Road to 3BLD

Autor: Cub de rubik

Modaltitat: Matmàtiques i Informàtica

15 de Desembre de 2023

Índex

Abstract

In this work I have learnt the techniques, methods and practices to solve the Rubik's cube blindfolded. In addition, all the content of the practical part of the work has been carried out with the aim of improving the learning of these methods. The practical part includes an application, a website and a tutorial where the whole process to solve the Rubik's cube blindfolded is explained.

En este trabajo ha sido realizado un aprendizaje de las técnicas, métodos y prácticas para resolver el cubo de Rubik a ciegas. Además, todo el contenido de la parte práctica del trabajo has sido realizado con el objetivo de mejorar el aprendizaje de estos métodos. La parte práctica incluye una aplicación, una web y un tutorial donde se explica todo el proceso para realizar el cubo de Rubik a ciegas.

Introducció

Vaig descobrir el cub de Rubik quan era petit i a casa feia competicions amb la meva família de fer una cara del cub, però mai l'havia arribat a fer sencer, algun Nadal m'havien regalat algun cub més bo, però tenia por a no saber tornar-lo a fer.

Als 12 anys a primer de l'ESO, un dia vaig decidir aprendre a fer-lo i vaig buscar a YouTube tutorials per poder resoldre'l i ho vaig aconseguir, aquell dia no vaig parar de desfer i fer, va ser com una connexió amb el cub que em cridava a tornar-lo a fer i a millorar-me a mi mateix. No vaig tardar a millorar el meu temps i vaig començar a provar cubs nous fins que va arribar un punt a on vaig perdre la motivació per fer els cubs, ja que aprendre noves categories no em cridava l'atenció. Vaig intentar fer el cub de Rubik a cegues, però vaig fallar i ho vaig deixar apart. Llavors va ser a primer de batxillerat escollint tema del TdR que em vaig decantar per aquest tema, i el treball no només serà aprendre, sinó que també ajudar a tothom a aprendre.

Durant aquest treball també entro en l'àmbit de la programació, una habilitat que jo no domino i només en tenia una petita base que vaig fer a quart d'ESO, igual que amb els cubs també he fallat anteriorment en aprendre-ho correctament. En aquest treball utilitzaré els llenguatges de programació Python, HTML, CSS i el sistema de composició de textos LaTeX.

Així aquest treball busca com a objectiu personal aprendre a fer el cub de Rubik sense mirar i millorar en l'àmbit de la programació, i com a objectiu del treball, fer un tutorial complet amb tota mena de recursos que puguin ajudar a tothom a entendre a fer el cub de Rubik sense mirar.

La motivació d'aquest treball ha estat aquests intents fallits a l'hora d'aprendre i el "fracàs" que havia experimentat.

Per fer aquest treball necessitaré un ordinador amb connexió a internet, el meu cub 3x3 i la meva passió pels cubs.

Metodologia

Durant aquest treball s'ha seguit una metodologia molt mecànica que es basa en aprendre-aplicar-aprendre-aplicar-redactar, és a dir, que per a cada àmbit del treball hi ha hagut primer una investigació prèvia, és a dir que en comptes de començar a investigar-ho tot i després aplicar-ho a la part pràctica, s'ha anat fet parts i l'última cosa del treball ha sigut la redacció.

Primer vaig començar buscant informació dels algoritmes de 3BLD i el seu concepte, després de buscar aquests algoritmes, vaig dedicar una gran part del temps a aplicar-ho i més tard vaig començar amb els llenguatges de programació ajudant-me de tutorials fins a arribar al resultat que volia, després de tot em vagi centrar en el LaTeX i la redacció.

Resumint, el treball s'ha fet pas a pas assegurant-me que tot el que aprenia se'm quedava al cap i no m'anava a oblidar durant els trams que no ho practiqués.

Part I

Marc Teòric

Capítol 1

Història del Cub de Rubik

1.1 Invent i Introducció (1974-1980)

El Cub de Rubik, va ser creat per Ernő Rubik el 1974 com una eina d'ensenyament dels conceptes espacials a estudiants d'arquitectura, es va llançar el 1975 a Budapest amb el nom "Cub Màgic". El seu disseny original constava de cares de colors sòlids. Aviat, el cub es va estendre per tot el món, però la seva resolució semblava un enigma que estava a l'abast de poca gent. [10] [19]

1.2 Primers Intents de Resolució (1980-1981):

David Singmaster, un estudiant d'enginyeria mecànica a Londres, va desenvolupar la primera notació per descriure els moviments del Cub i va crear el "mètode Singmaster" per resoldre'l, un fet molt important pel cub. [14]

1.3 Aparició dels Primers Campions (1982-1992):

A la dècada de 1980 van tenir lloc les primeres competicions de Cub de Rubik oficials , gràcies a la popularitat que estava agafant. Amb competicions de velocitat que van començar el 1982. Minh Thai es va convertir en el primer Campió Mundial. Més tard els mètodes de resolució van evolucionar, i el mètode Friedrich de Jèssica Friedrich es va convertir en un dels més populars.

1.4 L'Època dels Speedcubers (2003-2010):

La World Cube Association (WCA) es va fundar el 2003, establint estàndards i competicions oficials. Es va professionalitzar i van sorgir Speedcubers[1] que van elevar els nivells de les competicions.

1.5 L'Arribada de 3BLD (2004):

Més tard es va introduir la categoria de resolució a cegues (3BLD) a la WCA i Tyson Mao es va convertir en el primer campió de 3BLD el 2004.

1.6 Actualment(2023):

El Cub de Rubik i el 3BLD continuen sent una categoria molt emocionant i important en la comunitat de l'speedcubing i actualment el rècord de 3BLD es troba en 12.10 per Charlie Eggins d'Austràlia. Actualment s'estan buscant tècniques encara més avançades pel 3BLD, com algoritmes gairebé el doble d'eficients, però a la vegada el doble de casos. No sabem la direcció que agafarà el 3BLD, però de segur que ens espera un gran futur.[23]

[1]Persones que poden fer el cub de Rubik en molt poc temps

Capítol 2

Entrant al concepte del Cub de Rubik

2.1 Interpretar el concepte del cub

Una gran majoria de la població ha tingut a les seves mans un cub de Rubik, i han intentat resoldre'l sense èxit. Això és totalment normal, ja que només el saben resoldre un 5,8% de les persones que ho han intentat. [18]

Aquesta xifra se sol atribuir a la dificultat del cub, però després d'aprendre a fer el cub de Rubik te n'adones compte de què la raó no és aquesta. El fracàs a l'hora de trobar la solució ve donat pel fet d'interpretar malament el concepte del funcionament del cub.

La majoria de les persones es pensa que el cub conté 54 "peces" de colors perquè calculen que per cada cara hi ha 9 peces i en un cub hi ha 6 cares, per tant, estan treballant color a color.

$$\text{Nº Quadrats} = 3 \text{ Quadrats} * 3 \text{ Quadrats} * 6 \text{ Cares} = 54$$

La manera correcta d'interpretar el cub és pensar en el funcionament, com si el desmuntessis, ja que consta de 12 arestes i 8 cantonades, a més a més dels 6 centres que no poden permutar[1] amb cap altra peça ja que, només roten.

[1] Intercanvi de posició amb una altre peça i de l'ordre de tot el conjunt

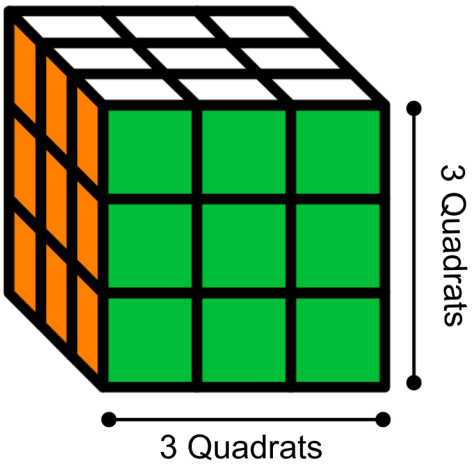

Figura 2.1: Plantejament típic però erroni del cub de rubik

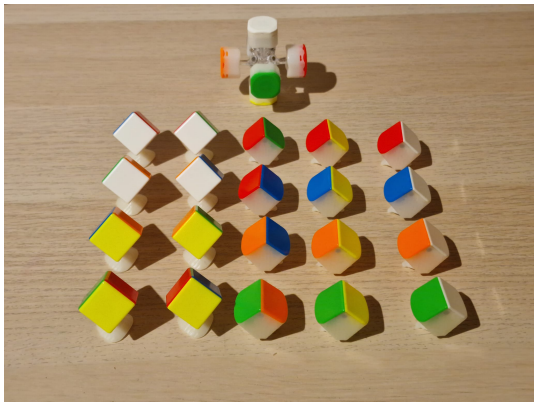

Figura 2.2: Cub Desmuntat

2.2 Aplicar les matemàtiques al concepte

Després d'entendre el funcionament podem aplicar les matemàtiques i extreure el nombre de combinacions possibles del cub. En primer instant dividim el càlcul en dos grups, per una part tenim cantonades i per l'altra arestes. Per la part de les cantonades es calcula:

Tenim 8 cantonades que es poden posar de manera aleatòria en els 8 llocs, i això es calcula com a $8!$ [2], després aquestes es poden orientar en 3 direccions diferents que matemàticament és 3^8. Per tant, les combinacions teòriques possibles amb un cub de només cantonades són:

$$\text{Nº Combinacions cantonades tèoric} = 8! * 3^8 = 264.539.520$$

[2]! és el símbol de factorial o $8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$

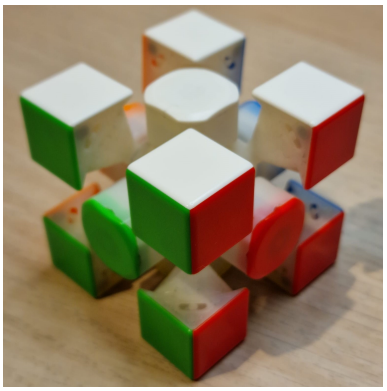

Figura 2.3: Cub amb només cantonades

Per altra banda, tenim 12 arestes que igual que amb les cantonades es calcula com a $12!$ i com que les arestes del cub només tenen dues orientacions ho multiplicarem per 2^12. Per tant, les combinacions teòriques possibles amb un cub de només arestes són:

$$\text{Nº Combinacions arestes tèoric} = 12! * 2^12 = 1.961.990.553.600$$

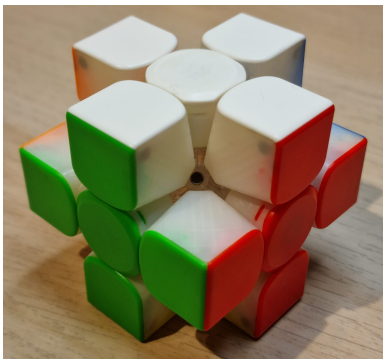

Figura 2.4: Cub amb només arestes

Llavors amb aquests càlculs podem extreure les conclusions que les combinacions possibles teòriques d'un cub de rubik són:

$$\text{Nº Combinacions tèoriques} = 8! * 3^8 * 12! * 2^{12} = 519.024.039.293.878.272.000$$

Però cal dir que aquestes no són les combinacions totals reals del cub de Rubik, ja que aquestes combinacions estan calculades com si demuntéssim el cub com a la 2.2 i recol·loquéssim les peces en un estat aleatori. Llavors per calcular les reals s'han de dividir per 12 els casos per restriccions com les que es mostren en la següent figura.

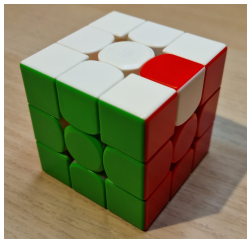
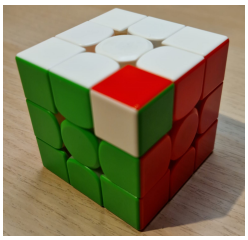
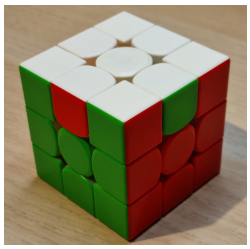

Figura 2.5: Exemple de casos impossibles

I tot ben calculat queda:

$$\text{Nº Combinacions Reals} = \frac{8! * 3^8 * 12! * 2^{12}}{12} = 43.252.003.274.489.856.000$$

Capítol 3

Notació dels Moviments

El cub de Rubik es resol gràcies a identificar patrons i executar algoritmes que resolen aquests patrons, aquests algoritmes han d'estar escrits en alguna part per poder-los memoritzar, i per això està la notació del cub de Rubik.

La notació consta de 6 moviments (F,B,R,L,U,D), que correspon a (Front, Back, Right, Left, Up, Down) que són les respectives direccions en anglès. Per exemple si faig el moviment F, gira la cara front, la que està més propera a la nostra visió, en sentit horari, en canvi, si fós F' seria antihorari. En les figures següents es mostra una representació gràfica per a cada capa. És un concepte difícil d'entendre, però de manera simplificada és girar la cara en sentit horari i antihorari des de la cara que vulguis. En les figures següents es mostra una representació gràfica per a cada capa.

Figura 3.1: Exemples de Movimients F y F'

Figura 3.2: Exemples de Movimients B y B'

Figura 3.3: Exemples de Movimients R y R’

Figura 3.4: Exemples de Movimients L y L’

Figura 3.5: Exemples de Movimients U y U’

Figura 3.6: Exemples de Movimients D y D’

Capítol 4

El Concepte de 3BLD

Per començar, cal entendre el funcionament d'una resolució de blind, primer el cub és barrejat per una persona i el posa dins d'una capsa o un cube cover[1], després es col·loca a la taula boca avall i la persona que l'ha de resoldre es pren el seu temps per respirar. Un cop fet això la persona que resol el cub encén el timer i destapa el cub, de manera que el temps comença a comptar i es comença a memoritzar. Un cop acabada la memorització el que resol el cub es tapa els ulls amb un antifaç i comença a resoldre el cub, mentre que una persona externa li posa una cartulina entre el cub i la seva cara per evitar trampes i mirar per sota de l'antifaç. Tots aquests passos s'han d'executar perfectament per asegurar-se de la resolució compti.

Figura 4.1: Materials necessaris per poder executar blind

[1] Un cube cover és una tapa per cubs feta de cartró i que s'utlitza a les competicions

En la imatge 4.1 es poden veure el timer[2], l'antifaç, la caixa per cobrir el cub, que en aquest cas jo utlizo una que tinc d'un cub, i a més a més uns cascos d'obra per aïllarte del soroll ambient.

4.1 Fases de la Resolució

Com ja he esmentat a la secció anterior, completar el cub de Rubik amb els ulls tancats, es divideix en dues grans fases, memorització i execució. I dins d'aquestes fases hi ha diferents procediments per poder aconseguir fer-ho correctament.

4.2 Memorització

Durant aquesta fase de memorització, com ja ho diu el seu nom, s'ha de memoritzar el cub. Molta gent pensa que els speedcubers que practiquem blind memoritzem el cub color per color mitjançant la memòria fotogràfica, però la veritat no és així, perquè la memòria fotogràfica només la té molt poca gent, i bé, jo m'en recordo dels objectes que tinc a la taula si tanco els ulls ara mateix, però memoritzar el cub d'aquesta manera porta molt de temps i no és la més eficient de fer-ho. El que fem és convertir aquestes posicions on estan les peces del cub, que "només" són 20 en lletres i ho fem d'una manera distribuïda en ordre que nosaltres ens memoritzem. L'esquema de lletres[3] que utilitzo es veu a la figura 4.2.

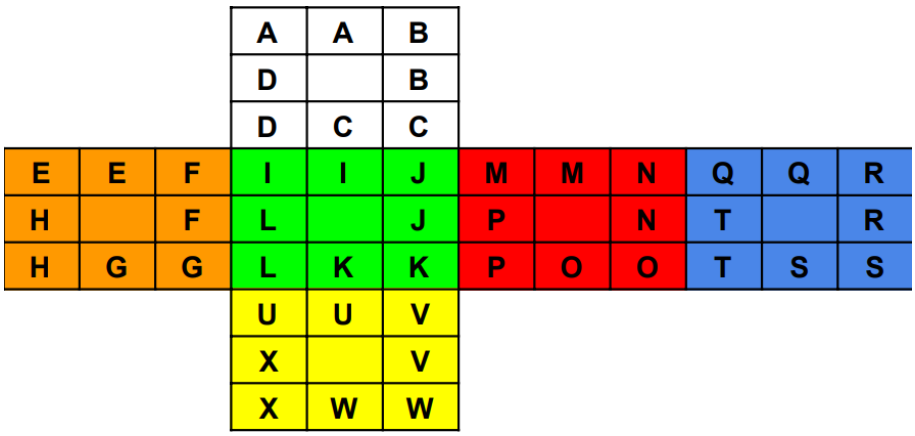

Figura 4.2: Esquema de Lletres

[2]El timer és el compatdor amb la forma de les mans que es veu al centre de la imatge
[3]És la distribució de lletres

Com es pot veure hi ha lletres repetides i això és degut al fet que hi ha memorització per arestes i memorització per cantonades. Com està mencionat a la secció 1.2.2 en el cub hi ha 8 arestes i 12 cantonades, per tant, haig de memoritzar respectivament 12 lletres d'arestes i 8 lletres de cantonades. Un altre cop tenim el problema de què no és gaire eficient memoritzar les lletres una per una i és per això que la manera correcta de fer-ho és la següent. Memoritzar dues lletres i amb aquestes dues lletres formar una paraula de la qual et sigui fàcil pensar en una imatge la qual pots recordar fàcilment. En resum, és convertir parells de lletres en en imatges [1] [5], per tant, ara tenim la meitat d'ítems a memoritzar. Un exemple d'aquesta fusió de lletres és:

Haig de memoritzar les lletres R i B $\rightarrow$ RedBull

Haig de memoritzar les lletres A i C $\rightarrow$ Aire Acondicionat (AC és el símbol)

De manera pràctica es comença a memoritzar des de la peça UK mirant el color de U, de la lletra en la posició U que és la inicial i treus una lletra i llavors mires a la posició on ha d'anar aquesta primera lletra que has trobat i mires quina lletra treus, i així fins que memoritzis totes les arestes i després fas el mateix amb les cantonades. És un concepte difícl d'explicar amb paraules i es veu millor al següent exemple. Cal destacar que en començar a memoritzar la posició UK la saps perquè col·loques el centre verd mirant cap a tu i el centre blanc mirant cap a dalt.

