



# **Road to 3BLD**

**Institut Lluís Companys**

**2n Batxillerat**

**Autor: Pol Sances Guirao**

**Tordera, Gener de 2024**

# 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.

# Índex

<b>Introducció</b>	<b>3</b>
<b>1 Història del Cub de Rubik</b>	<b>4</b>
1.1 Invent i Introducció (1974-1980)	4
1.2 Primers Intents de Resolució (1980-1981):	4
1.3 Aparició dels Primers Campions (1982-1992):	4
1.4 L'Època dels Speedcubers (2003-2010):	4
1.5 L'Arribada de 3BLD (2004):	4
1.6 Actualment(2023):	4
<b>2 Entrant al concepte del Cub de Rubik</b>	<b>6</b>
2.1 Interpretar el concepte del cub	6
2.2 Aplicar les matemàtiques al concepte	6
<b>3 Notació dels Moviments</b>	<b>8</b>
<b>4 El Concepte de 3BLD</b>	<b>9</b>
4.1 Fases de la Resolució	10
4.2 Memorització	10
4.3 Execució	13
4.4 El mètode d'execució per les arestes.	14
4.5 Mètode d'execució per les Cantonades	16
<b>5 Redacció d'aquest treball amb <math>\text{\LaTeX}</math></b>	<b>18</b>
5.1 Les Figures del Document	18
<b>6 Elaboració d'una App per Practicar la Memorització</b>	<b>20</b>
<b>7 Elaboració d'un PDF tutorial per a principiants</b>	<b>25</b>
<b>8 Elaboració d'una web</b>	<b>25</b>
<b>Bibliografia</b>	<b>27</b>



# Marc Teòric

## 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, aquest trencaclosques es va estendre per tot el món, però la seva resolució semblava un enigma que estava a l'abast de poca gent.

### 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.

### 1.3 Aparició dels Primers Campions (1982-1992):

A la dècada de 1980 van tenir lloc les primeres competicions de Cub de Rubik, 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 Fridrich de Jessica Fridrich 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<sup>1</sup> 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. Utilitzant ja algorismes específics.

### 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. Acu-

<sup>1</sup> Persones que poden fer el cub de Rubik en molt poc temps

talment 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.

## 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 Redbull, 2023.

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 trobar la solució bé 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.

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

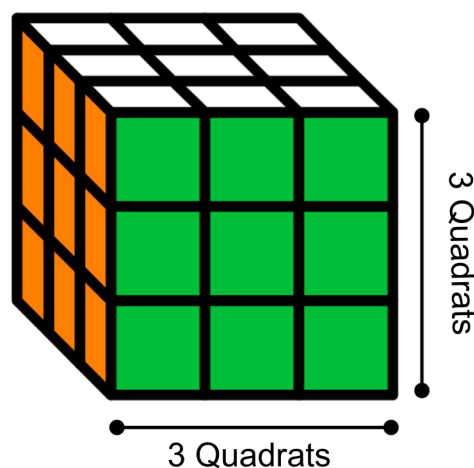


Figura 1: Plantejament típic però erroni del cub de rubik

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<sup>2</sup> amb cap altra peça ja que només roten.

### 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 divid el càlcul el dos grups, per una part tenim cantonades

<sup>2</sup>Intercanvi de posició amb una altra peça i de l'ordre de tot el conjunt

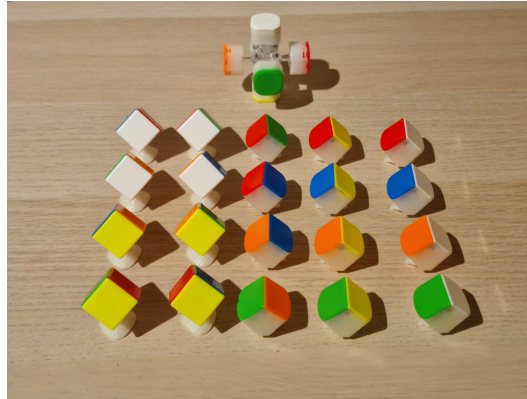


Figura 2: Cub Desmuntat

i per l'altra arestes. Per la part de les cantonades es calcula:

