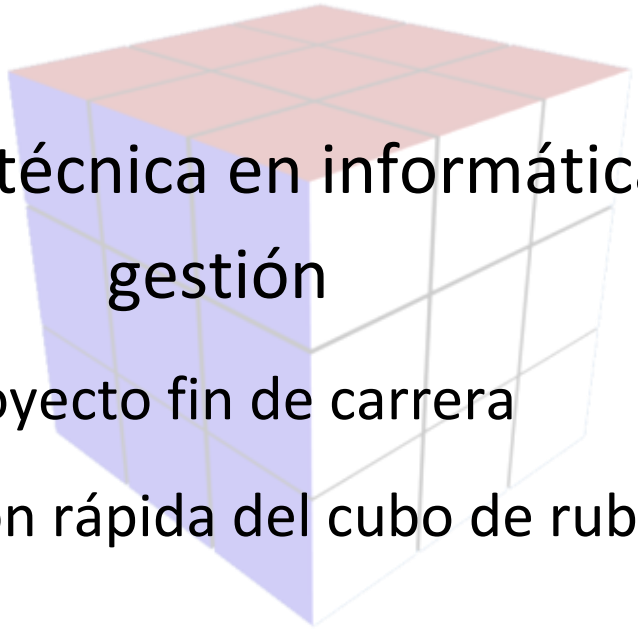

# Universidad Carlos III de Madrid

# Escuela politécnica superior



## Ingeniería técnica en informática de gestión

### Proyecto fin de carrera

### Resolución rápida del cubo de rubik

Autor: Miguel Abreu García

Tutor: Carlos Linares López

Noviembre 2011

# Índice

## Agradecimientos

Quiero aprovechar esta oportunidad para dar las gracias a todas las personas que me han ayudado todos estos años de carrera, y me han permitido llegar hasta aquí. Espero poder llegar lejos en la vida, y si lo consigo sabré que es gracias a vuestro impulso.

Primero dar gracias a mi familia, que siempre se ha esforzado mucho para poder brindarme todas las oportunidades que he necesitado. Siempre he contado con su apoyo incondicional.

También dar las gracias a mi pareja, que siempre me anima en las horas bajas, y me da su apoyo y ayuda en todas las situaciones imaginables.

Agradecer también a mis amigos y conocidos que siempre estén ahí. A los que me prestaron apuntes, a los que me ayudaron a repasar y a los que, simplemente, me dieron una palmadita cuando lo necesitaba.

También agradecer a mi tutor del proyecto, Carlos, su paciencia y entusiasmo durante todo el transcurso del proyecto. Sus ideas, su ayuda, entrega y entusiasmo han mantenido mi ilusión hacia este proyecto como el primer día.

# 1. Introducción

El cubo de rubik es uno de los puzles más extendidos del mundo, creado por Ernö Rubik en 1974. Dicho cubo alcanzó gran fama como juguete, y salieron al mercado distintas versiones del mismo, e incluso algunas que no tienen forma de cubo.

El cubo de rubik tradicional supone un desafío, puesto que tiene muchos estados y sólo uno se considera “estado final” o “solución”. También es un desafío crear una aplicación capaz de resolver el cubo.

Debido a la amplitud del problema, crear una aplicación que resuelva el cubo es una tarea complicada.

Las aplicaciones actuales buscan resolver el cubo de manera óptima, eso implica una exploración de un espacio de estados enorme, lo que consumiría mucho tiempo. Hoy día se usan distintos métodos para encontrar la solución óptima en el menor tiempo posible.

Pero hay un hecho curioso, en las competiciones oficiales del cubo de rubik, donde se trata de resolver en el menor tiempo posible, los participantes no resuelven el cubo de manera óptima, es decir, hacen más movimientos de los necesarios.

Entonces, ¿se puede resolver el cubo de rubik de manera sub-óptima, pero ganando mucho tiempo?

Siguiendo una de las metodologías que existen para resolverlo nos ahorramos la exploración de un árbol de estados muy grande dividiendo el desarrollo del cubo en etapas. Cada etapa tiene un objetivo concreto y definible, y es fácil verificar si se ha llegado al final de una etapa.

El resultado será una cadena de movimientos que resuelve un cubo dado, y dicha solución la obtendremos casi al instante.

# 2. Estado del arte

El cubo de rubik ha supuesto un reto desde que se inventó. Un puzle creado por Ernö Rubik, arquitecto, escultor y diseñador de la Escuela de Artes de Budapest. Se dice que es el juguete más vendido con más de 300 millones de unidades vendidas.

El cubo ha supuesto un reto desde que se inventó, ha sido objeto de estudio por parte de la inteligencia artificial, intentando buscar la solución óptima y creando heurísticas, bases de datos de patrones y otras técnicas que ayudaran a buscar la solución.

El enorme espacio de estados del cubo de rubik de 3x3x3 hace interesante su estudio, ya que para buscar la solución hay que explorar un árbol de estados enorme.

En julio de 2010 Tomas Rokicki, Herbert Kociemba, Morley Davidson, y John Dethridge probaron que el número máximo de movimientos que hay que realizar para resolver cualquier estado del cubo de rubik es 20. Para ello resolvieron todos los estados del cubo de rubik, con muchas máquinas muy potentes, y con técnicas que les permitían acelerar el proceso (podar el árbol de estado, eliminando estados repetidos, estados simétricos etc.…)

## 2.1. Otros puzles

Los puzles de ingenio de distintos tipos son casi tan antiguos como la humanidad misma.

### 2.1.1. Stomachion

El “Stomachion” se considera el puzle más antiguo del mundo. Similar al Tangram, el juego consistía en una serie de piezas que formaban un cuadrado. El objetivo era armar una serie de figuras usando las piezas.

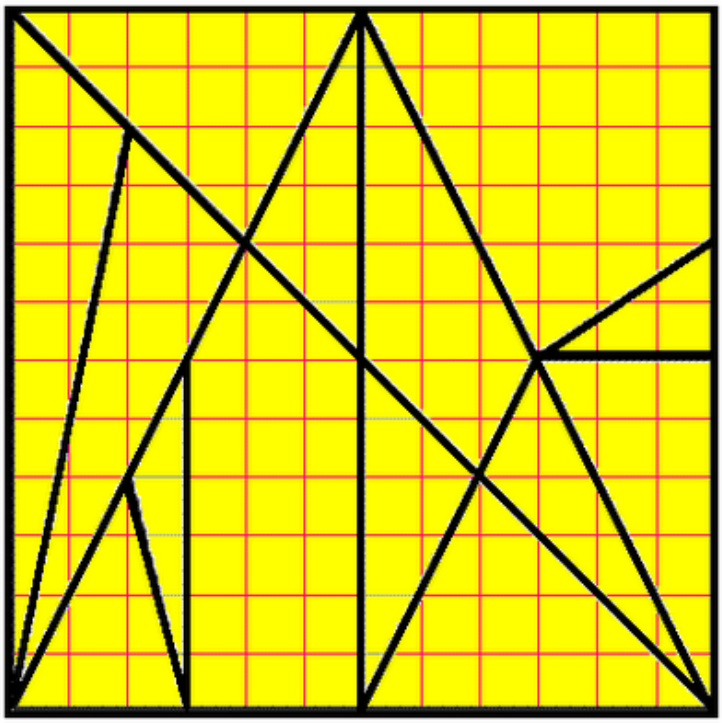


Ilustración 1: Stomachion

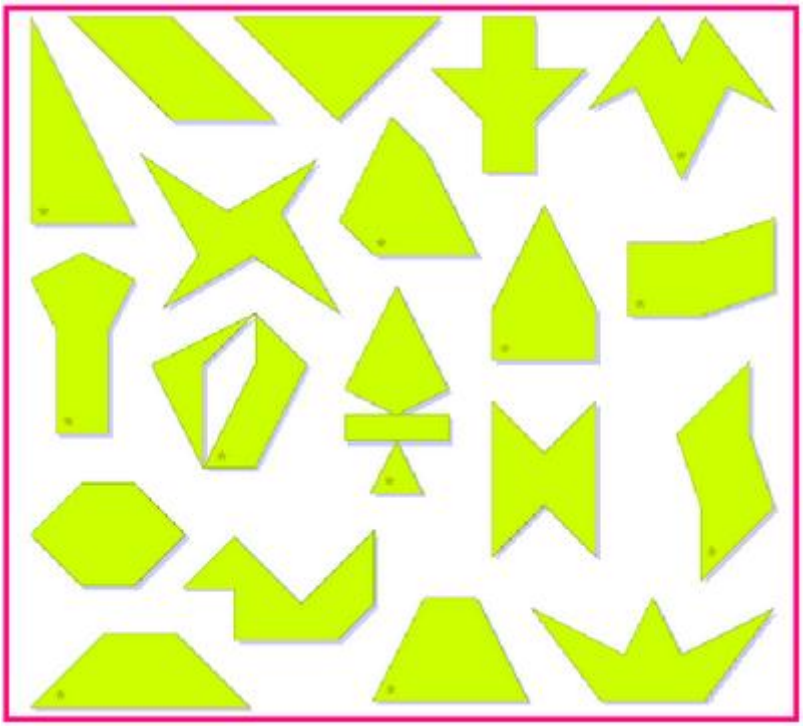


Ilustración 2: Figuras stomachion

Como se ve en la imagen superior, se tenían que conseguir una serie de figuras geométricas, o representando animales, símbolos o personas. Este puzle se dice que es de hace 2200 años y tiene origen en la antigua Grecia.

### 2.1.2. Torres de Hanoi

En el año 1883 el francés Éduard Lucas crea el juego de las torres de Hanoi. Este es un puzle muy conocido, que consiste en llevar la estructura de la posición inicial (izquierda) a la final (derecha) siguiendo las reglas de que sólo se puede mover una pieza si no tiene otra encima, y no se puede poner una pieza grande encima de otra de menor tamaño.

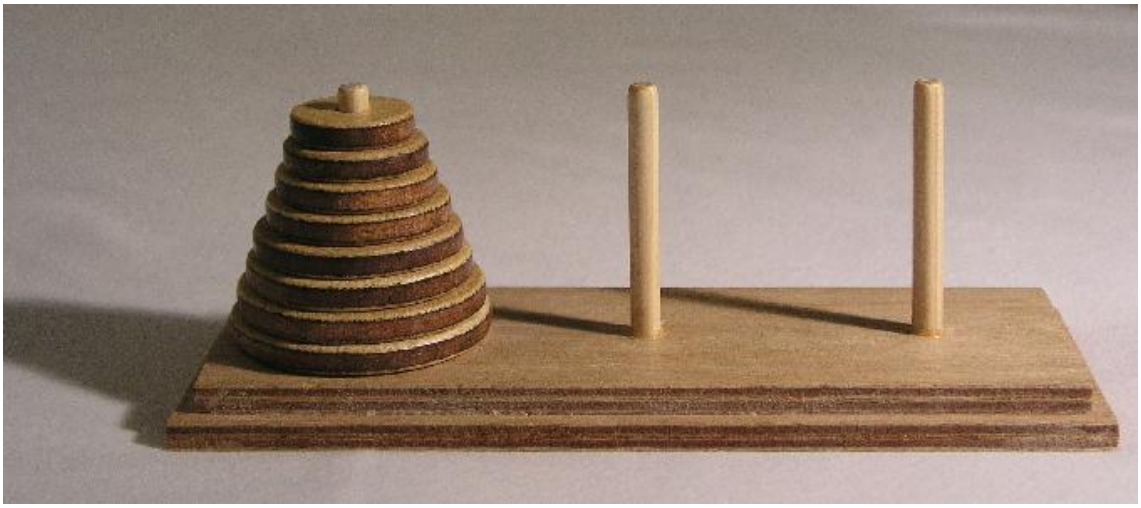


Ilustración 3: Torres de Hanoi

Este juego tiene la peculiaridad de que se puede graduar su dificultad añadiendo o quitando discos y palos intermedios, aunque la mecánica es la misma para cualquier caso, pero puede resultar lioso con muchos discos, y podemos equivocarnos.

Las torres de Hanoi se ven mucho durante la carrera, ya que son puestas de ejemplo en clases donde se trata la recursividad de funciones y en clases de inteligencia artificial.

### 2.1.3. Puzle 15

Este puzle se le atribuye a Noyes Palmer Chapman, un administrador de una oficina de correos de Nueva York. Se dice que en 1874 enseñó a sus amigos un puzle consistente en 16 bloques con números que había que agrupar en filas de 4, de tal manera que cada fila sumara 34.Ese fue considerado el precursor del puzle 15.

El hijo de Noyes llevó el puzle mejorado (el 15 puzle) a Syracusa (Nueva York), de ahí fue a Watch Hill (Rhode Island) y finalmente llegó Hartford (Connecticut), donde los estudiantes de la escuela americana de sordos empezaron a fabricarlo y distribuirlo de manera local.

Este puzle es muy conocido por los estudiantes de ingenierías. Tenemos una superficie cuadrada de 4x4 posiciones, 15 de ellas están numeradas del 1 al 15, la restante está en blanco. Se trata de colocar las piezas con número en orden ascendente, y el hueco blanco al final.


Ilustración 4: Puzle 15

Este puzle se usa como ejemplo en asignaturas relacionadas con la inteligencia artificial, cuando se habla de toma de decisiones basadas en heurísticas.

Se puede simplificar el puzle con una cuadrícula de 3x3 (puzle 8) o complicar con una de 5x5 o superior (puzle 24). Este puzle representa permutaciones de 16 elementos, con lo que el espacio de estados es considerable.

### 2.1.4. Tortitas (pancakes)

El problema de las tortitas apareció por primera vez en 1975, publicado en 1975 por Jacob E. Goodman en la revista *American Mathematical Monthly* número 82 (*1975*). Desde entonces ha sido objeto de estudio, y ha interesado como grupo de permutaciones y como método de ordenación

Bill Gates escribió un artículo para la revista *Discrete Mathematics*. 27, 47-57, 1979 hablando del algoritmo de las tortitas para ordenar listas de elementos, fue el único artículo matemático que se le conoce.

Tenemos una serie de tortitas apiladas de distintos tamaños, y queremos ordenarlas de tal forma que las más grandes queden debajo y las pequeñas arriba. Para ello disponemos de una espátula que, al insertarla en cualquier punto de la pila, invierte el orden de los elementos por encima de la espátula, como se ve en la siguiente imagen.

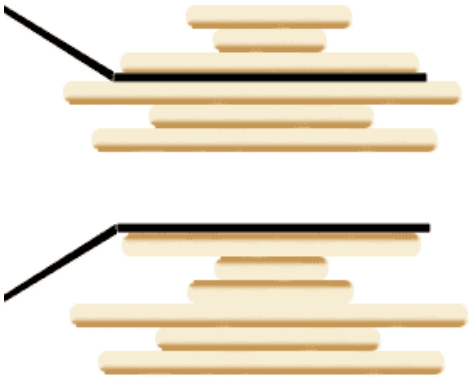


Ilustración 5: Pancakes

Este problema tiene su interés porque lo que estamos haciendo es ordenar una pila de N elementos en base a un criterio, y si se descubre un algoritmo que lo haga de manera eficiente se podrá aplicar como método para ordenar elementos.

Una variación de este problema es el de las “tortitas quemadas”, en el que todas las tortitas tienen un lado quemado, y al final han de quedar ordenadas, y con el lado quemado hacia abajo.

### 2.1.5. Topspin

Otro juego tipo puzle interesante es el llamado *topspin* (inventado por Ferdinand Lammertink en 1989), en donde tenemos una serie de números dispuestos en un bucle. Podemos desplazar los números y hacer que se muevan por el bucle. Hay una rueda que puede dar la vuelta a 4 números consecutivos:


Ilustración 6: Topspin

El objetivo del puzle es ordenar los números. Este juego se asemeja al cubo de rubik porque estamos ante un espacio de permutaciones, y también tiene interés porque pretendemos ordenar una lista de elementos (una lista circular, se podría considerar) con una operación sencilla.

Este juego tiene dos tipos: el original, con una lista de 20 números, y una versión de 26 piezas, en donde la rueda gira 5 piezas y que tiene dos carriles en los que cabe una pieza, es decir, que se puede sacar una pieza del bucle y meter donde se necesite. Con este elemento el juego se simplifica muchísimo.

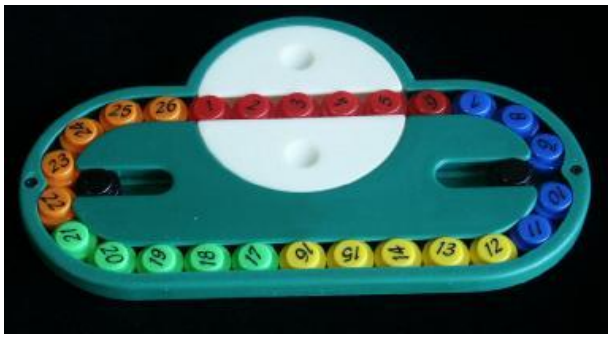


Ilustración 7: Topspin 26

### 2.1.6. Sokoban

Todos los puzles mostrados son de sobra conocidos, pero hay otros que también son interesantes de estudiar y se conocen menos. Uno de ellos es el Sokoban (creado por Hiroyuki Imabayashi en 1981):

Ilustración 8: Sokoban

Tenemos un tablero con una serie de cajas, un personaje que puede únicamente empujar (no tirar ni mover lateralmente) dichas cajas, el objetivo es poner todas las cajas en posiciones marcadas con puntos (en la imagen ya está resuelto).

En este juego tenemos que considerar muchos factores a la hora de decidir que caja queremos empujar en cada momento, por ejemplo:

- Posición inicial del personaje
- Posición de las cajas y los puntos
- Forma del tablero (los muros, pasillos y huecos problemáticos)
- Mover las cajas de forma que no bloqueen otras cajas.

La complejidad de este juego se puede equiparar con el ajedrez debido tanto a la amplitud del árbol de búsqueda como a la profundidad del mismo, por lo que estamos ante un problema de búsquedas muy amplio.

### 2.1.7. Rush hour

Inventado por Nob Yoshigahara a finales de 1970, era un juego de bloques, donde tenías un tablero y una serie de bloques representando los coches. Actualmente existen versiones de este juego para distintos dispositivos móviles, consolas y ordenadores.

El objetivo del juego es sacar un determinado coche de un aparcamiento moviendo el resto de los coches.

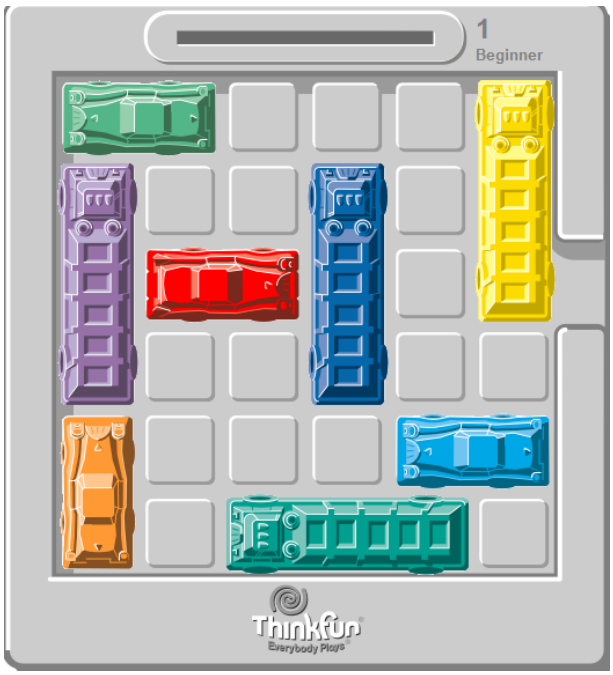



Ilustración 9: Rush hour

Los distintos vehículos se pueden desplazar sólo hacia delante o hacia detrás, siempre y cuando haya hueco (no se pueden montar dos vehículos en la misma casilla). El objetivo es sacar el coche rojo del aparcamiento, mientras lleguemos a ese estado da igual como queden los demás coches.

Estamos ante otro problema donde la amplitud y el profundidad del árbol de búsquedas puede ser muy grande (depende del número de elementos que manejemos).

### 2.1.8. M12

Uno muy poco conocido es el M12. Tenemos una fila de números (del 1 al 12) que están desordenados, y para ordenarlos debemos usar dos operaciones.

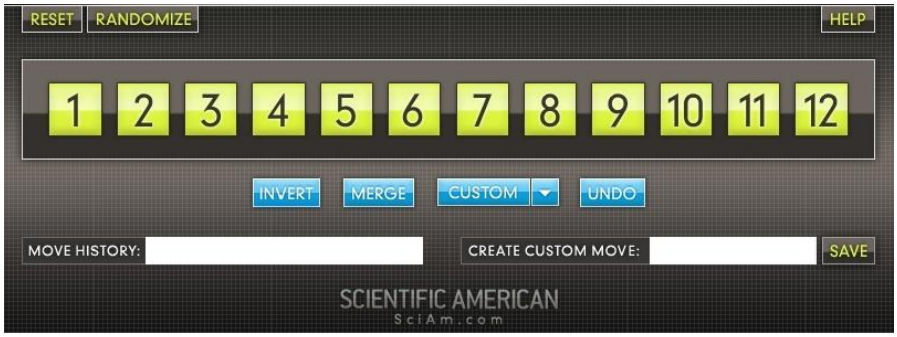



Ilustración 10: M12

Las operaciones permitidas son dos:

- *Invertir:* Invierte el orden de la lista, en el caso de la imagen de arriba sería empezar con el 12 en la posición del 1, el 11 en la del 2 etc...
- *Barajar:* Mezcla los números como si se barajaran, alternando las posiciones. La alternación funciona de la siguiente manera: El número de la primera posición se mantiene, en la segunda posición se pone el número que estaría al final, en la tercera el de la segunda posición, en la cuarta el penúltimo… Si barajamos la secuencia de la imagen de arriba el orden quedaría así:


Ilustración 11: M12 barajado

Es como barajar alternando las cartas.

El juego ofrece la posibilidad de crear tus propios movimientos, combinando los ya existentes, lo que te permite crear una cadena de movimientos con un objetivo concreto (por ejemplo, permutar dos posiciones sin que el resto se altere). De hecho las instrucciones del juego sugieren que es fundamental crear tus propios movimientos para resolverlo.

## 2.2. Puzles y videojuegos

Hoy día los puzles se utilizan en videojuegos de mucho éxito, por ejemplo tenemos la saga del Profesor Layton, muy famosa en nintendo DS. Combina una historia de misterio con puzles clásicos (como los mencionados antes).


Ilustración 12: Profesor Layton

## 2.3. La inteligencia artificial y los puzles

Los puzles siempre han supuesto un campo de trabajo muy amplio para la inteligencia artificial, puesto que trabajan con entornos bien definidos, con reglas claras y con un espacio de estados cerrado (muy grande en muchos casos).

Por lo tanto son un escenario ideal para probar las distintas técnicas existentes de toma de decisiones y búsqueda en un espacio de estados.

### 2.3.1. Representación

La representación de los puzles viene dada por un conjunto de estados que se relacionan mediante una serie de operaciones. Tiene que existir al menos un estado denominado "final" que es la solución del puzle. El estado por el que empezamos a resolver es el estado inicial.

Del estado inicial, según las operaciones permitidas, salen N estados "hijos", y de éstos pueden salir más, esto se denomina **árbol de búsqueda**. El número de niveles de un árbol de búsqueda se denomina **profundidad**, y la los nodos en un mismo nivel **amplitud.**

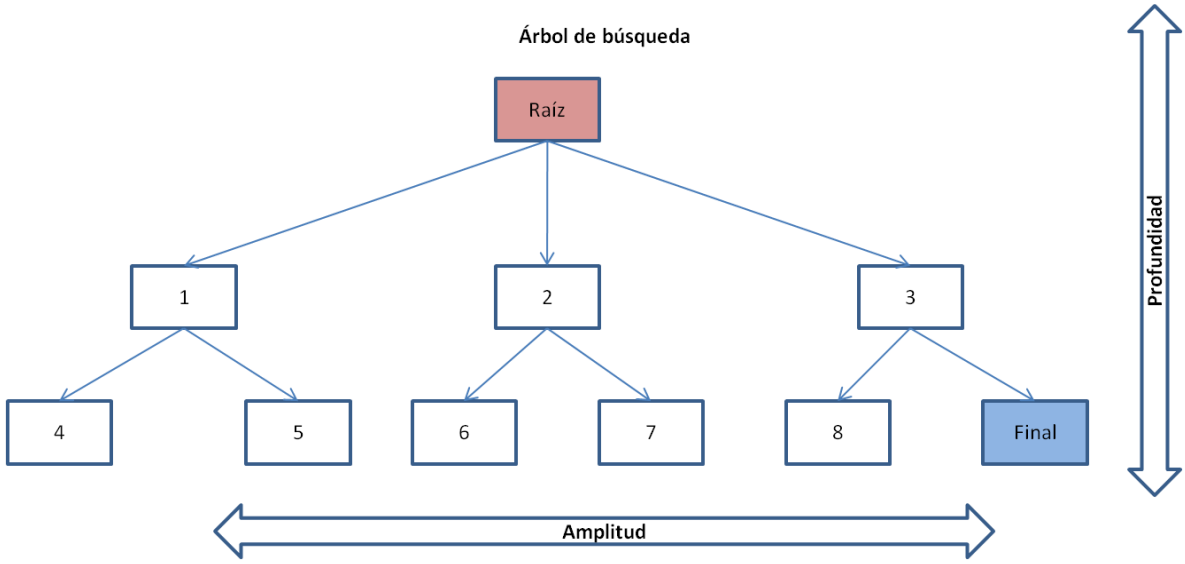



Ilustración 13: Árbol de búsqueda

### 2.3.2. Búsqueda no informada

Una búsqueda consiste en explorar el árbol de búsqueda (una vez generado o mientras se genera), y comprobando si un determinado estado es la solución. Pero dependiendo de cómo busquemos, la solución encontrada puede ser la mejor o no.

- *Admisible:* Encuentra la solución, y el camino elegido es el de menor coste.
- *Completo:* Encuentra la solución, pero no tiene por qué ser la de menor coste.

Si buscamos sin información adicional que ayude a elegir un camino (búsqueda no informada) tendremos dos casos de búsqueda:

- *Amplitud:* Partiendo del nodo raíz exploramos todos los hijos, una vez explorados expandimos cada hijo y exploramos todos sus hijos. Es decir, exploramos todos los niveles por completo. Este algoritmo es **completo**. Es **Admisible** si y sólo si el coste de todas las operaciones es el mismo. Si el coste de las operaciones no es el mismo, puede que haya una solución con menor coste a más profundidad.
- *Profundidad:* Partiendo del nodo raíz exploramos todos los hijos. Luego expandimos el hijo más a la izquierda y exploramos todos los hijos, repetimos la operación con los hijos de éste. No exploramos todos los niveles por completo, y en caso de tener un árbol de profundidad "infinita" puede que no se encuentre la solución.
  De manera práctica, lo que se suele hacer es poner una profundidad máxima, de tal forma que se explora el árbol hasta ese tope, y en caso de no encontrar la solución, se aumenta. En este caso es **completo** porque si existe la solución la encontrará, pero no es admisible, porque no nos garantiza que sea la mejor solución.

### 2.3.3. Búsqueda informada

Pero puede ser que tengamos información adicional que nos ayude a tomar una decisión, algo que nos diga cuál de las alternativas disponibles es la mejor en un determinado punto de la búsqueda. La información adicional suele dar una idea de cuánto falta para llegar a la solución desde un determinado estado.

Dicha información adicional se calcula teniendo en cuenta sólo los datos del estado, y la información que da no será completa ni exacta en la mayoría de los casos. Es lo que se llama la función heurística.

Dicha función tiene que tener una serie de características para que se considere que es adecuada como estimación de la distancia a la solución:

- *Admisible*: La distancia estimada a la solución ha de ser menor o igual a la distancia real. Si diera distancias mayores no sería una buena heurística, puesto que podría hacer que nos fuéramos por un camino más costoso.
- *Informada*: La información que da la heurística tiene que ser amplia, moverse en un rango razonable de valores, puesto que si tenemos un rango pequeño en relación al número de nodos hará que tengamos muchas opciones con la misma heurística, lo que provocará mucho *backtraking* y hará que perdamos tiempo y memoria.

Definir una buena heurística es la base de un buen proceso para explorar los estados del árbol. Para cada problema al que nos enfrentemos puede existir una heurística muy determinada que se ajuste a las exigencias de esa búsqueda, pero existen algunas heurísticas que son eficaces en muchos problemas.

