Обход конечным автоматом с четырьмя камнями k-мерного пространства за полиномиальное время

Гусев Даниил, ФИВТ МФТИ

2 апреля 2014 г.

1 Введение

Данная работа посвящена исследованию задач о поведении роботов в лабиринтах. Спектр подобных задач довольно обширен и затрагивает ключевые аспекты теоретической Computer Science. Конечно, решение таких задач не означает автоматическое решение сложных проблем теории сложности. Тем не менее рассмотрение данных вопросов может положительно сказаться на понимании сути теоретической Computer Science. Есть надежда, что поведение роботов с лабиринтах является хорошей моделью для нетривиальных теоретико-информационных задач, и отработка методов и подходов к исследованию поведения роботов даст более серьезные результаты с будущем. Также, стоит отметить, что подобные задачи и в частности задача из этой статьи, могут быть интересны для школьников, интересующихся теоретической Computer Science, и их учителей.

Основным действующим лицом исследуемой области является робот (некоторый конечный автомат), двигающийся в среде. Средой может является некоторый граф, часто это граф Келли некоторой группы, и основной задачей робота является посещение всех вершин этого графа. Довольно полный обзор подобных задач можно найти в этой книге [1]. Часто в среде помещаются некоторые движимые объекты, которые мы будем называть камнями. Робот может взаимодействовать с камнями, ощущая их и двигая. Подобное взаимодействие существенно разнообразит действие робота. Стандартная формулировка задачи в данном случае – это определить какое минимальное количество камней необходимо роботу для обхода среды. Некоторые примеры подобных задач рассматриваются здесь [2]. Также довольно интересными может оказаться рассмотрение поведения робота с случайной среде. Проблемы такого рода предполагаются к исследованию в последующих работах. Добавление возможности роботу пользоваться датчиком случайных битов может служить хорошей иллюстрацией значимости случайности в алгоритмических задачах.

В данной роботе будут рассмотрены вопросы построения полиномиальных по времени алгоритмов обхода роботом пространства $\mathbb{Z}^k$. Задача обхода роботом пространства $\mathbb{Z}^k$ полностью решена, то есть известны алгоритмы с минимальным количеством камней для всех k и известно, что они оптимальны. Для $k = 1$ необходимо 2 камня, для $k > 2$ 3. Тем не менее при $k > 3$ оптимальные алгоритмы используют эмуляцию машины Минского [3], что приводит к их очень медленной работе. Соответственно, встаёт вопрос построение быстрых алгоритмов обхода $\mathbb{Z}^k$. Быстрота может быть выражена следующим образом: робот сделает полиномиальное от максимума модулей координат

количество ходов, чтобы посетить точку с такими координатами. Для маленьких k задача имеет известные решения в частности для $k = 1$ необходимо 2 камня, для $k = 2, 3$ С.Буньковой показано, что достаточно 3-х камней. Алгоритм, представленный в этой работе, делает это с использованием 4-х камней.

2 Постановка задачи

Начнём с более формального описания модели, в которой мы будем работать, и задачи, которую мы хотим решить.

2.1 Базовые определения

В рамках данной работы, средой, в которой движется робот, будем считать некоторую группу. Тогда нашу модель можно описать с помощью следующих определений.

Определение 1. *Системой лабиринт-робот называется* $L = (G, R, n, D)$*, где* G *– некоторая группа,* R *- конечный автомат специального вида,* n *- неотрицательное целое число,* D *– подмножество* G*.*

Определение 2. *Конечный автомат* $R = (Q, q_0, \delta)$ *называется автоматом робота.* Q *– множество состояний автомата,* $q_0 \in Q$ *– начальное состояние автомата.* $\delta \subset Q \times \{0,1\}^n \to Q$ *– функция переходов в автомате.*

Замечание 1. *Переходы в автомате* $R = (Q, q_0, \delta)$ *осуществляются по битовым векторам длины* n*.*

Определение 3. *Состоянием системы лабиринт-робот называется набор* $(a, s_1, s_2, ..., s_n, q)$*, где* $a \in G$*,* $s_i \in G$*,* $q \in Q$*. Будем называть* a *– расположением робота,* s_i *- расположениями камней. Начальным состоянием системы лабиринт-робот будет* $(e, e, e, ..., e, q_0)$*, где* e *– нейтральный элемент группы* G*.*

Определение 4. $\{d_1, d_2, ..., d_m\} = D$ *называются переходными элементами системы лабиринт-робот.*

Определение 5. *Ходом в состоянии* $q \in Q$ *называется* (d, p)*, где* $d \in D$*,* $p \in \{0,1\}^n$*, причём если в состояние есть переход по вектору* p'*, в котором* i*-й бит* 0*, то в* p i*-й бит должен быть* 0*.*

Определение 6. *Результатом хода робота в состоянии системы* $(a, s_1, s_2, ..., s_n, q)$ *будет состояние* $(a', s'_1, s'_2, ..., s'_n, q')$ *со следующими свойствами. Пусть* (d, p) *соответствующий состоянию* q*;* $a' = ad$*,* $s'_i = s_i d$*, если* i*-ый бит* p *равен* 1*, иначе* $s'_i = s_i$*;* $w \in \{0,1\}^n$*, причем* i*-ый бит* w *равен* 1*, если* $a' = s'_i$*, иначе* 0*; из состояния* q *в автомате* R *по вектору* w *есть переход в состояние* q'*.*

В целом система лабиринт-робот L работает так. Начинаем с начального состояния, и поочерёдно изменяем состояния согласно ходам робота. То есть, можно сказать, что данная система генерирует последовательность состояний $\{l_k\}_{=1}^{\infty}$, согласно вышеописанным правилам.

Лемма 1. *Пусть система находится в состоянии* $l_k = (a, s_1, s_2, ..., s_n, q)$*, и* q *соответствует* (d, p)*. Тогда, если* i*-ый бит* p *равен* 1*, то* $a = s_i$*.*

Доказательство. В начальном состоянии это выполняется, т.к. для любого i $a = s_i$. Докажем для произвольного состояния $(a, s_1, s_2, ..., s_n, q)$. Пусть q соответствует (d, p), тогда если i-ый бит p равен 1, то, согласно определению, все вектора, по которым есть переходы в q, имеют i-ый бит равный 1. Следовательно, по определению результата хода робота для состояния l_{k-1}, $a = s_i$. ■

Более наглядно всю эту конструкцию можно представить так. Имеется робот, которому соответствует конечный автомат R, n камней и сам лабиринт (группа G). В каждый момент времени робот и камни находится в каком-то, элементе группы G, a и s_i соответственно. Также в каждый момент времени определено состояние автомата q. Каждому состоянию q соответствует ход робота (d, p), где d какой-то элемент группы, p – номера камней, которые необходимо подвинуть. В каждый момент времени робот делает ход, т.е. ходит сам и двигает некоторый набор камней (возможно пустой) в направлении элемента d. При этом, согласно лемме, могут двигаться только те камни, с которыми робот имеет одну позицию. После сделанного хода робот понимает, какие камни находятся с ним в одной позиции, и в соответствии с этим знанием изменяет состояние своего автомата.

Отметим, что с подобным определением не совсем удобно работать, поскольку построение конечного автомата R может быть технически сложной задачей. Поэтому мы будем представлять робота скорее как некоторую программу, пользующуюся конечной памятью. Аналогами переходов в автомате будут специальные инструкции «сделать ход в таком-то направлении взяв с собой такие-то камни». Эти инструкции будут возвращать наборы камней, с которыми робот находится в одной позиции. Более менее понятно почему эти два определения будут эквивалентны. С одной стороны конечный автомат может сэмулировать такую программу, взяв за состояния текущее местоположение в программе и состояние её памяти (их конечное количество). С другой стороны довольно легко можно написать программу, которая реализует конечный автомат. Тем не менее в ходе наших рассуждений, подобные программы, реализующие необходимую нам логику, мы не будем описывать явно, просто будем говорить, что они существуют.

2.2 Постановка задачи в общем случае

Задачи, которые предлагается решать в связи с системой робот-лабиринт, могут быть описаны в следующем ключе. Пусть зафиксирована группа G, направления ходов D, n количество камней. Вопрос, существует ли такой робот (автомат R), который сможет обойти все элементы G, т.е. для любого $a \in G$ будет существовать k, такое, что $l_k = (a, ...)$. Ещё более интересным вопросом может быть следующий: каково минимальное количество камней n при фиксированных группе G, направлениях возможных ходов D, при которых существует робот обходящий G.

Выбор группы G и множества D может быть различным. Обычно множество D состоит из порождающих элементов группы и их обратных. Например, для свободной группы F_2 порожденной a, b, множество D логично взять a, b, a^{-1}, b^{-1}. Случай, который будем изучать мы, это случай $\mathbb{Z}^k$. В этом случае множество D будет состоять из базисных элементов и их отрицаний.

Также может стать интересным вопрос количества ходов, которое потребуется для обхода. В случае бесконечных групп его можно выразить в следующих терминах: какое количество ходов потребуется для того, чтобы обойти какое-либо подмножество G (интересует прежде всего зависимость количества ходов от расположения, размера и других характеристик подмножества).

2.3 Постановка задачи в случае $\mathbb{Z}^k$

В случае $\mathbb{Z}^k$ минимальное количество камней, необходимых для обхода, зависит от k. В случае $k = 1$ необходимо 2 камня, в случае в остальных случаях известно, что можно обойтись 3-мя камнями. При этом эти оценки точны, то есть известно, что с меньшим количеством камней это сделать нельзя. Однако в известных решениях этой задачи, в случае $k > 3$ для обхода $\mathbb{Z}^k$ робот эмулирует машину Минского. Несмотря на то, что вычислительная мощность машины Минского и машины Тьюринга одинаковы, реализация даже самых простых алгоритмов требует экспоненциального от размера входа времени. Поэтому робот, использующий машину Минского для обхода $\mathbb{Z}^k$, потратит как минимум экспоненциальное время для её посещения. Тем не менее хочется предъявить какие-либо полиномиальные обходы при всех k. Формально это можно описать так. Введём на $\mathbb{Z}^k$ систему координат, тогда определение полиномиальности будет следующим.

