СПЕЦІАЛЬНІ РОЗДІЛИ ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №1

Багаторозрядна арифметика

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.

Теоретичні відомості

Багаторозрядна арифметика використовується для реалізації багатьох криптографічних примітивів, зокрема в алгоритмі RSA, для реалізації скінченних полів та еліптичних кривих. На сьогоднішній день в реальних системах використовуються числа довжиною тисяча біт та більше. Обчислення можуть здійснюватися як персональними комп'ютерами, так і різноманітними портативними пристроями (смарт-картами, телефонами тощо), тому суттєвим є ефективна за швидкодією (а часто – й за пам'яттю) реалізація багаторозрядної арифметики. На ПК така арифметика реалізується, як правило, в системі числення, основа якої відповідає довжині машинного слова процесора (в 32-х бітній системі, це дає, наприклад, 16 «цифр» для 512-бітної арифметики), що сильно підвищує швидкодію.

Операції додавання та віднімання реалізуються, як правило, звичайним методом «у стовпчик», який не викликає труднощів. Множення для невеликої (~10) кількості цифр може бути ефективно реалізоване за допомогою звичайного алгоритму «в стовпчик», також можливе використання інших алгоритмів, наприклад, алгоритму Карацуби. Множення за модулем та піднесення до степеня за модулем можна виконувати за допомогою множення та ділення (тут корисною буде формула:

$$a^n \pmod{m} = \prod_{i=0}^{[\log_2 n]} (a^{b_i 2^i} \bmod m),$$

де b_i - i-тий біт у двійковому розкладі n (зауважте: модуль не співпадає зі степенем). Ця формула дає змогу підносити до степеня n не більш ніж за $2[\log_2 n]$ множень, бо має місце рекурентне співвідношення $a^{2^i} \bmod m = (a^{2^{i-1}} \bmod m)^2 \bmod m$). Також існують ефективні спеціальні алгоритми багаторозрядної модулярної арифметики, такі як алгоритм Монтгомері та Барета.

Додаткові теоретичні відомості викладаються на лекціях курсу «Спецрозділи обчислювальної математики», їх також можна знайти у літературі, список якої наведений в кінці.

Вітається творчий підхід студентів до роботи, удосконалення існуючих та створення власних алгоритмів.

Завдання до комп'ютерного практикуму

А) Згідно варіанту розробити клас (чи бібліотеку функцій) для роботи з m-бітними цілими числами.

Повинні бути реалізовані такі операції:

1) переведення малих констант у формат великого числа (зокрема, 0 та 1)**;**
2) додавання чисел;
3) віднімання чисел;
4) множення чисел, піднесення чисел до квадрату;
5) ділення чисел;
6) знаходження остачі від ділення чисел (лишків) методом класичного ділення та методом Баррета;
7) множення чисел по модулю n;
8) піднесення до квадрату по модулю n;
9) піднесення числа до багаторозрядного степеня d по модулю n;
10) конвертування (переведення) числа в символьну строку та обернене перетворення символьної строки у число; обов'язкова підтримка шістнадцяткового представлення, бажана – десяткового та двійкового.

Бажано реалізувати такі операції:

1) визначення номеру старшого ненульового біта числа;
2) бітові зсуви (вправо та вліво);
3) обчислення НСД та НСК двох чисел;
4) знаходження числа, оберненого до даного по модулю простого p.

Мова програмування, семантика функцій та спосіб реалізації можуть обиратись довільним чином.

ВАРІАНТИ ЗАВДАНЬ

Номер варіанта	**Довжина чисел в бітах**	**Додаткове завдання**
1	256	Реалізація операції піднесення до квадрату (звичайного та модулярного) більш ефективним методом, ніж звичайним множенням; порівняння ефективності.
2	384	Реалізація ділення класичним алгоритмом «у стовпчик» (див. [1], [2]), порівняння ефективності із алгоритмом «зсувай@віднімай»
3	512	Реалізація модулярного множення алгоритмом Блеклі (див. [5]), порівняння ефективності із звичайним модулярним множенням.
4	768	Реалізація модулярного піднесення до степеню віконним методом (2-8 біт), порівняння ефективності із методом Горнера
5	1024	Реалізація множення методом Карацуби, порівняння ефективності із звичайним множенням
6	256	Реалізація модулярних операцій алгоритмами Монтгомері (для парних та непарних модулів), порівняння ефективності
7	384	Обчислення НСД алгоритмом Евкліда та бінарним алгоритмом, порівняння їх ефективності; обчислення НСК
8	512	Розв'язування простих систем порівнянь за китайською теоремою про лишки.

9	768	Обчислення символів Якобі, оцінка трудомісткості алгоритму в залежності від довжини чисел.
10	1024	Реалізація тесту ASK, оцінка швидкодії тесту
11	256	Реалізація тесту Соловея-Штрассена, оцінка швидкодії тесту
12	384	Реалізація тесту Міллера-Рабіна, оцінка швидкодії тесту
13	512	Обчислення звичайних квадратних (та, за бажанням кубічних) коренів.
14	768	Обчислення квадратних коренів за простим модулем
15	1024	Реалізація алгоритму Сільвера-Поліга-Хеллмана

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

1. $(a+b)\cdot c = c\cdot(a+b) = a\cdot c + b\cdot c$;
2. $n\cdot a = \underbrace{a+a+\ldots+a}_{n}$, де n повинно бути не менш за 100;
3. $a^{\varphi(n)} = 1 \bmod n$ (за умови $\gcd(a,n)=1$).

Перевірити останню тотожність для простого n; перевірити для $n = 3^k$ та довільного $a \vdots 3$: $\varphi(3^k) = 3^k - 3^{k-1} = 2\cdot 3^{k-1}$, отже, має виконуватись $a^{\varphi(3^k)} = a^{2\cdot 3^{k-1}} = 1 \bmod 3^k$.