- *Distancia de manhattan:* Se trata de buscar el camino más corto, pero usando las distancias absolutas (números enteros). En el caso de buscar el

recorrido más corto entre dos puntos en una ciudad, la distancia de manhattan sería la línea azul. Este sería el recorrido más corto desde un punto a otro, y se puede ver que es admisible (siempre dará un valor menor o igual a la distancia real) e informada (hay mucho rango de valores).

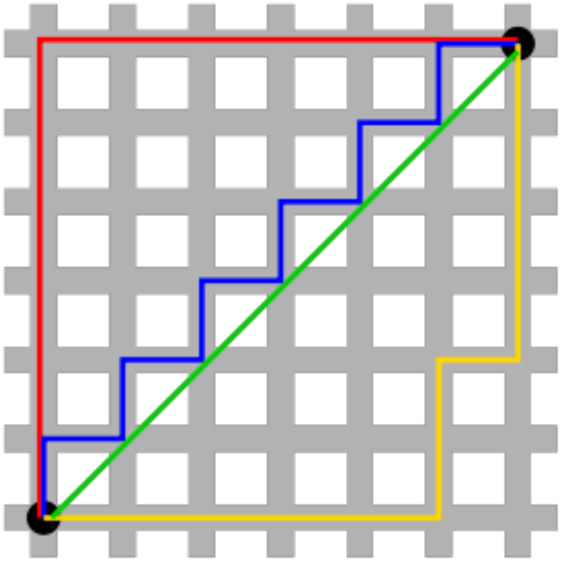


Ilustración 14: Distancias

- *Distancia euclídea:* Es, estrictamente, la distancia en línea recta entre dos puntos (en el anterior esquema, la línea verde). Dicha distancia será siempre menor o igual que la distancia real, y nos da un valor informado en cada estado. Hay que aclarar que no es ni mejor ni peor que el anterior método, depende de las características del problema.

Hay muchos problemas en los que es difícil encontrar una heurística, y en muchos casos, una vez encontrada, puede ser complicado determinar si es admisible o no. Si no se puede aplicar heurísticas existentes directamente, se puede utilizar un método para encontrar una heurística admisible.

- *Relajación de restricciones:* Es uno de los métodos más usados, ya que garantiza la admisibilidad. Se trata de eliminar alguna de las restricciones o normas que tiene el problema original, de forma que sería más fácil resolverlo, y por lo tanto sería fácil calcular la distancia a la solución real desde un estado concreto. Al quitar una restricción, sabemos que tendremos una heurística admisible y, dependiendo de la restricción quitada y de las características del problema, seguramente sea informada.

Este método no es de obligado seguimiento, cada problema puede tener una heurística distinta, y en muchos casos se puede diseñar sin seguir los métodos existentes. Y en la mayoría de los casos se adaptan los métodos o heurísticas existentes a los problemas, e incluso se pueden aplicar varias heurísticas, dependiendo de las características de un determinado estado. Incluso se puede completar una heurística admisible poco informada con una heurística no admisible pero más informada.

Una vez que se tiene una función heurística hay que decidir como explorar el árbol en función de esa información adicional que poseemos. Para ello existen varios métodos:

- *Primero el mejor:* Definida una función heurística admisible, h(x), se elije siempre para expandir el nodo que tenga mejor heurística. Es **completo**, pero puede ser no óptimo ya que sólo tenemos en cuenta el valor del nodo y no el coste de llegar hasta el mismo desde la raíz.
- *A*-estrella (A*): Tenemos una función h(x) admisible, definimos otra función g(x) que es el coste real desde el nodo raíz hasta el nodo en el que nos encontramos, y una función f(x) = h(x) + g(x). Cuando elegimos un nodo para estudiar, siempre escogemos el nodo que tenga menor f(x). Es **óptimo** siempre y cuando h(x) sea admisible. Si h(x) no está bien informada, es posible se exploren muchos nodos antes de dar con el mejor camino.

Estas y otras técnicas se utilizan para buscar en un árbol, en este proyecto se utiliza A* con alguna mejora para optimizar la búsqueda, reduciendo el número de estados que exploramos. Se podría decir que es un paso más a la hora de definir una búsqueda en un espacio de estados grande. Existen muchas formas de reducir el número de estados, y depende de las características del problema que se puedan aplicar o no. Aquí van algunos ejemplos

- *Eliminar estados repetidos:* Si estamos ante un problema que puede generar estados repetidos, nos interesa eliminar los estados que ya hemos estudiado, puesto que eso agilizará la búsqueda evitando expandir nodos que ya han sido explorados.
- *Eliminar simetrías:* Dos estados pueden no ser iguales, pero si simétricos, es decir, pueden ser equivalentes y generar caminos del mismo coste.
- *Utilizar más de una heurística:* Se tienen varias funciones heurísticas, que según sus características y como se usen pueden ayudar a agilizar la búsqueda:
    - *Heurística en función del estado:* Teniendo definidas un conjunto de funciones heurísticas admisibles y un conjunto de estados con una serie de características, aplicamos una de esas funciones según las características del estado. La comprobación del estado, así como las funciones heurísticas definidas, tienen que ser operaciones con un coste razonable, porque si son muy costosas puede que la búsqueda tarde demasiado.
    - *Heurística no admisible de apoyo:* Si tenemos una función heurística admisible pero poco informada, tendremos que recorrer muchos nodos que no pertenecen a la solución. Para ayudar en la elección de un nodo de entre varios con la misma puntuación vemos su valor con la función heurística no admisible, y escogemos la de menor valor. Como la función principal es admisible, y la no admisible sólo se usa para elegir entre los que son iguales, la búsqueda seguirá siendo óptima.

- *Bases de datos de patrones:* Esta es una técnica muy potente, y se suele considerar como un método independiente, pero realmente aplicamos el método de búsqueda informada que queramos, con la ayuda de bases de datos de patrones.
  Una base de datos de patrones es un conjunto de estados con unas características comunes. Dichos estados están en el camino a la solución, y nos interesa llegar a cualquiera de ellos. Ese conjunto se crea y agrupa en base a unas características (misma distancia a la solución, por ejemplo).
  Se pueden usar muchos patrones aplicados a un mismo problema, en distintas fases o combinar en una misma fase varios patrones.
  También se pueden aplicar distintas heurísticas, o incluso distintos métodos de búsqueda dependiendo de las características de un patrón determinado.

# 3. El cubo de rubik y su entorno

El cubo de rubik ha sido objeto de análisis desde sus orígenes debido a sus características. Un problema con un espacio de estados inmenso, siempre se han buscado métodos y maneras de resolverlo.

Actualmente existen varios métodos para que las personas puedan resolver el cubo de rubik rápidamente, pero también se buscan maneras en que un ordenador pueda resolver el cubo en un tiempo razonable, o en un número de movimientos mínimo.

Se ha podido demostrar que el número de movimientos máximo para resolver cualquier cubo es 20. Esto se ha sabido resolviendo todos los estados del cubo de rubik con una búsqueda que garantizada la admisibilidad.

Actualmente se pueden encontrar aplicaciones que resuelven el cubo en poco tiempo. Algunas son admisibles, pero o tardan mucho tiempo, u ocupan mucho espacio de disco duro porque necesitan instalar bases de datos de patrones muy grandes.

En este proyecto se busca una solución rápida, sin que ello signifique consumir mucho disco duro. Nos centraremos en la optimización en tiempo.

## 3.1. Representación y operaciones

El cubo de rubik se representa identificando de manera única sus elementos. Inicialmente se hace una distinción entre sus piezas, en función de dónde están colocadas.

- *Arista:* Pieza de dos colores situada en una arista del cubo.
- *Esquina:* Pieza de tres colores situada en un vértice del cubo.

La distinción entre piezas es importante porque las operaciones, las opciones y las posibles posiciones varían mucho dependiendo de cuál sea.

Para representar el cubo se hace asignando un identificador único a cada uno de sus elementos de la siguiente forma.

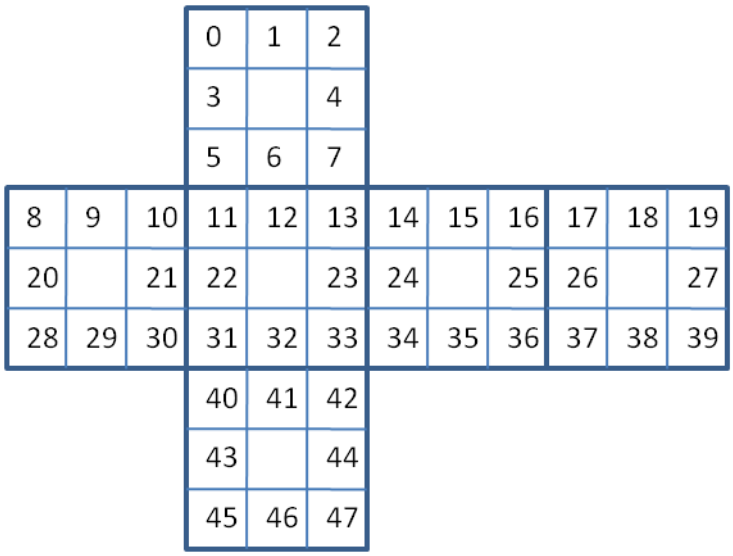



Ilustración 15: representación 3x3x3

Nótese que las piezas centrales no se numeran, puesto que no van a girar ya que la posición del cubo no varía a lo largo de todo el proceso. Debido a esto tenemos que cada cara recibe un nombre en función de si orientación, teniendo entonces esta distinción:

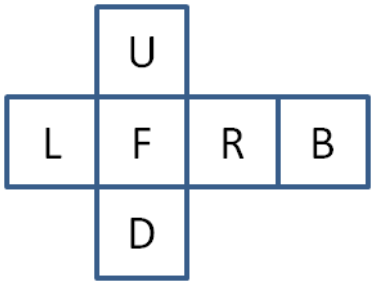



Ilustración 16: caras 3x3x3

Dicha distinción nos va a ayudar a nombrar las operaciones simples que pueden darse en el cubo, de esta forma tenemos que si ponemos el nombre de una cara, tenemos un giro de 90º de dicha cara. Si lo acompañamos de un '2' o un '3' tenemos un giro de 180º o 270º, en la representación numérica se vería así.

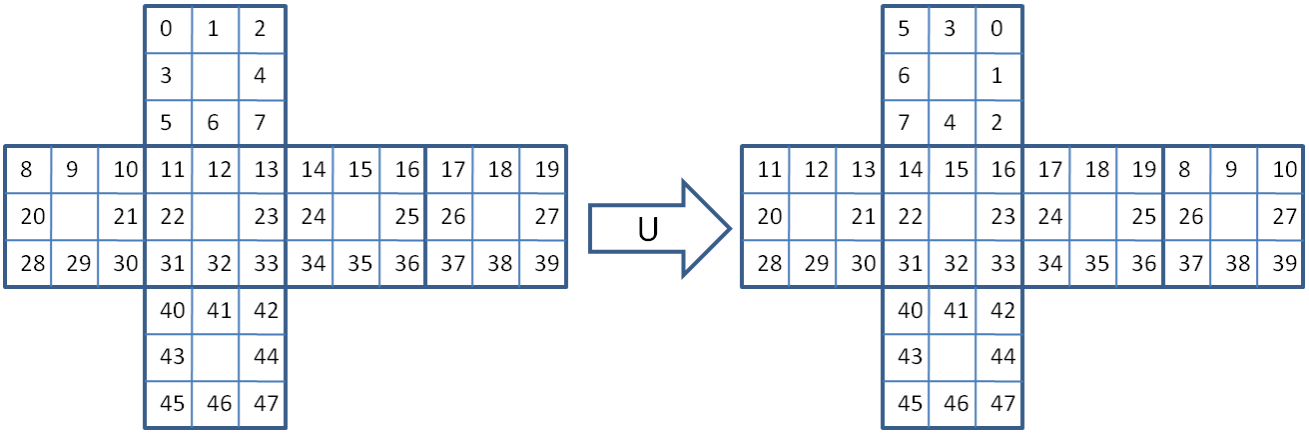



Ilustración 17: Movimiento U

El del ejemplo es un movimiento de la cara superior 'U', que se gira 90º en el sentido de las agujas del reloj, 'U2' sería una giro de 180º, y 'U3' un giro de 270º (o de 90º en sentido contrario a las agujas del reloj).

Como se puede observar lo que tenemos es una permutación de elementos. La representación de los estados y las permutaciones se hace mediante un array, intercambiando las posiciones de un estado según una permutación.

Esta representación nos da la ventaja de poder crear macrooperadores. Los macrooperadores son representación de varias operaciones simples en un solo operador, como ese operador resume varias operaciones se le llama macrooperador.

La representación de un macrooperador es la misma que la de un operador, y la operación se aplica de igual manera sobre un estado, pero al condensar varias operaciones tenemos un ahorro de tiempo importante.

## 3.2. Simetrías

Hablamos de simetría cuando tenemos dos estados del cubo que son distintos, pero equivalentes porque presenten una semejanza en cuanto a la distribución de sus piezas. Veamos un ejemplo.

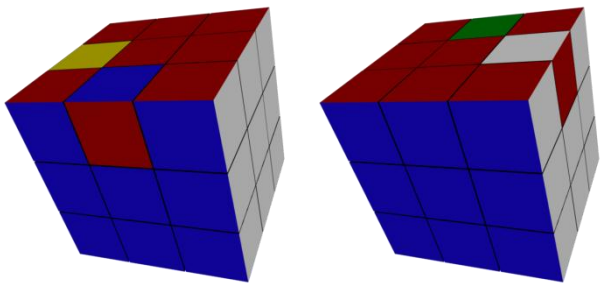


Ilustración 18: Simetría

Esto serían dos estados equivalentes, en los dos estados tenemos el cubo hecho salvo dos aristas adyacentes que están en su posición pero mal orientadas. Para resolver ambos casos se usarían movimientos simétricos. Si cambiáramos de posición el segundo cubo

para que las aristas mal orientadas estuvieran en la misma posición que en el primer cubo, las operaciones serían las mismas.

Se puede observar en la ilustración de ejemplo que ese caso concreto de simetría se puede repetir con todos los pares adyacentes de aristas, lo que nos da un grupo de simetrías grande, y hemos de tener en cuenta otras simetrías de las aristas y las simetrías en los vértices. La siguiente ilustración muestra los planos y ejes de simetría en un cubo.

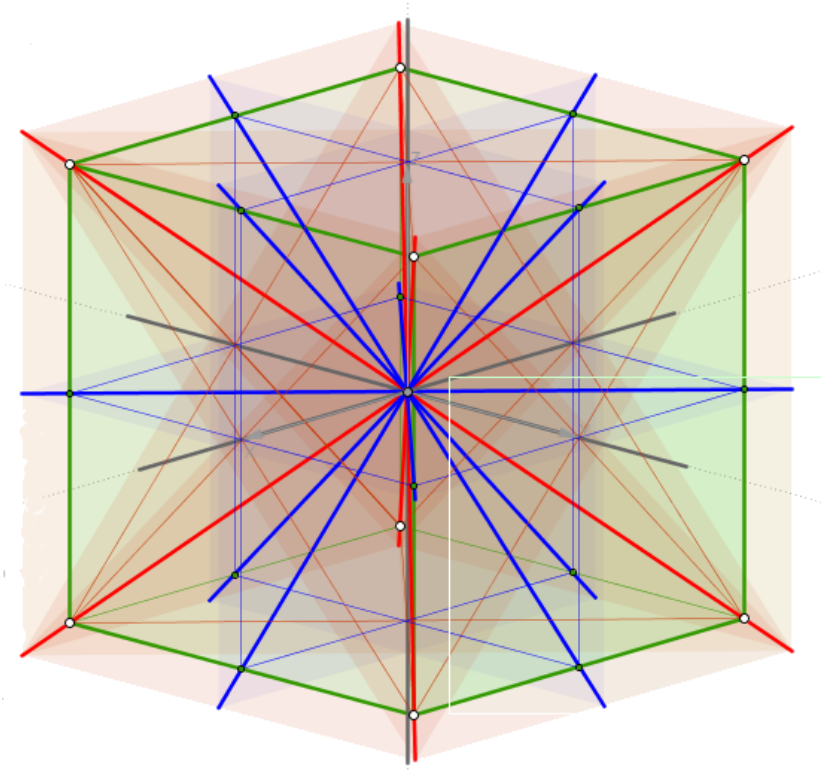


Ilustración 19: Simetrías en un cubo

La ilustración nos da una idea de las simetrías existentes en un cubo, que están presentes en el cubo de rubik y son fundamentales a la hora de que una persona pueda resolver un cubo. Hay que tener en cuenta que el cubo de rubik presenta los mismos elementos que un cubo, aristas, vértices y caras (o centros), es por ello que las simetrías son las mismas.

En las instrucciones existentes para que una persona pueda resolver el cubo recomiendan reorientar el cubo para colocar estados simétricos en la misma posición, y que de esta manera resulte más sencillo resolver el cubo.

Las simetrías del cubo también se utilizan en los programas que resuelven el cubo. Se pueden utilizar para crear bases de datos de patrones y para podar un árbol, eliminando o evitando expandir estados simétricos a los que ya hemos estudiado.

En nuestro caso, a la hora de generar macrooperadores, se ha tenido en cuenta que los presentes en el método de resolución elegido pueden tener simetrías que hay que considerar.

Todos los puzles similares a cubo de rubik presentan unas simetrías que son fundamentales a la hora de resolverlos, ya sea por una persona o por un programa.

## 3.3. Otros puzles

Hoy día existen muchos puzles que han tomado como base el cubo de rubik, estos puzles pueden ser más complejos o más simples que el cubo normal.

Lo que hay que destacar es que debido a su similitud con el cubo de 3x3x3, se pueden usar las técnicas para el mismo, haciendo una preparación previa o modificando mínimamente los pasos.

Las técnicas para el cubo 3x3x3 pueden no ser suficientes, puesto que en otros puzles podemos tener estados que en el cubo 3x3x3 no se pueden dar, y es necesario hacer algún paso extra.

Ilustración 20: Otras figuras

### 3.3.1. Cubo de rubik de 2x2x2

Este cubo es una simplificación del cubo de rubik, compuesto por 8 esquinas. Tiene un total de 4723920 de estados, muchos menos que el cubo tradicional, pero no por ello es fácil de resolver (por una persona).

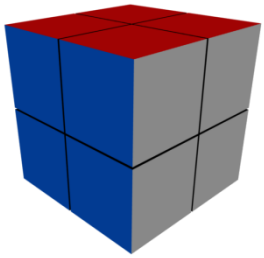


Ilustración 21: Cubo 2x2

Si nos enfrentamos a un cubo de 2x2x2 sin tener ninguna noción sobre el cubo de rubik, pensaremos al principio que es muy fácil, luego comprobaremos que no lo es tanto. Es más fácil pero requiere de cierta técnica y ciertos conocimientos del tema.

Cabe destacar la peculiaridad de que en las esquinas del de 2x2x2 se puede dar alguna combinación que es imposible en el cubo de 3x3x3. En este caso este hecho no complica la resolución del cubo, y es posible resolverlo sin dificultad si se sabe resolver el tradicional.

Para resolver este cubo usaremos los pasos 2, 6 y 7 de la resolución del cubo de 3x3x3. Esos 3 pasos son los que colocan las piezas en su sitio. Por lo tanto los pasos serían los siguientes:

1. *Hacer la primera capa:* Para hacer la primera capa hemos de colocar sus aristas igual que colocaríamos los del cubo de 3x3x3. Hay que situarlos debajo de la posición que les corresponde y aplicar el macrooperador correspondiente.
   Para colocarlo debajo de la posición correspondiente son necesarios de 1 a 4 movimientos, igual que en el caso de 3x3x3.
   Los macrooperadores de esta parte tienen 4 movimientos, y puede ser necesario aplicar más de uno.
   Tenemos que acabar esta parte así:

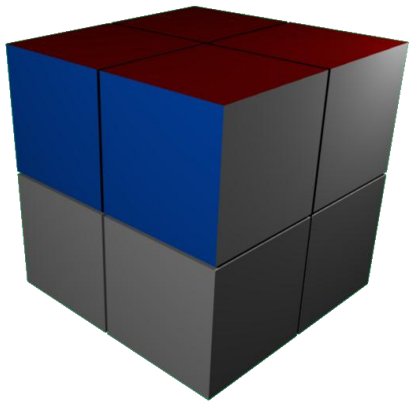


Ilustración 22: 2x2 primera capa

2. *Colocar en su posición los vértices de la capa inferior:* Hay que colocar las esquinas restantes en la posición que les corresponde, sin importar que estén bien orientados. En principio sólo hay que analizar en qué caso estamos y aplicar el macrooperador correspondiente del paso 6 del cubo de 3x3x3.
   Pero en este caso es donde se nos puede dar alguna combinación de esquinas inexistente en el cubo de 3x3x3. En el cubo de 3x3x3 los casos que se pueden presentar llegados a este punto son: Todos Las piezas bien colocados, uno bien colocado, el resto mal o todos descolocados. En el cubo de 2x2x2 podemos tener el caso de tener esquinas bien colocados y dos mal. Para salir de esta situación se puede aplicar el algoritmo avanzado de la T, que permite cambiar dos esquinas y dos aristas de un cubo de 3x3x3 de una tacada.

Como el cubo de 2x2x2 sólo tiene esquinas lo que conseguimos es cambiar la posición de dos esquinas adyacentes. Los movimientos son los siguientes: RDR3D3L3DRLD3R3D3RDL3D3x

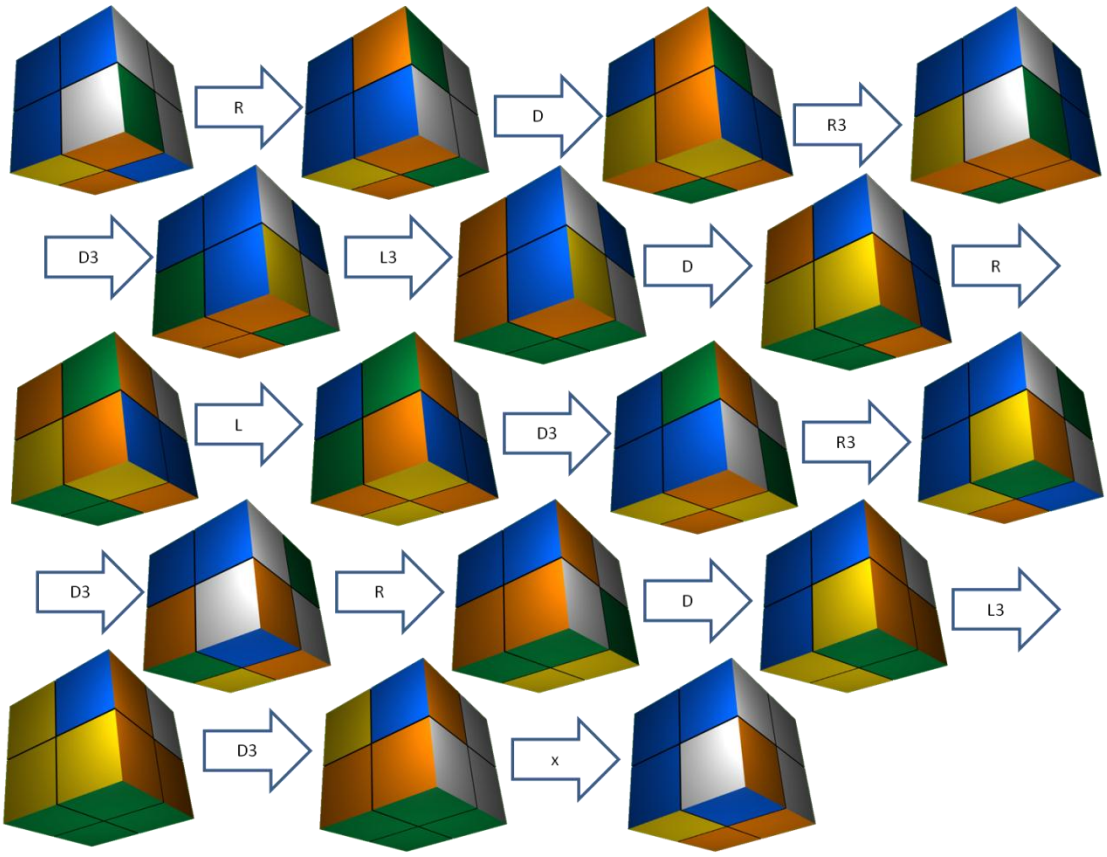



Ilustración 23: 2x2 la T

3. Esta es una de las formas de hacer el movimiento conocido como la T, pero hay muchas más. Ésta se adapta bien a las manos, y suele ser la que se usa en las competiciones de velocidad. El "movimiento" final 'x' indica que debemos reorientar el cubo entero para tenerlo en la misma orientación que al empezar, por lo tanto no es un movimiento que se tenga en cuenta para ver el número de operaciones sencillas realizadas.

   Así cambiaríamos los dos que están mal clocados y quedarían bien colocados. Esta peculiaridad puede hacer que un principiante se quede algo perplejo. Al terminar tenemos el cubo de esta forma:

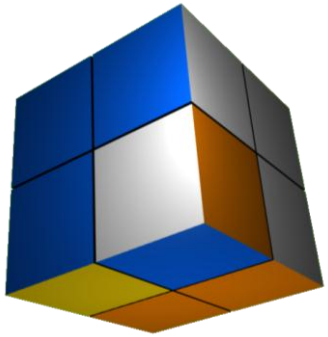


Ilustración 24: 2x2 orientado

4. *Orientar las esquinas:* Al llegar a este paso sólo hemos de analizar la situación en que nos encontramos y aplicar el macrooperador correspondiente. En este paso no encontraremos combinaciones no aceptadas en el cubo de 3x3x3.
   Al terminar este paso tendremos el cubo resuelto.

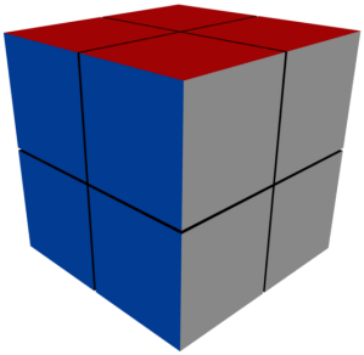


Ilustración 25: 2x2 terminado

Nos vamos a encontrar con un rompecabezas más complicado de lo que aparente.

### 3.3.2. Cubo de rubik de 4x4x4

Éste es el siguiente cubo regular al clásico, y es conocido como "la venganza de rubik". Añade 7 elementos nuevos por cara, es decir, pasamos de tener 9 elementos a tener 16, lo que hace que tengamos un total de 96 posiciones, frente a las 54 del cubo de 3x3x3 (48 si no contamos las centrales).

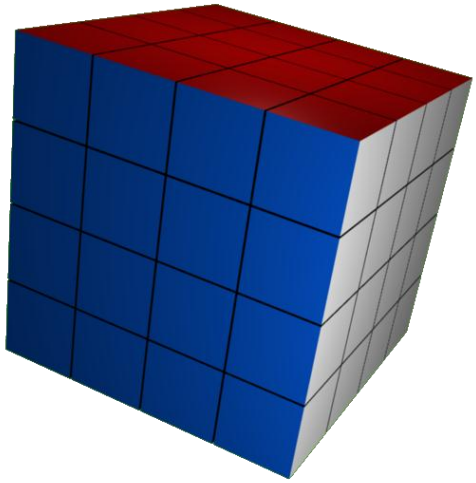


Ilustración 26: Cubo de 4x4x4

También se añaden piezas. Tenemos en total 56 piezas, frente a los 26 del cubo tradicional (20 si no contamos Las piezas centrales). La distribución de dichas piezas es la siguiente:

- *24 piezas centrales:* Piezas con un solo color que componen el centro de las caras. A diferencia de en el caso de 3x3x3, estos piezas sí pueden moverse y es necesario tenerlos en cuenta.
- *24 aristas:* Piezas situadas en las aristas del cubo, y que tienen 2 colores.
- *8 esquinas:* Las piezas de las esquinas con 3 colores. Existen los mismos que en el cubo de 3x3x3, ya que estos piezas no aumentan en número al aumentar los elementos, si no al variar las dimensiones de la figura.

El número de posibles combinaciones es aproximadamente $74 \cdot 10^{44}$ que es muy superior al número de combinaciones del cubo de 3x3x3, que es aproximadamente $43 \cdot 10^{18}$. Eso significa que tenemos un espacio de estados significativamente superior, por lo que tardaremos mucho más en resolverlo. Para hacernos una idea, el record del mundo en resolver este cubo está en 30'88 segundos (recordemos que el record del de 3x3x3 es de 5'66 segundos).