BARREJA: F2 B R2 U'L2 U2 B' L' F2 U' B2 U L2 U R2 F2 L2 U' L2 F

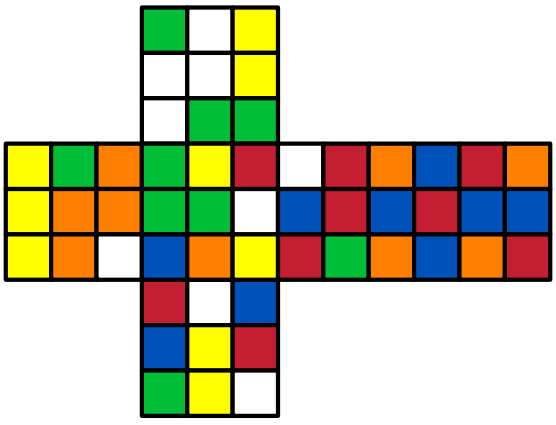

Figura 4.3: Cub barrejat per exemple de Blind

Comencem mirant al lloc de U K com a l'esquema de lletres i veiem que és blanca a lloc U i taronja al lloc K, llavors ens fixem en l'esquema de lletres en l'aresta blanca-taronja i veiem que és la lletra D. Després ens fixem en el lloc de la lletra D i està una peça blanca-verda que és la lletra C, però és un cas especial, que més tard parlo a la secció d'execució, però aquesta C es converteix en W, no s'ha de saber res més, només és per tenir el concepte entès. Això ho fem successivament i obtenim que les lletres totals que ens hem de memoritzar són:

Memorització Arestes: DW LA BV PX RI GT N

(En aquest cas surten 13 perquè ha sigut una barreja amb cas especial)

Memorització Cantonades: U MH VS MC

(Igual que amb les arestes al ser un cas especial el valor es veu afectat)

Memorització Total DW LA BV PX RI GT NU MH VS MC

Lletres	Transcripció
DW	DeU (pronunciació)
LA	Los Ángeles
BV	BBVA
PX	PiXel
DF	Districte Federal
Ri	RIu

Lletres	Transcripció
GT	Gran Turismo
NU	NUt (Femella en anglés)
MH	MoHa
VS	VerSuS
MC	MigCampista

4.3 Execució

L'execució és la segona fase per completar el cub a cegues i es fa d'una manera molt diferent de la que es fa el cub de manera convencional. Quan resols el cub mirant fas rotacions al cub per buscar les peces i fer moviments, en canvi, a les resolucions a cegues el cub no es rota en tota l'execució. Llavors les peces s'aconsegueixen col·locar a lloc gràcies a algoritmes[4]. Com que el cub no ha de rotar, les peces també s'han de quedar en el mateix lloc en l'acabar l'algoritme, és a dir, és executar un algoritme, mous les dues peces que vols i acabes amb la

[4]És una seqüència de moviments que permet realitzar una tasca

resta del cub igual però amb les dues peces canviades. [24]

Hi ha diferents nivells d'eficiència d'algoritmes, i és perquè amb els algoritmes més complicats es pot fer de manera més eficient i per tant, hi ha menys moviments, però, en canvi, és més difícil d'aprendre-se'ls. L'explicació darrere d'això és:

Quan fas un algoritme fas $\rightarrow YXY'X'$

El que es fa, és executar uns moviments que li direm seqüència Y, que normalment aquesta és per preparar el lloc on les peces es canviaran, després fem una seqüència X, en aquesta el que fem és intercanviar les peces entre elles, després fem la seqüència Y' que és la seqüència i al revés, aquesta el que fa és tornar enrere el moviment fet anteriorment i preparar el retorn de la seqüència Y'. FInalment, amb la seqüència Y' el que fem retornar el cub a l'estat anterior però amb les peces objectius canviades.

Un exemple d'això és un intercanvi de tres cantonades, que visualment pas a pas aplicant una seqüència darrera de l'altra es veu:

Figura 4.4: Secuencia Y (Esquerra) i Y X (Dreta)

Figura 4.5: Secuencia Y X Y' (Esquerra) i Y X Y'X' (Dreta)

Com es pot veure tot acaba per intercanviar 3 cantonades deixant el cub exactament igual, les seqüències són els moviments següents.

- X = U
- Y = R' D R

- X' = U'
- Y' = R' D' R

Aquesta explicació s'atribueix al mètode més avançat, que no és el que jo utilitzo, ja que el mètode més avançat són 400 algoritmes per cantonades i gairebé 400 per arestes, a més a més es tarden anys d'experiència no només per memoritzar sinó per obtenir la memòria muscular[5].

Jo utilitzo el mètode de dificultat mijta-alta, però que també té gran eficiència, tinc un mètode per arestes i un mètode per cantonades.

4.4 El mètode d'execució per les arestes.

Per les arestes utilitzo el mètode M2, que com ja ho diu el seu nom es basa en el moviment M2, que és el moviment de la capa del mig del cub 2 vegades.

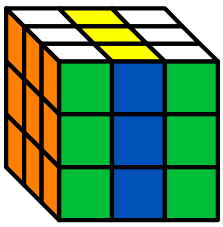

Figura 4.6: Exemple de Moviment M2

Aquest mètode intercanvia les peces d'una manera peculiar, ja que ha de fer dues vegades M2 per tornar a l'estat original i canviar dues peces. Per exemple, fas primer la seqüència Y que col·loca la peça al lloc d'intercanvi, després fas la seqüència X que en aquest cas és M2 i després fas Y' per retornar la peça intercanviada. Un cop fet això s'han canviat dues peces, però el cub no queda igual que abans perquè hem de tornar a fer un intercanvi, aquest segon intercanvi ha de ser amb una seqüència Z X Z' perquè hem d'intercanviar una peça que no sigui la mateixa. El mètode té aquests casos següents de les taules 4.1

Llavors a les figures 4.7, 4.8, 4.9, es mostra com es canviarien dues arestes amb el mètode M2, en aquest cas L i V.

[5]És la memòria que fa que nosaltres movem els músculs d'una manera quan pensem una cosa, com quan escrius a l'ordinador.

A	M2
B	R' U R U' M2 U R' U' R
C	U2 M' U2 M'
D	L U' L' U M2 U' L U L'
E	B L' B' M2 B L B'
F	B L2 B' M2 B L2 B'
G	B L B' M2 B L' B'
H	L B L' B' M2 B L B' L'
I	D M' U R2 U' M U R2 U' D' M2
J	U R U' M2 U R' U'
K	Posició d'intercanvi
L	U' L' U M2 U' L U

M	B' R B M2 B' R' B
N	R' B' R B M2 B' R' B R
O	B' R' B M2 B' R B
P	B' R2 B M2 B' R2 B
Q	U B' R U' B (M2) B' U R' B U'
R	U' L U M2 U' L' U
S	M2' D U R2 U' M' U R2 U' M D'
T	U R' U' M2 U R U'
U	Posició d'intercanvi
V	U R2 U' M2 U R2 U'
W	M U2 M U2
X	U' L2 U M2 U' L2 U

Taula 4.1: Taules Algoritmes M2

Figura 4.7: Secuencia Y (Esquerra) i Y X (Dreta)

Figura 4.8: Secuencia Y X Y' (Esquerra) i Y X Y'Z (Dreta)

Figura 4.9: Secuencia Y X Y'Z'X (Esquerra) i Y X Y'Z X Z' (Dreta)

4.5 Mètode d'execució per les Cantonades

Per les cantonades utilitzo el mètode Orozco, que utilitza un sistema similar al M2, ja que fa les seqüències Y X Y'X' i Z A Z' A' però en aquest cas la seqüència Z és diferent perquè es troba al segon lloc. De manera simple, si a la memorització tens la lletra, en segon lloc, has de fer l'algoritme alternatiu. Els casos d'Orozco són els següents:

AB	Basic A Perm
DB	U' (A Perm) U
EB	[R: [R D R', U]]
FB	[R': [U', R' D' R]]
GB	[U, R' D R]
HB	[R D' R', U']
IB	[R: [R D R', U2]]
KB	[D': [U, R' D R]]
LB	[D: [U, R' D' R]]
OB	[R D R', U']
PB	[U, R' D' R]
RB	[R': [U2, R' D' R]]
SB	[U, R' D2 R]
TB	[D: [R D' R', U']]
UB	[x': [R U R', D2]]
VB	[D' x': [R U R', D2]]
WB	[D x: [D2, R' U' R]]
XB	[x: [D2, R' U' R]]

BA	Reverse A Perm
BD	U' (Reverse A Perm) U
BE	[R: [U, R D R']]
BF	[R': [R' D' R, U']]
BG	[R' D R, U]
BH	[U', R D' R']
BI	[R: [U2, R D R']]
BK	[D': [R' D R, U]]
BL	[D: [R' D' R, U]]
BO	[U', R D R']
BP	[R' D' R, U]
BR	[R': [R' D' R, U2]]
BS	[R' D2 R, U]
BT	[D: [U', R D' R']]
BU	[x': [D2, R U R']]
BV	[D' x': [D2, R U R']]
BW	[D x: [R' U' R, D2]]
BX	[x: [R' U' R, D2]]

Taula 4.2: Algorimtes orozco

A la columna de l'esquerra hi ha els corresponents a la primera lletra del parell i a la dreta els corresponents a la segona lletra del parell. *A Perm* és un cas del cub de Rubik normal que es fa (R' U' R' D' R U' R' D R U R' D' R U R' D R2).

Aquests algoritmes estan escrits en una notació[6] diferent [Y,X]. El fet que estigui entre claudàtors indica que s'ha de fer en l'ordre Y X Y' X'. És una mica més difícil de visualitzar però

[6]Manera d'escriure els algoritmes

NB	[U, R' D R D' R' D' R]
QB	[R' D R D' R' D' R, U]

BN	[R' D R D' R' D' R, U]
BQ	[U, R' D R D' R' D' R]

Taula 4.3: Excepcions del mètode

més fàcil a l'hora d'aplicar aquest mètode. Exemple d'intercanvi de dues cantonades P i H

Figura 4.10: Secuencia Y (Esquerra) i Y X (Dreta)

Figura 4.11: Secuencia Y X Y' (Esquerra) i Y X Y' X'(Dreta)

Figura 4.12: Secuencia Y X Y' X' Z (Esquerra) i Y X Y' X' Z A(Dreta)

Figura 4.13: Secuencia Y X Y' X' Z A Z' (Esquerra) i Y X Y' X' Z A Z' A'(Dreta)

Com a orientació [Y,X] i [Z,A] són a la taula d'algoritmes PB i BH.[13]

Part II

Marc Pràctic

Capítol 5

Redacció d'aquest treball amb LaTeX

Tot aquest treball escrit està redactat en LaTeX, que és un sistema de composició textos orientat a la creació de documents profesional, especialment s'utilitza en la redacció d'articles científics. LaTeX no és un sistema WYSIWYG, que significa "Allò Que Veus És Allò Que Obtens". En lloc d'això, amb LaTeX, treballes amb un text sense format amb etiquetes. Aquestes etiquetes són instruccions del que ha de ser el text. Un exemple amb seccions i subseccions a LaTeX es veuria com al listing 5.1 i la figura 5.1.

```
1 \section{Secció de Prova}
2 Text de prova.
3
4 \subsection{Subsecció de prova}
```

Listing 5.1: Exemple de Secció i Subsecció a LaTeX

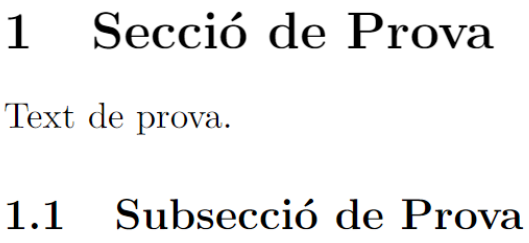

Figura 5.1: Exemple de Secció i Subsecció a LaTeX

A més a més, he estat mantenint un control de versions, però no està ennlaçat perquè està a Github i surt el meu nom, el que es pot veure es el codi sencer a cubderubik/TDR

5.1 Les Figures del Document

Les figures del document són totes d'elaboració pròpia, tant les imatges com els cubs representats al document. Els cubs estan fets també a LaTeX, utlitzen el paquet TikZ, i dins d'aquest paquet es troben els de rubikcube i rubikrotation per fer els cubs i que es mostrin amb els moviments inserits. El codi per formar un cub es pot veure al listing 5.2 i el resultat a la figura 5.2, hi ha moltes etiquetes, però el que mana a l'hora de "fer els moviments" al cub i que es mostri en una orientació o altra són les etiquetes de "RubikRotation" i "DrawRubikCube".

```
1   \begin{figure}[htbp]
2       \centering
3       \begin{subfigure}
4           \centering\RubikCubeSolvedWY
5           \RubikRotation{Y,M2,S2,E2}
6           \ShowCube{7cm}{0.5}{\DrawRubikCubeLU}
7       \end{subfigure}
8       \begin{subfigure}
9           \centering\RubikCubeSolvedWY
10          \RubikRotation{M2,S2,E2}
11          \ShowCube{7cm}{0.5}{\DrawRubikCubeRD}
12      \end{subfigure}
13  \end{figure}
```

Listing 5.2: Exemple de Cubs fets amb el paquet rubikcube

Figura 5.2: Cubs de demostració fets amb el paquet rubikcube

Els trossos de codi que es mostren durant l'explicació de l'App al capítol 6 són elaborats amb el paquet listings i s'escriu de la següent manera el codi 5.3

```
1   \begin{lstlisting}[language=Python, style=colorEX, caption=Inici
        de la classe i especificació de la finestra]
```

```
2         print("Hello World")
3     \end{lstlisting }
```

Listing 5.3: Exemple de codi Hello world mostrat amb el paquet listings

El meu tutor em va recomanar fer-ho en LaTeX i ell m'ha guiat durant tot el treball, també he buscat el funcionament d'algunes pel meu compte. [16] [20] [21]

Capítol 6

Elaboració d'una App per Practicar la Memorització

En saber tots aquests conceptes vaig decidir que hauria de portar aquesta eficiència a l'hora de resoldre, a l'aprenentatge. És a dir, que a la vegada que per resoldre el cub a cegues es busca l'eficiència en cada pas, però a l'hora d'aprendre tota aquesta llista de conceptes és un procés molt lent. I és per això que per posar en pràctica el procés de memorització és a dir traduir les lletres a les paraules per visualitzar imatges, el que vaig fer és una aplicació que et generi lletres, després te les amagui i tu hagis de posar les lletres que has memoritzat.

Aquesta app està realitzada amb el llenguatge de programació Python, i amb la biblioteca tkinter per crear interfície gràfica[1].

L'app consta de diferents labels[2], botons i textboxes[3], que es mostren es deixen de mostrar segons la funció que es doni i cada una d'aquestes funcions és escollida mitjançant els botons. Durant aquesta explicació del codi cal tenir en compte que les frases de color verd escrites després d'un asterisc són comentaris i no afecten al codi.

Per començar, importem els paquets[4] de tkinter i random, que el random es fa servir per generar les lletres aleatòriament. I després de tkinter importem els messageboxes.

[1] Una interfície gràfica és el que veus dins d'una aplicació, (els botons, menús…)

[2] És un text que l'usuari no pot alterar.

[3] És una "caixa" on l'usuari pot alterar el text.

[4] Un paquet és una col·lecció de fitxers i directoris necessaris per a una finalitat de programari

```
1 import tkinter as tk
2 import random
3 from tkinter import messagebox
```

Listing 6.1: Importació de paquets

Després inicio una classe on estarà tot el contingut i de l'app. Just a l'inici d'aquesta classe especifico el títol de la finestra on es mostra l'app a més a més de les dimensions que tindrà i el logo que es mostrarà.

```
1 class MemorizeApp:
2     def __init__(self, root):
3         self.root = root
4         self.root.title("Memo App by Cub de Rubik")
5         self.root.geometry("500x500")
6         self.root.iconbitmap("Brain.ico")
```

