Multiplication of Large Integers

Michael Angel

6 May 2019

Overview

Using Fourier Transforms to multiply integers

- Theory
- Speed
- Code
- Demo

How?

Consider integers as polynomials of their radix/base

$$\begin{aligned}\mathbf{4092} &= \mathbf{2}(10)^0 + \mathbf{9}(10)^1 + \mathbf{0}(10)^2 + \mathbf{4}(10)^3 \\ \mathbf{373} &= \mathbf{3}(10)^0 + \mathbf{7}(10)^1 + \mathbf{3}(10)^2 + \mathbf{0}(10)^3\end{aligned}$$

How?

Consider integers as polynomials of their radix/base

$$\begin{aligned}\mathbf{4092} &= \mathbf{2}(10)^0 + \mathbf{9}(10)^1 + \mathbf{0}(10)^2 + \mathbf{4}(10)^3 \\ \mathbf{373} &= \mathbf{3}(10)^0 + \mathbf{7}(10)^1 + \mathbf{3}(10)^2 + \mathbf{0}(10)^3\end{aligned}$$

Polynomial multiplication is the convolution of coefficients

How?

Consider integers as polynomials of their radix/base

$$\mathbf{4092} = \mathbf{2}(10)^0 + \mathbf{9}(10)^1 + \mathbf{0}(10)^2 + \mathbf{4}(10)^3$$
$$\mathbf{373} = \mathbf{3}(10)^0 + \mathbf{7}(10)^1 + \mathbf{3}(10)^2 + \mathbf{0}(10)^3$$

Polynomial multiplication is the convolution of coefficients

$a = ..0, 0, 2, 9, 0, 4, 0, 0..$
$b = ..0, 0, 3, 7, 3, 0, 0, 0, ..$
$c = a * b$

$$c_i = (a * b)_i = \sum_{k=-\infty}^{\infty} a_k b_{i-k}$$

$$c_i = (a * b)_i = \sum_{k=-\infty}^{\infty} a_k b_{i-k}$$

$$c_i = \sum_{j=0}^{n-1} a_j b_{i-j}$$

$$c_0 = a_0 b_0 + a_1 b_{-1} + a_2 b_{-2} + a_3 b_{-3} = 2 \cdot 3 + 9 \cdot 0 + 0 \cdot 0 + 4 \cdot 0 = 6$$

$$c_1 = a_0 b_1 + a_1 b_0 + a_2 b_{-1} + a_3 b_{-2} = 2 \cdot 7 + 9 \cdot 3 + 0 \cdot 0 + 4 \cdot 0 = 41$$

$$c_2 = a_0 b_2 + a_1 b_1 + a_2 b_0 + a_3 b_{-1} = 2 \cdot 3 + 9 \cdot 7 + 0 \cdot 3 + 4 \cdot 0 = 69$$

$$c_3 = \ldots$$

How?

Consider integers as polynomials of their radix/base

$$\mathbf{4092} = \mathbf{2}(10)^0 + \mathbf{9}(10)^1 + \mathbf{0}(10)^2 + \mathbf{4}(10)^3$$
$$\mathbf{373} = \mathbf{3}(10)^0 + \mathbf{7}(10)^1 + \mathbf{3}(10)^2 + \mathbf{0}(10)^3$$

Polynomial multiplication is the convolution of coefficients

$a = ..0, 0, 2, 9, 0, 4, 0, 0..$

$b = ..0, 0, 3, 7, 3, 0, 0, 0, ..$

$c = a * b = ..0, 0, 6, 41, 69, 39, 28, 12, 0, 0, ..$

$c = 6(10)^0+41(10)^1+69(10)^2+39(10)^3+28(10)^4+12(10)^5+0(10)^6$

carry the 10's

$c = 6(10)^0 + 1(10)^1 + 3(10)^2 + 6(10)^3 + 2(10)^4 + 5(10)^5 + 1(10)^6$

$\mathbf{4092} \cdot \mathbf{373} = 1526316$

The Schönhage-Strassen Algorithm

By Arnold Schönhage and Volker Strassen in 1971.

Summary

- $a = \mathcal{F}(a), b = \mathcal{F}(b)$
- $c = a \cdot b$
- $c = \mathcal{F}^{-1}(c)$
- carry(c), make every coefficient less than 10

The Schönhage-Strassen Algorithm

$\mathcal{O}(nlog(n))$ complex multiplications (Fast Fourier Transforms).
Extra time needed to compute roots of unity.

