Титульный лист

**Содержание**

1. Введение …………………………………………………………………3
2. Обзор литературы …………………………………………………….....5
   1. Язык вращений и метрика хода кубика Рубика …………………...5
   2. Число бога …………………………………………………………...5
   3. Алгоритм Тистлетуэйта и его модификации ………………….......6
   4. Алгоритм Корфа …………………………………………………….7
   5. Алгоритм послойной сборки …………………………………….....8
   6. Алгоритм Фридрих ……………………………………………….....9
3. Исследование ……………………………………………………………10
   1. Первый этап ………………………………………………………....10
   2. Второй этап ………………………………………………………….13
4. Заключение ……………………………………………………………...15
5. Приложение ……………………………………………………………..16
6. Список литературы ……………………………………………………..22
7. **Введение**

На сегодняшний день Кубик Рубика является одной из самых популярных головоломок, известной по всему миру. С его изобретения в 1974 году люди успели придумать множество способов его сборки. Один из самых популярных и эффективных – метод Джессики Фридрих, натренированному человеку он позволяет собрать куб в среднем за 5 – 8 секунд. Существует организация WCA (World Cube Association), проводящая соревнования и фиксирующая результаты участников. Официально зарегистрированный рекорд на сайте WCA [1] на данный момент составляет 3,47 секунды. Но все же, несмотря на существование такой организации, в мире не существует единого мнения, насчёт самого оптимального алгоритма сборки кубика.

С массовым распространением компьютеров люди стали создавать программы для игры в различные настольные игры, такие как шахматы, шашки, карты и прочее. Кубик Рубика так же не остался в стороне, и различные люди сразу же начали думать над тем, как написать алгоритм сборки для компьютера. Ключевым показателем алгоритма, показывающим его оптимальность стало количество ходов, которые он тратит на полную сборку кубика и время, за которое он находит решение. В настоящее время уже придумано много таких алгоритмов, таких как алгоритм Коцембы и Тистлетуэйта. К сожалению, такой же авторитетной организации как WCA, проводящей соревнования между различными компьютерными алгоритмами сборки кубика Рубика не существует. И точно так же, как и с алгоритмами, используемыми людьми нет единого мнения насчет того, какой алгоритм самый оптимальный. Это и является проблемой в нашем исследовании.

Мы заинтересовались темой сравнения различных алгоритмов сборки кубика Рубика и нахождения самого оптимального и решили провести исследование, в ходе которого мы планируем найти в интернете реализации самых популярных алгоритмов сборки, после чего протестировать их на конкретных примерах перемешанных кубиков Рубика, найти средние значения времени сборки и количества ходов.

Цель нашего исследования – на основе полученных данных сделать вывод о том, какой из алгоритмов самый оптимальный по времени и количеству ходов в решении, а также определить в каких случаях лучше использовать те или иные алгоритмы.

1. **Обзор литературы**
   1. **Язык вращений и метрика хода кубика Рубика**

Язык вращений кубика Рубика - это способ записи вращений сторон кубика Рубика [8].

Метрика хода – это соглашение для подсчета количества ходов. Одна и та же последовательность ходов может иметь разное количество ходов в зависимости от используемой метрики [4].

Для проведения исследования мы использовали язык вращений WCA [9] в метрике хода FTM (HTM), где поворот любой грани под любым углом считается за 1 поворот [4].

Список разрешенных ходов в используемом нами языке вращений:

* По часовой стрелке на 90 градусов: F (передняя грань), B (задняя), R (правая), L (левая), U (верхняя), D (нижняя).
* Против часовой стрелки на 90 градусов: F', B', R', L', U', D'.
* На 180 градусов: F2, B2, R2, L2, U2, D2.
  1. **Число бога**

Число бога – это такое число n, что существует хотя бы одна конфигурация головоломки, оптимальное решение которой состоит из n ходов, и не существует ни одной конфигурации, длина оптимального решения которой превышает n [10].

в 2010 году Томасом Рокицки, Гербертом Коцембой, Морли Дэвидсоном и Джоном Детриджом было доказано, что число бога в метрике хода FTM (HTM) равно 20 [11].

Данную информацию мы использовали для оценки оптимальности решения, найденного алгоритмами сборки.

