МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет прикладной математики, информатики и механики
Кафедра программного обеспечения и администрирования
информационных систем

Сравнительный анализ алгоритмов квантовых вычислений с их классическими аналогами

Магистерская диссертация
Направление 02.04.03. Математическое обеспечение и администрирование информационных систем
Программа Информационные технологии

Зав. кафедрой _____ д. ф.-м. н., проф. Артемов М. А. __.__.2019

Обучающийся _____ Черниговских Р. И.

Руководитель _____ к.ф.-м.н., доц. Барановский Е. С.

Воронеж 2019

Оглавление

Введение

Квантовые вычисления и обработка квантовой информации в настоящий момент являются дисциплинами, находящимися только в стадии становления. Данная работа посвящена проведению анализа наиболее известных и перспективных алгоритмов квантовых вычислений. В рамках данной работы реализованы, как и квантовые алгоритмы, так и классические для решения аналогичных задач. В первой главе рассмотрены такие понятия как квантовый компьютер, квантовое состояние, кубит, описаны преимущества и недостатки квантовых вычислений, а также сформулированы задачи теории квантовых вычислений. Во второй главе описаны алгоритмы, рассматриваемые в работе. Также в этой главе приводится постановка задачи, ее содержательное и формализованное описание. В третьей главе представлены средства реализации, требования к аппаратному и программному обеспечению, а также непосредственно реализация квантовых и классических алгоритмов.

Глава 1. Анализ задачи

1.1. История развития квантовых вычислений

Современные компьютеры – и в теории (Машины Тьюринга), и на практике, основаны на классической физике. Они ограничены, тем фактом, что система может быть только в одном состоянии. Используя современные знания в квантовой физике, можем предложить концепцию квантовой системы, которая может находится в суперпозиции нескольких различных состояний одновременно. Более того, территориально разделенные квантовые системы могут быть переплетены друг с другом и благодаря этому, операции перестают быть локальными.

Квантовые вычисления рассматривают вычислительные мощности и другие свойства компьютеров, спроектированные на основе принципов квантовой механики. Основная задача состоит в поиске квантовых алгоритмов, которые значительно быстрее любого классического алгоритма, решающего такую же проблему. Впервые формальная модель универсального квантового компьютера была предложена П. Бениоффом и развита Д. Дойчем.

Квантовое состояние представляет собой суперпозицию классических состояний, которые можно измерить или применить унитарную операцию. Представим, что некая физическая система, которая может быть в N различных классических состояний. Назовем эти состояния |1>, |2>, …,|N>. Грубо говоря, под классическим понимается состояние, в котором система может быть измерена. Квантовое состояние |φ> это суперпозиция классических состояний: $|\varphi\rangle = \alpha_1|1\rangle + \alpha_2|2\rangle + \ldots + \alpha_N|N\rangle$, где α_i – комплексное число. Таким образом, можно сказать, что система в квантовом состоянии находится во всех классических состояниях одновременно. Говоря математическим языком, состояние |1>, ..., |N> формирует ортонормальный базис N-размерного Гильбертова пространства, в котором квантовое состояние |φ> является вектором. С квантовым состоянием можно проводить 2 операции: измерить и изменить унитарно, без измерения.

Как мы уже выяснили, информация может быть закодирована и использована на основе принципов квантовой механики. Задача обработки квантовой информации состоит в решении определенного класса проблем, которых не могут решить классические компьютеры за приемлемое время. [7]

1.2. Преимущества и недостатки квантовых вычислений

Важнейшим преимуществом квантовой системы является квантовый параллелизм – возможность регистра находиться одновременно во всех своих состояниях. Классический n-битный регистр может находиться только в одном состоянии, в то время как n-битный квантовый регистр находится сразу во всех 2^N базисных состояниях. Для получения однозначного ответа для квантовых вычислений необходимо провести преобразование состояния таким образом, чтобы нужный ответ получил бы большую амплитуду, то есть проявился при измерении с большей вероятностью.

Недостатки квантовых вычислений являются продолжением их сильных сторон: ответ можно получить, лишь в результате измерения, которое является вероятностным процессом и приводит к безвозвратной потере информации об амплитудах полученных базисных состояний. Кроме того, в классических алгоритмах можно прервать вычисления, если ответ уже получен. Квантовые алгоритмы всегда выполняются до конца, что также требует специальной организации. Еще одна особенность квантовых вычислений – обратимость используемых преобразований – не представляет проблемы, если нет ограничений на размер квантового регистра. Однако при довольно умеренных ограничениях вычислений многих функций оказывается затруднено, а соответствующие задачи в таких моделях с ограничениями могут иметь экспоненциальную сложность.

Перечислим три основных мотивации изучения квантовых компьютеров:

Процесс миниатюризации, который сделал современные компьютеры мощными и дешевыми, практически достиг микроуровней, на которых проявляются квантовые эффекты. Производители чипов склонны перейти к большим размерам, чтобы подавить эти эффекты, но стоит попробовать сработаться с ними, открывая путь к дальнейшей миниатюризации.

Использование квантовых эффектов позволяет значительно ускорить некоторые вычисления и даже реализовать некоторые вещи, недоступные классическим компьютерам. Основная цель данной работы – демонстрация этих идей.

Одна из задач теории компьютерной науки звучит как «выявить возможности и ограничения самого допустимо-сильного вычислительного устройства, которое может позволить нам природа». [7]

1.3. Физическая реализация квантового компьютера

Первый 2х-кубитный компьютер был построен в 1997, а в 2001 5ти-кубитный компьютер успешно разложил число 15. На текущий момент, самый большой квантовый компьютер имеет несколько десятков кубит.

Хотя, уже доступны компьютеры с небольшим количеством кубит, практическая реализация более мощного квантового компьютера выглядит достаточно трудной задачей. Квантовая система всегда взаимодействует со своим окружением. Это взаимодействие влияет на состояние системы, вызывая потерю данных. Процесс влияния окружения на квантовую систему называется декогерированием. Проблема шумов и декогерирования в теории может быть решена с помощью квантового исправления ошибок и устойчивых алгоритмов вычисления, но эти проблемы все еще не решены на практике. Давид П. ДиВинцензо предложил список необходимых условий (критерии ДиВинцензо) для идентификации системы как квантового компьютера.

Критерии ДиВинцензо:

1. Масштабируемая физическая система с четко выраженными кубитами. Нам необходим квантовый регистр из кубитов для хранения информации, так же, как и классическому компьютеру требуется память. Самый простой способ физически реализовать кубит – использовать 2х уровневую квантовую систему. Для примера, электрон или фотон могут быть кубитами.
2. Возможность инициализировать состояние кубитов как начальное |00…0>. При отсутствии возможности сбросить состояние классического компьютера, нельзя доверять выводам вычислений, даже если процесс прошел корректно.

3. Время на декогерирование значительно длиннее, чем время на простейшую операцию. Декогерирование возможно наибольшая помеха к строительству жизнеспособного квантового компьютера. Декогерирование ведет к деградации многих возможностей квантового состояния из-за взаимодействия с окружающей средой, а также увеличивает время квантовых вычислений. Само по себе декогерирование не так важно. Гораздо важнее соотношение между временем декогерирования и времен на выполнение вентиля.

4. Универсальный набор квантовых вентилей. Допустим, есть классический компьютер с большой памятью. Необходимо управлять данными в этой памяти через различные логические вентили. Должна быть возможность применить независимый логический оператор на эту память, чтобы осуществить информационный процесс.

5. Возможность измерения кубитов. Результат классических вычислений должен быть выведен на экран или распечатан. В то время как процесс вывода на классическом компьютере достаточно тривиальная часть вычислений, это критически важная часть квантовых вычислений. После выполнения квантового алгоритма, состояние должно быть измерено, чтобы получить результат.

6. Возможность взаимопревращать стационарные и изменяющиеся кубиты. Некоторые реализации отличны в хранении квантовой информации, в то время как продолжительные трансформации квантовой информации могут потребовать дополнительных физических ресурсов. Это можно сравнить с обычным компьютером, в котором CPU и память сделаны из полупроводников, в то время как жесткий диск используется как устройство для хранения. Также и рабочий квантовый компьютер может потребовать несколько видов кубитов для представления квантовых вычислений.

