УДК 004.021, 519.116 DOI: 10.14529/cmse240102

РАСПРЕДЕЛЕНИЕ КВАДРАТОВ И ПРОВЕРКА ГИПОТЕЗ В НЕЧЕТНЫХ РАЗБИЕНИЯХ ЧИСЕЛ

© 2024 А.А. Самойлов

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: samoilovaa@susu.ru

Поступила в редакцию: 10.03.2023

В статье рассматриваются разбиения натурального числа n, части которого различны, нечетны и их произведение не является квадратом. Такие разбиения применимы для определения ранга группы центральных единиц целочисленного группового кольца знакопеременной группы. Количество разбиений растет экспоненциально, следовательно, задача перебора является вычислительно затратной. В статье предложен параллельный алгоритм в общей памяти для нахождения количества разбиений числа n с дополнительными условиями. Алгоритм основан на концепции распараллеливания по данным и использовании вложенного параллелизма. Выделяется множество длин K разбиения числа n, элементы которого обрабатываются параллельно. Во время обработки длины k разбиения числа n выделяется множество уровней L, рассмотрение которого также выполняется параллельно. Приемлемые значения ускорения и параллельной эффективности предложенного алгоритма получаются при использовании двух нитей на параллельный регион по длинам и двух — по уровням. Таким образом, ускорение при разных n превышает 2.1, а параллельная эффективность не опускается ниже 50%. Полученные результаты использованы для проверки гипотез Каргаполова и анализа распределения значений нечетных разбиений на некоторых диапазонах. Предложен алгоритм поиска оптимального коэффициента c. С помощью этого алгоритма получена асимптотическая формула количества разбиения числа n, в котором части различны и нечетны, а их произведение является квадратом. Эта формула основана на экспериментальных данных и сформулирована как гипотеза.

Ключевые слова: разбиение натурального числа, асимптотическая формула, параллельные вычисления, OpenMP.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Самойлов А.А. Распределение квадратов и проверка гипотез в нечетных разбиениях чисел // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 1. С. 22–37. DOI: 10.14529/cmse240102.

Введение

В этой работе рассматриваются следующие условия на разбиение $\lambda = (a_1, \ldots, a_k)$ натурального числа n.

1. a_i нечетно при $1 \le i \le k$.
2. $a_i \ne a_j$ при $i \ne j$.
3. $n \equiv k \pmod 4$.
4. $\prod\limits_{i=1}^{k} a_i$ *не является квадратом* натурального числа.

Число разбиений, удовлетворяющих первым двум условиям, обозначается $r(n)$, удовлетворяющих первым трем условиям — $r_4(n)$, а разбиение, у которого выполняются первые три условия и последнее нарушается, называется *квадратичным нечетным разбиением* числа n, и число таких разбиений обозначается $qop(n)$. Количество разбиений, удовлетворяющих всем условиям, обозначается $rank(n)$. Можно заметить, что $qop(n) = r_4(n) - rank(n)$.

Одно из практических применений этих четырех условий — это определение ранга группы центральных единиц целочисленного группового кольца знакопеременной группы A_n [1].

Эта проблема рассматривалась авторами из работы [2], которым удалось достигнуть результата для $n \leq 600$ с шагом 100. В работе [3] Каргаполовым достигнут результат для $n \leq 1000$ с шагом 10. Каргаполов выдвинул предположения, связанные с оценкой ранга группы центральных единиц целочисленного группового кольца знакопеременной группы A_n.

Разбиения чисел могут быть применены в криптографии для создания протоколов аутентификации и шифрования. Известно, что число классов сопряженных элементов симметрической группы S_n равно количеству разбиений числа n (см. [4]). Пример криптосистемы на основе симметрических групп можно найти в [5].

Целью исследования являются следующие задачи.

1. Проверить применимость предположений Каргаполова для различных диапазонов натуральных чисел.
2. Вычислить количество и найти оценки квадратичных нечетных разбиений для различных диапазонов натуральных чисел.

Для достижения поставленных задач предложен и реализован алгоритм с шагом 1, который использует стандарт для распараллеливания программ OpenMP.

Статья организована следующим образом. В разделе 1 описаны теоретические сведения о разбиении числа n, количестве и оценке. В разделе 2 приведен алгоритм подсчета количества разбиений с дополнительными условиями и предложен метод поиска оптимального коэффициента c. В разделе 3 представлены реализация алгоритма с использованием стандарта распараллеливания программ OpenMP и вычислительные эксперименты. В разделе 4 приведена проверка гипотез Каргаполова и дано предположение об асимптотической формуле числа квадратичных нечетных разбиений. Заключение содержит краткую сводку результатов, полученных в работе.

1. Разбиение числа, количество и оценка

Определение 1. Разбиение λ натурального числа n — представление этого числа в виде суммы натуральных чисел

$$n = a_1 + a_2 + \cdots + a_k, \tag{1}$$

где, a_i называются частями разбиения.

Разбиения одного числа считаются равными, если они различаются только порядком слагаемых. Вычисления, связанные с разбиениями чисел, являются вычислительно затратными, так как количество разбиений быстро увеличивается с ростом n [6]. Приблизительно число разбиений $p(n)$ можно найти с помощью формулы Харди—Рамануджана.

Теорема 1. (Теорема 6.3 [6], §2.7 [7]). При $n \to \infty$ имеет место асимптотическая формула для числа разбиений

$$p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{\frac{2n}{3}}}. \tag{2}$$

Один из подходов вычисления количества разбиений с условиями из введения следующий. Генерацию разбиений можно осуществить с помощью алгоритма Липского, который перебирает все разбиения числа n в невозрастающем порядке (алгоритм 1.22. [8]). Этот алгоритм нуждается в изменениях, чтобы части разбиения соответствовали условиям различности, нечетности, и добавлении проверки на квадрат. Для сокращения времени на

вычисление количества разбиений прибегают к параллельной обработке в распределенной памяти. Для этого все множество разбиений делят на равные части, и каждый узел кластера выполняет генерацию разбиений в заданном диапазоне.

В этой статье предложен параллельный алгоритм в общей памяти с использованием вложенного параллелизма для нахождения количества разбиений числа n с дополнительными условиями. Вложенный параллелизм заключается в том, что каждая нить, в которой встретится параллельный регион, породит для его выполнения новую группу нитей.

