Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

АЛГОРИТМЫ ДЛИННОЙ АРИФМЕТИКИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента(ки) 4 курса 441 группы
направления (специальности) 02.03.03Математическое обеспечение и
администрирование информационных систем
факультета компьютерных наук и информационных технологий
Пожидаевой Ольги Леонидовны

Научный руководитель
старший преподаватель кафедры
информатики и программирования _____ Е.Е. Лапшева
должность, уч. степень, уч. звание дата, подпись инициалы, фамилия

Заведующий кафедрой
кандидат физико-математических
наук, доцент кафедры информатики
и программирования _____ А.Г. Федорова
должность, уч. степень, уч. звание дата, подпись инициалы, фамилия

Саратов 2016 год

ВВЕДЕНИЕ

Быстрые алгоритмы – это область вычислительной математики, которая изучает алгоритмы вычисления заданной функции с заданной точностью с использованием как можно меньшего числа битовых операций. Такие алгоритмы, реализованные на ЭВМ в программном (а иногда и аппаратном) обеспечении, позволяют существенно увеличить производительность работы компьютера, а иногда и решить задачи, размер которых не позволял найти решение путём применения обычных методов вычисления.

Целью дипломной работы является изучение и реализация алгоритмов длинной арифметики. А также реализация некоторых алгоритмов не только в последовательной, но и параллельной версии, и сравнение работы написанных алгоритмов по быстродействию.

Алгоритмы длинной арифметики используются в криптографии, в математическом и инженерном программном обеспечении, требующем сверхвысокой точности, например, при расчете траектории движения и необходимой силы тяги космических носителей. Так же они применяются в бухгалтерии для точного подсчета денежных и других средств. К тому же длинная арифметика используется в олимпиадном (спортивном) программировании.

Исходя из указанной цели, можно выделить частные задачи, поставленные в дипломной работе:

1. обзор алгоритмов длинных чисел;
2. разработка и реализация основных алгоритмов на языке C++;
3. анализ алгоритмов перемножения длинных чисел;
4. рассмотрение основных алгоритмов умножения в параллельном представлении;
5. тестирование и анализ программ с использованием алгоритмов длинных чисел.

Для решения поставленных задач в дипломной работе использовались следующие методы исследования:

1. теоретический анализ учебных пособий по информатике;
2. теоретический анализ пособий по математике и олимпиадной информатике;
3. практическая проверка выполнения программ с использованием изученных алгоритмов;
4. сравнение времён выполнения алгоритмов в последовательном и параллельном вариантах.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Дипломная работа состоит из введения, четырех разделов, разделенных на подразделы, заключения, списка использованных источников и двадцати одного приложения.

В первом разделе «Основные понятия» рассказывается, почему необходима длинная арифметика, а также говорится, что «длинными» числами называются числа, для представления которых в стандартных компьютерных типах данных не хватает количества двоичных разрядов, и что реализация арифметических операций над такими «длинными» числами получила название «длинной арифметики».

Ко всему прочему в данном разделе также описываются возможные проблемы, которые могут возникнуть при работе с длинными числами. Например, возникают как проблемы их представления, так и проблемы проведения операций над ними.

В подразделе 1.1 «Представление длинных чисел» говорится о возможном представлении длинных чисел в виде массивов, в которых хранение чисел происходит в обратном порядке, что удобно при проведении операций на длинных числах.

Во втором разделе «Алгоритмы длинной арифметики» происходит разделение алгоритмов на четыре группы: алгоритмы сложения, вычитания, деления и умножения. В подразделах данного раздела подробно рассматриваются некоторые алгоритмы. Так, в подразделе «Алгоритмы сложения и вычитания длинных чисел» описывается алгоритм сложения, который реализуется, как обычное сложение чисел столбиком. При этом здесь уточняется, почему числа удобнее хранить в обратном порядке, а не прямом. А также рассказывается об алгоритме вычитания и о важности знания длин чисел в данном алгоритме. Это необходимо для определения знака получившегося числа (результата), то есть, если первое число меньше второго, то в ответе появится минус. Поэтому, если при вычитании длина

чисел заранее не известна, следует сначала найти, какое из чисел больше и только после этого производить вычисления для получения корректного результата.