7. Возможность надежно передавать изменяющиеся кубиты между разными локациями. Не приходится говорить, что это необходимое требование квантовой коммуникации, такой как распределенный квантовый ключ. Это условие также важно в распределенных квантовых вычислениях. [13]

1.4. Кубиты и квантовая память

В классических вычислениях единицей информации является бит, которые может быть 0 или 1. В квантовых вычислениях используются квантовые биты (кубиты), которые могут быть в суперпозиции 0 и 1. Рассмотрим систему, которая может быть в двух базовых состояниях, назовем их |0> и |1>. Будем идентифицировать эти состояния как векторы (1, 0) и (0, 1) соответственно. Представим кубит как $\alpha_0|0> + \alpha_1|1>$, $|\alpha_0|^2+ |\alpha_1|^2= 1$. Таким образом, единичный кубит живет в векторном пространстве C^2. Система из более одного кубита может находиться в пространстве тензорного произведения нескольких кубитных систем.

В более общем виде, регистр из n кубит может находится в 2^n базовых состояниях, каждое в виде $|b_1> \otimes |b_2> \otimes \ldots \otimes |b_n>$, где $b_i \in \{0, 1\}$, сократим до $|b_1b_2...b_n>$.

Квантовому компьютеру необходимо как минимум 10^2 – 10^3 кубит, для выполнения алгоритмов, более эффективных, чем их классические аналоги. [12]

1.5. Измерение по вычислительному базису

Допустим, мы измеряем состояние $|\varphi>$, в таком случае, мы увидим только одно классическое состояние $|j>$. Конкретное $|j>$ неизвестно заранее, оно проявится с вероятностью $|\alpha j|^2$, что представляет собой квадратичную норму соответствующей частоты a_j ($|a + ib| = \sqrt{a^2 + b^2}$). При этом, после измерения квантовое состояние $|\varphi>$ пропадает и остается классическое состояние $|j>$ с вероятностью $|\alpha_j|^2$, что является квадратичной нормой для соответствующей амплитуды a_j. Таким образом, измерение квантового состояния вызывает вероятностное размещение классических состояний, представленное квадратичными нормами амплитуд. Если мы измерим состояние $|\varphi>$ и увидим классическое состояние $|j>$ в результате, то само $|\varphi>$ исчезнет и останется только $|j>$. Другими словами, измерение $|\varphi>$ сводит квантовую суперпозицию $|\varphi>$ до классического состояния $|j>$, которое мы видим, а вся информация в амплитудах a_j пропадает. [7]

1.6. Унитарные преобразования

Вместо измерения |φ>, также можно применить к нему некоторый оператор, то есть поменять состояние на $|\psi\rangle = \beta_1|1\rangle + \beta_2|2\rangle + \ldots + \beta_N|N\rangle$. Квантовая механика допускает применение к квантовым состояниям только линейных операторов. Это значит, что если мы рассматриваем состояние |φ> как N-мерный вектор $(\alpha_1, \ldots, \alpha_N)^T$, то применение оператора, изменяющего |φ> на |ψ> соответствует умножению |φ> на NxN комплексную матрицу U:

$$U\begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix} \tag{1.6.1}$$

Поскольку измерение |ψ> также имеет вероятностное разложение, мы получим ограничение

$$\sum_{j=1}^{N} |\beta_j|^2 = 1. \tag{1.6.2}$$

Это значит, что оператор U должен сохранить норму векторов и являться унитарным. Это эквивалентно тому, что U всегда отразит вектор нормы 1 в другой вектор нормы 1. Поскольку унитарное преобразование всегда можно обратить, из этого следует что любая операция над квантовым состоянием может быть обращена через применение U^{-1} без потери данных в процессе. С другой стороны, измерение необратимо, поскольку мы не можем восстановить |φ> из классического состояния |j>. [13]

1.7. Квантовые вычислительные операции.

Рассмотрим, как квантовый компьютер может совершать вычисления над регистром кубитов. Рассмотрим метод, под названием квантовая модель цепи. В классической теории, логическая цепь представлена в виде конечного направленного незамкнутого графа с вентилями AND, OR и NOT. Входные биты проходят через эти вентили и, согласно цепи, принимают некоторые выходные значения. Будем считать, что цепь вычисляет некоторую булевскую функцию $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$ если итоговый узел содержит подходящее значение f(x) для каждого $X \in \{0, 1\}^n$.

Семейство цепей - это набор $C = \{C_n\}$ цепей, для каждого входного набора размером n. Каждая цепь имеет один выходной бит. Такая семья определяет язык $L \subseteq \{0, 1\}* = \cup_n \geq 0 \{0, 1\}^n$ если, для каждого n и каждого ввода $x \in \{0, 1\}^n$, цепь n возвращает 1 если $x \in L$ и возвращает 0 в противоположном случае. Семейство цепей называется равномерно полиномиальной, если есть детерминированная машина Тьюринга, которая возвращает цепь C_n в ответ на ввод n, используя пространство логарифмов из n. Заметим, что размер (количество вентилей) цепи C_n может расти полиноминально из n. Известно, что равномерно полиномиальное семейство цепей будет эквивалентно полиномиальной детерминированной машине Тьюринга: язык L может быть описан как равномерно полиномиальное семейство цепей, если $L \in P$, где P – класс языков, допустимых для полиномиальных машин Тьюринга.

Таким же способом можем описать вероятностные цепи, которые на вход, в дополнение к n входных бит получат несколько случайных битов. Вероятностна цепь вычисляет функцию f если она успешно возвращает правильный ответ f(x) с вероятностью не менее 2\3 для каждого x. [12]

Глава 2. Описание задачи

2.1. Постановка задачи

Данная работа имеет несколько целей, связанных с теорией квантовых компьютеров. Исследовательская цель состоит в изучении принципов работы квантовых компьютеров, алгоритмов квантовых вычислений и их реализации. После рассмотрения всех аспектов использования алгоритмов квантовых вычислений для различных целей необходимо провести сравнительный анализ с классическими способами решения этих же задач. На основании сравнений необходимо сделать вывод о преимуществах и недостатках алгоритмов квантовых вычислений. В ходе работы будут рассмотрены следующие алгоритмы.

- квантовый алгоритм для проверки сбалансированности функции;
- классические детерминированный и вероятностный алгоритм для проверки сбалансированности функции;
- квантовый алгоритм для факторизации;
- классический детерминированный алгоритм для проверки сбалансированности функции;
- квантовый алгоритм для систем линейных равенств;
- классический алгоритм для систем линейных равенств;

2.2. Ранние алгоритмы

Два наиболее крупных достижения квантовых алгоритмов — это алгоритм Шора для факторизации числа и алгоритм поиска Гровера. Для начала, опишем некоторые идеи, предшествующие им. Все квантовые алгоритмы работают с очередями в той или иной форме. Представим N-битный ввод $x = (x_0,\ldots,x_{n-1}) \in \{0, 1\}^N$. Обычно мы имеем $N=2^n$, поэтому мы можем получить доступ к биту x_i, используя n-битный индекс i. В данном случае ввод можно представить, как N-битную память, в которой мы можем получить доступ к любому элементу по его индексу. Представим в виде квантовой операции O_x: $|i, 0\rangle \rightarrow |i, x_i\rangle$. Первые n кубитов в состоянии также называются адресными кубитами или адресными регистрами, в то время как следующий n+1 кубит называется целевым кубитом.

2.3. Алгоритм Дейча.

Данный алгоритм один из первых, показавших превосходство квантовых алгоритмов над классическими. Невзирая на простоту, в алгоритме используется на полную принцип суперпозиции.

Пусть f: {0, 1} => {0, 1} бинарная функция. Заметим, что возможно только четыре значения для f

f_1:0→0, 1→0,

f_2:0→1, 1→1,

f_3:0→0, 1→1,

f_4:0→1, 1→0.

В первых двух случая, f_1 и f_2 назовем постоянными, а функции f_3 и f_4 сбалансированными. Для классического случая, для определения сбалансированности f, необходимо вычислить ее дважды, в то время как с использованием алгоритма Дейча хватит одного вычисления.

Пусть |0> и |1> соответствует классическим битам 0 и 1, определим состояние $|\psi_0\rangle=\frac{1}{2}(|00\rangle-|01\rangle+|10\rangle-|11\rangle)$. Применим f на это состояние в качестве унитарного оператора U_f : $|x, y\rangle \rightarrow |x, y\oplus f(x)\rangle$, где $\oplus$ сложение по модулю два(XOR). Для точности, мы имеем:

$$|\psi_1\rangle=U_f|\psi_0\rangle=\frac{1}{2}(|0,f(0)\rangle-|0,1\oplus f(0)\rangle+|1,f(1)\rangle-|1,1\oplus f(1)\rangle)$$

$$=\frac{1}{2}(|0,f(0)\rangle-|0,\neg f(0)\rangle+|1\rangle,f(1)\rangle-|1,\neg f(1)\rangle), \quad (2.3.1)$$

где ¬ соответствует отрицанию. Таким образом, данная операция не что иное, как вентиль CNOT с контрольным битом f(x); целевой бит y только в том случае, когда f(x)=1 и остается прежним во всех остальных случаях.