Listing 6.2: Inici de la classe i especificació de la finestra

Després inicio una classe on estarà tot el contingut i de l'app. Just a l'inici d'aquesta classe especifico el títol de la finestra on es mostra l'app a més a més de les dimensions que tindrà i el logo que es mostrarà. Just a sota del codi anterior declaro tots els objectes que estaran presents a l'app, labels que diuen el temps, textbox en la qual l'usuari omple amb un número i botons d'inici i tornar a iniciar el procediment. Com a referència els objectes comencen amb un self. I la seva continuació és el nom d'objecte que és.

```
1         self.letters = self.generate_letters()
2
3         self.label = tk.Label(self.root, text="", font=("Arial", 24))
4         self.label.pack(pady=20)
5
6         self.time_entry = tk.Entry(self.root, font=("Arial", 14))
7         self.time_entry.insert(0, "Write time in seconds")
8         self.time_entry.pack(pady=10)
9
10        self.start_button = tk.Button(self.root, text="Start",
```

```
            command=self.start_memorize)
11          self.start_button.pack(pady=5)
12
13          self.answer_entry = tk.Entry(self.root, font=("Arial", 14))
14          self.answer_entry.pack(pady=10)
15
16          self.submit_button = tk.Button(self.root, text="Submit",
                command=self.check_answers)
17          self.submit_button.pack(pady=5)
18
19          self.reset_button = tk.Button(self.root, text="Reset",
                command=self.reset)
20          self.reset_button.pack(pady=5)
21
22          self.answer_entry.config(state="disabled")
23          self.submit_button.config(state="disabled")
24          self.reset_button.config(state="disabled")
25
26          self.name = tk.Label(self.root, text="Made␣by␣Cub␣de␣Rubik",
                font=("Arial", 8))
27          self.name.pack(pady=20)
```

Listing 6.3: Declaració d'objectes necessaris pel funcionament de l'App

Després començo a escriure les funcions, en aquest cas començo amb la funció de generar les lletres gràcies al paquet random. Les funcions a Python comencen amb el def i el nom de la funció, aquesta funció inicia una llista on s'escriuran les lletres generades. A sota on posa for i in range[20], significa que està creant un bucle 20 vegades on es genera una lletra i s'escriu a la llista.

```
1       def generate_letters(self):
2           letters = []
3           for _ in range(20):
4               letter = chr(random.randint(65, 90))
```

```
            letters.append(letter)
        return letters
```

Listing 6.4: Funció per generar lletres

Les següents funcions són les de començar a memoritzar, aquestes funcions són les que s'atribueixen al botó d'inici de la memorització. El que fan és desactivar el botó start un cop clicat i afegir la llista de lletres dins del label perquè l'usuari pugui començar a memoritzar. Abans d'això la funció ha agafat el temps que havia introduït l'usuari i el converteix a mil·lisegons perquè el programa mostri les lletres la quantitat de temps desitjada a més a més d'amagar les lletres a l'usuari.

```
    def start_memorize(self):
        self.start_button.config(state="disabled")
        self.label.config(text="".join(self.letters))
        time_delay = int(self.time_entry.get()) * 1000  # Convert
           user seconds to milliseconds
        self.root.after(time_delay, self.show_entry)
        self.time_entry.pack_forget()

    def show_entry(self):
        self.label.config(text="")
        self.answer_entry.config(state="normal")
        self.submit_button.config(state="normal")
        self.reset_button.config(state="normal")
        self.answer_entry.focus_set()
```

Listing 6.5: Funcions pel botó d'inici

La següent funció és la de comprovar la resposta que has introduït, quan fas clic en el botó d'enviar la resposta, el que fa és dir-te si la resposta és correcta o incorrecta. I activar el botó de reset per poder tornar a començar.

```
    def check_answers(self):
        user_answers = self.answer_entry.get().upper()
        correct_answers = "".join(self.letters)

```

```
        if user_answers == correct_answers:
            messagebox.showinfo("Result", "Correct answers!")
        else:
            messagebox.showinfo("Result", "Incorrect answers. Try
                again.")

    self.reset()
```

Listing 6.6: Funció per comprovar la resposta

L'última funció és la de tornar a començar, la qual et torna a generar les lletres i torna a començar el compte enrere, de manera resumida, torna a l'estat on li havies fet clic en el botó d'inici.

```
    def check_answers(self):
        user_answers = self.answer_entry.get().upper()
        correct_answers = "".join(self.letters)

        if user_answers == correct_answers:
            messagebox.showinfo("Result", "Correct answers!")
        else:
            messagebox.showinfo("Result", "Incorrect answers. Try
                again.")

        self.reset()
```

Listing 6.7: Funció per tornar a començar

Finalment, tenim la part més important del codi que és el que crea la finestra principal i s'executa el bucle principal (mainloop) per mantenir l'aplicació en funcionament.

```
if __name__ == "__main__":
    root = tk.Tk()
    app = MemorizeApp(root)
    root.mainloop()
```

Listing 6.8: Bucle per mantenir l'aplicació en funcionament

El codi complet es pot veure a l'annex 1 i el resultat del codi executat en les diferents fases es pot veure a les figures 6.1, 6.2, 6.3. [6] [7] [9] [17]

Figura 6.1: Fase Incial

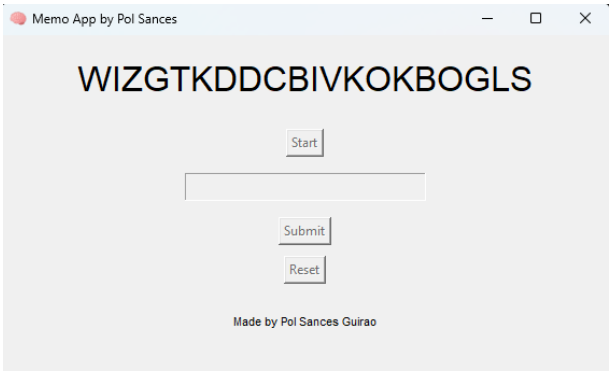

Figura 6.2: Fase de Memorització

Figura 6.3: Fase d'introducció de la resposta

Capítol 7

Elaboració d'un PDF tutorial per a principiants

Ja amb tots els conceptes entesos i la creació de l'app he fet un tutorial per resoldre el cub de Rubik a cegues per a totes les persones que tinguin un domini mínim del cub, és a dir, gent que ja sap fer el cub. En comparativa amb el contingut del treball, té algunes semblances, ja que tot està explicat de manera simple i conté algunes coses del treball però també inclou mètodes inicials i altres conceptes que no he introduït al treball perquè a l'hora de fer el cub sense mirar són essencials i, en canvi, a l'hora d'entendre el treball no.[11] [15] [25]

Si hagués de definir el tutorial amb poques paraules, seria que el contingut està explicat com a mi m'hauria agradat que m'ho haguessin explicat. El resultat del tutorial es pot veure a l'annex 2, encara que no està complet ja que quedaria un annex massa llarg al treball, per veure el tutorial complet he d'anar roadto3bld/tutorial. Dins de la web hi ha les taules de memorització de les 576 combinacions de dues lletres que es poden fer al 3BLD i és per això que és millor que el tutorial es visualitzi a la web.

Capítol 8

Elaboració d'una web

Per deixar-ho tot organitzat he decidit fer una web estàtica[1]. La idea darrere de la web és fer com un tipus de central en la qual et porti als diferents contiguts del treball, en resum està feta perquè serveixi com a tutorial a qualsevol que tingui una mica d'experiència en els cubs i es vulgui inciar en el blind. Està redactada en HTML5 i CSS3, no és una web molt complexa, però ha sigut desenvolupada sense gairebé experiència prèvia, només aprenent amb tutorials i llibres d'html es pot visitar a roadto3bld.

Dins de la web hi ha la pàgina principal on hi ha una presentació del contingut de la web, després tenim un menú que porta a les pestanyes de Tutorial, App i Contacte. [8]

[1] Una pàgina web estàtica és una pàgina web que es mostra al navegador de l'usuari tal i com està emmagatzemada

Capítol 9

Piera Open 2023

Aquest estiu vaig decidir anar a Piera a una competició de cubs de Rubik oficial, regulada per la WCA[1]. Aquesta competició va ser la meva primera competició oficial, on vaig participar en diferents categories, però prioritzant el 3BLD.

Les resolucions de 3BLD no van tenir èxit, ja que per aquell moment no tenia confiança i vaig fer DNF[2] a les tres, es pot veure una de les resolucions a youtube/tdrpolsances. Tot i no tenir èxit, m'emporto l'experiència i fer aquest TdR m'ha motivat més a poder anar a una altra competició i fer una resolució vàlida a 3BLD. Els resultats es poden veure al meu perfil de la WCA, el qual no puc compartir per la política de les bases, on es pot comprovar que no tenia prèvies competicions realitzades.

Figura 9.1: Jo al recinte del Piera Open

[1] Sigles de World Cube Asociation

[2] "Did Not Finish" (No acabat)

Capítol 10

Resolució del Cub

Amb tot aquest procés d'aprenentatge he sigut capaç de resoldre el cub de Rubik a cegues. A la competició de Piera encara no estava del tot preparat, però ara que ja he assimilat correctament tots els coneixements he aconseguit resoldre'l en la competició *online* (no-oficial) the cubers.io en 3:40.78 que és un temps bastant raonable.

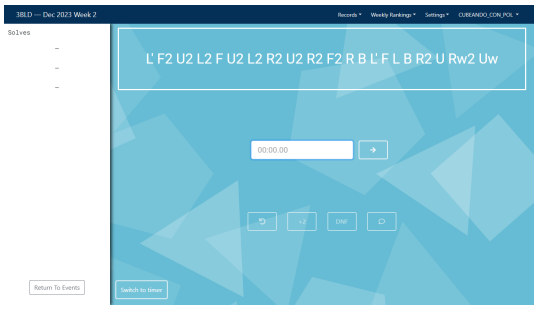

Figura 10.1: Pantalla de barreja de cubers.io

Es pot veue un video on faig el cub a cegues a youtube.com/TDRPolSances.