Определение 7. *Пусть робот начинает двигаться из начала координат и $T(d)$ это номер хода, когда впервые роботом посещена клетка d. Тогда обход считается полиномиальнымым, если существуют константы C, r такие, что для любой d $T(d) < Cp(d)^r$, где $p(d)$ – максимум модулей координат d.*

Для $k = 1, 2$ стандартные обходы (с 2-мя и 3-мя камнями) и так являются полиномиальными). Для $k = 3$ также известен полиномиальный обход использующий 3 камня. Для больших размерностей можно обобщить обход для $k = 3$, и получить полиномиальные обходы использующие k камней. Собственно, с улучшением последних оценок связана данная работа. В её рамкам будет предъявлен полиномиальный обход, использующий 4 камня, и работающий для любой размерности $k > 1$.

Общая схема доказательства существования такого обхода следующая. Для начала показываем, каким образом можно получить двоичное представление числа из внешней памяти (внешней памятью можно назвать некоторую специальную конфигурацию камней в пространстве). Потом предъявляем обход $\mathbb{Z}^k$ (в данном случае обход это некоторый путь на $\mathbb{Z}^k$, обходящий всё пространство), при этом описывается способ получения направления хода по его номеру в пути. Далее описываем сам обход: робот будет обходить $\mathbb{Z}^k$ по построенному нами пути, храня во внешней памяти номер текущего хода. Умение узнать двоичное представление номера хода позволит нам узнавать необходимое направление движения.

3 Простые алгоритмы обхода при различных k

Перед тем как представить основной результат работы, рассмотрим простые алгоритмы обхода $\mathbb{Z}^k$ при различных k, про которые говорилось выше. В частности будут описаны алгоритмы для $k = 1, 2$, семейство полиномиальных алгоритмов для больших размерностей и алгоритм эмулирующий машину Минского и использующий 3 камня.

3.1 Случай $k = 1$

Случай $k = 1$ естественно самый простой. Для обхода $\mathbb{Z}$ роботу достаточно использовать всего 2 камня (A и B). При описании будем считать, что на $\mathbb{Z}$ есть система координат, и робот и камни

располагаются в точках (клетках) с соответствующими им координатами. Схема алгоритма обхода в виде действий робота представлена ниже.

Начальная конфигурация: камни A и B и робот в точке 0.

1: **цикл** // *повторять бесконечно*
2: передвинуть камень A в положительную сторону
3: идти в отрицательную сторону до встречи с камнем B
4: передвинуть камень B в отрицательную сторону
5: идти в положительную сторону до встречи с камнем A

Очевидно, что в результате исполнения подобного алгоритма любая клетка $\mathbb{Z}$ будет посещена каждая клетка, причем легко посчитать, что для $n > 0$ потребуется $2n^2 - n$ ходов, чтобы посетить клетку с координатой n, а для $n < 0$ $2n^2 + n$ ходов.

3.2 Случай $k = 2$

Алгоритм обхода пространства $\mathbb{Z}^2$ будет использовать 3 камня. Один из вариантов построения этого алгоритма – это использование обхода всех клеток прямоугольного равнобедренного треугольника с стороной n с использованием 3-х камней (A, B и C). Схема этого обхода ниже

Начальная конфигурация: A, B и робот в точке $(0,0)$, C в точке $(n,0)$

Конечная конфигурация: A, B в точке $(n,0)$, C в точке (n,n), все клетки треугольника $(0,0)$, $(n,0)$, (n,n) посещены

1: **цикл**
2: перенести A в направлении $(1,0)$
3: перейти направлении $(-1,0)$
4: идти до камня B в направлении $(0,1)$
5: перенести B в направлении $(1,1)$
6: идти до камня A в направлении $(0,-1)$
7: **если** A и C в одной клетке **то**
8: **выход из цикла**
9: поменять местами B и C

Легко убедиться, что подобный алгоритм действительно исполняет требуемое, при этом он выполняется за $\Theta(n^2)$ ходов (каждая клетка треугольника посещается не больше константы раз, в основном 2 раза).

Теперь можно легко предъявить алгоритм обхода квадрата со стороной n.

Начальная конфигурация: A, B и робот в точке $(0,0)$, C в точке $(n,0)$

Конечная конфигурация: A, B и робот в точке $(0,0)$, C в точке $(n,0)$, все клетки квадрата $(0,0)$, $(n,0)$, (n,n) $(0,n)$ посещены

1: обойти треугольник $(0,0)$, $(n,0)$, (n,n)
2: обойти треугольник $(n,0)$, (n,n), $(0,n)$
3: обойти треугольник (n,n), $(0,n)$, $(0,0)$
4: обойти треугольник $(0,n)$, $(0,0)$, $(n,0)$

Аналогично асимптотика этого алгоритма $\Theta(n^2)$. В итоге обход всей $\mathbb{Z}^2$ можно совершить следующим образом.

Начальная конфигурация: A, B, C и робот в точке $(0,0)$

1: передвинуть C в направлении $(1,0)$
2: **цикл** // *i-ая итерация, A, B в точке* $(-i+1,-i+1)$, *C в точке* $(i,-i+1)$
3: обойти квадрат $(-i+1,-i+1)$, $(i,-i+1)$, (i,i), $(-i+1,i)$
4: передвинуть A и B в точку $(-i,-i)$
5: дойти до точки C и передвинуть в $(i+1,-i)$
6: вернуться в $(i,-i+1)$

Главная идея алгоритма – обходить квадраты, которые расширяются на каждой итерации на одну клетку в каждую сторону. Так как на каждую итерацию тратится квадратичное от её номера количество ходов, для обхода всех клеток с координатами по модулю меньшими n потребуется $\Theta(n^3)$ ходов.

3.3 Случай $k > 2$

Для начала предъявим алгоритм обхода $\mathbb{Z}^3$ с использованием 3-х камней. Общая схема обхода такова: будем поочерёдно обходить границы вложенных друг в друга кубов K_n с диагоналями $(-n+1,-n+1,-n+1)$ (n,n,n), начиная с $n = 1$. Границы таких кубов представляют собой 6 квадратов, поэтому их можно обойти с помощью обходов квадратов, представленных выше. Точное доказательство того, что это можно проделать с помощью 3 камней здесь переводить не будем. Однако заметим, что границу каждого K_n в данной схеме можно обойти за квадратичное по n время, поэтому обход целых K_n будет занимать $\theta(n^3)$ времени.

Для произвольного k алгоритм будет основан на той же идее, и будет использовать k камней. Будем обходить границы k-мерных кубов K_n с диагоналями $(-n+1,...,-n+1)$ $(n,...,n)$. Границы этих кубов составлены из $2k$ $k-1$-мерных кубов, поэтому нам необходим алгоритм обхода $k-1$-мерного куба со стороной n. Этот алгоритм будет состоять в следующем: пусть $k-1$ находится в одной точке, и ещё один камень на расстоянии n от них, например $(0,...,0)$ и $(n,...,0)$ (векторы длиной $k-1$). Будем обходить пространство $\mathbb{Z}^{k-1}$ с помощью $k-1$ камня, только в положительных координатах (для этого кубы K_n нужно расширять не по всем $2k-2$ направлениям, а по $k-1$) до тех пор по не наткнёмся на удалённый камень. Доказательство того факта, что с помощью можно обойти $\mathbb{Z}^k$ здесь приводить не будет, однако отметим, что асимптотика подобного обхода будет $\theta(n^k)$.

3.4 Обход с помощью машины Минского

Как говорилось раньше любое $\mathbb{Z}^k$ можно обойти, используя всего 3 камня. Для это роботу необходимо сэмулировать машину Минского. Машина Минского представляет собой конечный автомат и два счётчика, которые можно инкрементировать и декрементировать. Подобная система имеет ту же вычислительную мощность, что и машина Тьюринга. Робот может довольно легко сэмулировать машину, для этого ему достаточно хранить камни B и C на расстоянии значений счётчиков от камня A. Для обхода такой конструкцией $\mathbb{Z}^k$ можно использовать путь обходящий $\mathbb{Z}^k$ (например, такой как мы опишем ниже), то есть некоторую бесконечную последовательность направлений,

причем i-ое направления алгоритмически вычислимо по i. Тогда робот будет хранить в машине номер текущего направления, вычислять его сдвигать по этому направлению всю машину и увеличивать номер на единицу. Если считать что «центр» машины это камень A, тогда в ходе такого алгоритма A обойдёт все пространство. Отметим, что примерно такой же схемой мы воспользуемся в нашем алгоритме, только для считывания номера, будет использоваться 4 камня. Это позволит нам избавится от экспоненциальной асимптотики.

4 Получение двоичного представления числа

Первым результатом приближающем нас по построению полиномиального обхода, будет алгоритм получения получение двоичного представления числа. Предположим, что робот имеет во внешней памяти некоторую информацию, к примеру число n. Самый простой способ хранения n – это расположить два камня таком расстоянии. Очевидно, что для обработки такой информации требуется нетривиальная логика, т.к. внутренняя память робота ограничена константой. Способ обработки этой информации может существенно зависеть от задачи, которую необходимо выполнить роботу. В рамках нашей задачи необходимо получить двоичное представление числа n, начиная с младшего бита до самого старшего единичного. Более конкретно это означает, что робот получает биты в потоковом режиме, т.е. повторяет два действия: узнать следующий бит числа, если таковой имеется, проделать какую-либо внутреннюю логику (обработать этот бит). Покажем, что существует быстрый (полиномиальный от n) алгоритм получения числа n таким способом, использующий 2 дополнительных камня.

Пусть имеются камни A и B задающие n, будем считать, что они неподвижны. Также есть два вспомогательных камня C и D. В рамках данного алгоритма все камни будут находятся на одной прямой, для удобства введём на ней систему координат.

Для начала предъявим алгоритм деления отрезка длины k пополам и получение остатка при этом делении. Схема алгоритма приведена ниже.