Лемма 1. (Лемма 2 [2]) Число элементов разбиения k, удовлетворяющих условиям из введения, не превосходит целой части $\sqrt{n}$.

Определим множество длин K разбиения числа n, значения элементов которых не превышают $\sqrt{n}$ (см. лемму 1) и удовлетворяют третьему условию из введения $n \equiv k \pmod 4$, как

$$K = \{k_i : 1 \leq k_i \leq \lfloor \sqrt{n} \rfloor \wedge n \equiv k_i \pmod 4\}. \tag{3}$$

При генерации разбиений, элементы которых нечетны и попарно различны, можно заметить, что все части некоторых разбиений отличаются на 2. Например, для длины $k = 3$ существует следующий набор разбиений: $l_1 = (1, 3, 5), l_2 = (3, 5, 7), l_3 = (5, 7, 9), \ldots$ Назовем такие случаи *уровнями* и определим множество уровней L разбиения числа n длины k как

$$l_j = \{a_i : 1 \leq i \leq k \wedge a_1 = 1 + 2(j-1) \wedge a_i = a_{i-1} + 2\}, \tag{4}$$

$$L = \{l_j : 1 \leq j \leq \infty \wedge \sum_{i=1}^{k} a_i \leq n, \text{ где } a_i \text{ элемент } l_j\}. \tag{5}$$

Из работы [3] стоит отметить способ проверки на квадрат натурального числа — использование разложения на простые множители. Если число является квадратом, то показатели степеней простых чисел в его разложении четны. Для каждого натурального числа n сохраняется битовый вектор четностей показателей степеней простых чисел, где на i-й позиции стоит остаток от деления на 2 показателя степени i-го простого числа в разложении n.

В работе [3] Каргаполовым получена асимптотическая формула, которая позволяет вычислить приближенно $r(n)$, и высказаны предположения о $r_4(n)$ и $rank(n)$.

Теорема 2. (Теорема 3.1 [3]. При $n \to \infty$ имеет место асимптотическая формула

$$r(n) \sim \frac{1}{2\lambda_n^{\frac{3}{2}}\sqrt[4]{24}} e^{\frac{\pi\lambda_n}{\sqrt{6}}}, \text{ где } \lambda_n = \sqrt{n - \frac{1}{24}}. \tag{6}$$

Предположения Каргаполова

Предположение 1. При $n \to \infty$ имеет место асимптотическая формула

$$r_4(n) \sim \frac{r(n)}{2}. \tag{7}$$

Предположение 2. При $n \to \infty$ имеет место асимптотическая формула

$$rank(n) \sim r_4(n) \sim \frac{1}{4\sqrt[4]{24n^3}} e^{\pi\sqrt{\frac{n}{6}}}. \tag{8}$$

Из теорем 1 и 2 можно сказать об экспоненциальном росте числа разбиений и предположить, что при $n \to \infty$ имеет место следующая форма записи асимптотической формулы для количества разбиений

$$f(n) \sim \frac{a}{n^c} e^{b\sqrt{n}} \tag{9}$$

где, a, b, c некоторые коэффициенты. Гипотеза Каргаполова (8) также соответствует этому выражению.

2. Описание методов

2.1. Алгоритм подсчета числа разбиений

Алгоритм подсчета количества разбиений числа n с указными во ведении условиями состоит из следующих шагов.

1. Вычисление битовых векторов четностей показателей степеней простых чисел.
2. Определение множества длин K разбиения числа n.
3. Определение множества уровней L разбиения числа n длины k_i.
4. Рекурсивный подсчет количества разбиений числа n длины k_i уровня l_j.

Шаг 1. В алг. 1 описывается вычисление битовых векторов четностей показателей степеней простых чисел $D_{N,b}$, где N — количество натуральных чисел, b — количество бит. Для получения простых чисел (функция $GenPrimes$) можно использовать алгоритм решета Эратосфена (§7.1 [9]). Для $N = 1000$ достаточно $b = 168$ бит, так как в диапазоне $[1, 1000]$ существуют 168 простых чисел. Далее проверку на целочисленный квадратный корень можно осуществить путем применения операции xor над битовым вектором каждого элемента разбиения. Произведение частей разбиения числа n является квадратом в том случае, если на выходе получился нулевой вектор (§2.1 [3]).

Алгоритм 1 GenDecomposition (in N, b; out $D_{N,b}$)

```
1: D_{N,b} ← ∅; P ←GenPrimes(N)
2: for i ← 2 to N do
3:     t ← i; D_i ← 0 {инициализация нулевого битового вектора}
4:     for j ← 1 to b do
5:         while t mod P_j = 0 do
6:             D_{i,j} ← ¬D_{i,j}; t ← t/P_j
7: return D_{N,b}
```

Шаг 2. В алг. 2 описываются нахождение (строки 1–4) и параллельная обработка (строки 5–7) множества длин, которое определяется выражением (3). Вычислительные эксперименты показали (см. табл. 2), что большими затратами времени на нахождение количества разбиений обладает средний и ближние к среднему элементы множества K. Следовательно, при параллельном рассмотрении этого множества с использованием двух нитей, одной из них достанется элемент, который обладает большим временем выполнения, а другой — два соседних.

Алгоритм 2 LenParall (in n)

```
1: K_m ← ∅; m ← 0
2: for i ← 1 to ⌊√n⌋ do
3:     if n mod 4 = i mod 4 then
4:         K_m ← i; m ← m + 1
5: #pragma omp parallel for num_threads(2) schedule(dynamic, 1) reduction(+:r_4, qop, rank)
6: for i ← 1 to m do
7:     r_4, qop, rank ← r_4, qop, rank + LevelParall(n, K_i)
```

Шаг 3. Подсчет количества уровней (сами уровни определяются выражениями (4)–(5)) и их параллельная обработка описаны в алг. 3. Строки 1–2 соответствуют частному случаю, в котором происходит проверка на четность и квадрат числа n, когда длина разбиения $k = 1$ (описание функции *SpecialCase* опущена для краткости). Строки 4–12 соответствуют подсчету количества уровней разбиения числа n длины k. В строках 4–5 производится нахождение суммы первого уровня. Далее, в строках 6–12 происходит инкремент переменной m до тех пор, пока сумма элементов разбиения, по определению 1, не станет равной n. Строки 13–19 описывают параллельную обработку уровней. В строках 17–18 происходит генерация уровня l_j для разбиения числа n длины k, и вычисляется сумма частей этого разбиения.Вычислительные эксперименты показали (см. табл. 3), что чем меньше номер уровня, тем большими затратами времени на вычисление количества разбиений он обладает.