La forma habitual de proceder para resolver este cubo es transformarlo para simplificarlo, es decir, agrupar piezas para que el cubo tenga el aspecto de un cubo más sencillo que sepamos resolver. Así que las dos formas de agrupar son:

- *Agrupar en un cubo de 2x2x2:* Partiendo de las esquinas, vamos agrupando Las piezas hasta que tenga el aspecto de un cubo de 2x2x2, una vez llegados a ese punto, resolvemos como un cubo de 2x2x2 habitual. Quedaría así:

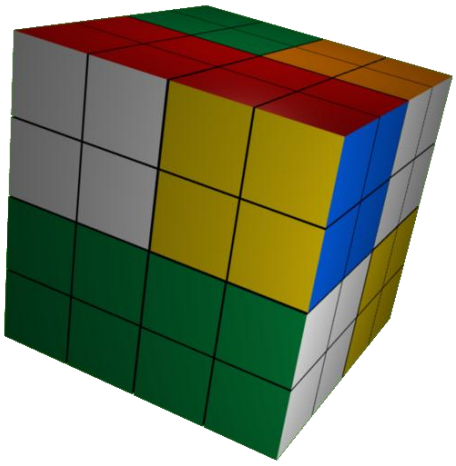


Ilustración 27: 4x4 transformado a 2x2

- *Agrupar en un cubo de 3x3x3:* Esta es la forma más popular de hacerlo, y realmente es más fácil de conseguir. Para ello hay que agrupar centros y luego aristas. Quedaría así:

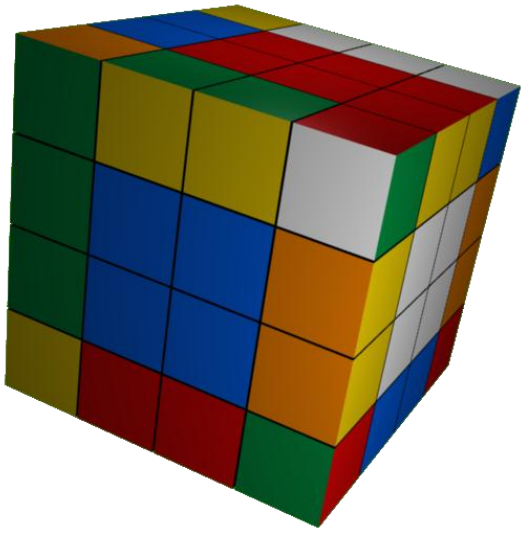


Ilustración 28: 4x4 transformado a 3x3

De estas dos formas, la preferida es la segunda. Y realmente es la manera más fácil de hacerlo. Analicemos los siguientes datos para verlo más claro:

- El cubo de 2x2x2 tiene un total de 4723920 de combinaciones. Y ese es el número de estados "finales" que existen para este paso intermedio.
- El cubo de 3x3x3 tiene un número aproximado de estados de $43 \cdot 10^{18}$, e igual que antes, es el número de estados "finales". Tenemos más estados finales, por lo que será más fácil completar este paso intermedio.
- Realmente, el cubo de 2x2x2 se resuelve utilizando técnicas del de 3x3x3, así que hemos de tener conocimientos del cubo normal.

- Es más general simplificar cubos de dimensión superior a 3 utilizando el cubo de 3x3x3, puesto que el de 2x2x2 sólo nos vale para cubos de dimensiones pares.

Vamos a optar por resolver el cubo de 4x4x4 agrupando piezas hasta tener un equivalente a un cubo de 3x3x3. Para ello es necesario crear Las piezas centrales y Las aristas. Los centros se harán juntando 4 piezas centrales, y Las aristas juntando 2 aristas del de 4x4x4.

El proceso es más laborioso que difícil, y teniendo conocimientos de resolución del cubo de 3x3x3 se puede llegar a resolver, aunque lleve tiempo.

Los pasos a seguir son los siguientes:

1. *Hacer los centros:* El primer paso es hacer los centros, juntando todas Las piezas centrales del mismo color en la misma cara. El proceso es sencillo, aunque se complica a medida que avanzamos. Hay que prestar atención a la disposición de los colores si no respetamos la disposición inicial luego no podremos resolver el cubo.
   La parte más delicada de este paso es mantener la disposición inicial de los centros, un error en esta parte y es posible que no nos demos cuenta hasta que empecemos con el último paso.
   Para hacer este paso se puede ir uno a uno, de dos en dos o hacer los 6 centros a la vez, eso ya depende del nivel de maestría. Lo normal es empezar de uno en uno, hacer uno, su opuesto, y luego los adyacentes.

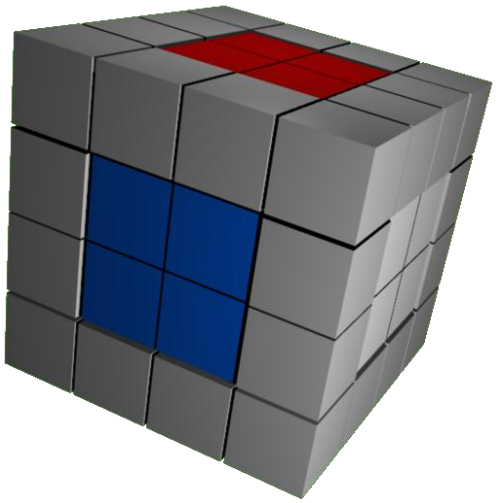


Ilustración 29: 4x4 centros

2. *Hacer Las aristas:* En esta parte tenemos que emparejar Las aristas que tengan los mismos colores. Tenemos que hacerlo de manera que tengan la misma orientación, y de esta forma, al finalizar esta parte, tendremos el cubo en formato 3x3x3. Como la parte anterior, es una tarea más laboriosa que difícil, aunque tiene más trabajo que la anterior.
   Hay que tener cuidado con ciertas operaciones, puesto que hay que

preservar el trabajo hecho en el anterior paso, sin alterar los centros ni su posición relativa.
En este paso pueden surgir problemas de paridad, pero no seremos capaces de detectarlos hasta que nos pongamos a resolverlo en el paso final. Dichos problemas se tratan cuando se encuentran, así que no es necesario prestarles atención en este punto.

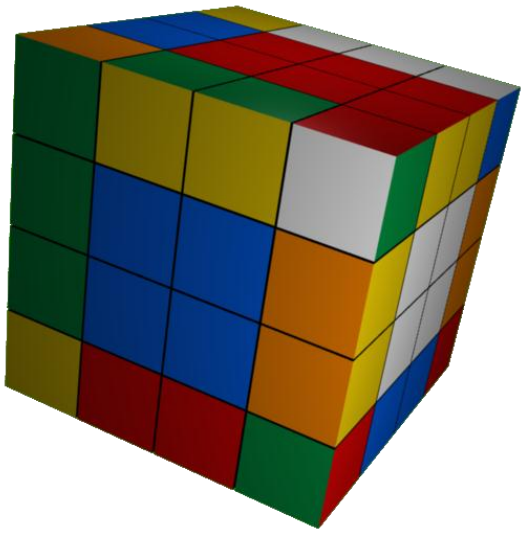


Ilustración 30: 4x4 aristas

3. *Resolver el cubo de 3x3x3:* Llegados a este punto lo resolveremos como un cubo de 3x3x3, aplicando los pasos ya vistos sin mayor novedad, pero es durante el desarrollo de esta parte donde podremos detectar ciertos problemas.
    a. *Centros mal colocados:* Si la posición relativa de dos centros está cambiada, veremos que no somos capaces de completar el paso de 2º del algoritmo del cubo de 3x3x3: completar la capa superior. Habrá una esquina que no lograremos orientar. En este caso hay que cambiar de posición 2 centros opuestos, manteniendo la posición relativa del resto de centros. Al realizar este paso también se nos descolocarán algunas parejas de aristas. No será como empezar de nuevo todo el trabajo, pero si tendremos que rehacer bastante, por ello hay que prestar atención a la disposición de colores.
    Aquí tenemos un cubo con los centros mal puestos, puesto que el color opuesto al rojo es el naranja y aquí está a la derecha del mismo.

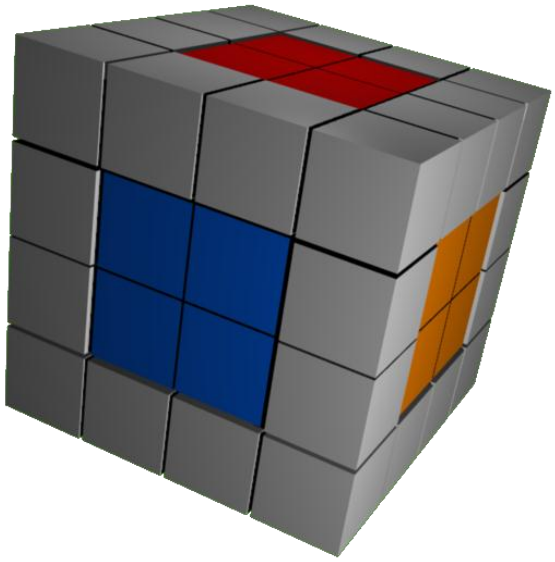


Ilustración 31: 4x4 centros mal puestos

b. *Problemas de paridad:* Puede darse el caso de encontrarnos ante una combinación que no se puede dar en un cubo de 3x3x3, como por ejemplo, tener todo el cubo hecho salvo una arista que está invertido.

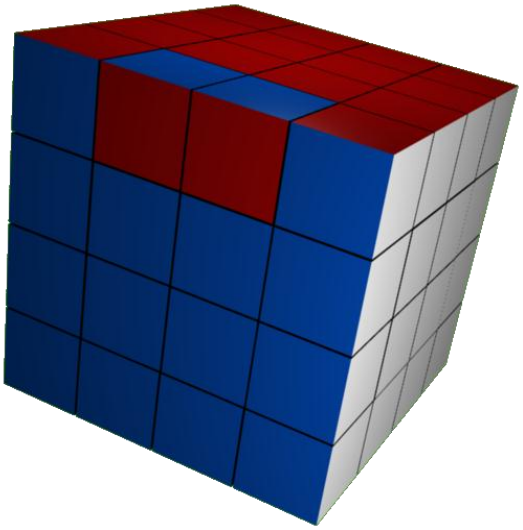


Ilustración 32: 4x4 Paridad

Se resuelve con estos pasos:

MR2B2U2MLU2MR3U2MRU2F2MRF2ML3B2MR2

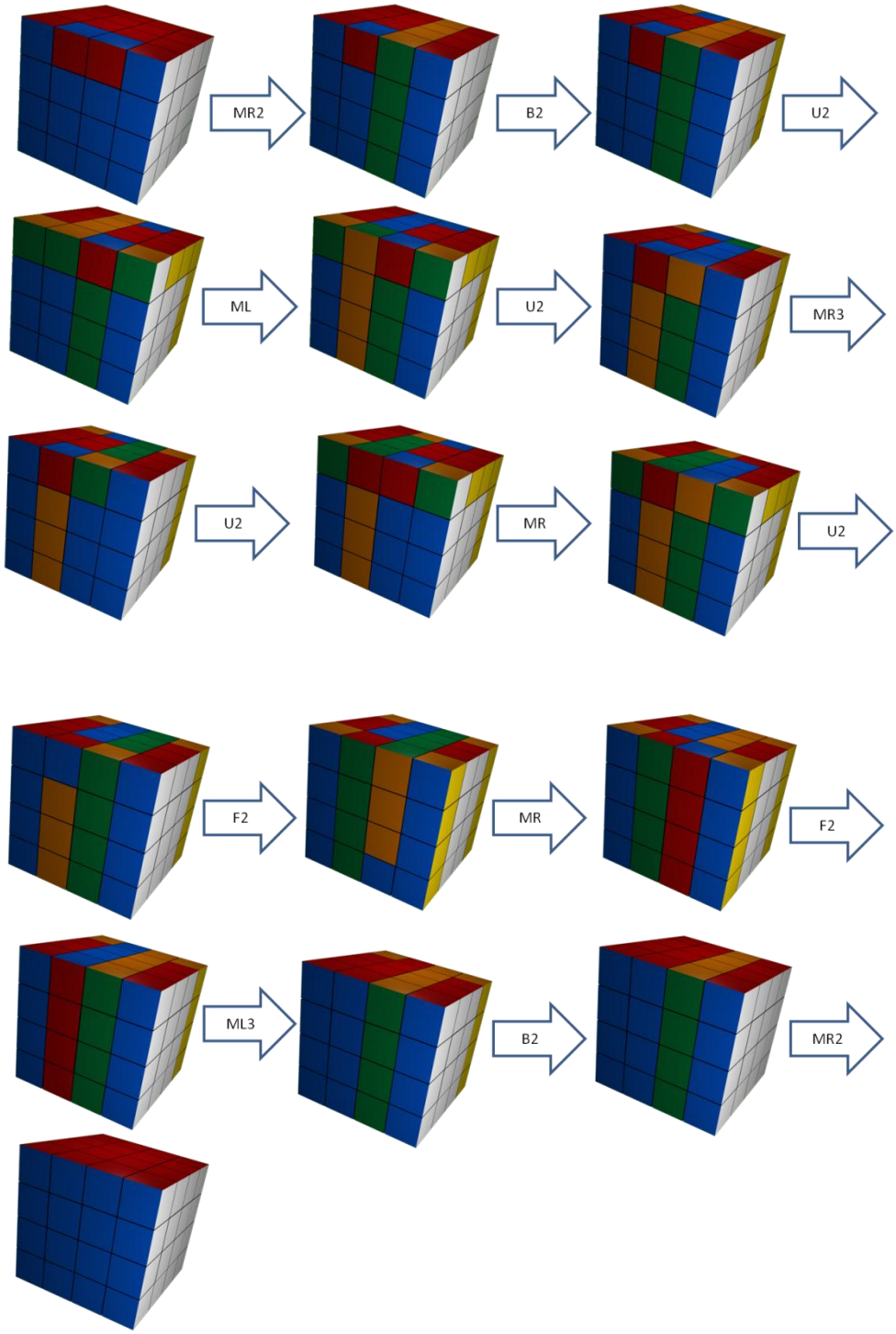



Ilustración 33: 4x4 operaciones paridad

Otros problemas de paridad son en los que hay dos aristas intercambiados al final, es decir, tenemos 2 aristas en su posición y

dos que no lo están. Este es el caso más sencillo (dos opuestos intercambiados) que se resuelve así: MR2U2MR2TU2MR2MU2

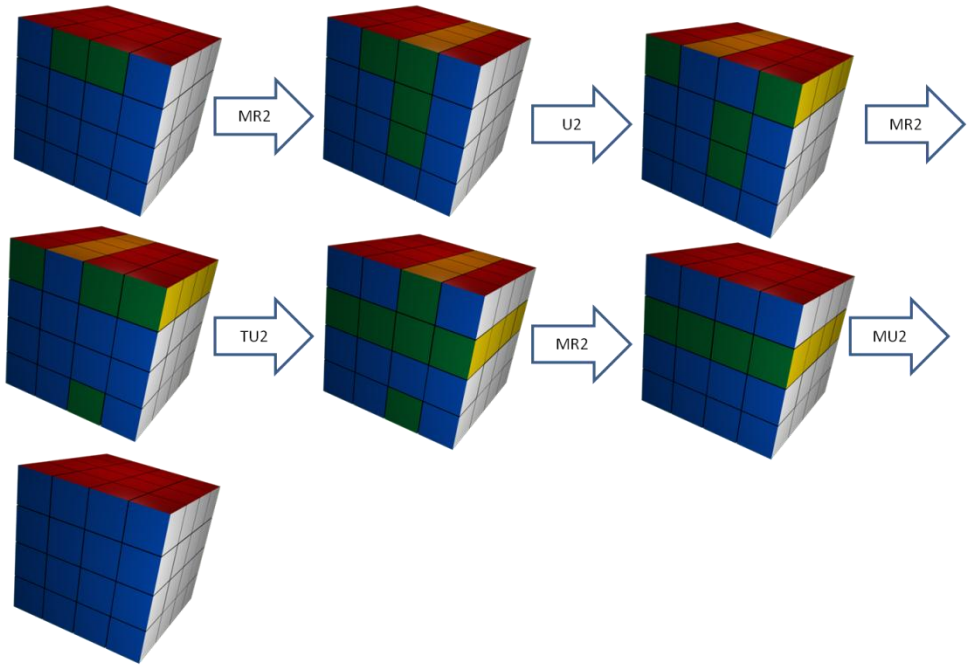



Ilustración 34: 4x4 intercambiar aristas

Si nos encontráramos algún otro problema de paridad (dos aristas no opuestos, dos esquinas…) se puede aplicar el algoritmo de arriba que deja el cubo en un estado que se puede resolver utilizando las técnicas del cubo de 3x3x3.

Los dos primeros pasos son fáciles de realizar para alguien que ya conoce el cubo de rubik, y luego solo queda resolver uno de 3x3x3 normal. El problema de paridad es lo único que requiere conocimientos adicionales.

### 3.3.3. Cubo de rubik de 5x5x5

Es el siguiente cubo regular que se comercializa, llamado "el cubo del profesor", haciendo referencia a la dificultad que supone hacerlo. Añade 9 elementos por cara con respecto al anterior, lo que hace que tengamos 25 elementos por cara, un total de 150 elementos.

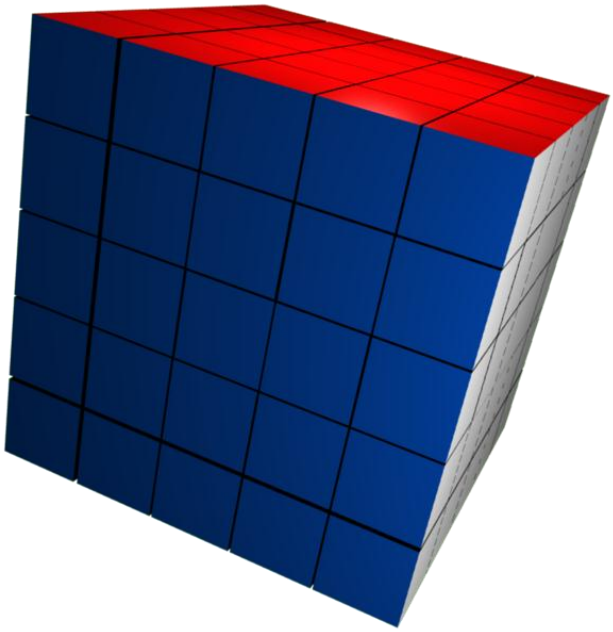


Ilustración 35: Cubo 5x5x5

Dichos elementos se agrupan en los conocidos piezas, dando un total de 98 piezas, frente a los 56 del anterior. Las piezas se van a clasificar siguiendo el mismo criterio que en anteriores ocasiones, lo que nos da una distribución de piezas:

- *6 piezas centrales:* Al igual que en el cubo de 3x3x3 tenemos elementos centrales que no se mueven y que determinan el color que va a tener la cara. Debido a ello se les clasifica a parte del resto de piezas que están en el interior del cubo.
- *48 piezas céntricas:* Cada pieza central tiene 8 piezas céntricas que lo rodean. Eso piezas si se pueden mover, y por ello se distinguen de los anteriores.
- *8 esquinas:* Las esquinas son los mismos en todos los cubos regulares.
- *36 aristas:* Por cada arista existente en un cubo de 3x3x3 tenemos ahora 3 aristas.

El número de estados que podemos tener con estos elementos es de aproximadamente $15 \cdot 10^{115}$, que es mucho mayor que en el anterior caso. Mientras que el cubo de 4x4x4 se resolvía en 30'88 seg, el record del cubo de 5x5x5 es de 1 minuto y 1'59 segundos, casi el doble.

La manera habitual de proceder, al igual que antes, es transformar el cubo en un cubo de 3x3x3 agrupando colores. A diferencia de en el anterior cubo, aquí no tenemos otra

opción de simplificación, puesto que es imposible agrupar colores para transformarlo en uno de 4x4x4 o de 2x2x2.

El proceder es similar al cubo anterior, con la dificultad añadida de que tenemos muchos más elementos que manejar. Primero hay que transformar el cubo en un cubo de 3x3x3, luego resolver el cubo de 3x3x3. Cabe destacar que, en este caso, los problemas de paridad se van a resolver antes de empezar con el cubo de 3x3x3, y que éste no va a presentar problemas de paridad ni de combinaciones imposibles.

1. *Hacer los centros:* Hay que agrupar todos Las piezas céntricos alrededor de los centrales de su color. Es una tarea laboriosa por el número de elementos que se manejan, pero no es difícil.
   En este caso tenemos unas piezas centrales que nos dicen cual va a ser el color de la cara, por lo que no es necesario fijarse en la disposición de los colores.

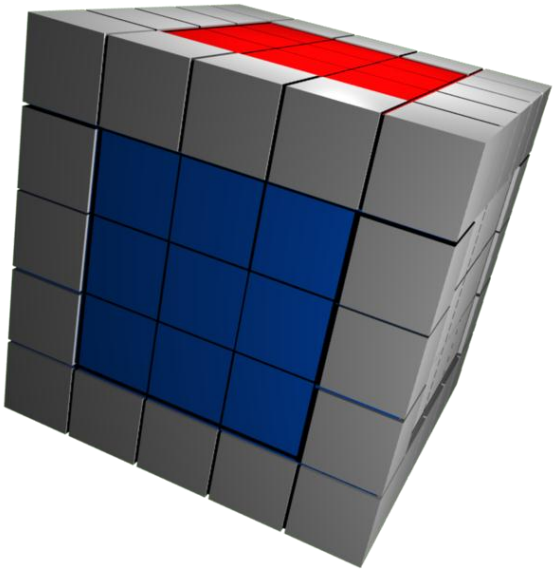


Ilustración 36: 5x5 centros

   La manera habitual de proceder es hacer un centro, luego su opuesto, y luego los adyacentes.
   Si se tiene habilidad en este tipo de puzles, se pueden intentar hacer los centros de 2 en 2, o incluso todos a la vez.
2. *Hacer Las aristas:* En este caso nos encontramos con un elemento más que en el anterior cubo, es decir, cada arista (del de 3x3x3 resultante) va a estar compuesto de 3 aristas.
   La forma de juntar estas aristas es hacerlo de dos en dos. Es decir, juntamos dos aristas consecutivas, y luego le añadimos el tercero. Es como el proceder del anterior cubo, pero haciendo dos veces el mismo paso.
   Cuando completemos este paso tendremos el cubo transformado a un cubo de 3x3x3.

A diferencia de lo ocurrido con el anterior paso, las paridades que se pueden presentar aquí se resuelven en este paso, y luego sólo hay que hacer el cubo de 3x3x3 de manera normal. Realmente no es necesario tratar estos casos de forma aislada, puesto que procediendo de manera normal se pueden eliminar y no dan problemas adicionales. Por ello no se detallan los macrooperadores para esta parte. Los tipos de paridad son:

a. *Pieza central girada:* La pieza central que conforma una arista está girada con respecto a las dos piezas que tiene a sus lados, está mal orientada con respecto a las mismas. Hay que volverla a orientar intentando no afectar al resto del cubo, se puede conseguir con un poco de esfuerzo y sin conocimientos adicionales a los del cubo de 3x3x3, aunque haya algoritmos específicos para ello.
b. *Otros casos de paridad:* Se pueden presentar más casos parecidos al anterior (piezas intercambiadas de sitio, piezas que requieren una rotación para estar en su sitio…) para los que existen formas de resolverlos, pero no son necesarias ya que con los conocimientos del cubo de 3x3x3 se pueden resolver perfectamente, aunque se tarde un poco más.

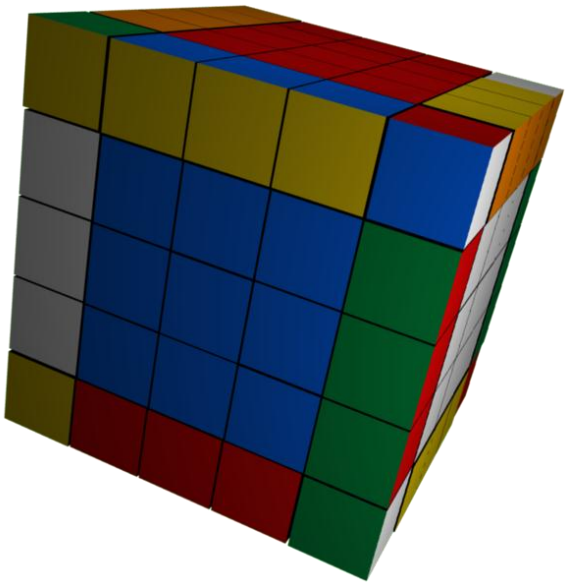


Ilustración 37: 5x5 transformado en 3x3

3. *Resolver el cubo de 3x3x3:* Llegados a este punto sólo queda resolver el cubo de 3x3x3 de la manera que habitual. Debido a la configuración del cubo, y a sus similitudes con el cubo de 3x3x3, no nos vamos a encontrar ningún caso de paridad ni ningún estado que no fuera posible en el cubo de 3x3x3.

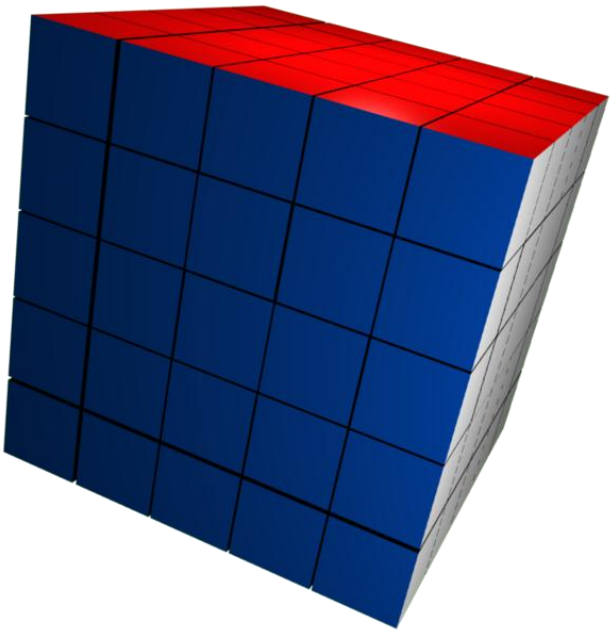


Ilustración 38: 5x5 resuelto

En el caso de este cubo el esfuerzo se centra antes de llegar al cubo de 3x3x3, puesto que tenemos que trabajar con más elementos para hacer los centros y Las aristas, pero una vez terminado, podemos proceder como en un cubo de 3x3x3 sin que se presenten problemas adicionales.

### 3.3.4. Cubos regulares de dimensiones superiores

Hoy día se pueden encontrar cubos regulares de hasta 11x11x11, muy laboriosos pero no tienen por qué ser más difíciles que los cubos ya vistos.

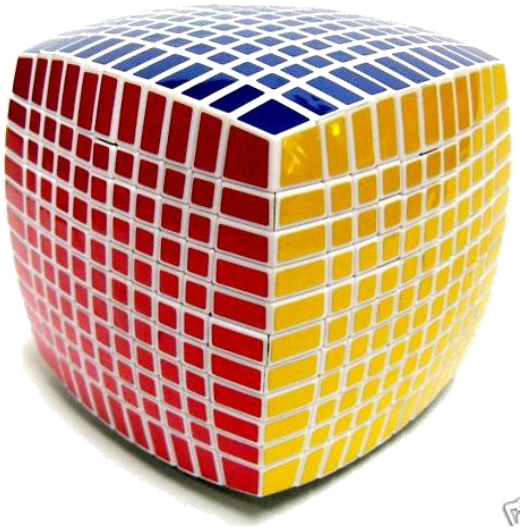


Ilustración 39: Cubo 11x11x11

El incluir más elementos hace que el espacio de estados aumente exponencialmente, pero el proceso a seguir es similar a lo ya visto. Transformarlo en uno de 3x3x3, resolver el de 3x3x3 y tratar los problemas de paridad en el proceso (en caso de ser un cubo de dimensiones

pares, se tratan al final de todo el proceso, y de ser de dimensiones impares se tratan antes de empezar a resolver el de 3x3x3).

Pero hay más puzles que han nacido del cubo de rubik clásico y que no tienen forma de cubo, o que la tienen pero pueden perderla.

### 3.3.5. Combinaciones en cubos regulares

Se ha obtenido una fórmula que calcula el número de estados existente en un cubo regular, sea cual sea su dimensión. La fórmula tiene en cuenta las particularidades de los cubos, estados repetidos, estados similares, estados imposibles y las normas de la combinatoria.

$$\frac{8!3^7(12n-24\lfloor\frac{n}{2}\rfloor)!2^{10(n-2\lfloor\frac{n}{2}\rfloor)}(24!)^{\lfloor\frac{n-2}{2}\rfloor(1+\lceil\frac{n-2}{2}\rceil)}}{24^{6\lfloor\frac{n-2}{2}\rfloor\lceil\frac{n-2}{2}\rceil}(24+46\lfloor\frac{n}{2}\rfloor-23n)}$$

Es una fórmula muy extensa, pero que nos sirve para calcular los estados de cualquier cubo de nxnxn, siendo n el número de elementos en una arista.