Во втором подразделе «Деление длинных чисел» подробно описывается алгоритм деления, приводится его схема и предоставляется пример его работы на двух числах. В основу данного алгоритма ложится идея того, что мы делим не сразу все число u на число v, а поэтапно, каждый раз находя в цикле разряд частного, при этом проверяя остаток от деления, который должен быть меньше делителя.

Подраздел «Алгоритмы умножения длинных чисел» разделяется на пять частей, в которых рассматриваются алгоритмы (методы) умножения. В первой рассказывается о базовом алгоритме, который представляет собой алгоритм по школьному методу «в столбик». Перемножение двух n – значных целых чисел обычным школьным методом «в столбик» сводится, по сути, к сложению n – значных чисел. Допустим, необходимо умножить x и y. Тогда нужно умножить x на каждую цифру числа y и сложить результаты с соответствующими сдвигами.

Во второй части данного подраздела говорится о методе Аль-Хорезми. Его суть состоит в том, чтобы записать множители x и y рядом. После чего повторять следующие действия: поделить первое число пополам, отбросив дробную часть $\frac{1}{2}$ (если число было нечётным), а второе число удвоить. При этом делать это нужно до тех пор, пока первое число не станет единицей. После этого необходимо вычеркнуть все строки, в которых первое число чётно, и сложить оставшиеся числа из второй колонки, получив тем самым ответ.

В третьей части «Быстрый столбик» говорится об алгоритме, который, по сути, разновидность умножения в «столбик» и требует такое же количество операций сложения и умножения, что и базовый алгоритм, но в

другом порядке.

Для очень больших целых чисел A и B можно построить алгоритм умножения более быстрый, чем классический. Идея этого способа принадлежит А. Карацубе. Анатолий Алексеевич Карацуба – советский и российский математик. Создатель первого быстрого метода в истории математики – метода умножения больших чисел (умножение Карацубы, данный алгоритм рассматривается в 2.3.4 «Метод Карацубы»).

Алгоритм Карацубы – метод быстрого умножения со сложностью вычисления $n^{\log_2 3}$. В то время как базовый алгоритм умножения в столбик, требует n^2 операций. Но следует заметить, что при длине чисел короче нескольких десятков знаков быстрее работает обычное умножение.

Суть данного алгоритма в том, что каждое из двух чисел, которые необходимо перемножить, можно представить в виде суммы их двух частей, половинок длиной $k = \frac{n}{2}$. После чего высчитать три промежуточных коэффициента C_0, C_1 и C_2 и подставить получившиеся числа в формулу $A \cdot B = C_0 + C_2 \cdot M^k + C_1 \cdot M^{2k}$.

Последним в подразделе «Алгоритмы умножения длинных чисел» рассматривается алгоритм Toom-Cook 3-way. Идея данного алгоритма заключается в разделении входных данных (множителей) на 3 равные части, после чего высчитываются коэффициенты, которые будут складываться друг с другом с необходимым сдвигом для получения результата.

Подраздел «Параллельные алгоритмы умножения длинных чисел» также разделяется на несколько частей. В «Общих понятиях параллельных задач» говорится о том, какие задачи выполняются параллельно. А также упоминается о целях параллелизма.

Цель технологий параллелизма – обеспечить условия, позволяющие компьютерным программам делать больший объем работы за тот же интервал времени. Поэтому проектирование программ должно ориентироваться не на выполнение одной задачи в некоторый промежуток времени, а на

одновременное выполнение нескольких задач, на которые предварительно должна быть разбита программа. Возможны ситуации, когда целью является не выполнение большего объема работы в течение того же интервала времени, а упрощение решения с точки зрения программирования. Иногда имеет смысл думать о решении проблемы как о множестве параллельно выполняемых задач. Обычно не слишком полезно (или эффективно) выполнять одну подзадачу в один период времени, а другую – совершенно в другой. Именно параллельность обоих процессов дает естественную форму искомого решения проблемы. Иногда к параллельности прибегают, чтобы увеличить быстродействие программы или приблизить момент ее завершения. В других случаях параллельность используется для увеличения продуктивности программы (объема выполняемой ею работы) за тот же период времени при вторичности скорости ее работы. Например, для некоторых Web-сайтов важно как можно дольше удерживать пользователей. Поэтому здесь имеет значение не то, насколько быстро будет происходить подключение (регистрация) и отключение пользователей, а сколько пользователей сможет этот сайт обслуживать одновременно. Следовательно, цель проектирования программного обеспечения такого сайта – обрабатывать максимальное количество подключений за как можно больший промежуток времени. Наконец, параллельность упрощает само программное обеспечение. Зачастую сложную последовательность операций можно упростить, организовав ее в виде ряда небольших параллельно выполняемых операций. Независимо от частной цели (ускорение работы программ, обработка увеличенной нагрузки или упрощение реализации программы), главная цель – усовершенствовать программное обеспечение, воспользовавшись принципом параллельности.

