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

Introducció

Vaig descobrir el cub de Rubik desde ben petit però no va ser fins als 12 anys que el vaig aprendre a resoldre, després de la primera vegada ja no podia parar, va ser una connexió amb el cub de manera molt forta i que em generava una addicció molt gran a seguir resolent-lo i millorar els meus temps. Mës tard vaig aprendre a fer diferents tipus de cubs, i em va entrar la curiositat de fer-lo sense mirar. Vaig intentar aprendre però vaig fallar, i no només un cop va ser unes quantes vegades.

Així aquest treball busca com objectiu personal aprendre a fer el cub de Rubik sense mirar, i com a objectiu del treball, realitzar un tutorial complet amb tot tipus de recursos que puguin ajudar a qualsevol a entendre a fer el cub de rubik sense mirar.

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

Per realitzar aquest treball necessitaré un ordinador amb connexío a internet i el meu cub 3x3.

1 Marc Teòric

1.1 Història del Cub de Rubik

1.1.1 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.

1.1.2 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.

1.1.3 Aparició dels Primers Campions (1982-1992):

La dècada de 1980 van tenir lloc les primeres competicions de CUb de Rubik, gràcies a la popularitat quee 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 Jessica Fridrich es va convertir en un dels més populars.

1.1.4 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.

1.1.5 L'Arribada de 3BLD (2004):

La disciplina de resolució a cegues (3BLD) es va introduir més tard a la WCA i Tyson Mao es va convertir en el primer campió de 3BLD el 2004. Utilitzant ja algoritmes específics.

1.1.6 Avui en Dia (2023 en Endavant):

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. Acutalment s'estan buscant tècniques encara més avançades pel 3BLD, com algoritmes gairebé el doble

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

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.

1.2 Entrant al concepte del Cub de Rubik

1.2.1 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 [**?**].

Aquesta xifra es sol atribuir a la dificultat del cub, però després d'aprendre a fer el cub de rubik t'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.

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

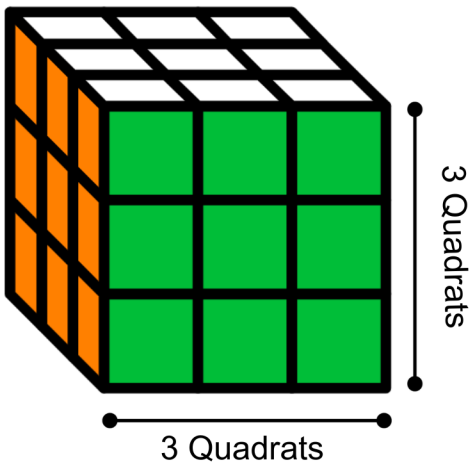

Figura 1: Plantejament típic però erroni del cub de rubik

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[2] amb cap altra peça ja que només roten.

1.2.2 Aplicar les matemàtiques al concepte

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

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

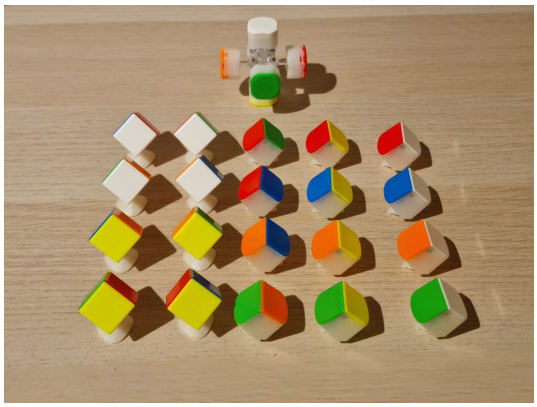

Figura 2: Cub Desmuntat

Tenim 8 cantonades que es poden posar de manera aleatoria en els 8 llocs, i això es calcula com a $8!$ [3], 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:

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

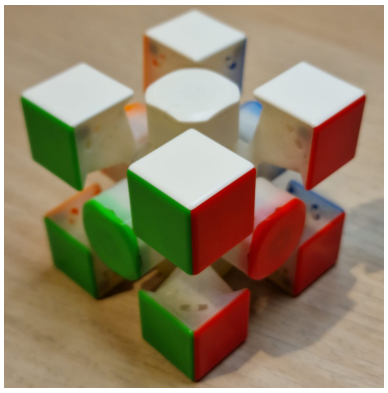

Figura 3: Cub amb només cantonades

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:

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

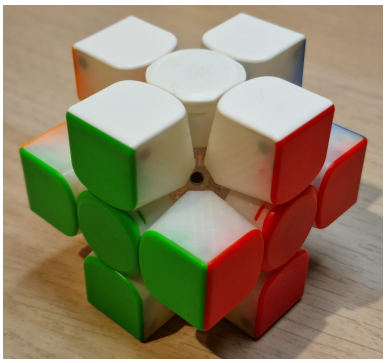

Figura 4: Cub amb només arestes

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

Però cal dir que aquestes no són les combinacions totals reals del cub de rubik ja que aquestes combinacions estan calculades com si demuntessim 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.

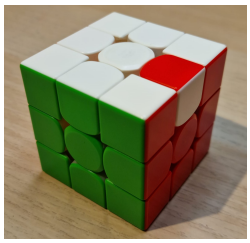
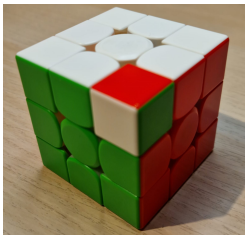
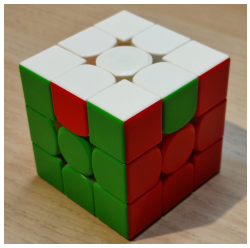

Figura 5: Exemple de casos impossibles

I tot ben calculat queda:

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

1.3 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 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 capa front la que eta més propera a mi 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 anti-horari desde la cara que vulguis. En les figures següents es mostra una respresentació gràfica per a cada capa.

Figura 6: Exemples de Movimients F y F'

Figura 7: Exemples de Movimients B y B'

Figura 8: Exemples de Movimients R y R'

1.4 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[4], 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 perr asegurar-se de la resolució compti.

En la imatge anteriori es poden veure el timer[5], 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]Un cube cover és una tapa per cubs feta de cartró i que s'utlitza a les competicions
[5]El timer és el compatdor amb la forma de les mans que es veu al centre de la imatge


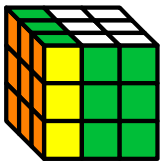

Figura 9: Exemples de Movimients L y L'


Figura 10: Exemples de Movimients U y U'

1.5 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.

1.5.1 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 peçes del cub, que "només" són 20 en lletres i ho fem d'una manera distribuida en ordre que nosaltres ens memoritzem. L'esquema de lletres[6] que utilizo és el que es veu a la següent figura.

Com es pot veure hi han lletres repetides i això és degut a 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 que no és gaire eficient memoritar les lletres una per una i és per això

[6] És la distribució de lletres


Figura 11: Exemples de Movimients D y D'

Figura 12: Materials necessaris per poder executar blind

			A	A	B						
			D		B						
			D	C	C						
E	E	F	I	I	J	M	M	N	Q	Q	R
H		F	L		J	P		N	T		R
H	G	G	L	K	K	P	O	O	T	S	S
			U	U	V						
			X		V						
			X	W	W						

Figura 13: Esquema de LLetres

que la manera correcta de fer-ho és:

Memoritzar dos lletres i amb aquestes dos lletres formar una paraula de la qual puguis pensar en una imatge en la que et puguis enrecordar. En resum, és converir parells de lletres en en imatges, per tant ara tenim la meitat d'ítems a memoritzar. Un exemple d'aquesta fusío 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 maner pràctica es comença a memortizar desde la peça UK mirant el color de U, de la lletra en la posicó U que és la inicial i treus una lletra i llavors mires a la posció on hi 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 díficl d'explicar amb paraules i es veu millor al següent exemple. Cal destacar que al començar a memortizar la posció UK la saps perquè poses 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