Tenim 8 cantonades que es poden posar de manera aleatoria en els 8 llocs, i això es calcula com a  $8!$ <sup>3</sup>, després aquestes es poden orientar en 3 direccions diferents que matemàticament és  $3^8$ . Per tant les combinacions tèoriques possibles amb un cub de només cantonades són:

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

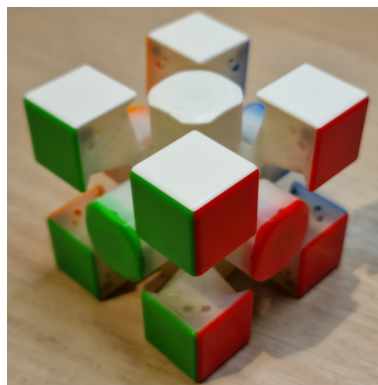


Figura 3: Cub amb només cantonades

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 dos orientacions ho multiplicarem per  $2^{12}$ . Per tant les combinacions tèoriques possibles amb un cub de només arestes són:

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

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

<sup>3</sup>! és el símbol de factorial o  $8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$



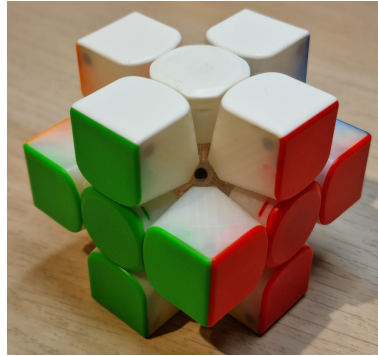


Figura 4: Cub amb només arestes

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

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 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.

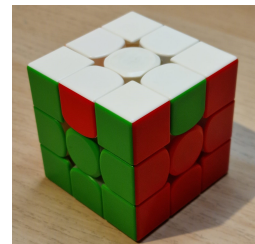
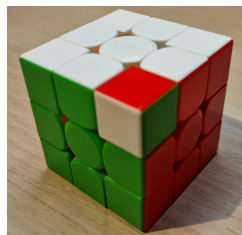


Figura 5: Exemple de casos impossibles

I tot ben calculat queda:

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

### 3 Notació dels Moviments

El cub de rubik es resol gràcies a identificar patrons i executar algorismes que resolen aquests patrons, aquests algorismes 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 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 representació gràfica per a cada capa.

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



Figura 6: Exemples de Moviments F y F'



Figura 7: Exemples de Moviments B y B'



Figura 8: Exemples de Moviments R y R'

## 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<sup>4</sup>, 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 antifàs 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'antifàs. Tots aquests passos s'han d'executar perfectament per assegurar-se de la resolució compti.

<sup>4</sup>Un cube cover és una tapa per cubs feta de cartró i que s'utilitza a les competicions

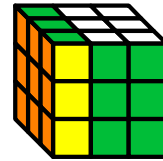
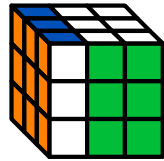


Figura 9: Exemples de Moviments L y L'

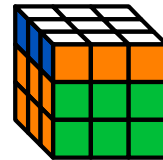
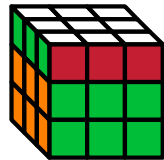


Figura 10: Exemples de Moviments U y U'

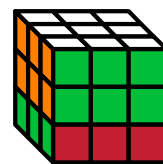
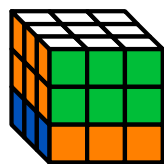


Figura 11: Exemples de Moviments D y D'

En la imatge 12 es poden veure el timer<sup>5</sup>, l'antifaç, la caixa per cobrir el cub, que en aquest cas jo utilizo una que tinc d'un cub, i a més a més uns cascos d'obra per aïllar-te 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 dos grans fases, memorització i execució. I dins d'aquestes fases hi han 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'enrecordo 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 es 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<sup>6</sup> que utilizo es veu a la figura 13.