В результате, мы применяем вентиль Адамара на первый кубит, чтобы получить

$$|\psi_2\rangle=(UH\otimes I)|\psi_1\rangle=\frac{1}{2}|\sqrt{2}[(|0\rangle+|1\rangle)(|f(0)\rangle-|\neg f(0)\rangle)+(|0\rangle-|1\rangle)(|f(1)\rangle-|\neg f(1)\rangle)]. \quad (2.3.2)$$

Волновая функция упрощается до:

$$|\psi_2\rangle=1/\sqrt{2}|0\rangle(|f(0)\rangle-|\neg f(0)\rangle) \quad (2.3.3)$$

В случае если f постоянная, для которой $|f(0)\rangle=|f(1)\rangle$, и

$$|\psi_2\rangle=1/\sqrt{2}|1\rangle(|f(0)\rangle-|\neg f(0)\rangle) \quad (2.3.4)$$

для $|\neg f(0)\rangle=|f(1)\rangle$. Таким образом, значение первого кубита покажет является ли функция f постоянной или сбалансированной.

Составим квантовую цепь для имплементации алгоритма Дейча. Для начала применим трансформацию Адамара $W_2 = UH\otimes UH$ на $|01\rangle$ для получения $|\psi_0\rangle$. Определим условный вентиль U_f, то есть управляемый вентиль NOT с контрольным битом f(x) и действием $U_f : |x, y\rangle \rightarrow |x, y\oplus f(x)\rangle$. Затем вентиль Адамара применяется к первому кубиту, до его измерения.

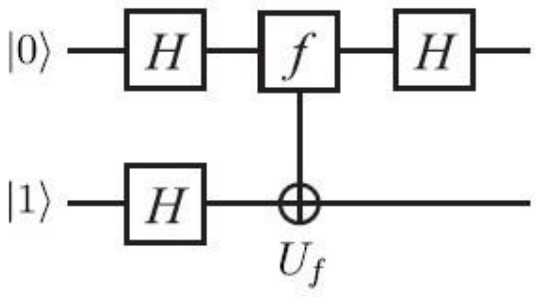

Рис. 2.3.1. Квантовая цепь для алгоритма Дейча

В квантовой цепи мы считаем вентиль U_f своеобразным черным ящиком, т.е. алгоритмом, для которого известны лишь входные и выходные данные. Такие структуры иногда называются оракулом, а вентиль U_f называется оракулом Дейча. Таким образом, имея матрицу U_f, можно составить цепь на рис. 2.3.1 и применить ее на входное состояние $|01\rangle$. После этого, можно сказать является f постоянной или сбалансированной за одно применение U_f.

2.4. Алгоритм Дейча-Джоза

Алгоритма Дейча может быть обобщен в виде алгоритма Дейча-Джоза. Допустим, есть некая бинарная функция $f:S_n \equiv \{0,1,...,2^{n-1}\} \rightarrow \{0,1\}$. Будем считать, что f может быть либо постоянной, либо сбалансированной. Когда f постоянная, она принимает значение 0 или 1 соответственно входному значению x. Сбалансированная функция принимает значение f(x)=0 для половины $x \in S_n$ и f(x)=1 для остальных. Другими словами, $|f^{-1}(0)|=|f^{-1}(1)| =2^{n-1}$, где |A| указывает на количество элементов во множестве A, известное как мощность множества A, Хотя существуют функции, которые не являются ни постоянными, ни сбалансированными, мы не будем рассматривать здесь эти случаи. Задача найти алгоритм, который определит, является ли f постоянной или сбалансированной за наименьшее число вычислений f.

Очевидно, что необходимо как минимум $2^{n-1} + 1$ шагов в худшем случае с классическими манипуляциями, для утверждения что f(x) постоянная или сбалансированная со стопроцентной вероятностью. Покажем, что количество шагов сократится до одного, при использовании квантового алгоритма.

Подготовить n+1 кубитный регистр в состоянии $|\psi_0> = |0>^{\otimes n} \otimes |1>$. Первые n кубит работают как входные кубиты, в то время как n+1й кубит будет служить своеобразной «временной памятью», такие кубиты для хранения временной информации назовем служебными кубитами.

Применим трансформация Уэлша-Адамара к регистру. Получим состояние:

$$|\varphi_1\rangle = U_H^{\oplus n+1}|\varphi_0\rangle = \frac{1}{\sqrt{2^n}}(|0\rangle + |1\rangle)^{\oplus n} \oplus \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|\varphi_0\rangle \oplus \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.4.1)$$

Применим f(x)-управляемый вентиль NOT к регистру, который изменит n+1 кубит только в том случае, если для входного кубита f(x) = 1. Следовательно,

необходим вентиль U_f, который посчитает f(x) и повлияет на регистр как U_f |x>|c>= |x>|c⊕f(x)>, , где |c> однокубитное состояние n+1го кубита. Заметим, что |c> обратилось только тогда, когда f(x) = 1 и осталось неизменным во всех остальных случаях. Получим состояние:

$$|\varphi_2\rangle = U_f|\varphi_1\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|\mathrm{x}\rangle\ \frac{1}{\sqrt{2}}(|\mathrm{f(x)}\rangle - |\neg f(x)\rangle) = \frac{1}{\sqrt{2^n}}\sum_{x}(-1)^{f(x)}|\mathrm{x}\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.4.2)$$

Хотя вентиль U_f применен однажды для всех, он одновременно применен для всех n-кубитных состояний.

Трансформация Уэлша-Адамара применяется к первым n кубит. Получим:

$$|\varphi_3\rangle = (W_n \oplus \mathrm{I})|\varphi_2\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}U_H^{\oplus \mathrm{n}}|\mathrm{x}\rangle\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.4.3)$$

Для наглядности, запишем применение однокубитного вентиля Адамара в следующем виде:

$$U_H|\mathrm{x}\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) = \frac{1}{\sqrt{2}}\sum_{y\in\{0,1\}}(-1)^{xy}|\mathrm{y}\rangle \quad (2.4.4)$$

где x∈ {0,1}, для нахождения результата. Применение трансформации Уелша-Адамара на $|x\rangle=|x_{n-1}\ldots x_1x_0\rangle$ даст:

$$W_n|\mathrm{x}\rangle = (U_H|\mathrm{x}_{n-1}\rangle)(U_H|\mathrm{x}_{n-2}\rangle)\ldots(U_H|\mathrm{x}_0\rangle) = \frac{1}{\sqrt{2^n}}\sum_{y_{n-1},y_{n-2},\ldots y_0\in\{0,1\}}(-1)^{x_{n-1}y_{n-1}+x_{n-2}y_{n-2}+\cdots+x_0y_0)} \times |y_{n-1},y_{n-2},\ldots y_0\rangle \quad (2.4.5)$$

где $x*y=x_{n-1}y_{n-1} \oplus x_{n-2}y_{n-2} \oplus \ldots \oplus x_0y_0$. Заменив результат по рис. 2.4.3 получим

$$|\varphi_3\rangle = \frac{1}{2^n}(\sum_{x,y=0}^{2^n-1}(-1)^{f(x)}(-1)^{x*y}|y\rangle)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \quad (2.4.6)$$

Первые n кубит измерены. Допустим f(x) постоянная.

$$|\varphi_3\rangle = \frac{1}{2^n}\left(\sum_{x,y}(-1)^{x*y}|y\rangle\right)\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.4.7}$$

В итоге получаем:

$$\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{x*y} \tag{2.4.8}$$

с фиксированным y ∈ S_n. Это выражение сводится к нулю, при x*y=0 для половины от всех x и 1 для другой половины x, если y не равен 0.Тогда сумма возвращает δ_{y0}. Теперь состояние сводится к

$$|\varphi_3\rangle = |0\rangle^{\oplus n}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.4.9}$$

и измерение выходных первых n кубитов всегда 00…0. Теперь допустим, что f(x) сбалансированная. Вероятность |y=0> в $|\varphi_3>$ пропорциональна

$$\sum_{x=0}^{2^n-1}(-1)^{f(x)}(-1)^{x*0} = \sum_{x=0}^{2^n-1}(-1)^{f(x)} = 0 \tag{2.4.10}$$

Таким образом, пропадает вероятность в результате измерения получить 00…0 для первых n кубитов. В результате, функция f постоянная, если мы получили 00…0 при измерении первых n кубитов состояния $|\varphi_3>$, и сбалансированная в другом случае.