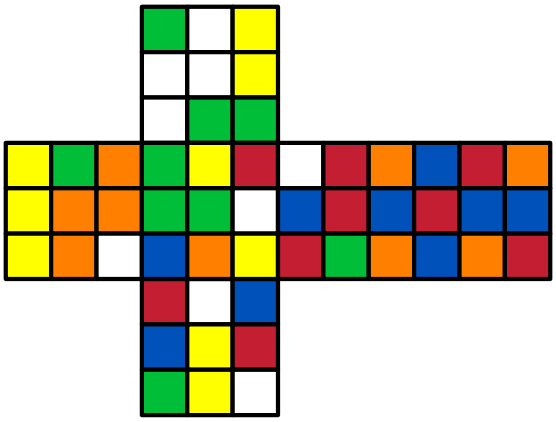

Figura 14: Cub barrejat per exemple de Blind

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 mirem 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

1.5.2 Execució

L'execució és la segona fase per completar el cub a cegues i es fa d'una maner molt diferent a la que es fa el cub de manera convencional. Quan tu resols el cub mirant tu 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

Lletres	Transcripció
DW	DeU (pronunciació)
LA	Los Ángeles
BV	BBVA
PX	PeiX
DF	DoFí
Ri	RIu

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

les peces s'aconsegueixen col·locar a lloc gràcies a algoritmes[7].
Com que el cub no ha de rotar, les peces també s'han de quedar en el mateix lloc a l'acabar l'algoritme, és a dir, és executar un algoritme, 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'algoritmes, i es 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 difícl d'aprendre-se'ls. L'explicació darrere d'això és:

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

El que fem 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 en, després fem la secuencia Y' que és la secuencia y al revés, aquesta el que fa es tornar enrere el moviment fet anteriorment i preparar el retorn de la secuència Y'. FInalment amb la secuencia 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:

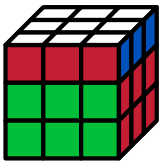


Figura 15: Secuencia Y (Esquerra) i Y X (Dreta)

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

- X = U

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


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

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

Aquesta explicació s'atribueix al mètode més avançat que ja, 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[8].

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

1.5.3 El mètode d'execució per les arestes.

Per les arestes utlitzo 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.

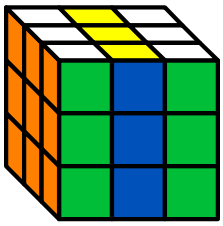

Figura 17: Exemple de Moviment M2

Aquest mètode intercanvia les peces d'una manera peculiar, ja que ha de fer dos vegades M2 per tornar a l'estat original i canviat 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. El mètode té aquests casos següents:

Exemple d'intercanvi d'arestes L i V:

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

[8] 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


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

1.6 Mètode d'execució per les Cantonades

Per les cantonades utlitzo el mètode orozco, que 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 següents:

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[9] 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

[9]Manera d'escriure els algoritmes


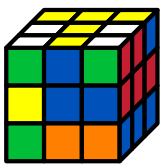

Figura 19: Secuencia Y X Y' (Esquerra) i Y X Y'Z (Dreta)


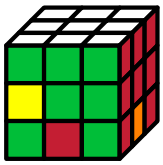

Figura 20: Secuencia Y X Y'Z'X (Esquerra) i Y X Y'Z X Z' (Dreta)

d'aplicar aquest mètode. Exepmle d'intercanvi de dues cantonades P i H


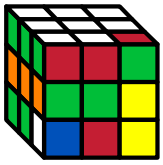

Figura 21: Secuencia Y (Esquerra) i Y X (Dreta)

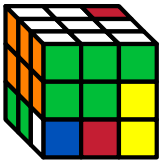
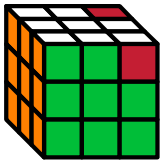