**2.3 Алгоритм Тистлетуэйта и его модификации**

Одним из первых, кто начал заниматься разработкой компьютерного алгоритмасборки кубика Рубика был Морвин Тистлетуэйт - британский профессор математики в Университете Теннесси в Ноксвилле [2]. Свой алгоритм он разработал в 1981 году.

Его алгоритм работает следующим образом: сначала делается несколько ходов, пока не возникнет позиция, из которой куб можно решить, не используя ходы U, D, U', D' (повороты на 180 градусов использовать можно). Затем он переходит к решению куба, не используя ходы U, D, U', D', пока не возникает позиция, из которой куб можно решить, не используя ходы F, B, F’, B’, U, D, U', D'. С этими дополнительными ограничениями достигается положение, которое для решения не требует поворотов на 90 градусов и, следовательно, может быть решено только с помощью поворотов на 180 градусов (U2, D2, R2, L2, F2, B2). На последнем этапе куб приводится в собранное состояние лишь с помощью поворотов на 180 градусов [12].

Список разрешенных ходов на каждом этапе алгоритма Тистлетуэйта:

1. U, D, B, F, R, L, U’, D’, B’, F’, R’, L’, U2, D2, B2, F2, R2, L2
2. B, F, R, L, B’, F’, R’, L’, U2, D2, B2, F2, R2, L2
3. R, L, R’, L’, U2, D2, B2, F2, R2, L2
4. U2, D2, B2, F2, R2, L2

Для поиска последовательности ходов, необходимых для приведения куба из одной позиции в следующую на каждом этапе используются большие справочные таблицы для всех позиций.

Алгоритм Тистлетуэйта не раз совершенствовался различными учеными. Например, в 1992 году Майкл Рид модифицировал алгоритм Тистлетуэйта, сократив количество этапов до 3 [3], и доказал, что для решения кубика Рубика из любого состояния достаточно 39 ходов в метрике хода FTM (HTM) [3].

Список разрешенных ходов на каждом этапе алгоритма Майкла Рида:

1. U, D, B, F, R, L, U’, D’, B’, F’, R’, L’, U2, D2, B2, F2, R2, L2
2. U, F, R, U’, F’, R’, U2, F2, R2
3. U, U’, U2, F2, R2

Так же в 1992 году алгоритм Тистлетуэйта модифицировал учитель математики из Дармштадта Герберт Коцемба, сократив количество этапов до 2 [14].

Список разрешенных ходов на каждом этапе алгоритма Коцембы:

1. U, D, B, F, R, L, U’, D’, B’, F’, R’, L’, U2, D2, B2, F2, R2, L2
2. R, L, R’, L’, U2, D2, B2, F2, R2, L2

Еще в 1992 году используя свой алгоритм Коцемба смог найти решения различных состояний куба менее чем за 22 хода [18].

**2.4 Алгоритм Корфа**

В 1997 году Ричард Корф создал алгоритм, позволяющий находить оптимальные решения кубика Рубика. Его алгоритм работает за счет разделения сборки куба на подзадачи, достаточно простые для того, чтобы осуществить полный перебор. Перебор осуществляется с помощью алгоритма IDA\* [13] [15].

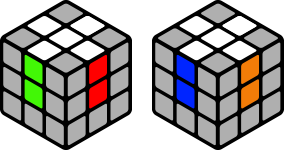
Ввиду того, что поиск решения осуществляется с помощью перебора, время за которое алгоритм Корфа находит решение может составлять от нескольких часов до нескольких недель [14].

**2.5 Алгоритм послойной сборки («Новичковый»)**

Алгоритм послойной сборки – это один из самых простых алгоритмов сборки и как правило, используя его, большинство людей учатся собирать куб, чем и объясняется такое название.

Алгоритм делится на следующие этапы [16]:

1. Сборка правильного креста



Правильный крест

1. Сборка правильной стороны (первого слоя)

Первый слой

1. Сборка второго слоя

Второй слой

1. Сборка желтого креста



Желтый крест

1. Сборка желтой стороны



Желтая сторона

1. Ориентация углов желтой стороны



Ориентированные углы

1. Ориентация ребер желтой стороны (окончание сборки)

**2.6 Алгоритм Фридрих**

В 1981 году Джессика Фридрих усовершенствовала алгоритм послойной сборки и сократила его до 4 этапов [17]:

1. Сборка креста
2. Сборка первых двух слоев (F2L)
3. Сборка желтой стороны (OLL)
4. Ориентация углов и ребер желтой стороны (PLL)
5. **Исследование**

**3.1 Первый этап**

Для проведения исследования нами был установлен модуль solvecube для языка программирования Python [6], в котором было реализовано 3 алгоритма сборки кубика Рубика, а именно алгоритм Коцембы, алгоритм Джессики Фридрих и «Новичковый» алгоритм.