2.5. Алгоритм Шора

Возможно наиболее важный квантовый алгоритм на данный момент – это алгоритм факторизации Шора. Он способен найти делители составного числа N за грубо говоря $(\log N)^2$ шагов. С другой стороны, на сегодняшний день нет ни одного детерминистического или случайного алгоритма, который может факторизовать N за полиномиальное время. Лучший известный классический алгоритм по времени работает примерно $2^{(\log N)\alpha}$, где $\alpha = 1/3$ для эвристической верхней границы и $\alpha = ½$ для точной верхней границы. По факту, большинство современной криптографии основывается на предположении о том, что не существует классического алгоритма факторизации. Вся криптография, к примеру, RSA будет взломана, если будет реализован алгоритм Шора.

Крайне важное наблюдение Шора в том, что есть эффективный квантовый алгоритм для проблемы нахождения периода, и что это может быть сведено к факторизации. Допустим, мы хотим найти делители составного числа $N > 1$. Предположим, что N нечетное и не простое число, так как это может быть легко отсеяно классическим алгоритмом. Случайно выберем некоторое целое число $x \in \{2,..., N-1\}$, которое взаимно простое с N. Если x не взаимно простое N, тогда наибольший общий делитель для x и N это ненулевой делитель числа N. Теперь будем считать x и N взаимно простыми, поэтому x – элемент мультипликативной группы Z_N. Рассмотрим последовательность

$$1 = x^0 (\mathrm{mod}\ N),\ x^1 (\mathrm{mod}\ N),\ x^2 (\mathrm{mod}\ N),\ ... \qquad (2.5.1)$$

Последовательность зациклится через некоторое время: есть как минимум $0<r<=N$, таких что $x^r = 1 \pmod N$. Назовем r периодом последовательности. Допустив что N нечетное и не простое, можем увидеть, что с вероятностью больше или равной 1/2, период r четный и $x^{r/2} + 1$ и $x^{r/2} - 1$ не множители N.

В этом случае имеем:

$$\begin{aligned} x^r &\equiv 1 \mod N \iff \\ (x^{r/2})^2 &\equiv 1 \mod N \iff \\ (x^{r/2}+1)(x^{r/2}-1) &\equiv 0 \mod N \iff \\ (x^{r/2}+1)(x^{r/2}-1) &= kN \text{ for some } k. \end{aligned}$$

Рис. 2.5.1. Период последовательности

Заметим, что $k > 0$, поскольку в обоих случаях $x^{r/2}+ 1 > 0$ и $x^{r/2}- 1 > 0$ ($x > 1$). В результате $x^{r/2} + 1$ или $x^{r/2} – 1$ будут иметь общий множитель с N. Так как $x^{r/2} + 1$ и $x^{r/2} – 1$ не являются сомножителями N, этот общий множитель будет меньше N, и по факту оба этих числа имеют нетривиальный сомножитель с N. В связи с этим, если мы знаем r, тогда мы можем высчитать наибольшие общие делители НОД($x^{r/2}+ 1$, N) и НОД($x^{r/2} − 1$, N) и оба этих числа будут нетривиальными множителями N.

Таким образом, проблема факторизации сводится к нахождению периода r функции.

2.6. Алгоритм Шора для нахождения периода

Покажем как с помощью алгоритма Шора найти период r функции f, заданной в качестве черного ящика, который отражает $|a\rangle|0^n\rangle \rightarrow |a\rangle|f(a)\rangle$. Выберем такое $q = 2^l$, что $N^2 < q \leq 2N^2$. Реализуем преобразование Фурье F_q, используя $O((\log N)^2)$ операций. Обозначим O_f унитарную операцию, которая отражает $|a\rangle|0^n\rangle \rightarrow |a\rangle|f(a)\rangle$, в котором первый регистр состоит из l кубитов, а второй $n = [\log N]$ кубит.

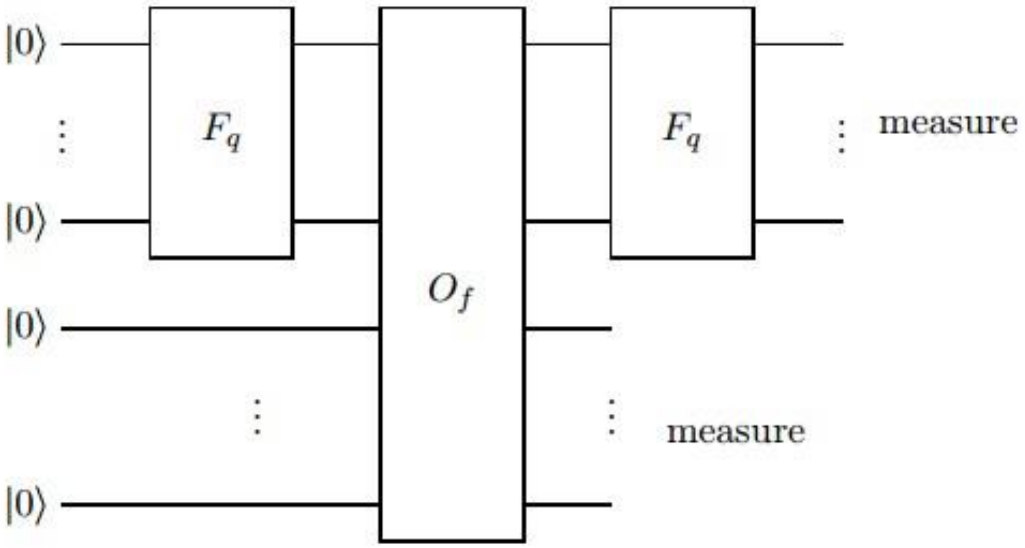

Рис. 2.6.1 Алгоритм Шора для нахождения периода

Начнем с $|0^\ell\rangle|0^n\rangle$. Применим l вентилей Адамара к первому регистру для создания суперпозиции.

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a>|0^n>. \qquad (2.6.1)$$

Второй регистр все еще заполнен нулями. Теперь используем принцип «черного ящика» для вычисления f(a) в квантовой параллели.

$$\frac{1}{\sqrt{q}}\sum_{a=0}^{q-1}|a>|f(a)>. \qquad (2.6.2)$$

Измерение второго регистра дает некоторое значение f(s), с $s < r$. Пусть m будет количеством элементов {0,…q-1} соответствующим исследуемому значению f(s). Поскольку f(a) = f(s) тогда и только тогда, когда a=s mod r, то в таком виде a = jr+s (0<=j<m) подходящее a для которого f(a) = f(s). Таким

образом первый регистр сводится к суперпозиции |s>, |r + s>, |2r + s>, |3r + s>, . . .; эта суперпозиция продолжится пока не будет найдено такое последнее число jr+s < q, объявим m количеством элементов в этой суперпозиции, то есть количество целочисленных j таких что jr+s ∈ {0, ..., q − 1}. Второй регистр сводится к классическому состоянию |f(s)>. Теперь можно игнорировать второй регистр, а в первом получим:

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m-1}|jr+s>. \qquad (2.6.3)$$

Применив квантовое преобразование Фурье, получим:

$$\frac{1}{\sqrt{m}}\sum_{j=0}^{m=1}\frac{1}{\sqrt{q}}\sum_{b=0}^{q=1}e^{2\pi i\frac{(jr+a)b}{q}}|b\rangle=\frac{1}{\sqrt{mq}}\sum_{b=0}^{q=1}e^{2\pi i\frac{ab}{q}}\left(\sum_{j=0}^{m=1}e^{2\pi i\frac{jrb}{q}}\right)|b\rangle \qquad (2.6.4)$$

Заметим, что q/r больше не целое число, а m =[q/r]. Используя $|1-e^{i0}| = 2|\sin\theta/2)|$, можем переписать абсолютное значение в этом случае.

2.7. Квантовый алгоритм для систем линейных равенств

Алгоритм Харроу-Хассидима-Ллойда (ХХЛ) для решения больших систем линейных уравнений. Зададим систему матрицей N × N с вещественными или комплексными элементами и N-мерным ненулевым вектором b. Для простоты предположим, что $N = 2^n$.

