УДК 004.021.53

Бритиков К.И., студент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»
Научный руководитель: Гапанюк Ю.Е., к.т.н., доцент,
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

АЛГОРИТМЫ СБОРКИ КУБИКА РУБИКА

Ключевые слова: *кубик Рубика, алгоритм, теория групп, Python, скорость сборки, число Бога, оптимальный алгоритм, теория графов.*

Аннотация: *В данной статье будут рассмотрены различные алгоритмы для сборки кубика Рубика. Они реализованы на Python 3.6 со стандартными библиотеками. Данные алгоритмы основаны на теории групп, теории графов и комбинаторике. Все сборки кубика произведены на одном компьютере. В данной статье рассматриваются четыре алгоритма для сборки.*

Key words: *Rubiks Cube, algorithm, group theory, Python, assembly speed, number of God, optimal algorithm , graph theory.*

Annotation: *Different algorithms of Rubiks cube assembly will be considered here. This algorithms are realized by Python 3.6 with standard libraries. Given algorithms are based on group theory, graph theory and combinatorics. All cube's assemblies done with one computer. In this article four algorithms of assembly are considered.*

Кубик Рубика – это головоломка, созданная венгерским архитектором Эрнё Рубиком в 1974 году. Для создания любого алгоритма необходимо понимать саму структуру данной головоломки. Кубик Рубика состоит из 21 детали: 1 центральная (6 центральных квадратов), 8 угловых(3 квадрата в углах куба), 12 боковых(2 квадрата между центрами). Кубик покрашен в 6 различных цветов, по количеству граней. Суть головоломки состоит в том, чтобы собрать (на 1 грани 1 цвет) кубик.

Рис.1 Эрнё Рубик.

Алгоритмы, которые существуют в настоящее время для сборки Кубика, основаны на теории графов, теории групп, теории вычислимости и комбинаторике. В этой статье для обозначения последовательности поворотов граней кубика Рубика 3×3×3 используется «нотация Сингмастера», разработанная Дэвидом Сингмастером и опубликованная им в 1981 году. Сборка куба происходит в метрике QTM, грани поворачиваются только на 90 градусов.

Буквы L , R , F , B , U , D обозначают поворот на 90° по часовой стрелке левой (left), правой (right), передней (front), задней (back), верхней (up) и нижней (down) граней соответственно. Знак « ’ » обозначает поворот против часовой стрелки.

Как уже говорилось ранее, алгоритмы сборки основываются на теории групп. Группа кубика Рубика — подгруппа симметрической группы S48, элементы которой соответствуют преобразованиям кубика Рубика. Под преобразованием подразумевается эффект поворота любой из граней или последовательности поворотов граней.

Группа кубика Рубика G определяется как подгруппа S 48, порождаемая поворотами шести граней на 90°:

$$G = \langle L , F , R , B , U , D \rangle$$

Порядок группы G равен:

$$| G | = \frac{8! \cdot 12! \cdot 3^8 \cdot 2^{12}}{3 \cdot 2 \cdot 2} = 43\ 252\ 003\ 274\ 489\ 856\ 000 = 2^{27} \cdot 3^{14} \cdot 5^3 \cdot 7^2 \cdot 11.$$

По сути – это количество всех возможных комбинаций цветов кубика.

G не является абелевой группой, так как, например, F R ≠ R F. Другими словами, не все пары элементов коммутируют.

В июле 1981 года Jesper C. Gerved и Torben Maack Bisgaard доказали, что группа кубика Рубика содержит элементы 73 различных порядков от 1 до 1260, и нашли число элементов каждого возможного порядка

Группа кубика Рубика содержит циклические подгруппы порядков

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 18, 20, 21, 22, 24, 28, 30, 33, 35, 36, 40, 42, 44, 45, 48, 55, 56, 60, 63, 66, 70, 72, 77, 80, 84, 90, 99, 105, 110, 112, 120, 126, 132, 140, 144, 154, 165, 168, 180, 198, 210, 231, 240, 252, 280, 315, 330, 336, 360, 420, 462, 495, 504, 630, 720, 840, 990, 1260.

