Алгоритм Шора

Ю. Лифшиц*

12 декабря 2005 г.

План лекции

1. Подготовка
   (a) Разложение чисел на множители
   (b) Квантовые вычисления
   (c) Эмуляция классических вычислений
2. Алгоритм Саймона
   (a) Квантовый параллелизм
   (b) Задача Саймона
3. Алгоритм Шора
   (a) Основные шаги
   (b) Разбор алгоритма
   (c) Реализация
4. Задача

1 Подготовка

1.1 Разложение чисел на множители

Постановка задачи разложения числа на множители выглядит следующим образом: на вход подается составное число N в двоичной записи, на выход должны быть выданы два числа p, q, такие что $N = pq$. Типичный размер — N порядка 2^{2000}.

Мотивацией для решения данной задачи является отсутствие на данный момент полиномиального классического алгоритма.

*Законспектировал Е. Решетников.

Решение этой задачи позволит, например, взломать систему RSA. Сейчас это одна из самых знаменитых алгоритмических проблем.

Лучший из известных классических алгоритмов имеет $O(2^{\sqrt[3]{n}})$ в качестве оценки времени работы. Уже сегодня существует квантовый алгоритм, который решает эту задачу за $O(n^2)$ [Питер Шор, 1994].

1.2 Квантовые вычисления

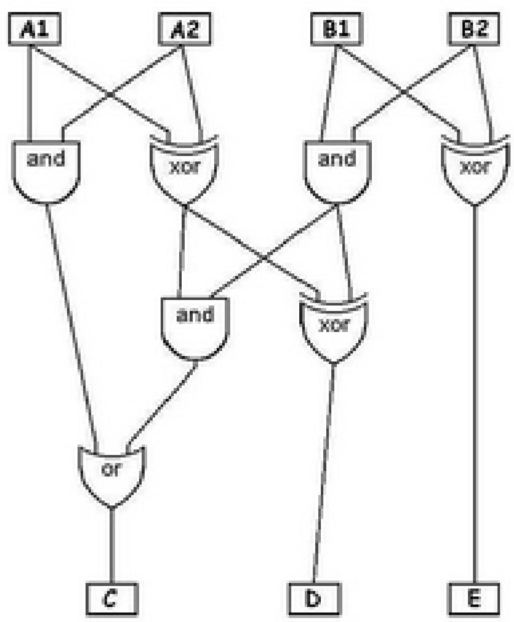
Основные определения, связанные с квантовыми компьютерами, а также примеры простейших вычислений с их использованием можно найти в лекции N9 "Введение в квантовые вычисления".

1.3 Эмуляция классических вычислений

Теорема. Для каждой классической логической схемы из AND и NOT можно построить квантовую схему, вычисляющую "почти"ту же функцию.

Логическая схема

Покажем, что такое логическая схема.



То есть на вход подаются какие-то числа (на нашей схеме они сверху). Можно считать, что все ребра ориентированы сверху вниз. Ребро, ведущее к какой-то операции, означает, что число, которое стоит сверху будет аргументом. Если операция бинарная, к ней будет вести два ребра, если унарная, то одно.

В общем случае, для n-арной операции имеем n входящих ребер. После, полученный результат является аргументом для аналогичных преобразований. В конечном итоге получаем набор чисел — выходные значения (находятся в нижнем ряду нашей схемы).

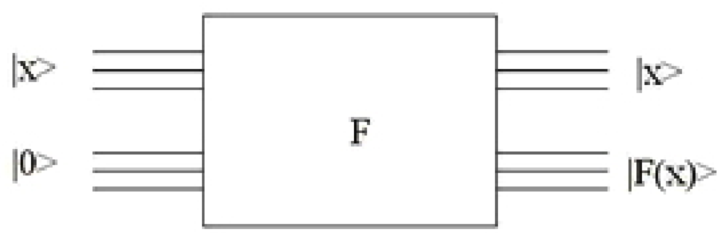
Обратимость

Квантовые вычисления обратимы. То есть нельзя реализовать необратимые преобразования. А это означает, что любое преобразование U инъективно (то есть, сохраняет длину аргумента).

Следствие. Любое преобразование U инъективно.