Задача решения линейной системы заключается в нахождении N-мерного вектора x такого, что Ax = b.

Решение больших систем линейных уравнений чрезвычайно важно во многих вычислительных задачах в промышленности, науке, оптимизации, машинном обучении и т. д. Во многих приложениях достаточно найти вектор x’, близкий к фактическому решению x.

Мы будем предполагать, что A обратимо (имеет ранг N), чтобы гарантировать существование единственного решения вектора x, который в таком случае равен A^{-1}*b. Если A не имеет полного ранга, тогда приведенные ниже методы все же позволят инвертировать его, заменяя A^{-1} с использованием псевдо инверсии Мур-Пенроуза.

Алгоритм ХХЛ может эффективно решать большие линейные системы (при определенных допущениях), но его слабой стороной является то, что вместо вывода вектора решения x его целью является вывод состояния n-кубита

$$|x\rangle := \frac{1}{||x||}\sum_{i=0}^{N-1} x_i\,|\mathrm{i}\rangle \qquad (2.7.1)$$

или некоторого другого n-кубитного состояния, близкого к |x>. Это называется проблемой квантовой линейной системы:

QLSP: нахождения состояния n-кубита |x’> такого, что |||x> - |x’>|| ≤ ε и Ax = b.

Важно, что QLSP является по своей сути квантовой проблемой, поскольку цель состоит в том, чтобы создать n-кубитное состояние, у которого

амплитудный вектор (с точностью до нормализации и до ε-ошибки) является решением линейной системы. В общем случае это не так эффективно, как просто записать N-мерный вектор x на лист бумаги, но в некоторых случаях, когда нам нужна только некоторая частичная информация об x, этого может быть достаточно, чтобы приблизительно построить |x>.

Добавим некоторые ограничения, которые сделают линейную систему «хорошо» и пригодной для HHL алгоритма.

Имеем унитарный оператор, который может подготовить вектор b как n-кубитное квантовое состояние

$$|b\rangle := \frac{1}{||b||}\sum_{i=0}^{N-1} b_i \,|\mathrm{i}\rangle \qquad (2.7.2)$$

используя цепь B 2x-кубитных вентилей. Для простоты мы предполагаем, что ||b|| = 1.

Матрица A является s-разреженной, и мы имеем разреженный доступ к ней. Такая разреженность не является обязательной для алгоритма и может быть заменена другими свойствами, которые позволяют эффективное блочное кодирование A.

Матрица A уравновешенная: максимальное соотношение между ее наибольшим и наименьшим единичным значением - к. Для простоты предположим, что наименьшее значение ≥ 1/к, а наибольшее ≤ 1. Другими словами, все собственные значения A лежат в интервале $[-1, -1/к] \cup [1/к, 1]$. Чем меньше k, тем лучше будет для алгоритма. Предположим нам известно к или, по крайней мере, известна верхняя граница к.

Вектор решения x, который мы ищем, равен A^{-1}*b, поэтому мы хотим применить матрицу A^{-1} к b. Если A имеет спектральное разложение:

$$\mathrm{A} = \sum_{j=0}^{N-1} \lambda_j \, a_j a_j^T \qquad (2.7.3)$$

Тогда отражение A^{-1} совпадает с отражением $a_j \rightarrow \frac{1}{\lambda_j} a_j$. Умножим собственный вектор a_j на скаляр $1/\lambda_j$. Вектор b также может быть представлен в виде линейной комбинации собственных векторов a_j:

$$b = \sum_j \beta_j \, a_j \tag{2.7.4}$$

(нам не нужно знать коэффициенты β_j для дальнейших вычислений). Применим A^{-1} к b , чтобы получить

$$A^{-1} b = \sum_j \beta_j \frac{1}{\lambda_j} a_j \tag{2.7.5}$$

нормализованной как квантовое n-кубитное состояние.

Отображения A и A^{-1} не являются унитарными (если $|\lambda_j| = 1$ для любых j), поэтому мы не можем просто применить A^{-1} в качестве квантовой операции для состояния |b>, чтобы получить состояние |x>.

$$U = e^{iA} = \sum_j e^{i\lambda_j} \, a_j a_j^T \tag{2.7.6}$$

U является унитарным и имеет те же собственные векторы, что и A и A^{-1}. Мы можем представить U и степени U с помощью моделирования Гамильтона, а затем использовать оценки фазы для оценки λ_j связанного с собственным вектором $|a_j\rangle$ с небольшой погрешностью аппроксимации (для этого простоты этого примера предположим , что ошибка 0). При условии нашей оценки λ_j мы можем повернуть вспомогательный |0>-кубит

$$\frac{1}{k\lambda_j} |0\rangle + \sqrt{1 - \frac{1}{\left(k\lambda_j\right)^2}} |1\rangle \tag{2.7.7}$$

(это допустимое состояние, потому что $|k\lambda_j| \geq 1$). Далее мы отменяем фазовую оценку, чтобы установить состояние обратно в |0>. Избавившись от вспомогательных кубитов, содержащих временные результаты оценки фазы, мы получаем унитарное отображение

$$|b\rangle|0\rangle \rightarrow |a_j\rangle\left(\frac{1}{k\lambda_j}|0\rangle + \sqrt{1-\frac{1}{\left(k\lambda_j\right)^2}}|0\rangle\right) \qquad (2.7.8)$$

Если мы подготовим копию |b>|0> = $\sum_j \beta_j |a_j\rangle|0\rangle$ и применим к нему указанное выше унитарное отображение, тогда получим

$$\sum_j \beta_j |a_j\rangle\left(\frac{1}{k\lambda_j}|0\rangle + \sqrt{1-\frac{1}{(k\lambda_j)^2}}|1\rangle\right) = \frac{1}{k}\sum_j \beta_j \frac{1}{\lambda_j}|a_j\rangle|0\rangle + |\varphi\rangle|1\rangle \qquad (2.7.9.)$$

где мы не учитываем (субнормализованное) состояние |φ>. Обратим на это внимание, потому что, потому что $\sum_j |\beta_j/\lambda_j|^2 \geq \sum_j |\beta_j|^2$ норма части состояния, оканчивающейся на кубит |0>, не меньше 1/κ. Соответственно, теперь мы можем применить O(κ) циклов усиления амплитуды, чтобы усилить эту часть состояния, чтобы иметь амплитуду 1. Это подготавливает состояние |X>? что приводит к алгоритму, который создает состояние |x'> которое ε-близко к |x>, используя примерно $k^2 s/\varepsilon$ запросов к H и примерно κs (κn / ε + B) других 2-кубитных вентилях

Глава 3. Реализация задачи.

3.1. Средства реализации

В качестве средства реализации была выбрана платформа *.NET Core SDK 2.2* с пакетом *Microsoft Quantum Development Kit 0.6.* Данный пакет содержит все необходимые инструменты для квантовых вычислений, а также удобен для разработчиков с опытом использования *Microsoft Visual Studio* или *Visual Studio Code*.

Для написания кода квантовых алгоритмов выбран язык *Q#*. Базовая возможность языка: создание и использование кубитов для алгоритмов. Важнейшая особенность – возможность запутываться и создавать квантовую суперпозицию между кубитами через вентили CNOT и Адамара.

Для реализации классических алгоритмов выбран язык C#, как современный язык, отвечающий всем принципам ООП.

3.2. Требования к программному и аппаратному обеспечению

Приложение предназначено для использования на IBM PC - совместимых компьютерах с операционной системой *Windows 7* и более поздних. Необходимым требованием является наличие платформы *.Net Core SDK 2.1* и более поздник версий.

Минимальные системные требования приложения обусловлены требованиям операционной системы. Требования к ОЗУ – минимальный объем 4 ГБ.

3.3. Реализация

По итогам изучения алгоритма Дейча-Джозы была составлена блок-схема для его реализации.