Начальная конфигурация: камни C и D в точках с координатами 0 и k, робот в точке 0

Конечная конфигурация: C, D и робот в точке $\lfloor k/2 \rfloor$, в память записан остаток при делении на 2

1: **цикл**
2: идти в положительную сторону до камня D
3: передвинуть камень D на одну клетку в отрицательную сторону
4: **если** камни C и D на одной клетке **то**
5: записать в память, что остаток равен 1
6: **выход из цикла**
7: идти в отрицательную сторону до камня C
8: передвинуть камень C на одну клетку в положительную сторону
9: **если** камни C и D на одной клетке **то**
10: записать в память, что остаток равен 0
11: **выход из цикла**

Легко видеть, что после d итераций цикла координаты камней C и D будут d и $k - d$ соответ-

ственно, поэтому при четном k алгоритм завершиться после $k/2$ циклов с нужным результатом, а при нечётном после $\lceil k/2 \rceil$ циклов.

Теперь пользуясь данным делением, предъявим алгоритм получения всех битов числа n начиная с младшего.

Начальная конфигурация: камни A, C, D и робот в точке 0, B в точке n

Конечная конфигурация: камни A, C, D и робот в точке 0, B в точке n, в память поочерёдно записаны биты числа n начиная с младшего

1: идти в положительную сторону с камнем D до камня B
2: вернуться в точку 0
3: **цикл** // *i-ая итерация, C в точке 0, D в точке $\lfloor n/2^{i-1} \rfloor$*
4: выполнить деление для камней C и D // *C и D в точке $\lfloor n/2^{i} \rfloor$*
5: записать остаток при делении в память
6: выполнить внутреннюю логику // *эта часть зависит от применения алгоритма*
7: **если** C и D находятся в точке 0 **то**
8: **выход из цикла**
9: **иначе**
10: перенести C в точку 0

Заметим, что в ходе i-ой итерации цикла происходит деление отрезка длиной $\lfloor n/2^{i-1} \rfloor$. Остаток при этом делении соответствует i-ому биту числа n (если младший бит принять первым), поэтому алгоритм будет записывать в память действительно требуемый результат. Теперь осталось доказать, что алгоритм работает за допустимое время.

Лемма 2. *Алгоритм получения битов числа n работает за время $T(n) = \Theta(n^2)$.*

Доказательство. Посчитаем для начала сколько ходов тратиться на каждую операцию деления. Пусть начальное расстояние между C и D равно k. Тогда k ходов потратится на сдвиги камней, а на проходы между камнями будет потрачено $\sum_{i=1}^{k} i$. Итого всего ходов на одно деление потратится:

$$k + \sum_{i=1}^{k} i = k + \frac{(k+1)k}{2} = \frac{(k+3)k}{2}$$

Очевидно, что в ходе алгоритма будет выполнено деление числа n, поэтому $T(n) \geq (n+3)n/2$. Для оценки сверху посчитаем количество всех ходов в алгоритме. Сначала будет выполнено n ходов, потом для каждой операции деления числа k будет выполнено k дополнительных ходов и сама операция деления (всего $(k+5)k/2$). Пусть $2^s - 1$ наименьшее число такого вида, которое не меньше n. В алгоритме для $2^s - 1$ и n одинаковое количество делений (s), но каждое деление в алгоритме для $2^s - 1$ занимает не меньше времени, чем соответствующие деление для n. Поэтому верно следующее:

$$T(n) \le T(2^s - 1) = 2^s - 1 + \sum_{i=1}^{s} \frac{(2^i + 4)(2^i - 1)}{2} =$$

$$= 2^s - 1 + \sum_{i=1}^{s} (2^{2i-1} + 3 * 2^{i-1} - 4) \le 2^{s+2} + \sum_{i=1}^{s} 2^{2i-1} \le$$

$$\le 2^{s+2} + \sum_{i=1}^{2s-1} 2^i < 2^{s+2} + 2^{2s} \le 8n + 4n^2$$

Таким образом верно $n^2/2 \le T(n) \le 8n + 4n^2$ откуда следует требуемое. ■

5 Построение пути обхода $\mathbb{Z}^k$

Существенная часть алгоритма полиномиального обхода роботом $\mathbb{Z}^k$ – это построение пути обхода $\mathbb{Z}^k$. Обход, который будет построен нами построен, пройдет через каждую клетку пространства ровно 1 раз. Общий подход к его построению таков: строим семейство обходов k-мерных кубов с длинами сторон равными степеням двойки, потом на основе этих обходов получаем обход всего пространства. Опишем основные термины, в которых будем производить построение.

На $\mathbb{Z}^k$ есть $2k$ возможных направлений движения. Если есть направление движения i, противоположное направление будем обозначать $-i$. Введём определение пути.

Определение 8. *Путь – упорядоченный набор, конечный или бесконечный, направлений движения. Если m – путь, то $m[i]$ – направление движение i-ого хода этого пути. Нумерация ходов ведётся с единицы.*

Определение 9. *Пусть $m = a_1 a_2 ... a_k$ – путь, тогда:*

1. $-m = -a_1 - a_2 ... - a_3$
2. $\overline{m} = a_k a_{k-1} ... a_1$
3. $\overline{-m}$ *– обход пути m в обратную сторону.*

Замечание 2. *Конечные самонепересекающиеся пути длины k затрагивают $k + 1$ клетку пространства.*

Замечание 3. *Чтобы получить сдвиг между начальной и конечной точкой пути по направлению i, нужно просуммировать все ходы в направлении i с плюсом и в направлении $-i$ с минусом.*

Основным инструментом построения обхода будет группа G_k, действующая на направлениях движения. Она будет действовать так: если элемент $g \in G_k$ переводит направление i в j, то он $-i$ переведет в $-j$. Эта группа изоморфна $S_k \times \mathbb{Z}_2 \times ... \times \mathbb{Z}_2$ ($\mathbb{Z}_2$ - k раз). Тривиально определяется действие этой группы на путях одинаковой длины: $g(a_1 a_2 ... a_k) = g(a_1) g(a_2) ... g(a_k)$, где $g \in G_k$, $a_1 a_2 ... a_k$ - путь длины k.

5.1 Обход k-мерного куба со стороной со стороной 2

Для построения обхода всего пространства нам потребуются обходы кубов с различными длинами сторон. Начнём с построения обхода куба со стороной 2. На самом деле, построенный нами обход

эквивалентен обходу полученному с помощью кодов Грея [4]. Тем не менее необходимо описать подобный обход в наших терминах.

Рассмотрим k-мерный куб со стороной 2, $k \geq 1$. Обход такого куба можно представить как самонепересекающийся путь длины $2^k - 1$. Построим семейство M_k таких обходов следующим образом: предъявим один обход, а остальные получим как орбиту действия G_k над этим обходом. Заметим, что любой $g \in G_k$ соответствует какому-то движению в пространстве $\mathbb{Z}^k$ поэтому при действии элементами G_k на обход куба, будет получатся также обход куба.

Обозначим направления движения $x_1, -x_1, x_2, -x_2, x_3, -x_3 ....$ Рассмотрим последовательность путей $\{m_i\}$ построенную по следующему рекуррентному правилу: $m_1 = 1$, $m_i = m_{i-1} x_i \overline{-m_{i-1}}$.

Лемма 3. *Путь m_k с началом в клетке $(\underbrace{0,0,...,0}_{k})$ обходит куб с диагональю $(\underbrace{0,0,...,0}_{k})$ $(\underbrace{1,1,...,1}_{k})$ (номера координат соответствуют номерам направлений).*

Доказательство. Будем доказывать по индукции. Для $k = 1$ доказательство очевидно. Пусть для $k = i - 1$ факт верен. Докажем для $k = i$.

Заметим, m_{i-1} – обход куба с диагональю $(\underbrace{0,0,...,0}_{i})(\underbrace{1,1,...,1}_{i-1},0)$. Ход по направлению x_i переводим нас в точку с последней координатой равной 1. Так как путь $\overline{-m_{i-1}}$ обратен пути m_{i-1}, в проекция на первые $i-1$ координату мы вернёмся обратно, точно таким же путём как и пришли. Следовательно, учитывая ход в направлении x_i, с помощью $\overline{-m_{i-1}}$ мы обойдём куб с диагональю $(\underbrace{0,0,...,0}_{i-1},1)(\underbrace{1,1,...,1}_{i})$. Заметим, сначала мы обошли «нижнюю» половину куба, перешли в «верхнюю» и обошли её также как и «нижнюю», только в обратную строну. Итого каждая клетка куба с диагональю $(\underbrace{0,0,...,0}_{i})(\underbrace{1,1,...,1}_{i})$ была посещена. ■

Рассмотрим теперь произвольный элемент $m \in M_k$. Так как M_k – орбита элемента m_k при действии группы G_k, m можно представить в следующем виде $m = g(m_k), g \in G_k$. Докажем несколько лемм.

Лемма 4. *Начальная и конечная точки пути m соседние, причём чтобы перейти из начала в конец необходимо пойти по направлению $g(x_k)$.*

Доказательство. Достаточно доказать этот факт для m_k и направления x_k. Посчитаем сдвиг по каждому направлению. Заметим, что каждому направлению в m_{k-1} соответствует противоположное направление в $\overline{-m_{k-1}}$. Поэтому сдвиги m_{k-1} и $\overline{-m_{k-1}}$ компенсируются. Таким образом остаётся один ход в направлении x_k. Отсюда следует требуемое. ■

Определение 10. *Будем называть направление $g(x_k)$ основным направлением $m = g(m_k)$. Ребро между начальной и конечной клеткой также будем называть основным.*

Определение 11. *Пусть c – клетка куба C, который обходит путь m. Тогда будем говорить, что направления $a_1, a_2, ...a_k$ ребер куба C выходящих из c задают клетку при обходе m.*

Лемма 5. *Начальная клетка обхода m задаётся $g(x_1)$, $g(x_2)$, ... $g(x_k)$, конечная $g(x_1)$, $g(x_2)$, ..., $g(x_{k-1})$, $g(-x_k)$.*

Доказательство. Для доказательства достаточно показать, что начальная и конечные клетки m_k задаются $x_1, x_2, ..., x_k$ и $x_1, x_2, ... - x_k$. Действительно, при построении m_k начальная клетка была 0,

а весь куб лежал в неотрицательной части пространства. А конечная клетка должна только одним направлением от начальной, и это направление x_k (заменятся на $-x_k$). ■

Лемма 6. *Количество путей из M_k имеющих одинаковое основное направление и одинаковые направления, задающие начальную клетку (следовательно и конечную) равно* $(k-1)!$

Доказательство. 2^k вариантов выбрать направления для начальной клетки, из них выбрать основное k вариантов, итого $k2^k$ вариантов. Все такие конфигурации равнозначны. Мощность M_k – $k!2^k$, следовательно ответ $k!2^k/(k2^k) = (k-1)!$ ■

5.2 Обход k-мерного куба со стороной 2^n

Теперь построим множество $M_{k,n}$ обходов k-мерного куба со стороной 2^n, для $k \geq 2, n \geq 1$. Основное свойство, которое мы хотим реализовать в $M_{k,n}$, – это то, что начальная и конечная клетки будут вершинами куба, лежащими на одном ребре. Аналогично будем называть это ребро и направление ему соответствующее основными. Построение будет проводиться также как и для M_k: сначала построим одного базового представителя данного множества, а остальных будем получать как орбиту при действии на этого представителя группой G_k. То что в итоге мы построим будет обобщением кривой Гилберта (Пеано) на k-мерный случай [5, 6, 7].

