МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра компьютерной алгебры и теории чисел

Быстрое умножение

АВТОРЕФЕРАТ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

студента 4 курса 421 группы

направление 02.03.01 — Математика и компьютерные науки

механико-математического факультета

Жукова Алексея Сергеевича

Научный руководитель
зав. каф., к.ф.-м.н., доцент А.М. Водолазов

Зав. кафедрой
зав. каф., к.ф.-м.н., доцент А.М. Водолазов

Саратов 2020

Введение. Вычисление произведения двух n-битных целых чисел является важной проблемой в алгоритмической теории чисел и алгебре. Наивный подход приводит к алгоритму, который использует $O(n^2)$ битовых операций. Карацуба показал, что некоторые операции умножения такого алгоритма могут быть заменены менее дорогостоящими операциями сложения, что сокращает общее время выполнения алгоритма до $O(n^{\log_2 3})$ битовых операций. Вскоре после этого этот результат был улучшен Тоомом , который показал, что для любого $\epsilon > 0$ целочисленное умножение может быть выполнено за $O(n^{1+\epsilon})$ время. Это привело к вопросу о том, можно ли дополнительно улучшить временную сложность, заменив член $O(n^{\epsilon})$ на полилогарифмический фактор. В крупном прорыве Шонхаге и Штрассен дали два эффективных алгоритма для умножения целых чисел с использованием быстрого полиномиального умножения. Один из алгоритмов достигал времени выполнения $O(n \cdot \log n \cdot \log\log n \ldots 2^{O(\log^* n)})$, используя арифметику над комплексными числами (аппроксимированную с подходящей точностью), тогда как другой использовал арифметику по модулю тщательно выбранных целых чисел для улучшения сложность дальше до $O(n \cdot \log n \cdot \log\log n)$. Несмотря на многие усилия, модульный алгоритм оставался лучшим до недавнего замечательного результата Фюрера. Фюрер дал алгоритм, который использует арифметику над комплексными числами и работает за $O(n \cdot \log n \cdot 2^{O(\log^* n)})$ времени. На сегодняшний день это лучший результат сложности времени, известный как целочисленное умножение.

Шонхаге и Штрассен представили два, казалось бы, разных подхода к целочисленному умножению - использование сложной и модульной арифметики. Алгоритм Фюрера улучшает сложность времени в сложной арифметической настройке, умело сводя некоторые дорогостоящие умножения к простому сдвигу. Однако алгоритм должен аппроксимировать комплексные числа с определенной точностью во время вычислений. Это вводит дополнительную задачу ограничения общих ошибок усечения при анализе алгоритма. Наоборот, в модульной настройке анализ ошибок практически отсутствует. Кроме того, модульная арифметика дает дискретный подход к дискретной задаче, такой как умножение целых чисел.

Актуальность данной работы обуславливается недавними открытиями. Алгоритм Мартина Фюрера был разработан в 2007 году и до недавнего времени являлся самым оптимальным и самым быстрым. В 2019 году, австралийские математики Дейвид Харви и Йорисом ван дер Ховеном опубликовали новый алгоритм умножения, достигающий значения $O(n \log n)$. Также, проблема быстрого умножения актуальна, так как произведение двух больших чисел занимает много времени и компьютерной памяти. Использование оптимального алгоритма быстрого умножения приводит к снижению затрат и к возможности новых исследований, например, возможности поиска новых простых чисел Мерсенна и вычисления цифр числа π.

Целью данной бакалаврской работы является изучение известных алгоритмов быстрого умножения и реализация одного из них на языке программирования C++. Кроме того, будет приведено сравнение реализованного алгоритма с наивным алгоритмом произведения. Для достижения данной цели будут рассмотрены вспомогательные теоремы и свойства, на которые опираются алгоритмы.

Для достижения поставленной цели, необходимо решить несколько задач:

1. Рассмотреть дискретное преобразование Фурье, а также вспомогательные теоремы, леммы и их свойства;

2. Подробно изучить наиболее значимые алгоритмы быстрого умножения;

3. Реализовать изученный алгоритм на языке программирования C++.

Основное содержание работы содержит 5 разделов:

1. Введение

2. Быстрое преобразование Фурье.

3. Быстрое умножение.

4. Алгоритм Фюрера.

5. Заключение

Во **введении** формулируется цель работы и решаемые задачи.

В **первом разделе** рассматривается дискретное преобразование Фурье и обратное к нему, алгоритм БПФ, а также БПФ при использовании битовых операций.