Одним из самых популярных современных методов сборки кубика Рубика является метод FTM. Этот метод состоит из 4 шагов:

1. Собрать на верхней грани крест
2. Собрать верхнюю грань
3. Собрать на боковой грани 2 верхних слоя
4. Собрать нижний слой

На всех этих четырех шагах существуют определенные ограничения при поворотах граней куба. Этот метод так же описывается на основе теории групп, он менее эффективен чем другие методы, но очень прост для освоения человеком. Именно этот метод используют люди, собирающие кубик на скорость.

Не смотря на простоту для человека, этот метод трудоемок для написания кода не менее, чем другие, более эффективные методы. Мы рассмотрим две реализации данного метода.

Первая, и самая простая реализация – это интуитивный метод. Существует определенная последовательность действий и закономерность для каждого из этапов этого метода.

Для первого этапа мы ищем боковые квадраты верхней грани на остальных гранях. В связи с закономерностями сборки кубика они могут присутствовать только на боковых квадратах граней. Обнаруживая их по одному, мы ставим их на место, так чтобы не сдвинуть квадраты, которые уже присутствуют на верхней грани. Для таких действий есть определенная последовательность поворотов: F U' R U. Далее начинаем этап с начала для поиска еще одного квадрата.

После этого, при помощи определенных комбинаций мы делаем так, чтобы квадраты на боковых элементах совпадали с квадратами боковых граней.

Для выполнения второго этапа сперва необходимо установить местоположение наших угловых элементов. После того как мы найдем угловой элемент, существует несколько вариантов развития событий:

1. Угловой элемент на месте, но он повернут неверно. Тогда проворачиваем его при помощи следующей комбинации (угол смотрит на нас и находится справа): R' D' R D
2. Угловой элемент на нижней грани. Проворачиваем нижнюю грань, до тех пор, пока цвета углового элемента не будут совпадать с цветами боковых соседних. После этого повторяем комбинацию из 1-го варианта.
3. Угловой элемент на верхней грани, но не на своем месте, тогда размещаем кубик, как в первом варианте, после этого выполняем комбинацию R' D' RD. И переходим ко второму варианту.

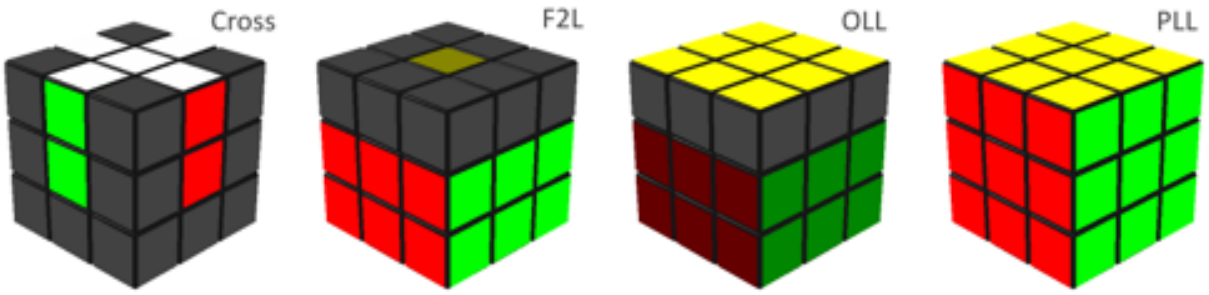

Рис.2 Положения кубика во время сборки методом FTM

После этого переходим к третьему этапу. Сперва перевернем куб, так чтобы нижняя грань стала сверху. Начинаем движение по боковым элементам. Ищем боковой элемент между передней и правой гранью. Существует 3 варианта местоположения бокового элемента:

1. Боковой элемент на месте, но неправильно повернут. Поворачиваем куб, так чтобы один из квадратов элемента был справа, а другой спереди. После этого поворачиваем элемент при помощи комбинации до того, как куб будет сложен: U R U' R'U' F' U F
2. Боковой элемент находится между другими гранями. Поворачиваем куб относительно элемента так как в первом. Производим комбинацию U R U' R'U' F' U F, после этого проворачиваем верхнюю грань, до тех пор , пока элемент не встанет над левой гранью. После этого делаем комбинацию из первого.
3. Боковой элемент сверху. Делаем 2-й

Четвертый этап состоит из нескольких подразделов:

1. Собрать сверху правильный крест. Сперва просто получим сверху крест. Делаем 1,2 или 3 раза(до креста) F R U R' U' F'. После этого делаем правильный крест. Вращаем верхний слой, чтобы какие-либо ДВА ребра совпали по цветам с центрами из среднего слоя. Может получиться одна из двух ситуаций. Если стоят неправильно два противолежащих угла, то делаем R U R' U

R U2 R'. Если два собранных ребра, стоят под углом, два остальных нужно поменять местами комбинацией, при этом кубик держим, чтобы угол смотрел от вас и вправо R U R' U R U2 R' U.

2. Расставить углы верхнего слоя по местам. Сперва надо несколько раз сделать комбинацию U R U' L' U R' U' L. После этого один из угловых элементов станет на место. Три несобранных угла нужно перемещать по часовой стрелке до его сборки. Собранным углом нужно повернуть к себе и справа, U R U' L' U R' U' L.

Куб собран. При помощи компьютерного алгоритма куб собирается за 170-220 шагов, при этом затрачивается от 0,6 до 1,2 секунды. Человек собирает этот куб за 70-150 ходов в нотации Сингмастера, потому что он также использует логические выводы по ходу сборки, в отличии от машинального подхода компьютера, который просто однообразно перебирает ходы.

Также FTM-сборку можно реализовать генетическим методом. Несмотря на более низкую эффективность этот метод представляет интерес как пример саморазвивающейся системы, которая отбрасывает наименее слабые генерации, и представляет пример эволюции.

Он движется по тем же шагам что и простой FTM. В данном случае шаг(обычный поворот грани) называется геном. Новые гены генерируются случайно, при этом прибавляясь к геному алгоритма сборки. Важным параметром мутации является максимальное возможное число генов, которые будут добавлены во время следующей мутации. Это важное число, которое регулирует, на сколько сложную модификацию Куб сможет сгенерировать за один раз. После мутации Куба считается его фитнесс, а затем и фитнесс всех остальных кубов. Наиболее эффективный геном на определенный момент выживает и передается как основа для дальнейших мутаций.

Основная особенность состоит в том, что максимальное количество генов (шагов) приходится увеличивать до 30 раз, из-за того, что последний этап является намного сложнее первых, и ради решения задачи приходится пренебрегать сложностью решения. Это необходимо, чтобы избегать генетических ям, и не прекращать поиск на некотором локальном максимуме.

В среднем этот алгоритм решает куб за 300 - 400 шагов, и за 2 – 4 минуты, что проигрывает даже обычному наивному методу, недаром Скиена советует использовать алгоритм имитации отжига вместо генетического алгоритма везде, где это возможно. Единственное что в нем привлекает – это симуляция эволюционного процесса.

Один из самых эффективных алгоритмов сборки, а также самым популярным машинным алгоритмом является алгоритм Коцембы. Он основывается на алгоритме М. Тистлетуэйта. Тистлеуэйт предложил ограничить сборку куба до четырех этапов, а Коцемба и вовсе до двух:

G_0 = ⟨ U , D , L , R , F , B ⟩

G_1 = ⟨ U , D , L^2 , R^2, F^2, B^2 ⟩

G_2 = { 1 }

Наглядное описание группы G_1 можно получить, если произвести следующую разметку:

- Все этикетки U и D пометить знаком «+».
- Этикетки F и B на рёберных элементах FR, FL, BL, BR пометить знаком «-».
- Все остальные этикетки оставить без меток.