$\mathcal{O}(nlog(n) \cdot log(log(n)))$ total running time

The Schönhage-Strassen Algorithm

$\mathcal{O}(nlog(n))$ complex multiplications (Fast Fourier Transforms).
Extra time needed to compute roots of unity.

$\mathcal{O}(nlog(n) \cdot log(log(n)))$ total running time

Schönhage-Strassen conjectured existence of $\mathcal{O}(nlog(n))$ integer multiplication algorithm.

The Schönhage-Strassen Algorithm

$\mathcal{O}(nlog(n))$ complex multiplications (Fast Fourier Transforms).
Extra time needed to compute roots of unity.

$\mathcal{O}(nlog(n) \cdot log(log(n)))$ total running time

Schönhage-Strassen conjectured existence of $\mathcal{O}(nlog(n))$ integer multiplication algorithm.

18 March, 2019 (seven weeks ago) David Harvey and Joris Van Der Hoeven published $\mathcal{O}(nlog(n))$ algorithm. [HVD]

Naïve Algorithm

```
   1234
 x 1234
     16
    120
    800
   4000
    120
    900
   6000
  30000
    800
   6000
  40000
 200000
   4000
  30000
 200000
1000000
```

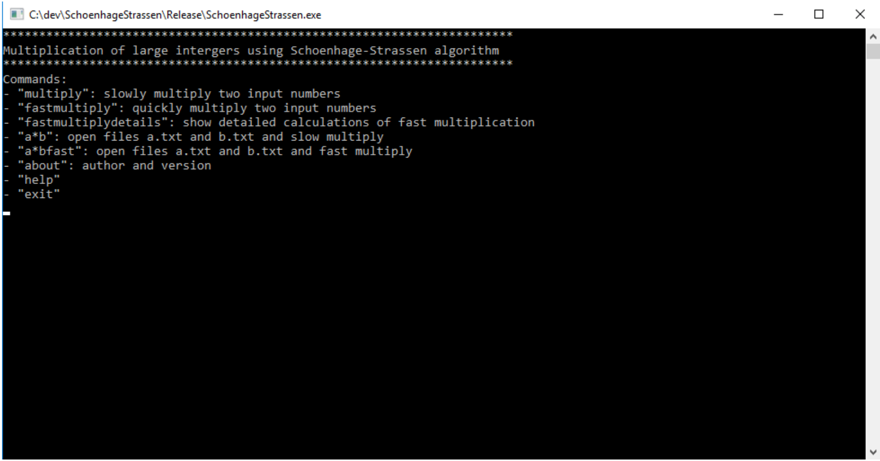
For each digit, multiply for each digit, then add.