<sup>5</sup>El timer és el comptador amb la forma de les mans que es veu al centre de la imatge

<sup>6</sup>És la distribució de lletres



Figura 13: Esquema de Lletres

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

De manera pràctica es comença a memortizar desde 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 posció 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 una concepte difícil d'explicar amb paraules i es veu millor al següent exemple. Cal destacar que al començar a memortizar 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

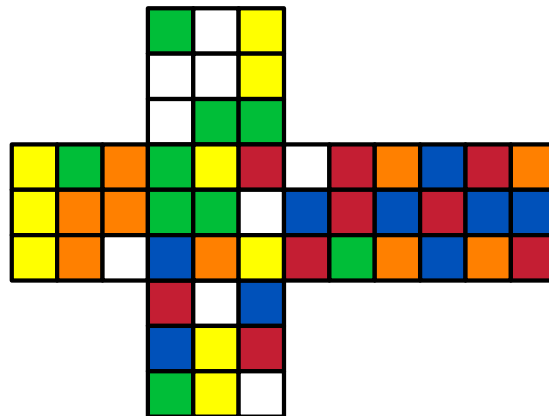


Figura 14: Cub barrejat per exemple de Blind

Començem mirant all 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 succesivament 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	PeiX
DF	DoFí
Ri	Rlu

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 a 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 algorismes<sup>7</sup>.

Com que el cub no ha de rotar, les peces també s'han de quedar en el mateix lloc a l'acabar l'algorisme, és a dir, és executar un algorisme, mous les dues peces que vols i acabes amb la resta del cub igual però amb les dues peces canviades.

Hi han diferents nivells d'eficiència d'algorismes, i es perquè amb els algorismes 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 algorisme 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 es intercanviar les peces entre elles, després fem la seqüència Y' que és la seqüència y 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:

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

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

<sup>7</sup> És una seqüència de moviments que permet realitzar una tasca

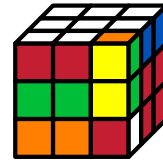
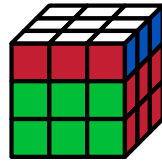


Figura 15: Secuencia Y (Esquerra) i Y X (Dreta)

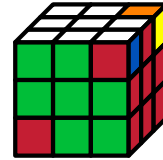
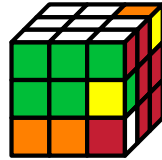


Figura 16: Secuencia Y X Y' (Esquerra) i Y X Y' X' (Dreta)

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

Aquesta explicació s'atribueix al mètode més avançat, que no és el que jo utilitzo, ja que el mètode més avançat són 400 algorismes 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<sup>8</sup>.

Jo utilitzo el mètode de dificultat mitja-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.

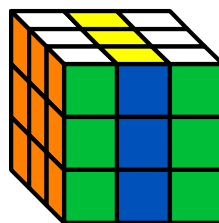


Figura 17: Exemple de Moviment M2

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

<sup>8</sup>És la memòria que fa que nosaltres movem els músculs d'una manera quan pensem una cosa, com quan escrivim a l'ordinador.

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:

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	$U' L2 U M2 U' L2 U$

Exemple d'intercanvi d'arestes L i V:

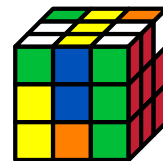
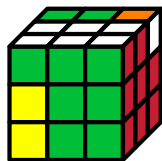


Figura 18: Secuencia Y (Esquerra) i Y X (Dreta)

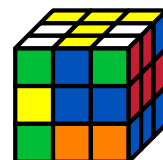
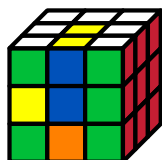


Figura 19: Secuencia Y X Y' (Esquerra) i Y X Y' Z (Dreta)

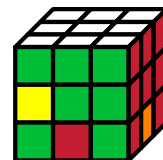
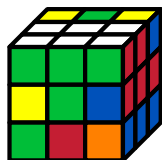


Figura 20: Secuencia Y X Y' Z' X (Esquerra) i Y X Y' Z' X Z' (Dreta)

Lavors així és com es canviarien dues arestes amb el mètode M2, en aquest cas L i V.