Продумати та реалізувати свої тести на коректність.

Для перевірки роботи операцій із простими числами можна використовувати заздалегідь відомі прості числа (наприклад, числа Мерсенна).

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.

Продемонструвати працюючу програму викладачеві!
(бажано компілювати на місці, щоб була можливість змінювати програму)

Технічні зауваження

Наступні зауваження та рекомендації можуть виявитись корисними під час виконання комп'ютерного практикуму.

– Числа рекомендується реалізовувати беззнаковими;

– довжина числа повинна легко параметризуватися, тобто програма повинна легко модифікуватися для роботи з числами іншої довжини;

– незалежно від внутрішньої реалізації числа (наприклад, масив 32-х бітних слів фіксованої чи змінної довжини, або масив байтів, що містять двійкові цифри) зовнішній користувач повинен бачити m-бітне число у природному (символьному) вигляді; обов'язково реалізувати представлення чисел у шістнадцятковому вигляді, за бажанням – у десятковому або іншій системі числення.

– рекомендована (але не обов'язкова) внутрішня реалізація – масив 32-х бітних слів (64-бітних для відповідних архітектур) фіксованої довжини; це досить просто та ефективно по швидкодії та пам'яті (взагалі, чим більший розмір машинного слова, тим краще);

– реалізація чисел зі змінною внутрішньою довжиною не рекомендується (потрібно обґрунтувати, що це дасть суттєвий виграш у швидкодії);

– реалізації типу «одна комірка масиву = один біт числа» є дуже неефективними по швидкодії;

– якщо при виконанні немодулярного додавання (множення) результат перевищує допустиму довжину числа, можна обрубати результат до потрібної довжини; також можна реалізувати підтримку чисел подвійної довжини або якийсь інший варіант; вибір за вами;

– модулярне множення m-бітного числа на m-бітне число по модулю m-бітного числа повинно виконуватись арифметично правильно (тобто без втрати розрядів проміжних результатів, якщо алгоритм передбачає такі проміжні результати);

– час можна вимірювати як за допомогою власних тестів із заміром часу, так і за допомогою profiler'a – спеціального інструменту, який призначений, зокрема, і для визначення часової ефективності програми; профайлери вбудовані в більшість середовищ розробки;

– для підвищення точності вимірювання треба викликати піддослідну функцію у циклі декілька (сто, тисяча, мільйон) разів;

– оптимізувати слід лише ті функції, які забирають суттєвий відсоток часу;

– критичні за часом ділянки можна реалізувати на асемблері (за допомогою асемблерних вставок або як окремі асемблерні функції); також слід проконтролювати, щоб в опціях компілятора була встановлена оптимізація по швидкості (без неї час роботи може збільшитися в декілька разів, це особливо важливо, коли не використовується асемблер).

Якщо Ви бажаєте написати програму на Java – будь-ласка, але клас BigInteger дозволяється використовувати лише таким чином:

1) перевірити коректність роботи свого класу;

2) щоб показати, в скільки разів Ваша реалізація швидша за стандартну.

Те ж саме стосується будь-яких інших вбудованих засобів, реалізованих бібліотек багаторозрядної арифметики та відповідних open source проектів (для будь-якої мови програмування).

PHP, Javascript та інші інтерпретаторні мови п'ятого покоління, функціональні мови програмування, мова Brainfuck тощо не допускаються через вкрай низьку швидкодію.

Оформлення звіту

Звіт оформлюється відповідно до стандартних правил оформлення наукових робіт. Звіт має містити:

1) мету, теоретичну частину та завдання, наведені вище, згідно варіанту;
2) стисло викладені теоретичні відомості по додатковому завданню;
3) бітові/символьні зображення тестових чисел та результатів операцій над ними;
4) бітові/символьні зображення результатів контролю за пунктом Б);
5) аналітичні відомості за пунктом В) завдання;
6) текст програми (у додатку)

Також текст програми передається викладачеві в електронному вигляді.

Оцінювання лабораторної роботи

За виконання лабораторної роботи студент може одержати до 15 рейтингових балів; зокрема, оцінюються такі позиції:

- реалізація основного завдання – до 6-ти балів;
- реалізація додаткового завдання – до 3-х балів;

- виконання аналітичного завдання (за пунктом В) – до 2-х балів;
- оформлення звіту – 2 бали;
- своєчасне виконання основного завдання – 1 бал;
- своєчасне виконання додаткового завдання – 1 бал;
- несвоєчасне виконання роботи – (-1) бал за кожні два тижні пропуску.

Контрольні питання

1. Які Ви знаєте алгоритми множення багаторозрядних чисел? Яка їх обчислювальна складність?
2. Які Ви знаєте алгоритми модулярного множення багаторозрядних чисел? Яка їх обчислювальна складність?
3. В скільки разів множення чисел з 64-бітними цифрами швидше за множення чисел з 1-бітними цифрами (при використанні 64-бітних регістрів)? Чому?
4. В скільки разів піднесення до степеня чисел з 64-бітними цифрами швидше за піднесення до степеня чисел з 1-бітними цифрами (при використанні 64-бітних регістрів)? Чому?

Література

1. *Д. Кнут.* Искусство программирования на ЭВМ. Т.2. – М.:Мир, 1976. –724 с.
2. *О.Н. Василенко.* Теоретико-числове алгоритмы в криптографии
3. *А.А. Болотов.* Алгоритмические основы современной криптологии
4. *А.В. Анісімов.* Алгоритмічна теорія великих чисел. Модулярна арифметика великих чисел. – К.: Академкнига, 2001.
5. *Çetin Kaya Koç*. High-Speed RSA Implementation.