Blockchain

Contents

[1 Einleitung 2](#_Toc95128512)

[2 Theoretischer Teil: 2](#_Toc95128513)

[2.1 Theorie 1 – Hashing Algorithmen 2](#_Toc95128514)

[2.2 Theorie 2 2](#_Toc95128515)

[3 Praktischer Teil 2](#_Toc95128516)

[3.1 Versuch 1 2](#_Toc95128517)

[3.2 Versuch 2 2](#_Toc95128518)

[4 Auswertung 2](#_Toc95128519)

[5 Fazit 2](#_Toc95128520)

[6 Literaturverzeichnis 2](#_Toc95128521)

# Einleitung

# Theoretischer Teil:

## Theorie 1 – Superflip

(<https://wikigerman.edu.vn/wiki30/2021/09/11/zauberwurfel-wikipedia/> - ND - 12.04.2022)

Singmaster Notation:

Um über Zauberwürfel reden zu können, müssen wir zuerst die Notation verstehen. Universell wird die Singmaster Notation benutzt. Sie funktioniert folgendermassen:

„

* F (Vorne): die Seite, die gerade dem Solver zugewandt ist
* B (Rückseite): die der Vorderseite gegenüberliegende Seite
* U (Up): die Seite über oder auf der Vorderseite
* D (Unten): die der Oberseite gegenüberliegende Seite, unter dem Cube
* L (Links): die Seite direkt links von der Vorderseite
* R (Rechts): die Seite direkt rechts von der Vorderseite

Wenn auf einen Buchstaben ein Strichsymbol ( ′ ) folgt, bedeutet dies eine Drehung gegen den Uhrzeigersinn; während ein Buchstabe ohne Prime-Symbol eine Drehung im Uhrzeigersinn bezeichnet. Diese Richtungen sind so, wie man auf das angegebene Gesicht schaut.

„ Zitiert von die O.G. quelle.

Würfeltheorie

(<https://ruwix.com/the-rubiks-cube/notation/> - Denes Ferenc – 16.04.2022)

Eine Permutation ist eine Position in der sich das Zauberwürfel befinden kann. Man kann Permutationen in zwei Kategorien unterteilen:   
Mögliche Permutationen sind Permutationen aus denen man das Zauberwürfel mit die 12 oben gegebene Bewegungen lösen kann.   
Unlösbare Permutationen sind Würfelpositionen mit denen man das Würfel nicht mit die normale Bewegungen lösen kann. Dass bedeutet dass jemand eine ecke verdreht hat, oder eine Kante, es gab ein Regelverstoß.

Wie können wir herausfinden wie viele Permutationen es geben kann. Folgendermassen:

Diagram, engineering drawing

Description automatically generated<https://3dwarehouse.sketchup.com/model/7e5c8f29d01deedf6dff1e05fba01b14/Rubiks-Cube-Core?hl=pl>

Wir wissen dass es 12 Kanten gibt, die in zwei weisen orientiert sein können, und 12 Ecken, die in 3 weisen orientiert sein können. Wenn wir uns vorstellen, dass wir das ganze würfel auseinander gebaut haben, und dass es nur noch den Kern übrig bleibt. Dann um die erste kante rein zu kriegen, kann man es in zwölf mögliche plätze tun, ohne an der orientierung zu denken. Dann wenn wir die zweite kante positionieren, haben wir 11 möglichkeiten, bis wir zur letzten kante kommen, da gibt es nur eine möglichkeit. Das ist 12!. Wenn wir jetzt dass selbe aber mit den Ecken machen, dann werden, da es 8 Ecken gibt, ist die anzahl von positionen in denen man die Ecken tun kann, 8!, weil alle Ecken anders sind. Dann müssen wir aber noch die orientierung beachten. Bei den Kanten gibt es zwei Orientierungen, und es gibt zwei orientierungen für die 12 Kanten, also ist der Faktor 212 und bei der Orientierung von den Ecken, gibt es 8 ecken, die in drei weisen orientiert sein können. Also 38 weil jede einzelne Ecke seine eigene orientierung hat.

212 \* 38 \* 12! \* 8! = 519.024.039.293.878.272.000. Dass ist die zahl für alle die positionen, wenn man das Würfel abbaut, und direkt in das Würfel reintut, ohne zu überprüfen, ob diese Permutation überhaupt durch legale drehungen gelöst werden kann. Dann dividiert man durch 12 und die antwort ist:

Es gibt 43.252.003.274.489.856.000 Moegliche Permutationen fuer ein Zauberwürfel.