Каждый из алгоритмов представлен как функция, принимающая строку, описывающую текущее состояние куба и возвращающая массив строк, каждая из которых – один ход в метрике FTM (HTM).

Далее нами был написан генератор скрамблов (скрамбл – последовательность движений для перемешивания кубика Рубика) на языке программирования C#. Исходный код на C# и Python можно найти на репозитории GitHub [7], а также в Приложении (Приложение 1, 2).

Генератор скрамблов представлен как функция, принимающая нужное количество скрамлов и количество ходов в каждом скрамбле. Функция выводит на экран скрамблы в метрике FTM (HTM), а также строки, описывающие куб после выполнения ходов из ранее сгенерированных скрамблов, в формате, который принимают функции сборки куба на языке python. Генератор работает следующим образом:

Все ходы скрамбла генерируются случайно. В случае, если сгенерированный ход описывает вращение той же грани, что описывает и предыдущий ход, текущий ход заменяется на случайный другой, не описывающий вращение этой грани. После генерации последнего хода алгоритм переходит к генерации следующего скрамбла и выводит сгенерированный скрамбл на экран. После генерации всех скрамлов алгоритм выводит на экран строки, описывающие состояние куба, после выполнения ходов из сгенерированных скрамблов.

Написанная нами python программа принимает любое количество строк, описывающих состояние куба, после чего находит решение для каждого из описанных кубов всеми тремя алгоритмами. Количество ходов определяется по количеству элементов массива строк, который возвращает функция, находящая решение куба. Для замеров времени работы алгоритмов программа замеряет время до вызова функции решения куба и после того, как функция отработала, время, затраченное на вывод решения на экран не учитывается. Полученная разность времени после окончания работы функции и до нее и является временем работы функции.

Мы проводили тесты сессиями из 100 скрамблов. Количество ходов в скрамблах одной сессии было одинаковым. В каждой последующей сессии количество ходов в скрамбле увеличивалось на 2. Таким образом в первой сессии количество ходов в скрамбле было равно 2, а в последней - 34. В ходе каждой сессии Python программа выводила решения скрамблов тремя вышеназванными алгоритмами, а также время работы каждого алгоритма и количество ходов в каждом решении. После чего на основе полученных данных мы создавали таблицу Excel, где высчитывали средние значения количества ходов в решениях каждого алгоритма и времени работы каждого алгоритма, ввиду того, что в каждой таблице по 100 позиций, мы были вынуждены предстваить их вам в электронном виде [5].

На основе полученных средних значений времени работы и количества ходов мы построили следующие графики:

**1**

**2**

Как видно по графику 1 среднее время работы алгоритма Коцембы стремительно возрастает после отметки 8 ходов, а после отметки 12 ходов среднее время работы алгоритма превышает 1 секунду, что критично, в случае, если важно быстродействие алгоритма. Среднее время работы алгоритма Коцембы возрастает до отметки в 2,5 – 3 секунды. Такие отклонения в измерении времени работы алгоритма могут быть вызваны выполнением других процессов компьютером. Среднее время работы «Новичкового» алгоритма и алгоритма Фридрих не возрастает.

Как видно по графику 2 среднее количество ходов в решении алгоритмов Фридрих и «Новичкового» возрастает до достижения отметки 14 ходов в скрамбле, после чего количество ходов в решении не увеличивается. Среднее количество ходов в решении алгоритма Коцембы возрастает до достижения отметки 16 ходов в скрамбле, после чего количество ходов в решении не увеличивается.