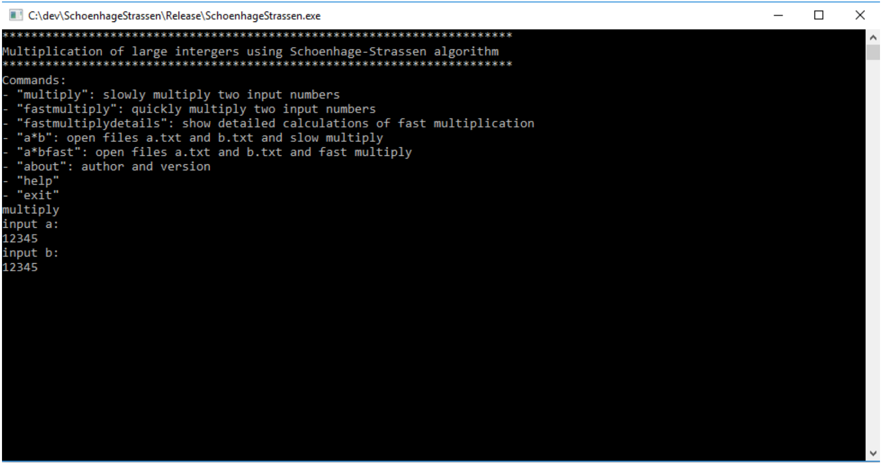
$\mathcal{O}(n^2)$ running time

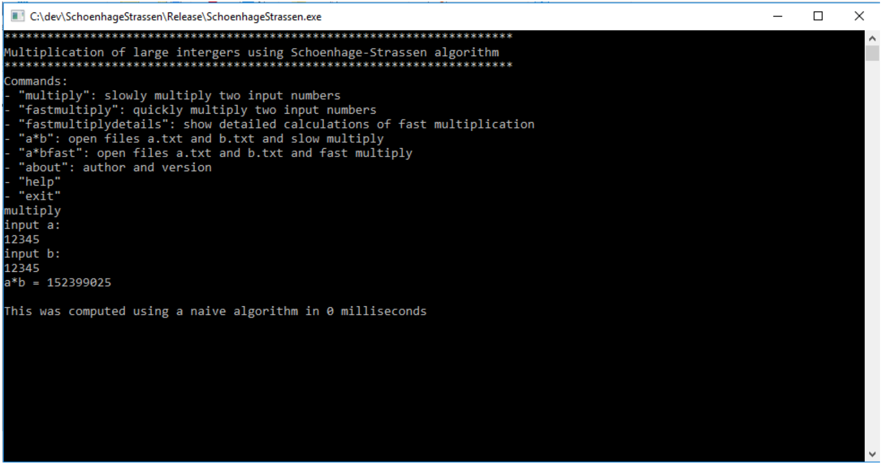
```cpp
bigInt operator * (const bigInt &v) const {
    bigInt product;
    for (long i = 0; i < a.size(); ++i) {
        for (long j = 0; j < v.a.size(); ++j) {
            vector<int> p(i + j, 0);
            int h = a[i] * v.a[j];
            while (h) {
                p.insert(p.begin(), h % 10);
                h = int(h / 10);
            }

            for (auto x : p) cout << x;
            cout << endl;

            product = product + bigInt(p);
        }
    }
    return product;
}
```

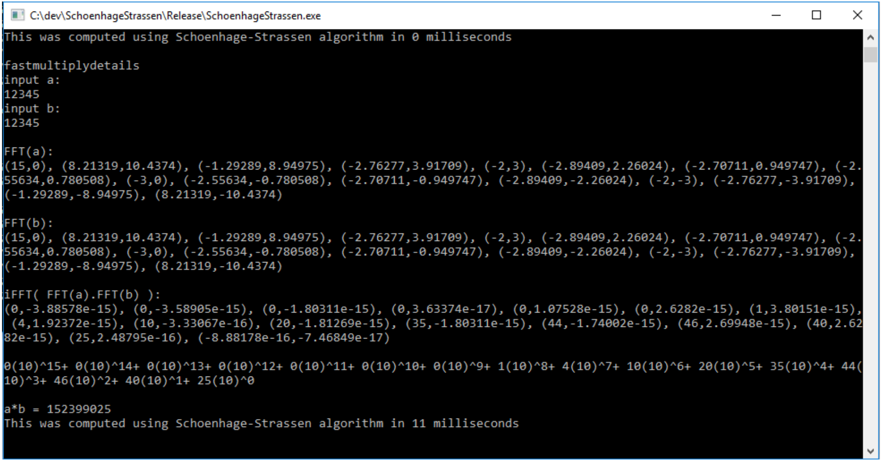
```
valarray<int> multiply(PolynomialCoefficients a, PolynomialCoefficients b, bool show_details) {
    PolynomialCoefficients c;
    FFT(a);
    FFT(b);
    c = a * b;
    iFFT(c);

    valarray<int> res(a.size());
    for (long i = 0; i < a.size(); i++) {
        res[i] = round(real(c[i]));
    }

    for (long i = a.size() - 1; i > 0; i--) {
        res[i] += res[i-1] % 10;
        res[i - 1] = int(res[i - 1] / 10);
        if (res[i] > 9) res[i - 1] += int(res[i] / 10);
        res[i] %= 10;
    }

    return res;
```