Преобразование Фурье возникает во многих задачах теоретического и прикладного характера. Во многих приложениях бывает удобно преобразо-

вать задачу в другую, более легкую. Примером служит вычисление произведения двух полиномов. С вычислительной точки зрения целесообразно сначала применить линейное преобразование к векторам коэффициентов полиномов, затем над образами коэффициентов выполнить операцию, более простую, чем свертка, и, наконец, к результату применить обратное преобразование, чтобы получить искомое произведение. В данном случае подходящим линейным преобразованием будет дискретное преобразование Фурье.

Если преобразование Фурье проводится для упрощения вычисления свертки, то часто нужен точный результат. Если элементы берутся из кольца вещественных чисел, их надо аппроксимировать с помощью конечного числа разрядов, и это порождает ошибки. Избежать этих ошибок можно, если производить вычисления в конечном поле. При использовании конечного поля часто бывает трудно выбрать подходящее поле с подходящим корнем n-й степени из единицы. Поэтому, будем пользоваться кольцом R_m целых чисел по модулю m, где m будет таким, чтобы в R_m был примитивный корень n-й степени из единицы. К тому же, если n-степень числа 2, то подходящее число m существует всегда, и оно равно примерно 2^n. В частности, покажем, что когда n и $\omega > 1$ являются степенями числа 2, можно вычислять свертки в кольце вычетов по модулю $\omega^{n/2} + 1$ с помощью преобразования Фурье, покомпонентного умножения и обратного преобразования.

Алгоритм БПФ представим следующим образом:

На входе: Вектор $a = [a_0, a_1, \ldots, a_{n-1}]^T$, где $n = 2^k$ для некоторого числа k.

На выходе: Вектор $F(a) = [b_0, b_1, \ldots, b_{n-1}]^T$, где $b_i = \sum_{j=0}^{n-1} a_j \omega^{ji}$ для $0 \leq i < n$.

1. Пусть $r_{0k} = \sum_{j=0}^{n-1} a_j x^j$.

Полином представлен своими коэффициентами, так что в строке 1 не производится никаких вычислений. Остаток от деления полинома $\sum_{j=0} a_j x^j$ на q_{lm} будет представлен полиномом r_{lm}.

2. **for** $m = k - 1$ **step** -1 **until** 0 **do**

3. **for** $l = 0$ **step** 2^{m+1} **until** $n - 1$ **do**

begin

4. пусть $r_{l,m+1} = \sum_{j=0}^{2^{m+1}-1} a_j x^j$

Вычисляем остатки меньшей степени, используя коэффициенты полинома $r_{i,m+1}$

5. $s = rev(l/2^m)$

6. $r_{lm}(x) = \sum_{j=0}^{2^m-1} (a_j + \omega^s a_{j+2^m}) x^j$

7. $r_{l+2^m,m}(x) = \sum_{j=0}^{2^m-1} (a_j + \omega^{s+n/2} a_{j+2^m}) x^j$

end

8. **for** $l = 0$ **until** $n-1$ **do** $b_{rev(l)} = r_{l0}$

Модификация выше описанного алгоритма для вычисления обратных преобразований состоит в замене ω на ω^{-1} (для этого в строках 6 и 7 меняются знаки показателей степеней элемента ω). Кроме того, в строке 8 $b_{rev(l)}$ делится на n.

Во **втором разделе** рассматриваются алгоритмы быстрого умножения.

Алгоритм «Разделяй и властвуй» (или алгоритм Карацубы) — метод быстрого умножения, который позволяет перемножать два n-значных числа с битовой вычислительной сложностью: $O(n^{\log_2 3})$, $\log_2 3 = 1{,}5849\ldots$.

Метод основан на разбиении двух n-значных чисел x и y на две равные части в соответствии с рисунком 1, замене дорогостоящих операций умножения на сложение и разность, такая замена приводит к асимптотической эффективности, т. к. умножение выполнять труднее, чем сложение, и для достаточно больших чисел любое фиксированное число n-разрядных сложений требует меньше времени, чем n-разрядное умножение.

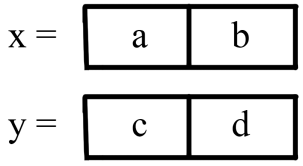

Рисунок 1 — Разбиение чисел на две равные части

Сам алгоритм выглядит так:

$$u = (a + b) * (c + d);$$
$$v = a * c;$$
$$w = b * d$$
$$z = v * 2^n + (u - v - w) * 2^{n/2} + w.$$

Схема для умножения двух n-разрядных чисел требует только трех умножений $(n/2)$-разрядных чисел и нескольких сложений и сдвигов. Для вычисления произведений u, v и w можно применять этот алгоритм рекурсивно.