5.2.1 Общий случай обхода

Для $M_{k,1}$ построение очевидно, просто положим $M_{k,1} = M_k$.

Для $M_{k,2}$ построение будет менее тривиальным. Общий вид базового пути $m_{k,2}$ будет следующим: s, k, $g_{-k}(\overline{-s})$. При этом g_{-k} меняет местами x_k и $-x_k$, s – это последовательность следующих ходов $g_1(m_k)$, $m_k[1]$, $g_2(m_k)$, $m_k[2]$, ... , $m_k[2^{k-1}-1]$, $g_{2^{k-1}}(m_k)$, где $g_i \in G_k$ некоторые элементы группы. Поскольку $m_k = m_{k-1}x_k\overline{-m_{k-1}}$ и в m_{k-1} нет ходов по направлениям x_k и $-x_k$, по другому $m_{k,2}$ можно представить как 2^k упорядоченных элементов из M_k, причем между парами соседних есть ходы из m_k (их как раз 2^k-1). Другими словами $g_1(m_k)$, $m_k[1]$, $g_2(m_k)$, $m_k[2]$, ... , $m_k[2^k-1]$, $g_{2^k}(m_k)$, где $g_i = g_{-k}g_{-}g_{2^k-i+1}$, g_- переводит все i в $-i$.

Геометрически этот этот обход выглядит следующим образом. Наш k-мерный куб C можно представить как куб C_0 со стороной 2, у которого клетки – кубы со стороной 2. Пусть эти клетки $C_1, C_2, ..., C_{2^k}$ занумерованы в таком порядке, в каком базовый путь m_k обходит куб C_0. Тогда, путь $m_{k,2}$ обойдет поочерёдно все $C_1, C_2, ..., C_{2^k}$, делая переходы от C_i к C_{i+1} в соответствии с m_k (то есть просто делая ход в направлении $m_k[i]$).

Посмотрим, как конкретно необходимо обойти C_i (или другими словами задать g_i), чтобы путь получился корректным. Для начала докажем лемму.

Лемма 7. *Пусть C_i и C_{i+1} соседние кубы в C_0, а d клетка в C_i. Тогда существует путь $m \in M_k$ обходящий C_i такой, что клетка d начальная, а конечная клетка касается C_{i+1}.*

Доказательство. Так как начальная клетка пути уже зафиксирована, необходимо правильно выбрать основное ребро. Заметим, что в силу построения C_i и C_{i+1} имеют общую $k-1$-мерную границу. Есть два случая: d касается этой границы и d не касается. В первом случае есть $k-1$ вариантов

выбрать основное ребро так, чтобы конечная клетка тоже лежала на этой границе (все рёбра идущие из d кроме параллельно границе). Во втором случае такое ребро только одно – ребро ведущее из d перпендикулярное границе. С фиксированным основным ребром и фиксированной начальной клеткой есть $(k-1)!$ путей. Таким образом, в одном случае есть $(k-1)(k-1)!$ подходящих путей, в другом $(k-1)!$. ■

Лемма 8. *Обход C_1 всегда имеет основное направление x_1.*

Доказательство. Переход из C_1 в C_2 происходит по направлению $m_k[1] = x_1$. Начальная клетка C_1 задаётся направлениями x_1, x_2, ..., x_k, поэтому выполняется первый случай из доказательства леммы, следовательно основное направление x_1. ■

Теорема 1. *Существует набор g_i, с которыми $m_{k,2}$ действительно обходит куб C.*

Доказательство. Заметим, что в силу построения пути, вторая половина – это первая, отражённая относительно гиперплоскости делящей куб пополам и перпендикулярной x_k и пройденная в обратную сторону. Таким образом, нам необходимо зафиксировать обходы в первых 2^{k-1} маленьких кубов так, чтобы мы дошли из начальной точки до границы $C_{2^{k-1}}$ и $C_{2^{k-1}+1}$ (она лежит на предыдущей гиперплоскости).

В кубе C_1 известна начальная клетка. Это начальная клетка всего пути (её можно задать направлениями x_1, x_2, ..., x_k в большом кубе. Выберем обход в этом кубе так, чтобы конечная клетка касалась C_2 (это можно сделать согласно лемме). Теперь должен произойти ход $m_k[1]$. После него мы действительно в кубе C_2, так как сейчас находимся на границе этих кубов и $m_k[1]$ перпендикулярно её по построению. Теперь нам известна начальная клетка в C_2. Аналогично построим путь в C_2 и так далее, до $C_{2^{k-1}}$ включительно, в итоге получаем требуемый путь. ■

Заметим, что в построенном нами обходе начальная и конечная клетки находятся в вершинах куба C, причем начальная клетка задаётся направлениями x_1, x_2, ..., x_k, а основное направление x_k.

Построим теперь $M_{k,n}$, а точнее одного его представителя $m_{k,n}$. Положим его следующим: $g_1(m_{k,n-1})$, $m_k[1]$, $g_2(m_{k,n-1})$, $m_k[2]$, ... , $m_k[2^k-1]$, $g_{2^k}(m_{k,n-1})$ ($g_i \in G_k$ такие же как и для $m_{k,2}$). Таким образом $m_{k,n}$ выражается рекурсивно через $m_{k,n-1}$.

Теорема 2. *Путь $m_{k,n}$ обходит k-мерный куб со стороной 2^n, причем начальная клетка пути задаётся направлениями x_1, x_2, ..., x_k, а основное направление x_k.*

Доказательство. Докажем по индукции. Для $n = 2$ это утверждение мы уже доказали. Пусть факт верен для $n-1$, докажем для n.

Заметим, что если в пути $m_{k,2}$ все обходы маленьких кубов C_i (они обходятся путями $g_i(m_k)$), заменить на пути $g_i(m_{k,n-1})$, с одной стороны получается путь $m_{k,n}$ (он по определению таков). С другой стороны обходы маленьких кубов C_i заменились на обходы кубов размера 2^{n-1}, причем направления задающие начальные и конечные клетки не изменились (в силу предположения индукции). Следовательно, переходы между кубами остались корректными (если конечная клетка C_i была соседней начальной клетки C_{i+1}, то в новых кубах они тоже будут соседними, причём переход между ними будет иметь такое же направление $m_k[i]$). Начальные и конечные клетки всего пути

аналогично остались x_1, x_2, ..., x_k и x_1, x_2, ..., $-x_k$. Таким образом $m_{k,n}$ действительно поочерёдно обходит 2^k кубов размера 2^{n-1}, то есть куб размера 2^n. ■

5.2.2 Пример выбора g_i

Как было показано выше, существуют такие g_1, g_2, ..., g_{2^k}, при которых пути $m_{k,n}$ обходят кубы соответствующих размеров. При этом, в доказательстве допускалось в некоторых ситуациях выбирать произвольное основное ребро, т.е. $g_i(k)$ было не фиксировано. В этой части мы предъявим конкретный выбор $g_i(k)$, при котором обход будет корректен. Это знание нам необязательно для дальнейшего построения алгоритма, поэтому можно его рассматривать как дополнительную информацию.

Будем, строить значения $g_i(k)$ на основе ходов в пути m_k. Положим $g_1(x_k) = x_1$, $g_{2^{k-1}}(x_k) = x_k$, $g_{2q}(x_k) = g_{2q+1}(x_k) = m_k[2q]$, где q от 1 до $2^{k-2}-1$ (можно задать первые 2^{k-1}, остальные получаются из симметрии). При этом желательно знать направления, задающие начальную клетку $g_i(m_k)$, то есть те направления куда переходят x_1, x_2, ... x_k под действием g_i (не обязательно знать что куда конкретно переходит, можно определить просто множество). Очевидно, что для g_1 это множество и есть x_1, x_2, ... x_k. Теперь пусть для g_i это множество известно и равно a_1, a_2, ... ,a_k (a_j – это или x_j или $-x_j$), тогда если i четное, то для g_{i+1} оно останется таким же, иначе a_1 и $a_{g_i(x_k)}$ обратятся (если $i = 1$, то при этом ничего не изменится).

Лемма 9. *При выше описанных g_i обход куба $m_{k,2}$, а значит и $m_{k,n}$, корректен.*

Доказательство. Рассмотрим произвольное g_i с множеством направлений a_1, a_2, ... ,a_k. Если i – чётное, то основное ребро в обходе куба C_i имеет направление $m_k[i]$. Значит если из начальной клетки C_i мы шагнём в направлении $m_k[i]$ мы попадем в конечную клетку C_i. Если шагнем ещё раз, мы попадём в куб C_{i+1}, причем в клетку с такими задающими направлениями как и начальная клетка C_i. Заметим, что это начальная клетка C_{i+1}, поэтому переход из C_i в C_{i+1} корректен. Аналогично, можно доказать для случая $i = 1$, т.к. $m_k[1] = 1$.

