10}$.

Then $fg = (F_0 + F_1)(G_0 + G_1)t^{10}$
$+ (F_0 G_0 - F_1 G_1 t^{10})(1 - t^{10})$.

20 adds for $F_0 + F_1$, $G_0 + G_1$.
300 mults for three products
$F_0 G_0$, $F_1 G_1$, $(F_0 + F_1)(G_0 + G_1)$.
243 adds for those products.
9 adds for $F_0 G_0 - F_1 G_1 t^{10}$
with subs counted as adds
and with delayed negations.
19 adds for $\cdots(1 - t^{10})$.
19 adds to finish.

Total 300 mults, 310 adds.
Larger coefficients, slight expense;
still saves time.

Can apply idea recursively
as poly degree grows.

o: polynomial Karatsuba

ch work to multiply polys

$+ f_1 t + \cdots + f_{19} t^{19}$,

$+ g_1 t + \cdots + g_{19} t^{19}$?

ne obvious method:

ff mults, 361 coeff adds.

Write f as $F_0 + F_1 t^{10}$;

$+ f_1 t + \cdots + f_9 t^9$;

$+ f_{11} t + \cdots + f_{19} t^9$.

write g as $G_0 + G_1 t^{10}$.

$= (F_0 + F_1)(G_0 + G_1) t^{10}$

$- F_1 G_1 t^{10})(1 - t^{10})$.

20 adds for $F_0 + F_1$, $G_0 + G_1$.

300 mults for three products

$F_0 G_0$, $F_1 G_1$, $(F_0 + F_1)(G_0 + G_1)$.

243 adds for those products.

9 adds for $F_0 G_0 - F_1 G_1 t^{10}$

with subs counted as adds

and with delayed negations.

19 adds for $\cdots (1 - t^{10})$.

19 adds to finish.

Total 300 mults, 310 adds.

Larger coefficients, slight expense;

still saves time.

Can apply idea recursively

as poly degree grows.

Many ot

in polyn

"Toom,"

Increasi

polynom

$O(n \lg n$

to comp

Useful f

that occ

In some

But Kar

for prim

on most

nial Karatsuba

o multiply polys
$+ f_{19}t^{19}$,
$\cdot + g_{19}t^{19}$?

method:
61 coeff adds.

s $F_0 + F_1 t^{10}$;
$\cdot + f_9 t^9$;
$\cdots + f_{19} t^9$.
ns $G_0 + G_1 t^{10}$.

$F_1)(G_0 + G_1)t^{10}$
$^{10})(1 - t^{10})$.

20 adds for $F_0 + F_1$, $G_0 + G_1$.
300 mults for three products
$F_0 G_0$, $F_1 G_1$, $(F_0 + F_1)(G_0 + G_1)$.
243 adds for those products.
9 adds for $F_0 G_0 - F_1 G_1 t^{10}$
with subs counted as adds
and with delayed negations.
19 adds for $\cdots (1 - t^{10})$.
19 adds to finish.

Total 300 mults, 310 adds.
Larger coefficients, slight expense;
still saves time.

Can apply idea recursively
as poly degree grows.

Many other algebr
in polynomial mul
"Toom," "FFT," e

Increasingly impor
polynomial degree
$O(n \lg n \lg \lg n)$ co
to compute n-coef

Useful for sizes of
that occur in cryp
In some cases, yes
But Karatsuba is t
for prime-field EC
on most current C

suba

polys

?

dds.

^{10};

^{9}.

t^{10}.

$G_1)t^{10}$

).

20 adds for $F_0 + F_1$, $G_0 + G_1$.
300 mults for three products
F_0G_0, F_1G_1, $(F_0 + F_1)(G_0 + G_1)$.
243 adds for those products.
9 adds for $F_0G_0 - F_1G_1t^{10}$
with subs counted as adds
and with delayed negations.
19 adds for $\cdots(1 - t^{10})$.
19 adds to finish.

Total 300 mults, 310 adds.
Larger coefficients, slight expense;
still saves time.

Can apply idea recursively
as poly degree grows.

Many other algebraic speedu
in polynomial multiplication
"Toom," "FFT," etc.

Increasingly important as
polynomial degree grows.
$O(n \lg n \lg \lg n)$ coeff operati
to compute n-coeff product.

Useful for sizes of n
that occur in cryptography?
In some cases, yes!
But Karatsuba is the limit
for prime-field ECC/ECDLP
on most current CPUs.

20 adds for $F_0 + F_1$, $G_0 + G_1$.
300 mults for three products
F_0G_0, F_1G_1, $(F_0 + F_1)(G_0 + G_1)$.
243 adds for those products.
9 adds for $F_0G_0 - F_1G_1t^{10}$
with subs counted as adds
and with delayed negations.
19 adds for $\cdots(1 - t^{10})$.
19 adds to finish.

Total 300 mults, 310 adds.
Larger coefficients, slight expense;
still saves time.

Can apply idea recursively
as poly degree grows.

Many other algebraic speedups
in polynomial multiplication:
"Toom," "FFT," etc.

Increasingly important as
polynomial degree grows.
$O(n \lg n \lg \lg n)$ coeff operations
to compute n-coeff product.