Доказательство. Допустим обратное. Пусть $U|x\rangle = |z\rangle = U|y\rangle$, где $x \neq y$. Тогда $U|x-y\rangle = 0$ (по линейности). Но тогда получаем противоречие с унитарностью. Значит обратное неверно.

Как выйти из данной ситуации? Ответ достаточно прост: будем хранить вспомогательные биты.

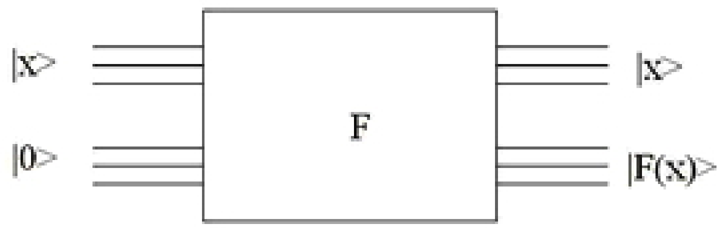


То есть, выделены дополнительные биты на результат. До выполнения операции начальные биты содержат само число, а последующие биты (столько, сколько необходимо для хранения результата) заполнены нулями. После выполнения операции начальные биты не меняются, а добавленные биты содержат результат преобразования над числом, поданным на входе.

2 Алгоритм Саймона

2.1 Квантовый параллелизм

Рассмотрим еще раз схему вычисления значения функции.



Данная схема позволяет считать нужную функцию сразу для многих аргументов. Допустим, что наша система из $(n + k)$ кубит находилась в промежуточном состоянии. В данном случае n — количество бит, необходимых для хранения нужных нам данных, а k — количество бит для хранения обработанных данных (значения функции, посчитанной для данного числа). До вычисления значения функции последние k бит любого из векторов начального состояния равны нулю. После применения преобразования к исходной системе, снова получим промежуточное состояние, в котором каждый вектор в качестве первых n бит будет содержать одно из поданных на вход чисел, а в качестве последних k бит — значение функции, вычисленной для этого числа.

Заметим, что любое состояние системы из n бит можно рассматривать как точку на единичной сфере в 2^n-мерном пространстве, где базисными векторами являются все базовые состояния системы.

Итак,

$$\sum_x |x\rangle|0\rangle \quad \to \quad \sum_x |x\rangle|f(x)\rangle$$

Тогда, измерив систему после выполнения функции, получим какую-то конкретную пару $x, f(x)$. Заметим, что после измерения можно получить все равно лишь значение функции для одного аргумента. В связи с чем появляется другая идеологическая задача: получить результат, который будет характеризовать все множество значений $f(\cdot)$.

Итог: с помощью квантового компьютера за одну итерацию можно вычислить значение функции сразу на многих входах.

2.2 Задача Саймона

Формулировка: дана функция F в виде (квантовой) схемы. Известно, что есть такое y, что $F(x) = F(x+y)$ (здесь “+” — это побитовый XOR). Нужно найти y.

Чтобы понять решение, предложенное Саймоном, вспомним преобразования Адамара для одного и для n кубит.

Преобразование Адамара

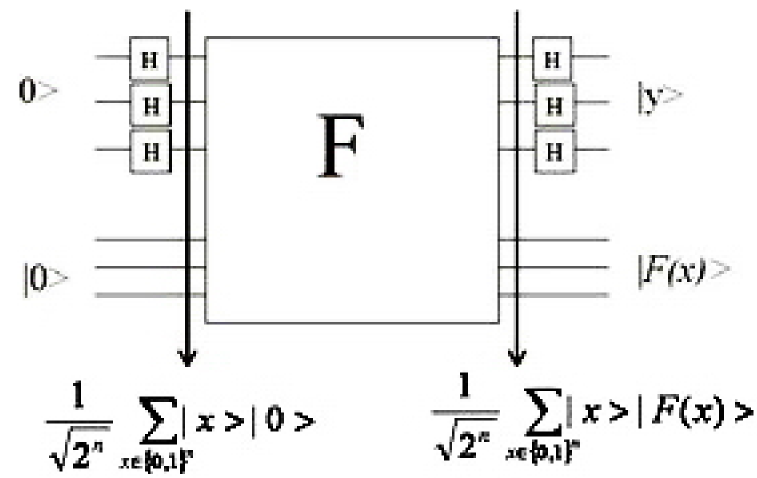
Для одного кубита:

$$|0\rangle \to \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$
$$|1\rangle \to \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$$