Если i – нечётное, то доказательство будет аналогично предыдущему пункту, только шагать надо сначала в направлении $g_i(x_k)$, при этом обратится направление $g_i(x_k)$ (станет $-g_i(x_k)$), потом в направлении $m_k[i]$, (станет направление $-m_k[i]$). Т.к. все нечётные $m_k[i]$ либо x_1, либо $-x_1$, клетка, которую мы пришли будет задаваться направлениями $-a_1$, a_2, ...,$-a_{g_i(x_k)}$, ...,a_k, что соответствует начальной клетке C_{i+1}, опять получаем, что переход корректен.

Если $i = 2^{k-1}$, то в силу симметрии построения $g_{2^{k-1}} = g_{2^{k-1}-1}$, поэтому здесь ситуация будет аналогична чётному случаю. ■

Таким образом мы получили набор g_i с явно заданными основными направлениями. Заметим, что даже в этих условиях каждое g_i можно выбирать по разному, а именно $(k-1)!$ способами (этот факт мы доказывали ранее). Приведём явные примеры построения $m_{k,n}$ для $k = 2, 3$.

В случае $k = 2$ существует всего один набор g_i образующих корректный путь. Этот набор следующий: g_1 переводит $x_1 \to x_2$, $x_2 \to x_1$, $g_2 = g_3$ тождественны, g_4 переводит $x_1 \to -x_2$, $x_2 \to -x_1$. Сами $m_{2,n}$ можно увидеть на рисунке.

При $k = 3$ каждую g_i можно выбрать как минимум 2 способами, при этом $g_i(x_3)$ равны следующим направлениям x_1, x_2, x_2, x_3, x_3, $-x_2$, $-x_2$, $-x_1$. Примеры на рисунке.

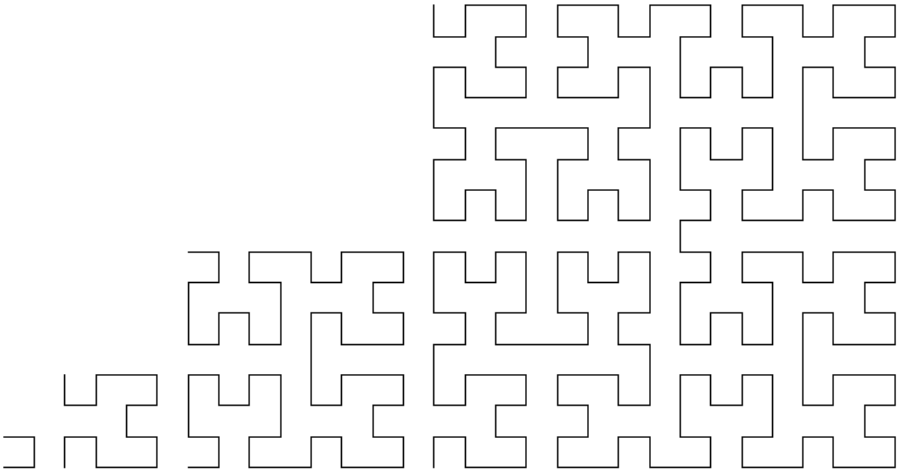

Рис. 1: Пути m_2, $m_{2,2}$, $m_{2,3}$, $m_{2,4}$. Начало координат – левая нижняя точка, направление x_1 – влево, x_2 – вверх

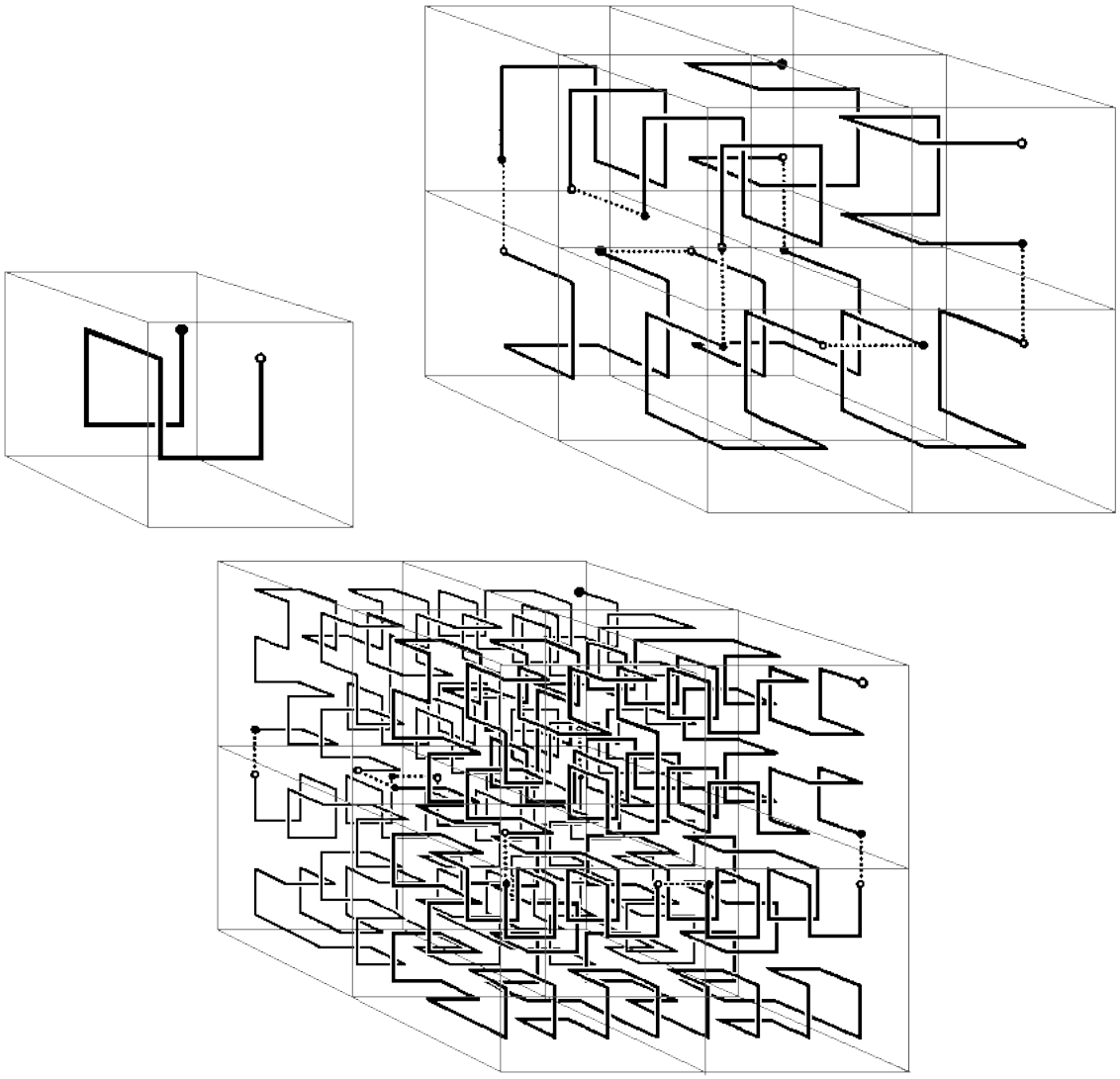

Рис. 2: Примеры путей m_3, $m_{3,2}$, $m_{3,3}$. Начало координат – левая верхняя ближняя точка, направление x_1 – вниз, x_2 – влево, x_3 – вглубь

5.3 Обход $\mathbb{Z}^k$

Построение пути обхода куба со стороной 2^n нам было необходимо для постройки обхода $\mathbb{Z}^k$. Наш обход посетит каждую клетку один раз, причем число ходов необходимое для посещения клетки будет полиномиальным от её координат.

Начнём построение обхода с введения специального элемента $g_H \in G_k$. g_H будет действовать на основные k направлений следующим образом: $x_1 \to -x_1$, $x_2 \to -x_k$, $x_i \to x_{i-1}$, для $i > 2$.

Лемма 10. *Порядок g_H равен $2(k-1)$*

Доказательство. Легко видеть, что при действии g_H образуется два цикла: x_1 и $-x_1$, все остальные направления. Длины циклов 2 и $2(k-1)$, откуда получаем требуемое. ■

Теперь определим рекурсивно следующий бесконечную последовательность путей $\{h_n\}_{n=1}^{\infty}$. $h_1 = m_k$, $h_n = h_{n-1} g_H^{n-1}(l)$, где l – это $m_k[1]$, $g_2(m_{k,n-1})$, $m_k[2]$, ... , $m_k[2^k-1]$, $g_{2^k}(m_{k,n-1})$, то есть $g_H^n(m_{k,n})$ без обхода первого куба со стороной 2^{n-1}.

Теорема 3. *Путь h_n – это обход k-мерного куба со стороной 2^n, причём конечная клетка обхода – это вершина куба заданная направлениями $g_H^n(-x_1)$, $g_H^n(x_2)$, ..., $g_H^n(x_k)$.*

Доказательство. Докажем по индукции. При $n = 1$ достаточно показать, что конечная клетка искомая. Действительно в обходе m_k конечная клетка задаётся направлениями x_1, x_2, ...,$-x_k$. Легко видеть, что $g_H(-x_1)$, $g_H(x_2)$, ..., $g_H(x_k)$ те же самые направления.

Пусть факт верен для $n-1$, докажем для n. Рассмотрим наш куб, как кубы $C_1, C_2, ..., C_{2^k}$ со стороной 2^{n-1}. Заметим, что h_n - это $g_H^{n-1}(m_{k,n-1})$ с заменой обхода куба C_1 на h_{n-1}. Покажем, что замена произведена корректно, для этого достаточно доказать, что последняя клетка нового обхода h_{n-1} и старого обхода $g_H^{n-1}(g_1(m_{k,n-1}))$ куба C_1 совпадают.

Заметим, что при любом выборе набора g_i, основное направление старого обхода куба C_1 – $g_H^{n-1}(x_1)$. Следовательно, его конечная клетка задаётся направлениями $g_H^{n-1}(-x_1)$, $g_H^{n-1}(x_2)$, ..., $g_H^{n-1}(x_k)$ (начальная задаётся $g_H^{n-1}(x_1)$, $g_H^{n-1}(x_2)$, ..., $g_H^{n-1}(x_k)$). По предположению индукции аналогичными направлениями задаётся последняя клетка обхода h_{n-1}.