Алгоритм 3 LevelParall (in n, k; out $r_4, qop, rank$)

```
1: if k = 1 then
2:     return SpecialCase (in n; out r_4, qop, rank)
3: m ← 0; sum ← 0; r_4 ← 0; rank ← 0; qop ← 0
4: for i ← 1 to k do
5:     sum ← sum + 1 + 2(i − 1)
6: while true do
7:     if sum > n then
8:         break
9:     else if sum = n then
10:        m ← m + 1; break
11:    else
12:        m ← m + 1; sum ← sum + 2k
13: #pragma omp parallel for num_threads(2) schedule(dynamic, 1) reduction(+:r_4, qop, rank)
14: for j ← 1 to m do
15:    L_k^k ← 0; sum ← 0
16:    for i ← 1 to k do
17:        L_i^1 ← 1 + 2(j − 1) + 2(i − 1)
18:        sum ← sum + L_i^1
19:    r_4, qop, rank ← r_4, qop, rank + LevelRec(2, sum, n, k, L_k^k)
20: return r_4, qop, rank
```

Шаг 4. На последнем этапе (см. алг. 4) выполняется рекурсивная генерация разбиений числа n длины $k > 1$ от второго элемента к последнему в рамках заданного уровня l. Перед вызовом нового шага рекурсии сохраняется текущее разбиение (строка 4). Для удовлетворения условиям нечетности и различности элементов на каждом j-м шаге после вызова рекурсии прибавляется к частям разбиения, которые стоят на позициях в диапазоне $[j, k]$, число 2 и пересчитывается их сумма (строки 6–8). При достижении $j = k$ прибавляется к последнему элементу разбиения недостающая сумма, чтобы соответствовать определению 1 (строка 10). Далее, выполняется проверка на квадрат числа с помощью показателей степеней простых чисел (строки 13–15).

Алгоритм 4 LevelRec (in j, sum, n, k, L_k^k; out qop, $rank$, r_4)

1: **while** $true$ **do**
2: **if** $sum < n$ **then**
3: **if** $k \neq j$ **then**
4: $L^{j+1} \leftarrow L^j$ {Сохранение текущего разбиения для следующего шага рекурсии}
5: $qop, rank, r_4 \leftarrow qop, rank, r_4 + \text{LevelRec}(j+1, sum, n, k, L_k^k)$
6: $sum \leftarrow sum + 2(k-j-1)$ {Сумма частей разбиения на текущем шаге рекурсии}
7: **for** $i \leftarrow j$ **to** k **do**
8: $L_i^j \leftarrow L_i^j + 2$ {Прибавление числа 2 к частям разбиения начиная с позиции равному шагу рекурсии}
9: **else**
10: $L_k^j \leftarrow L_k^j + (n - sum)$; $sum \leftarrow n$ {На последнем шаге рекурсии прибавляется недостающая сумма к последнему элементу}
11: **else if** $sum = n$ **then**
12: $r_4 \leftarrow r_4 + 1$; $factor \leftarrow 0$
13: **for** $i \leftarrow 1$ **to** k **do**
14: $factor \leftarrow factor \oplus D_{L_i^j}$ {Использование битовых векторов четностей показателей степеней простых чисел для проверки на квадрат}
15: **if** $factor = 0$ **then**
16: $qop \leftarrow qop + 1$ {Произведение элементов разбиения является квадратом}
17: **else**
18: $rank \leftarrow rank + 1$
19: **return** $qop, rank, r_4$
20: **else**
21: **return** $qop, rank, r_4$

2.2. Поиск оптимального коэффициента c

В алг. 5 описывается поиск оптимального коэффициента c. Пусть X_N содержит значения $n \in [1, N]$, Y_N — количество разбиений числа n, и задан замкнутый интервал $C \in [begin, end]$ с шагом ϵ. Возьмем во внимание, что при выполнении аппроксимации на разных диапазонах заданных точек можно получить разные значения параметров. *Поиск оптимального коэффициента c* заключается в нахождении такого значения коэффициента c, при котором отклонение двух других коэффициентов a и b из (9) на меньшем и большем диапазонах аппроксимации минимально. Будем считать большим диапазоном аппроксима-

ции равный $n \in [1, N]$, а меньшим — $n \in [\lfloor N/8 \rfloor, \lfloor N/4 \rfloor]$ (строки 2–3). Последовательно рассматривая C с шагом ϵ и выполняя функцию $CurveFit$ получим значения коэффициентов a_1, b_1 для меньшего и a_2, b_2 для большего диапазонов аппроксимации (строки 4–11). Для подбора коэффициентов по формуле (9) и в качестве реализации функции $CurveFit$ можно использовать нелинейный метод наименьших квадратов (§2.4 [10]). Эта функция на вход принимает коэффициент c и табличные значения X_N, Y_N (строки 6–7). Обозначим разность коэффициентов как $t_1 = |a_2 - a_1|$ и $t_2 = |b_2 - b_1|$. Выбираем то значение c, у которого на замкнутом интервале C значения t_1 и t_2 минимальны (строки 8–10). Будем считать оптимальным значением c то, у которого на меньшем промежутке аппроксимации значения t_1 и t_2 минимальны (строки 12–13). Например, для $rank(n)$ при $C \in [0, 1]$ и $\epsilon = 0.001$ по предложенному алгоритму получен результат $c = 0.731 \sim 3/4$, который близок к значению в гипотезе Каргаполова (8).