Se puede simplificar la fórmula si hacemos distinciones entre caso par y caso impar:

- *Caso par:*

$$\frac{8!3^7(24!)^{\frac{n(n-2)}{4}}}{24^{\frac{3n^2-12n+14}{2}}}$$

- *Caso impar:*

$$\frac{8!12!3^7 2^{10}(24!)^{\frac{n^2-2n-3}{4}}}{24^{\frac{3n^2-12n+9}{2}}}$$

Hemos de tener en cuenta que el caso par presenta menos combinaciones imposibles, frente al caso impar que tiene muchas.

Existen muchos puzles inspirados en el cubo de rubik a parte de los ya vistos, que suponen una dificultad añadida o un diseño curioso. Vamos a ver brevemente los más comunes:

### 3.3.6. Cuboku

Es un cubo de rubik de 3x3x3 normal, salvo porque tiene todas las caras del mismo color, pero con números. Se trata de tener un "sudoku" en cada cara, es decir, que para que el cubo esté resuelto hay que conseguir que no haya ningún número repetido en todas las caras.

Ilustración 40: Cuboku

Está construido de tal forma que sólo exista una solución posible. Aquí los centros no nos indican el color de la cara, sino el número que ya está en la cara.

Este cubo es curioso porque para un ordenador sería igual de fácil de resolver que uno de 3x3x3 normal, mientras que para una persona es mucho más difícil.

Además, al terminar el cubo podemos encontrarnos con la sorpresa de que algún número central está girado con respecto al resto, y para dejarlo de verdad resuelto habría que girarlo.

### 3.3.7. Cubo puzle

Es un cubo de rubik que en lugar de colores, cada cara forma un puzle. En principio tiene la misma dificultad que un cubo de 3x3x3 normal, pero al llegar al final nos podemos encontrar con la sorpresa de que la pieza central está girada con respecto al resto.


Ilustración 41: Cubo puzle

Para rotar una pieza central sin alterar el resto del trabajo hecho, se pueden aplicar operadores cíclicos (que repetidos un número de veces vuelven a dejar el cubo como al principio) que en muchos casos rotan la pieza central. El operador conocido como la T es uno de los más populares.

### 3.3.8. Void cube

Es un cubo de rubik normal, pero hueco por el centro. Tiene la dificultad añadida de que no tenemos los colores centrales como referencia. Es muy interesante el mecanismo ideado, puesto que en el cubo de rubik normal tiene una serie de ejes en los centros.

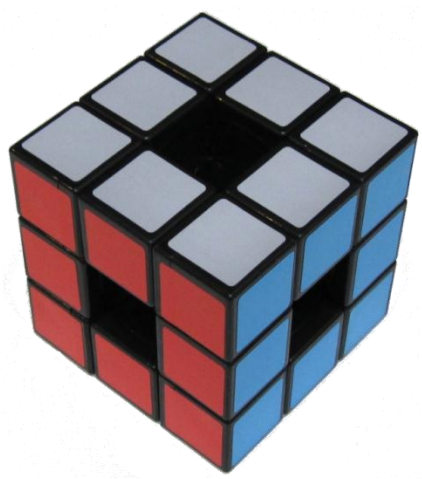


Ilustración 42: Void cube

También hay que tener en cuenta que estados que no serían solución en un cubo normal, aquí sí. Tener en cubo de rubik normal un estado como éste:

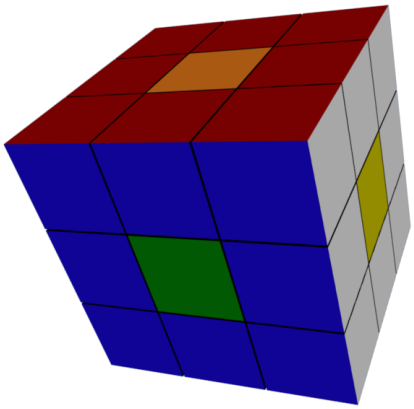


Ilustración 43: 3x3 void

Sería una solución válida en un void cube.

### 3.3.9. Mirror cube

Otro cubo de 3x3x3 que, en principio, tiene la misma dificultad que uno normal, pero que en lugar de colores juega con dimensiones, Es decir, el tamaño de las piezas no es el mismo, y el cubo desarmado puede tener más aspecto de una pequeña bola que de un cubo. La dificultad radica en que hay que tener una buena visión espacial para saber situar los elementos.

Ilustración 44: Mirror cube

### 3.3.10. Megaminx

Un puzle que llama la atención por su forma de dodecaedro, y que da una imagen de ser mucho más complicado que los puzles en forma de cubo. Pero el número de posibles estados se acerca mucho más al cubo de 4x4x4 que al de 5x5x5, y queda muy lejos de cubos de dimensiones superiores.

La forma de resolverlo es muy similar a la del cubo de rubik, ampliándolo para resolver más capas. Con los conocimientos necesarios para resolver el cubo de rubik una persona podría resolver el megaminx.

Ilustración 45: Megaminx

### 3.3.11. Cubo de engranajes

Es un cubo de 3x3x3 pero que tiene una serie de engranajes que conectan las piezas, haciendo que giren con ellas. Al girar una cara, también se ven afectadas otras piezas, y además podemos tener giros incompletos, es decir, una arista que está girado 90 grados con respecto a su posición inicial, de esta forma sus colores no estarían en ninguna cara. Desde un punto de vista informático sólo habría que añadir esas posiciones giradas 90 grados como elementos independientes, es decir, Las aristas pasarían a tener 4 elementos.

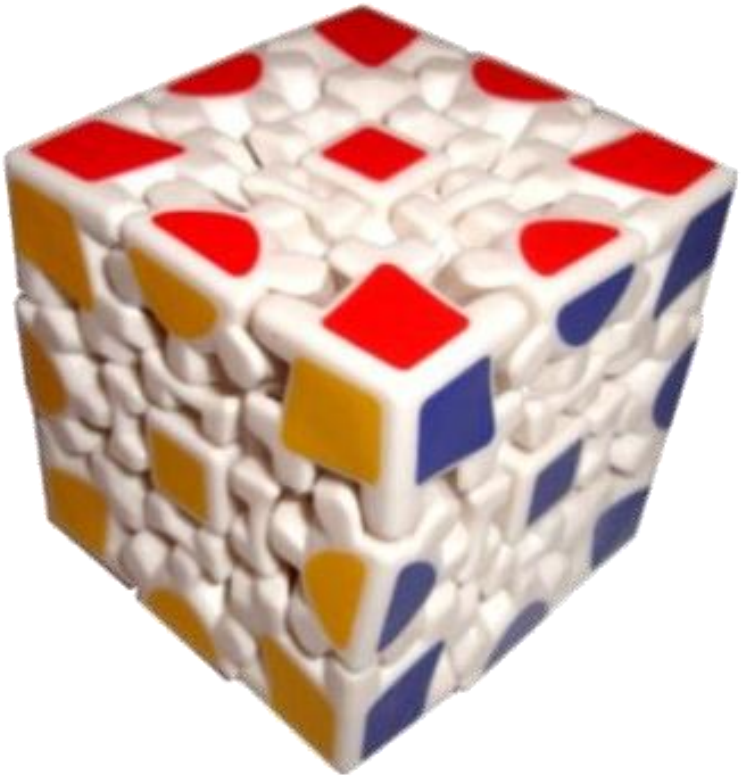


Ilustración 46: Engranajes

### 3.3.12. Piraminx

Pirámide de rubik con 4 caras que tienen 9 elementos cada una. Es decir, es como un cubo, pero con 18 elementos menos. Su forma, que se sale del cubo habitual, puede dar una falsa impresión de que es un cubo difícil, pero no es así, es más fácil que el cubo normal. El record del mundo de velocidad de resolver un pyraminx es de 1'92 segundos.

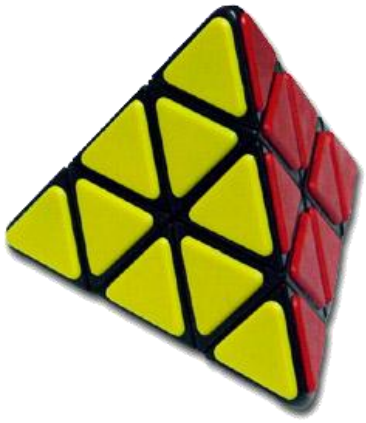


Ilustración 47: Piraminx

## 4. Objetivos

Se pretende desarrollar una aplicación que resuelva el cubo de rubik de 3x3x3 en imitación al modelo de iniciados para resolverlo. Dicha aplicación será muy rápida, por lo que estamos sacrificando eficiencia para obtener eficacia.

En las competiciones oficiales existentes para armar un cubo de 3x3x3 el tiempo de resolución es muy rápido, por lo que vamos a estudiar las formas que existen para que las personas resuelvan el cubo. Se espera que con la capacidad de cálculo de un ordenador normal, se resuelva un cubo antes que una persona.

Hay que decir que los métodos existentes para que las personas puedan resolver el cubo de rubik tampoco lo resuelven de manera óptima, porque se buscan ciertas reglas para hacer que lo pueda memorizar fácilmente una persona. No se ha conseguido encontrar un método que sea fácil de aprender para una persona que siempre encuentre la solución óptima.

Existen dos métodos, el de principiantes y el de avanzados, cada uno con sus características:

- *Principiantes:* Siete pasos fáciles de recordar para iniciarse con el cubo.
  - Son siete pasos: cruz en la cara superior, completar la capa superior, completar la capa media, cruz en la cara inferior, orientar la cruz, colocar las esquinas de la capa inferior y permutar las esquinas de la capa inferior.
  - El trabajo realizado en cada paso se preserva en los siguientes pasos, por eso los pasos son más específicos y costosos según se avanza
  - Los movimientos necesarios para resolver el cubo rondan los 100
  - El tiempo para resolver un cubo usando exclusivamente este método es de 1 a 5 minutos (dependiendo de la experiencia).
- *Expertos:* Método pensado para personas que ya dominan el de principiantes y quieren hacerlo más rápido. Implica aprenderse de memoria muchos movimientos.
  - Son tres pasos: cruz en la capa superior, completar capa intermedia y superior y hacer la capa inferior (orientación y permutación).
  - El trabajo realizado en cada paso se preserva en los siguientes pasos, y además se quiere avanzar mucho en cada paso.
  - Hay que aprenderse muchos operadores, es un método más difícil de memorizar.
  - Los movimientos necesarios para resolver el cubo son aproximadamente 55.
  - El tiempo de resolución va de 5’66 segundos (actual record mundial) a 40 segundos. Cuando se empieza con este método puede ser más tiempo.

Una persona que quiera resolver el cubo suele empezar por el método de principiantes y luego va añadiendo operadores del método de expertos, hasta que consigue aprendérselo.

El método elegido es el de principiantes por varias razones:

- El número de macrooperadores que hay que codificar es menor.
- Cada paso requiere un número pequeño de comprobaciones. El método de expertos requiere muchas comprobaciones en sus pasos. Al ser más pasos pero con menos comprobaciones se espera tardar menos.

Para comprobar la eficacia de la aplicación se va a desarrollar otra aplicación que resuelve de manera óptima el cubo de rubik de 2x2x2. Se espera que, al ser óptimo, tarde más que el del cubo de 3x3x3, ya que tiene que hacer una búsqueda más exhaustiva.

Dicha aplicación hará una búsqueda en amplitud de la solución, se puede hacer la búsqueda de esa manera debido al reducido espacio de estados. Al ser una búsqueda en amplitud, y teniendo que todas las operaciones tienen igual coste, la búsqueda es admisible.

# 5. Desarrollo cubo de 3x3x3

A continuación se describen paso a paso el proceso seguido para crear la aplicación que resuelve de manera sub-óptima el cubo de rubik de 3x3x3.

Para su resolución sub-óptima vamos a seguir el algoritmo en 7 pasos conocido por ser el de principiantes, por lo que vamos a dividir la resolución en etapas, resolviendo cada una de ellas como si fueran sub-problemas.

De esta forma partimos de un cubo totalmente desordenado. En las sucesivas etapas el cubo irá acercándose a la solución, y el resultado de la última etapa será el cubo resuelto, y el camino seguido a través de todas las etapas serán las operaciones necesarias para resolver el cubo partiendo del inicial.

## 5.1. Primer paso

El primer paso consiste en conseguir una cruz en la cara superior, es decir, que todas las aristas de la cara superior deben estar bien colocadas (en su sitio) y bien orientados (con sus "cubitos" apuntando a su cara), se pretende llegar al estado que ilustra la siguiente imagen. Las piezas en gris están así porque no nos importa su posición, solo nos importa para este paso la posición de las piezas coloreadas.

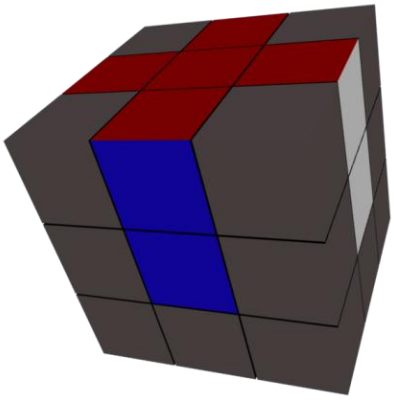


Ilustración 48: Cruz de la cara superior

La "distancia" estimada a la solución parcial es la mayor de las distancias de cada uno de las cuatro aristas. La distancia de una arista es el número de operaciones básicas necesarias para llevar esa arista a su posición final, y bien orientado.

Dicha distancia es una aproximación, y siempre será menor o igual a la distancia real.

### 5.1.1. Heurística

Se dan cuatro casos, que la distancia sea 0, 1, 2 o 3 y, por simetría, dichos casos son iguales para los cuatro aristas que estamos tratando en esta parte. Para los ejemplos visuales nos centraremos en La arista rojo-azul

- Caso 0: Las aristas están bien colocados, por lo que la distancia (estimada y real) es 0.

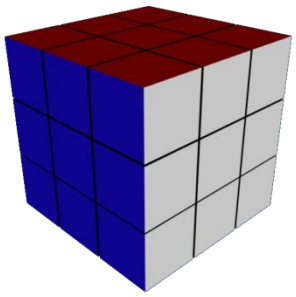


Ilustración 49: Cubo resuelto

- Caso 1: La arista está a un giro de estar bien colocado, esto pasa si está bien orientado en una de las dos caras a las que pertenece. Con un movimiento básico estará en su posición final.

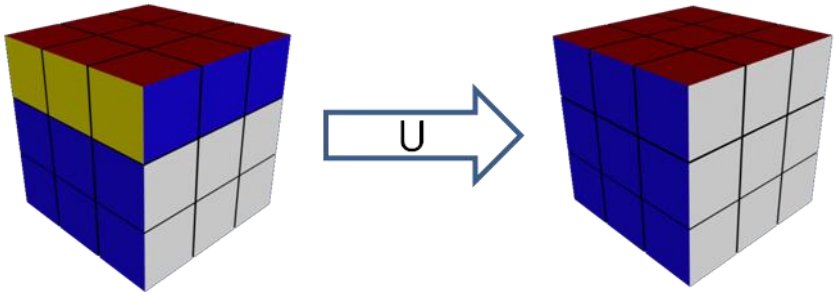



Ilustración 50: Operación U

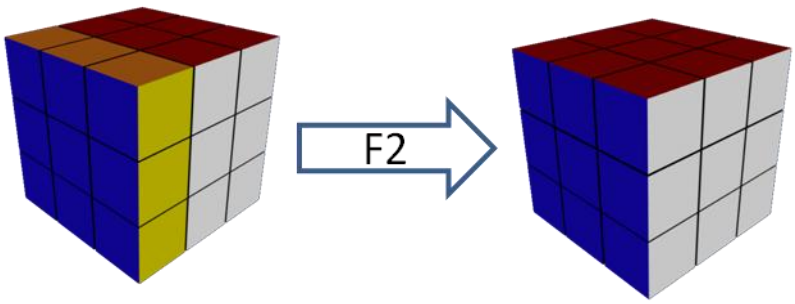



Ilustración 51: Operación F2

*Observar que en este caso La pieza del ejemplo (azul-rojo) se encuentra en la cara inferior.*

- Caso 2: La arista está a dos giros de estar bien colocado, esto es:
  - La arista está mal orientado en una de sus caras, sin estar ni en su propia posición ni en la opuesta,

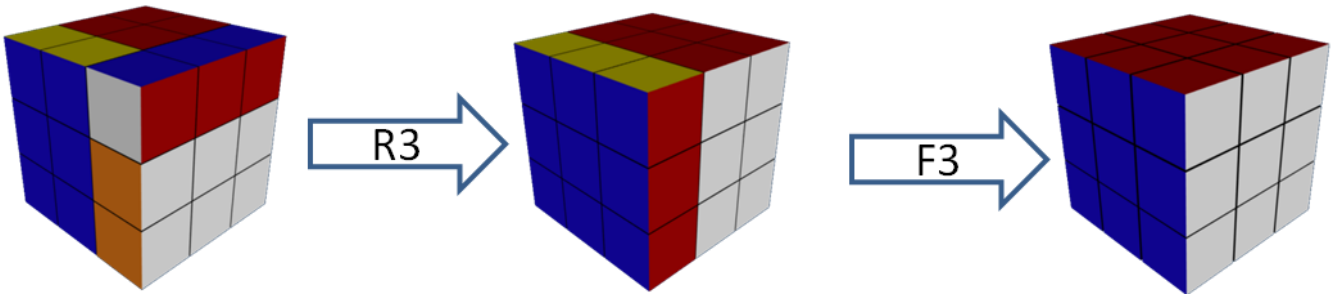



Ilustración 52: Operaciones R3 F3

*La otra opción de este caso es simétrica, la pieza se encontraría en la cara opuesta con la misma orientación en sus colores (azul en U y rojo en L).*

  - La arista está bien orientado en alguna de las caras a las que no pertenece (esto implica que ninguno de sus "colores" está en la cara que le corresponde).

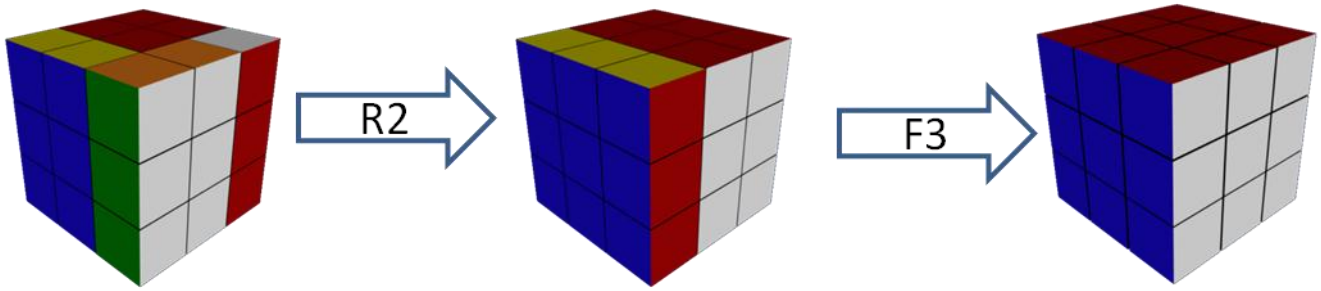



Ilustración 53: Operaciones R2 F3

- Caso 3: La arista se encuentra en alguno de los casos no citados anteriormente, es decir:
    - La arista está en su posición o en la opuesta en alguna de sus caras y mal orientado.

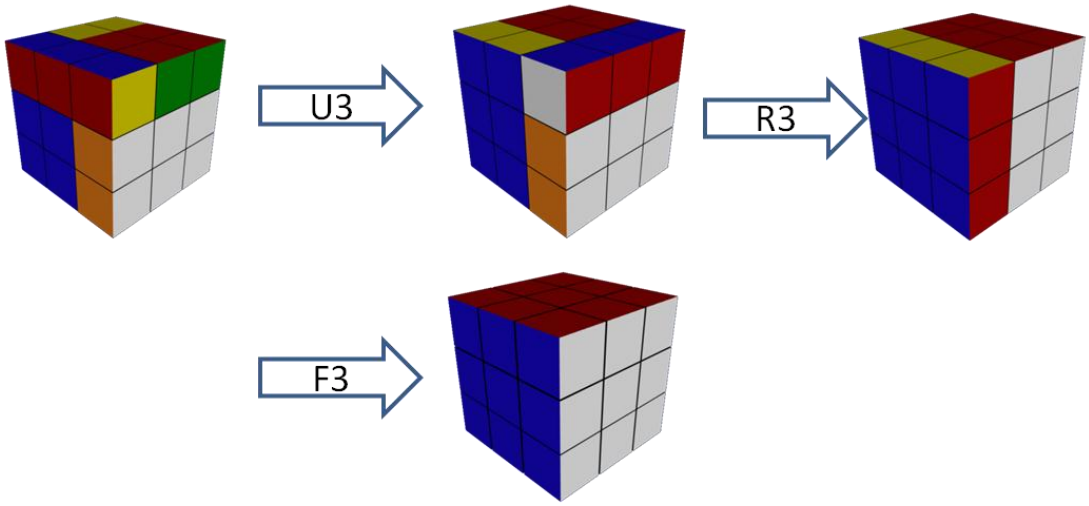



Ilustración 54: Operaciones U3 R3 F3

    - La arista está mal orientado en alguna de las caras a las que no pertenece.

### 5.1.2. Análisis

Una arista puede estar en 12 posiciones distintas, con 2 orientaciones distintas, esto es, puede estar en un total de 24 posiciones. De dichas 24 posiciones tenemos:

- Caso 0: 1 posición cumple con este caso, que esté en su sitio y bien orientado
- Caso 1: 6 posiciones cumplen con este caso.
- Caso 2: 13 posiciones cumplen con este caso, es el caso más probable.
- Caso 3: 4 posiciones cumplen con este caso, es el caso menos probable (descontando el caso 0).

Si tomamos todas estas posibilidades tenemos que:

$$\left(\frac{(0 \cdot 1) + (1 \cdot 6) + (2 \cdot 13) + (3 \cdot 4)}{24}\right) = 1.83333$$

Es decir que una arista estará prácticamente a 2 operaciones de estar bien colocada.

Multiplicar el resultado anterior por 4 no nos da el número medio de movimientos necesarios para acabar la primera parte, hemos de tener en cuenta que un movimiento afecta a todo el cubo.

El número máximo de movimientos para terminar la primera parte es de 8. Dada cualquier configuración del cubo, la cruz en la cara superior estará a 8 movimientos o menos. Durante las pruebas de esta aplicación hemos obtenido que el número medio de movimientos es 4.

La heurística elegida es admisible pero poco informada, puesto que sólo podemos tener los valores 0, 1, 2, 3. Teniendo en cuenta que el factor de ramificación es de 23 (cada nodo va a generar 23 sucesores) nos encontraremos muchos casos con la misma heurística en el mismo nivel de profundidad.

Por ello se va a utilizar una heurística más informada pero **no admisible** para ordenar los nodos con el mismo valor, de esta forma se intenta que el nodo elegido como siguiente candidato a ser explorado tenga más información que la aportada por la heurística.

La heurística no admisible utilizada es la suma de las distancias de las cuatro aristas a colocar. Así que cada nodo tendrá dos valores en los que nos apoyaremos para elegir el siguiente candidato a expandir.

### 5.1.3. Estructura de los datos: lista cerrada

La lista cerrada contendrá los nodos ya estudiados. Los nodos se irán insertando al final de la lista, y cada nodo apunta a su padre. De esta forma cuando lleguemos a la solución sólo hay que recorrer en orden inverso desde la solución hasta el estado inicial y así hallamos el camino a la solución.

Por ejemplo, de tener un árbol en este estado (en rojo los nodos explorados, en azul el nodo final):

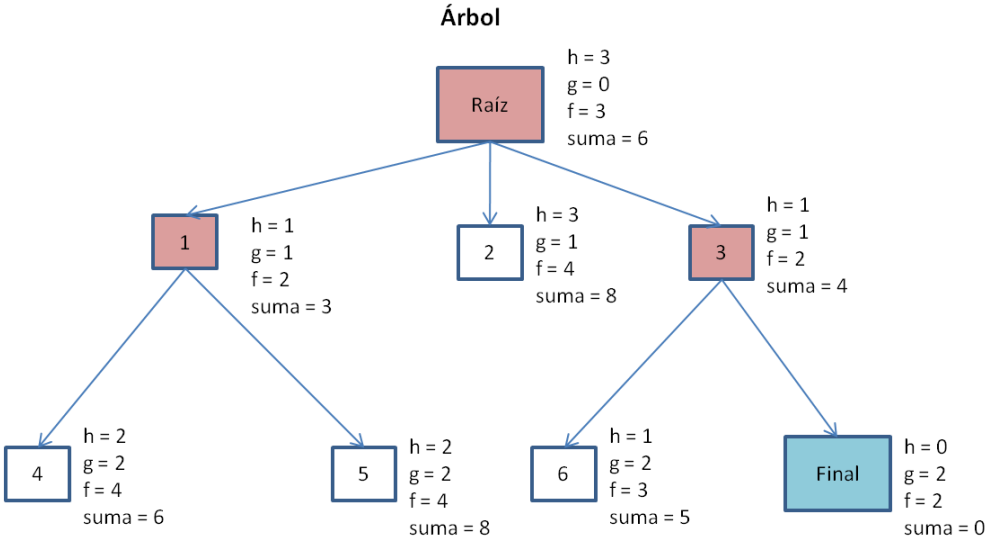



Ilustración 55: Árbol lista cerrada

La lista cerrada sería como se muestra a continuación:

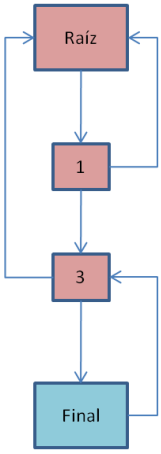



Ilustración 56: Lista cerrada

Suponemos para este ejemplo que primero se estudia el nodo raíz, luego el 1, el 3 y el final. Se ve que cada nodo apunta al siguiente elemento insertado en la lista, y cada elemento apunta a su padre. Así, partiendo del nodo final vamos a su padre, subiendo hasta llegar al nodo raíz y así tenemos el camino óptimo.

### 5.1.4. Estructura de datos: lista abierta

La lista abierta va a ser una lista de listas. Como hemos visto vamos a tener muchos nodos con el mismo valor para f, que ordenaremos según la heurística no admisible. De esta forma tenemos una lista general para los valores f, y otra ordenada para un mismo valor de f. El propósito de hacer la lista así es poder realizar inserciones en tiempo constante.

La lista abierta va a manejar dos heurísticas, una admisible pero poco informada, y otra no admisible pero más informada. Inicialmente los nodos se van ordenando según el valor de f directamente. Es decir, el valor de f indica la posición que va a ocupar en dicha lista. Una vez en dicha posición, los valores con esa misma f se ordenan según el valor de la heurística no admisible.

Los nodos se van insertando en orden, y tienen un puntero a su padre (que estará en la lista cerrada).

Por ejemplo, para este árbol:

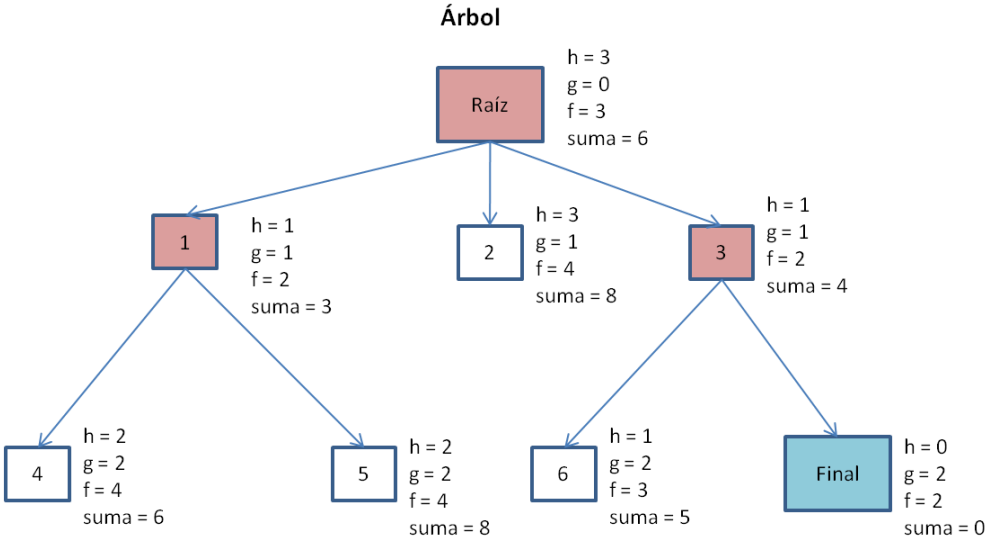



Ilustración 57: Árbol lista abierta

Tendríamos esta estructura que se indica a continuación, los nodos en verde son los que pasan de la lista abierta a la cerrada, y se borrarían de la lista abierta:

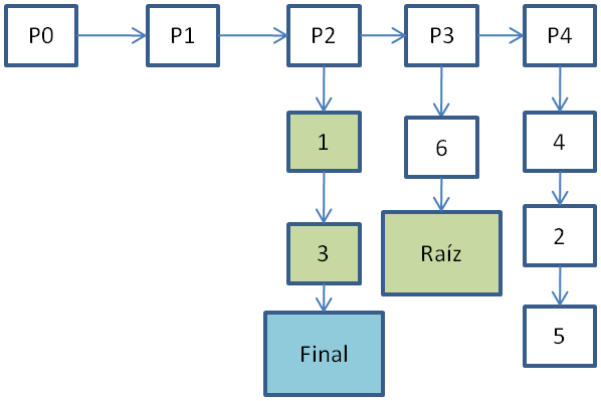



Ilustración 58: Lista abierta

Los nodos marcados en verde se han mantenido para mostrar cómo funciona.