На практике алгоритм становится эффективнее обычного умножения при умножении чисел длиной порядка сотен двоичных (десятков десятичных) разрядов, на числах меньшей длины алгоритм не даёт существенного преимущества из-за большего, чем в наивном алгоритме, числа требуемых сложений, вычитаний и сдвигов.

Алгоритм Шенхаге — Штрассена — асимптотически быстрый алгоритм умножения больших чисел основанный на быстром преобразовании Фурье. Он был разработан Арнольдом Шёнхаге и Фолькером Штрассеном в 1971 году. Сложность алгоритма составляет $O(N \log N \log\log N)$ битовых операций, где N - количество двоичных цифр в произведении. Алгоритм рекурсивно использует быстрое преобразование Фурье над кольцом из $2^{2^n} + 1$ элементов, специальный тип дискретного преобразования Фурье - теоретико-числовое преобразование.

Сам алгоритм выглядит так:

На входе два $n-$разрядных двоичных целых числа u и v, где $n = 2^k$.

На выходе получаем $(n+1)$-разрядное произведение uv по модулю $2^n + 1$.

Метод: если n мало, нужно умножить u на v по модулю $2^n + 1$. Для большого n положим $b = 2^{k/2}$, если k четно, и $b = 2^{(k-1)/2}$, если k нечетно. Пусть $l = n/b$. Представим u и v в виде $\sum_{i=0}^{b-1} u_i 2^{li}$ и v в виде $\sum_{i=0}^{b-1} v_i 2^{li}$, где u_i и v_i — целые числа между 0 и $2^l - 1$ (т.е. числа u_i — это блоки, составленные из l разрядов числа u, аналогично v_i — блоки из l разрядов числа v).

1. Вычисляем преобразование Фурье по модулю $2^{2l}+1$ векторов

$$[u_0, \psi u_1, \ldots, \psi^{b-1}u_{b-1}]^T \quad \text{и} \quad [v_0, \psi v_1, \ldots, \psi^{b-1}v_{b-1}]^T,$$

где $\psi = 2^{2l/b}$ и ψ^2 берется в качестве примитивного корня b-й степени из единицы.

2. Вычисляем по модулю $2^{2l}+1$ покомпонентное произведение преобразований Фурье, полученных на шаге 1, при помощи данного алгоритма, который рекурсивно применяется для вычисления произведения каждой пары соответствующих компонент.

3. Вычисляем обратное преобразование Фурье по модулю $2^{2l}+1$ вектора, равного покомпонентному произведению, полученному на шаге 2. Результатом будет вектор $[\omega_0, \psi\omega_1, \ldots, \psi^{b-1}\omega_{b-1}]^T$ по модулю $2^{2l}+1$, где ω_i обозначает i-ю компоненту отрицательно обернутой свертки векторов $[u_0, u_1, \ldots, u_{b-1}]^T$ и $[v_0, v_1, \ldots, v_{b-1}]^T$. Вычисляем $\omega_i'' = \omega_i$ по модулю $2^{2l}+1$, умножая $\psi^i\omega_i$ на ψ^{-i} по модулю $2^{2l}+1$.

4. Вычисляем $\omega_i' = \omega_i \ mod \ b$ следующим образом.

(а) Пусть $u_i' = u_i$ по модулю b и $v_i' = v_i$ по модулю b для $0 \leqslant i < b$.

(б) Построим числа $\hat{u}$ и $\hat{v}$, выписывая в цепочки числа u_i' и v_i' и вставляя между ними $2\ \log b$ нулей, т.е.

$$\hat{u} = \sum_{i=0}^{b-1} u_i' 2^{(3\log b)i} \quad \text{и} \quad \hat{v} = \sum_{i=0}^{b-1} v_i' 2^{(3\log b)i}.$$

(в) Вычисляем произведение $\hat{u}\hat{v}$ с помощью алгоритма из раздела 2.1.

(г) Произведение $\hat{u}\hat{v}$ имеет вид $\sum\limits_{i=0}^{2b-1} y_i' 2^{(3\log b)i}$, где $y_i' = \sum\limits_{j=0}^{2b-1} u_j' v_{i-j}'$. Числа ω_i по модулю b можно восстановить по этому произведению, вычисляя $\omega_i' = y_i' - y_{b+i}'$ по модулю b для $0 \leqslant i < b$.

