ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

УДК 519.71

О СЛОЖНОСТИ ПОСТРОЕНИЯ ТАБЛИЦЫ ПРОСТЫХ ЧИСЕЛ НА МАШИНЕ ТЬЮРИНГА[1]

И. С. Сергеев

ФГУП «НИИ «Квант», г. Москва, Россия

Доказывается, что сложность построения таблицы простых чисел от 1 до n на многоленточной машине Тьюринга равна $\mathrm{O}(n \log n)$.

Ключевые слова: *машина Тьюринга, сложность, просеивание.*

DOI 10.17223/20710410/31/8

ON THE COMPLEXITY OF COMPUTING PRIME TABLES ON THE TURING MACHINE

I. S. Sergeev

Technology Federal State Unitary Enterprise "Research Institute Kvant", Moscow, Russia

E-mail: isserg@gmail.com

It is proved that the complexity of computing the table of primes up to n on a multitape Turing machine is $\mathrm{O}(n \log n)$.

Keywords: *Turing machine, complexity, sieving.*

Введение

Составление таблицы простых чисел является необходимым предварительным этапом в современных алгоритмах факторизации [1]; среди других известных приложений можно указать вычисление факториалов [2, § 6.5].

Для теоретической оценки сложности обычно используется вычислительная модель log-RAM (т. е. RAM-программа, выполняющая арифметические действия с аргументами длины $\mathrm{O}(\log n)$ за время $\mathrm{O}(1)$, где n — размер входа, [3, 4]), адекватная предполагаемой реализации на стандартных компьютерах. Лучшая известная оценка сложности построения таблицы простых, не превосходящих n, в этой модели составляет $\mathrm{O}(n/\log\log n)$ [5] (см. также [1, § 3.2]).

Модель многоленточной машины Тьюринга (ММТ) служит универсальным средством анализа вычислительных алгоритмов, вне предположения об определённом способе реализации.

С понятием ММТ можно познакомиться в [2, 3]. Неформально говоря, ММТ состоит из нескольких (т. е. $\mathrm{O}(1)$) потенциально бесконечных лент двоичных символов

[1]Работа поддержана грантом РФФИ № 14-01-00671а.

с подвижными указателями и процессора, выполняющего элементарные инструкции. Процессор использует ленты в качестве памяти. К инструкциям относятся: сдвиг указателя на одну позицию влево или вправо, чтение символа из указанной ячейки, запись символа в указанную ячейку, бинарная булева операция над двумя прочитанными символами, условная операция (выбор одного из двух действий в зависимости от значения прочитанного символа). Сложность измеряется числом инструкций.

Замечание 1. На самом деле, функциональность машины Тьюринга как абстрактной модели вычислений шире. Приведённое описание апеллирует к осуществлённым реализациям машины Тьюринга, например в [2].

А. Шёнхаге [2] адаптировал решето Эратосфена и при помощи адаптированного алгоритма оценил сложность $P(n)$ построения таблицы простых как $\mathrm{O}(n \log^2 n \log\log n)$ с замечанием, что оценка может быть улучшена при более аккуратном анализе алгоритма. Для цели работы [2] — вычисления $n!$ со сложностью $\mathrm{O}(M(n \log n))$ — такой оценки было достаточно, пока для сложности $M(n)$ умножения n-разрядных чисел не были получены оценки вида $M(n) = \mathrm{o}(n \log n \log\log n)$.

Авторы работы [6] переложили на машины Тьюринга метод [7], основанный на специальном «квадратичном» решете, и доказали оценку $\mathrm{O}(n \log^2 n / \log\log n)$.

В действительности аккуратный анализ позволяет для метода [2] получить оценку $\mathrm{O}(n \log^2 n)$ — такую же, как для простой версии метода [6]. А приём исключения повторного учёта чисел с малыми простыми делителями, предложенный П. Притчардом [5] (см. также [1, § 3.2]), позволяет понизить сложность до $\mathrm{O}(n \log^2 n / \log\log n)$ — как и в методе [6].

В работе [6] предложен ещё один метод с оценкой сложности $P(n) = \mathrm{O}(M(n \log n))$. Метод может давать преимущество в предположении $M(n) = \mathrm{o}(n \log n)$ (но справедливость такого предположения сегодня представляется маловероятной).