Параллельное выполнение набора задач можно обеспечить с помощью concurrency::parallel_invoke, о чем и говорится в 2.4.2 «Parallel_invoke».

В 2.4.3 и 2.4.4 рассказывается, как именно можно распараллелить базовый алгоритм и метод Карацубы. Для реализации базового алгоритма в

параллельной версии достаточно разделить второй множитель на части равной длины, чтобы после вычислять слагаемые параллельно. В методе Карацубы параллельно можно вычислять части чисел, а именно A_0, A_1, B_0, B_1.

Третий раздел «Олимпиадное программирование» посвящен одной из областей программирования. В данном разделе описывается, что требуется от участника во время соревнований, а также на что следует обратить внимание, чтобы не потерять баллы. Так, например, выделяется шесть типов ошибок:

1. Wrong answer (неверный ответ). Эта ошибка означает, что результат работы программы не совпадает с ответом жюри, а значит, скорее всего, была допущена алгоритмическая ошибка в программе или же использован неверный формат представления выходных данных.
2. Time limit exceeded (превышен указанный в задаче лимит времени). Программа выполняется дольше установленного времени. Возможные причины этого: неэффективное решение или ошибка в программе.
3. Presentation Error (отсутствие выходного файла output.txt). Любая олимпиадная задача подразумевает входные и выходные данные. Т.е. в формулировке задания обязательным образом описывается формат входных и выходных данных, а программа участника олимпиады должна считать эти данные, обработать и вывести результат в установленном формате. Чаще всего чтение происходит из некоторого файла input.txt, а вывод в некоторый файл output.txt. Если же выходной файл не создан, написано неверное имя файла или происходит сбой программы до открытия выходного файла, тогда возникает ошибка presentation error.
4. Compilation error (ошибка компиляции). В результате компиляции не создан исполняемый файл. Причиной данной ошибки может служить синтаксическая ошибка в программе или неверно указано расширение файла.

5. Memory limit exceeded (превышен указанный в задаче лимит памяти). Программа использует больше установленного размера памяти. Чаще всего данная ошибка возникает при неэффективном алгоритме.
6. Runtime error (ошибка исполнения). В случае этой ошибки результат работы программы не проверяется, т.к. она завершила работу с ненулевым кодом возврата. Возможно, в программе произошло обращение к несуществующему элементу массива, деление на ноль. Если программа написана на языке C++, стоит ли проверить завершается ли она «return 0».

В подразделе 3.1 приводятся примеры олимпиадных задач, в которых используется длинная арифметика.

Раздел 4 отведен практической части, в которой были реализованы некоторые алгоритмы длинной арифметики на языке C++ в последовательной версии, а также реализованы алгоритмы перемножения длинных чисел в параллельном представлении. После чего было проведено сравнение работы алгоритмов, которые были представлены как в последовательной, так и в параллельной версиях.

В данном разделе на диаграммах видно, что в последовательной работе обычное умножение работает быстрее на коротких числах, а метод Карацубы в остальных случаях, особенно это видно на длинных числах. В параллельной версии умножение столбиком также проигрывает методу Карацубы. Если же сравнивать алгоритмы в различных версиях, то умножение столбиком будет работать быстрее в последовательной версии только на небольших числах, а метод Карацубы будет всегда работать быстрее в параллельной версии.

В приложениях предоставлены реализации алгоритмов.

ЗАКЛЮЧЕНИЕ

В дипломной работе были изучены алгоритмы длинной арифметики и реализованы некоторые из них на языке программирования C++. Теоретически были рассмотрены алгоритмы сложения, вычитания, деления и умножения. А именно:

1. сложение, вычитание и умножение столбиком;
2. метод Аль-Хорезми, метод Карацубы;
3. умножение быстрым столбиком и алгоритм Toom-Cook-3-way.

