

СПЕЦІАЛЬНІ РОЗДІЛИ

ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2

Багаторозрядна модулярна арифметика

1. Мета роботи

Отримання практичних навичок програмної реалізації багаторозрядної арифметики; ознайомлення з прийомами ефективної реалізації критичних по часу ділянок програмного коду та методами оцінки їх ефективності.

3. Завдання до комп'ютерного практикуму

А) Доопрацювати бібліотеку для роботи з m -бітними цілими числами, створену на комп'ютерному практикумі №1, додавши до неї такі операції:

- 1) обчислення НСД та НСК двох чисел;
- 2) додавання чисел за модулем;
- 3) віднімання чисел за модулем;
- 4) множення чисел та піднесення чисел до квадрату за модулем;
- 5) піднесення числа до багаторозрядного степеня d по модулю n .

Модулярну арифметику рекомендовано реалізовувати на базі редукції Баррета, піднесення до степеня – на базі схеми Горнера. Мова програмування, семантика функцій та спосіб реалізації можуть обиратись довільним чином.

Окрім основного завдання, ви також можете виконати додаткове завдання згідно варіанту.

Б) Проконтролювати коректність реалізації алгоритмів; зокрема, для декількох багаторозрядних a, b, c, n перевірити тотожності

В) Обчислити середній час виконання реалізованих арифметичних операцій. Підрахувати кількість тактів процесора (або інших одиниць виміру часу) на кожну операцію. Результати подати у вигляді таблиць або діаграм.

Хід роботи

Після до написання бібліотеки класу `bigint`, а саме функцій для роботи з модулярною арифметикою, створимо файл тестування що перевірить коректність нашої роботи. В ньому ми порівнюємо значення обраховані за допомогою бібліотеки та без неї.

Демонстрація роботи:

```
[[APython 3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.31.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from sympath.bignum import *
In [2]: from random import getrandbits
In [3]: a,b,c = getrandbits(100),getrandbits(100),getrandbits(100)
In [4]: A,B = bn(a),bn(b)
In [5]: gf = GF(n)
In [6]: A = gf(A)
In [7]: (A+B).base10()
Out[7]: 7314279215255366708751060950186973012557918216070663432393830978745501313277858801949714358823855644274276139519550389703371926199301855572800558795264946001826055723155522803299273
9873916628768808306107399098908453587319400084321423658219748535668803885801104586767063301261395212824231940553788695485673
In [8]: (a+b)%n
Out[8]: 7314279215255366708751060950186973012557918216070663432393830978745501313277858801949714358823855644274276139519550389703371926199301855572800558795264946001826055723155522803299273
9873916628768808306107399098908453587319400084321423658219748535668803885801104586767063301261395212824231940553788695485673
In [9]: A>B
Out[9]: False
In [10]: (B-A).base10()
Out[10]: 7841802387909561814939930409124114437189639488404364381841195358408505055223254083935929242819402489217649917883179210746178400097792158671210163533955792155591715924514481092161279
4746886225884480178023392423954176703443028466859352020541272471684504893759102626118712783805212521816098036430270567072841
In [11]: (b-a)%n
Out[11]: 7841802387909561814939930409124114437189639488404364381841195358408505055223254083935929242819402489217649917883179210746178400097792158671210163533955792155591715924514481092161279
4746886225884480178023392423954176703443028466859352020541272471684504893759102626118712783805212521816098036430270567072841
In [12]: (A*B).base10()
Out[12]: 348465320496295972386419674566373315140185459890058778445515028943141482207427884341219831855761721944120130470275415537018929042473804565481858196348257760371801987759566994475369
241981122497924995601582561932556163286938850449306726484824839798280876967052568331486914645950871551267475879181557439288800
In [13]: (a*b)%n
Out[13]: 348465320496295972386419674566373315140185459890058778445515028943141482207427884341219831855761721944120130470275415537018929042473804565481858196348257760371801987759566994475369
241981122497924995601582561932556163286938850449306726484824839798280876967052568331486914645950871551267475879181557439288800
```