В настоящей работе устанавливается оценка $P(n) = \mathrm{O}(n \log n)$. Из неё, в частности, следует, что сложность построения таблицы составных чисел от 1 до n равна $\Theta(n \log n)$. Ещё один вывод: построение таблицы простых не влияет на порядок сложности вычисления $n!$, который очевидно (из размерности задачи) не меньше $n \log n$.

1. Общая схема вычислений

Решето Эратосфена и решето [7] в методах [2, 6] приводят к завышенным оценкам сложности вследствие многократного опробования одних и тех же составных чисел. В этом отношении, следуя [5] (см. также [1, § 3.2]), выгодно взять за основу неизбыточный способ перечисления составных, предложенный Мэрсоном [8].

Будем обозначать через p_i последовательные простые числа: $p_1 = 2$, $p_2 = 3$ и т. д. Процедура Мэрсона позволяет выписать все составные числа в заданном интервале, избегая повторений, а именно: для каждого i строится цепочка C_i чисел $C_{i,j} = k_j \cdot p_i$, не кратных простым числам, меньшим p_i. Множители k_j определяются из результатов предшествующего просеивания условием $k_j \notin \bigcup\limits_{i'<i} C_{i'}$.

По существу, предлагаемый далее алгоритм является адаптацией алгоритма из [1, § 3.2], основанного на идеях [5, 8].

В n ячейках одной из лент ММТ мы формируем признаки простоты чисел от 1 до n. В начальный момент времени лента заполнена нулями, в конце алгоритма ноль в позиции k означает, что число k простое, единица — что составное.

Последовательно увеличивая индекс i, выполняем проход ленты, проставляя единицы в позициях $C_{i,j}$. Поскольку просмотр всей ленты требует $\mathrm{O}(n)$ операций, то с ро-

стом i мы вынуждены просматривать всё более короткие начальные отрезки ленты, длина которых тем не менее позволяет корректно сформировать множество множителей k_j для следующего шага.

По мере вычислений цепочки (упорядоченные списки) C_i записываются на другие ленты ММТ. Далее выполняется стандартная процедура попарного слияния цепочек в сортированные списки до тех пор, пока общее число списков не станет меньше $\log n$. Чтобы уменьшить стоимость сравнений при сортировке, списки хранятся и обрабатываются в сжатом виде. В конце концов, каждый список отображается на ленту признаков простоты.

2. Основной результат

В рассуждениях ниже используются стандартные факты, связанные с распределением простых чисел — их можно найти, например, в [9, 10]. Как обычно, функция $\pi(n)$ обозначает число простых чисел, не превосходящих n.

Теорема 1. Сложность построения таблицы простых чисел от 1 до n равна $\mathrm{O}(n \log n)$.

Доказательство. Сделаем несколько предварительных замечаний.

При движении указателя по ленте обычно требуется обновлять номер позиции указателя. Поэтому напомним, что прибавление или вычитание единицы в среднем выполняется за $\mathrm{O}(1)$ битовых операций: в половине случаев изменяется только младший разряд, в $1/4$ случаев — два младших разряда, в $1/8$ случаев — три разряда и т. д.

При движении к позиции с заданным номером попутно требуется выполнять сравнение текущего номера с заданным. Такое сравнение в среднем также выполняется со сложностью $\mathrm{O}(1)$: в половине случаев достаточно выполнить сравнение младших разрядов (если они отличаются), в $1/4$ случаев — сравнить два разряда и т. д.

Перейдём к изложению собственно алгоритма.

0) На ленте T в конце работы программы будут сформированы 1-битные признаки $a(k)$ простоты чисел k. При первом заполнении ленты $a(2) = \ldots = a(n) = 0$.

1) Для каждого простого числа $p_i \leqslant \log n$ выполняем проход ленты T и запись $a(kp_i) = 1$, $k > 1$. При этом само число p_i определяется как позиция i-го нулевого бита. При движении по ленте отслеживаем значение номера позиции по модулю p_i, т. е. при сдвиге указателя инкрементируем текущий номер и сравниваем его с p_i. Если получаем равенство, то записываем единицу в ячейку и обнуляем номер. Сложность этого шага равна $\mathrm{O}(n \cdot \pi(\log n)) = \mathrm{O}(n \log n / \log\log n)$.