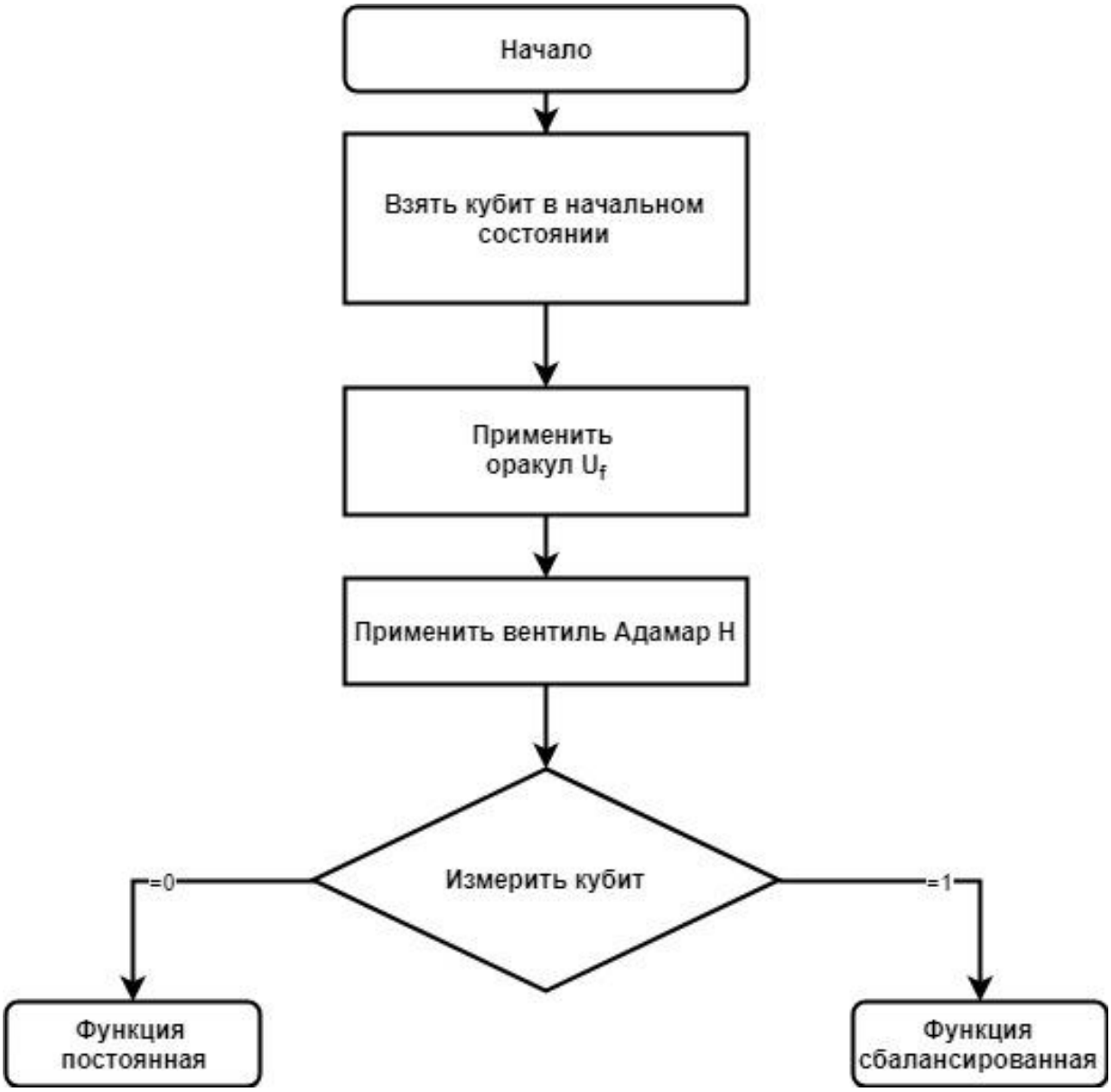

Рис. 3.3.1. Блок-схема алгоритма Дейча-Джозы

Для классического детерминированного алгоритма в худшем случае понадобится вызвать функцию на 2n-1 + 1 аргументах: ровно на половине и еще одном. Если все вычисленные значения одинаковы, то функция, очевидно константна. Если же существуют хотя бы два различных результата, функция сбалансирована. Сложность детерминированного алгоритма экспоненциальна и составляет O(2n-1 + 1).

Рассмотрим вероятностный алгоритм для определения сбалансированности функции. Допустим, мы вычислили функцию на k<2n-1+1 аргументах. Если среди значений функции есть два различных, то функция сбалансирована. Иначе, мы объявляем ее константной с вероятностью p(k)<1. Составим блок-схему этого алгоритма.

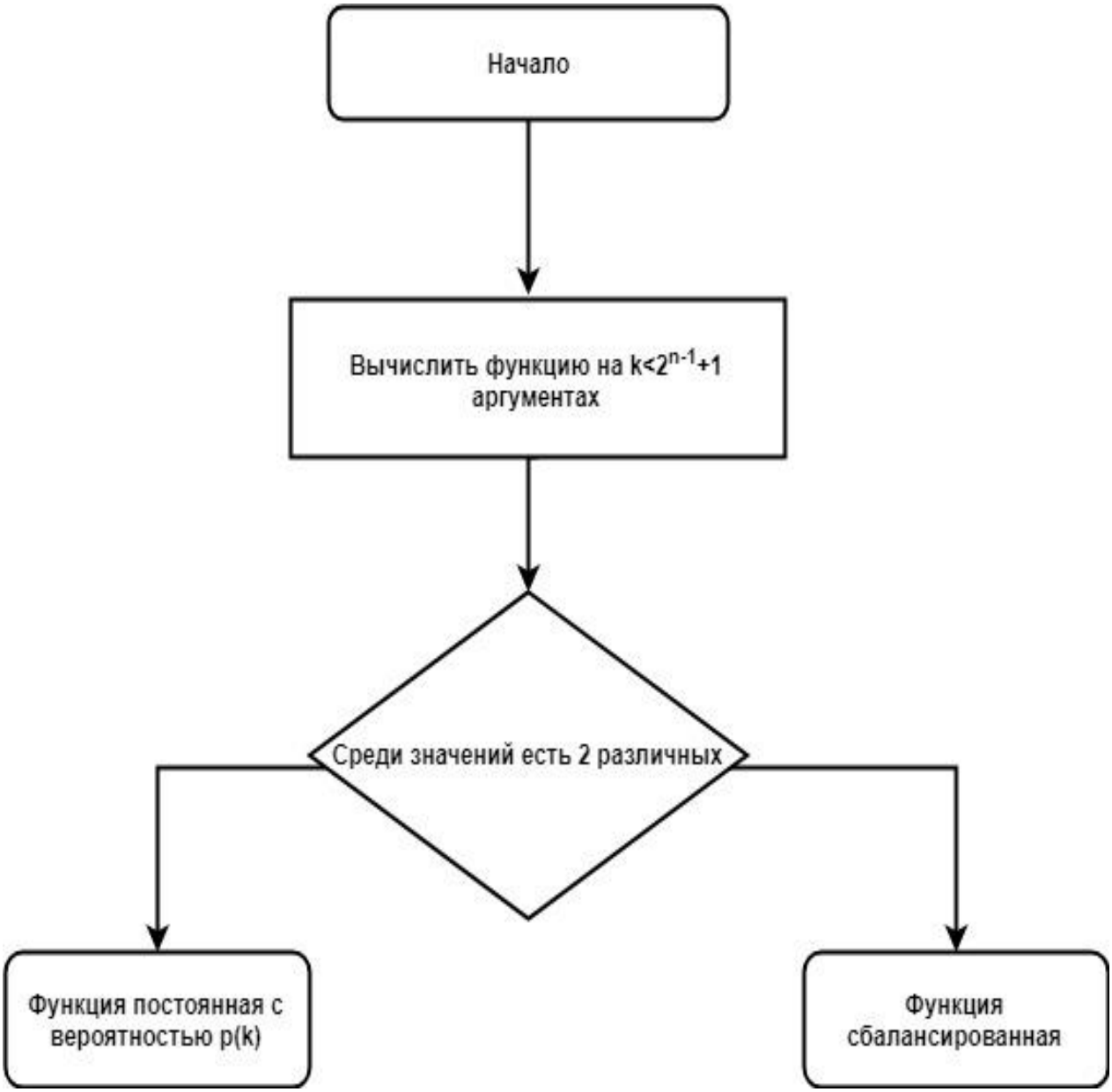

Рис. 3.3.2. Классический вероятностный алгоритм

Объявим функцию постоянной с вероятностью p(k) <1. В таком случае, вероятность ошибки будет равно 1 – p(k). Если мы выбирали аргументы равномерно, то вероятность того, что два подряд идущих значений функции одинаковы, равна ½, а вероятность встретить k одинаковых подряд идущих значений равна $½^{(k)}$. Таким образом:

$$1 – p(k)= ½^{(k)} \quad (3.3.1)$$

$$p(k) = 1 – ½^{(k)} \quad (3.3.2)$$

Обратная функция:

$$k(p) = \log_2 \frac{1}{1-p} \quad (3.3.3)$$

При фиксированном p сложность классического вероятностного алгоритма константна и равна $O(\log_1 \frac{1}{1-p})$.