Conclusions

L'objectiu principal del treball era realitzar un tutorial complet per realitzar el cub de Rubik a cegues per a tothom, i ha estat assolit completament, ja que el tutorial que he realitzat conté conceptes que van des del inicis amb el cub, la notació fins a mètodes intermedis que s'utilitzen a competicions oficials. No només s'ha realitzat un tutorial, sinó que s'ha creat el que podria ser una "plataforma" per aprendre a fer el cub, ja que la web serveix de concentrador del tutorial i l'aplicació per ajudar a memoritzar.

Els objectius personals del treball, també han sigut assolits, ja que he aconseguit aprendre a fer el cub de Rubik de sense mirar i també he aprofundit en l'àmbit de la programació. He realitzat una aplicació amb Python que té interfície gràfica, cosa que mai m'havia passat pel cap, amb HTML he realitzat una web estàtica, que no és la web més complexa, però jo no comptava amb cap coneixement d'HTML. Finalment, la redacció del document fet amb LaTeX, és juntament amb el fet de fer el cub a cegues, el més important del treball.

Durant el treball, he tingut molts problemes, sobretot amb la programació, ja que a l'hora d'aprendre el procés és prova i error. Amb el cub també he tingut alguns problemes però no tants com amb la programació.

Part III

Annexos

Annex 1 Codi de l'App

El codi de l'app tot junt és el següent.

```
    # Importing packages

import tkinter as tk
import random
from tkinter import messagebox

# Begin and personalize window

class MemorizeApp:
    def __init__(self, root):
        self.root = root
        self.root.chapter("Memo App by Cub de Rubik")
        self.root.geometry("500x500")
        self.root.iconbitmap("Brain.ico")

    # Creating all the assets
        self.letters = self.generate_letters()

        self.label = tk.Label(self.root, text="", font=("Arial", 24))
        self.label.pack(pady=20)

        self.time_entry = tk.Entry(self.root, font=("Arial", 14))
        self.time_entry.insert(0, "Write time in seconds")
```

```
24          self.time_entry.pack(pady=10)
25
26          self.start_button = tk.Button(self.root, text="Start",
                command=self.start_memorize)
27          self.start_button.pack(pady=5)
28
29          self.answer_entry = tk.Entry(self.root, font=("Arial", 14))
30          self.answer_entry.pack(pady=10)
31
32          self.submit_button = tk.Button(self.root, text="Submit",
                command=self.check_answers)
33          self.submit_button.pack(pady=5)
34
35          self.reset_button = tk.Button(self.root, text="Reset",
                command=self.reset)
36          self.reset_button.pack(pady=5)
37
38          self.answer_entry.config(state="disabled")
39          self.submit_button.config(state="disabled")
40          self.reset_button.config(state="disabled")
41
42          self.name = tk.Label(self.root, text="Made␣by␣Cub␣de␣Rubik",
                font=("Arial", 8))
43          self.name.pack(pady=20)
44
45      # Function to generate letters
46
47      def generate_letters(self):
48          letters = []
49          for _ in range(20):
50              letter = chr(random.randint(65, 90))
51              letters.append(letter)
```

```
52          return letters
53
54      # Function to show letters with the time that user put in the
            time_entry textbox.
55
56      def start_memorize(self):
57          self.start_button.config(state="disabled")
58          self.label.config(text="".join(self.letters))
59          time_delay = int(self.time_entry.get()) * 1000  # Convert
                user seconds to milliseconds
60          self.root.after(time_delay, self.show_entry)
61          self.time_entry.pack_forget()
62
63      # Function to hide letters
64
65      def show_entry(self):
66          self.label.config(text="")
67          self.answer_entry.config(state="normal")
68          self.submit_button.config(state="normal")
69          self.reset_button.config(state="normal")
70          self.answer_entry.focus_set()
71
72      # Function to check awnsers
73
74      def check_answers(self):
75          user_answers = self.answer_entry.get().upper()
76          correct_answers = "".join(self.letters)
77
78          if user_answers == correct_answers:
79              messagebox.showinfo("Result", "Correct␣answers!")
80          else:
81              messagebox.showinfo("Result", "Incorrect␣answers.␣Try␣
```

```
                again.")
82
83          self.reset()
84
85      # Reset function
86
87      def reset(self):
88          self.letters = self.generate_letters()
89          self.answer_entry.delete(0, tk.END)
90          self.label.config(text="")
91          self.start_button.config(state="normal")
92          self.answer_entry.config(state="disabled")
93          self.submit_button.config(state="disabled")
94          self.reset_button.config(state="disabled")
95
96
97  if __name__ == "__main__":
98      root = tk.Tk()
99      app = MemorizeApp(root)
100     root.mainloop()
```

Listing 10.1: Codi sencer de l'App

Per veure el codi de l'App es pot veure a: roadto3bld/App.

Annex 2 Tutorial 3BLD

1.1 Coses a saber abans de començar

1.1.1 Notació dels Moviments

El cub de rubik es resol gràcies a identificar patrons i executar algoritmes que resolen aquests patrons, aquests algoritmes han d'estar escrits en alguna part per poder-los memoritzar i per això estaà la notació del cub de rubik.

La notació consta de 6 moviments (F,B,R,L,U,D), que correspon a (Front, Back, Right, Left, Up, Down) que son les respectives direccions en anglés. Per exemple si faig el moviment F gira la cara front la que està més propera a la nostra visió, en sentit horari, en canvi si fós F' seria antihorari. En les figures següents es mostra una respresentació gràfica per a cada capa.

És un concepte difícil d'entendre però de manera simplificada és girar la cara en sentit horari i antihorari desde la cara que vulguis. En les figures següents es mostra una respresentació gràfica per a cada capa.

Figura 1.2: Exemples de Movimients F y F'

1.1.2 Interpretar el concepte del cub

La manera correcta d'interpretar el cub és pensar en el funcionament, com si el desmuntessis, ja que consta de 12 arestes i 8 cantonades, a més a més dels 6 centres que no poden

Figura 1.3: Exemples de Movimients B y B’

Figura 1.4: Exemples de Movimients R y R’

Figura 1.5: Exemples de Movimients L y L’

Figura 1.6: Exemples de Movimients U y U’

Figura 1.7: Exemples de Movimients D y D'

permutar[1] amb cap altra peça ja que només roten.

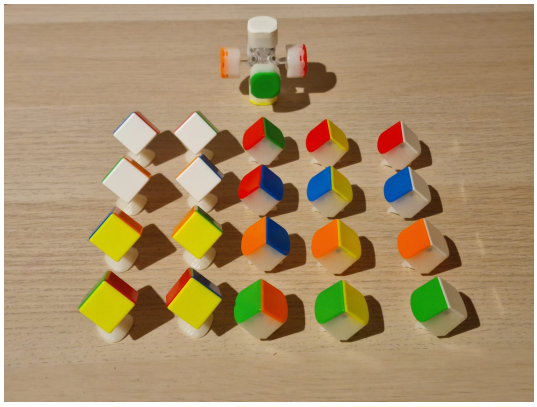

Figura 1.8: Cub Desmuntat

[1] Intercanvi de posició amb una altre peça i de l'ordre de tot el conjunt

1.2 El Concepte de 3BLD

Per començar cal entendre el funcionament d'una resolució de blind, primer el cub és barrejat per una persona i el posa dins d'una capsa o un cube cover[2], després es col·loca a la taula boca avall i la persona que l'ha de resoldre es pren el seu temps per respirar. Un cop fet això la persona que resol el cub encén el timer i destapa el cub, de manera que el temps comença a comptar i es comença a memoritzar. Un cop acabada la memorització el que resol el cub es tapa els ulls amb un antifaç i comença a resoldre el cub, mentre que una persona externa li posa una cartiluna entre el cub i la seva cara per evitar trampes i mirar per sota de l'antifaç. Tots aquests passos s'han d'executar perfectament per asegurar-se de la resolució compti.

Figura 1.9: Materials necessaris per poder executar blind

1.2.1 Fases de la Resolució

Com ja he esmentat a la seccio anterior, completar el cub de Rubik amb els ulls tancats, es divideix en dos grans fases, memorització i execució. I dins d'aquestes fases hi han diferents procediments per poder aconseguir fer-ho correctament.

[2]Un cube cover és una tapa per cubs feta de cartró i que s'utlitza a les competicions

1.2.2 Memorització

En aquesta fase com ja ho diu el seu nom has de memoritzar el cub. La manera de memoritzar no és la més convencional ja que no memoritzes color sinó peces i com que hem d'interpretar el cub com si fossin 20 peces el que fem es donar-li una lletra per la qual pots identificar a cada peça. Fent aquestes conversions has d'arribar a tenir el teu propi esquema de lletres, el que utilitzo jo és el de la figura 1.10, i el pots agafar com a plantilla per fer el teu o directament utilitzar-lo ja que si t'acostumes no et limitarà res durant el procés.