Useful for sizes of n
that occur in cryptography?
In some cases, yes!
But Karatsuba is the limit
for prime-field ECC/ECDLP
on most current CPUs.

for $F_0 + F_1$, $G_0 + G_1$.
ts for three products
$_1G_1$, $(F_0 + F_1)(G_0 + G_1)$.
s for those products.
or $F_0G_0 - F_1G_1t^{10}$
os counted as adds
n delayed negations.
for $\cdots(1 - t^{10})$.
to finish.

00 mults, 310 adds.
oefficients, slight expense;
es time.

ly idea recursively
degree grows.

Many other algebraic speedups
in polynomial multiplication:
"Toom," "FFT," etc.

Increasingly important as
polynomial degree grows.
$O(n \lg n \lg \lg n)$ coeff operations
to compute n-coeff product.

Useful for sizes of n
that occur in cryptography?
In some cases, yes!
But Karatsuba is the limit
for prime-field ECC/ECDLP
on most current CPUs.

Modular

How to

Can use
f mod p
Can mul
precomp
easily ad

Slight sp
"Montgo

F_1, $G_0 + G_1$.
e products
$F_1)(G_0 + G_1)$.
e products.
$F_1 G_1 t^{10}$
as adds
negations.
$- t^{10})$.

310 adds.
, slight expense;

cursively
ows.

Many other algebraic speedups in polynomial multiplication: "Toom," "FFT," etc.

Increasingly important as polynomial degree grows. $O(n \lg n \lg \lg n)$ coeff operations to compute n-coeff product.

Useful for sizes of n that occur in cryptography? In some cases, yes! But Karatsuba is the limit for prime-field ECC/ECDLP on most current CPUs.

Modular reduction

How to compute f

Can use definition
$f \bmod p = f - p \lfloor$
Can multiply f by
precomputed $1/p$
easily adjust to ob

Slight speedup: "2
"Montgomery redu

G_1.

s

$+ G_1$).

.

pense;

Many other algebraic speedups in polynomial multiplication: "Toom," "FFT," etc.

Increasingly important as polynomial degree grows. $O(n \lg n \lg \lg n)$ coeff operations to compute n-coeff product.

Useful for sizes of n that occur in cryptography? In some cases, yes! But Karatsuba is the limit for prime-field ECC/ECDLP on most current CPUs.

Modular reduction

How to compute $f \bmod p$?

Can use definition: $f \bmod p = f - p \lfloor f/p \rfloor$. Can multiply f by a precomputed $1/p$ approxima easily adjust to obtain $\lfloor f/p$

Slight speedup: "2-adic inve "Montgomery reduction."

Many other algebraic speedups in polynomial multiplication: "Toom," "FFT," etc.

Increasingly important as polynomial degree grows. $O(n \lg n \lg \lg n)$ coeff operations to compute n-coeff product.

Useful for sizes of n that occur in cryptography? In some cases, yes! But Karatsuba is the limit for prime-field ECC/ECDLP on most current CPUs.

Modular reduction

How to compute $f \bmod p$?

Can use definition: $f \bmod p = f - p \lfloor f/p \rfloor$. Can multiply f by a precomputed $1/p$ approximation; easily adjust to obtain $\lfloor f/p \rfloor$.

Slight speedup: "2-adic inverse"; "Montgomery reduction."

ther algebraic speedups
omial multiplication:
" "FFT," etc.

ngly important as
nial degree grows.
lg lg n) coeff operations
ute n-coeff product.

or sizes of n
ur in cryptography?
cases, yes!
atsuba is the limit
e-field ECC/ECDLP
current CPUs.

Modular reduction

How to compute $f \bmod p$?

Can use definition:
$f \bmod p = f - p \lfloor f/p \rfloor$.
Can multiply f by a
precomputed $1/p$ approximation;
easily adjust to obtain $\lfloor f/p \rfloor$.

Slight speedup: "2-adic inverse";
"Montgomery reduction."

e.g. 314

Precomp
$\lfloor 100000$
$= 36787$

Comput
314159
$= 11557$

Comput
3141592
$= 57823$
Oops, to
578230
306402

raic speedups
tiplication:
etc.

tant as
grows.
eff operations
ff product.

n
tography?
!
the limit
C/ECDLP
PUs.

Modular reduction

How to compute f mod p?

Can use definition:
$f \bmod p = f - p\lfloor f/p \rfloor$.
Can multiply f by a
precomputed $1/p$ approximation;
easily adjust to obtain $\lfloor f/p \rfloor$.

Slight speedup: "2-adic inverse";
"Montgomery reduction."

e.g. 31415926535

Precompute
$\lfloor 1000000000000/$
$= 3678796$.

Compute
$314159 \cdot 3678796$
$= 1155726872564$

Compute
$31415926535 8 - 1$
$= 578230$.
Oops, too big:
$578230 - 271828$
$306402 - 271828$

ups

:

ons

Modular reduction

How to compute f mod p?

Can use definition:
$f \bmod p = f - p \lfloor f/p \rfloor$.
Can multiply f by a precomputed $1/p$ approximation; easily adjust to obtain $\lfloor f/p \rfloor$.

Slight speedup: "2-adic inverse"; "Montgomery reduction."

e.g. 314159265358 mod 271

Precompute
$\lfloor 1000000000000/271828 \rfloor$
$= 3678796$.

Compute
$314159 \cdot 3678796$
$= 1155726872564$.