Figura 22: Secuencia Y X Y' (Esquerra) i Y X Y' X'(Dreta)


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

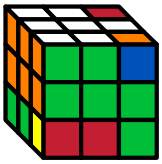
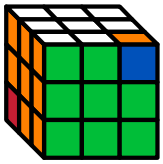

Figura 24: Secuencia Y X Y' X' Z A Z' (Esquerra) i Y X Y' X' Z A Z' A'(Dreta)

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

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 1: Algorimtes orozco

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 2: Excepcions del mètode

2 Marc Pràctic

2.1 Elaboració d'una App amb python

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[10].

L'app consta de diferents labels[11], botons i textboxes[12], que es mostren os 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 estarisc són comentaris i no afecten al codi.

Per començar importem els paquets[13] 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 1: Importació de paquets

Després incio una classe on estarà tot el contigut i de l'app. Just a l'inici d'aquesta classe específico 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␣Pol␣Sances")
5         self.root.geometry("500x500")
6         self.root.iconbitmap("Brain.ico")
```

Listing 2: Inici de la classe i especificació de la finestra

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))
```

[10]Una interfície gràfica és el que veus dins d'una aplicació, (els botons, menús...)
[11]És un text que l'usuari no pot alterar.
[12]És una "caixa" on l'usuari pot alterar el text.
[13]Un paquet és una col·lecció de fitxers i directoris necessaris per a una finalitat de programari

```
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 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 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 4: 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 introduit 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 5: Funcions pel botó d'inici

La següent funció és la de comprovar la resposta que has introduit, 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 6: 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.

```
    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 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 8: Bucle per mantenir l'aplicació en funcionament