Алгоритм 5 OPTIMIZE (IN $begin$, end, ϵ, X_N, Y_N; OUT opt_{t_1}, opt_{t_2}, opt_c)

```
1:  opt_t1 ← ∞; opt_t2 ← ∞; opt_c ← begin
2:  for n ← ⌊N/8⌋ to ⌊N/4⌋ do
3:      A ← X_1..n; B ← Y_1..n {Копируем n элементов}
4:      tmp_t1 ← ∞; tmp_t2 ← ∞; tmp_c ← begin; c ← begin
5:      while c ≤ end do
6:          a_1, b_1 ← CurveFit(A, B, c)
7:          a_2, b_2 ← CurveFit(X, Y, c)
8:          t_1 ← |a_2 − a_1|; t_2 ← |b_2 − b_1|
9:          if tmp_t1 > t_1 and tmp_t2 > t_2 then
10:             tmp_t1 ← t_1; tmp_t2 ← t_2; tmp_c ← c
11:         c ← c + ε
12:     if opt_t1 > tmp_t1 and opt_t2 > tmp_t2 then
13:         opt_t1 ← tmp_t1; opt_t2 ← tmp_t2; opt_c ← tmp_c
14: return opt_t1, opt_t2, opt_c
```

3. Реализация и вычислительные эксперименты

Алгоритм подсчета числа разбиений реализован на языке C при использовании стандарта для распараллеливания программ OpenMP. Поиск оптимального коэффициента c реализован на языке Python. Исходные коды свободно доступны в репозитории GitHub [11].

По предложенному алгоритму возможно выполнить вложенный параллелизм следующим образом.

1. Параллельный регион из двух OpenMP-нитей на обработку множества длин K разбиения числа n.
2. Параллельный регион из двух OpenMP-нитей на обработку множества уровней L разбиения числа n в рамках соответствующей длины k.

На рис. 1 представлена схема, согласно которой программа создаст, при максимальной загрузке, четыре OpenMP-нити. Для поддержки вложенного параллелизма, необходимо изменить переменную окружения, отвечающую за данную опцию, с помощью функции $omp\_set\_nested(1)$.

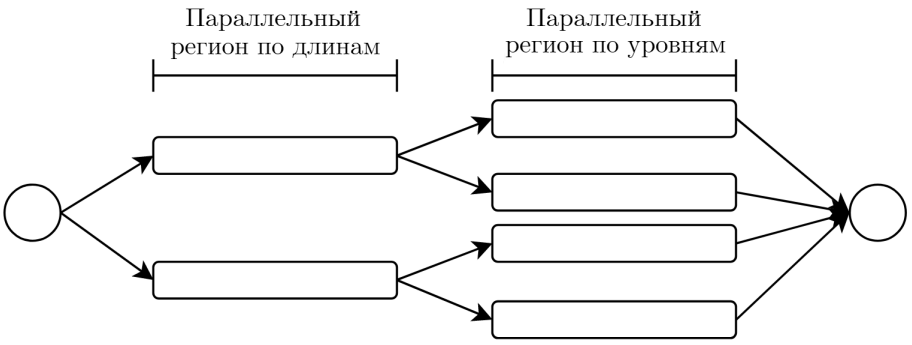


Рис. 1. Схема распараллеливания

Для вычислительных экспериментов использован персональный компьютер, который обладает характеристиками, указанными в табл. 1.

Таблица 1. Характеристики

Параметр	**Значение**
Процессор	AMD Ryzen 9 5950X 3.40 GHz
Число физ./лог. ядер	16/32
Кэш	L1 — 1.0 МБ, L2 — 8.0 МБ, L3 — 64.0 МБ
Операционная система	Windows 10 Pro x64
Оперативная память	32 ГБ, 3200 MHz

Время выполнения алгоритма для элементов k_i из множества длин K отображено в табл. 2 в зависимости от n. Вычислительные эксперименты показали, что количество разбиений числа n длины k_i возрастает до некоторого элемента множества K, а затем уменьшается. В большинстве случаев основное время вычисления занимает элемент с индексом $i = \lceil (|K| + 1)/2 \rceil$.

Таблица 2. Время выполнения k_i, с.

$\boldsymbol{n}$	$\boldsymbol{i = 1}$	$\boldsymbol{i = 2}$	$\boldsymbol{i = 3}$	$\boldsymbol{i = 4}$	$\boldsymbol{i = 5}$	$\boldsymbol{i = 6}$	$\boldsymbol{i = 7}$
400	0.001	0.387	1.662	0.043	0.001	-	-
425	0.001	0.007	1.586	2.882	0.032	-	-
450	0.001	0.051	4.748	4.125	0.019	-	-
475	0.001	0.374	13.104	5.192	0.009	-	-
500	0.001	2.042	30.408	5.646	0.004	-	-
525	0.001	0.016	9.909	62.869	5.353	0.001	-
550	0.001	0.144	34.640	107.794	4.288	0.001	-
575	0.001	1.229	111.656	165.038	2.956	0.001	-
600	0.002	7.842	301.919	214.401	1.698	0.001	-
625	0.001	0.033	44.023	725.841	247.097	0.817	0.001
650	0.001	0.338	176.511	1447.473	245.358	0.322	-
675	0.001	3.326	650.631	2547.758	211.521	0.101	-
700	0.003	24.070	2008.258	3903.361	156.567	0.025	-

Время выполнения алгоритма для элементов l_j из множества уровней L отображено в столбцах 5–9 в табл. 3 в зависимости от n. Во втором столбце указана длина k_i разбиения числа n, где $i = \lceil (|K|+1)/2 \rceil$, в третьем — мощность множества $|L|$, в четвертом — количество $r_4(n)$ для числа n длины k_i. Вычислительные эксперименты показали, что количество разбиений числа n длины k_i уровня l_j уменьшается с ростом j.