Compute
$314159265358 - 1155726 \cdot 2$
$= 578230$.
Oops, too big:
$578230 - 271828 = 306402$.
$306402 - 271828 = 34574$.

Modular reduction

How to compute $f \bmod p$?

Can use definition:
$f \bmod p = f - p \lfloor f/p \rfloor$.
Can multiply f by a precomputed $1/p$ approximation; easily adjust to obtain $\lfloor f/p \rfloor$.

Slight speedup: "2-adic inverse"; "Montgomery reduction."

e.g. 314159265358 mod 271828:

Precompute
$\lfloor 1000000000000/271828 \rfloor$
$= 3678796$.

Compute
$314159 \cdot 3678796$
$= 1155726872564$.

Compute
$314159265358 - 1155726 \cdot 271828$
$= 578230$.
Oops, too big:
$578230 - 271828 = 306402$.
$306402 - 271828 = 34574$.

r reduction

compute f mod p?

definition:

$\flat = f - p\lfloor f/p \rfloor$.

ltiply f by a

puted $1/p$ approximation;

djust to obtain $\lfloor f/p \rfloor$.

peedup: "2-adic inverse";

omery reduction."

e.g. 314159265358 mod 271828:

Precompute

$\lfloor 1000000000000/271828 \rfloor$

$= 3678796$.

Compute

$314159 \cdot 3678796$

$= 1155726872564$.

Compute

$314159265358 - 1155726 \cdot 271828$

$= 578230$.

Oops, too big:

$578230 - 271828 = 306402$.

$306402 - 271828 = 34574$.

We can

p is cho

to make

Special

for $\mathbf{F}_p^*$,

but not

Curve25

NIST P-

secp112

Divides

gls1271:

degree-2

n

f mod p?

:

$\lfloor f/p \rfloor$.

a

approximation;

tain $\lfloor f/p \rfloor$.

2-adic inverse";

uction."

e.g. 314159265358 mod 271828:

Precompute

$\lfloor 1000000000000/271828 \rfloor$

$= 3678796$.

Compute

$314159 \cdot 3678796$

$= 1155726872564$.

Compute

$314159265358 - 1155726 \cdot 271828$

$= 578230$.

Oops, too big:

$578230 - 271828 = 306402$.

$306402 - 271828 = 34574$.

We can do better:

p is chosen with a

to make f mod p

Special primes hur

for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$),

but not for elliptic

Curve25519: $p =$

NIST P-224: $p =$

secp112r1: $p = (2$

Divides special for

gls1271: $p = 2^{127}$

degree-2 extension

ntion;

⌋.

erse";

e.g. 314159265358 mod 271828:

Precompute

$\lfloor 1000000000000/271828 \rfloor$

$= 3678796$.

Compute

$314159 \cdot 3678796$

$= 1155726872564$.

Compute

$314159265358 - 1155726 \cdot 271828$

$= 578230$.

Oops, too big:

$578230 - 271828 = 306402$.

$306402 - 271828 = 34574$.

We can do better: normally

p is chosen with a special fo

to make f mod p much fast

Special primes hurt security

for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc.,

but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{9}$

secp112r1: $p = (2^{128} - 3)/$

Divides special form.

gls1271: $p = 2^{127} - 1$, with

degree-2 extension (a bit sca

e.g. 314159265358 mod 271828:

Precompute

$\lfloor 1000000000000/271828 \rfloor$

$= 3678796$.

Compute

$314159 \cdot 3678796$

$= 1155726872564$.

Compute

$314159265358 - 1155726 \cdot 271828$

$= 578230$.

Oops, too big:

$578230 - 271828 = 306402$.

$306402 - 271828 = 34574$.

We can do better: normally p is chosen with a special form to make f mod p much faster.

Special primes hurt security for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc., but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{96} + 1$.

secp112r1: $p = (2^{128} - 3)/76439$.
Divides special form.

gls1271: $p = 2^{127} - 1$, with degree-2 extension (a bit scary).

159265358 mod 271828:

oute

0000000/271828⌋

796.

e

3678796

726872564.

e

265358 − 1155726 · 271828

30.

oo big:

− 271828 = 306402.

− 271828 = 34574.

We can do better: normally p is chosen with a special form to make f mod p much faster.

Special primes hurt security for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc., but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{96} + 1$.

secp112r1: $p = (2^{128} - 3)/76439$. *Divides* special form.

gls1271: $p = 2^{127} - 1$, with degree-2 extension (a bit scary).

Small ex

Then 10

e.g. 314

314159 ·

314159(

−94247

−67711

Easily ac

to the ra

by addin

e.g. −67

8 mod 271828:

271828⌋

.

.155726 · 271828

= 306402.

= 34574.

We can do better: normally p is chosen with a special form to make f mod p much faster.

Special primes hurt security for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc., but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{96} + 1$.

secp112r1: $p = (2^{128} - 3)/76439$. *Divides* special form.

gls1271: $p = 2^{127} - 1$, with degree-2 extension (a bit scary).

Small example: p

Then $1000000a +$

e.g. 31415926535

$314159 \cdot 1000000$

$314159(-3) + 265$

$-942477 + 26535$

-677119.

Easily adjust $b - 3$

to the range $\{0, 1,$

by adding/subtrac

e.g. $-677119 \equiv 3$

.828:

271828

.

We can do better: normally p is chosen with a special form to make f mod p much faster.