Проведемо тести, запишемо час, і запустимо профайлер.

```
gratigo@dedsec:~/Documents/term5/SROM/lab1/prikhodko_fb12_lab1$ ./modte
[*] Checking addition...
Abn + Bbn == Bbn + Abn: True
A + B == Abn + Bbn: True
(A + B) + C == Abn + (Bbn + Cbn): True
[!] Addition seems right checking subtraction...
Abn - Bbn == Bbn - Abn: True
A - B == Abn - Bbn: True
[!] Subtraction seems right
[*] Checking multiplication...
Abn * Bbn == Bbn * Abn: True
A * B == Abn * Bbn: True
(A * B) * C == Abn * (Bbn * Cbn): True
(Abn+Bbn)*Cbn == Abn*Cbn + Bbn*Cbn: True
[!] Multiplication seems right
[*] Checking power...
base**Bbn == base**B: False
a^phi(n) == 1 mod n: 1
[!] Seems right
[*] Starting time tests
Average addition time: 0.000000128015 seconds
Average subtraction time: 0.000000148797 seconds
Average multiplication time: 0.000003274895 seconds
Average powering time: 0.000006112500 seconds
```

Додавання

234749 function calls (229661 primitive calls) in 0.082 seconds					
Ordered by: standard name					
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.082	0.082	<string>:1(<module>)
22912	0.018	0.000	0.035	0.000	bignum.py:12(__init__)
1	0.000	0.000	0.000	0.000	bignum.py:121(__ge__)
1	0.000	0.000	0.000	0.000	bignum.py:139(__lt__)
1698	0.001	0.000	0.001	0.000	bignum.py:158(mulStep)
2546/2	0.004	0.000	0.082	0.041	bignum.py:171(__mul__)
2546/2	0.009	0.000	0.082	0.041	bignum.py:270(karatSubaStep)
5936	0.003	0.000	0.003	0.000	bignum.py:28(base10)
3394	0.001	0.000	0.001	0.000	bignum.py:310(shiftLeft)
1	0.000	0.000	0.082	0.082	bignum.py:402(barrettReduction)
2	0.000	0.000	0.000	0.000	bignum.py:423(__init__)
1	0.000	0.000	0.082	0.082	bignum.py:427(__add__)
6787	0.018	0.000	0.033	0.000	bignum.py:57(__add__)
1699	0.004	0.000	0.006	0.000	bignum.py:82(sub_s)
1697	0.001	0.000	0.010	0.000	bignum.py:98(__sub__)
22912	0.012	0.000	0.015	0.000	conv_types.py:15(convert)
2547	0.001	0.000	0.001	0.000	conv_types.py:3(getDigits)
1	0.000	0.000	0.082	0.082	modprof_tests.py:10(add)
1	0.000	0.000	0.082	0.082	{built-in method builtins.exec}
43278	0.003	0.000	0.003	0.000	{built-in method builtins.isinstance}
74826	0.004	0.000	0.004	0.000	{built-in method builtins.len}
11880	0.002	0.000	0.002	0.000	{built-in method builtins.max}
30017	0.002	0.000	0.002	0.000	{method 'append' of 'list' objects}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
64	0.000	0.000	0.000	0.000	{method 'pop' of 'list' objects}

Віднімання:

4668355 function calls (4566370 primitive calls) in 1.618 seconds					
Ordered by: standard name					
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	1.618	1.618	<string>:1(<module>)
451326	0.364	0.000	0.696	0.000	bignum.py:12(__init__)
3	0.000	0.000	0.001	0.000	bignum.py:121(__ge__)
3	0.000	0.000	0.000	0.000	bignum.py:139(__lt__)
33434	0.024	0.000	0.028	0.000	bignum.py:158(mulStep)
50148/6	0.075	0.000	1.616	0.269	bignum.py:171(__mul__)
50148/6	0.170	0.000	1.616	0.269	bignum.py:270(karatSubaStep)
116998	0.051	0.000	0.051	0.000	bignum.py:28(base10)
66862	0.015	0.000	0.015	0.000	bignum.py:310(shiftLeft)
3/2	0.000	0.000	1.618	0.809	bignum.py:402(barrettReduction)
6	0.000	0.000	0.000	0.000	bignum.py:423(__init__)
3/1	0.000	0.000	1.618	1.618	bignum.py:431(__sub__)
133720	0.350	0.000	0.653	0.000	bignum.py:57(__add__)
33437	0.085	0.000	0.111	0.000	bignum.py:82(sub_s)
33433/31735	0.021	0.000	0.261	0.000	bignum.py:98(__sub__)
451326	0.231	0.000	0.303	0.000	conv_types.py:15(convert)
50151	0.009	0.000	0.010	0.000	conv_types.py:3(getDigits)
1	0.000	0.000	1.618	1.618	modprof_tests.py:12(sub)
1	0.000	0.000	1.618	1.618	{built-in method builtins.exec}
852504	0.062	0.000	0.062	0.000	{built-in method builtins.isinstance}
1510188	0.087	0.000	0.087	0.000	{built-in method builtins.len}
234019	0.036	0.000	0.036	0.000	{built-in method builtins.max}
600447	0.037	0.000	0.037	0.000	{method 'append' of 'list' objects}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
192	0.000	0.000	0.000	0.000	{method 'pop' of 'list' objects}