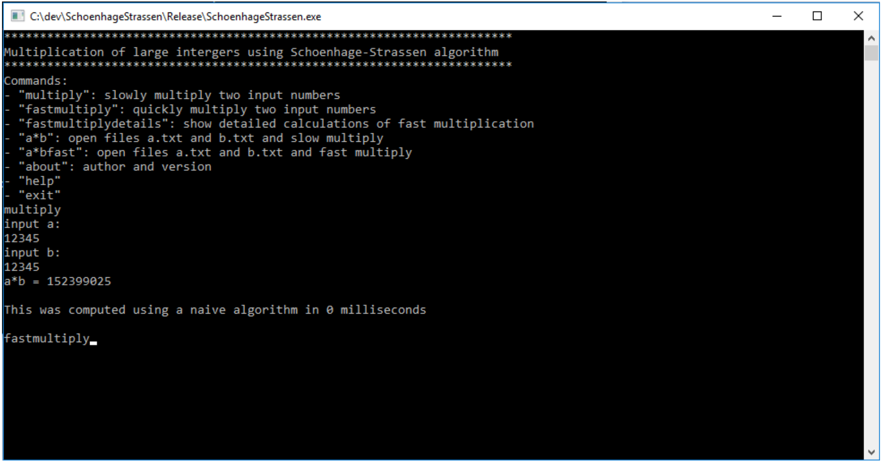
```cpp
const double PI = 3.141592653589793238460;

void FFT(PolynomialCoefficients& a) {
    // Cooley-Tukey algorithm: radix-2 decimation-in-time Fast Fourier Transform
    const size_t N = a.size();
    if (N <= 1) return;

    PolynomialCoefficients even = a[slice(0, N / 2, 2)];
    PolynomialCoefficients odd = a[slice(1, N / 2, 2)];

    FFT(even);
    FFT(odd);

    for (size_t k = 0; k < N/2; ++k)
    {
        Complex t = polar(1.0, -2*PI*k/N) * odd[k];
        // polar(rho, theta) = rho(cos(theta) + isin(theta)) = rho*e^(i*theta)
        // polar(1.0, -2*PI*k/N) = e^i(-2*PI*k/N)
        a[k] = even[k] + t;
        a[k + N / 2] = even[k] - t;
    }
}
```


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
****
Multiplication of large intergers using Schoenhage-Strassen algorithm
****
Commands:
- "multiply": slowly multiply two input numbers
- "fastmultiply": quickly multiply two input numbers
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
****
Multiplication of large intergers using Schoenhage-Strassen algorithm
****
Commands:
- "multiply": slowly multiply two input numbers
- "fastmultiply": quickly multiply two input numbers
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"
multiply
input a:
12345
input b:
12345

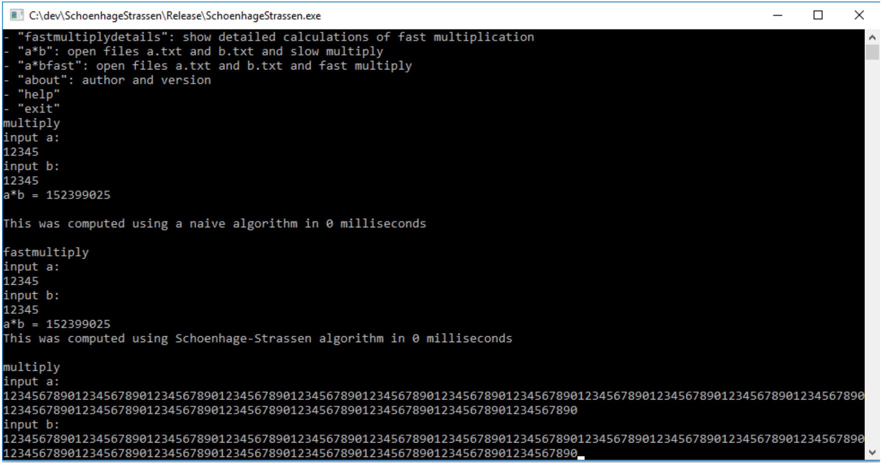

C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
****
Multiplication of large intergers using Schoenhage-Strassen algorithm
****
Commands:
- "multiply": slowly multiply two input numbers
- "fastmultiply": quickly multiply two input numbers
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"
multiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using a naive algorithm in 0 milliseconds