### 5.1.5. Funcionamiento

Se evalúa el estado inicial para ver si es la solución, en caso de no serlo, se inserta en la lista cerrada y se expanden sus hijos, evaluándolos. Los hijos se insertan ascendentemente en función de f en la lista abierta.

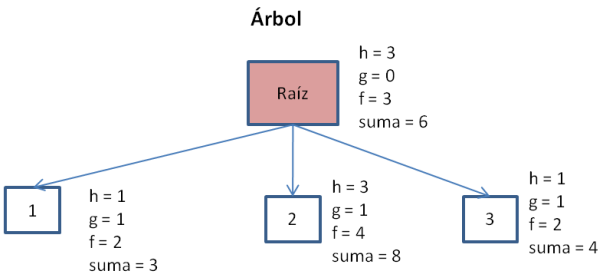



Ilustración 59: Árbol ejemplo 1

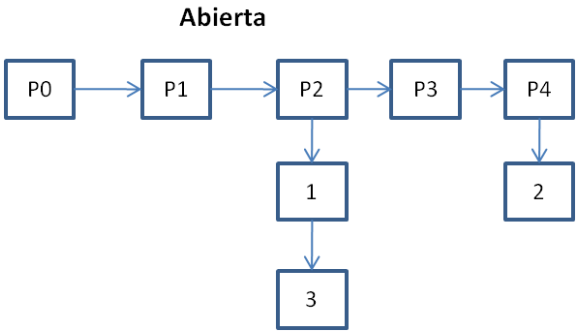



Ilustración 60: Abierta ejemplo 1

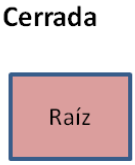



Ilustración 61: Cerrada ejemplo 1

Buscamos la primera posición en abierta que tenga elementos (la P2) y cogemos el primer elemento. Insertamos ese nodo en la lista cerrada, lo eliminamos de la lista abierta, expandimos sus hijos y los insertamos ordenados en la lista abierta. En este caso hay un empate en la P4, puesto que el nodo 2 y el 5 tienen la misma suma. Si se dan empates se inserta el nodo más nuevo después de los que estaban en la lista con su mismo valor.

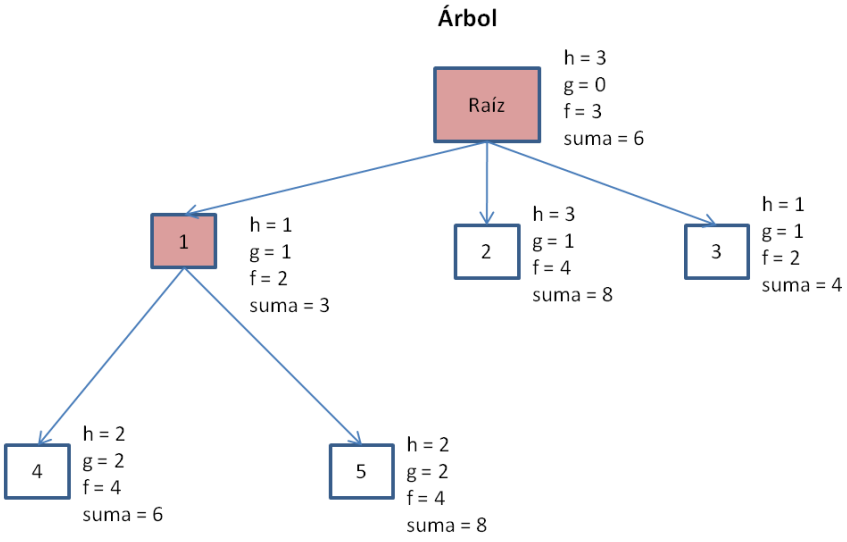



Ilustración 62: Árbol ejemplo 2

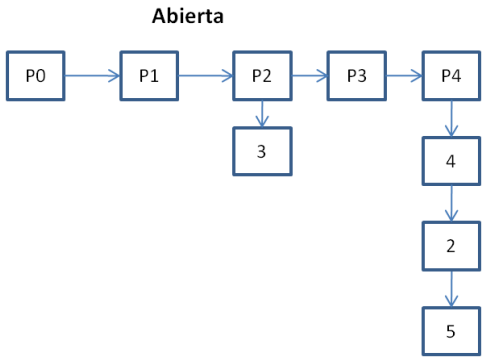



Ilustración 63: Abierta ejemplo 2

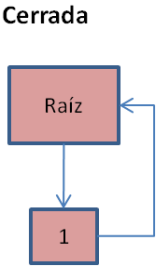



Ilustración 64: Cerrada ejemplo 2

El siguiente nodo a estudiar es el 3, repetimos los pasos: ponemos el 3 en la lista cerrada, eliminándolo de la abierta, expandimos los nodos del 3 y los guardamos en la lista cerrada en orden.

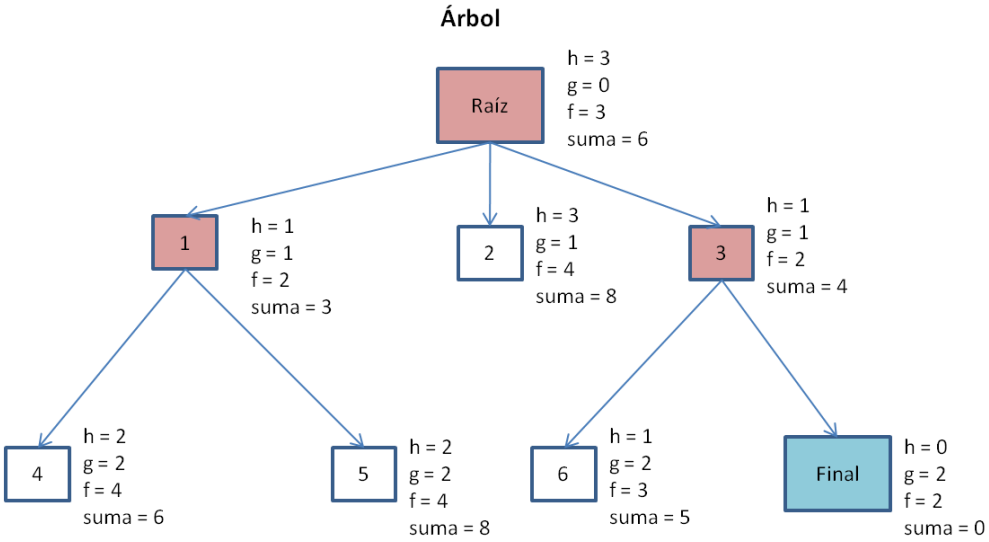



Ilustración 65: Árbol ejemplo 3

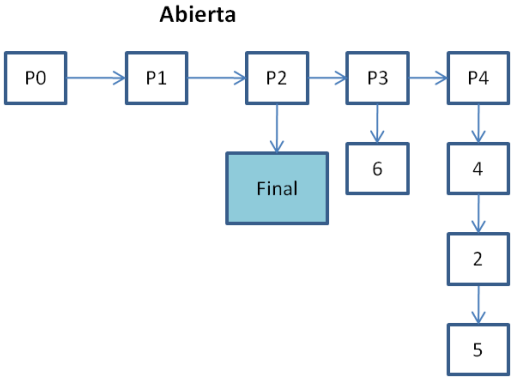



Ilustración 66: Abierta ejemplo 3

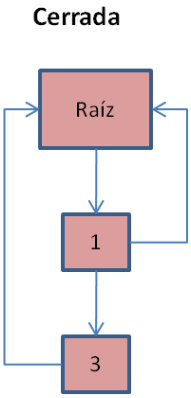



Ilustración 67: Cerrada ejemplo 3

En el último paso estudiamos el siguiente elemento de la lista abierta, que en este caso es un nodo final. Se inserta en la lista cerrada y se da por finalizada la búsqueda.

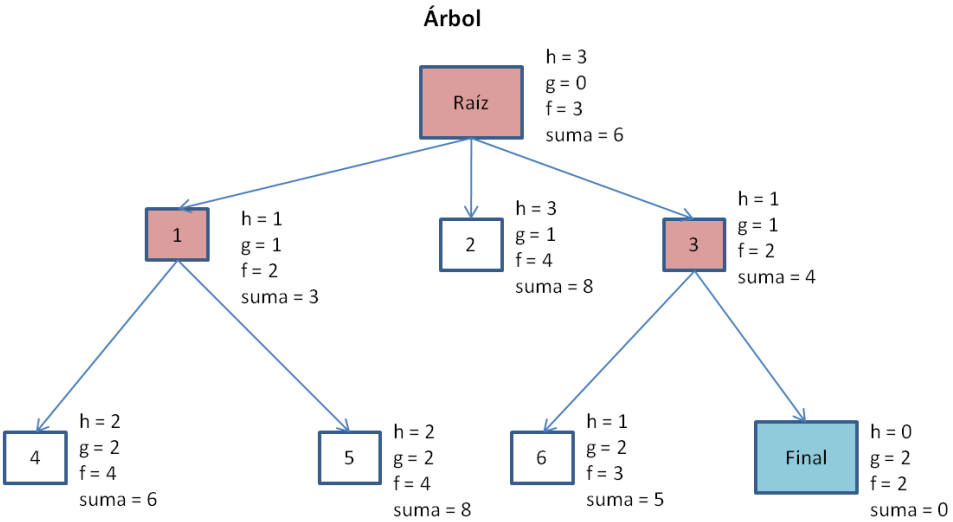



Ilustración 68: Árbol ejemplo 4

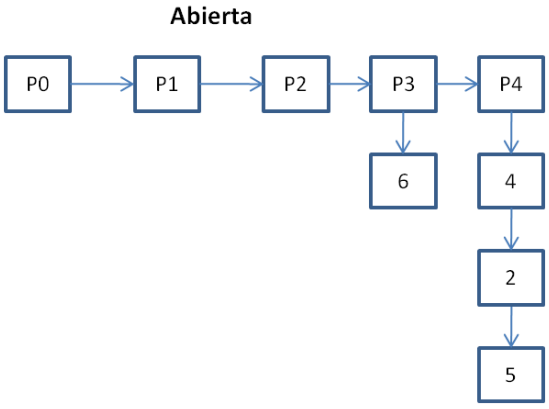



Ilustración 69: Abierta ejemplo 4

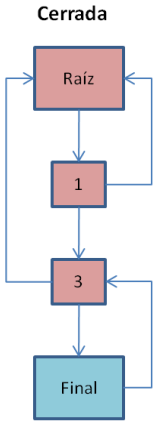



Ilustración 70: Cerrada ejemplo 4

Recorremos la lista cerrada desde el nodo final hasta el nodo raíz y obtenemos un camino que es la solución óptima.

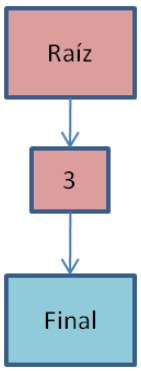



Ilustración 71: Solución ejemplo

### 5.1.6. Optimización del funcionamiento:

A la hora de implementar el algoritmo A* se ha añadido cierta funcionalidad con el objetivo de optimizar recursos.

- *Eliminar estados repetidos:* al realizar operaciones en el cubo de rubik es común encontrarnos con estados que ya habíamos estudiado previamente.

Por ello se eliminan de la lista abierta aquellos estados que ya estén en la lista cerrada.

- *Insertar en cerrada el nodo raíz:* el proceder normal sería insertar el nodo raíz en la lista abierta, para luego pasarlo a la cerrada. Aquí directamente se inserta en la lista cerrada, y luego sus hijos se evalúan y se insertan en la lista abierta. Realmente esta mejora no supone una optimización apreciable.
- *Evitar operaciones no trascendentes:* si una operación no permuta Las piezas que queremos colocar en la cara superior, el estado correspondiente no se inserta en la lista abierta. Se ha comprobado que en casos concretos esta medida reduce sensiblemente el tiempo de ejecución.

### 5.1.7. Observaciones

- El número máximo de operaciones obtenido en las pruebas de esta parte es 8.
- Debido a la heurística usada podemos asegurar que esta parte es óptima.

## 5.2. Segundo paso

Una vez completada la cruz en la cara superior, el siguiente paso es completar la capa superior colocando las esquinas que quedan en su posición y bien orientados.

Llegaremos a este estado cuando acabemos el segundo paso. La capa superior está completa y bien orientada.

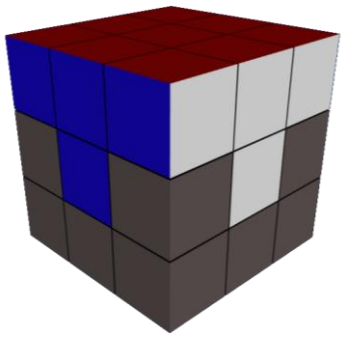


Ilustración 72: Capa superior

### 5.2.1. Proceso

Ahora ya tenemos colocados 4 piezas en su sitio, y queremos colocar otros 4, y así completar la capa superior y pasar al siguiente paso.

Hemos de tener en cuenta que hay que preservar la cruz de la cara superior así que los movimientos necesarios para colocar una esquina no deben alterar la cruz, es decir, cuando una esquina esté colocada en su sitio, la cruz ha de seguir igual.

Al tener que conservar el trabajo realizado al finalizar el paso, los movimientos necesarios para colocar una pieza no son tan fáciles de realizar como pueda parecer. Por ello se han codificado macrooperadores para esta parte.

En este caso los macrooperadores están pensados para colocar una arista que esté situado justo debajo de la posición que le corresponde.

Por lo tanto, antes de aplicar un macrooperador debemos realizar una serie de operaciones simples para colocar una pieza en posición de aplicar un macrooperador.

### 5.2.2. Operaciones simples

Son las operaciones necesarias para colocar una pieza en la posición deseada para aplicar un macrooperador. No están codificadas en un solo operador, debido a su simpleza se ha decidido aplicar las operaciones simples correspondientes.

A la hora de colocar una esquina en la posición necesaria, nos da igual su orientación, sólo nos interesa su posición. Lo que si tenemos que tener en cuenta es que tenemos que alterar la capa superior lo menos posible, y tenemos que preservar la cruz en la cara superior conseguida en el paso 1.

Para colocar una arista nos podemos encontrar 4 casos (son 4 casos para cada esquina, sólo se muestran 4 como ejemplo.)

- Queremos bajar la esquina a la posición directamente inferior, conservando el resto de la cara superior:

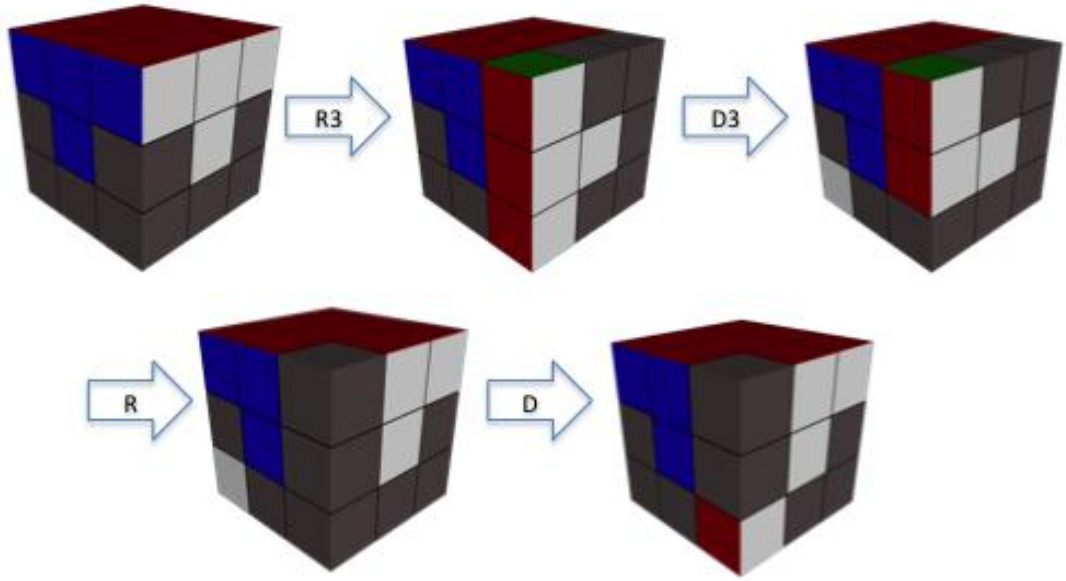



Ilustración 73: Bajar pieza caso 1

- Queremos colocar la esquina en la cara inferior en una posición que no es la inmediatamente inferior:

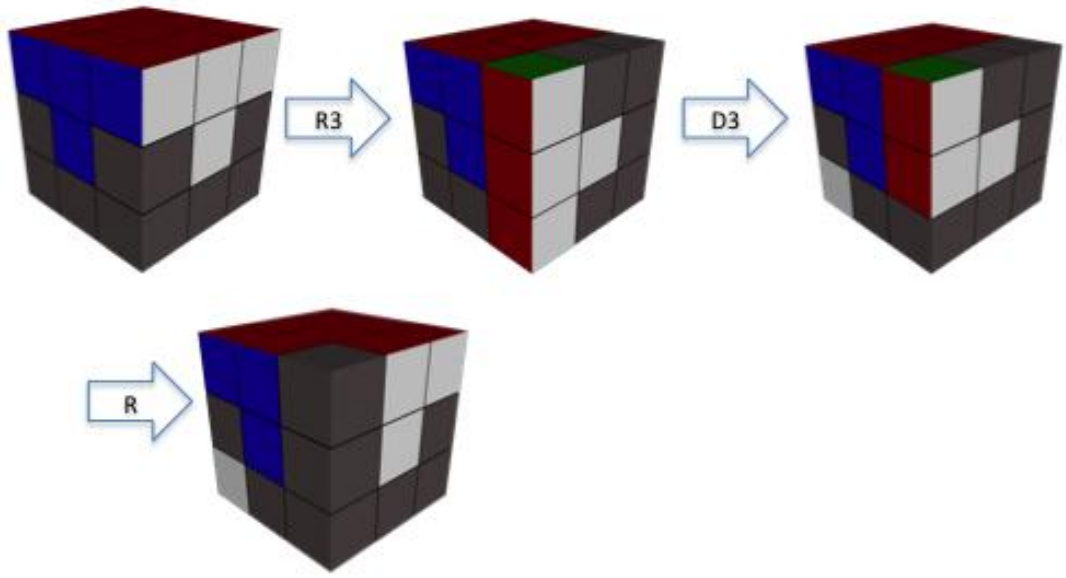



Ilustración 74: Bajar pieza caso 2

Nótese que estos ejemplos son ilustrativos, sólo queremos mostrar cómo se baja una esquina hasta la posición deseada. Hay más casos pero son similares a los dos mostrados.

Tenemos estos dos casos para colocar y bajar una esquina a su posición. Si queremos bajar la esquina a la capa inferior, a la posición inmediatamente inferior, necesitaremos 4 movimientos.

En caso de querer colocarlo en cualquiera de las restantes posiciones válidas de la capa inferior, usaremos 3 movimientos.

En caso de que la esquina esté colocada en la capa inferior, sólo habrá que girar dicha capa hasta que la esquina quede bien colocado para aplicar un macrooperador, con un solo movimiento.

Las operaciones para colocar una esquina en una posición que permita aplicar un macrooperador no modifican el resto de la capa superior, es decir, si queremos bajar una esquina, sólo se verá afectada la posición de la esquina que baja, y el resto de la capa superior no se verá afectada.

### 5.2.3. Macrooperadores

Se trata de agrupar un número determinado de operaciones simples conocidas en un solo operador, que tendrá coste computacional de una operación, aunque englobe varios movimientos.

Esto nos ayuda a optimizar el tiempo de resolución del cubo, puesto que para un estado determinado, aplicamos un macrooperador, que coloca una o varias piezas en la posición deseada.

Por lo tanto, en este paso, en lugar de buscar en un árbol, vamos a analizar las condiciones para que se aplique un operador u otro. La pieza ya está colocado en el sitio

deseado para aplicar un macrooperador, pero se aplica uno u otro dependiendo de su orientación.

Los macrooperadores son:

- 1: D3R3DR

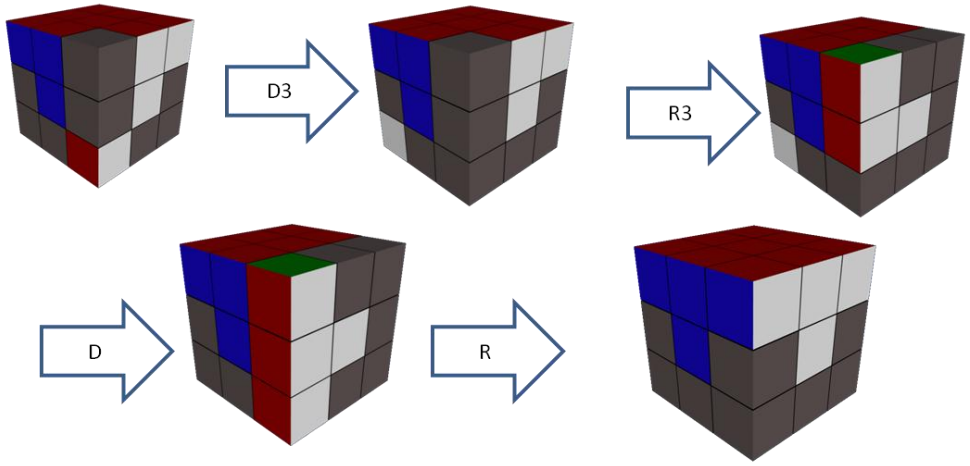



Ilustración 75: Paso 2 macrooperador 1

- 2: DFD3F3

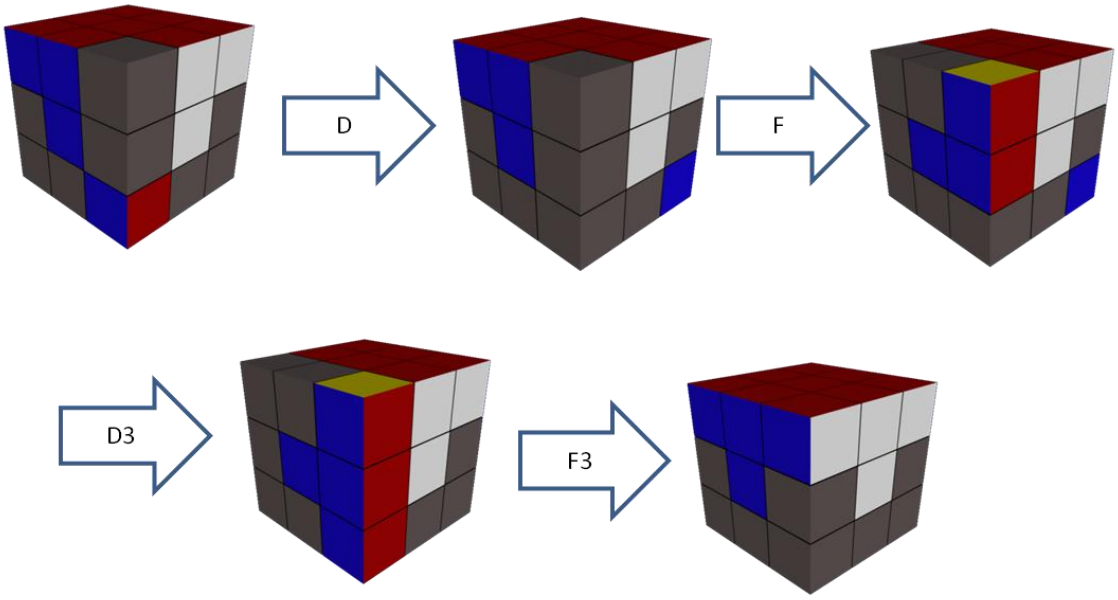



Ilustración 76: Paso 2 macrooperador 2

- 3: DRD3R2DR

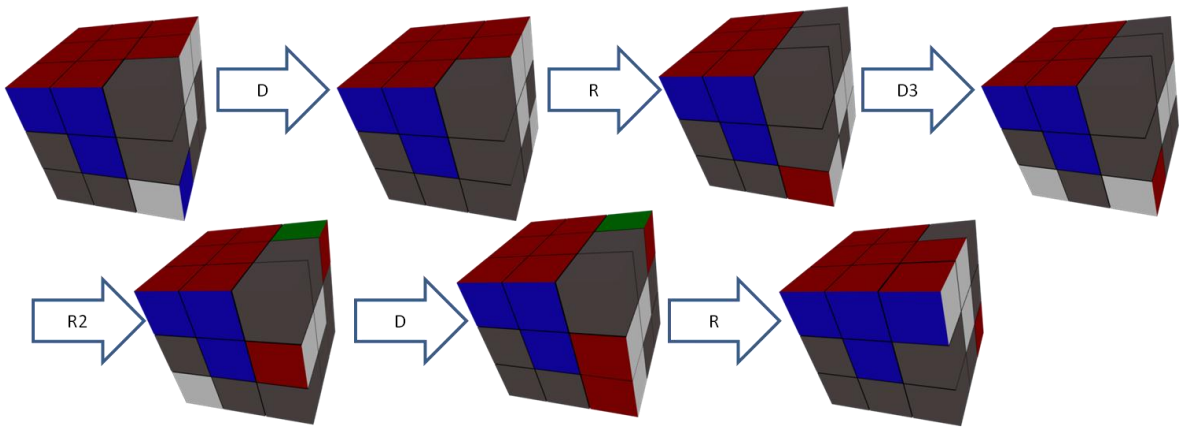



Ilustración 77: Paso 2 macrooperador 3

Se puede apreciar que el tercer macrooperador afecta a otra esquina, dejando la cruz intacta. Esto quiere decir que tendremos que hacer varias pasadas, para asegurarnos de recolocar las esquinas que se hayan podido descolocar.

No hay peligro de entrar en un bucle infinito ya que una arista que se descoloque por aplicar este macrooperador no va a necesitar en ningún caso el tercer macrooperador, por lo tanto sólo se requerirá un macrooperador extra (el primero o el segundo, nunca el tercero) por cada vez que apliquemos el tercer macrooperador.

### 5.2.4. Pasos a seguir

Observamos que los macrooperadores sólo pueden usarse si la esquina que queremos colocar está en la cara inferior y si su posición final es la inmediatamente superior. Así que antes de aplicar los macrooperadores es necesario bajarlos y situarlos en la cara inferior y en la posición necesaria para aplicar el macrooperador.

También observamos que al bajar una arista, otro arista situado en la capa inferior sube, por lo que si intentamos hacer un primer paso bajando todos Las aristas primero, nos podemos topar con bucles infinitos o con una cadena larga de operaciones que podría reducirse aplicando otro criterio. Bajar una arista sólo altera ese arista que queremos bajar de la cara superior, por lo que si tenemos Las aristas bien colocados en la cara superior, no se verán afectados

Por otro lado, se observa que el operador 3º afecta a otro esquina de la capa superior, por lo que nos podemos encontrar que al aplicar este operador se descoloque una pieza bien colocado. Lo más lógico sería aplicar primero el operador 3º, y luego el resto, pero de esa forma es más costoso por el número de comprobaciones que tenemos que añadir.

Una forma de solucionar esto sería bajar uno a uno todas Las aristas a su posición, comprobar si alguno requiere el macrooperador 3º y aplicarlo en ese caso. El problema es la pérdida de eficiencia al tener que repetir muchas operaciones para realizar esta operación la primera.