Таблица 3. Время выполнения l_j, с.

n	k_i	$\lvert L\rvert$	r_4	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$
400	12	11	$1.3 \cdot 10^8$	0.943	0.430	0.185	0.073	0.026
425	9	20	$1.4 \cdot 10^8$	0.506	0.361	0.252	0.173	0.117
450	10	18	$4.2 \cdot 10^8$	1.770	1.167	0.753	0.474	0.289
475	11	17	$1.1 \cdot 10^9$	5.524	3.352	1.979	1.128	0.619
500	12	15	$2.5 \cdot 10^9$	14.308	7.967	4.275	2.186	1.069
525	13	14	$4.8 \cdot 10^9$	33.040	16.661	8.021	3.660	1.571
550	14	13	$7.9 \cdot 10^9$	61.741	28.069	12.045	4.841	1.800
575	15	12	$1.2 \cdot 10^{10}$	100.423	40.685	15.391	5.377	1.707
600	16	11	$1.5 \cdot 10^{10}$	140.244	50.385	16.635	4.977	1.324
625	13	18	$5.5 \cdot 10^{10}$	334.202	193.965	107.724	58.396	30.497
650	14	17	$1.1 \cdot 10^{11}$	724.543	391.159	199.100	97.121	45.406
675	15	16	$1.8 \cdot 10^{11}$	1377.487	677.709	313.998	137.651	57.033
700	16	14	$2.7 \cdot 10^{11}$	2280.791	1014.834	422.054	164.246	59.489

Таблица 4 содержит затраченное время в секундах на выполнение алгоритма поиска количества разбиения с дополнительными условиями для числа n различной размерности. В таблице th_k — число нитей на параллельный регион по длинам и th_l — число нитей на параллельный регион по уровням.

Таблица 4. Время выполнения алгоритма, с.

n	$th_k=1$				$th_k=2$			
	$th_l=1$	$th_l=2$	$th_l=3$	$th_l=4$	$th_l=1$	$th_l=2$	$th_l=3$	$th_l=4$
400	2.094	1.179	1.123	1.105	1.672	0.958	0.958	0.973
500	38.255	20.773	19.532	19.306	30.770	15.684	14.721	14.636
600	530.554	299.363	269.692	268.925	308.938	156.577	147.153	146.164
700	6091.733	3422.781	3143.981	3137.729	3920.940	2323.431	2300.573	2309.608

Ускорение, получаемое при использовании параллельного алгоритма для p ядер, по сравнению с последовательным вариантом выполнения вычислений, определяется как

$$S_p(n) = \frac{T_1(n)}{T_p(n)}, \tag{10}$$

где $T_1(n)$ — время выполнения последовательного алгоритма и $T_p(n)$ — время выполнения параллельного алгоритма при использовании p ядер.

Эффективность — средняя доля времени выполнения параллельного алгоритма, в течение которого ядра процессора реально используются для решения задачи, определяется

как

$$E_p(n) = \frac{S_p(n)}{p}. \tag{11}$$

Из практики известно, что одна нить использует одно ядро в многоядерном процессоре, тогда число используемых ядер равно $th_k * th_l$. Результаты исследования показаны на рис. 2 при $th_k = 2$. Вычислительные эксперименты показали, что оптимальным вариантом является использование двух нитей на параллельный регион по длинам $th_k = 2$ и двух — по уровням $th_l = 2$. Например, при использовании 6 ядер ускорение увеличивается незначительно, а эффективность становиться меньше 50% при некоторых n.