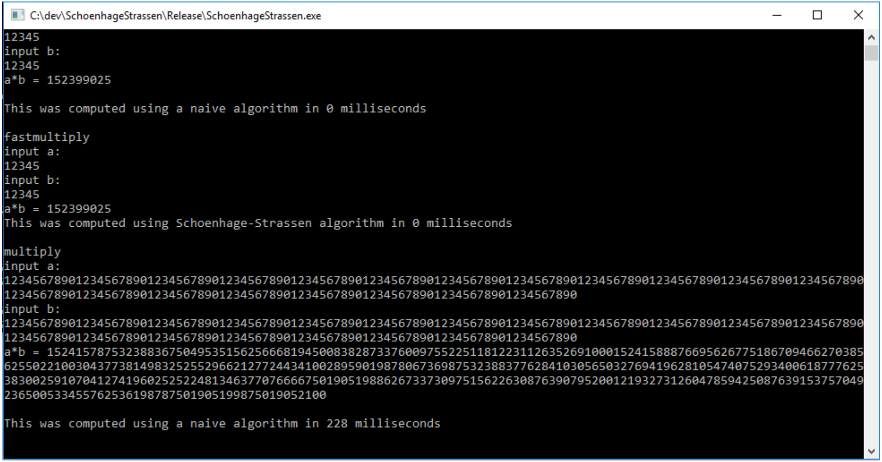

```
C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds

fastmultiplydetails
input a:
12345
input b:
12345

FFT(a):
(15,0), (8.21319,10.4374), (-1.29289,8.94975), (-2.76277,3.91709), (-2,3), (-2.89409,2.26024), (-2.70711,0.949747), (-2.
55634,0.780508), (-3,0), (-2.55634,-0.780508), (-2.70711,-0.949747), (-2.89409,-2.26024), (-2,-3), (-2.76277,-3.91709),
(-1.29289,-8.94975), (8.21319,-10.4374)

FFT(b):
(15,0), (8.21319,10.4374), (-1.29289,8.94975), (-2.76277,3.91709), (-2,3), (-2.89409,2.26024), (-2.70711,0.949747), (-2.
55634,0.780508), (-3,0), (-2.55634,-0.780508), (-2.70711,-0.949747), (-2.89409,-2.26024), (-2,-3), (-2.76277,-3.91709),
(-1.29289,-8.94975), (8.21319,-10.4374)

iFFT( FFT(a).FFT(b) ):
(0,-3.88578e-15), (0,-3.58905e-15), (0,-1.80311e-15), (0,3.63374e-17), (0,1.07528e-15), (0,2.6282e-15), (1,3.80151e-15),
 (4,1.92372e-15), (10,-3.33067e-16), (20,-1.81269e-15), (35,-1.80311e-15), (44,-1.74002e-15), (46,2.69948e-15), (40,2.62
82e-15), (25,2.48795e-16), (-8.88178e-16,-7.46849e-17)

0(10)^15+ 0(10)^14+ 0(10)^13+ 0(10)^12+ 0(10)^11+ 0(10)^10+ 0(10)^9+ 1(10)^8+ 4(10)^7+ 10(10)^6+ 20(10)^5+ 35(10)^4+ 44(
10)^3+ 46(10)^2+ 40(10)^1+ 25(10)^0

a*b = 152399025
This was computed using Schoenhage-Strassen algorithm in 11 milliseconds
```


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
****
Multiplication of large intergers using Schoenhage-Strassen algorithm
****
Commands:
- "multiply": slowly multiply two input numbers
- "fastmultiply": quickly multiply two input numbers
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"
multiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using a naive algorithm in 0 milliseconds
fastmultiply

C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe

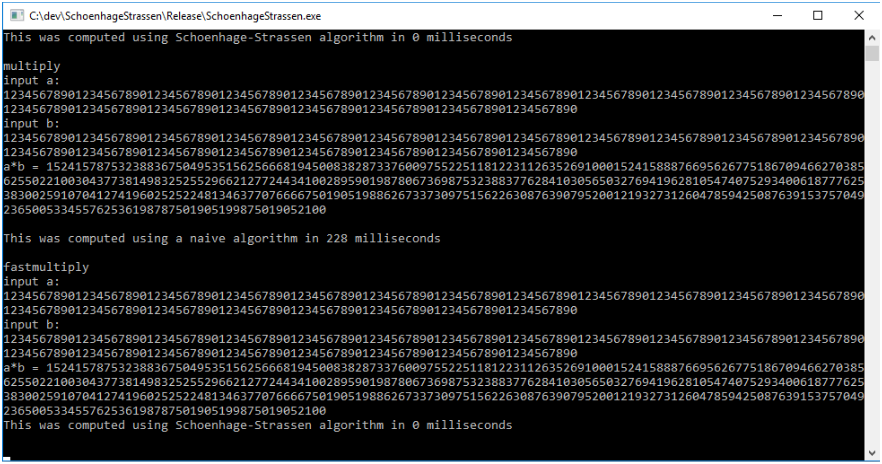
```
***********************************************************************
Multiplication of large intergers using Schoenhage-Strassen algorithm
***********************************************************************
Commands:
- "multiply": slowly multiply two input numbers
- "fastmultiply": quickly multiply two input numbers
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"
multiply
input a:
12345
input b:
12345
a*b = 152399025

This was computed using a naive algorithm in 0 milliseconds

fastmultiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds
```