По итогам изучения алгоритма Шора была составлена блок-схема для его реализации.

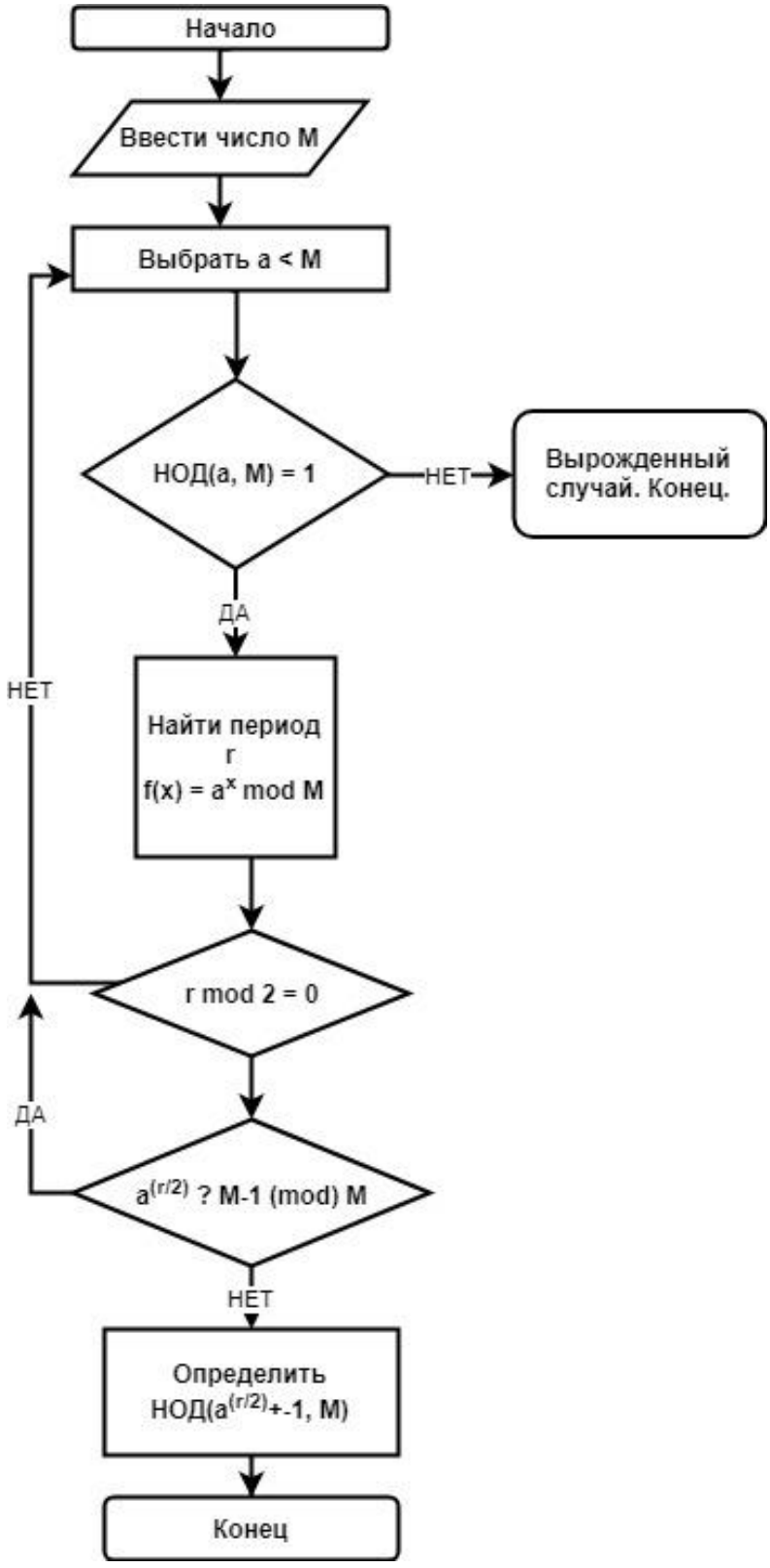

Рис.3.3.3. Блок-схема реализации алгоритма Шора

Опишем алгоритм Шора для реализации на компьютере. Отметим, что хотя теоретическая часть алгоритма связана с эквивалентными преобразованиями квадратичных форм, практическая часть алгоритма выполняется на основе вычисления коэффициентов P, Q, r метода непрерывных дробей без обращения к формам. Каждая итерация цикла соответствует одной операции применения оператора редукции к соответствующей форме. При необходимости можно восстановить формы f_k = (ak, bk, ck) по формулам $(a_k, b_k, c_k) = ((-1)^{k-1}Q_{k-1}, 2P_k, (-1)^kQ_k)$

Вход: составное число n, достаточно маленькое число b

Выход: Нетривиальный делитель n

Инициализация алгоритма.

Проверим, является ли n полным квадратом. Если да, то вычислим d=$\sqrt{n}$, и завершим вычисление. Иначе, перейдем к следующему пункту.

Если n≡1(mod4), тогда заменим n на 2n. Определим D=4n, q0=$b\sqrt{D}c$.

Определим исходные значения параметров P, Q, r:

P0=0, Q0=1, r0=P1=b$\sqrt{n}$c, Q1=n−r_0^2, r1=b2r0/Q1c.

Первый цикл

$P_k=r_{k-1}\cdot Q_{k-1}-P_{k-1}$,

$Q_k=Q_{k-2}+(P_{k-1}-P_k)\cdot r_{k-1}$,

$$r_k = b\frac{P_k + b\sqrt{n}\,c}{Q_k}c, k \geq 2$$

Продолжаем вычисления коэффициентов Pk, Qk, rk до тех пор, пока не найдем Q_k, являющееся полным квадратом. Это должно произойти при некотором k. Пусть $Q_k=d^2$ для целого d>0. Перейдем к следующему циклу. Второй цикл:

Начнем цикл вычислений новых параметров P'_j, Q'_j, r'_j. Формулы для реализации второго цикла останутся такими же, как раньше. Изменятся только начальные значения параметров P′, Q′, r′:

$P'_0 = -P_k$,

$Q'_0 = d$,

$$r'_0 = b\,\frac{P'_0 + b\sqrt{n}c}{Q'_0}\,c$$

$P'_1 = r'_0 \cdot Q'_0 - P'_0$,

$Q'_1 = (N - P'^2_1)/Q'_0$.

Вычисление следует продолжать, пока два подряд идущих значения P'_j, P'_{j+1} не окажутся равными. Тогда, значение Q_j даст искомый делитель числа n.

3.4. Библиотека QuantumUtils

Для квантовых алгоритмов была реализована библиотека QuantumUtils со следующими операциями для трансформации состояния |x, y⟩ в состояние |x, y ⊕ f(x)⟩

- f(x) = 0

  Входные данные:

  1) N кубитов в произвольном состоянии |x⟩ (входной регистр)

  2) Кубит в произвольном состоянии |y⟩ (выходной кубит)

  ```
  operation Oracle_Zero (x : Qubit[], y : Qubit) : Unit
  ```

- f(x) = 1

  Входные данные:

  1) N кубитов в произвольном состоянии |x⟩ (входной регистр)

  2) Кубит в произвольном состоянии |y⟩ (выходной кубит)

  ```
  operation Oracle_One (x : Qubit[], y : Qubit) : Unit
  ```

- f(x) = x (значение k-того кубита)

  Входные данные:

  1) N кубитов в произвольном состоянии |x⟩ (входной регистр)

  2) Кубит в произвольном состоянии |y⟩ (выходной кубит)

  3) Индекс кубита во входном регистре (0 <= k < N)

  operation Oracle_Kth_Qubit (x : Qubit[], y : Qubit, k : Int) : Unit

- f(x) = 1 если x имеет четное число единиц, и 0 в противоположном случае

  Входные данные:

  1) N кубитов в произвольном состоянии |x⟩ (входной регистр)

  2) Кубит в произвольном состоянии |y⟩ (выходной кубит)

  ```
  operation Oracle_OddNumberOfOnes (x : Qubit[], y : Qubit) :
  Unit
  ```

- $f(x) = \Sigma_i r_i x_i$ по модулю 2 для битового вектора r

  Входные данные:

  1) N кубитов в произвольном состоянии $|x\rangle$ (входной регистр)