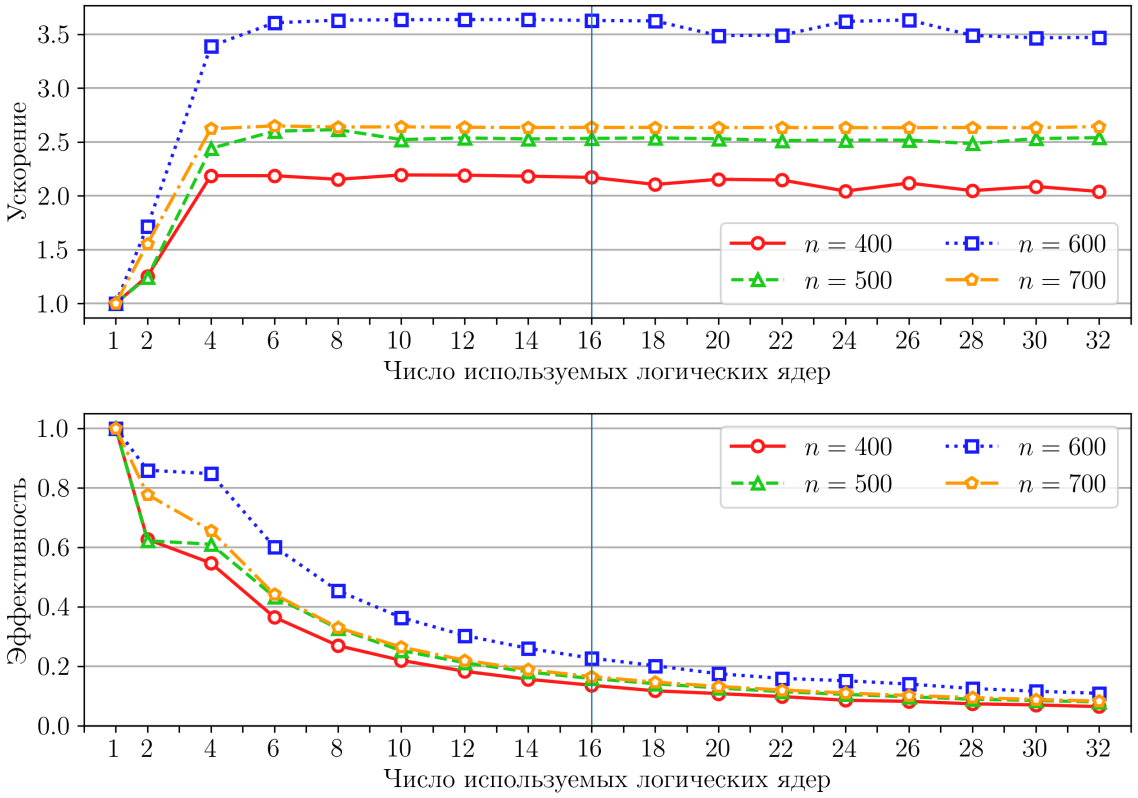


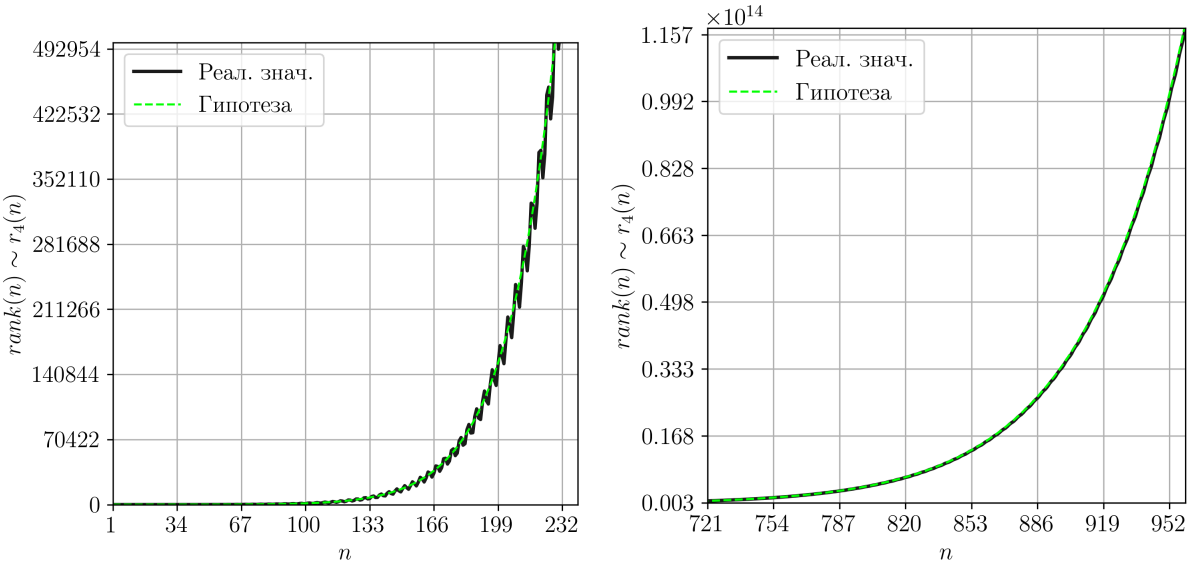
Рис. 2. Ускорение и параллельная эффективность алгоритма при $th_k = 2$

4. Анализ результатов

4.1. Гипотезы Каргаполова

Вычислены значения количества разбиений для трех величин, соответствующим условиям из введения, для $n \leq 960$ с шагом 1. Для проверки применимости предположений Каргаполова (7) и (8), рассмотрим два диапазона значений $n \in [1, 240]$ и $n \in [721, 960]$. Выбор обусловлен тем, что при небольших значениях n присутствуют колебания графика количества разбиений, но с возрастанием n они визуально исчезают.

Расчет по формулам (7) и (8) показал, что обе гипотезы дают приблизительно одинаковые значения. На рис. 3 изображен график функции из второй гипотезы Каргаполова. На рис. 3а присутствуют колебания $rank(n) \sim r_4(n)$, но при увеличении n кривая сглаживается (рис. 3б). Относительная погрешность гипотезы от реальных значений показана на рис. 4 (красной линией обозначено пороговое значение 1%).



а) в диапазоне $n \in [1, 240]$ б) в диапазоне $n \in [721, 960]$

Рис. 3. Предположение Каргаполова

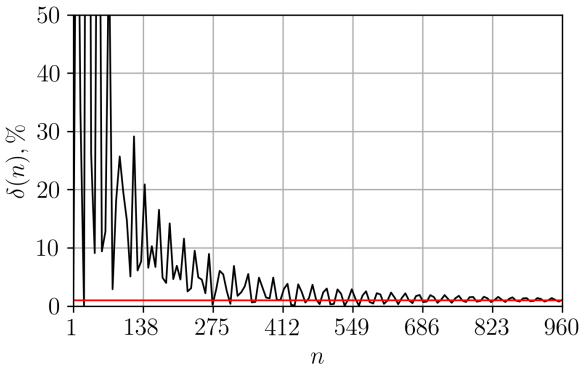


Рис. 4. Относительная погрешность гипотезы

Имеет место следующий вывод по гипотезам Каргаполова, что $rank(n) \sim r_4(n) \sim r(n)/2$ с ростом n (см. табл. 5). Таким образом, можно допустить, что при $n > 960$ гипотезы будут давать точные результаты, но при небольших n имеет место значительное расхождение с реальными значениями.

4.2. Приближенная формула для квадратичных нечетных разбиений

Для получения приближенной формулы воспользуемся методом, описанным в подразделе 2.2. Для $qop(n)$ при $C \in [0, 2]$ и $\epsilon = 0.001$ получаем коэффициент $c = 1.449$. После выполнения аппроксимации по формуле (9) для квадратичных нечетных разбиений, стали известны коэффициенты $a = 0.43$ и $b = 1$. Подставляя результаты в (9) получим при $n \to \infty$

$$qop(n) \sim \frac{0.43}{n^{1.449}} e^{\sqrt{n}}. \tag{12}$$

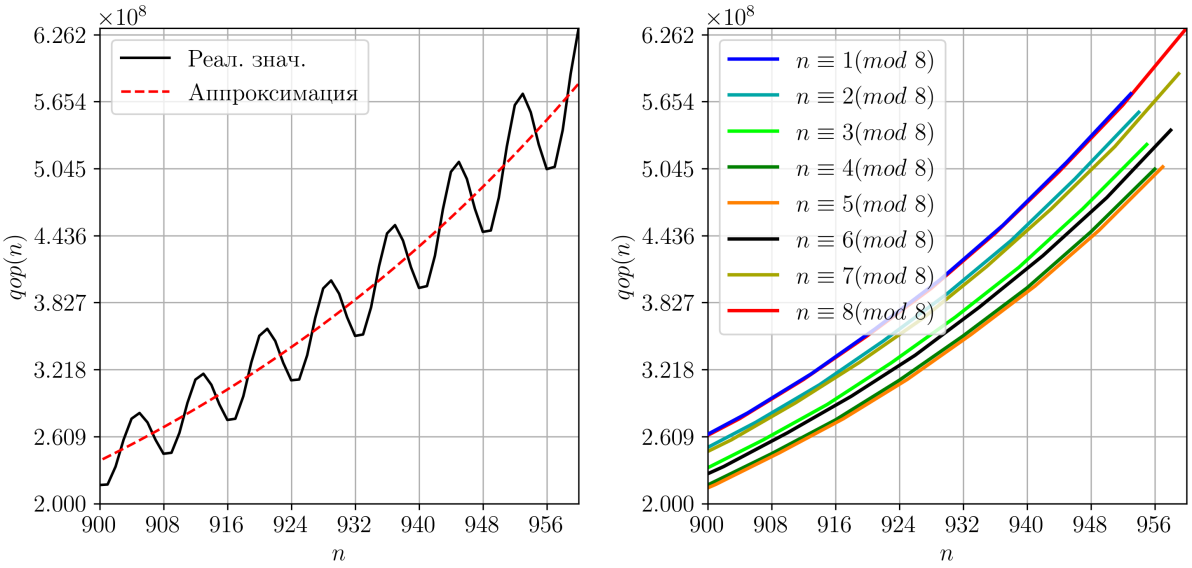
Из графика (см. рис. 5а) функции (12) видно, что простое применение метода не учитывает колебания, которые с ростом n не сглаживаются для $qop(n)$, и имеют период $T = 8$. Разобьем на части в зависимости от сравнения по модулю 8 числа n. Таким образом, на