Special primes hurt security for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc., but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{96} + 1$.

secp112r1: $p = (2^{128} - 3)/76439$. *Divides* special form.

gls1271: $p = 2^{127} - 1$, with degree-2 extension (a bit scary).

Small example: $p = 100000$

Then $1000000a + b \equiv b - 3$

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$ to the range $\{0, 1, \ldots, p - 1$ by adding/subtracting a few e.g. $-677119 \equiv 322884$.

We can do better: normally p is chosen with a special form to make f mod p much faster.

Special primes hurt security for $\mathbf{F}_p^*$, Clock($\mathbf{F}_p$), etc., but not for elliptic curves!

Curve25519: $p = 2^{255} - 19$.

NIST P-224: $p = 2^{224} - 2^{96} + 1$.

secp112r1: $p = (2^{128} - 3)/76439$. *Divides* special form.

gls1271: $p = 2^{127} - 1$, with degree-2 extension (a bit scary).

Small example: $p = 1000003$. Then $1000000a + b \equiv b - 3a$.

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358 \equiv$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$ to the range $\{0, 1, \ldots, p - 1\}$ by adding/subtracting a few p's: e.g. $-677119 \equiv 322884$.

do better: normally
sen with a special form
f mod p much faster.

primes hurt security
Clock($\mathbf{F}_p$), etc.,
for elliptic curves!

519: $p = 2^{255} - 19$.

224: $p = 2^{224} - 2^{96} + 1$.

r1: $p = (2^{128} - 3)/76439$.
special form.

$p = 2^{127} - 1$, with
extension (a bit scary).

Small example: $p = 1000003$.
Then $1000000a + b \equiv b - 3a$.

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358 \equiv$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$
to the range $\{0, 1, \ldots, p - 1\}$
by adding/subtracting a few p's:
e.g. $-677119 \equiv 322884$.

Hmmm,

Conditio
and leak
Can elim
but adju

Speedup
for inter
"Lazy re
Adjust o

$b - 3a$ is
to contin

normally
special form
much faster.

rt security
etc.,
curves!

$2^{255} - 19$.

$2^{224} - 2^{96} + 1$.

$^{128} - 3)/76439$.
rm.

$- 1$, with
n (a bit scary).

Small example: $p = 1000003$.
Then $1000000a + b \equiv b - 3a$.

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358 \equiv$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$
to the range $\{0, 1, \ldots, p - 1\}$
by adding/subtracting a few p's:
e.g. $-677119 \equiv 322884$.

Hmmm, is adjustm

Conditional branch
and leak secrets th
Can eliminate the
but adjustment isn

Speedup: Skip the
for intermediate re
"Lazy reduction."
Adjust only for ou

$b - 3a$ is small eno
to continue compu

orm

er.

$^6 + 1$.

76439.

ary).

Small example: $p = 1000003$.
Then $1000000a + b \equiv b - 3a$.

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358 \equiv$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$
to the range $\{0, 1, \ldots, p - 1\}$
by adding/subtracting a few p's:
e.g. $-677119 \equiv 322884$.

Hmmm, is adjustment so ea

Conditional branches are slo
and leak secrets through tim
Can eliminate the branches,
but adjustment isn't free.

Speedup: Skip the adjustme
for intermediate results.
"Lazy reduction."
Adjust only for output.

$b - 3a$ is small enough
to continue computations.

Small example: $p = 1000003$.
Then $1000000a + b \equiv b - 3a$.

e.g. $314159265358 =$
$314159 \cdot 1000000 + 265358 \equiv$
$314159(-3) + 265358 =$
$-942477 + 265358 =$
-677119.

Easily adjust $b - 3a$
to the range $\{0, 1, \ldots, p - 1\}$
by adding/subtracting a few p's:
e.g. $-677119 \equiv 322884$.

Hmmm, is adjustment so easy?

Conditional branches are slow
and leak secrets through timing.
Can eliminate the branches,
but adjustment isn't free.

Speedup: Skip the adjustment
for intermediate results.
"Lazy reduction."
Adjust only for output.

$b - 3a$ is small enough
to continue computations.

xample: $p = 1000003$.

$000000a + b \equiv b - 3a$.

$159265358 =$

$1000000 + 265358 \equiv$

$-3) + 265358 =$

$7 + 265358 =$

9.

djust $b - 3a$

ange $\{0, 1, \ldots, p - 1\}$

ng/subtracting a few p's:

$77119 \equiv 322884$.

Hmmm, is adjustment so easy?

Conditional branches are slow and leak secrets through timing. Can eliminate the branches, but adjustment isn't free.

Speedup: Skip the adjustment for intermediate results. "Lazy reduction." Adjust only for output.

$b - 3a$ is small enough to continue computations.

Can dela

multiplic

e.g. To s

in $\mathbf{Z}/10$

$3t^5 + 1t$

obtainin

$14t^7 + 4$

$82t^3 + 4$

Reduce:

$(-3c_i)t^i$

$64t^3 - 3$

Carry: 8

$1t^3 + 2t$

$= 1000003$.

$b \equiv b - 3a$.

$3 =$

$+ 265358 \equiv$

$5358 =$

$8 =$

$3a$

$, \ldots, p - 1\}$

ting a few p's:

22884.

Hmmm, is adjustment so easy?

