References

1. Bonnans, J.F. Perturbations analysis of optimization problems / J.F. Bonnans, A. Shapiro. – New York: Springer-Verlag, 2000.
2. Luderer, B. Multivalued analysis and nonlinear programming problems with perturbations / B. Luderer, L. Minchenko, T. Satsura. – Dordrecht: Kluwer Acad. Publ., 2002. – 222 p.
3. Minchenko, L.I. Parametric nonlinear programming problems under relaxed constant rank regularity condition / L. Minchenko, S. Stakhovski // SIAM Journal on Optimization. – 2011. – Vol. 21. – N 1.

УДК 511.1

ПРОСТЫЕ ЧИСЛА МЕРСЕННА

Аксамит М.В., Былинович В.Н.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск
Научный руководитель: Стройникова Е.Д.

Основная теорема арифметики гласит, что всякое натуральное число $n > 1$ можно представить в виде произведения простых множителей:

$$n = p_1 * p_2 * \dots * p_m,$$

где $p_1, p_2, \dots, p_m$ – простые числа.

Разложение натурального числа на составляющие его простые множители называется факторизацией числа. В настоящий момент неизвестен такой алгоритм факторизации, который мог бы разложить любое большое число на простые множители за полиномиальное время. Благодаря этому простые числа нашли широкое применение в криптографии. Например, RSA: для того, чтобы взломать данный шифр, нужно разложить большое число n, известное по открытому ключу, на простые множители, которых всего два. В том же алгоритме RSA для генерации ключей требуется найти большие простые числа, что на сегодняшний день гораздо проще факторизации.

Небольшие простые числа из начала списка простых можно получить с помощью таких несложных алгоритмов, как решето Эратосфена, Сундарама или Аткина. Но на практике, как правило, нужны числа более высоких порядков. Тут на помощь приходят тесты на простоту – алгоритмы, проверяющие, является ли число простым. Но само число сначала нужно сгенерировать.

Существует ряд чисел специального вида, простоту которых можно доказать эффективными алгоритмами и за полиномиальное время:

Числа Мерсенна – самые большие из известных простых чисел, очень распространены благодаря существованию эффективного теста на простоту Люка–Лемера, однако бесконечность таких чисел до сих пор не доказана. Их можно представить в виде

$$M_p = 2^p - 1,$$

где p – простое число. Замечание: простоты числа p недостаточно для доказательства простоты числа Мерсенна. Для этого существует специальный алгоритм: тест Люка – Лемера для чисел Мерсенна. Тест основывается на том, что простота числа M_p влечет за собой простоту числа p. Пусть p – простое число, большее либо равное трем. Зададим последовательность:

$$L_0 = 4,$$
$$L_{n+1} = L_n^2 + 2.$$

Тогда M_p простое тогда и только тогда, когда $L_{p-2} \equiv 0 \ (mod\ M_p)$.

При реализации теста вычисляют не сами значения $L_0, L_1, \dots, L_k$, длина которых растет по экспоненте, а только их остатки от деления на M_p.

Вычислительная сложность теста равна $O(q^3)$, так как производится $O(q)$ возведений в квадрат и делений по модулю, в то время как самый простой алгоритм умножения имеет сложность $O(q^2)$.

25 января 2013 г. математик Кертис Купер (США) в рамках проекта распределенных вычислений GIMPS с помощью теста Люка–Лемера нашел 48-е простое число Мерсенна: $M_{48} - 2^{57885161} - 1$. Оно состоит из 17424170 десятичных цифр. Оно же на сегодняшний день является самым большим известным простым числом.

Авторами доклада была разработана учебная программа на языке С, находящая простые числа Мересенна. Она генерирует числа Мерсенна и проверяет их на простоту, пока не находит простое. При реализации программы была использована длинная арифметика, т. к. нужные нам числа превышают разрядность регистров процессора: длинное число представляется в виде одномерного массива из его десятичных цифр. Для всех арифметических операций были написаны отдельные функции: вычисление сложения, вычитания и умножения «столбиком», а деления – «уголком». Также был использован рекурсивный алгоритм возведения в степень по модулю.

Вот некоторые примеры сгенерированных простых чисел: $2^{107} - 1 =$ =162259276829213363391578010288127 (время выполнения: 21,801 сек.), $2^{607} - 1 =$ 531137992816767098689588206552468627329593117727031923199444138200403559860852242739162502265229285668889329486246501015346579337652707239409519978766587351943831270835393219031728127 (время выполнения: 193,305 сек.).

Данная программа наглядно демонстрирует работу алгоритмов длинной арифметики и теста Люка–Лемера, т.к. используются всем известные алгоритмы для арифметических операций и числа в привычном десятичном представлении. Существует способ немного ускорить программу: использовать числа в системе счисления по большему основанию, чем 10. Это позволит сократить длину массива, что уменьшит количество вычислений.

На рисунке 1 приведена гистограмма сравнения времени выполнения программы (в сек.) и степени, в которую мы возводим число 2 при генерации числа Мерсенна.

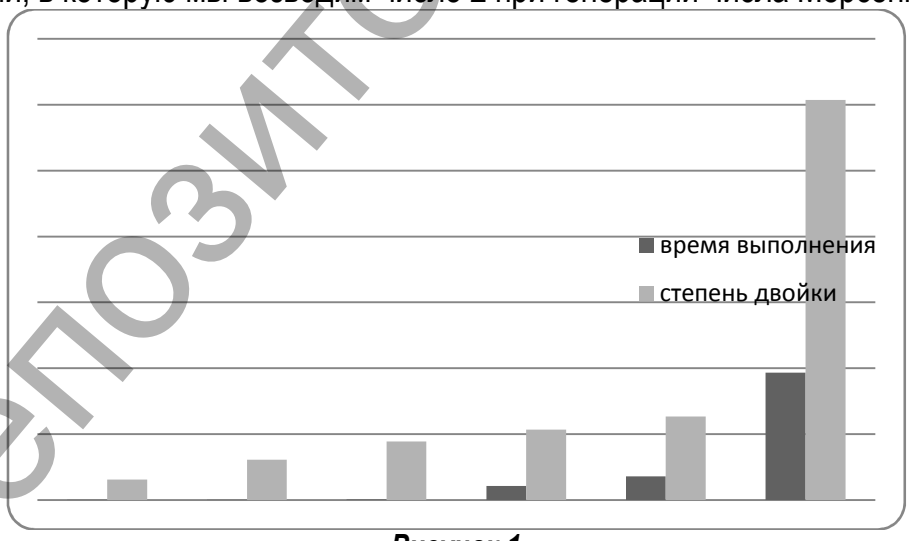

Рисунок 1

Из рисунка 1 мы видим, что время выполнения вычислений растет быстрее, чем искомая степень. Но для поиска чисел Мерсенна до шестисотой степени скорости выполнения хватает.

Список цитированных источников

1. Крэндалл, Р. Простые числа. Криптографические и вычислительные аспекты: научная и учебная литература / Р. Крэндалл, К. Померанс. – Москва, 2011. – 664 с.