5) Вычисляем точные значения ω_i по формуле

$$\omega_i = (2^{2l}+1)((\omega_i' - \omega_i'') \ mod \ b) + \omega_i'',$$

учитывая, что ω_i лежит между $-(b-1-i)2^{2l}$ и $(i+1)2^{2l}$.

6) Вычисляем $\sum_{j=0}^{b-1} \omega_i 2^{li}$ по модулю $(2^n + 1)$. Это и есть искомый результат.

Приложения алгоритма Шенхаге — Штрассена включают в себя как чисто математические задачи, такие как поиск простых чисел Мерсенна и вычисление цифр числа π, так и практические, такие как вычисление подстановки Кронекера, в которой умножение многочленов с целыми коэффициентами можно эффективно заменить умножением больших чисел, это используется на практике для факторизации целых чисел методом эллиптических кривых Ленстры.

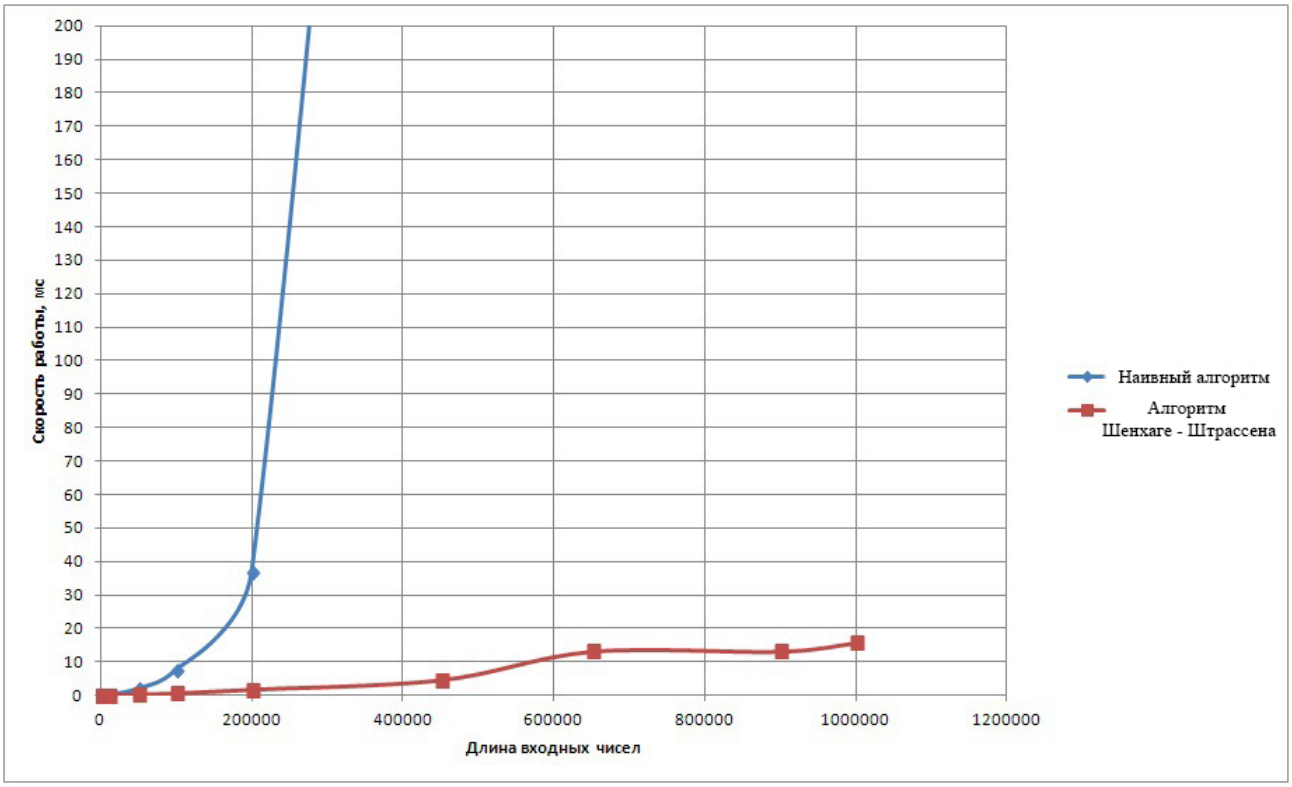

Рисунок 2 — Сравнение наивного алгоритма и алгоритма Шенхаге — Штрассена

Третий раздел посвящен алгоритму Фюрера, который является улучшенной версией алгоритма Шенхаге — Штрассена.