  2) Кубит в произвольном состоянии $|y\rangle$ (выходной кубит)

  3) Вектор размера N

  ```
  operation Oracle_ProductFunction (x : Qubit[], y : Qubit, r
  : Int[]) : Unit
  ```

- $f(x) = \Sigma_i (r_i x_i + (1 - r_i)(1 - x_i))$ по модулю 2 для битового вектора r

  Входные данные:

  1) N кубитов в произвольном состоянии $|x\rangle$ (входной регистр)

  2) Кубит в произвольном состоянии $|y\rangle$ (выходной кубит)

  3) Вектор размера N

  ```
  operation Oracle_ProductWithNegationFunction (x : Qubit[],
  y : Qubit, r : Int[]) : Unit
  ```

- $f(x) = \Sigma_i x_i$ + (1 если префикс x равен вектору, 0 если нет) по модулю 2

  Входные данные:

  1) N кубитов в произвольном состоянии $|x\rangle$ (входной регистр)

  2) Кубит в произвольном состоянии $|y\rangle$ (выходной кубит)

  3) Вектор размера N

  ```
  operation Oracle_HammingWithPrefix (x : Qubit[], y : Qubit,
  prefix : Int[]) : Unit
  ```

- f(x) = 1 если *x* имеет два или три бита (из трех) установленных в *1*

  Входные данные:

  1) N кубитов в произвольном состоянии $|x\rangle$ (входной регистр)

  2) Кубит в произвольном состоянии $|y\rangle$ (выходной кубит)

  ```
  operation Oracle_MajorityFunction (x : Qubit[], y : Qubit)
  : Unit
  ```

3.5. Библиотека MathLib

В библиотеке MathLib реализованы методы для работы классических алгоритмов.

Класс матрицы Matrix со следующими методами и операторами:

```
class Matrix<T> : ICloneable
```

- Метод для транспонирования матрицы
  `void Transporate()`
- Метод для заполнения матрицы
  `void Fill(T[,] matrix)`
- Оператор умножения матрицы matr1 на матрицу matr2
  `Matrix<T> operator * (Matrix<T> matr1, Matrix<T> matr2)`
- Метод умножения матрицы на значение value
  `void MultiplyByValue(int value)`
- Метод, возвращающий матрицу, обратную матрице matr
  `Matrix<double> Inverse(Matrix<double> matr)`
- Метод для подсчета определителя матрицы
  `double Determinant`
- Метод возвращает матрицу без указанных строк и столбцов. Исходная матрица не изменяется.
  `Matrix<T> Exclude(int row, int column)`

Заключение

В результате работы были реализованы решения популярных задач компьютерных вычислений с применением классических и квантовых алгоритмов. По итогам работы было проведен анализ эффективности алгоритмов по времени и количеству операций. Итоги представлены в таблице ниже.

Сбалансированность функции (детерминированный)	$O(2^{n-1} + 1)$
Сбалансированность функции (вероятностный)	$O(log\frac{1}{1-p})$
Алгоритм Дейча-Джозы	$O(1)$
Метод квадратичных форм Шенкса	$O(n^{\frac{1}{5}+\varepsilon})$
Метод решета числового поля	$exp((\frac{32}{9}\ log\ N)^{\frac{1}{3}} * log\ log\ N^{\frac{2}{3}})$
Алгоритм Шора	$O(log\text{^}3(N))$
Алгоритм ХХЛ	$O(Nlog\ N\ k^2)$
Разложение Гаусса	$O(N^3)$

Список литературы

1. Вирт Н. Алгоритмы и структуры данных / Н. Вирт ; пер. с англ. Д. Б. Подшиваловой. – 2-е изд., испр. – Санкт–Петербург : Нев. Диалект. – 2001. – 352 с.

2. Кнут Д.Э. Искусство программирования / Д.Э. Кнут. ; пер. с англ. и ред. В.Т. Тертышного, И.В. Красикова; под общ. ред. Ю.В. Козаченко. – М. ; СПб. ; Киев : Вильямс. – 2000. – Т. 3 : Сортировка и поиск. – 822 с.

3. Дейт К. Дж. Введение в системы баз данных / Introduction to Database Systems. – 8-е изд. – М.: Вильямс. – 2006. – 1328 с.

4. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд.

5. Шилдт Герберт. C# 4.0: полное руководство.: Пер. с англ. – М.: ООО «И.Д. Вильямс». – 2011 –1056 с.

6. Bill Evjen, Scott Hanselman, Devin Rader. Professional ASP.NET 4 in C# and VB. – 2010 – 1539 с.

7. R. de Wolf. Quantum Computing and Communication Complexity, University of Amsterdam. – 2001 – 155 с.

8. A. Ambainis. Quantum lower bounds by quantum arguments. Journal of Computer and System Sciences. – 2002. – v.94, – N5, – pp.1634-1639.

9. A. Ambainis. Polynomial degree vs. quantum query complexity. Journal of Computer and System Sciences. – 2006. – v.94, – N5, – pp.1634-1639.

10. D. Deutsch. Quantum theory, the Church-Turing principle, and the universal quantum Turing machine. In Proceedings of the Royal Society of London. – 1985.

11. D. Deutsch. Quantum computational networks. In Proceedings of the Royal Society of London. – 1989

12. Nakahara, Mikio. Quantum computing : from linear algebra to physical realizations / M.Nakahara and Tetsuo Ohmi. – 2008 – 421 с.

13. R. de Wolf. Quantum Computing and Communication Complexity. PhD thesis, University of Amsterdam. – 2001.

14. D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. In Proceedings of the Royal Society of London. –1992. – С. 553–558.

15. R. Feynman. Simulating physics with computers. International Journal of Theoretical Physics. – 1982. – С. 467–488.

16. Р. Фейнман, Моделирование физики на компьютерах. Сборник «Квантовый компьютер и квантовые вычисления.» Вып.2. Ижевск. – 1999. – С. 53-95

17. К. А. Валиев, А. А. Кокин. Квантовые компьютеры: надежды и реальность. Регулярная и хаотическая динамика (РХД) М.-Ижевск. – 2001. – 350 с.

18. Д. Бауместер, А. Экерт, А. Цайлингер. Физика квантовой информации. – 2002. – 375 с.

19. П. Шор. Полиномиальные по времени алгоритмы разложения числа на простые множители и нахождение дискретного алгоритма для квантового компьютера. Сборник. «Квантовый компьютер и квантовые вычисления» вып.2 Ижевск. – 1999. –С. 200-247.

20. Л. К. Гровер. Квантовая механика помогает найти иголку в стоге сена. Сборник «Квантовый компьютер и квантовые вычисления» вып.2. Ижевск. – 1999. –С.101-109.

21. D. G. Cory and other. Ensemble Quantum Computing by NMR Spectroscopy/Proc.Natl.Acad.Sci.USA. – 1997. –С.1634-1639.

22. D. G. Cory and other . Experimentally Accesible Paradigm for Quantum Computing. Physica. – 1997. – C.82-101.

23. J. A. Jones, M. Mosca. Implementation of Quantum Algorithm on NMR Quantum Computer. J. Chem.Soc.. – 1998, – C.1648 — 1653.

24. А. Л. Чуанг и др. Экспериментальная реализация квантового алгоритма. Сборник «Квантовый компьютер и квантовые вычисления» вып.2. Ижевск. – 1999. – с.130 — 140.

25. D. Loss. D. DiVincenzo. Quantum Computation with Quantum Dots. Phys.Rev. – 1998. – C.120-126.

26. G. Burkard and others. Coupled Quantum Dots as Quantum Gates. Phys.Rev. – 1999. – C.2070.

27. L. Fedichkin and others. Novel Coherent Quantum Bit Using Spatial Quantization Levels in Semiconductor Quantum Dots. Квантовые компьютеры и квантовые вычисления. – 2000. – т.1. – С. 120 — 126.

28. D. V. Averin. Quant Computing and Quantum Measurement with Mesoscopic Josevson Junctions. – 2000. – C.13.

29. B. Kane. A silicon-based nuclear spin quantum computer. – 1998. – C.133

30. R. Vrijen, D. Di Vincenzo. Electron Spin Resonance Transistor for Quantum Computation in Silicon-Germanium Heterostructure. Phys.Rev.A. – 2000.

31. T. Klimov, I. G. Neizvestny, S. P. Suprun, V. N. Shumsky. Medium for interaction between two qubits in quantum computatios. Quantum computer and quantum computing. – 2001. C.79-84.