Тогда все конфигурации группы G_1 будут иметь одну и ту же разметку (совпадающую с разметкой на собранном кубике).

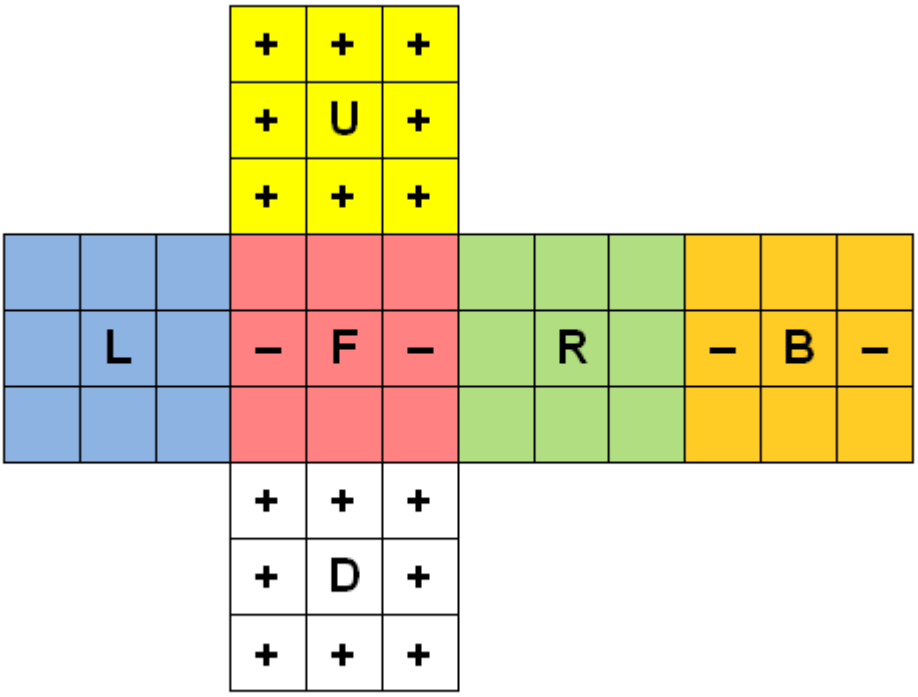

Рис.3 Разметка куба по методу Коцембы. Отмеченные квадраты будут на месте после первого этапа алгоритма.

Решение состоит из двух этапов. На первом этапе заданная конфигурация x $\in G_0$ переводится последовательностью ходов S_1 в некоторую конфигурацию X′ $\in$ G_1. Иными словами, задача первого этапа — восстановить разметку, соответствующую начальной конфигурации, игнорируя цвета этикеток. Эта сборка производится путем обхода графа в определенной последовательности.

Просто проанализировав обычный обход графа в глубину мы получим, что при таком лобовом подходе объем просматриваемых вариантов был бы слишком велик. Так, первый ход можно сделать 6X3=18 способами (любую из шести граней можно повернуть на один из трех углов — 180°, ±90°), на втором и каждом из следующих ходов число способов равно 15, так как нет смысла дважды подряд поворачивать одну грань. Таким образом, возникает «дерево вариантов», от каждой ветви которого отходят пять следующих вет-

вей. (На самом деле, начиная с определенного момента некоторые ветви будут срастаться, потому что разные цепочки ходов могут порождать одинаковые преобразования кубика.) Число цепочек ходов длины, не превосходящей n, равно сумме геометрической прогрессии 18(1+15+...+15n-1).

Между прочим, только при n=18 это число превысит общее число N состояний кубика, а значит, заведомо найдутся состояния, которые нельзя упорядочить меньше чем за 18 ходов.