## 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:

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

Taula 1: Algoritmes 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 ( $R' U' R' D' R U' R' D R U' R' D' R U' R' D R2$ ).

<b>NB</b>	$[U, R' D R D' R' D' R]$	<b>BN</b>	$[R' D R D' R' D' R, U]$
<b>QB</b>	$[R' D R D' R' D' R, U]$	<b>BQ</b>	$[U, R' D R D' R' D' R]$

Taula 2: Excepcions del mètode

Aquests algoritmes estan escrits en una notació<sup>9</sup> diferent  $[Y, X]$ . El fet que estigui entre claudators

<sup>9</sup>Manera d'escriure els algoritmes

indica que s'ha de fer en l'ordre  $Y X Y' X'$ . És una mica més difícil de visualitzar però més fàcil a l'hora d'aplicar aquest mètode. Exemple d'intercanvi de dues cantonades P i H



Figura 21: Secuencia Y (Esquerra) i Y X (Dreta)



Figura 22: Secuencia Y X Y' (Esquerra) i Y X Y' X' (Dreta)



Figura 23: Secuencia Y X Y' X' Z (Esquerra) i Y X Y' X' Z A (Dreta)



Figura 24: 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.

# Marc Pràctic

## 5 Redacció d'aquest treball amb $\text{LaTeX}$

Tot aquest treball escrit està redactat en LaTeX, que és un sistema de composició textos orientat a la creació de documents professional, 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 1 i la figura 25

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

Listing 1: Exemple de Secció i Subsecció a LaTeX

### 1 Secció de Prova

Text de prova.

#### 1.1 Subsecció de Prova

Figura 25: Exemple de Secció i Subsecció a LaTeX

A més a més he estat mantenint un control de versions a Github que es pot veure a [PolSances13/TDR](https://github.com/PolSances13/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, utilitzen 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 insertats. El codi per formar un cub es pot veure al listing 2 i el resultat a la figura 26, 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 2: Exemple de Cubs fets amb el paquet rubikcube

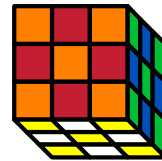
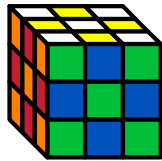


Figura 26: Cubs de demostració fets amb el paquet rubikcube

## 6 Elaboració d'una App per Practicar la Memorització

Al 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 el 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<sup>10</sup>.

L'app consta de diferents labels<sup>11</sup>, botons i textboxes<sup>12</sup>, que es mostren o 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 estel·lisc són comentaris i no afecten al codi.

Per començar importem els paquets<sup>13</sup> de tkinter i random, que el random es fa servir per generar les lletres aleatòriament. I després de tkinter importem els messageboxes.

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

Listing 3: Importació de paquets

Després incio 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("MemoApp by Pol Sances")
5         self.root.geometry("500x500")
6         self.root.iconbitmap("Brain.ico")
```

Listing 4: Inici de la classe i especificació de la finestra

<sup>10</sup>Una interfície gràfica és el que veus dins d'una aplicació, (els botons, menús...)

<sup>11</sup>És un text que l'usuari no pot alterar.

<sup>12</sup>És una "caixa" on l'usuari pot alterar el text.

<sup>13</sup>Un paquet és una col·lecció de fitxers i directoris necessaris per a una finalitat de programari

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'inci i tornar a inciar 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_Pol_Sances_Guirao",
        font=("Arial", 8))
27        self.name.pack(pady=20)
```

Listing 5: 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 es s'escriu a la llista.

```

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