Осталось показать, что конечная клетка h_n это $g_H^n(-x_1)$, $g_H^n(x_2)$, ..., $g_H^n(x_k)$. Конечная клетка обхода – это конечная клетка пути $g_H^{n-1}(m_{k,n-1})$. Конечная клетка $g_H^{n-1}(m_{k,n-1})$ такая же как конечная клетка $g_H^{n-1}(m_k)$, а для неё этот факт очевиден доказательства для $n = 1$. ■

Теперь на основе этой последовательности построим обход $\mathbb{Z}^k$. Начнём обход из нулевой точки и будем идти по следующему бесконечному пути: $h_1, h_2 \setminus h_1, h_3 \setminus h_2$, ..., $h_n \setminus h_{h-1}$ ... (где $h_i \setminus h_{i-1}$ обход h_i без первой части которая совпадает с h_{i-1}). Фактически это «предел» последовательности $\{h_n\}_{n=1}^{\infty}$, поэтому будем обозначать его h_∞. Части h_∞ соответствующие h_i-ым будем называть этапами обхода пространства.

Лемма 11. *Бесконечный путь h_∞ не имеет самопересечений.*

Доказательство. Действительно, пусть произошло самопересечение, тогда оно произошло на i-ом этапе построения h_∞, при этом известно, что h_i обходит соответствующий ему куб без самопересечений. Получаем противоречие. ■

Теперь перейдём к описанию того, так h_∞ заполняет пространство. Пусть K_n – куб, который обходит h_∞ по завершении n-ого этапа.

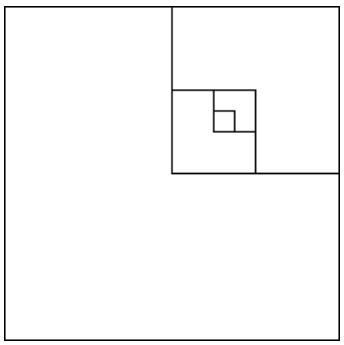

Рис. 3: Растяжения кубов в случае $k = 2$. Направление x_1 – влево, направление x_2 – вверх

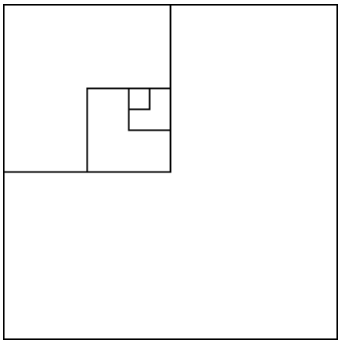

Рис. 4: Растяжение кубов в случае $k = 3$, проекция на направления x_2, x_3. Направление x_2 – влево, направление x_3 – вверх

Лемма 12. *Куб K_{n+1} можно получить растяжением куба K_n по направлениям $g_H^n(x_1)$, $g_H^n(x_2)$, ..., $g_H^n(x_k)$.*

Доказательство. Заметим, что K_n это первый маленький куб при обходе K_{n+1} с помощью $g_H^n(m_{k,n})$, а значит он соответствует вершине с направлениями рёбер $g_H^n(x_1)$, $g_H^n(x_2)$, ..., $g_H^n(x_k)$. Поэтому если растянуть K_n по этим направлениям получится K_{n+1}. ■

Таким образом часть пространства, которую обходит h_∞, растёт с помощью растяжений кубов в различные стороны в 2 раза. Так g_H имеет порядок $2(k-1)$ направления растяжений будут повторятся с такой же частотой.

Рассмотрим, какие габариты будут иметь кубы K_n. Для этого будем исследовать их проекции на каждую из k пар направлений. На первом этапе все проекции будут отрезками $[0, 1]$. Изменение этих отрезков будет значительно отличается для направлений $x_1, -x_1$ и всех остальных. Будем обозначать за l_n и r_n координаты левого и правого концов отрезков на n-ом этапе, а за a_n минимум их модуля.

Лемма 13. *При проекции K_n на направление $-x_1, x_1$ выполнено $a_n \geq \frac{2}{3}(2^n - 1)$.*

Доказательство. В силу выбора g_H, растяжения в направлениях x_1 и $-x_1$ будут чередоваться, причём переходе от $2q-1$-ому к $2q$-ому этапу будет растяжение по -1, а при переходе от $2q$-ому к $2q+1$-ому этапу по 1. Соответственно в первом случае из координаты левого отрезка надо вычесть 2^{2q-1}, во втором к координате правого прибавить 2^{2q}. Итого:

$$l_{2q-1} = l_{2q} = -\sum_{i=1}^{q} 2^{2i-1} = -\frac{2}{3}(4^q - 1); r_{2q} = r_{2q+1} = 1 + \sum_{i=1}^{q} 2^{2i} = 1 + \frac{4}{3}(4^q - 1)$$

Теперь легко проверить, что и для чётных, и для нечётных n, $|l_n|$ и r_n не меньше чем $\frac{2}{3}(2^n - 1)$. ■

Лемма 14. *При проекции K_n на направление $-x_i, x_i$ $(i > 1)$ выполнено $a_n \geq \frac{(2^k-2)(2^{n-2(k-1)}-1)}{2^{2(k-1)}-1}$.*

Доказательство. Заметим, что за $2(k-1)$ подряд идущих растяжений по $k-1$ растяжению будет в направлениях x_i и $-x_i$. Рассорим этап n следующие за ним $2(k-1)$ этап. Так как по каждому направлению было $k-1$ растяжений, концы отрезка в каждую сторону сдвинулись не меньше чем на $\sum_{i=1}^{k-1} 2^{n+i-1}$ (эта величина, когда все растяжения были в начале). Тогда $a_{2(k-1)q+1}$ можно оценить следующим способом.

$$a_{2(k-1)q+1} \geq \sum_{i=1}^{q}\sum_{j=1}^{k-1} 2^{2(k-1)(i-1)+j} = \sum_{i=1}^{q} 2^{2(k-1)(i-1)}(\sum_{j=1}^{k-1} 2^j) =$$
$$= (2^k-2)\sum_{i=1}^{q} 2^{2(k-1)(i-1)} = \frac{(2^k-2)(2^{2(k-1)q}-1)}{2^{2(k-1)}-1}$$

Тогда для произвольного a_n можно записать, что $a_n \geq a_{2(k-1)q+1}$, где $2(k-1)q+1$ наибольшее число такого вида, меньшее n. Тогда:

$$a_n \geq a_{2(k-1)q+1} \geq \frac{(2^k-2)(2^{2(k-1)q}-1)}{2^{2(k-1)}-1} \geq \frac{(2^k-2)(2^{n-2(k-1)}-1)}{2^{2(k-1)}-1}$$

■

Теорема 4. *Путь h_∞ посетит каждую клетку d, причём если p – максимальный модуль координат клетки, то количество ходов $T(d)$, которые мы прошли до неё, будет $O(p^k)$.*

Доказательство. Так как a_n для всех направлений стремятся к бесконечности, на каком-то этапе все они станут больше p, следовательно наша клетка, будет лежать внутри куба, обойденного к тому времени.

Пусть $d \in K_n$, но не $d \in K_{n-1}$, то есть мы посетим клетку d на n-ом этапе. Тогда существует пара направлений $-x_i, x_i$, для которых $a_{n-1} < p$. Тогда из предыдущих двух лемм следует, что $\log_2 p > n+c$, где c какая-то константа. Заметим, что $T(d) \leq 2^{kn}$ (число клеток в кубе K_n). Тогда,

$$T(d) \leq 2^{kn} < 2^{(log_2 p - c)k} = 2^{ck}p^k$$

Таким образом получаем требуемое. ■

6 Определение направления хода по его номеру

Теперь когда, в нашем распоряжении имеется обход $\mathbb{Z}^k$, нам нужен инструмент для построения его локально. Например, было бы неплохо уметь определять направление хода по его номеру. Для этого построим следующую конструкцию: конечный автомат, которому на вход подаётся последовательность цифр 2^k-ичного представления номера хода, начиная с самого младшего разряда, на выходе получается состояние соответствующие направлению движения.

6.1 Описание автомата в общем случае

Построим конечный автомат A_k, с $4k^2-2k-2$ состояниями. Среди них есть по $2k-2$ состояния для каждого из $2k$ направлений движения (будем их обозначать (d,i), где d - направление, i -

какой-то остаток при делении на $2k-2$) и $2k-2$ состояний для различных степеней g_H (порядок элемента g_H в G_k равен $2k-2$). Переходы в автомате устроены следующим образом. Для каждого состояния есть 2^k переходов соответствующих остаткам при делении на 2^k. У любого состояния (d,i) из первого множества переходы ведут в состояние $(g_H^i g_{a+1} g_H^{2k-1-i}(d), i+1)$, где a остаток при делении на 2^k (здесь и далее второй параметр состояния берётся по модулю $2k-2$). У любого состояния из второго множества, соответствующего g_H^i, если остаток a равняется 0, переход будет в g_H^{i+1}, иначе переход будет в состояние $(g_H^i(m_k[a]), i+1)$.

Докажем основное свойство этого автомата.

Лемма 15. *Пусть на вход автомату A_k поданы остатки при делении на 2^k a_1, a_2, ... ,a_n, такие, что $\overline{a_n a_{n-1} ... a_2 a_1} = p$ как 2^k-ичное число (первые цифры могут быть нулями). Тогда, в конце автомат окажется либо с состоянии g_H^n, если все остатки 0, либо в состоянии, соответствующем направлению $(g_H^{n-1}(m_{k,n}[p]), n)$.*

Доказательство. Будем доказывать по индукции. Для $n = 1$ утверждение очевидно: если $p = 0$, то переход будет в состояние, соответствующее g_H^1. Если $p \neq 0$, то переход будет в состояние $(g_H^0(m_k[p]), 1) = (m_{k,1}[p], 1)$.

Пусть утверждение верно для $n = i$, докажем для $n = i+1$.