В ходе реализации алгоритмов умножения было проведено их сравнение между собой. А также сравнение работы метода Карацубы и умножения в столбик в последовательной и параллельной версиях.

В процессе изучения и анализа алгоритмов умножения было выяснено, что базовый алгоритм будет работать быстрее с более маленькими числами, нежели метод Карацубы. В свою очередь этот метод выгодно использовать с большими числами приблизительно одинаковой длины. Также распараллеливая алгоритмы длинной арифметики можно в разы выигрывать во времени их выполнения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бекман, И. Н. Компьютерные науки: курс лекций. Л. 7. Алгоритмы / И. Н. Бекман. МГУ, 2011. С. 7.
2. Алгоритмы и фундаментальное программирование в СГУ. [Электронный ресурс]: архив задач. URL: http://acm.sgu.ru/univer/problemset.php (дата обращения: 03.06.2016).
3. Окулов, С. М. «Длинная арифметика» / С. М. Окулов // Журн. Информатика. М.: «Первое сентября», 2000, № 4. С. 19 – 23.
4. Ахо, А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. М.: Мир, 1979. С. 284.
5. Карацуба, А. А. «Сложность вычислений» / А. А. Карацуба //

Оптимальное управление и дифференциальные уравнения: сборник статей к семидесятилетию со дня рождения академика Евгения Фроловича Мищенко, М.: МИАН, Наука, Физматлит, 1995, Т. 211. С. 186 – 202.

6. Статья «Умножение длинных чисел методом Карацубы» [Электронный ресурс]. URL: http://habrahabr.ru/post/124258/ (дата обращения: 07.03.2016).
7. Статья «Длинная арифметика от Microsoft» [Электронный ресурс]. URL: https://habrahabr.ru/post/207754 (дата обращения: 07.03.2016).
8. Дасгупта, С. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани. М.: МЦНМО, 2014. С. 15 – 19.
9. Статья «Длинная арифметика в C++» [Электронный ресурс]. URL: http://cppstudio.com/post/5036/ (дата обращения: 08.04.2016).
10. Кнут, Д. Э. Искусство программирования. В 2 т. Т. 2 / Д.Э. Кнут. Вильямс, 2001. С. 294 – 304.
11. Окулов, С. М. Алгоритмы компьютерной арифметики/ С. М. Окулов, А. В. Лялин, О. А. Пестов, Е. В. Разова. М.: БИНОМ. Лаборатория знаний, 2015. С. 9 – 186.
12. Архипов, Г. И. О математических работах профессора А. А. Карацубы / Г. И. Архипов, В. Н. Чубариков. М.: МИАН, 1997, Т. 218. С 7 – 19.
13. Гриценко, С. А. Научные достижения Анатолия Алексеевича Карацубы. Математика и информатика / С. А. Гриценко, Е. А. Карацуба, М. А. Королёв, И. С. Резвякова, Д. И. Толев, М. Е. Чанга // Современные проблемы математики, М.: МИАН, 2012, Т. 16. С. 7–30.
14. Крэндалл, Р. Простые числа. Криптографические и вычислительные аспекты / Крэндалл Р., Померанс К. М.: УРСС: Книжный дом «ЛИБРОКОМ», 2011. С. 531 – 535.
15. Хьюз, К. Параллельное и распределенное программирование на C++ / К. Хьюз, Т. Хьюз. 2004. С. 25 – 29.

16. Эндрюс, Г.Р. Основы многопоточного, параллельного и распределенного программирования / Г. Р. Эндрюс. Вильямс, 2003. С. 17 – 20.
17. Статья «Параллельные алгоритмы» [Электронный ресурс]. URL: https://technet.microsoft.com/ru-ru/dd470426(v=vs.90).aspx (дата обращения: 08.04.2016).
18. Статья «Распараллеливание алгоритмов умножения чисел многократной точности» [Электронный ресурс]. URL: http://old.ugatu.ac.ru/publish/vu/stat/ugatu-2011-5(45)/19.pdf (дата обращения: 03.04.2015).
19. Статья «Олимпиадное программирование как искусство» [Электронный ресурс]. URL: http://codeforces.com/blog/entry/2100 (дата обращения: 07.06.2016).
20. Статья «Функция parallel_invoke» [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/dd504887.aspx (дата обращения: 10.05.2015).