Llavors tot el codi junt queda:

```
    # 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.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=
                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 Pol Sances Guirao",
                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 9: Python example

El control de versions és pot veure a github a l'enllaç següent Memo-App, i el resultat del codi en les diferents fases és el seguent:

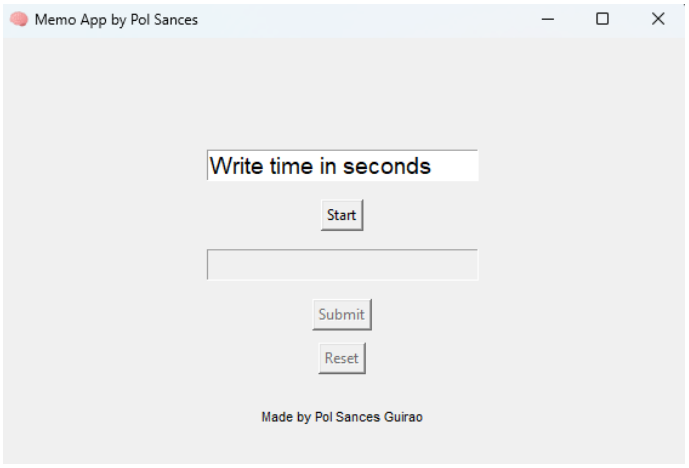

Figura 25: Fase Incial

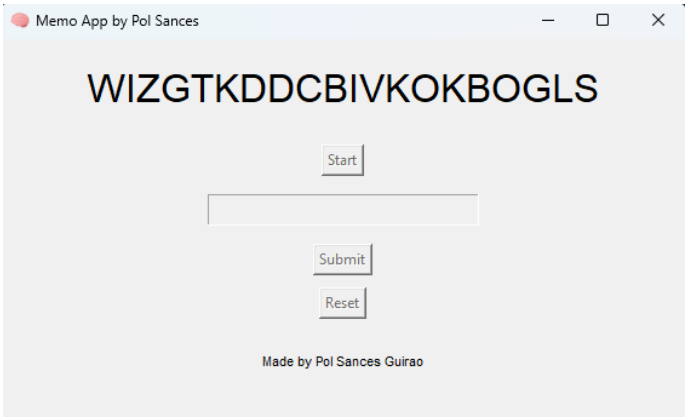

Figura 26: Fase de Memorització

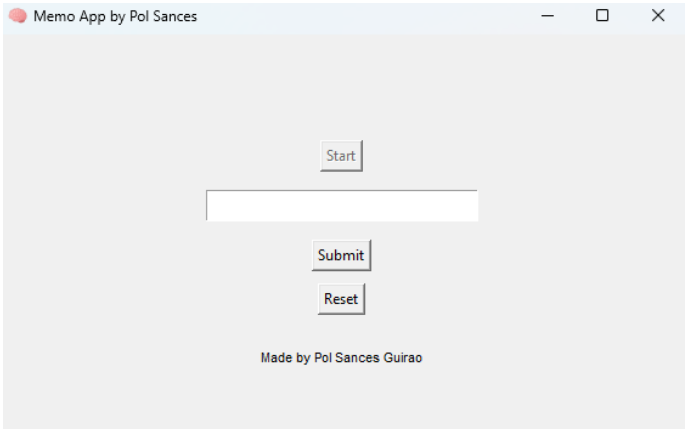

Figura 27: Fase d'introducció de la resposta

2.2 Elaboració d'una web

Per deixar-ho tot organitzat he decidit fer un web estàtica, és a dir que l'usuari no pot canviar el contigut de la web ni personalitzar-lo. 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.

La web encara no està acabada i per tant no puc completar aquesta secció.

2.3 Elaboració d'un PDF tutorial per a principiants

El PDF tutorial també està encara en procés de creació, i per tant tampoc puc escriure al respecte.

Conclusions

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

Referències

[1] Ajuda web. https://www.youtube.com/watch?v=PgAZ8KzfhO8.

[2] Aplicació cstimer. https://cstimer.net/.

[3] Aplicació cube desk. https://www.cubedesk.io/.

[4] David singmaster. https://docs.kde.org/trunk5/en/kubrick/kubrick/singmaster-moves.html.

[5] Free code camp. https://www.freecodecamp.org/espanol/news/guia-de-funciones-de-python-con-ejemplos/.

[6] Header amb fancy. https://stackoverflow.com/questions/68669076/how-to-correctly-set-a-fancy-header-and-footer-in-a-included-pdf-which-contains.

[7] Llibreria de generació de gràfiques amb python. https://matplotlib.org/stable/tutorials/index.html.

[8] Logo cervell. https://icon-icons.com/es/icono/cerebro/109577.

[9] Mapa de progressió d'un excampió mundial. https://www.youtube.com/watch?v=yv_hNlR5Qg4.

[10] Orozco method. https://www.youtube.com/watch?v=LTRYeFsao9Q.

[11] Orozco tutorial. https://docs.google.com/document/d/1sPvzowlU1M6PV_CQhQnf6RthclGAMJEvf5wXNSZJQAg/edit.

[12] Recursos python. https://recursospython.com/guias-y-manuales/iconos-en-ventanas-de-tk-tkinter/.

[13] Rubik's cube tikz. https://tex.stackexchange.com/questions/459254/easy-way-to-generate-rubiks-cube-diagrams.

[14] Tkinter. https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/.

[15] Xerrada rubik's cube. https://www.ted.com/talks/chris_olson_how_a_rubik_s_cube_solved_my_life.

[16] Cuberoot; ruimin. https://www.cuberoot.me/3bld/, febrer 2019.

[17] J. [@JavaCuber] Cuber. I spent 6 months practicing 3x3 blindfolded. https://www.youtube.com/watch?v=B3vM7ejx584, August 2022.

[18] Andy Klise. Andy klise's m2/old pochmann guide. https://www.kungfoomanchu.com/guides/andy-klise-m2-oldpoch.pdf.

[19] J. [@JPerm] Perm. Rubik's cube blindfolded: M2 method tutorial. https://www.youtube.com/watch?v=JTkSQxWk_u8, May 2017.

[20] Daniel Ross. *Rubik's Cube Best Algorithms*. Independently published, 2017.