```

Listing 6: Funció per generar lletres

La següent funcions són les de començar a memoritzar, aquestes funció són les que s'atribueixen al botó d'inici de la memorització. El que fan es 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 milisegons perquè el programa mostri les lletres la quantitat de temps desitjada a més a més d'amagar les lletres a l'usuari.

```

1      def start_memorize(self):
2          self.start_button.config(state="disabled")
3          self.label.config(text="".join(self.letters))
4          time_delay = int(self.time_entry.get()) * 1000 # Convert user
                    seconds to milliseconds
5          self.root.after(time_delay, self.show_entry)
6          self.time_entry.pack_forget()
7
8      def show_entry(self):
9          self.label.config(text="")
10         self.answer_entry.config(state="normal")
11         self.submit_button.config(state="normal")
12         self.reset_button.config(state="normal")
13         self.answer_entry.focus_set()

```

Listing 7: Funcions pel botó d'inici

La següent funció és la de comprovar la resposta que has introduït, quan li dones al 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.

```

1      def check_answers(self):
2          user_answers = self.answer_entry.get().upper()

```

```
3         correct_answers = "".join(self.letters)
4
5         if user_answers == correct_answers:
6             messagebox.showinfo("Result", "Correct_answers!")
7         else:
8             messagebox.showinfo("Result", "Incorrect_answers. Try again.")
9
10        self.reset()
```

Listing 8: Funció per comprovar la resposta

L'última funció és la de tornar 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 donat al botó d'inici.

```
1        def check_answers(self):
2            user_answers = self.answer_entry.get().upper()
3            correct_answers = "".join(self.letters)
4
5            if user_answers == correct_answers:
6                messagebox.showinfo("Result", "Correct_answers!")
7            else:
8                messagebox.showinfo("Result", "Incorrect_answers. Try again.")
9
10           self.reset()
```

Listing 9: 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.

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

Listing 10: 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 27, 28, 29.





Figura 27: Fase Inicial

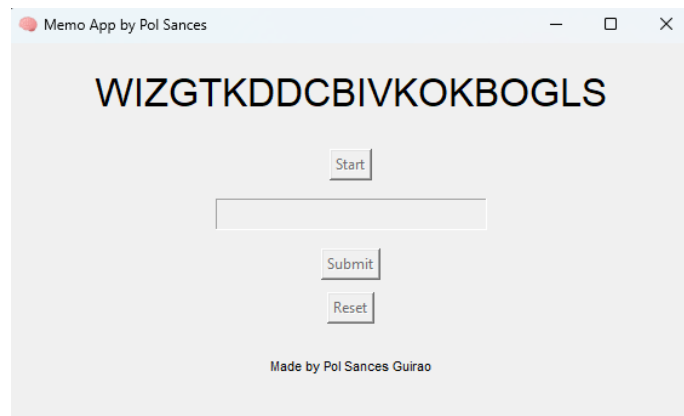


Figura 28: Fase de Memorització



Figura 29: Fase d'introducció de la resposta

## 7 Elaboració d'un PDF tutorial per a principiants

Ja amb tot 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 incials 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.

Si hagués de definir el tutorial amb poques paraules, seria que el contingut està explicat com a mi m'hagués agradat que m'ho haguessin explicat. El resultat del tutorial es pot veure a l'annex 2.

## 8 Elaboració d'una web

Per deixar-ho tot organitzat he decidit fer un web estàtica<sup>14</sup>. La idea darrere de la web és fer com un tipus de central en la qual et porti als diferents continguts 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 iniciar 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.

---

<sup>14</sup>Una pàgina web estàtica és una pàgina web que es mostra al navegador de l'usuari tal i com està emmagatzemada

## Conclusions

Encara no puc donar unes conclusions ja que el treball no està del tot completat.