А на n кубитах:

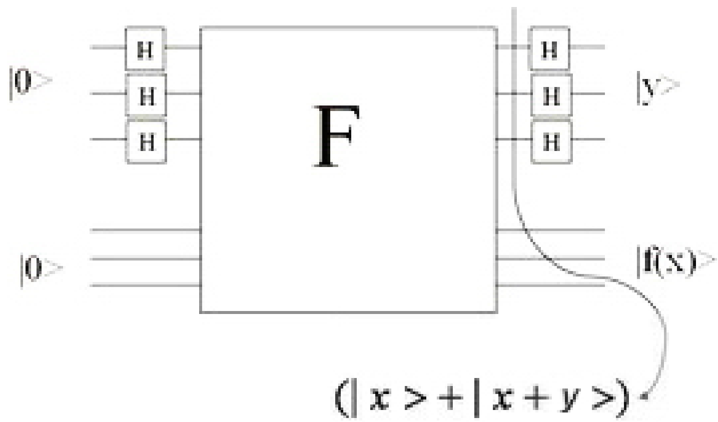
$$|0^n\rangle \to (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle)^n = \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle$$

Рассмотрим схему решения, предложенного Саймоном.



Остановимся подробнее на схеме. Изначально имеем систему в базовом состоянии, когда все $(n+k)$ битов равны нулю. После применения к первым n битам гейта Адамара, получим систему в промежуточном состоянии, содержащую 2^n векторов (все слова из нулей и единиц длины n), дополненных k нулевыми битами. Затем вычислим функцию для нашей системы, в результате получим 2^n векторов, в которых n первых бит — аргумент, а k последних бит — значение функции F для данного аргумента. После этого, измерив биты, содержащие $F(x)$, получим некоторое значение $|z\rangle$. Состояние верхних n битов перейдет в $\sum_{F(x)=z} |x\rangle$.

В нашем случае, получим состояние $|x\rangle|z\rangle + |x+y\rangle|z\rangle$. Как найти число y?



Применим к верхним битам гейты Адамара.

$$|x\rangle \to \frac{1}{\sqrt{2}}|0\rangle + (-1)^x \frac{1}{\sqrt{2}}|1\rangle$$

Для n переменных:

$|x_1 \dots x_n\rangle \to \frac{1}{\sqrt{2^n}} \sum_{u_1 \dots u_n} (-1)^{\sum x_i u_i} |u_1 \dots u_n\rangle$

$|x_1 \dots x_n + y_1 \dots y_n\rangle \to \frac{1}{\sqrt{2^n}} \sum_{u_1 \dots u_n} (-1)^{\sum x_i u_i + y_i u_i} |u_1 \dots u_n\rangle$

Знаки совпадают в точности для тех $|u\rangle$, для которых $y \cdot u = 0$. Теперь, измерив биты, получим один такой вектор u. Повторив много раз, сможем восстановить y.

Итог: с помощью квантового компьютера научились быстро решать задачу Саймона.

3 Алгоритм Шора

3.1 Основные шаги

1. Выбрать случайный остаток a по модулю N
2. Проверить НОД$(a, N) = 1$
3. Найти порядок r остатка a по модулю N
4. Если r четен, вычислить НОД$(a^{r/2} - 1, N)$

Анализ алгоритма: с большой вероятностью полученное на четвертом шаге число будет нетривиальным делителем N.

Трудный шаг: найти порядок a по модулю N.

Определение: миниамальное r такое, что $a^r \equiv 1 \mod N$ называется порядком a по модулю N.

Порядок r является периодом функции $f(x) = a^x \mod N$.

3.2 Разбор алгоритма

Почему алгоритм Шора работает?

Пусть есть число N. Будем случайно подбирать число a так, чтобы оно было взаимно-простым с N. Такое число найдется с большой вероятностью. Повторив несколько раз, найдем такое число (число a называется взаимно-простым к b, если их наибольший общий делитель равен 1). После этого определим порядом r остатка a по модулю N. Получим: $a^r \equiv 1 \mod N$. Мы хотим, чтобы r было четным. Если это не так, вернемся к шагу выбора числа a.