Por lo tanto, de cara a la eficiencia en la resolución, hemos optado por considerar todos los macrooperadores iguales, y el proceso a seguir es el siguiente:

1. **Colocar una pieza en su posición:** seleccionamos una arista de la capa superior que no esté en su posición. La selección no tiene ningún criterio en especial. En este paso se trata de colocar La arista en la posición que debe estar para aplicar un macrooperador sin alterar la capa superior. Para ello necesitaremos de 1 a 4 movimientos.
2. **Selección de operador:** Una vez dicho pieza está en la posición necesaria para aplicar un macrooperador, vemos qué macrooperador necesita para estar bien colocado en la capa superior, y aplicamos dicho operador.
3. **Repetir los pasos 1 y 2 para todas Las aristas mal colocados:** repetimos los pasos 1 y 2 para colocar todas Las aristas en la capa superior.
4. **Repetir pasos 1 a 3 si es necesario:** como el macrooperador 3º puede descolocar una arista que estaba en su sitio, volvemos a comprobar si todos Las aristas están bien colocados, en caso contrario repetimos los pasos 1 al 3. Cabe destacar que la arista descolocada a causa del macrooperador 3º no va a necesitar dicho macrooperador para colocarse en su sitio, por lo que como mucho repetiríamos los pasos 1 al 3 una vez.

### 5.2.5. Observaciones

Al realizar muchas ejecuciones de este paso observamos que:

- La capa superior queda resuelta, en la mayoría de los casos, aplicando de 10 a 20 operadores simples.
- Este segundo macrooperador tiene una media de 16 movimientos.
- Hay casos raros y extremos, en que la capa superior queda resuelta en 2 movimientos, o que queda resuelta en 28.
- También es un caso muy raro que necesitamos repetir todos los pasos del paso 2 porque una arista haya quedado mal colocado a causa del operador 3º.

## 5.3. Tercer paso

Llegados a este punto queremos completar la capa intermedia, dejando así el cubo.

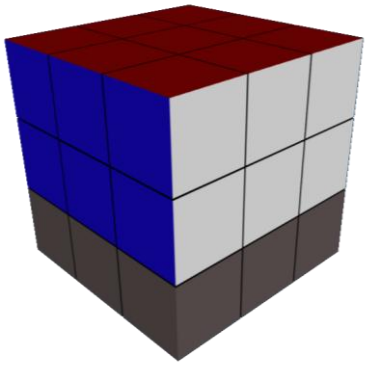


Ilustración 78: Capa media

Como se observa no se quiere alterar en ningún momento lo hecho hasta ahora (pasos 1 y 2), y ahí radica la dificultad de los sucesivos pasos, puesto que al tener más piezas colocadas donde deseamos, mover el resto sin alterar lo hecho es más complejo.

Por lo tanto la complejidad de los operadores aumenta, pues hay que hacer muchos movimientos para colocar Las piezas en la posición deseada, manteniendo las dos capas ya hechas.

Pero al tener ya muchas piezas colocados en su posición, las posibles posiciones del resto de piezas se reducen considerablemente.

En el anterior paso había tres macrooperadores porque, una vez colocado la pieza en su sitio, podía tener 3 orientaciones. En este caso trabajamos con aristas que sólo tienen 2 posibles orientaciones, así que una vez colocados en la posición deseada sólo habrá dos posibles macrooperadores atendiendo a su orientación.

### 5.3.1. Proceso

El proceso es similar al anterior, tenemos que aplicar un macrooperador a cada pieza para colocarlo en su posición y bien orientado.

Para poder aplicar un macrooperador necesitamos que La pieza esté colocada en una posición determinada, así que lo primero será colocarla para poder aplicarlo.

El proceso va a ser el mismo. Seleccionamos una pieza, la colocamos en la cara inferior, en la posición necesaria para aplicar un macrooperador, y aplicamos el macrooperador.

En este caso el proceso para colocar una pieza en posición de aplicar un macrooperador es más costoso, por ello se han codificado macrooperadores auxiliares que agrupan dichas operaciones.

A diferencia del paso anterior, todos los macrooperadores tienen el mismo número de operaciones, y no alteran el trabajo realizado, por lo que no va a ser necesario repetir el proceso, es decir, con hacer el proceso 4 veces (una por pieza) ya habremos terminado.

### 5.3.2. Operaciones simples

Necesitamos bajar La arista que queremos colocar a la cara inferior, y de ahí colocarlo en la posición necesaria para que se pueda aplicar un macrooperador.

En este caso esta operación es algo compleja, puesto que queremos bajar una arista sin alterar el trabajo ya hecho y sin alterar Las aristas que ya hayamos colocado.

Para hacer esto necesitamos aplicar 7 operaciones simples para bajarlo y, ya que la cara inferior no nos importa que se altere, otra más para colocarlo en el sitio deseado con una operación de la cara inferior.

Pero observamos que 7 movimientos para colocarlo en la cara inferior son muchos giros, así que para mejorar la eficiencia del programa vamos a crearnos otros 4 macrooperadores (uno por cada posición de la capa intermedia donde puede estar una arista) que hagan estos movimientos.

Dichos 4 macrooperadores son equivalentes, y serían similares a este:

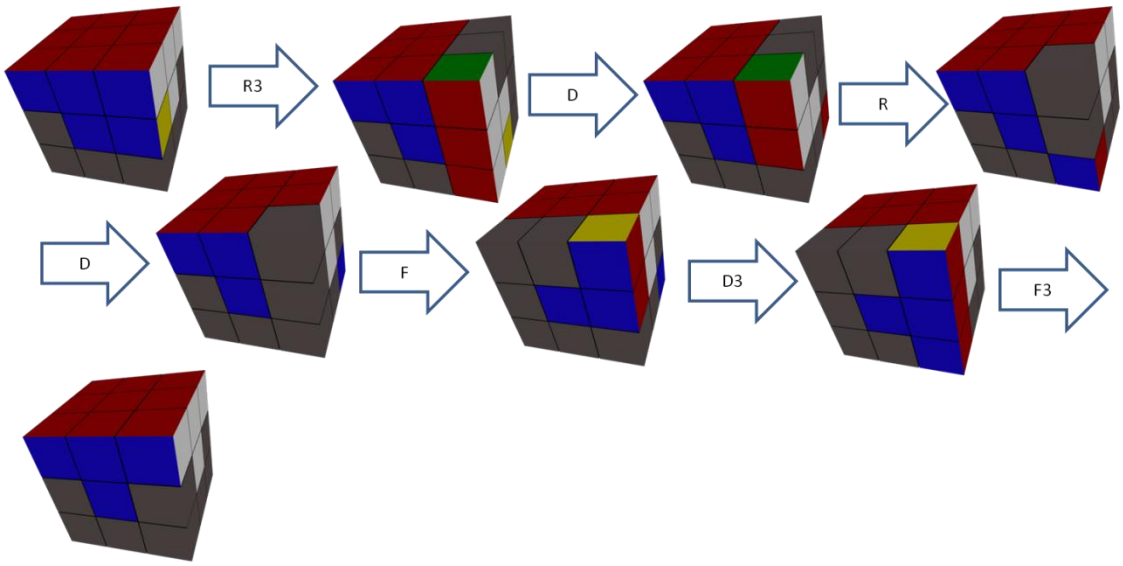



Ilustración 79: Paso 3 operaciones

Finalizados estos movimientos la arista que estaba mal colocada (en la capa intermedia) pasa a estar en la capa inferior, no se aprecia en la imagen superior, pero si giramos el cubo 180 grados se ve:

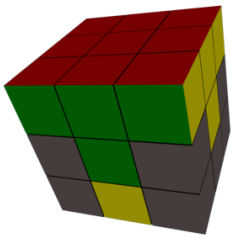


Ilustración 80: Esquina capa inferior

Una vez aplicado este macrooperador (si es necesario hacerlo) pasamos a colocar La arista que estamos tratando en la posición requerida para aplicar uno de los macrooperadores del siguiente paso.

Por lo tanto tenemos 3 opciones:

- La pieza está ya bien colocado para aplicar el macrooperador, 0 operaciones.
- La pieza está en la capa inferior: 1 operación.
- La pieza está en la capa intermedia: 7 operaciones (si termina bien colocado) u 8 operaciones (si después de aplicar el macrooperador necesita colocarse)

Se observa un salto muy importante en el número de operaciones necesarias.

### 5.3.3. Macrooperadores

Una vez colocado en la posición necesaria, aplicamos el macrooperador correspondiente según su orientación. Como Las aristas tienen 2 *"caritas"* tenemos 2 macrooperadores posibles para cada arista, lo que nos da 2 macrooperadores. El resto son equivalentes a los mostrados.

En este caso los dos macrooperadores no alteran lo que ya tenemos colocado del cubo (ni los pasos 1 y 2, ni lo que llevemos hecho del 3), por lo que una vez aplicados a los 4 piezas ya tendremos la capa intermedia completa, así que no hace falta repetir.

Los macrooperadores son los siguientes:

- 1: D3R3DRDFD3F3

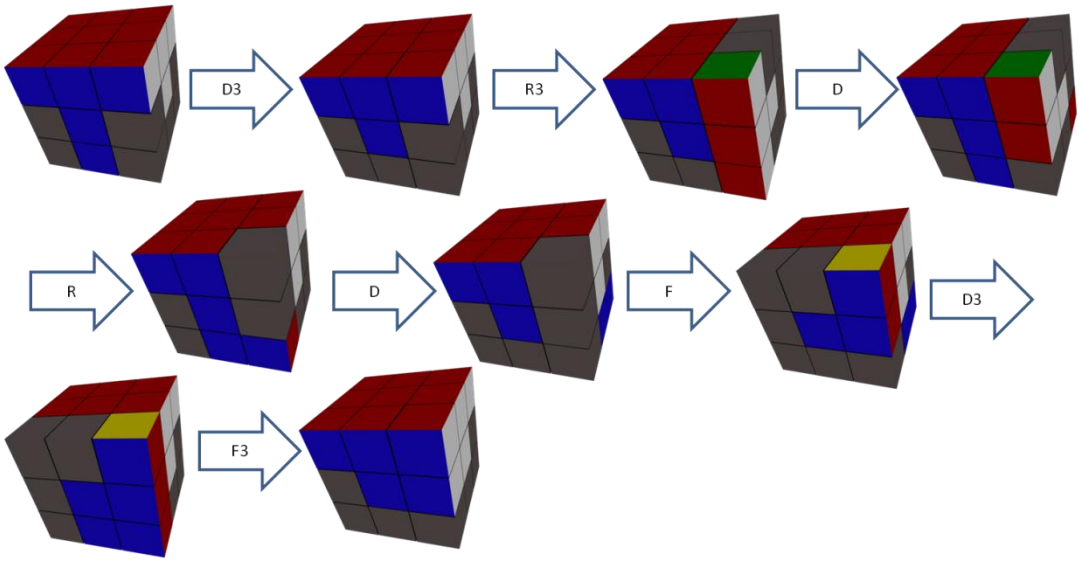



Ilustración 81: Paso 3 macrooperador 1

- 2: D2FD3F3D3R3DR

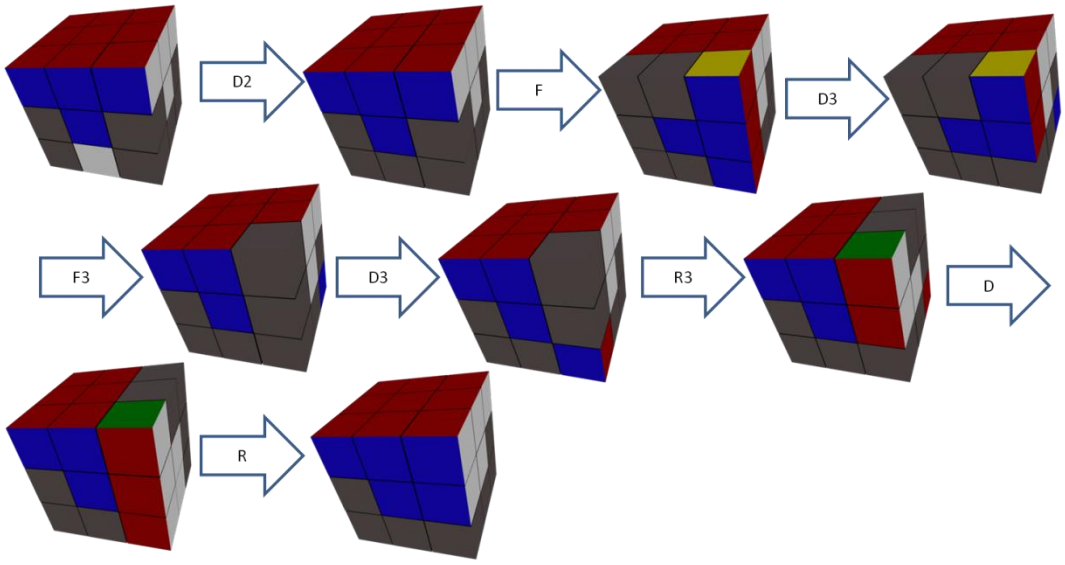



Ilustración 82: Paso 3 macrooperador 2

### 5.3.4. Pasos a seguir

Como se ha dicho anteriormente los pasos a seguir no es necesario repetirlos como hicimos en el 2º paso. Cuando una pieza esté bien colocada no cambia de posición mientras operamos con el resto de piezas, de esta forma, al aplicar los pasos a las 4 piezas, tendremos la capa intermedia bien colocada, sin alterar los pasos 1 y 2, y sin necesidad de repetir el proceso, por lo que los pasos a seguir son:

1. **Bajar una pieza:** si la pieza que queremos operar está en la capa intermedia, pero sin colocar en su posición, o en su posición pero mal orientado, debemos bajarla a la cara inferior, usando el macrooperador creado a tal efecto.

2. **Colocar una pieza:** si la pieza está en la cara inferior, pero no está en la posición requerida para usar un macrooperador, se le coloca en dicha posición con una sola operación, un giro de la cara inferior.
3. **Aplicar macrooperador:** una vez en el sitio deseado aplicamos el macrooperador correspondiente, y La pieza acabará bien colocado. Una vez hechos estos tres pasos con todos Las piezas, tendremos la capa intermedia hecha.

### 5.3.5. Observaciones

Este paso es uno de los que más movimientos consumen, teniendo una media aproximada de 30 movimientos. Esto se debe a varios factores:

- *Preparación:* Los movimientos necesarios para colocar una pieza en una posición que permita usar un macrooperador son muchos. En el resto de los pasos no son necesarios tantos movimientos de preparación.
  Además, debido a las posibles posiciones que puede adoptar en este punto, una arista tiene un 43'7% de posibilidades de necesitar este posicionamiento previo, eso quiere decir que hay muchas posibilidades de que alguno de los cuatro aristas que queremos colocar esté en esta situación.
- *Repetición:* Es necesario repetir todo el proceso para cada arista que no esté colocado en su sitio, es decir, que puede ser necesario repetir el proceso 4 veces. En el paso anterior también había que repetir el proceso, pero la preparación requería menos movimientos.
  En pasos sucesivos los macrooperadores agrupan más giros, pero basta con aplicar uno para terminar el paso, además la preparación es mucho menos costosa.
- *Datos:* Cuando queremos colocar una pieza en este paso nos podemos encontrar varios casos, según su posición, que hacen que necesite más o menos movimientos para estar en su sitio.
  - Mejor caso: La arista está colocado en su sitio→0 movimientos.
  - Caso medio: La arista está colocado en la capa inferior→1 movimiento para ponerlo en su sitio + 8 de macrooperador = 9 movimientos
  - Peor caso: La arista está en la capa intermedia pero no en su posición → 7 movimientos para bajarlo + 1 para posicionarlo + 8 de macrooperador = 16 movimientos.

Si el peor caso se repitiera para los 4 aristas, necesitaríamos un total de 64 movimientos para terminar este paso.

## 5.4. Cuarto paso

Llegados a este paso, ya tenemos las dos primeras capas completas, y sólo queda completar la última capa para terminar el cubo.

Pero hemos de tener en cuenta que ahora mismo nos encontramos a mitad de camino, aun queda mucho por hacer para tener el cubo colocado, aunque parezca que no. Este punto es bastante "peligroso" (desde el punto de vista de una persona) porque si se realiza mal algún paso podemos desordenar todo el cubo, y tendríamos que volver a empezar.

Sin embargo la complejidad computacional de resolver este cuarto paso es inferior a la de los anteriores pasos, puesto que tenemos movimientos muy limitados, y hay que hacer pocas comprobaciones en comparación.

Nuestro objetivo, llegados a este punto, es completar la cruz en la cara inferior, pero a diferencia del primer paso (cruz en la cara superior) sólo nos importa la cruz en la cara inferior, sin importar la posición relativa de las aristas inferiores con respecto al resto del cubo.

Sería de esta forma.

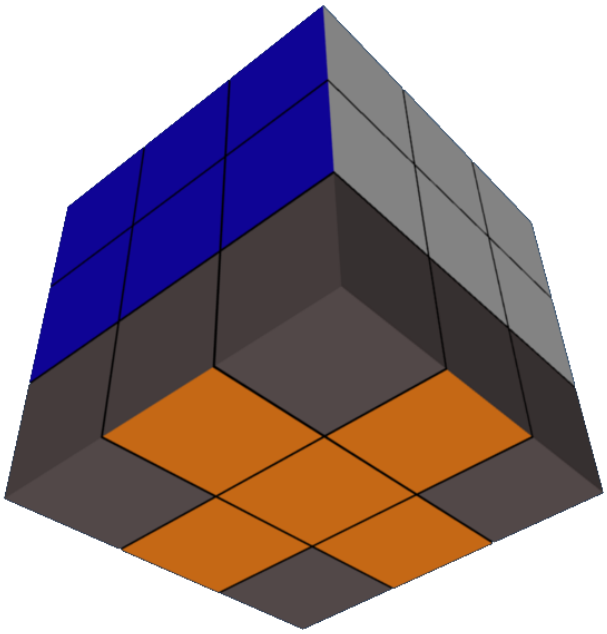


Ilustración 83: Cruz cara inferior

Al llegar a este paso, habiendo seguido los anteriores, nos encontramos con ciertas peculiaridades que nos harán el trabajo más fácil:

- **No es necesario colocar primero:** En los anteriores pasos necesitábamos colocar la pieza a tratar en una posición determinada para poder aplicarle un operador. En este caso no, las piezas ya están en la posición deseada.
- **Posibilidades restringidas:** Sólo es posible que tengamos bien orientados un número par de aristas. Es decir, podemos tener 0, 2 o 4. Esto se debe a que los posibles estados de un cubo de rubik vienen limitados por las características de sus operadores, y no son posibles todos los estados.
- **Macrooperadores de una pasada:** De los macrooperadores propuestos sólo hay que aplicar uno para conseguir el objetivo, es decir, son excluyentes. No es necesario aplicar más operadores ni hacer un bucle para repetir pasos. Esto hace que en este paso el número de operaciones sea bastante bajo a pesar del número de giros en cada macrooperador, ya que en los anteriores pasos podría ser necesario aplicar hasta 4 macrooperadores y, en algún caso especial, más. Aquí sólo se aplica uno.

Por lo tanto en este paso sólo hay que comprobar en qué estado nos encontramos y aplicar el macrooperador correspondiente a dicho estado

### 5.4.1. Macrooperadores

- 1: F3L3D3LDF

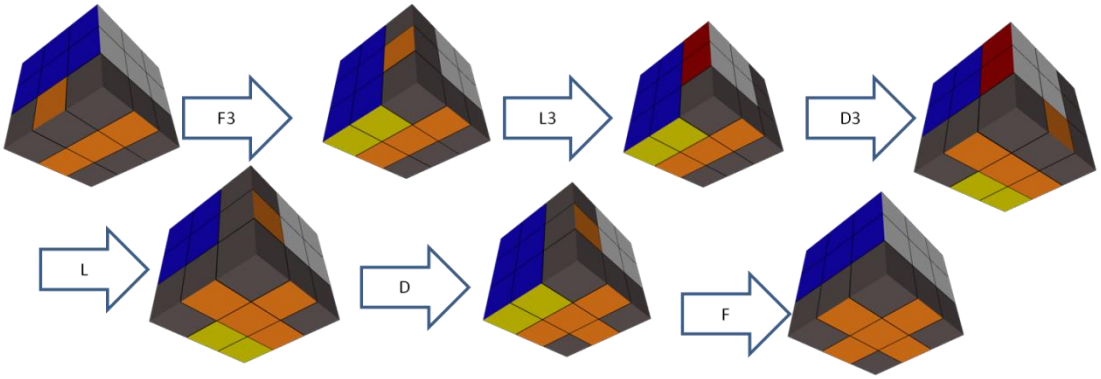



Ilustración 84: Cuarto paso macrooperador 1

Para este macrooperador observamos que sólo existen dos simetrías en lugar de cuatro, como venía pasando hasta ahora. Al mover todas Las piezas a tratar a la vez nos vamos a encontrar esta peculiaridad más de una vez.

- 2: BDRD3R3B3

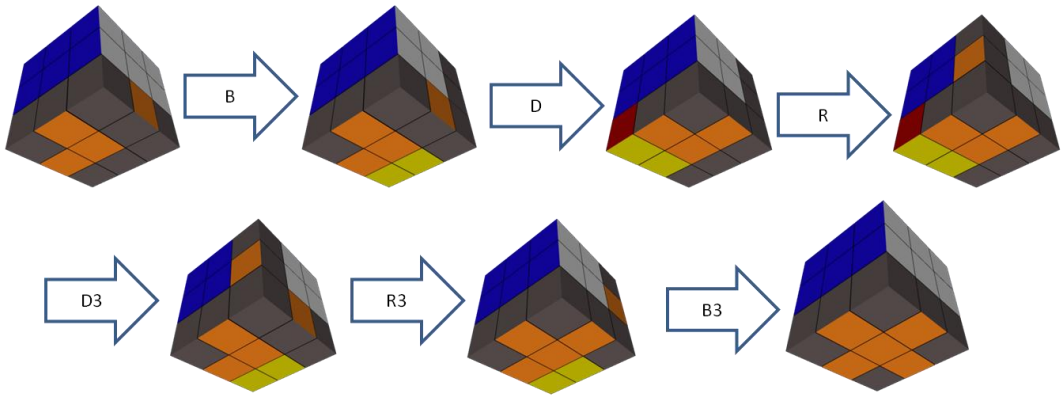



Ilustración 85: Cuarto paso macrooperador 2

En este macrooperador se observa que tenemos las 4 simetrías, por lo que tendremos 4 macrooperadores.

- 3: LR3DBD3B3D3B3D3BDL3R

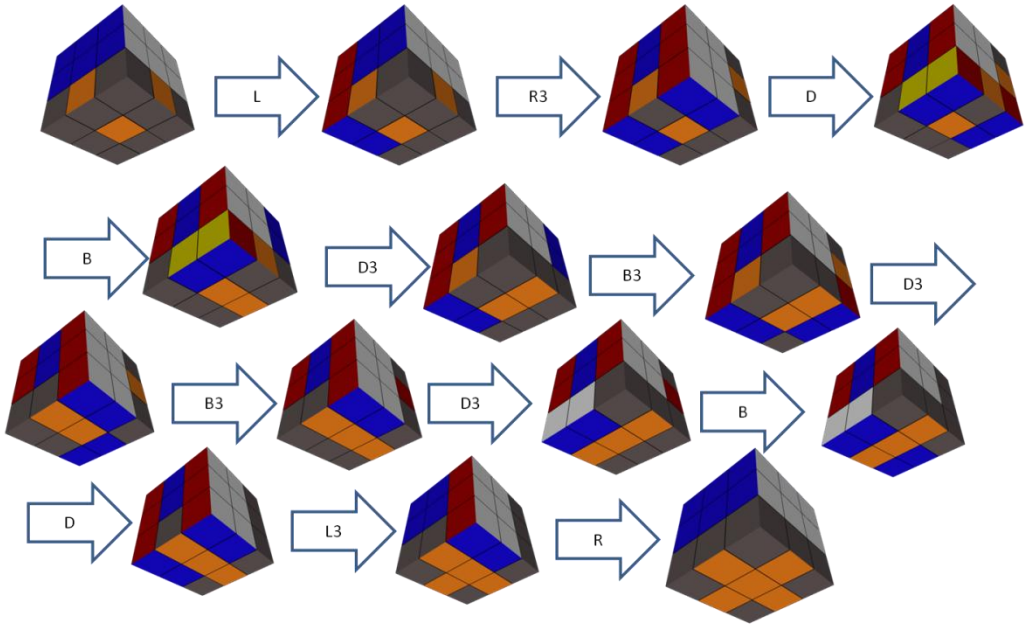



Ilustración 86: Cuarto paso macrooperador 3

En este caso solo tenemos un macrooperador, por lo que, aunque tenga muchas operaciones, será el menos probable.

### 5.4.2. Observaciones

Este es el primero de los pasos en los que sólo tenemos que aplicar un macrooperador, y no hay preparación previa necesaria. Esto sucederá con los siguientes pasos, sólo el paso quinto necesitará preparación (1 movimiento). Por ello se observa lo siguiente:

- Los distintos macrooperadores son muy diferentes dependiendo del caso en que nos encontremos. En el último caso su coste es muy superior al de los demás.

- Atendiendo a las piezas que queremos colocar en este paso, teniendo en cuenta las permutaciones imposibles presentes en el cubo, tenemos 8 posibles configuraciones que se resuelven con nuestros macrooperadores. Así se aplican:
  - *Colocado:* En 1 caso entre 8 llegaremos a esta parte con la cruz de la cara inferior hecha.
  - *Línea:* Caso en el que hay que aplicar el macrooperador 1. 2 posibilidades de 8.
  - *L:* Caso en el que hay que aplicar el macrooperador 2. 4 posibilidades de 8.
  - *Punto:* Caso en el que hay que aplicar el macrooperador 3. 1 posibilidad de 8.
- Debido a esa distribución de los posibles estados, el número medio de movimientos en el paso 4 está próximo a 6.
- Se observa que el número medio de movimientos desciende significativamente con respecto al paso anterior.

## 5.5. Quinto paso

En este paso se quiere completar la cruz en la cara inferior. Llegamos con las aristas de la cara inferior bien orientados (con el color correspondiente a la cara inferior puesto en dicha cara) pero en una posición que no es la suya.

En este paso se trata de completar la cruz en la cara inferior para que quede igual que la cruz de la cara superior (paso 1), y Las aristas estén bien orientadas y en su posición.

Se puede apreciar que, según se va avanzando, los pasos son más lentos. Para colocar la capa superior y la de en medio hemos necesitado 3 pasos, y para la última capa necesitamos 4 pasos. Esto se debe a que las operaciones que hacemos deben mantener todo el trabajo anterior, y ello tiene un coste más alto según se avanza.

El objetivo de este paso es terminar con el cubo así:

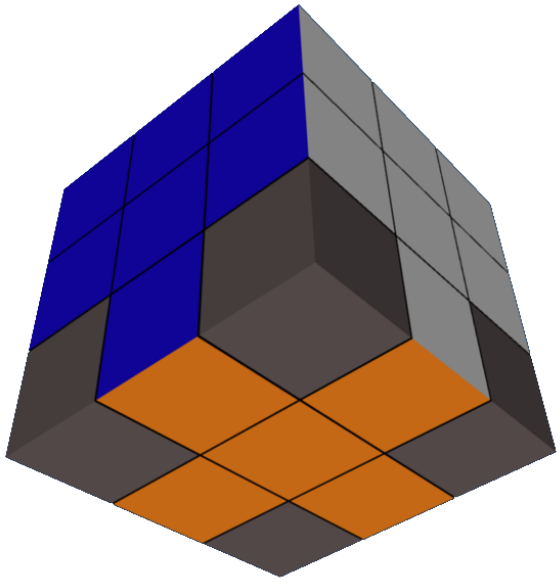


Ilustración 87: Cruz completa cara inferior

### 5.5.1. Proceso

Se necesita una mínima preparación para poder aplicar los macrooperadores, ya que éstos están pensados para colocar las aristas en su sitio si hay una ya colocada, o dos en posiciones opuestas. Por lo tanto es necesario girar la capa inferior un hasta que se dé uno de los casos de arriba.

Se puede apreciar que la preparación para usar los macrooperadores es mínima (1 operación).

También se observa que sólo será necesario aplicar un macrooperador para colocar todas las aristas en su sitio, por lo que no será necesario repetir pasos o aplicar más de un macrooperador.

Por lo tanto, al igual que con el paso anterior, aplicando un solo macrooperador habremos terminado, y no será necesario hacer más operaciones.