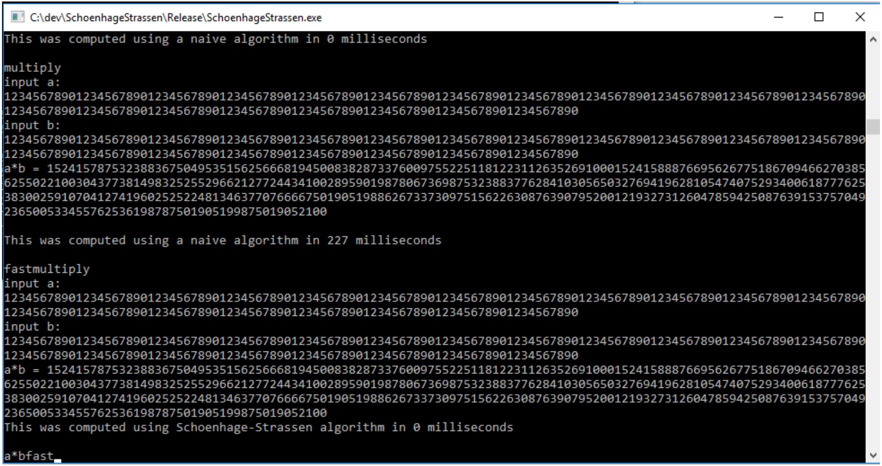

C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
- "fastmultiplydetails": show detailed calculations of fast multiplication
- "a*b": open files a.txt and b.txt and slow multiply
- "a*bfast": open files a.txt and b.txt and fast multiply
- "about": author and version
- "help"
- "exit"
multiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using a naive algorithm in 0 milliseconds
fastmultiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds
multiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890

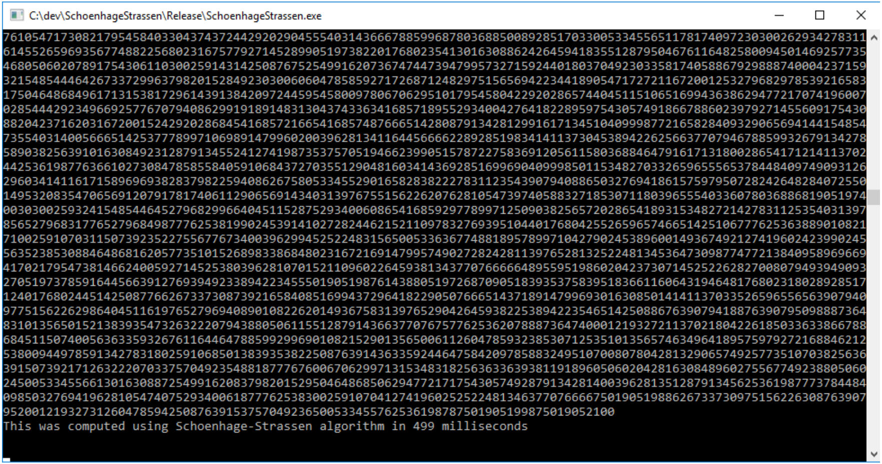

C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
12345
input b:
12345
a*b = 152399025
This was computed using a naive algorithm in 0 milliseconds
fastmultiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds
multiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
236500533455762536198787501905199875019052100
This was computed using a naive algorithm in 228 milliseconds