2) Для каждого из последующих простых чисел p_i, $\log n < p_i \leqslant \sqrt{n}$, просматриваем только первые n/p_i позиций ленты. Номером i-й нулевой позиции определяется само число p_i. Продолжая движение по ленте, выполняем два действия. Во-первых, переписываем на другую ленту последовательность C_i составных чисел kp_i, не кратных простым числам, меньшим p_i. В качестве критерия принадлежности числа последовательности используется условие $a(k) = 0$, где $k \geqslant p_i$. Во-вторых, записываем единицы в позиции с номерами, кратными p_i.

При записи цепочки составных чисел применяем сжатие. Для этого разобьём простые числа на группы: к j-й группе отнесём числа $p \in [2^j, 2^{j+1})$. Цепочку составных, относящихся к простому числу из j-й группы, разбиваем (с точностью до округления) на $n2^{-2j}$ секций: в одной секции располагаются числа, отличающиеся лишь младшими $2j$ разрядами. (Здесь и ниже опускаем округления в формулах, если они не влияют на результаты, в целях наглядности изложения.) Старшие $\log n - 2j$ разрядов числа определяются номером секции. Поэтому для записи очередного числа цепочки мы ис-

пользуем $2j$ младших битов, размещая его в подходящей секции. Технически при записи последовательности можно использовать разделяющие биты: единицами отделять секции, нулями — числа внутри секции.

В цепочке C_i содержится по порядку не более

$$\frac{n}{p_i}\prod_{j=1}^{i-1}\left(1-\frac{1}{p_j}\right)\asymp\frac{n}{p_i\log p_i}$$

чисел (равенство выполнено согласно известному результату Чебышёва — Мертенса); знак $\asymp$ обозначает равенство по порядку. Вычисления выполняются так: при просмотре ленты T фиксируются разности между позициями соседних нулей. По достижении позиции с нулём накопленная разность s умножается на p_i. Такое умножение методом сдвигов и сложений легко выполнить за $\mathrm{O}(\log p_i\cdot\log s)$ операций, если знать длину числа s (длину s удобно фиксировать на отдельной ленте). Сумма логарифмов t величин, суммарно не превосходящих th, не превосходит $t\log h$ (сумма максимальна, когда все величины приблизительно равны). Поэтому сложность умножений можно оценить сверху, подставив вместо s среднее значение разности, равное по порядку $\log p_i$. Получаем оценку сложности порядка $\mathrm{O}(n\log\log p_i/p_i)$.

Наконец, сложность построения нового члена цепочки, т. е. прибавления найденной разности к предыдущему члену и добавления нового члена в цепочку, можно (грубо) оценить как $\mathrm{O}(\log n)$.

Суммируя по всем индексам i, сложность этапа можно оценить сверху величиной порядка

$$n\sum_{i=\pi(\log n)+1}^{\pi(\sqrt{n})}\frac{\log\log p_i}{p_i}+n\log n\sum_{i=\pi(\log n)+1}^{\pi(\sqrt{n})}\frac{1}{p_i\log p_i}\asymp n\log n/\log\log n.$$

3) В результате предыдущего шага построено около $\pi(\sqrt{n})$ упорядоченных списков составных чисел, включающих в совокупности порядка

$$n\sum_{i=\pi(\log n)+1}^{\pi(\sqrt{n})}\frac{1}{p_i\log p_i}\asymp n/\log\log n$$

членов. Эти списки отнесены к примерно $(1/2)\log n$ группам (см. выше) — для списков одной группы используется одна и та же формула сжатия. Выполним сортировку внутри каждой группы. По построению, j-я группа содержит порядка $2^j/j$ списков и порядка

$$n\sum_{i=\pi(2^j)+1}^{\pi(2^{j+1})}\frac{1}{p_i\log p_i}\asymp n/j^2$$

чисел. Деревом слияний [11, § 5.4] эти списки упорядочиваются за $\mathrm{O}(n/j)$ операций сравнения и перезаписи, каждая эквивалентна $\mathrm{O}(j)$ битовым операциям. (Сортировка s списков с суммарным числом N элементов выполняется за $\mathrm{O}(N\log s)$ операций сравнения и чтения/записи — для этого достаточно трёх лент ММТ [11, 2].)