### 5.5.2. Macrooperadores

- 1: LD2L3D3LD3L3:

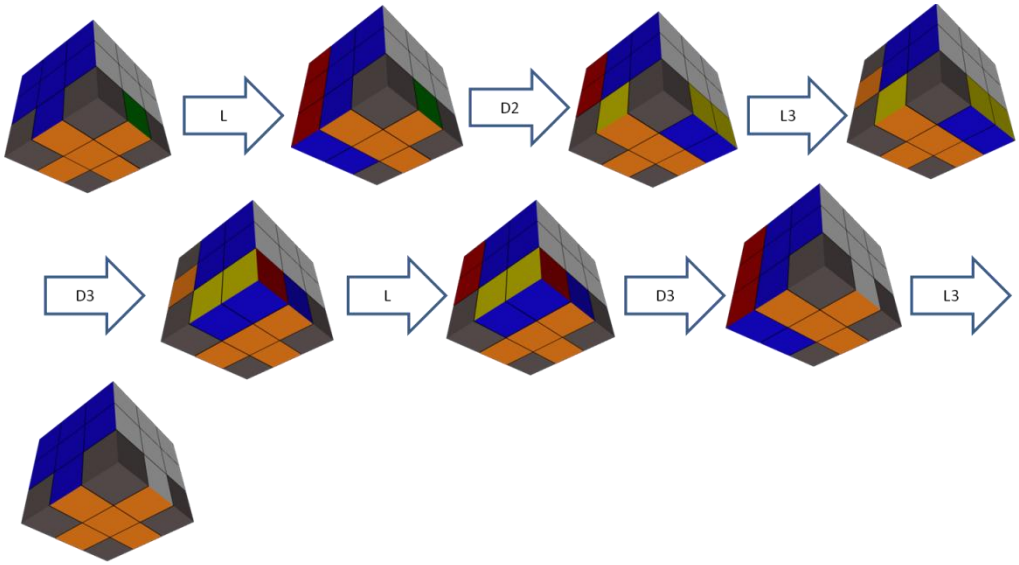



Ilustración 88: Quinto paso macrooperador 1

- 2: LDL3DLD2L3

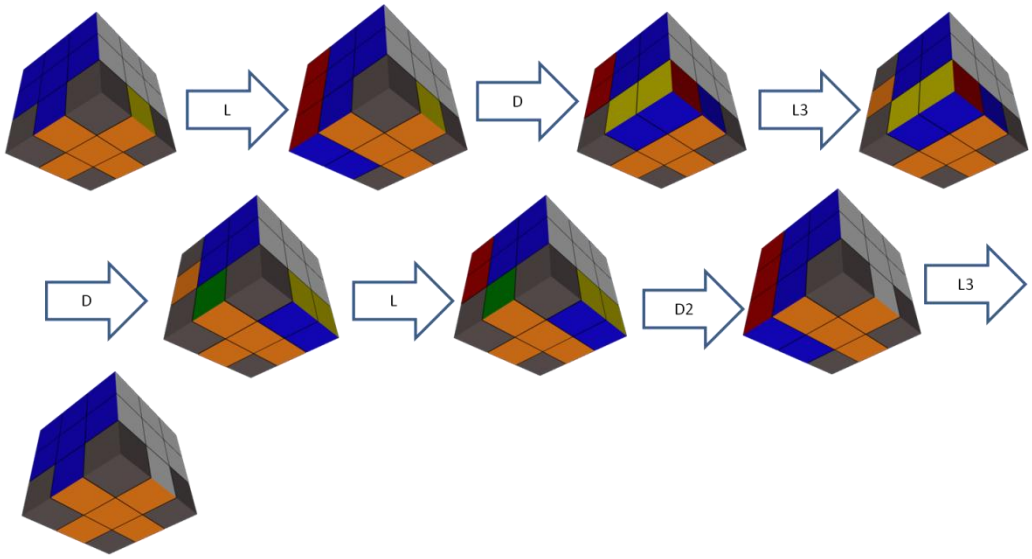



Ilustración 89: Quinto paso macrooperador 2

- 3: L3FLF3B3LFB3LF3L3B2D2

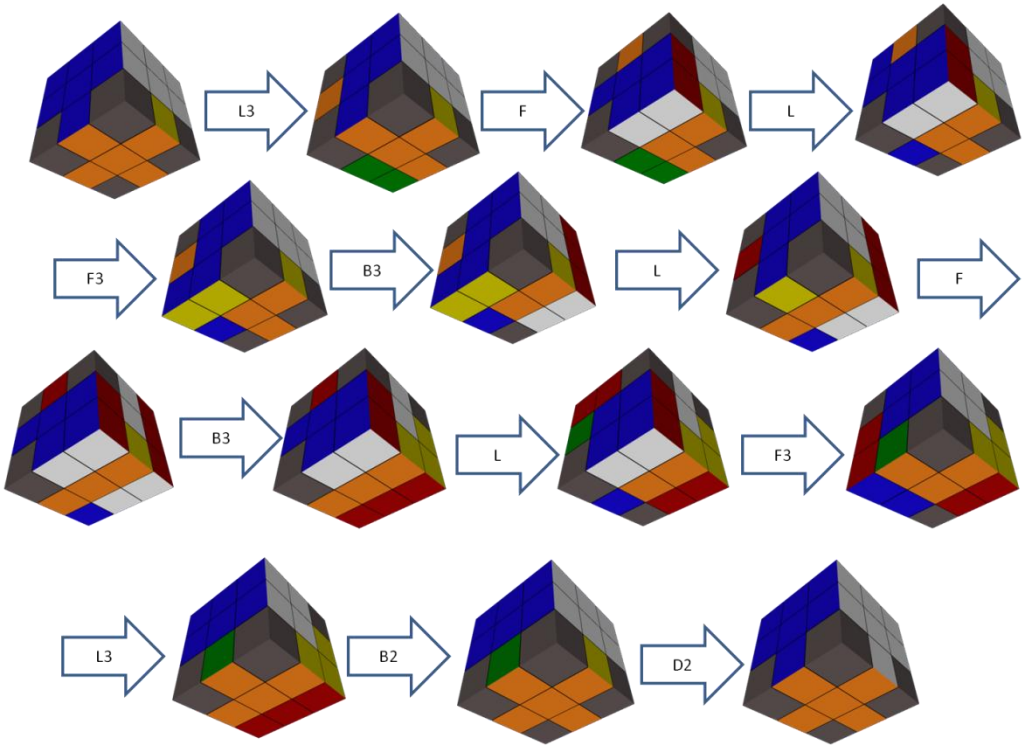



Ilustración 90: Quinto paso macrooperador 3

### 5.5.3. Observaciones

La preparación necesaria en este paso es mínima, porque sólo requiere una operación sencilla en caso de no estar en condiciones de aplicar un macrooperador. Con ello tenemos algo similar a lo ocurrido en el caso anterior, macrooperadores pesados, uno mucho más que el resto pero que tiene menos posibilidades de aplicarse.

- Tenemos dos macrooperadores de 7 pasos y uno de 13 pasos.
- Los macrooperadores menos pesados son los que tienen más posibilidades de aplicarse, teniendo una distribución. Una vez colocado el cubo, tenemos 6 posibles casos atendiendo a las piezas que nos interesa colocar, con esta distribución según el macrooperador a aplicar:
  - *Colocado:* En 1 caso entre 6 llegaremos a este paso con Las aristas bien colocados.
  - *Una arista:* En 4 entre 6 tendremos bien colocada una arista, dependiendo de la posición relativa de las otras tres aplicaremos el macrooperador 1 o 2.
  - *Dos aristas:* En 1 caso entre 6 tendremos 2 aristas bien colocadas y necesitaremos colocar las otras dos, que es cuando se aplica el macrooperador más costoso.
- Con estos datos, la media de movimientos necesarios para acabar esta parte es aproximadamente 7.
- Debido a las características del cubo de rubik, llegados a este paso hay muchas combinaciones que no son posibles. Esto se da en los últimos pasos, ya que teniendo colocadas las dos primeras capas, hay combinaciones en la última que no se pueden dar.

## 5.6. Sexto paso

El objetivo en este paso es colocar las esquinas de la cara inferior en su sitio, sin importar su orientación (que se arreglará en el siguiente paso).

De nuevo vemos que tenemos que dividir nuestros esfuerzos, ya que en el paso 2 las esquinas de la cara superior quedaban bien colocados y orientados, mientras que ahora necesitamos un paso para colocarlos y otro para orientarlos.

Como en los anteriores pasos basta con aplicar un solo macrooperador para que quede como queremos, es decir, que aunque los macrooperadores engloben muchos movimientos es posible que se hagan menos movimientos que en los primeros pasos donde se tienen que repetir macrooperadores.

Tendremos algo similar a la siguiente figura al final del proceso.

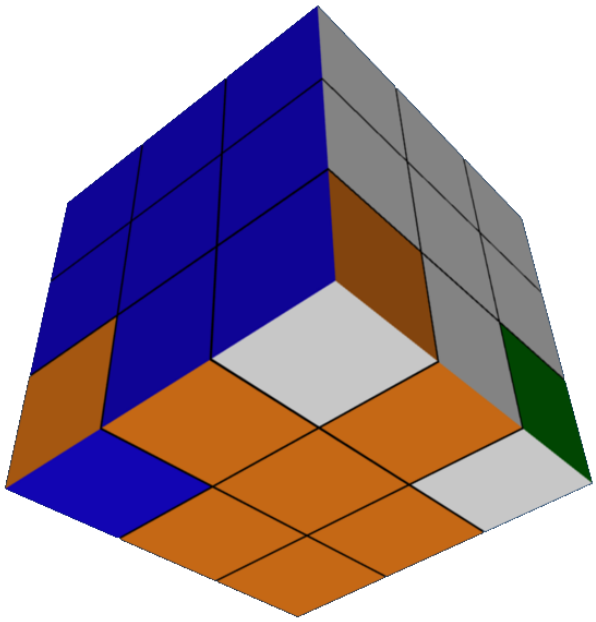


Ilustración 91: Esquinas colocados

### 5.6.1. Proceso

No es necesaria una preparación para aplicar un macrooperador, se comprueba en que caso estamos de los posibles y se aplica el macrooperador correspondiente.

Se observa que los casos posibles son menos de lo que sugeriría la lógica, pero en el cubo de rubik hay ciertas "permutaciones imposibles" y ello hace que se nos limite el número de posibilidades, reduciendo el número de macrooperadores y de comprobaciones.

### 5.6.2. Macrooperadores

- 1: L3DRD3LDR3D3

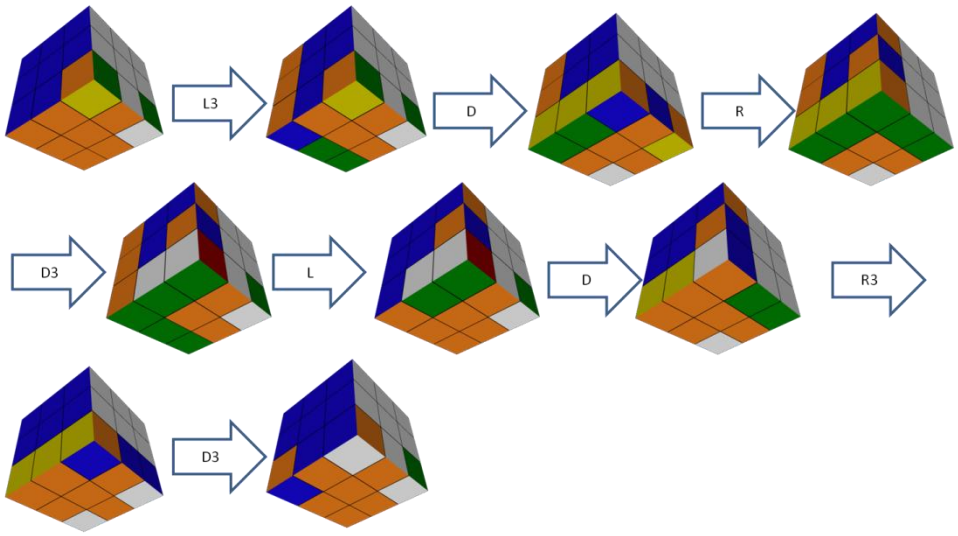



Ilustración 92: Sexto paso macrooperador 1

- 2: DRD3L3DR3D3L

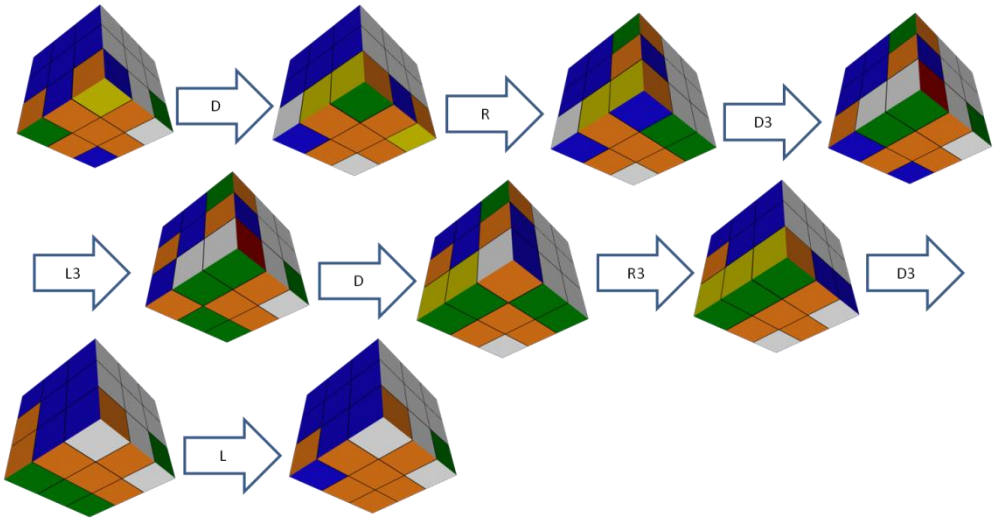



Ilustración 93: Sexto paso macrooperador 2

- 3: F3D3R3DRD3R3DRD3R3DRF

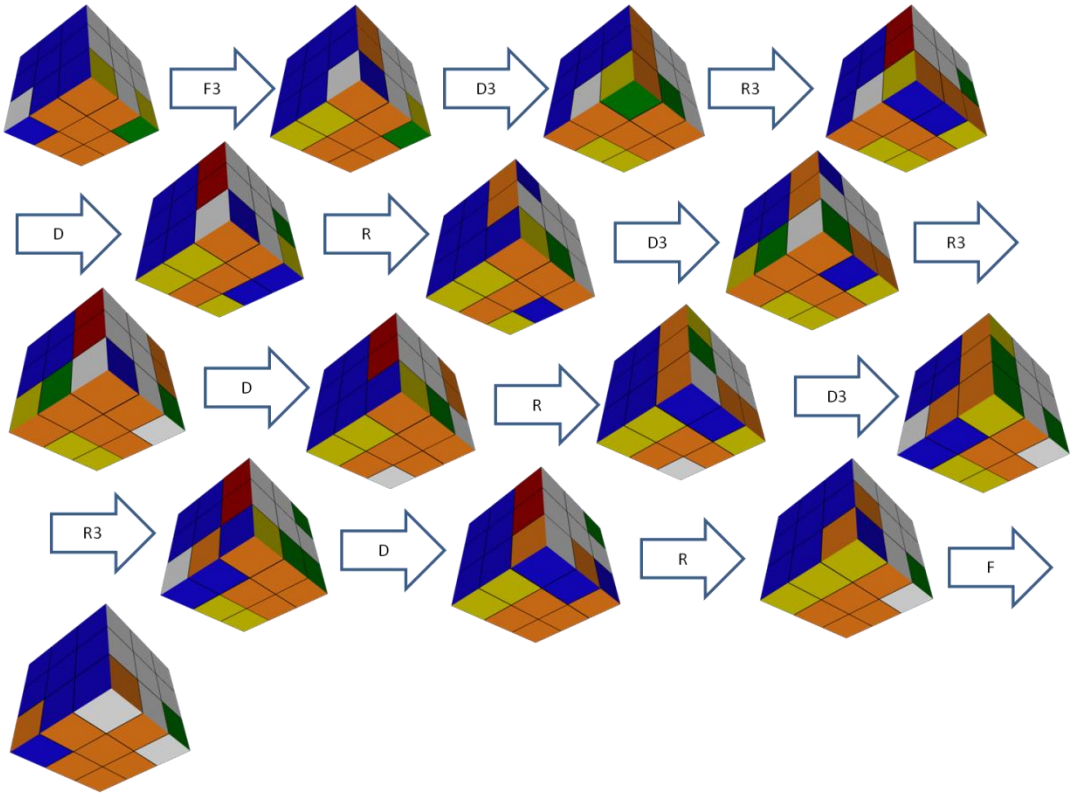



Ilustración 94: Sexto paso macrooperador 3

- 4: LRD2L3R3B3F3D2BFD2

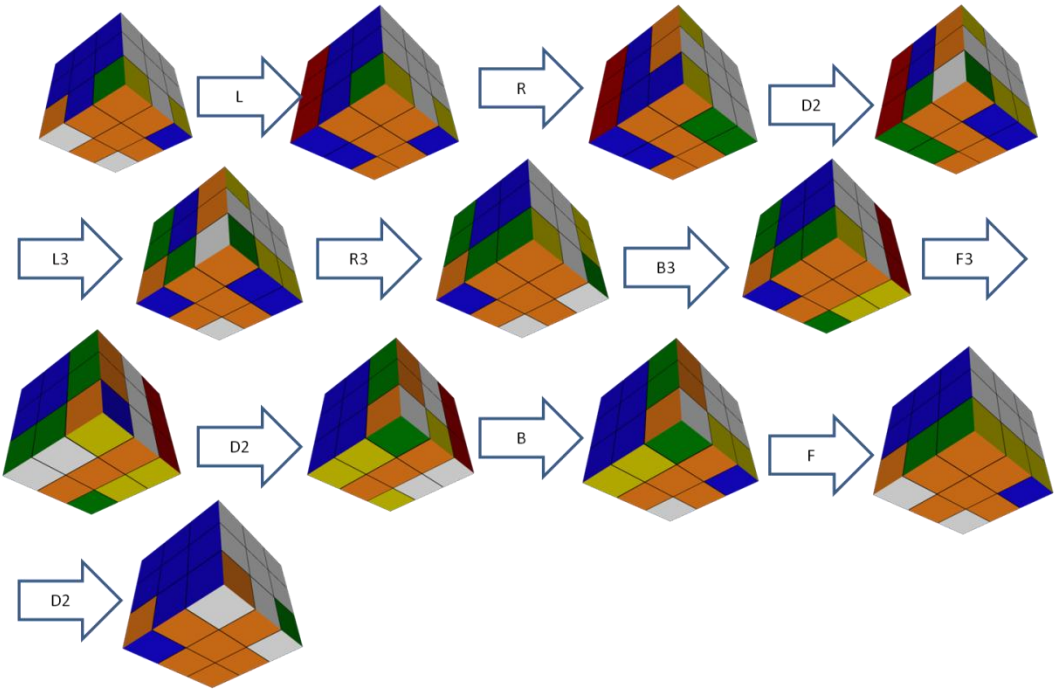



Ilustración 95: Sexto paso macrooperador 4

### 5.6.3. Observaciones

No requiere ninguna preparación, puesto que ya sólo tenemos que mover las cuatro últimas piezas y los macrooperadores abarcan todas las posibilidades. Vemos también que los estados que son imposibles facilitan el trabajo reduciendo los casos que se pueden presentar.

- Tenemos 4 macrooperadores, 2 de 8 operaciones, 1 de 14 operaciones y 1 de 11.
- La distribución es similar a los casos anteriores, hay dos estados que son mucho más frecuentes que el resto. Atendiendo a la posición y orientación de las piezas tenemos 12 posibles casos, cuya distribución (atendiendo al macrooperador que los resuelve) es la siguiente:
  - *Colocado:* El cubo está bien colocado y no es necesario hacer ninguna operación. 1 posibilidad entre 12.
  - *Un vértice bien colocado caso 1:* Un vértice está bien colocado y el resto precisan una rotación horaria. 4 casos de 12.
  - *Un vértice bien colocado caso 1:* Un vértice está bien colocado y el resto precisan una rotación anti horaria. 4 casos de 12.
  - *Ninguno bien colocado:* Ningún vértice está en su posición. 1 caso de 12
- Con esa distribución de los posibles estados, y junto con las características de los macrooperadores tenemos que en este paso la media aproximada de movimientos es 9.
- Nótese que ahora la media de movimientos empieza a subir después de la bajada notable en el paso 4 (de 31 movimientos del paso 3 a 6 del paso 4).

- El modo de operar en estos últimos pasos ha contribuido a la bajada de movimientos porque sólo se necesita aplicar un macrooperador.

## 5.7. Séptimo paso

Ya estamos en el último paso del algoritmo para principiantes. Tenemos las esquinas de la cara inferior bien colocadas pero mal orientadas, y el objetivo de este paso es orientarlos correctamente.

Llegados a este paso hay dos formas de hacerlo, una más intuitiva, pero con mayor coste, y otra de menor coste pero con más macrooperadores y muchas comprobaciones.

Hemos optado por la segunda opción para reducir el coste de este paso y, dentro del método para principiantes, obtener el menor coste posible.

Tenemos 26 macrooperadores (7 macrooperadores distintos, el resto son simetrías) en total sólo para este paso, con sus correspondientes 26 comprobaciones.

Como en pasos anteriores nos basta con aplicar un macrooperador para tener el cubo resuelto.

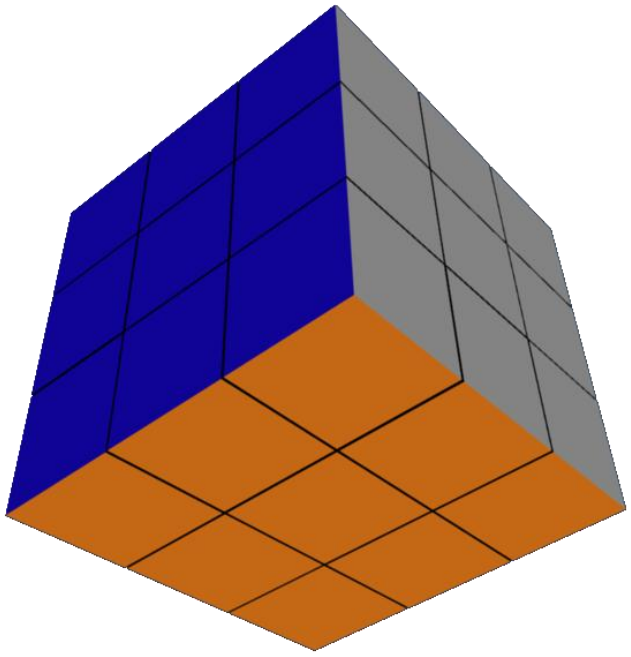


Ilustración 96: Cubo resuelto

### 5.7.1. Proceso

No se requiere ninguna preparación, simplemente comprobar en qué caso nos encontramos y aplicar el macrooperador correspondiente, y ya tendremos el cubo resuelto.

En este paso tendemos los macrooperadores más largos, pero como sólo necesitaremos aplicar uno de ellos, es posible que hagamos menos movimientos que en pasos donde era necesario repetir macrooperadores.

El número de macrooperadores aumenta mucho con respecto a los pasos anteriores.