Conditional branches are slow and leak secrets through timing. Can eliminate the branches, but adjustment isn't free.

Speedup: Skip the adjustment for intermediate results. "Lazy reduction." Adjust only for output.

$b - 3a$ is small enough to continue computations.

Can delay carries u

multiplication by 3

e.g. To square 314

in $\mathbf{Z}/1000003$: Sq

$3t^5 + 1t^4 + 4t^3 +$

obtaining $9t^{10} + 6$

$14t^7 + 48t^6 + 72t$

$82t^3 + 43t^2 + 90t$

Reduce: replace (c

$(-3c_i)t^i$, obtainin

$64t^3 - 32t^2 + 48t$

Carry: $8t^6 - 4t^5 -$

$1t^3 + 2t^2 + 2t^1 -$

3.

a.

$\equiv$

}

p's:

Hmmm, is adjustment so easy?

Conditional branches are slow and leak secrets through timing. Can eliminate the branches, but adjustment isn't free.

Speedup: Skip the adjustment for intermediate results. "Lazy reduction." Adjust only for output.

$b - 3a$ is small enough to continue computations.

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1$ obtaining $9t^{10} + 6t^9 + 25t^8$ $14t^7 + 48t^6 + 72t^5 + 59t^4$ $82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 +$ $64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 +$ $1t^3 + 2t^2 + 2t^1 - 3t^0$.

Hmmm, is adjustment so easy?

Conditional branches are slow and leak secrets through timing. Can eliminate the branches, but adjustment isn't free.

Speedup: Skip the adjustment for intermediate results. "Lazy reduction." Adjust only for output.

$b - 3a$ is small enough to continue computations.

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$, obtaining $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 + 32t^4 + 64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 + 1t^3 + 2t^2 + 2t^1 - 3t^0$.

is adjustment so easy?

nal branches are slow
secrets through timing.
inate the branches,
stment isn't free.

: Skip the adjustment
mediate results.
eduction."
nly for output.

s small enough
nue computations.

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$, obtaining $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 + 32t^4 + 64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 + 1t^3 + 2t^2 + 2t^1 - 3t^0$.

To minin
mix redu
carrying

e.g. Star
$25t^8 + 1$
$82t^3 + 4$

Reduce
$t^5 \rightarrow t^6$
$56t^6 - 5$
$90t^1 + 8$

Finish re
$64t^3 - 3$
$t^0 \rightarrow t^1$
$-4t^5 - 2$

ment so easy?

hes are slow
hrough timing.
branches,
n't free.

e adjustment
esults.

tput.

ough
utations.

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$, obtaining $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 + 32t^4 + 64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 + 1t^3 + 2t^2 + 2t^1 - 3t^0$.

To minimize poly
mix reduction and
carrying the top sc

e.g. Start from squ
$25t^8 + 14t^7 + 48t^6$
$82t^3 + 43t^2 + 90t$

Reduce $t^{10} \to t^4$
$t^5 \to t^6$: $6t^9 + 2$
$56t^6 - 5t^5 + 2t^4$
$90t^1 + 81t^0$.

Finish reduction:
$64t^3 - 32t^2 + 48t$
$t^0 \to t^1 \to t^2 \to$
$-4t^5 - 2t^4 + 1t^3 +$

sy?

w

ning.

ent

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$, obtaining $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 + 32t^4 + 64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 + 1t^3 + 2t^2 + 2t^1 - 3t^0$.

To minimize poly degree, mix reduction and carrying, carrying the top sooner.

e.g. Start from square $9t^{10}$ –
$25t^8 + 14t^7 + 48t^6 + 72t^5 +$
$82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce $t^{10} \to t^4$ and carry
$t^5 \to t^6$: $6t^9 + 25t^8 + 14t$
$56t^6 - 5t^5 + 2t^4 + 82t^3 + 4$
$90t^1 + 81t^0$.

Finish reduction: $-5t^5 + 2t$
$64t^3 - 32t^2 + 48t^1 - 87t^0$.
$t^0 \to t^1 \to t^2 \to t^3 \to t^4$ –
$-4t^5 - 2t^4 + 1t^3 + 2t^2 - 1t^1$

Can delay carries until after multiplication by 3.

e.g. To square 314159 in $\mathbf{Z}/1000003$: Square poly $3t^5 + 1t^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$, obtaining $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce: replace $(c_i)t^{6+i}$ by $(-3c_i)t^i$, obtaining $72t^5 + 32t^4 + 64t^3 - 32t^2 + 48t^1 - 63t^0$.

Carry: $8t^6 - 4t^5 - 2t^4 + 1t^3 + 2t^2 + 2t^1 - 3t^0$.

To minimize poly degree, mix reduction and carrying, carrying the top sooner.

e.g. Start from square $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce $t^{10} \to t^4$ and carry $t^4 \to t^5 \to t^6$: $6t^9 + 25t^8 + 14t^7 + 56t^6 - 5t^5 + 2t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Finish reduction: $-5t^5 + 2t^4 + 64t^3 - 32t^2 + 48t^1 - 87t^0$. Carry $t^0 \to t^1 \to t^2 \to t^3 \to t^4 \to t^5$: $-4t^5 - 2t^4 + 1t^3 + 2t^2 - 1t^1 + 3t^0$.

ay carries until after
cation by 3.

square 314159
00003: Square poly
$^4 + 4t^3 + 1t^2 + 5t^1 + 9t^0$,
g $9t^{10} + 6t^9 + 25t^8 +$
$8t^6 + 72t^5 + 59t^4 +$
$3t^2 + 90t^1 + 81t^0$.

replace $(c_i)t^{6+i}$ by
, obtaining $72t^5 + 32t^4 +$
$2t^2 + 48t^1 - 63t^0$.