## Bibliografia

- 3BLD. (2019). CubeRoot; Ruimin. <https://www.cuberoot.me/3bld/>
- Allen, S. (2017). *Memoriza como Sherlock Holmes / Aprende la técnica del palacio de la memoria*. CreateSpace Independent Publishing Platform.
- Cstimer. (2023). Aplicació Cstimer. <https://cstimer.net/>
- Cubedesk. (2023). Aplicació Cube desk. <https://www.cubedesk.io/>
- Cuber, J. [ (2022). I spent 6 months practicing 3x3 blindfolded. <https://www.youtube.com/watch?v=B3vM7ejx584>
- Docs, G. (2023). Orozco Tutorial. [https://docs.google.com/document/d/1sPvzowIU1M6PV\\_CQhQnf6RthclGAMJEvf5wXN/edit](https://docs.google.com/document/d/1sPvzowIU1M6PV_CQhQnf6RthclGAMJEvf5wXN/edit)
- Elle, A. (2023). *Técnicas de Memoria Veloz*. CREATSPACE.
- Freecodecamp. (2023). Free code camp. <https://www.freecodecamp.org/espanol/news/guia-de-funciones-de-python-con-ejemplos/>
- GeeksforGeeks. (2023). Tkinter. <https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/>
- Icon-icons. (2023). Logo Cerebro. <https://icon-icons.com/es/icono/cerebro/109577>
- JPerm. (2017). Rubik's cube blindfolded: M2 method tutorial. [https://www.youtube.com/watch?v=JTSQxWk\\_u8](https://www.youtube.com/watch?v=JTSQxWk_u8)
- KDE. (2023). David Singmaster. <https://docs.kde.org/trunk5/en/kubrick/kubrick/singmaster-moves.html>
- Klise, A. (2023). Andy Klise's M2/Old Pochmann Guide. <https://www.kungfoomanchu.com/guides/andy-klise-m2-oldpoch.pdf>
- LaTeX, L. (2023). *Lección 01*. <https://www.learnlatex.org/ca/lesson-01>
- Matplotlib. (2023). Llibreria de Generació de Gràfiques amb python. <https://matplotlib.org/stable/tutorials/index.html>
- RecursosPython. (2023). Recursos Python. <https://recursospython.com/guias-y-manuales/iconos-en-ventanas-de-tk-tkinter/>
- Redbull. (2023). Cubo de Rubik: 10 curiosidades que no sabías. <https://www.redbull.com/es-es/cubo-rubik-10-curiosidades>
- Ross, D. (2017). *Rubik's Cube Best Algorithms*. Independently published.
- Stackexchange, T. (2023). Rubik's Cube Tikz. <https://tex.stackexchange.com/questions/459254/easy-way-to-generate-rubiks-cube-diagrams>
- 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>
- Ted. (2023). Xerrada Rubik's Cube. [https://www.ted.com/talks/chris\\_olson\\_how\\_a\\_rubik\\_s\\_cube\\_solved\\_my\\_life](https://www.ted.com/talks/chris_olson_how_a_rubik_s_cube_solved_my_life)
- Youtube. (2023a). Ajuda web. <https://www.youtube.com/watch?v=PgAZ8KzfhO8>

Youtube. (2023b). MAPA de Progressió d'un excampió mundial. [https://www.youtube.com/watch?v=yv\\_hNIR5Qg4](https://www.youtube.com/watch?v=yv_hNIR5Qg4)

Youtube. (2023c). Orozco Method. <https://www.youtube.com/watch?v=LTRYeFsao9Q>

# Annexos

## Annex 1 Codi de l'App

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

```
1      # Importing packages
2
3  import tkinter as tk
4  import random
5  from tkinter import messagebox
6
7  # Begin and personalize window
8
9  class MemorizeApp:
10     def __init__(self, root):
11         self.root = root
12         self.root.title("Memo App by Pol Sances")
13         self.root.geometry("500x500")
14         self.root.iconbitmap("Brain.ico")
15
16     # Creating all the assets
17         self.letters = self.generate_letters()
18
19         self.label = tk.Label(self.root, text="", font=("Arial", 24))
20         self.label.pack(pady=20)
21
22         self.time_entry = tk.Entry(self.root, font=("Arial", 14))
23         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=
33         self.check_answers)
34     self.submit_button.pack(pady=5)
35
36     self.reset_button = tk.Button(self.root, text="Reset", command=self
37         .reset)
38     self.reset_button.pack(pady=5)
39
40     self.answer_entry.config(state="disabled")
41     self.submit_button.config(state="disabled")
42     self.reset_button.config(state="disabled")
43
44     self.name = tk.Label(self.root, text="Made_by_Pol_Sances_Guirao",
45         font=("Arial", 8))
46     self.name.pack(pady=20)
47
48     # Function to generate letters
49
50     def generate_letters(self):
51         letters = []
52         for _ in range(20):
53             letter = chr(random.randint(65, 90))
54             letters.append(letter)
55         return letters
56
57     # Function to show letters with the time that user put in the
58     time_entry textbox.
59
60     def start_memorize(self):
61         self.start_button.config(state="disabled")
62         self.label.config(text="".join(self.letters))
63         time_delay = int(self.time_entry.get()) * 1000 # Convert user
64             seconds to milliseconds
65         self.root.after(time_delay, self.show_entry)
66         self.time_entry.pack_forget()
67