Таблица 5. Относительная погрешность, %

n	Гипотеза 1	Гипотеза 2
236	2.55	2.39
237	4.15	4.00
238	6.63	6.80
239	9.32	9.49
240	1.69	1.53
...	...	...
956	1.00	1.08
957	0.68	0.76
958	0.92	1.01
959	1.25	1.33
960	1.01	1.09

рис. 5б для $qop(n)$ можно заметить образование четырех пар. Следовательно, для оценки колебаний рассмотрим разложение на 4 части

$$\{8k+1,\ 8k+8\};\ \{8k+2,\ 8k+7\};\ \{8k+3,\ 8k+6\};\ \{8k+4,\ 8k+5\}. \tag{13}$$



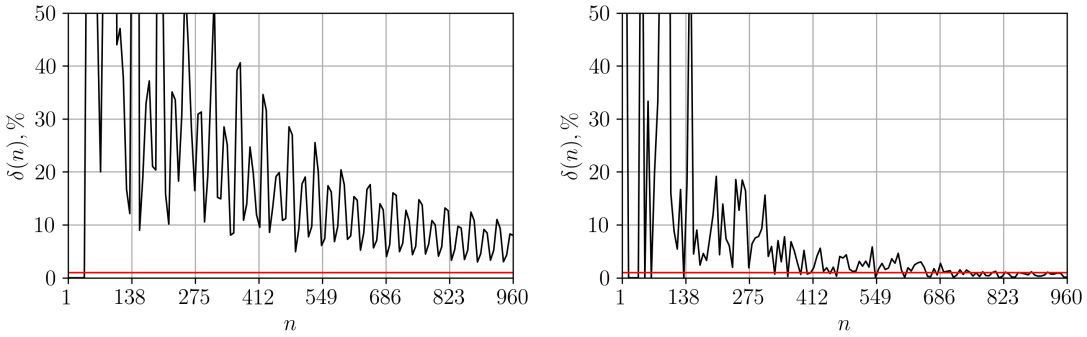
а) без разложения б) разложение на части

Рис. 5. Квадратичные нечетные разбиения

При таком рассмотрении количества квадратичных нечетных разбиений, получены коэффициенты c для каждой части из списка, и найдены a и b при выполнении аппроксимации по формуле (9). Полученные результаты сформулированы как предположение 1. При разложении $qop(n)$ на 4 части относительная погрешность уменьшилась (см. рис. 6 и табл. 6).

Таблица 6. Относительная погрешность $qop(n)$, %

n	Без разложения	Разложение на 4 части
236	49.69	19.43
237	67.48	9.58
238	30.19	17.88
239	18.25	1.96
240	36.85	11.40
...	...	...
956	8.87	0.69
957	10.02	0.38
958	4.75	0.83
959	2.94	0.47
960	7.99	0.10



а) без разложения б) разложение на 4 части

Рис. 6. Относительная погрешность $qop(n)$

Предположение 1. При $n \to \infty$ имеет место асимптотическая формула

$$qop(n) \sim \begin{cases} \frac{3.301}{n^{1.802}} e^{1.015\sqrt{n}}, & \text{при } n \equiv 1 \pmod 8 \text{ или } n \equiv 8 \pmod 8 \\ \frac{1.888}{n^{1.711}} e^{1.011\sqrt{n}}, & \text{при } n \equiv 2 \pmod 8 \text{ или } n \equiv 7 \pmod 8 \\ \frac{0.277}{n^{1.381}} e^{0.998\sqrt{n}}, & \text{при } n \equiv 3 \pmod 8 \text{ или } n \equiv 6 \pmod 8 \\ \frac{0.003}{n^{0.536}} e^{0.952\sqrt{n}}, & \text{при } n \equiv 4 \pmod 8 \text{ или } n \equiv 5 \pmod 8. \end{cases} \tag{14}$$

Заключение

В работе рассмотрены разбиения натурального числа n, части которого различны, нечетны и их произведение не является квадратом. Задача нахождения количества разбиений числа n является вычислительно затратной, так как число разбиений быстро увеличивается с ростом n.

Предложен параллельный алгоритм в общей памяти для нахождения количества разбиений числа n с дополнительными условиями. Алгоритм основан на концепции распараллеливания по данным и использовании вложенного параллелизма. Выделяется множество длин K разбиения числа n, элементы которого обрабатываются параллельно. Во время обработки длины k разбиения числа n выделяется множество уровней L, рассмотрение которого также выполняется параллельно. Приемлемые значения ускорения и параллель-

ной эффективности предложенного алгоритма получаются при использовании двух нитей на параллельный регион по длинам и двух — по уровням. Таким образом, ускорение при разных n превышает 2.1, а параллельная эффективность не опускается ниже 50%.

Рассмотрены гипотезы Каргаполова для оценки количества нечетных разбиений, соответствующие условиям из введения, для $r_4(n)$ и $rank(n)$. Анализ результатов показал, что на диапазоне $n \in [1, 960]$ с возрастанием n разница между реальными и оценочными значениями быстро уменьшается и относительная погрешность также незначительна. Таким образом, можно допустить, что оба предположения Каргаполова верны при $n > 960$.