$t^6 - 4t^5 - 2t^4 +$
$^2 + 2t^1 - 3t^0$.

To minimize poly degree,
mix reduction and carrying,
carrying the top sooner.

e.g. Start from square $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce $t^{10} \to t^4$ and carry $t^4 \to t^5 \to t^6$: $6t^9 + 25t^8 + 14t^7 + 56t^6 - 5t^5 + 2t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Finish reduction: $-5t^5 + 2t^4 + 64t^3 - 32t^2 + 48t^1 - 87t^0$. Carry $t^0 \to t^1 \to t^2 \to t^3 \to t^4 \to t^5$: $-4t^5 - 2t^4 + 1t^3 + 2t^2 - 1t^1 + 3t^0$.

Speedup

$p = 2^{61}$

Five coe
$f_4 t^4 + f_3$
Most co

Square
Coeff of

Reduce:
$\cdots + (2$
Coeff co
Very litt
addition
on 32-bi

until after
3.

4159
uare poly
$1t^2 + 5t^1 + 9t^0$,
$5t^9 + 25t^8 +$
$t^5 + 59t^4 +$
$t^1 + 81t^0$.

$c_i)t^{6+i}$ by
g $72t^5 + 32t^4 +$
$t^1 - 63t^0$.

$- 2t^4 +$
$3t^0$.

To minimize poly degree, mix reduction and carrying, carrying the top sooner.

e.g. Start from square $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce $t^{10} \to t^4$ and carry $t^4 \to t^5 \to t^6$: $6t^9 + 25t^8 + 14t^7 + 56t^6 - 5t^5 + 2t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Finish reduction: $-5t^5 + 2t^4 + 64t^3 - 32t^2 + 48t^1 - 87t^0$. Carry $t^0 \to t^1 \to t^2 \to t^3 \to t^4 \to t^5$: $-4t^5 - 2t^4 + 1t^3 + 2t^2 - 1t^1 + 3t^0$.

<u>Speedup: non-inte</u>

$p = 2^{61} - 1$.

Five coeffs in radi
$f_4 t^4 + f_3 t^3 + f_2 t^2$
Most coeffs could

Square $\cdots + 2(f_4 f$
Coeff of t^5 could

Reduce: $2^{65} = 2^4$
$\cdots + (2^5(f_4 f_1 + f_3$
Coeff could be $>$
Very little room fo
additions, delayed
on 32-bit platform

$+ 9t^0$,

$+$

$+$

$32t^4 +$

To minimize poly degree, mix reduction and carrying, carrying the top sooner.

e.g. Start from square $9t^{10} + 6t^9 + 25t^8 + 14t^7 + 48t^6 + 72t^5 + 59t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Reduce $t^{10} \to t^4$ and carry $t^4 \to t^5 \to t^6$: $6t^9 + 25t^8 + 14t^7 + 56t^6 - 5t^5 + 2t^4 + 82t^3 + 43t^2 + 90t^1 + 81t^0$.

Finish reduction: $-5t^5 + 2t^4 + 64t^3 - 32t^2 + 48t^1 - 87t^0$. Carry $t^0 \to t^1 \to t^2 \to t^3 \to t^4 \to t^5$: $-4t^5 - 2t^4 + 1t^3 + 2t^2 - 1t^1 + 3t^0$.

Speedup: non-integer radix

$p = 2^{61} - 1$.

Five coeffs in radix 2^{13}? $f_4 t^4 + f_3 t^3 + f_2 t^2 + f_1 t^1 + f$
Most coeffs could be 2^{12}.

Square $\cdots + 2(f_4 f_1 + f_3 f_2)t^5$
Coeff of t^5 could be $> 2^{25}$.

Reduce: $2^{65} = 2^4$ in $\mathbf{Z}/(2^{61}$
$\cdots + (2^5(f_4 f_1 + f_3 f_2) + f_0^2)t$
Coeff could be $> 2^{29}$.
Very little room for additions, delayed carries, et on 32-bit platforms.

To minimize poly degree,
mix reduction and carrying,
carrying the top sooner.

e.g. Start from square $9t^{10}+6t^9+25t^8+14t^7+48t^6+72t^5+59t^4+82t^3+43t^2+90t^1+81t^0$.

Reduce $t^{10}\to t^4$ and carry $t^4\to t^5\to t^6$: $6t^9+25t^8+14t^7+56t^6-5t^5+2t^4+82t^3+43t^2+90t^1+81t^0$.

Finish reduction: $-5t^5+2t^4+64t^3-32t^2+48t^1-87t^0$. Carry $t^0\to t^1\to t^2\to t^3\to t^4\to t^5$: $-4t^5-2t^4+1t^3+2t^2-1t^1+3t^0$.

Speedup: non-integer radix

$p=2^{61}-1$.

Five coeffs in radix 2^{13}?
$f_4t^4+f_3t^3+f_2t^2+f_1t^1+f_0t^0$.
Most coeffs could be 2^{12}.