```

```

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)

```







## Índex

<b>1 Coses a saber abans de començar</b>	<b>4</b>
1.1 Notació dels Moviments . . . . .	4
1.2 Interpretar el concepte del cub . . . . .	5
<b>2 El Concepte de 3BLD</b>	<b>6</b>
2.1 Fases de la Resolució . . . . .	6
2.2 Memorització . . . . .	6
2.3 Execució . . . . .	7
<b>3 El mètode principiants "Old Pochmann"</b>	<b>8</b>
3.1 Arestes . . . . .	8
3.2 Cantonades . . . . .	8
3.3 Casos Especials . . . . .	9
3.4 Exemple de Memorització per a aquest mètode . . . . .	9
<b>4 Mètode Intermig (M2/Orozco)</b>	<b>10</b>
4.1 Mètode d'execució per les Cantonades . . . . .	11

Aquest tutorial està pensat per a persones que ja tenen un domini mínim del cub, es recomana que les persones sapiguen fer el cub de Rubik ja que facilita molt el procés.





Figura 4: Exemples de Moviments L y L'



Figura 5: Exemples de Moviments U y U'



Figura 6: Exemples de Moviments D y D'

## 1.2 Interpretar el concepto 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 permutar<sup>1</sup> amb cap altra peça ja que només roten.

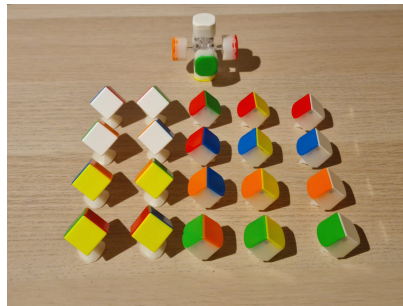


Figura 7: Cub Desmuntat

<sup>1</sup>Intercanvi de posició amb una altra peça i de l'ordre de tot el conjunt

---

Pàg 38 / 45





### 3 El mètode principiants "Old Pochmann"

El mètode 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 successivament. 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.

#### 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'algorisme d'intercanvi del mètode, després desfarem el setup move executant-lo però a revés.

Algorisme d'intercanvi  $\rightarrow R U R' U' R' F R^2 U' R' U' R U R' F'$

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

A	Lw2 D' L2	M	buffer
B	buffer	N	Dw L
C	Lw2 D L2	O	D2 L' Dw' L
D	Fer algorisme	P	Dw' L'
E	L Dw' L	Q	Lw' D L2
F	Dw' L	R	L
G	L' Dw' L	S	Lw' D' L2
H	Dw L'	T	Dw2 L'
I	Lw D' L2	U	D' L2
J	Dw2 L	V	D2 L2
K	Lw D L2	W	D L2
L	L'	X	L2

Taula 1: Setup Moves Arestes

#### 3.2 Cantonades

Per les cantonades és la mateixa història (setup move  $\rightarrow$  algorisme d'intercanvi  $\rightarrow$  setup move), el que canvia és l'algorisme 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).