C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe

```
fastmultiply
input a:
12345
input b:
12345
a*b = 152399025
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds

multiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
236500533455762536198787501905199875019052100

This was computed using a naive algorithm in 228 milliseconds

fastmultiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds
multiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
236500533455762536198787501905199875019052100
This was computed using a naive algorithm in 228 milliseconds
fastmultiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
236500533455762536198787501905199875019052100
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
This was computed using a naive algorithm in 0 milliseconds
multiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
2365005334557625361987875019051998750190521OO
This was computed using a naive algorithm in 227 milliseconds
fastmultiply
input a:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
input b:
123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
12345678901234567890123456789012345678901234567890123456789012345678901234567890
a*b = 152415787532388367504953515625666819450083828733760097552251181223112635269100015241588876695626775186709466270385
625502210030437738149832525529662127724434100289590198780673698753238837762841030565032769419628105474075293400618777625
383002591070412741960252522481346377076666750190519886267337309751562263087639079520012193273126047859425087639153757049
236500533455762536198787501905199875019052100
This was computed using Schoenhage-Strassen algorithm in 0 milliseconds
a*bfast


C:\dev\SchoenhageStrassen\Release\SchoenhageStrassen.exe
7610547173082179545840330437437244292029045554031436667885996878036885008928517033005334556511781740972303002629342783311
6145526596935677488225680231675779271452899051973822017680235413016308862426459418355128795046761164825800945014692577735
4680506020789175430611030025914314250876752549916207367474473947995732715924401803704923033581740588679298887400042371559
3215485444642673372996379820152849230300606047858592717268712482975156569422344189054717272116720012532796829785392165883
1750464868496171315381729614391384209724459545800978067062951017954580422920286574404511510651699436386294772170741960007
0285444292349669257767079408629919189148313043743363416857189552934004276418228959754305749186678860239792714556091754330
8820423716203167200152429202868454168572166541685748766651428087913428129916171345104099987721658284093290656941441548554
7355403140056665142537778997106989147996020039628134116445666622892851983414113730453894226256637707946788599326791342778
5890382563910163084923128791345524127419873537570519466239905157872275836912056115803688464791617131800286541712141137002
4425361987763661027308478585558405910684372703551290481603414369285169969040999850115348270332659655565378448409749093126
2960341411617158969693828379822594086267580533455290165828382227831123543907940886503276941861575979507282426482840725550
1495320835470656912079178174061129065691434031397675515622620762810547397405883271853071180396555403360780368868190519774
0030300259324154854464527968299664045115287529340060865416859297789971250903825657202865418931534827214278311253540313977
8565279683177652796849877762538199024539141027282446215211097832769395104401768042552659657466514251067776253638890108221
7100259107031150739235227556776734003962994525224831565005336367748818957899710427902453896001493674921274196024239902445
5635238530884648681620577351015268983386848023167216914799574902728242811397652813252248134536473098774772138409589696669
4170217954738146624005927145253803962810701521109602264593813437707666664895595198602042373071452522628270080794939490093
2705197378591644566391276939492338942234555019051987614388051972687090518393537583951836611606431946481768023180289285177
1240176802445142508776626733730873921658408516994372964182290507666514371891479969301630850141411370335265965565639079440
9775156226298640451161976527969408901082262014936758313976529042645938225389422354651425088676390794188763907950988873664
8310135650152138393547326322207943880506115512879143663770767577625362078887364740001219327211370218042261850336338667888
6845115074005636335932676116446478859929969010821529013565006112604785932385307125351013565746349641895759792721688462112
5380094497859134278318025910685013839353822508763914363359244647584209785883249510700807804281329065749257735107038256336
3915073921712632220703375704923548818777676006706299713153483182563633639381191896050602042816308489602755677492388050660
2450053345566130163088725499162083798201529504648685062947721717543057492879134281400396281351287913456253619877737844884
0985032769419628105474075293400618777625383002591070412741960252522481346377076666750190519886267337309751562263087639007
952001219327312604785942508763915375704923650053345576253619878750190519987501905210
This was computed using Schoenhage-Strassen algorithm in 499 milliseconds

References

[SSA] A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing 7 (1971)

[HVD] David Harvey, Joris Van Der Hoeven. Integer multiplication in time O(n log n). 2019. ffhal-02070778f https://hal.archives-ouvertes.fr/hal-02070778/document