Общая сложность этапа оценивается как $\mathrm{O}(n\log n)$ — по $\mathrm{O}(n)$ для каждой группы. Отметим, что сжатие позволило понизить сложность сравнения в j-й группе с $\mathrm{O}(\log n)$ до $\mathrm{O}(j)$ и порядок сложности этапа — с $n\log n\log\log n$ до $n\log n$.

4) Для каждого из полученных на предыдущем этапе списков S выполняем обновление ленты T, т. е. проставление $a(k)=1$, если $k\in S$. Операция выполняется за один

просмотр списка и один проход по ленте со сложностью $O(n)$. Таким образом, общая сложность этапа — $O(n \log n)$.

5) После предыдущего шага лента T полностью сформирована: нули стоят строго в позициях с простыми номерами. Выписать последовательность простых чисел теперь можно за один просмотр ленты со сложностью порядка $\pi(n) \log n + n \asymp n$. ■

ЛИТЕРАТУРА

1. *Крэндалл Р., Померанс К.* Простые числа: криптографические и вычислительные аспекты. М.: УРСС: Либроком, 2011. 664 с.
2. *Schönhage A., Grotefeld A. F. W., and Vetter E.* Fast Algorithms: a Multitape Turing Machine Implementation. Mannheim, Leipzig, Wien, Zürich: BI-Wissenschaftsverlag, 1994. 298 p.
3. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
4. *Fürer M.* How fast can we multiply large integers on an actual computer? `arXiv:1402.1811`. 2014.
5. *Pritchard P.* A sublinear additive sieve for finding prime numbers // Comm. ACM. 1981. V. 24. No. 1. P. 18–23.
6. *Farach-Colton M. and Tsai M.-T.* On the complexity of computing prime tables. `arXiv:1504.05240`. 2015.
7. *Atkin A. O. L. and Bernstein D. J.* Prime sieves using binary quadratic forms // Math. Comput. 2004. V. 73. No. 246. P. 1023–1030.
8. *Mairson H. G.* Some new upper bounds on the generation of prime numbers // Comm. ACM. 1977. V. 20. No. 9. P. 664–669.
9. *Rosser J. and Schoenfeld L.* Approximate formulas for some functions of prime numbers // Ill. J. Math. 1962. V. 6. P. 64–94.
10. *Sándor J., Mitrinović D. S., and Crstici B.* Handbook of Number Theory. I. Dordrecht: Springer, 2006. 622 p.
11. *Кнут Д.* Искусство программирования. Т. 3. Сортировка и поиск. М.: Вильямс, 2008. 832 с.

REFERENCES

1. *Crandall R. and Pomerance C.* Prime Numbers. A Computational Perspective. Springer, 2005. 597 p.
2. *Schönhage A., Grotefeld A. F. W., and Vetter E.* Fast Algorithms: a Multitape Turing Machine Implementation. Mannheim, Leipzig, Wien, Zürich, BI-Wissenschaftsverlag, 1994. 298 p.
3. *Aho A., Hopcroft J., and Ullman J.* The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1974.
4. *Fürer M.* How fast can we multiply large integers on an actual computer? `arXiv:1402.1811`. 2014.
5. *Pritchard P.* A sublinear additive sieve for finding prime numbers. Comm. ACM, 1981, vol. 24, no. 1, pp. 18–23.
6. *Farach-Colton M. and Tsai M.-T.* On the complexity of computing prime tables. `arXiv:1504.05240`. 2015.
7. *Atkin A. O. L. and Bernstein D. J.* Prime sieves using binary quadratic forms. Math. Comput., 2004, vol. 73, no. 246, pp. 1023–1030.
8. *Mairson H. G.* Some new upper bounds on the generation of prime numbers. Comm. ACM, 1977, vol. 20, no. 9, pp. 664–669.

9. *Rosser J. and Schoenfeld L.* Approximate formulas for some functions of prime numbers. Ill. J. Math., 1962, vol. 6, pp. 64–94.
10. *Sándor J., Mitrinović D. S., and Crstici B.* Handbook of Number Theory. I. Dordrecht, Springer, 2006. 622 p.
11. *Knuth D. E.* The Art of Computer Programming. Vol. 3. Sorting and Searching. Reading, Massachusetts, Addison-Wesley, 1998.