Предложен алгоритм поиска оптимального коэффициента c. Этот алгоритм позволяет подобрать коэффициенты a и b с учетом того, что на разных диапазонах аппроксимации, могут получится разные значения этих коэффициентов. Для оценки количества квадратичных нечетных разбиений, множество значений $qop(n)$ разбивается на 4 части в зависимости от числа n. Затем использовался предложенный метод поиска оптимального коэффициента c. После подбора коэффициентов a, b и c из полученных результатов сформулировано предположение 1.

В качестве дальнейшего исследования может быть предложено теоретическое доказательство или опровержение предположения 1.

Литература

1. Ferraz R.A. Simple components and central units in group algebras // Journal of Algebra. 2004. Vol. 279, no. 1. P. 191–203. DOI: 10.1016/j.jalgebra.2004.05.005.
2. Алеев Р.Ж., Каргаполов А.В., Соколов В.В. Ранги групп центральных единиц целочисленных групповых колец знакопеременных групп // Фундаментальная и прикладная математика. 2008. Т. 14, № 7. С. 15–21.
3. Каргаполов А.В. Центральные единицы целочисленных групповых колец знакопеременных групп: дис. ... канд. / Каргаполов Андрей Валерьевич. Южно-Уральский Государственный Университет, Россия, 2012.
4. Sagan B.E. The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions. New York: Springer New York, 2001. XVI, 240 p. DOI: 10.1007/978-1-4757-6804-6.
5. Doliskani J.N., Malekian E., Zakerolhosseini A. A Cryptosystem Based on the Symmetric Group S_n // IJCSNS International Journal of Computer Science and Network Security. 2008. Vol. 8, no. 2. P. 226–234.
6. Эндрюс Г. Теория разбиений. Москва: Наука, 1982. 256 с.
7. Постников А.Г. Введение в аналитическую теорию чисел. Москва: Наука, 1971. 416 с.
8. Липский В. Комбинаторика для программистов. Москва: Мир, 1988. 200 с.
9. Андерсон Д.А. Дискретная математика и комбинаторика. Москва: Вильямс, 2004. 960 с.
10. Kelley C.T. Iterative Methods for Optimization. University City, Philadelphia: Society for Industrial and Applied Mathematics, 1999. 196 p. DOI: 10.1137/1.9781611970920.
11. Самойлов А.А. Исследование разбиений с дополнительными условиями. URL: https://github.com/SashaSamoilov/qop-partitions (дата обращения: 25.09.2023).

Самойлов Александр Андреевич, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240102

DISTRIBUTION OF SQUARES AND HYPOTHESIS VERIFICATION IN ODD INTEGER PARTITIONS

© 2024 A.A. Samoilov

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: samoilovaa@susu.ru

Received: 10.03.2023

The article considers partitions of a natural number n whose parts are distinct, odd and their product is not a square. Such partitions are applicable for determining the rank of the group of central units of an integral group ring of an alternating group. The number of partitions grows exponentially, hence the enumeration task is computationally expensive. The article proposes a parallel algorithm in shared memory for finding the number of partitions of the n with additional conditions. The algorithm is based on the concept of data parallelism and the use of nested parallelism. A set of lengths K of the partition of the n is obtained, the elements of which are processed in parallel. During the processing of the length k of the partition of the n, the set of levels L is obtained, the elements of which are processed in parallel as well. Acceptable values of speedup and parallel efficiency of the proposed algorithm are obtained by using two threads per parallel length region and two threads per parallel level region. Thus, the speedup for distinct n increases to 2.1, and the parallel efficiency does not decrease below 50%. The results obtained were used to check Kargapolov's hypotheses and analyze the distribution of the number of odd partition in certain ranges. An algorithm for finding the optimal coefficient c is proposed. Using this algorithm, an asymptotic formula for the number of partitions of the n is obtained, in which the parts are distinct, odd and their product is a square. This formula is based on experimental data and formulated as a conjecture.

Keywords: integer partition, asymptotic formula, parallel computing, OpenMP.

FOR CITATION

Samoilov A.A. Distribution of Squares and Hypothesis Verification in Odd Integer Partitions. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 1. P. 22–37. (in Russian) DOI: 10.14529/cmse240102.

References

1. Ferraz R.A. Simple components and central units in group algebras. Journal of Algebra. 2004. Vol. 279, no. 1. P. 191–203. DOI: 10.1016/j.jalgebra.2004.05.005.
2. Aleev R.Z., Kargapolov A.V., Sokolov V.V. The ranks of central unit groups of integral group rings of alternating groups. Fundamental and Applied Mathematics. 2008. Vol. 14, no. 7. P. 15–21. (in Russian).
3. Kargapolov A.V. Central units of integral group rings of alternating groups: PhD thesis / Kargapolov Andrey Valerievich. South Ural State University, Russia, 2012. (in Russian).
4. Sagan B.E. The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions. New York: Springer New York, 2001. XVI, 240 p. DOI: 10.1007/978-1-4757-6804-6.

5. Doliskani J.N., Malekian E., Zakerolhosseini A. A Cryptosystem Based on the Symmetric Group S_n. IJCSNS International Journal of Computer Science and Network Security. 2008. Vol. 8, no. 2. P. 226–234.

6. Andrews G. The Theory of Partitions. Moscow: Nauka, 1982. 256 p. (in Russian).

7. Postnikov A.G. Introduction to Analytic Number Theory. Moscow: Nauka, 1971. 416 p. (in Russian).

8. Lipsky V. Combinatorics for Programmers. Moscow: Mir, 1988. 200 p. (in Russian).

9. Anderson J.A. Discrete Mathematics with Combinatorics. Moscow: Williams, 2004. 960 p. (in Russian).

10. Kelley C.T. Iterative Methods for Optimization. University City, Philadelphia: Society for Industrial and Applied Mathematics, 1999. 196 p. DOI: 10.1137/1.9781611970920.

11. Samoilov A.A. Study of partitions with additional conditions. URL: https://github.com/SashaSamoilov/qop-partitions (accessed: 25.09.2023).