Ich arbeite hier noch gerade,

<https://youtu.be/z2-d0x_qxSM>

<https://www.youtube.com/watch?v=o-RxLzRe2YE&ab_channel=JPerm>

## Theorie 2 – Devils Algorithm

# Praktischer Teil

## Code schreiben

Der Code wurde in Python 3.10 geschrieben, es benutzt 4 externe Büchereien:

* Copy: Um eine Deepcopy von unsere Datenstruktur machen zu können, für Clock und Anticlockturn
* Random für random\_scrambler, da will ich immer unterschiedliche drehungen durchführen, deswegen benutze ich ein zufälligkeitsmodul.
* Time um zu messen, wie lange der code gerannt ist.
* Colorama, um farben, im Text der Ausgangsdatenstruktur sehen zu können (sehr primitiv)

Es hat eine Länge von ungefähr 1500 Zeilen, und ist in Spanisch, Englisch und deutsch kommentiert.

A picture containing text, athletic game, sport

Description automatically generatedA picture containing shoji, building, clipart

Description automatically generatedDiagram

Description automatically generatedIn einem durchschnitt von 184 drehungen (Stichprobengrösse von 10 durchgänge) loest er es. Mit eine durchschnittliche zeit von 0,045 Sekunden.

Ich habe eine Datenstruktur definiert, die das Ausklappen eines Wuerfels entspricht. Es sind drei hauptlisten, die erste beinhaltet, die hintere seite(Orange). Die zweite hat die drei seiten die sich in die Mitte befinden, Grün, Weiß, Blau und Gelb. Die dritte liste hat die Fordere Seite. Die rote Seite.

Ich habe auch ein paar variablen ganz oben vom Code definiert, die zahlen, denen diese seiten vom Würfel zugeordnet wurden, entsprechen, die index positionen, von den verketteten listen in denen die Farben gespeichert wurden:

Redface = Frontface = 2, 1  
Whiteface = Upface = 1, 1  
Greenface = Leftface = 1, 0  
Blueface = Rightface = 1, 2  
Yellowface = Downface = 1, 3  
Orangeface = Backface = 0, 1

Dann habe ich zwei hauptfunktionen definiert, Clockturn(seite=) und AntiClockturn(seite=) die als parameter die seite die zum drehen benutzt wird nehmen. Mein ganzes code ist nach diese Funktionen definiert. Ich habe die drehungen gemacht, indem ich eine Kopie vom gegenwärtigen Würfelposition eine kopie gemacht habe, und indem ich die neue position der farben die kopierte und bewegte position der alten zugebe, wie es im bild zu sehen ist. Ich habe eine kopie gemacht, weil ich sonst die information verliere, von wie der würfel gerade aussieht verliere.

Mit diese Starke basis, von diese Stammalgorithmen, bin ich weitergegangen, indem ich die Beginners Method vom Loesen des Zauberwuerfels implementiert habe.

### Diagram Description automatically generatedVersuch 1 – Zauberwürfel lösen – Anfänger Methode

(<https://ruwix.com/the-rubiks-cube/how-to-solve-the-rubiks-cube-beginners-method/> - Denes Frenc – 15.04.2022)

White Cross:

Dafür muss man zuerst ein kreuz im würfel machen, und die kanten müssen die Mitten-Steine zustimmen. Das schwere dabei ist, dass wenn man zu eine Person zeigt, wie man dieses Teil des Würfel löst, gibt es keine Algorithmen, sondern es geht mit die Intuition, und Computer haben eine hohe rechen Leistung, aber Intuition haben sie keine. Die loesung ist alle die weisse kanten um das Gelbe Mittelpunkt zu kriegen, dann kann ich sehen ob die farben von die zweite stufe übereinstimmen, und dann kann dreht es die jeweilige seite zwei mal. Dafür gibt es die funktionen yellow\_dot\_white\_cross und white\_cross.

White Corners:  
Dann nehme ich die weisse ecken hoch, mit einem sehr einfachen algorithmus hoch. R’D’RD ich habe dann auch noch das FDF‘ implementiert, was die anzahl bewegungen bis der würfel fertig ist, brutal runtergebracht hat. Um ungefähr 20 drehungen im durchschnitt.

Second Layer:  
Dann habe ich das Second layer geloest, mit die zwei algorithmen die es gibt: U'L'ULUFU'F' und URU'R'U'F'UF ich habe die Fälle geschrieben in denen diese algorithmen anwendbar sind.

Yellow Cross:  
Um das gelbe kreuz zu lösen, wird dieses Algorithmus benutzt , FRUR’U’F‘.

Yellow Edges:  
Dann muss man die gelbe kanten mit RUR'URU2R'U loesen, wenn sie nicht schon geloest sind.

Permute Corners:   
Um die gelbe ecken auszutauschen benutzt man dieses Algorithmus URU'L'UR'U'L.

Orient Corners:  
Um die gelbe ecken zu orientieren benutze ich wieder der algorithmus um die ecken zu orientieren: R’D’RD. Und dann ist es gelöst.

## Versuch 2

# Auswertung

# Fazit

# Literaturverzeichnis