Пусть первые i остатков были нули, тогда после принятия этих остатков мы окажемся в состоянии g_H^i. Если $a_{i+1} = 0$, то как и требуется мы перейдем в состояние g_H^{i+1} просто по построению автомата. Если $a_{i+1} \neq 0$, то мы перейдем в состоянии $(g_H^i(m[a_{i+1}]), i+1)$. Рассмотрим направление движения $g_H^i(m_{k,i+1}[p])$. Заметим, что p делится на 2^i, следовательно ход $m_{k,i+1}[p]$ соответствует переходу между «маленькими» кубами $C_{a_{i+1}}$ и $C_{a_{i+1}+1}$ при обходе $m_{k,i+1}$. Следовательно $m_{k,i+1}[p] = m_k[a_{i+1}]$. Отсюда следует, что $g_H^i(m_{k,i+1}[p]) = g_H^i(m[a_{i+1}])$, следовательно как и требовалось мы перейдем в состояние $(g_H^i(m_{k,i+1}[p]), i+1)$.

Теперь пусть среди первых i цифр были не нули. Тогда до перехода по a_{i+1} мы находимся в состоянии $(g_H^{i-1}(m_{k,i}[p_i]), i)$, где p_i - число, составленное из последних i цифр p. Посмотрим куда нас приведёт переход по a_{i+1}. Направление изменится на следующее:

$$g_H^i g_{a_{i+1}} g_H^{2k-1-i} g_H^{i-1}(m_{k,i}[p_i]) = g_H^i g_{a_{i+1}}(m_{k,i}[p_i])$$

Также заметим, что в силу построения $m_{k,i+1}$ имеет место $m_{k,i+1}[p] = g_{a_{i+1}}(m_{k,i})[p_i]$. Действительно, $m_{k,i+1}[p]$ находится в «маленьком» кубе $C_{a_{i+1}+1}$ (обходится с помощью $g_{a_{i+1}}(m_{k,i})$), причем в этом кубе это p_i-ый ход. Значит, состояние, в которое мы перейдем будет $(g_H^i(m_{k,i+1}[p]), i+1)$, что и требовалось доказать.

■

Теорема 5. *Пусть на вход автомату A_k поданы остатки при делении на 2^k a_1, a_2, ... ,a_n, такие, что $\overline{a_n a_{n-1} ... a_2 a_1}$ 2^k-ичная запись числа p без незначащих нулей. Тогда в конце автомат окажется в состоянии (d,n), где $d = h_\infty[p]$.*

Доказательство. Заметим, что a_n - не равен 0, поэтому согласно предыдущей лемме мы окажемся в состоянии $(g_H^{n-1}(m_{k,n}[p]), n)$. В силу построения h_n, ходы h_n и $g_H^{n-1}(m_{k,n})$ с номерами от 2^{n-1} до $2^n - 1$ одинаковы. Заметим, что $2^{n-1} \geq p \geq 2^n - 1$, поэтому $h_\infty[p] = h_n[p] = g_H^{n-1}(m_{k,n}[p])$, что нам и требуется. ■

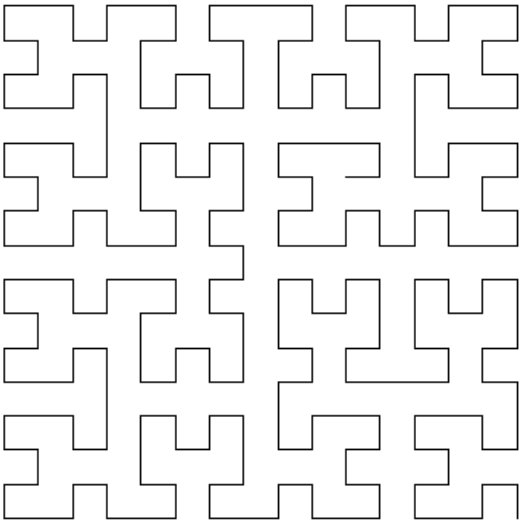

Рис. 5: путь h_4. Направление x_1 – влево, направление x_2 – вверх

Таким образом, после принятия цифр числа p автомат A_k окажется в одном из состояний соответствующих направлению p-ого хода обхода h_∞, то есть в наших руках имеется инструмент для «локального» построения обхода $\mathbb{Z}^k$.

6.2 Частный случай: $k = 2$

Для большей наглядности в этой секции представим обход $\mathbb{Z}^2$ и автомат, который его строит.

На картинке представлен обход h_∞, точнее его часть h_4. Для $k = 2$ g_H переводит x_1 в $-x_1$, x_2 в $-x_2$, поэтому кубы (точнее квадраты) K_i растут по одной диагонали в разные стороны. Подобное поведение характерно, только для случая $k = 2$. В больших размерностях рост кубов более сложно устроен.

Такая простота позволяет немного упростить A_2, без потери его целевого свойства. Для каждого направления достаточно использовать по одному состоянию. Новый автомат можно увидеть на рисунке. Подобное упрощение сложно сделать в больших размерностях, т.к. g_H и g_i не всегда коммутируют в группе G_k. Именно поэтому пришлось для каждого направления ввести по $2(k-1)$ состояний.

7 Общий алгоритм полиномиального обхода с 4-мя камнями

Наконец в наших руках имеются все инструменты для описания итогового полиномиального алгоритма обхода $\mathbb{Z}^k$ роботом, использующим 4 камня. Общая схема обхода такова. Робот на каждой итерации привязан к какой-то текущей клетке. Он держит один камень в этой клетке и ещё один на расстоянии n от неё в направлении x_1. С помощью алгоритма получения битов, он получает биты числа n, агрегирует в блоки по k, и даёт на вход автомату A_k. Когда биты заканчиваются, робот определяет какому направлению соответствует текущее состояние A_k. По этому направлению он сдвигает дальний камень и текущую клетку вместе с камнями на ней и увеличивает расстояние между камнями на 1. Потом переходит на следующую итерацию. Заметим, что в такой схеме

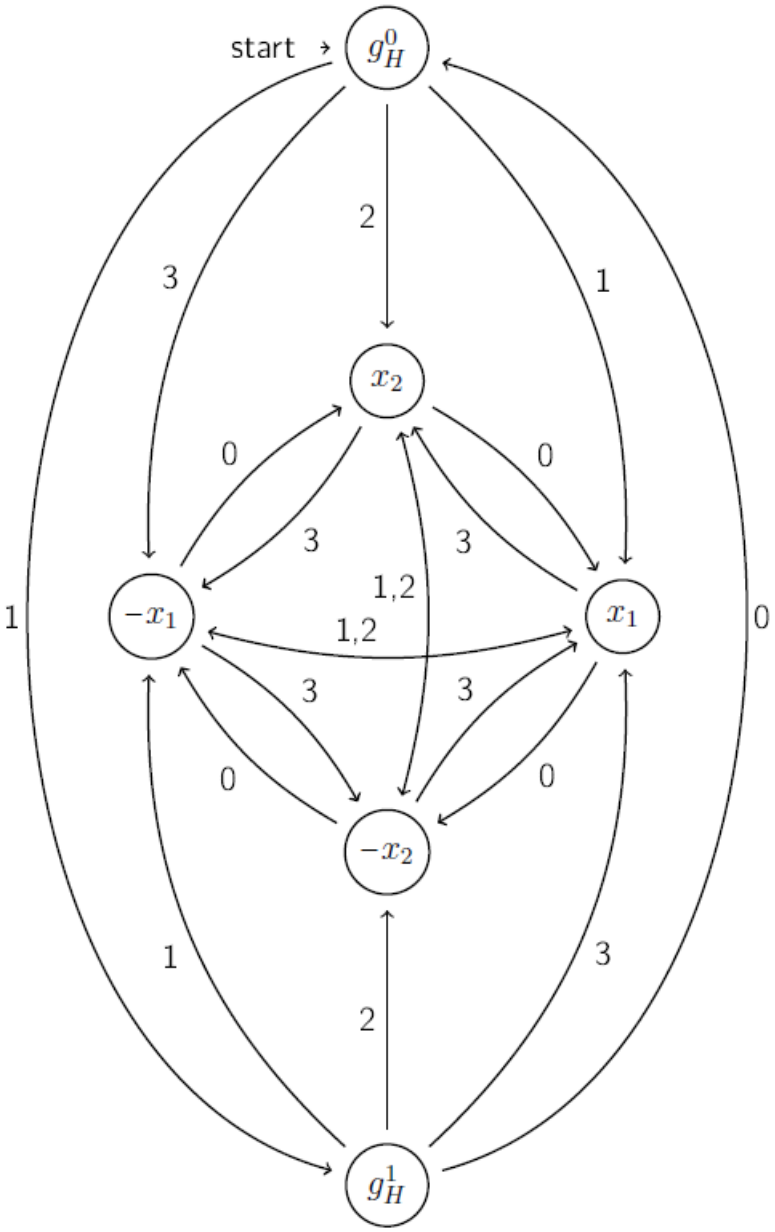

Рис. 6: Упрощённый автомат A_2

построения текущие клетки будут изменятся в соответствии с обходом h_∞.

Количество каменей которой будет использовать робот – 4: два, которые мы описали – основные, и два вспомогательных для получения битов числа. При этом, стоит заметить, что подобный алгоритм может быть реализовать на конечной памяти. Во-первых в каждый момент мы храним не больше k битов числа n, во-вторых размер автомата A_k зависит только от k, и его можно сэмулировать на конечной памяти. Теперь опишем более подробно алгоритм.

Начальная конфигурация: робот и 4 камня A, B, C, D в точке 0

1: перенести камень B в направлении x_1 и вернуться
2: **цикл** // $n = 1, 2..., \infty$, a – *текущая клетка*
3: инициализировать автомат A_k
4: идти в направлении x_1 с камнем D до камня B // *Начинаем выполнять считывание битов*
5: вернуться в клетку a
6: **цикл** // *i-ая итерация, C в клетке a, D в клетке на расстоянии $\lfloor n/2^{i-1} \rfloor$ от a*
7: выполнить деление для камней C и D
8: записать остаток при делении на 2 (бит) в память
9: **если** i делится на k **то**
10: агрегировать последние k полученных битов в блок (остаток при делении на 2^k) и передать автомату A^k
11: **если** C и D находятся в клетке a **то**
12: **выход из цикла**
13: **иначе**
14: перенести C в клетку a
15: **если** остались необработанные биты **то**
16: дописать нулевые биты в блок и передать автомату A^k
17: получить из автомата A_k направление d
18: дойти до B, сдвинуть в направлении d и в направлении x_1 // *для увеличения n*
19: вернуться в a и сдвинуть все камни в направлении d

Посчитаем время, которое роботу понадобиться, чтобы посетить клетку a, у которой p – модуль максимальной координаты. Для этого будем посчитаем, когда клетка a станет текущей, для какой-то итерации алгоритма (в этом случае робот её точно посетит).