В действительности, из-за сращивания ветвей дерева вариантов число 18 можно еще увеличить.

Для сокращения обхода Коцемба догадался использовать фильтры. Они хранят информацию о цепочках, например, из 7 ходов. Алгоритм порождает первые два хода, и проходит через фильтр на 7 ходов, если это состояние не отсеется, то включается фильтр на 6 ходов и т.д.

На втором этапе конфигурация $X' \in G_1$ переводится последовательностью ходов S_2 в начальную конфигурацию. При этом повороты боковых граней на ±90° не используются, благодаря чему разметка автоматически сохраняется.

Склейка последовательностей ходов S_1 и S_2 является субоптимальным решением к исходной конфигурации.

Для поиска решений на каждом из двух этапов применяется алгоритм IDA* с эвристическими функциями, основанными на стоимостях решения соответствующих подзадач.

Задача первого этапа сводится к поиску пути в пространстве с тремя координатами из разметки с координатами (x1, y1, z1) в разметку (0, 0, 0):

1. Ориентация x1 восьми угловых элементов (2187 значений)
2. Ориентация y1 двенадцати рёберных элементов (2048 значений)
3. Расстановка z1 четырёх рёберных элементов FR, FL, BL и BR (495 значений)

Рассмотрим три подзадачи поиска пути из разметки (x1, y1, z1) в разметки (x1’, y1’, 0), (x1’, 0, z1’), (0, y1’, z1’). Значение эвристической функции h1, используемой на первом этапе, равно максимальной из стоимостей решения этих подзадач. Стоимости решения подзадач предварительно вычисляются и хранятся в виде баз данных с шаблонами.

Задача второго этапа сводится к поиску пути в пространстве с тремя координатами из конфигурации (x2, y2, z2) в конфигурацию (0, 0, 0):

1. Перестановка x2 восьми угловых элементов (40320 значений)
2. Перестановка y2 восьми рёберных элементов верхней и нижней граней (40320 значений)
3. Перестановка z2 четырёх рёберных элементов среднего слоя (24 значения)

Рассмотрим три подзадачи поиска пути из конфигурации (x2, y2, z2) в конфигурации (x2’, y2’, 0), (x2’, 0, z2’), (0, y2’, z2’). Значение эвристической функции h2, используемой на втором этапе, равно максимальной из стоимостей решения этих подзадач.

В алгоритме применяются дополнительные оптимизации, направленные на увеличение быстродействия и уменьшение памяти, занимаемой таблицами.

Данный алгоритм был реализован на языке Python. Он является достаточно быстрым, и в то же время эффективным по количеству ходов. Он решает куб в среднем за 22-30 ходов, и в тоже время не более чем за 1,8 секунды.

Этот алгоритм несомненно является крайне эффективным, в особенности для специальных машин «Cube solver’ов» , так как он быстро дает краткое решение. Но этот алгоритм не подходит для поиска алгоритма бога(кратчайшего решения).