Square $\cdots+2(f_4f_1+f_3f_2)t^5+\cdots$.
Coeff of t^5 could be $>2^{25}$.

Reduce: $2^{65}=2^4$ in $\mathbf{Z}/(2^{61}-1)$;
$\cdots+(2^5(f_4f_1+f_3f_2)+f_0^2)t^0$.
Coeff could be $>2^{29}$.
Very little room for
additions, delayed carries, etc.
on 32-bit platforms.

mize poly degree,
uction and carrying,
the top sooner.

rt from square $9t^{10}+6t^9+$
$4t^7+48t^6+72t^5+59t^4+$
$43t^2+90t^1+81t^0$.

$t^{10} \to t^4$ and carry $t^4 \to$
^{5}: $6t^9+25t^8+14t^7+$
$5t^5+2t^4+82t^3+43t^2+$
$81t^0$.

eduction: $-5t^5+2t^4+$
$32t^2+48t^1-87t^0$. Carry
$\to t^2 \to t^3 \to t^4 \to t^5$:
$2t^4+1t^3+2t^2-1t^1+3t^0$.

Speedup: non-integer radix

$p = 2^{61} - 1$.

Five coeffs in radix 2^{13}?
$f_4 t^4 + f_3 t^3 + f_2 t^2 + f_1 t^1 + f_0 t^0$.
Most coeffs could be 2^{12}.

Square $\cdots + 2(f_4 f_1 + f_3 f_2) t^5 + \cdots$.
Coeff of t^5 could be $> 2^{25}$.

Reduce: $2^{65} = 2^4$ in $\mathbf{Z}/(2^{61}-1)$;
$\cdots + (2^5(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.
Coeff could be $> 2^{29}$.
Very little room for
additions, delayed carries, etc.
on 32-bit platforms.

Scaled:
f_4 is mu
f_3 is mu
f_2 is mu
f_1 is mu
f_0 is mu
$\cdots + (2$

Better:
f_4 is mu
f_3 is mu
f_2 is mu
f_1 is mu
f_0 is mu
Saves a

degree,
carrying,
ooner.

uare $9t^{10}+6t^9+$
$+72t^5+59t^4+$
$^1+81t^0$.

and carry $t^4 \to$
$5t^8+14t^7+$
$+82t^3+43t^2+$

$-5t^5+2t^4+$
$^1-87t^0$. Carry
$t^3 \to t^4 \to t^5$:
$-2t^2-1t^1+3t^0$.

Speedup: non-integer radix

$p = 2^{61} - 1$.

Five coeffs in radix 2^{13}?
$f_4 t^4 + f_3 t^3 + f_2 t^2 + f_1 t^1 + f_0 t^0$.
Most coeffs could be 2^{12}.

Square $\cdots + 2(f_4 f_1 + f_3 f_2) t^5 + \cdots$.
Coeff of t^5 could be $> 2^{25}$.

Reduce: $2^{65} = 2^4$ in $\mathbf{Z}/(2^{61}-1)$;
$\cdots + (2^5(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.
Coeff could be $> 2^{29}$.
Very little room for
additions, delayed carries, etc.
on 32-bit platforms.

Scaled: Evaluate a
f_4 is multiple of 2^5
f_3 is multiple of 2^3
f_2 is multiple of 2^2
f_1 is multiple of 2^1
f_0 is multiple of 2^0
$\cdots + (2^{-60}(f_4 f_1 +$

Better: Non-intege
f_4 is multiple of 2^4
f_3 is multiple of 2^3
f_2 is multiple of 2^2
f_1 is multiple of 2^1
f_0 is multiple of 2^0
Saves a few bits in

$+6t^9+$

$59t^4+$

$t^4 \rightarrow$

$7 +$

$43t^2+$

$4 +$

Carry

$\rightarrow t^5$:

$+3t^0$.

Speedup: non-integer radix

$p = 2^{61} - 1$.

Five coeffs in radix 2^{13}?
$f_4 t^4 + f_3 t^3 + f_2 t^2 + f_1 t^1 + f_0 t^0$.
Most coeffs could be 2^{12}.

Square $\cdots + 2(f_4 f_1 + f_3 f_2)t^5 + \cdots$.
Coeff of t^5 could be $> 2^{25}$.

Reduce: $2^{65} = 2^4$ in $\mathbf{Z}/(2^{61} - 1)$;
$\cdots + (2^5(f_4 f_1 + f_3 f_2) + f_0^2)t^0$.
Coeff could be $> 2^{29}$.
Very little room for
additions, delayed carries, etc.
on 32-bit platforms.

Scaled: Evaluate at $t = 1$.
f_4 is multiple of 2^{52};
f_3 is multiple of 2^{39};
f_2 is multiple of 2^{26};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0. Reduce:
$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2$

Better: Non-integer radix 2
f_4 is multiple of 2^{49};
f_3 is multiple of 2^{37};
f_2 is multiple of 2^{25};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0.
Saves a few bits in coeffs.

Speedup: non-integer radix

$p = 2^{61} - 1$.

Five coeffs in radix 2^{13}?
$f_4 t^4 + f_3 t^3 + f_2 t^2 + f_1 t^1 + f_0 t^0$.
Most coeffs could be 2^{12}.