Для умножения двух неотрицательных целых чисел по модулю $2n + 1$ представим их как полиномы от $R[y]$, где $R = C[x]/(x^P + 1)$, и умножаем эти полиномы с помощью преобразования Фурье следующим образом. Пусть $P = O(\log n)$ округляется до степени 2. Бинарные целые числа, которые нужно умножить, разлагаются на (большие) куски длины $P^2/2$. Опять же, каждый

такой кусок разлагается на маленькие кусочки длины P. Если $a_{iP2?1}, \ldots, a_{i0}$ - маленькие кусочки, принадлежащие общему большому кусочку a_i тогда они представляются как

$$\bar{a_i} = \sum_{j=0}^{P-1} a_{ij} x^j \in R = C[x]/(x^P + 1)$$

где

$$a_{iP-1} = a_{iP-2} = \cdots = a_{iP/2} = 0$$

Таким образом, каждый большой кусок представляется как элемент R, который является коэффициентом полинома от y.

Эти элементы R сами являются полиномами от x. Их коэффициенты являются целыми числами в начале и в конце алгоритма. Промежуточные результаты, а также корнями единицы являются многочлены с комплексными коэффициентами, которые сами по себе представлены парами действительных чисел, которые должны быть аппроксимированы численно.

Теперь каждый коэффициент $\sum_{i=0}^{N-1} \bar{a}_i 2^{P^2/2}$ представлен полиномом $\sum_{i=0}^{N-1} a_i y^i$. БПФ вычисляет значения такого многочлена в этих корнях, которые являются нечетными степенями корня из единицы $\rho(x) \in R$, определенных в разделе **??**. Значения умножаются, и обратное БПФ производит другой многочлен от $R[y]$. Из этого полинома полученное целое произведение может быть восстановлено просто делая несколько простых дополнений. Соответствующие части коэффициентов теперь возрастают до некоторой длины $O(P)$ от начальной длины P. (Постоянный фактор роста может быть фактически уменьшен до коэффициента $2 + o(1)$ путем увеличения параметра P из $O(\log n)$ в $O(\log^2 n)$, но это будет влиять только на постоянный коэффициент перед $\lg^*$ в показателе времени выполнения.)

Таким образом, алгоритм работает почти так же, как алгоритм Шонхаге — Штрассена, за исключением того, что поле C или кольцо целых чисел по модулю m-ого простого числа Ферма F_m был заменен кольцом $R = C[x]/(x^P + 1)$, и БПФ разлагается более равномерно. Стандартное разложение N-точечного БПФ на два $N/2$-точечных БПФ и более 2-х точечное БПФ не позволят такое улучшение. Тем не менее, нет необходимости для

балансировки полностью. Вместо рекурсивного разложения $N = O\left(\frac{n}{\log^2 n}\right)$ точечное БПФ в середине (разделяй и властвуй), разлагаем на $2P$-точки БПФ и $N/(2P)$ -точечные БПФ. В основном это сделано для простоты. Обе версии эффективный (хотя различия в постоянных коэффициентах $\lg^*$ в экспоненте), а только для каждого $\log P$-ого уровня общего БПФ требуется сложное умножение с трудными корнями единицы. На всех других уровнях коэффициенты являются полиномами . Умножения с этими двойными коэффициентами являются просто циклическими вращениями P-векторов из коэффициентов элементов R, с изменением знака на обратный.

После вычисления преобразования Фурье над кольцом R, элементы из R являются полиномами над C по модулю x^P+1. Коэффициенты представлены парами вещественных чисел с фиксированной точностью для вещественной и мнимой частей. Нужно знать числовую точность, необходимую для коэффициентов этих многочленов. После двух параллельных БПФ и умножения соответствующих значений с последующим обратным БПФ, видно, что результат снова имеет целочисленные коэффициенты. Следовательно, точность должна быть такой, чтобы в конце абсолютные погрешности были меньше $1/2$. Значит, набор операций окончательного округления дает правильный результат.

Заключение. В данной бакалаврской работе были рассмотрены такие алгоритмы быстрого умножения как: «Разделяй и властвуй», алгоритм Шенхаге — Штрассена и алгоритм Фюрера. Также, были рассмотрены вспомогательные темы, а именно быстрое преобразование Фурье (БПФ) и обратное к нему, БПФ при использовании битовых операций. Сформулированны и доказаны теоремы, которые фигурируют в перечисленных выше алгоритмах, а также в оценке их точности и сложности.

Приведена реализация алгоритма Шенхаге — Штрассена на языке программирования C++, а также, в виде графика, приведено сравнение данного алгоритма с наивным.