Теорема 6. *Пусть a клетка, у которой p – модуль максимальной координаты, тогда для того, чтобы она стала текущей клеткой алгоритма, потребуется $T(p) = O(p^{3k})$ ходов алгоритма.*

Доказательство. Пусть a – текущая клетка n-ой итерации. Будем считать, что на i-ую итерацию тратится не больше ci^2 ходов, где c – константа (действительно, самое трудоёмкое действие это получение битов числа i, а оно квадратично по времени). Тогда всего количество ходов алгоритма за первые n итераций не больше чем столько:

$$\sum_{i=1}^{n} ci^2 = \frac{c}{6} n(n+1)(2n+1) \le c' n^3$$

где c' – другая константа.

Из теоремы о асимптотике пути h_∞ известно, что $n(p) = O(p^k)$. Тогда получается следующая ситуация: $T(p) \le c'n(p)^3 = O(p^{3k})$, что нам и требовалось. ■

В итоге получаем, что в результате работы алгоритма каждая клетка $\mathbb{Z}^k$ будет посещена, при этом в время, которое для этого потребуется будет полиномиально от координат точки.

8 Нижние оценки для оптимального числа камней

Наравне с задачей нахождения алгоритма, использующего минимальное количество каменей, всегда существует задача доказательства того, что меньшим количеством камней нельзя обойтись. Для задачи простого обхода $\mathbb{Z}^k$ нижние и верхние оценки количества камней совпадают. Однако если мы будем рассматривать задачу полиномиального обхода, ситуация не такая радужная. Для $k > 3$ нижние и верхние оценки имеют зазор в один камень: нижняя оценка говорит, что 2-х камней недостаточно, а верхняя, что существует обход с 4-камнями. При этом сокращение этого расстояния представляется довольно трудоёмкой задачей, поскольку с одной стороны доказательство, что 3-мя камнями нельзя обойтись должно пользоваться полиномиальностью обхода (поскольку экспоненциальные обходы $\mathbb{Z}^k$ существуют). С другой стороны придумать полиномиальный обход с 3-мя камнями тоже не просто.

В этот секции будут приведены доказательства существующих нижних оценок. В частности, будет показано, что одного камня недостаточно для обхода любого $\mathbb{Z}^k$ и, то что 2-ух камней недостаточно для обхода $\mathbb{Z}^k$ $k > 1$.

Теорема 7. *Одного камня недостаточно для обхода $\mathbb{Z}^k$.*

Доказательство.

Примерная схема доказательства такова. Пусть количество различных состояний робота равно s. Допустим, что робот смог придти в клетку d на расстоянии много большем s от камня (расстояние будем считать в L_1 метрике). Рассмотрим, какие состояния принимал робот от последнего касания до прихода в эту клетку. Эти состояния выглядели так: сначала $r \le s$ каких-то состояний, потом состояния начинают повторятся с периодичностью $T \le s$. Тогда относительное расположение робота в момент времени $r + iT + c, c < T$ можно выразить вектором $\overline{a(c)} + i\bar{b}$ (где длина $\overline{a(c)}$ не больше $2s$ для любого $c < T$, и длина $\bar{b}$ не больше s и не нулевая). Тогда начиная с какого-то i это вектор никогда не станет равным 0, следовательно робот не вернётся в камню, а так же на никогда не посетит клетки $-i\bar{b}$ для какого-то большого i. Следовательно, робот всегда вынужден находится рядом с камнем.

Пусть F – множество всех возможных относительных конфигураций робот-камень. Как мы только что показали, оно должно быть конечно. Пусть f_n – конфигурация в момент n, q_n – состояние робота в момент n. Заметим, что начиная с какого-то момента пары (f_n, q_n) зациклятся (так как множество $F \times Q$ конечно и каждая пара однозначно определяет следующую). Пусть за один цикл камень сдвинется на вектор $\overline{a}$, тогда если $\overline{a} = 0$, то очевидно, что камень на сдвинется далеко от начального положения, а следовательно робот не сможет посетить все клетки. Если $\overline{a} \neq 0$, то очевидно, что робот не сможет посетить клетки $-i\overline{a}$ для достаточно больших i.

■

Теорема 8. *Двух камней недостаточно для обхода $\mathbb{Z}^k, k > 1$.*

Доказательство. Примерная схема доказательства такова. Пусть существует такая константа c, что существует бесконечно много моментов времени когда камни находятся на расстоянии меньше c. Тогда какая-то взаимная конфигурация камней, робота и его состояния случается хотя бы два раза. Следовательно подобные конфигурации зацикливаются. Аналогично доказательству для одного камня, можно показать, что за такой цикл оба камня сдвигаются на какой-то вектор, а значит робот не обойдёт далёкие клетки пространства, в противоположном этому вектору направлении.

Пусть для любой константы c существует такой момент, начиная с которого камни всегда на расстоянии большем c. Будем рассматривать поведение робота начиная с такого момента для константы c много большей s (s количество состояний робота). Поведение робота можно описать следующим образом: робот как-то взаимодействует с первым камнем, потом переходит ко второму камню, взаимодействует с ним, переходит обратно к первому и так далее. Назовём один такой цикл итерацией. Пусть робот последний раз касается первого камня в состоянии q_1 потом идёт к камню 2 и первый раз касается его в состоянии q_2. Заметим, что после того как робот последний раз коснулся первого камня, не более чем через s ходов состояния начнут изменяться циклически, причём длина цикла не больше s. Тогда вектор между камнями можно записать так:

$$\overline{v} = \overline{r(q_1)} + i\overline{a(q_1)} + \overline{p(q_2)}$$

где $\overline{r(q_1)}$ вектор, который проходится до начала цикла, $\overline{a(q_1)}$ вектор проходимый за цикл, $\overline{p(q_2)}$ вектор отвечающий за фазу при приходе во второй камень. Длины $\overline{r(q_1)}$, $\overline{p(q_2)}$ и $\overline{a(q_1)}$ не больше s, $i > 0$ т.к. длина $\overline{v}$ много больше s. Пусть $f(\overline{x})$ – это вектор, где все координаты координаты $\overline{x}$ взяты по модулю $s!$. Заметим, что по q_1 и $f(\overline{v})$ можно однозначно определить состояние q_2, поскольку координаты вектора $\overline{a(q_1)}$ не превосходят s по модулю. Таким образом по q_1 и $f(\overline{v})$ можно определить q_2, по q_2 можно однозначно определить как робот будет взаимодействовать со вторым камнем, значит можно однозначно определить состояние робота q_3 при последнем касании со вторым камней, и вектор $f(\overline{u})$, где $\overline{u}$ – вектор между камнями в этот момент. По этому вектору и q_3 аналогично можно однозначно определить q_4 – состояние при первом касании первого камня. По вектору $f(\overline{u})$ и состоянию q_4 однозначно определяются новые состояние q_1 и вектор $f(\overline{v})$. Таким образом по состоянию q_1 и вектору $f(\overline{v})$ какой-либо итерации, можно определить эти же состояние и вектор для следующей итерации. Так как состояний и подобных векторов конечное число, в какой-то момент они зациклятся. Это будет означать, что зациклятся состояния q_2 и q_4, а следовательно зациклятся сдвиги обоих камней. Пусть за такой цикл один камень сдвигается на вектор $\overline{g}$ другой на вектор $\overline{h}$. Тогда если векторы коллинеарны, то пусть $\overline{w}$ перпендикулярен им, тогда робот никогда не посетит клетку с координатами $i\overline{w}$, где i достаточно большое. Иначе робот не посетит клетку с координатами $-i(\overline{g} + \overline{h})$, где i достаточно большое.

■

Таким образом мы получаем требуемые оценки. Заметим, что в обоих доказательствах мы действовали следующим образом, брали некоторую характеристику состояния процесса, и показывали, что эта характеристика зацикливается. Например, можно заметить, что в случае двух камней существует такое число, что координаты камней по этому модулю цикличны. Если мы посмотрим на систему с тремя камнями, подобную цикличность вряд ли можно будет обнаружить, так как уже в простых алгоритма (например обход плоскости) подобной цикличности не наблюдается. Тем не

менее у последовательности для конкретной характеристики состояния можно смотреть не только на её цикличность или ацикличность, но и на другие свойства, например информационные. Возможно поиск и рассмотрение других свойств подобных последовательностей является ключом к доказательству невозможности обхода за полиномиальное время с тремя камнями.

9 Заключение

По результатам работы нами получено существенное уменьшение количества требуемых камней для полиномиального обхода по $\mathbb{Z}^k$ сравнению в базовыми алгоритмами: 4 против k. Также стоит отметить, что представленный нами алгоритм не сильно медленнее алгоритмов использующих k камней: $O(x^{3k})$ против $O(x^k)$. В рассматриваемой задаче остаётся некоторый зазор между верхней и нижней оценкой в один камень. Разрешение данного вопроса пока видится сложной задачей, так либо требуется привести по-видимому довольно нетривиальный алгоритм работающий с тремя камнями, либо доказательство невозможности обхода с 3 камнями, возможно требующее довольно серьёзной техники.

Список литературы

[1] Budach L. *Automata and labyrinths* Mathematische Nachrichten, 1978

[2] `http://www.turgor.ru/lktg/2004/avtom.ru/ru.ps`

[3] `http://en.wikipedia.org/wiki/Counter_machine`

[4] F. Gray. *Pulse code communication* U.S. Patent 2,632,058, 1953

[5] D. Hilbert *Uber die stetige Abbildung einer Linie auf ein Flachenstuck* Mathematische Annalen, 1891

[6] G.Peano *Sur une courbe, qui remplit toute une aire plane* Mathematische Annalen, 1890

[7] A. R. Butz *Convergence with Hilbert's space-filling curve* Journal of Computer and System Sciences, 1969.