Square $\cdots + 2(f_4 f_1 + f_3 f_2) t^5 + \cdots$.
Coeff of t^5 could be $> 2^{25}$.

Reduce: $2^{65} = 2^4$ in $\mathbf{Z}/(2^{61} - 1)$;
$\cdots + (2^5 (f_4 f_1 + f_3 f_2) + f_0^2) t^0$.
Coeff could be $> 2^{29}$.
Very little room for
additions, delayed carries, etc.
on 32-bit platforms.

Scaled: Evaluate at $t = 1$.
f_4 is multiple of 2^{52};
f_3 is multiple of 2^{39};
f_2 is multiple of 2^{26};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0. Reduce:
$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.

Better: Non-integer radix $2^{12.2}$.
f_4 is multiple of 2^{49};
f_3 is multiple of 2^{37};
f_2 is multiple of 2^{25};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0.
Saves a few bits in coeffs.

: non-integer radix

$- 1$.

ffs in radix 2^{13}?

$_3 t^3 + f_2 t^2 + f_1 t^1 + f_0 t^0$.

effs could be 2^{12}.

$\cdots + 2(f_4 f_1 + f_3 f_2) t^5 + \cdots$.

t^5 could be $> 2^{25}$.

$2^{65} = 2^4$ in $\mathbf{Z}/(2^{61} - 1)$;

$^5(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.

uld be $> 2^{29}$.

le room for

s, delayed carries, etc.

t platforms.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};

f_3 is multiple of 2^{39};

f_2 is multiple of 2^{26};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0. Reduce:

$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};

f_3 is multiple of 2^{37};

f_2 is multiple of 2^{25};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0.

Saves a few bits in coeffs.

More ba

NIST P-

$2^{256} - 2$

i.e. $t^8 -$

evaluate

eger radix

× 2^{13}?

$+ f_1 t^1 + f_0 t^0$.

be 2^{12}.

$_1 + f_3 f_2) t^5 + \cdots$.

be $> 2^{25}$.

in $\mathbf{Z}/(2^{61} - 1)$;

$f_2) + f_0^2) t^0$.

2^{29}.

or

carries, etc.

s.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};
f_3 is multiple of 2^{39};
f_2 is multiple of 2^{26};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0. Reduce:
$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};
f_3 is multiple of 2^{37};
f_2 is multiple of 2^{25};
f_1 is multiple of 2^{13};
f_0 is multiple of 2^0.
Saves a few bits in coeffs.

More bad choices

NIST P-256 prime
$2^{256} - 2^{224} + 2^{192}$
i.e. $t^8 - t^7 + t^6 +$
evaluated at $t = 2$

$f_0 t^0$.

$+ \cdots$.

$- 1)$;

0.

c.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};

f_3 is multiple of 2^{39};

f_2 is multiple of 2^{26};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0. Reduce:

$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2) t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};

f_3 is multiple of 2^{37};

f_2 is multiple of 2^{25};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0.

Saves a few bits in coeffs.

More bad choices from NIST

NIST P-256 prime:

$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

i.e. $t^8 - t^7 + t^6 + t^3 - 1$

evaluated at $t = 2^{32}$.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};

f_3 is multiple of 2^{39};

f_2 is multiple of 2^{26};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0. Reduce:

$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2)t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};

f_3 is multiple of 2^{37};

f_2 is multiple of 2^{25};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0.

Saves a few bits in coeffs.

More bad choices from NIST

NIST P-256 prime:

$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

i.e. $t^8 - t^7 + t^6 + t^3 - 1$

evaluated at $t = 2^{32}$.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};

f_3 is multiple of 2^{39};

f_2 is multiple of 2^{26};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0. Reduce:

$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2)t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};

f_3 is multiple of 2^{37};

f_2 is multiple of 2^{25};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0.

Saves a few bits in coeffs.

More bad choices from NIST

NIST P-256 prime:

$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

i.e. $t^8 - t^7 + t^6 + t^3 - 1$

evaluated at $t = 2^{32}$.

Reduction: replace $c_i t^{8+i}$ with

$c_i t^{7+i} - c_i t^{6+i} - c_i t^{3+i} + c_i t^i$.

Minor problem: often slower than

small const mult and one add.

Scaled: Evaluate at $t = 1$.

f_4 is multiple of 2^{52};

f_3 is multiple of 2^{39};

f_2 is multiple of 2^{26};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0. Reduce:

$\cdots + (2^{-60}(f_4 f_1 + f_3 f_2) + f_0^2)t^0$.

Better: Non-integer radix $2^{12.2}$.

f_4 is multiple of 2^{49};

f_3 is multiple of 2^{37};

f_2 is multiple of 2^{25};

f_1 is multiple of 2^{13};

f_0 is multiple of 2^0.

Saves a few bits in coeffs.

More bad choices from NIST

NIST P-256 prime:

$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

i.e. $t^8 - t^7 + t^6 + t^3 - 1$

evaluated at $t = 2^{32}$.

Reduction: replace $c_i t^{8+i}$ with

$c_i t^{7+i} - c_i t^{6+i} - c_i t^{3+i} + c_i t^i$.

Minor problem: often slower than

small const mult and one add.

Major problem: With radix 2^{32},

products are almost 2^{64}.

Sums are slightly above 2^{64}:

bad for every common CPU.

Need very frequent carries.