Итак, имеем: $a^r \equiv 1 \mod N$, где r — четно, тогда можно написать:

$a^{r/2^2} \equiv 1 \mod N$, или

$(a^{r/2^2} - 1) \mod N = 0$, или

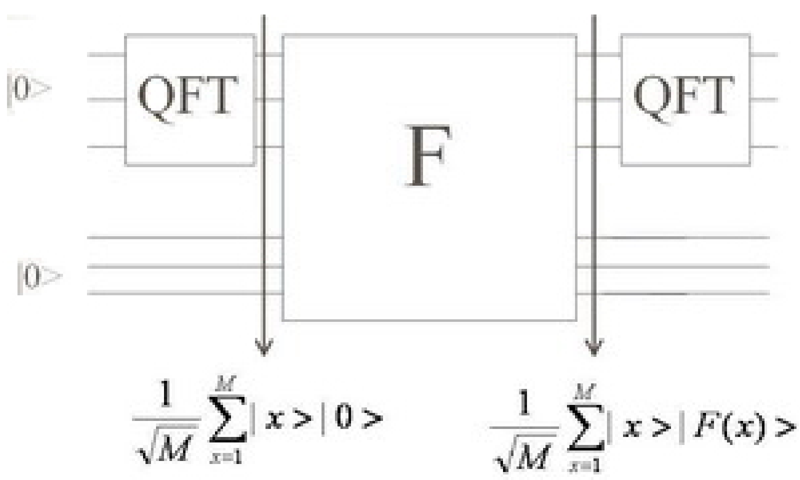
$(a^{r/2} - 1)(a^{r/2} + 1) = cN$, где c — некоторое целое положительное число.

Нетрудно доказать, что одна из скобок имеет с N общий нетривиальный делитель. Тогда, взяв $gcd(x^{r/2} - 1, N)$, получим один делитель N (может получиться, что общий нетривильный делитель число N будет иметь со второй скобкой, тогда мы вновь должны будем повторить выбор числа a). Разделив N на полученное число, находим второй делитель. Задача решена.

Учитывая выкладки, приведенные выше, задача разложения числа N на множители, сводится к быстрому нахождению периода r для случайно подобранного числа a. Для решении данной задачи снова воспользуемся квантовым компьютером.

3.3 Реализация

Рассмотрим следующую схему:



Данная схема почти аналогична рассмотреной выше схеме Саймона. Главным отличием является то, что, вместо преобразования Адамара, используется квантовое преобразование Фурье, и количество векторов равно не 2^n, как в схеме Саймона, а M, где M - количество степеней x, для которых мы хотим узнать остатки. M - положительное число из интервала, причем $M >= N^2$. Взяв число M степенью двойки из интервала $[N^2, 2 \cdot N^2)$, можно заменить первое преобразование Фурье, обычным преобразованием Адамара для верхних $log(M)$ бит.

Рассмотрим состояние системы после вычисления функции F и измерения нижних $ceil(log(N))$ битов (именно столько битов мы зарезервировали для хранения значений функции, $ceil$ - обозначение математического потолка для числа). Система будет находиться в состоянии: $|d\rangle + |d+r\rangle + \dots$, где d - наименьший аргумент, для которого $F(d) = x$ (x - какое-то число, которое мы померили), а r - период.

Опять появилась задача извлечения числа r. Для того, чтобы узнать r, найдем сначала M/r, после чего будет нетрудно вычислить и само число r.

Для того, чтобы найти M/r произведем еще раз преобразование Фурье и сделаем измерение.

Дискретное преобразование Фурье, вычисленное на состоянии $|a\rangle$, поменяет его следующим образом:

$$|a >= \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c > e^{2\pi i a c/q}$$

где q - количество базовых значений системы.

Преобразование, примененное ко всей системе, преобразует ее к следующему виду:

$$\frac{1}{\sqrt{||A||}} \sum_{a' \in A} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} |c, k > e^{2\pi i a' c/q}$$

где $||A||$ — количество элементов в системе.

После преобразования Фурье и измерения всех битов, с большой вероятностью получим число M/r, если M кратно r, и число близкое к M/r в обратном случае. Повторив много раз, узнаем действительное значение M/r.

4 Задача

Как будет работать алгоритм Саймона, если период y не единственный? Можете ли вы модифицировать алгоритм на этот случай?