**3.2 Второй этап**

Обобщив все полученные данные, мы решили провести еще один тест, для получения более точных данных, касательно количества ходов в решении и времени работы каждого из алгоритмов, состоящий из 1000 скрамблов, в каждом из которых по 32 движения, так как на этой отметке прекращают свой рост все показатели всех алгоритмов. По итогам теста, мы также нашли худшие и лучшие значения времени и количества ходов и готовы вам представить следующую таблицу:

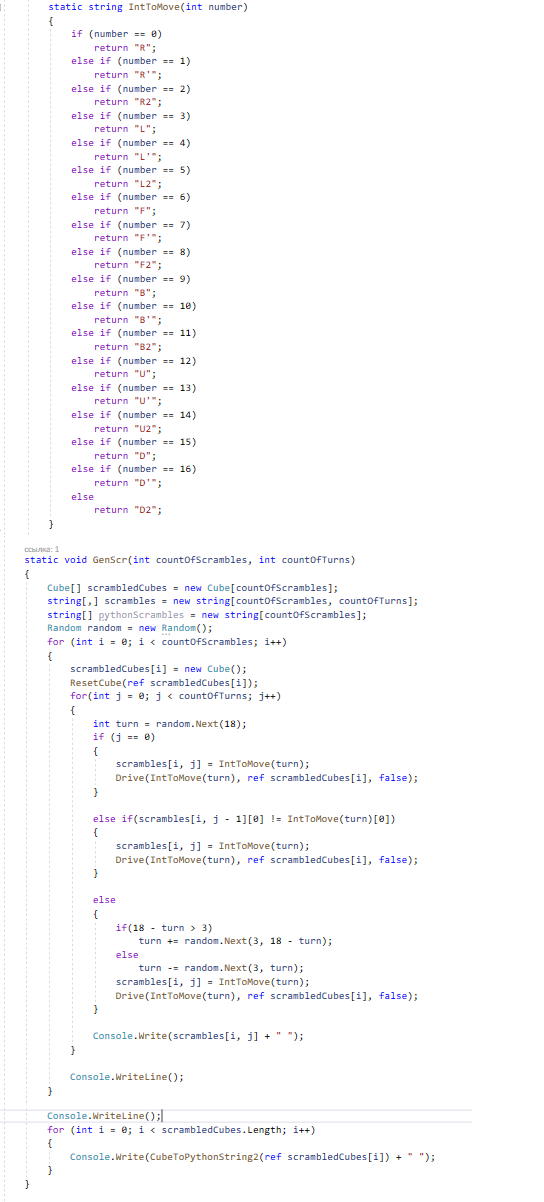
|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм |  | Время, с | Количество ходов |
| Коцемба | Среднее | 2,846012 | 20,71 |
| Минимальное | 0,006001 | 17 |
| Максимальное | 24,57182 | 22 |
| «Новичковый» | Среднее | 0,073968892 | 149,229 |
| Минимальное | 0,046009779 | 82 |
| Максимальное | 0,134030104 | 223 |
| Фридрих | Среднее | 0,050396998 | 74,63 |
| Минимальное | 0,033007383 | 51 |
| Максимальное | 0,091110945 | 95 |

1. **Заключение**

Исходя из данных таблицы и графиков можно сделать следующий вывод: в случае, если количество ходов в скрамбле меньше или равно 12, то для решения кубика Рубика лучше использовать алгоритм Коцембы, для получения решения из 17 – 22 ходов, в случае, если количество ходов в скрамбле больше 12 и для вас критично время работы алгоритма, то лучше использовать алгоритм Фридрих, для получения решения из 51 – 95 ходов, менее чем за 0,1 секунды. Для получения самого оптимального по количеству ходов решения нужно использовать алгоритм Коцембы, который находит решение, состоящее в среднем из 20,71 хода, что значительно меньше, чем у Фридриха (74,63 ходов) и «Новичкового» (149,229 ходов) алгоритмов. В среднем алгоритм Коцембы решает куб за 20,71 ходов, что всего на 0,71 больше числа бога. Это говорит о том, что алгоритм Коцембы можно считать одним из самых оптимальных алгоритмов в плане количества ходов в решении.