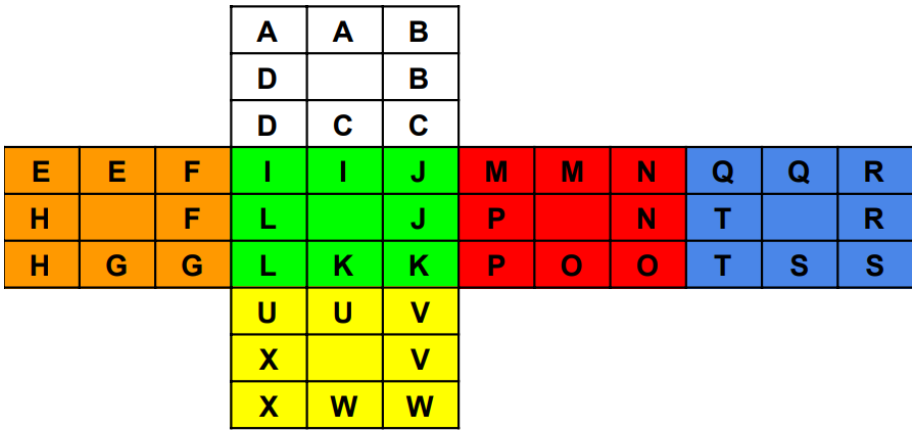

Figura 1.10: Esquema de Lletres

Hi han lletres repetides perquè primer memoritzes arestes i després cantonades, aquestes lletres a l'hora de memoritzar-les, no les memoritzes a força bruta, sinó que les converteixes en paraules que puguis convertir en imatges per poder entrecordar-te. Sembla molt complex però no ho és com es pot veure en el següent exemple. Per practicar aquesta transformació d'imatges pots anar a la pàgina web roadto3bld on podràs veure el codi de la app i on hi haurà un link a github on podràs descarregar i utilitzar l'app.

Haig de memoritzar les lletres R i B $\rightarrow$ RedBull

Haig de memoritzar les lletres A i C $\rightarrow$ Aire Acondicionat (AC és el símbol)

Llavors has de tenir per a cada parell de lletres una paraula clau per memoritzarles ràpidament. Et pots inspirar en les d'algú, com les meves de les taules al final del document, però aquestes si que no les hauries de copiar, ja que aquestes paraules han de ser les que primer et vinguin al cap pensant en les dues lletres, és un treball una mica pesat però amb el temps

dona el seu fruit. No hi ha secret per integrar aquestes paraules a la teva ment, el que has de fer és practicar, practicar i practicar, i amb el temps et sortirà sol.

1.2.3 Execució

A diferència de una resolució normal, a l'hora d'executar els moviments tu no pots rotar el cub, perquè tens que actuar segons les lletres que has memoritzat i si rotes el cub canvies la posició de les peces respecte a les lletres indirectament. Per tant només fas moviments de les capes. Els algoritmes per intercanviar les peces només interfereixen en les lletres que has memoritzat i deixent la resta del cub exactament igual quan fas els passos. Per començar a fer el cub has d'aprendre el mètode principants que explicaré a la següent secció. Durant l'execució se segueix l'ordre CEEC que diu que memoritzes cantonades, després arestes, executes arestes i després executes cantonades.

1.3 El mètode principiants ”Old Pochmann”

El métode principiants o Old Pochmann és un mètode pel qual intercanvies les peces que vols una a una, és a dir, que mires la peça que primer has memoritzat i la canvies per la peça que està al seu lloc i així ja tens una peça que està al seu lloc i una altra que has de col·locar, i així succesivament. Aquesta peça sobre la que sempre estàs treballant es diu ”Buffer”.El mètode és el mateix concepte tant per arestes com cantonades però el dividirem en dos perquè l’explicació sigui més fàcil.

1.3.1 Arestes

En les arestes el buffer serà la peça BM, (l’aresta blanca-vermella), i a l’hora de memoritzar començarem des d’allà. Un cop memoritzat un ”setup move”, que consisteix en un moviment que posa la peça que vols intercanviar en el lloc AD, (l’aresta blanca-taronja) i seguit del setup move farem l’algoritme d’intercanvi del mètode, després desfarem el setup movem executant-lo però a revés.

Algortime d’intercanvi $\rightarrow$ R U R’ U’ R’ F R2 U’ R’ U’ R U R’ F’

Els setups moves més òptims per cada peça de les arestes són els següents:

A	Lw2 D’ L2
B	buffer
C	Lw2 D L2
D	Fer algoritme
E	L Dw’ L
F	Dw’ L
G	L’ Dw’ L
H	Dw L’
I	Lw D’ L2
J	Dw2 L
K	Lw D L2
L	L’

M	buffer
N	Dw L
O	D2 L’ Dw’ L
P	Dw’ L’
Q	Lw’ D L2
R	L
S	Lw’ D’ L2
T	Dw2 L’
U	D’ L2
V	D2 L2
W	D L2
X	L2

Taula 1.1: Setup Moves Arestes

1.3.2 Cantonades

Per les cantonades és la mateix història (setup move →algortime d'intercanvi →setup move), el que canvia és l'algoritme d'intercanvi i que la posició buffer és la AER (cantonada blanca-taronja-blava) i la posició d'intercanvi per la qual fem els setup moves és la VKP (cantonada groga-verda-vermella).

Algortime d'intercanvi → F R U' R' U' R U R' F' R U R' U' R' F R F'

Els setups moves més òptims per cada peça de les cantonades són els següents:

A	buffer
B	R2
C	F2 D
D	F2
E	buffer
F	F' D
G	F'
H	D' R
I	F R'
J	R'
K	F' R'
L	F2 R'

M	F
N	R' F
O	R2 F
P	R F
Q	R D'
R	buffer
S	D F'
T	R
U	D
V	Fer algoritme
W	D'
X	D2

Taula 1.2: Setup Moves Cantonades

1.3.3 Casos Especials

Hi han dos tipus de casos especials que et poden sorgir, un és quan estàs memoritzant i et trobes la peça del buffer, en aquest cas memoritzes una altra peça que no hagis memoritzat i continues com normal perquè automàticament al final et quedara bé. El segonc as especial és quan has acabat de memoritzar i et queden nombres imparells d'arestes i cantonades memoritzades, llavors el que has de fer, és, entre l'execució de les arestes i les cantondes fer l'algoritme de partitat aquest cas i contrinuar la resolució de manera normal.

Algortime de Paritat → R U R' F' R U2 R' U2 R' F R U R U2 R' U'

1.3.4 Exemple de Memorització per a aquest mètode

Barreja: → F2 B R2 U' L2 U2 B' L' F2 U' B2 U L2 U R2 F2 L2 U' L2 F

Memorització Arestes → VP QD CK ER SG HT N

Memorització Cantonades → UN HV TM C

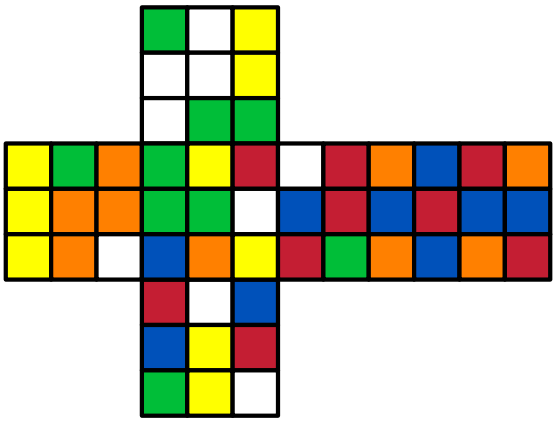

Figura 1.11: Cub barrejat per exemple de Memorització

1.4 Mètode Intermig (M2/Orozco)

El mètode intermig per resoldre el cub a cegues consta del mètode M2 per les arestes i l'Orozco per les cantonades. El mètode M2, que com ja ho diu el seu nom es basa en el moviment M2, que és el moviment de la capa del mig del cub 2 vegades, a més a més, el buffer és UK (Aresta)

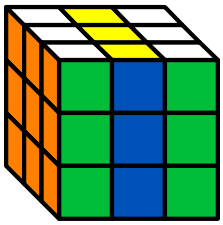

Figura 1.12: Exemple de Moviment M2

Aquest mètode intercanvia les peces d'una manera peculiar, ja que ha de fer dues vegades M2 per tornar a l'estat original i canviar dues peces. Per exemple, fas primer la seqüència Y que col·loca la peça al lloc d'intercanvi, després fas la seqüència X que en aquesta cas és M2 i després fas Y' per retornar la peça intercanviada. Un cop fet això s'han canviat dues peces però el cub no queda igual que abans perquè hem de tornar a fer un intercanvi, aquest segon intercanvi ha de ser amb una seqüència Z X Z' perquè hem d'intercanviar una peça que no sigui la mateixa.