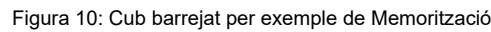


Figura 11: Exemple de Moviment M2

Figura 12: Secuencia Y (Esquerra) i Y X (Dreta)



Figura 13: Sequencia  $Y X Y'$  (Esquerra) i  $Y X Y'Z$  (Dreta)



Figura 14: Sequencia 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 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 3: Casos del Mètode M2

#### 4.1 Mètode d'execució per les Cantonades

El mètode orozco, 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 de la taula 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 ( $R' U' R' D' R U'$

<b>AB</b>	Basic A Perm
<b>DB</b>	$U'$ (A Perm) $U$
<b>EB</b>	$[R: [R D R', U]]$
<b>FB</b>	$[R': [U', R' D' R]]$
<b>GB</b>	$[U, R' D R]$
<b>HB</b>	$[R D' R', U']$
<b>IB</b>	$[R: [R D R', U_2]]$
<b>KB</b>	$[D': [U, R' D R]]$
<b>LB</b>	$[D: [U, R' D' R]]$
<b>OB</b>	$[R D R', U']$
<b>PB</b>	$[U, R' D' R]$
<b>RB</b>	$[R': [U_2, R' D' R]]$
<b>SB</b>	$[U, R' D_2 R]$
<b>TB</b>	$[D: [R D' R', U']]$
<b>UB</b>	$[x': [R U R', D_2]]$
<b>VB</b>	$[D' x': [R U R', D_2]]$
<b>WB</b>	$[D x: [D_2, R' U' R]]$
<b>XB</b>	$[x: [D_2, R' U' R]]$
<b>BA</b>	Reverse A Perm
<b>BD</b>	$U'$ (Reverse A Perm) $U$
<b>BE</b>	$[R: [U, R D R']]$
<b>BF</b>	$[R': [R' D' R, U']]$
<b>BG</b>	$[R' D R, U]$
<b>BH</b>	$[U', R D' R']$
<b>BI</b>	$[R: [U_2, R D R']]$
<b>BK</b>	$[D': [R' D R, U]]$
<b>BL</b>	$[D: [R' D' R, U]]$
<b>BO</b>	$[U', R D R']$
<b>BP</b>	$[R' D' R, U]$
<b>BR</b>	$[R': [R' D' R, U_2]]$
<b>BS</b>	$[R' D_2 R, U]$
<b>BT</b>	$[D: [U', R D' R']]$
<b>BU</b>	$[x': [D_2, R U R']]$
<b>BV</b>	$[D' x': [D_2, R U R']]$
<b>BW</b>	$[D x: [R' U' R, D_2]]$
<b>BX</b>	$[x: [R' U' R, D_2]]$

Taula 4: Algorimtes orozco

R' D R U R' D' R U R' D R2). Cal també tenir en compte que els algorismes estan una notació diferents perquè es vegi de millor forma al ser tants algorismes. Funciona de la següent manera  $[A,B] = A.B,A'.B'$ .

<b>NB</b>	[U, R' D R D' R' D' R]
<b>QB</b>	[R' D R D' R' D' R, U]

<b>BN</b>	[R' D R D' R' D' R, U]
<b>BQ</b>	[U, R' D R D' R' D' R]

Taula 5: Excepcions del mètode

Aquests algorismes estan escrits en una notació<sup>3</sup> diferent  $[Y,X]$ . El fet que estigui entre claudators indica que s'ha de fer en l'ordre  $Y\ X\ Y'\ X'$ . És una mica més difícil de visualitzar però més fàcil a l'hora d'aplicar aquest mètode. Exemple d'intercanvi de dues cantonades  $P$  i  $H$

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

### <sup>3</sup>Manera d'escriure els algoritmes



Figura 15: Secuencia Y (Esquerra) i Y X (Dreta)



Figura 16: Secuencia Y X Y' (Esquerra) i Y X Y' X'(Dreta)



Figura 17: Sequencia  $Y X Y' X' Z$  (Esquerra) i  $Y X Y' X' Z A$  (Dreta)



Figura 18: Sequencia  $Y X Y' X' Z A Z'$  (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.