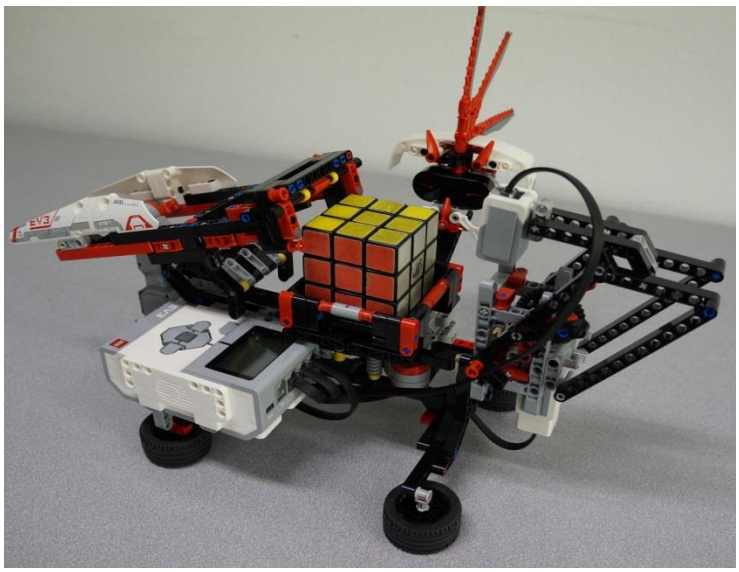

Рис.4 Cubesolver , созданный на базе конструктора LEGO Mindstorm.

Суть сборки состоит в том , чтобы собрать куб за минимальное количество времени и ходов, поэтому следует рассмотреть алгоритм для нахождения числа Бога для кубика.

Лучшим алгоритмом для этой цели является алгоритм Корфа. Алгоритм Коцембы позволяет быстро находить короткие, субоптимальные решения. Тем не менее, может потребоваться значительное время, чтобы доказать оптимальность найденного решения. Был необходим специализированный алгоритм поиска оптимальных решений.

В 1997 году Ричард Корф опубликовал алгоритм, позволявший оптимально решать произвольные конфигурации кубика Рубика. Десять выбранных случайным образом конфигураций были решены не более чем в 18 ходов FTM[55][56].

Совсем недавно, в 2010 году группа программистов из Google, совместно с Гербертом Коцембой , доказали что любой куб можно решить не более чем за 20 ходов. Они использовали модифицированный алгоритм Корфа, и исследовали 55 888 296 уникальных множеств состояний куба. На это потребовалось 35 лет компьютерного времени Google.

Собственно алгоритм Корфа работает следующим образом. В первую очередь определяются подзадачи, достаточно простые для того, чтобы осуществить полный перебор. Были использованы следующие три подзадачи

1. Состояние головоломки определяется только восемью угловыми кубиками, положения и ориентации рёбер игнорируются.
2. Состояние головоломки определяется шестью из 12 рёберных кубиков, другие кубики игнорируются.
3. Состояние головоломки определяется другими шестью рёберными кубиками.

Количество ходов, необходимое для решения каждой из этих подзадач, является нижней оценкой количества ходов, необходимого для полного решения. Произвольно заданная конфигурация кубика Рубика решается с помощью алгоритма IDA*, использующего эту оценку в качестве эвристики. Стоимости решения подзадач хранятся в виде баз данных с шаблонами.

Хотя этот алгоритм будет всегда находить оптимальные решения, он не позволяет напрямую определить, как много ходов может потребоваться в худшем случае.

Этот алгоритм самый медленный. Исполнение программы на моем компьютере занимает от 2 минут и более, так как он исследует буквально все ходы данного решения, и находит самое оптимальное.

В результате мы получили, что наиболее эффективным, с точки зрения машины, вариантом сборки кубика, является сборка методом Коцембы. Она одновременно является быстрой и эффективной, выдавая погрешность лишь в пределе 5 ходов. Второй по эффективности – это метод Корфа, и на последнем месте, ожидаемо находиться наивный метод. Таким образом математическими методами можно ускорить решения как этой, так и других подобных задач.

Список литературы:

1) Объяснение алгоритма Коцембы от его создателя(англ.) // kociemba.org пуб. Сайт Герберта Коцембы. URL:kociemba.org/math/imptwophase.html
2) *В. Дубровский, А. Калинин* Новости кубологии (рус.) // Квант. — 1992. — № 11. — С. 52 — 56.
3) *David Joyner*. Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys. — Second edition. // JHU Press, 2008. — 328 p. — ISBN 0-801-89726-2, 978-0-801-89726-9.
4) Сборка Кубика Рубика генетическим алгоритмом //harbhabr.ru: ежедн. интернет издание. URL: https://habrahabr.ru/post/260965/