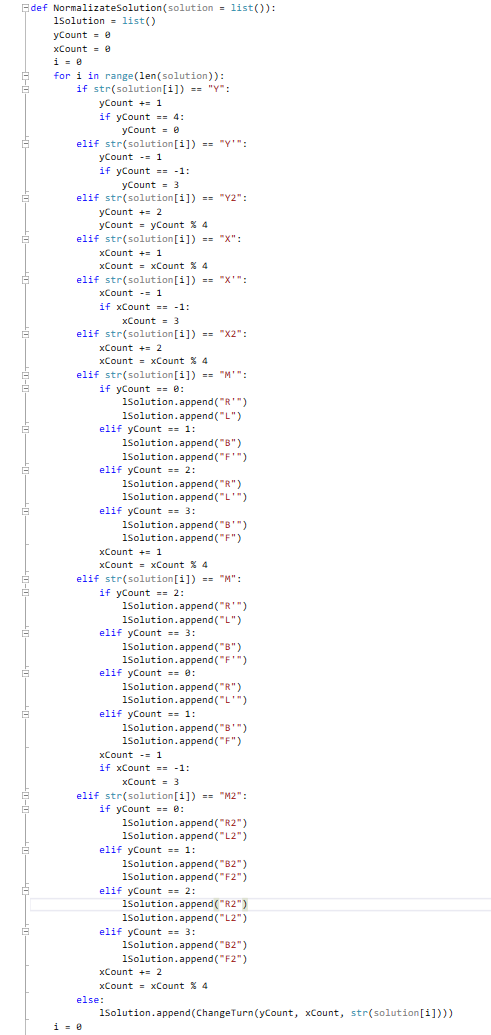
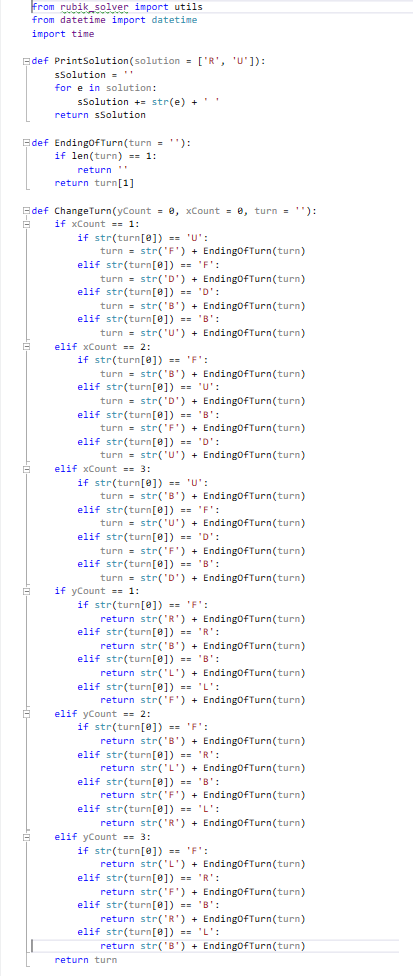
1. **Приложение**

**1**





**2**



****

1. **Список литературы**
2. <https://www.worldcubeassociation.org/results/records>
3. <https://buildwiki.ru/wiki/Morwen_Thistlethwaite>
4. <http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/michael_reid__new_upper_bound.html>
5. <https://www.speedsolving.com/wiki/index.php/Metric#:~:text=List%20of%20metrics-,HTM,one%20move%20instead%20of%20two>.
6. <https://drive.google.com/drive/folders/1TwYflkouitogVg09-zENA2IwM7Wydkmh?usp=sharing>
7. <https://pypi.org/project/solvecube/>
8. <https://github.com/azim2007/NIDResourceCode/tree/master>
9. <https://kubikus.top/metod-fridrih-kubik-rubik/52-yazyk-vrashchenij-kubika-rubika-3x3.html>
10. <https://www.worldcubeassociation.org/regulations/translations/russian/#12a1>
11. <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%B1%D0%BE%D0%B3%D0%B0>
12. <http://www.cube20.org/>
13. <https://www.jaapsch.net/puzzles/thistle.htm>
14. <https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0_%D0%BA%D1%83%D0%B1%D0%B8%D0%BA%D0%B0_%D0%A0%D1%83%D0%B1%D0%B8%D0%BA%D0%B0>
15. <https://books.google.ru/books?id=3aUMamjxVpMC&pg=PA58&redir_esc=y#v=onepage&q&f=false>
16. [https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9\_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4\_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0#IDA\*](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0#IDA*)
17. <https://speedcubing.ru/tutorial/3x3x3/beginner>
18. <https://speedcubing.com.ua/howto/fridrich>
19. <http://kociemba.org/math/imptwophase.htm>