Exemple d'intercanvi d'arestes L i V:

Figura 1.13: Secuencia Y (Esquerra) i Y X (Dreta)

Figura 1.14: Secuencia Y X Y' (Esquerra) i Y X Y'Z (Dreta)


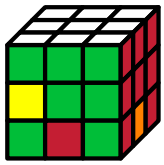

Figura 1.15: Secuencia Y X Y'Z'X (Esquerra) i Y X Y'Z X Z' (Dreta)

Llavors així és com es canviarien dues arestes amb el mètode M2, en aquest cas L i V. El mètode té els casos de la taula 1.3.

A	M2
B	R' U R U' M2 U R' U' R
C	U2 M' U2 M'
D	L U' L' U M2 U' L U L'
E	B L' B' M2 B L B'
F	B L2 B' M2 B L2 B'
G	B L B' M2 B L' B'
H	L B L' B' M2 B L B' L'
I	D M' U R2 U' M U R2 U' D' M2
J	U R U' M2 U R' U'
K	Posició d'intercanvi
L	U' L' U M2 U' L U

M	B' R B M2 B' R' B
N	R' B' R B M2 B' R' B R
O	B' R' B M2 B' R B
P	B' R2 B M2 B' R2 B
Q	U B' R U' B (M2) B' U R' B U'
R	U' L U M2 U' L' U
S	M2' D U R2 U' M' U R2 U' M D'
T	U R' U' M2 U R U'
U	Posició d'intercanvi
V	U R2 U' M2 U R2 U'
W	M U2 M U2
X	U' L2 U M2 U' L2 U

Taula 1.3: Casos del Mètode M2

1.4.1 Mètode d'execució per les Cantonades

El mètode orozco, utlitza un sistema similar al M2, ja que fa les seqüències Y X Y'X' i Z A Z' A' però en aquest cas la seqüència Z és diferent perquè es troba al segon lloc. De manera simple, si a la memorització tens la lletra en segon lloc has de fer l'algoritme alternatiu. Els casos d'orozco són els de la taula 1.4

A la columna de l'esquerra hi ha els corresponents a la primera lletra del parell i a la dreta els corresponents a la segona lletra del parell. *A perm* és un cas del cub de rubik normal que es fa (R' U' R' D' R U' R' D R U R' D' R U R' D R2). Cal tambe tenir en compte que els algoritmes estan una notació diferents perquè es vegi de millor forma al ser tants algoritmes. Funciona de

AB	Basic A Perm	**BA**	Reverse A Perm
DB	U' (A Perm) U	**BD**	U' (Reverse A Perm) U
EB	[R: [R D R', U]]	**BE**	[R: [U, R D R']]
FB	[R': [U', R' D' R]]	**BF**	[R': [R' D' R, U']]
GB	[U, R' D R]	**BG**	[R' D R, U]
HB	[R D' R', U']	**BH**	[U', R D' R']
IB	[R: [R D R', U2]]	**BI**	[R: [U2, R D R']]
KB	[D': [U, R' D R]]	**BK**	[D': [R' D R, U]]
LB	[D: [U, R' D' R]]	**BL**	[D: [R' D' R, U]]
OB	[R D R', U']	**BO**	[U', R D R']
PB	[U, R' D' R]	**BP**	[R' D' R, U]
RB	[R': [U2, R' D' R]]	**BR**	[R': [R' D' R, U2]]
SB	[U, R' D2 R]	**BS**	[R' D2 R, U]
TB	[D: [R D' R', U']]	**BT**	[D: [U', R D' R']]
UB	[x': [R U R', D2]]	**BU**	[x': [D2, R U R']]
VB	[D' x': [R U R', D2]]	**BV**	[D' x': [D2, R U R']]
WB	[D x: [D2, R' U' R]]	**BW**	[D x: [R' U' R, D2]]
XB	[x: [D2, R' U' R]]	**BX**	[x: [R' U' R, D2]]

Taula 1.4: Algorimtes orozco

la següent manera [A,B] = A,B,A',B'.

NB	[U, R' D R D' R' D' R]	**BN**	[R' D R D' R' D' R, U]
QB	[R' D R D' R' D' R, U]	**BQ**	[U, R' D R D' R' D' R]

Taula 1.5: Excepcions del mètode

Aquests algoritmes estan escrits en una notació[3] diferenti [Y,X]. El fet que estigui entre claudators indica que s'ha de fer en l'ordre Y X Y' X'. És una mica més díficil de visualitzar però més fàcil a l'hora d'aplicar aquest mètode. Exepmle d'intercanvi de dues cantonades P i H

Com a orientació [Y,X] i [Z,A] són a la taula d'algoritmes PB i BH.

[3]Manera d'escriure els algoritmes

Figura 1.16: Secuencia Y (Esquerra) i Y X (Dreta)

Figura 1.17: Secuencia Y X Y' (Esquerra) i Y X Y' X'(Dreta)

Figura 1.18: Secuencia Y X Y' X' Z (Esquerra) i Y X Y' X' Z A(Dreta)

Figura 1.19: Secuencia Y X Y' X' Z A Z' (Esquerra) i Y X Y' X' Z A Z' A'(Dreta)

Fins aquí el tutorial de 3BLD, espero que hagi sigut de fàcil comprensió i que t'hagi ajudat a resoldre el cub a cegues a les següents pàgines trobaràs les taules de memorització i dels algoritmes d'orozco i M2.

Bibliografia

[1] Allen, S. (2017). *Memoriza como Sherlock Holmes / Aprende la técnica del palacio de la memoria*. CreateSpace Independent Publishing Platform.

[2] Cstimer. (2023). Aplicació Cstimer. https://cstimer.net/

[3] Cubedesk. (2023). Aplicació Cube desk. https://www.cubedesk.io/

[4] Cuber, R. (2023). Orozco Tutorial. https://www.reddit.com/r/Cubers/comments/84r39g/orozco_intermediate_3bld_method_tutorial/

[5] Elle, A. (2023). *Técnicas de Memoria Veloz*. CREATESPACE.

[6] Freecodecamp. (2023). Free code camp. https://www.freecodecamp.org/espanol/news/guia-de-funciones-de-python-con-ejemplos/

[7] GeeksforGeeks. (2023). Tkinter. https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/

[8] GreatStack. (2023). Ajuda web. https://www.youtube.com/watch?v=PgAZ8KzfhO8

[9] Icon-icons. (2023). Logo Cervell. https://icon-icons.com/es/icono/cerebro/109577

[10] Insider. (2018). How The Rubik's Cube Became One Of The Bestselling Toys In History. https://youtu.be/Y5tyCcEVsyY

[11] JavaCuber. (2022). I spent 6 months practicing 3x3 blindfolded. https://www.youtube.com/watch?v=B3vM7ejx584

[12] JPerm. (2017). Rubik's cube blindfolded: M2 method tutorial. https://www.youtube.com/watch?v=JTkSQxWk_u8

[13] JPerm. (2023). Orozco Method. https://www.youtube.com/watch?v=LTRYeFsao9Q

[14] KDE. (2023). David Singmaster. https://docs.kde.org/trunk5/en/kubrick/kubrick/singmaster-moves.html

[15] Klise, A. (2023). Andy Klise's M2/Old Pochmann Guide. https://www.kungfoomanchu.com/guides/andy-klise-m2-oldpoch.pdf

[16] LaTeX, L. (2023). Lliçó 01. https://www.learnlatex.org/ca/lesson-01

[17] RecursosPython. (2023). Recursos Python. https://recursospython.com/guias-y-manuales/iconos-en-ventanas-de-tk-tkinter/

[18] Redbull. (2023). Cubo de Rubik: 10 curiosidades que no sabías. https://www.redbull.com/es-es/cubo-rubik-10-curiosidades

[19] Ross, D. (2017). *Rubik's Cube Best Algorithms*. Independently published.

[20] Stackexchange, T. (2023). Rubik's Cube Tikz. https://tex.stackexchange.com/questions/459254/easy-way-to-generate-rubiks-cube-diagrams

[21] Stackoverflow. (2023). Header amb Fancy. https://stackoverflow.com/questions/68669076/how-to-correctly-set-a-fancy-header-and-footer-in-a-included-pdf-which-contains

[22] Ted. (2023). Xerrada Rubik's Cube. https://www.ted.com/talks/chris_olson_how_a_rubik_s_cube_solved_my_life

[23] WCA. (2023). Rankings 3BLD | World Cube Association. https://www.worldcubeassociation.org/results/rankings/333bf/single

[24] Yan, R. (2019). CubeRoot; Ruimin. https://www.cuberoot.me/3bld/

[25] Youtube. (2023). MAPA de Progressió d'un excampió mundial. https://www.youtube.com/watch?v=yv_hNlR5Qg4