Множення

379513 function calls (371299 primitive calls) in 0.136 seconds					
Ordered by: standard name					
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.136	0.136	<string>:1(<module>)
36983	0.030	0.000	0.058	0.000	bignum.py:12(__init__)
1	0.000	0.000	0.000	0.000	bignum.py:121(__ge__)
1	0.000	0.000	0.000	0.000	bignum.py:139(__lt__)
2741	0.002	0.000	0.002	0.000	bignum.py:158(mulStep)
4110/3	0.007	0.000	0.136	0.045	bignum.py:171(__mul__)
4110/3	0.015	0.000	0.136	0.045	bignum.py:270(karatSubaStep)
9583	0.005	0.000	0.005	0.000	bignum.py:28(base10)
5479	0.001	0.000	0.001	0.000	bignum.py:310(shiftLeft)
1	0.000	0.000	0.096	0.096	bignum.py:402(barrettReduction)
2	0.000	0.000	0.000	0.000	bignum.py:423(__init__)
1	0.000	0.000	0.136	0.136	bignum.py:438(__mul__)
10955	0.029	0.000	0.053	0.000	bignum.py:57(__add__)
2741	0.008	0.000	0.010	0.000	bignum.py:82(sub_s)
2739	0.002	0.000	0.016	0.000	bignum.py:98(__sub__)
36983	0.019	0.000	0.026	0.000	conv_types.py:15(convert)
4111	0.002	0.000	0.002	0.000	conv_types.py:3(getDigits)
1	0.000	0.000	0.136	0.136	modprof_tests.py:14(mul)
1	0.000	0.000	0.136	0.136	{built-in method builtins.exec}
69856	0.005	0.000	0.005	0.000	{built-in method builtins.isinstance}
119551	0.007	0.000	0.007	0.000	{built-in method builtins.len}
19175	0.003	0.000	0.003	0.000	{built-in method builtins.max}
50322	0.003	0.000	0.003	0.000	{method 'append' of 'list' objects}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
64	0.000	0.000	0.000	0.000	{method 'pop' of 'list' objects}

Степінь:

164172 function calls (160674 primitive calls) in 0.057 seconds					
Ordered by: standard name					
ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.057	0.057	<string>:1(<module>)
15788	0.013	0.000	0.024	0.000	bignum.py:12(__init__)
3	0.000	0.000	0.000	0.000	bignum.py:146(__eq__)
3	0.000	0.000	0.000	0.000	bignum.py:147(<listcomp>)
3	0.000	0.000	0.000	0.000	bignum.py:148(<listcomp>)
1174	0.001	0.000	0.001	0.000	bignum.py:158(mulStep)
1757/8	0.003	0.000	0.057	0.007	bignum.py:171(__mul__)
1	0.000	0.000	0.057	0.057	bignum.py:199(__pow__)
1757/8	0.006	0.000	0.057	0.007	bignum.py:270(karatSubaStep)
4081	0.002	0.000	0.002	0.000	bignum.py:28(base10)
2340	0.001	0.000	0.001	0.000	bignum.py:310(shiftLeft)
1	0.000	0.000	0.000	0.000	bignum.py:34(baseN)
4672	0.012	0.000	0.023	0.000	bignum.py:57(__add__)
1166	0.003	0.000	0.004	0.000	bignum.py:82(sub_s)
1166	0.001	0.000	0.006	0.000	bignum.py:98(__sub__)
15788	0.008	0.000	0.011	0.000	conv_types.py:15(convert)
1762	0.000	0.000	0.000	0.000	conv_types.py:3(getDigits)
1	0.000	0.000	0.057	0.057	modprof_tests.py:16(pow)
1	0.000	0.000	0.057	0.057	{built-in method builtins.exec}
29814	0.002	0.000	0.002	0.000	{built-in method builtins.isinstance}
53479	0.003	0.000	0.003	0.000	{built-in method builtins.len}
8178	0.001	0.000	0.001	0.000	{built-in method builtins.max}
1	0.000	0.000	0.000	0.000	{built-in method math.log}
21232	0.001	0.000	0.001	0.000	{method 'append' of 'list' objects}
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}
1	0.000	0.000	0.000	0.000	{method 'find' of 'str' objects}
1	0.000	0.000	0.000	0.000	{method 'rstrip' of 'str' objects}