### 5.7.2. Macrooperadores

- 1: R2DF2D3RB2R3DF2D3RB2R

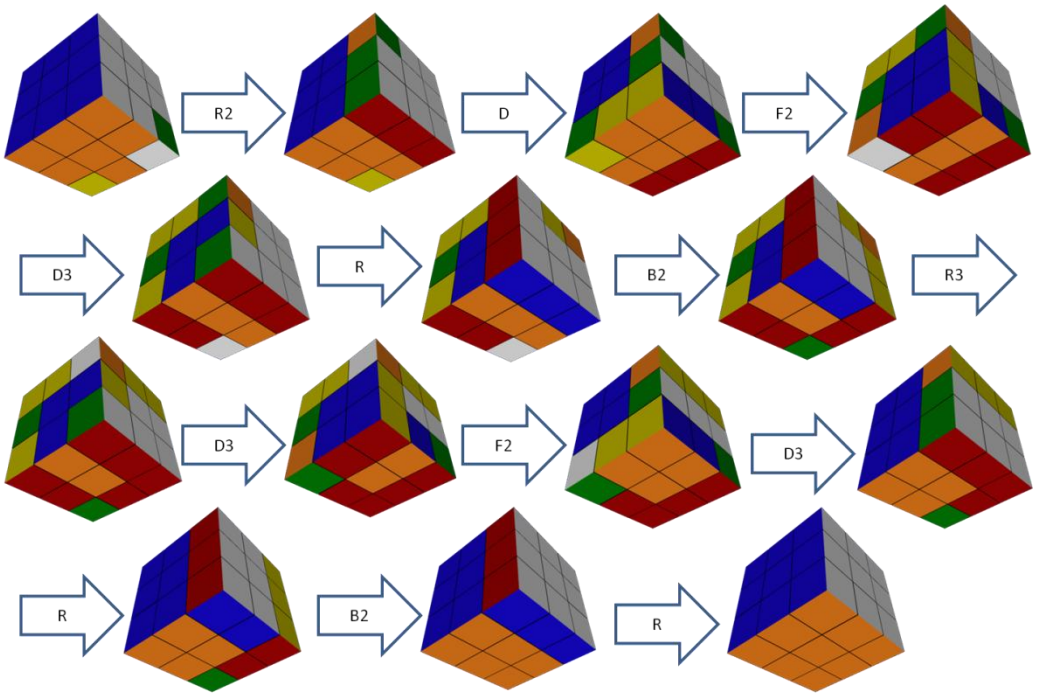



Ilustración 97: Séptimo paso macrooperador 1

- 2: B2U3B2DB3L2DL2D3B2UB3D3

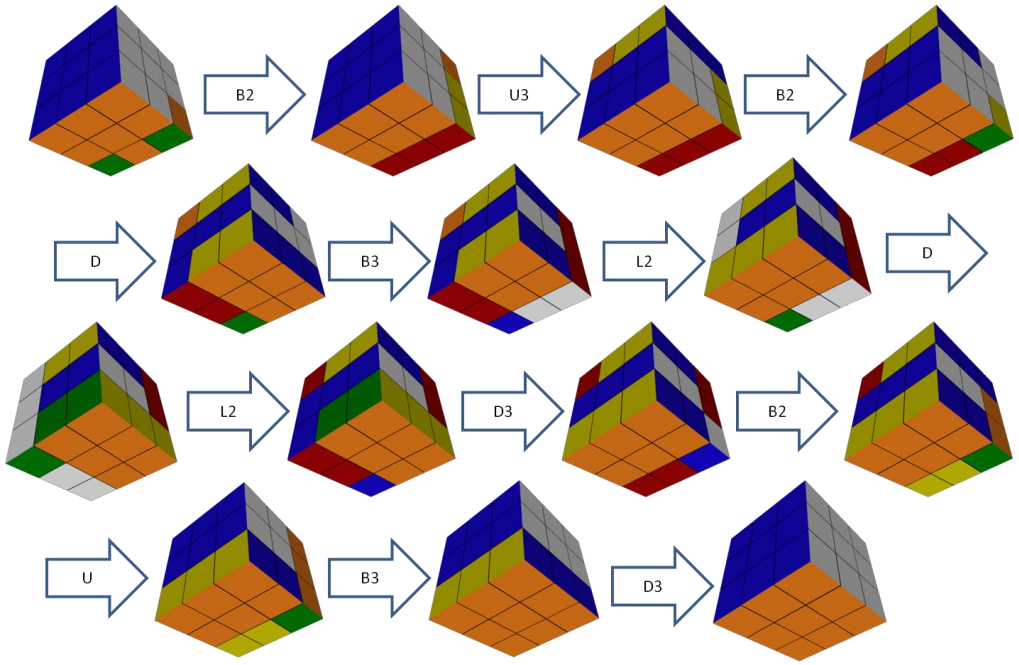



Ilustración 98: Séptimo paso macrooperador 2

- 3: BL3U2LB3D2BL3U2LB3D2

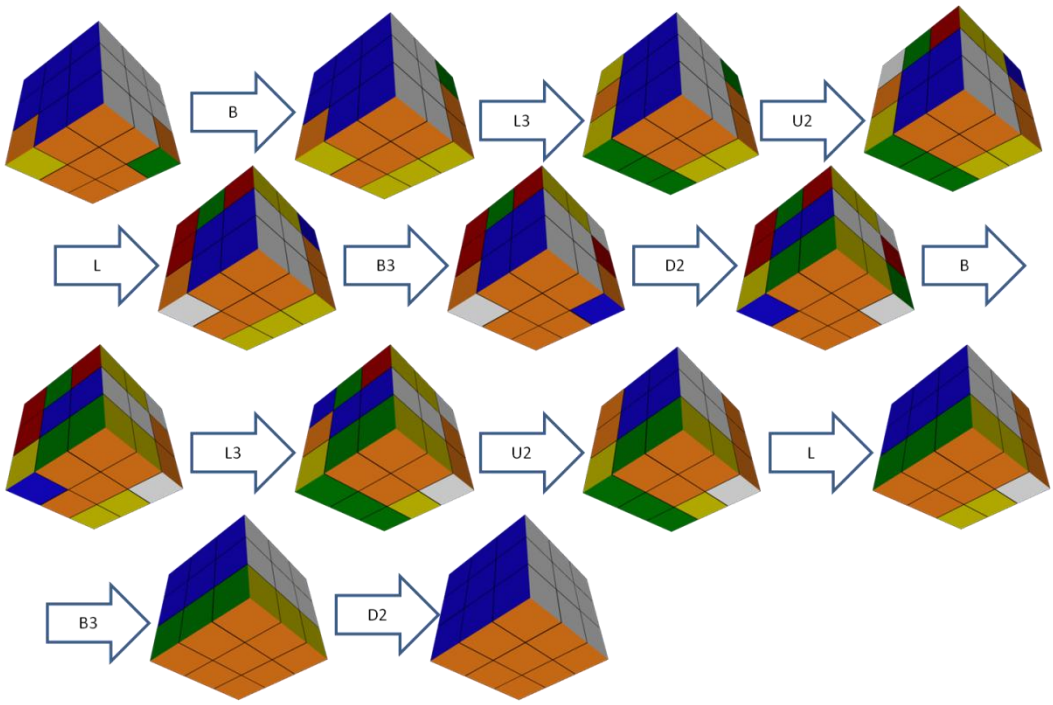



Ilustración 99: Séptimo paso macrooperador 3

- 4: BD3FD2B2U3L2UDF3D3BD3

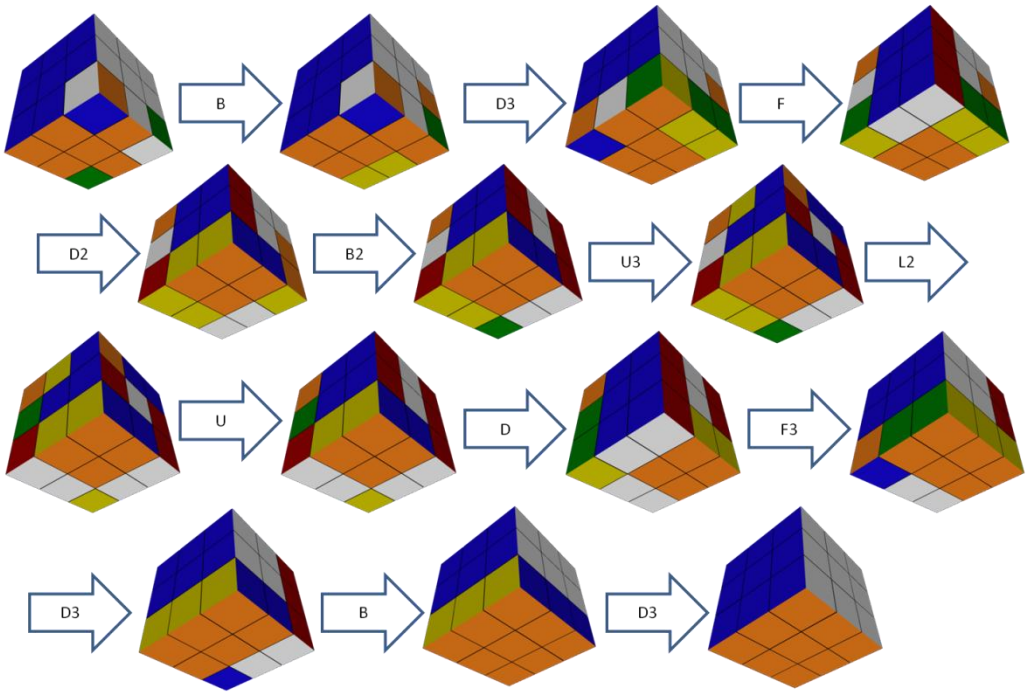



Ilustración 100: Séptimo paso macrooperador 4

- 5: R3DLU3D3B2UR2D2L3DR3D

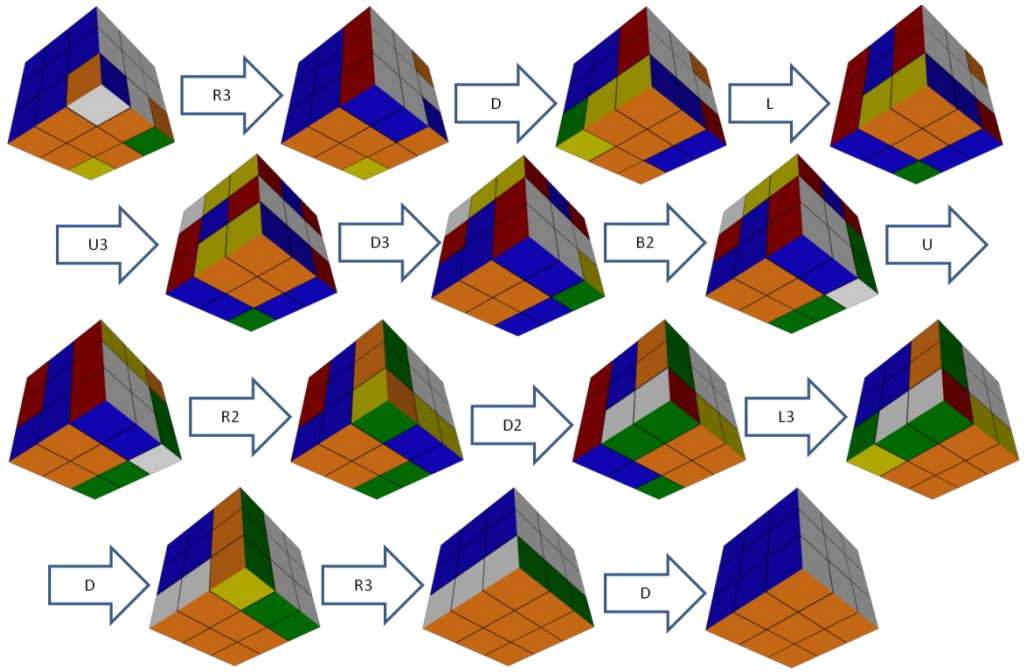



Ilustración 101: Séptimo paso macrooperador 5

- 6: R3D2RD2R2F3U3R3BR2B3UFD3R3D3

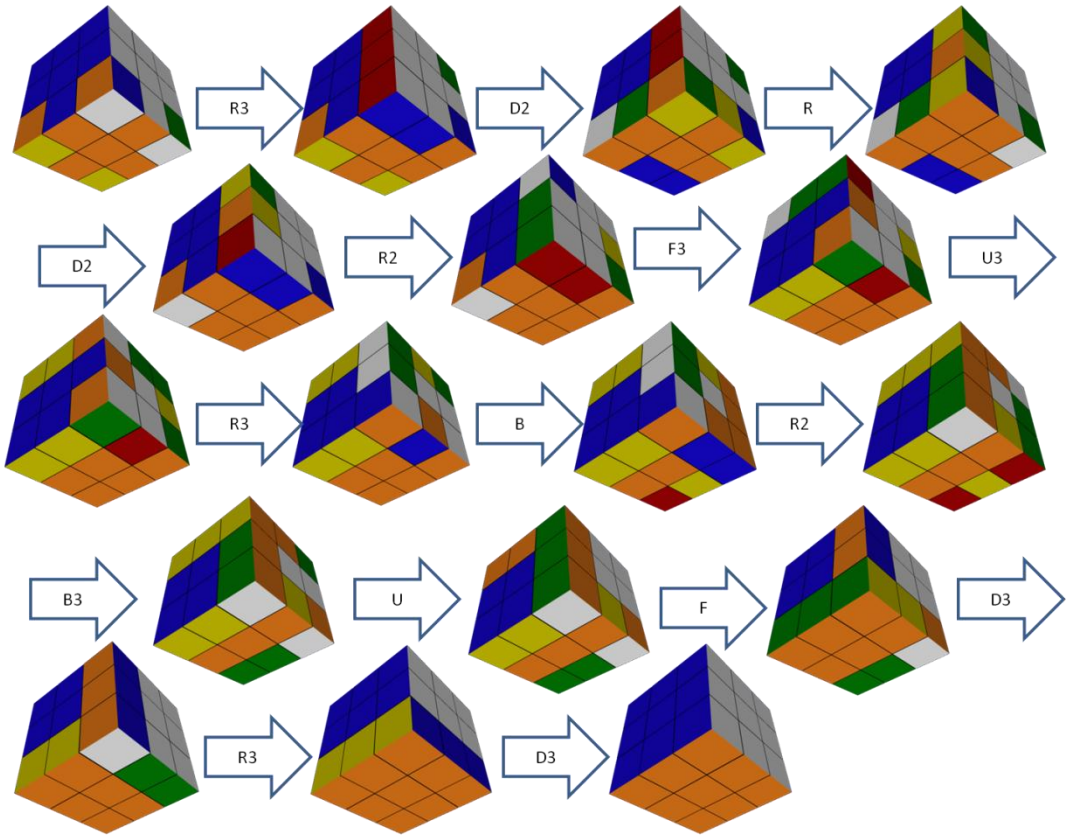



Ilustración 102: Séptimo paso macrooperador 6

- 7: L3RUL2D2LU3L2D2R3DR2F2R2D3

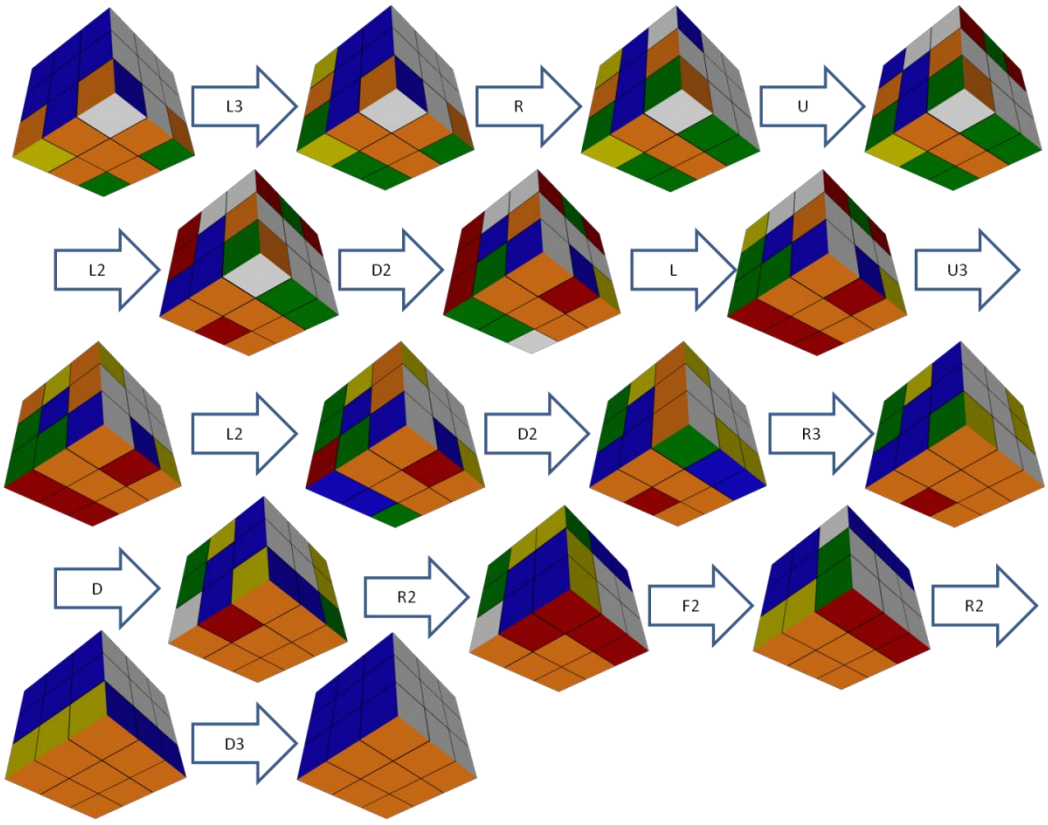



Ilustración 103: Séptimo paso macrooperador 7

### 5.7.3. Observaciones

El último paso es el que engloba más macrooperadores, siempre que busquemos la solución menos costosa dentro del algoritmo de principiantes. Como en los anteriores pasos se aplica un solo macrooperador para resolver el cubo.

- El coste de los macrooperadores es superior a los anteriores casos.
- Hay más casos que en los anteriores pasos. Podría pensarse que al estar cerca del final, y al tener sólo cuatro piezas que permutar tendríamos menos casos. Debido a los estados imposibles que facilitaban los anteriores casos aquí tenemos más posibles estados y más comprobaciones. Hay un total de 25 casos que se resuelven con los 7 macrooperadores. Su distribución es la siguiente:
  - *Colocado:* El cubo está resuelto al llegar a este paso. 1 caso de 25
  - *Girar dos esquinas de la misma arista, caso 1:* Tenemos dos vértices de la misma arista bien colocados y los otros no. 4 casos de 25
  - *Girar dos esquinas de la misma arista, caso 2:* Tenemos dos vértices de la misma arista bien colocados y los otros no. La orientación de Las piezas es diferente al caso anterior. 4 casos de 25
  - *Girar dos vértices de aristas distintas:* Tenemos dos vértices de aristas distintas bien colocados y los otros no. 4 casos de 25
  - *Girar tres vértices, caso 1:* Tenemos un vértice sólo bien orientado. 4 casos de 25.

  - *Girar tres vértices, caso 2:* Tenemos un vértice sólo bien orientado. La orientación de Las piezas es diferente al anterior caso. 4 casos de 25.
  - *Girar cuatro vértices, caso 1:* Ningún vértice está bien colocado. 2 casos de 25.
  - *Girar cuatro vértices, caso 2:* Ningún vértice está bien colocado. 2 casos de 25.
- Vemos que al tener más posibles estados que en anteriores pasos la distribución de los mismos es más uniforme, aunque algunos estados son más comunes que otros.
- Los macrooperadores son más costosos que en pasos anteriores.
- La media de movimientos necesaria para terminar este paso y resolver el cubo es aproximadamente 13.

# 6. Desarrollo del cubo de 2x2x2

Se pretende encontrar la solución óptima del cubo de rubik simplificado, el de 2x2x2. Dicho cubo ofrece un espacio de estados reducido, por lo que se podrá hacer una búsqueda en amplitud sin riesgo de que tarde demasiado.

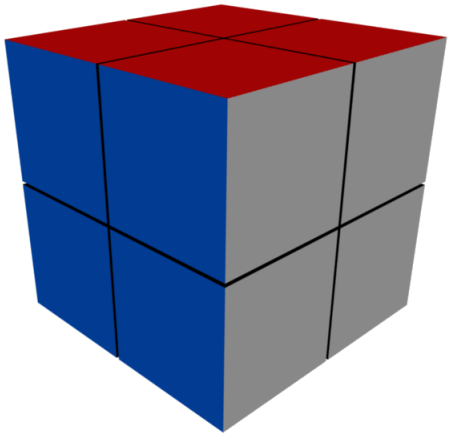


Ilustración 104: Cubo 2x2x2

## 6.1. Curiosidades

Aunque el cubo de 2x2x2 ofrece un espacio de estados muy pequeño (en comparación con el de 3x3x3) hallar la solución sin tener conocimientos del cubo es una tarea complicada.

Para su resolución no óptima se pueden usar algoritmos del cubo de 3x3x3, concretamente los aplicados a las esquinas. De todas formas hay que tener claro siempre qué colores son opuestos, puesto que no tendremos los cubitos centrales que nos sirvan de referencia.

También conviene mencionar que en el cubo de 2x2x2 pueden presentarse estados que no se darían en el de 3x3x3. Es decir, podemos tener un estado en el de 2x2x2, pero ese mismo estado puede ser imposible de conseguir en las esquinas de un cubo de 3x3x3. Esto se expande a todos los cubos regulares, los de índice par presentan esos estados, pero los de índice impar no. A la hora de resolver un cubo de NxNxN, hay que tener en cuenta este hecho.

La solución de cualquier cubo de 2x2x2 tiene 11 operaciones o menos.

## 6.2. Operaciones

En este caso usaremos sólo operaciones sencillas, sin necesidad de recurrir a macrooperadores, dichas operaciones se representan de igual manera que para el cubo de 3x3x3.

## 6.3. Estructuras de datos

Las estructuras de datos son bastante similares a las usadas en el cubo de 3x3x3. El árbol generado es igual, pero con menos información, debido a que hacemos una búsqueda en amplitud y no se utilizan heurísticas.

La lista abierta presenta algún cambio, simplificándose puesto no la ordenamos, y se van insertando los nodos al final de la lista según vamos expandiendo el árbol. Tendríamos la siguiente estructura.

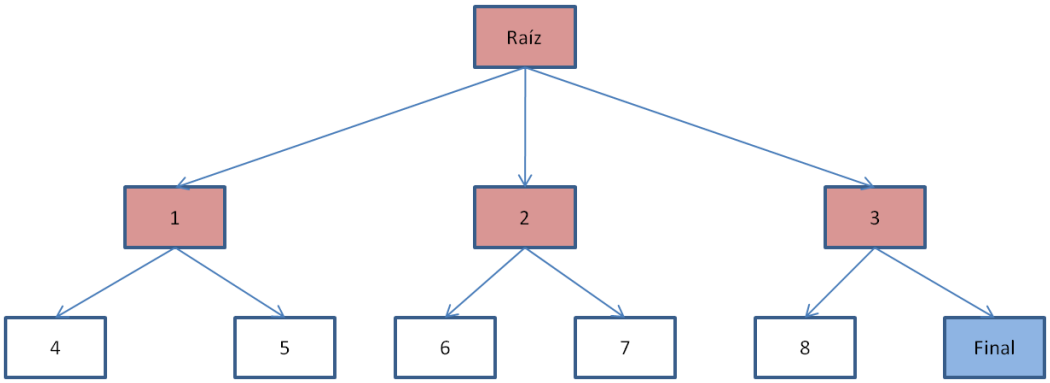



Ilustración 105: Árbol 2x2

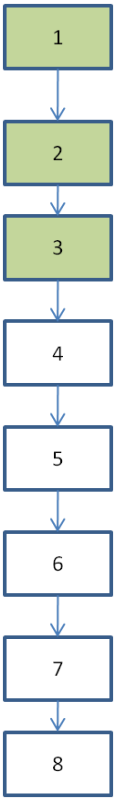



Ilustración 106: Abierta 2x2

## 6.4. Comparación de dos estados

En el cubo de 3x3x3 fijábamos una posición, como si dijéramos que un determinado color puede estar sólo en una posición (arriba, derecha…), debido a la configuración de dicho cubo esto hace que sólo pueda existir una solución posible, lo que simplifica el proceso de búsqueda.

En el cubo de 2x2x2, al no tener centros, hace que dos cubos puedan ser iguales, aun teniendo distinta orientación, este hecho hace que tengamos que realizar una operación extra de comparación más compleja que la que se usaba en el cubo de 3x3x3.

Por ejemplo, tenemos que estos dos cubos son iguales, el primero el cubo solución, y el segundo un cubo que también es estado final, sólo que esta girado con LR3, que cambia la posición del cubo.

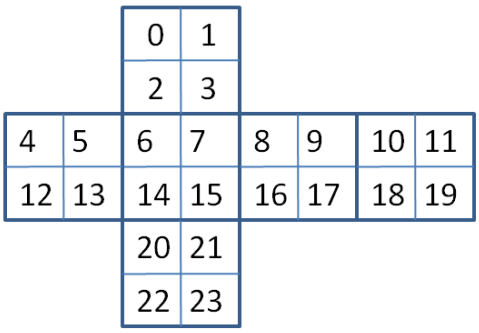



Ilustración 107: Estado final 1

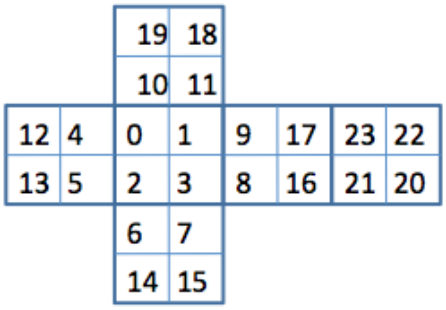



Ilustración 108: Estado final 2

Por ello se ha realizado un procedimiento que compare dos cubos y nos diga sin son iguales. Se utiliza en dos ocasiones: para podar el árbol de estados y evitar expandir nodos que ya han sido explorados, quitando los estados repetidos, y para saber si se ha llegado al final, comparando un estado con el estado final.

El procedimiento funciona de la siguiente manera:

- Se reciben dos cubos, y hay que comparar si son iguales, en este caso comparamos dos cubos que ya están en estado final:

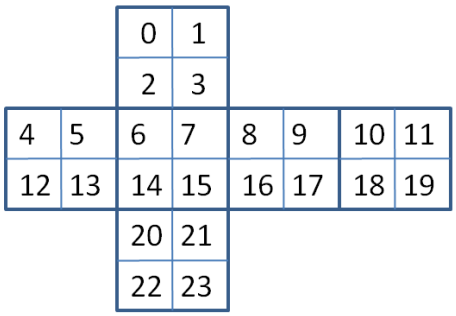



Ilustración 109: Comparar 1

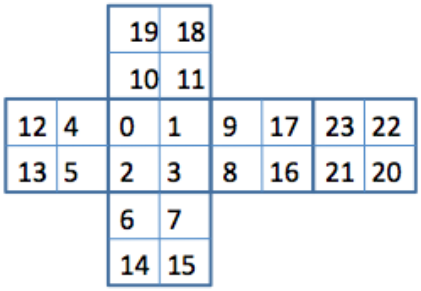



Ilustración 110: Comparar 2

- Primero buscamos a ver si la cara U del primer cubo está en el segundo, es decir, en caso de no estar ya podemos asegurar que los cubos no son iguales y no se seguiría comparando.

- Rotamos el segundo cubo para que la cara U coincida con la del primer cubo, esto lo hacemos para fijar una posición, y poder comparar. Se cambia la posición del cubo pero no se altera.
- Comparamos el primer cubo con el segundo (rotado) en caso de ser iguales devolvemos 1.

Así sabemos si dos estados son iguales, proceso fundamental para comparar dos cubos y saber si hemos llegado al final.

## 6.5. Funcionamiento

Se comprueba si el nodo inicial es estado final. De ser estado final no entraríamos a generar el árbol. De no ser estado final, se introduce el nodo raíz en la lista cerrada y se expanden y comprueban sus hijos, al no ser ninguno de ellos estado final se insertan en orden de izquierda a derecha en la lista abierta.

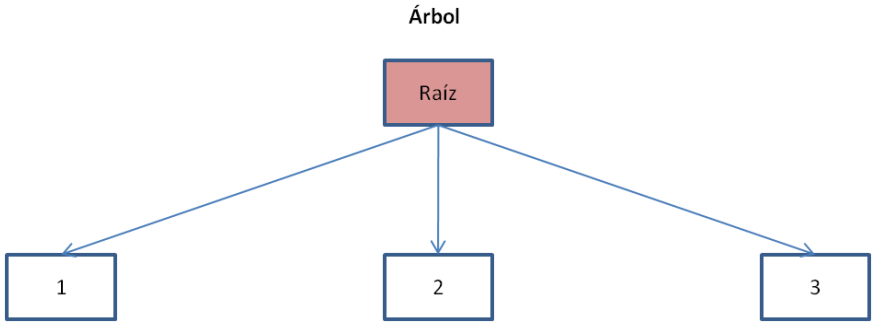



Ilustración 111: Árbol ejemplo 1

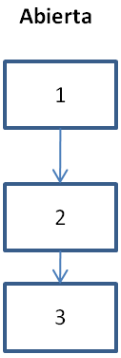



Ilustración 112: Abierta ejemplo 1


Ilustración 113: Cerrada ejemplo 1

Comprobamos si el nodo 1 está en lista cerrada, es decir, si ese estado contiene un cubo que ya hemos explorado, no lo tiene. Sacamos de la lista abierta el nodo 1 y lo pasamos a la lista cerrada, expandimos sus hijos, comprobamos si alguno es estado final y, al no serlo, insertamos los hijos del nodo 1 al final de la lista abierta.

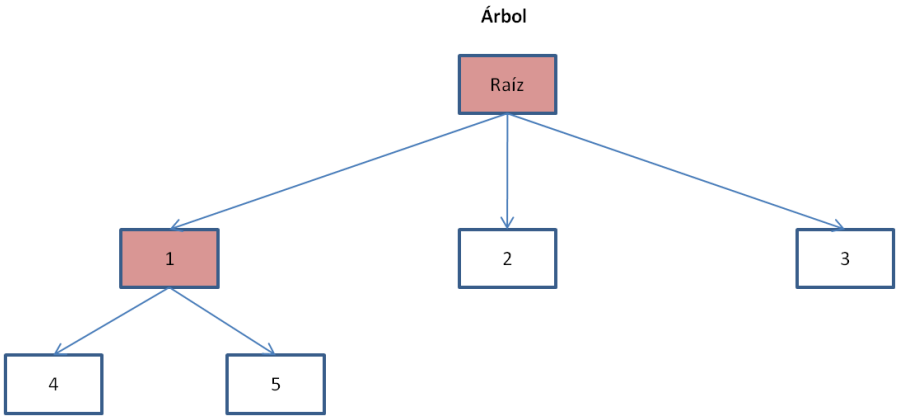



Ilustración 114: Árbol ejemplo 2

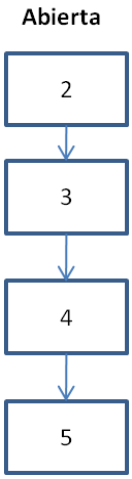



Ilustración 115: Abierta ejemplo 2

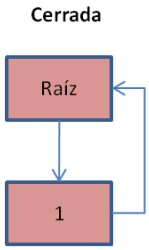



Ilustración 116: Cerrada ejemplo 2

Comprobamos si el nodo 2 está en lista cerrada, es decir, si ese estado contiene un cubo que ya hemos explorado, no lo tiene. Pasamos el nodo 2 de lista abierta a lista cerrada, expandimos sus hijos y comprobamos si alguno es solución, al no serlo los insertamos al final de la lista abierta.

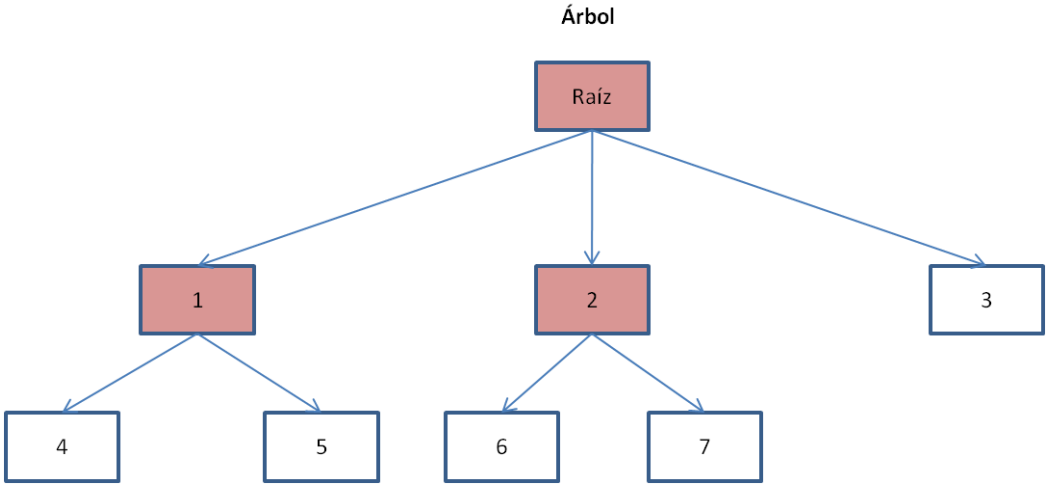



Ilustración 117: Árbol ejemplo 3

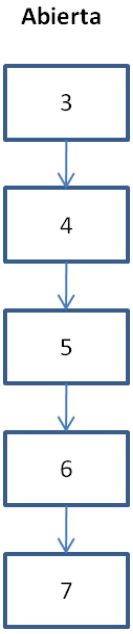



Ilustración 118: Abierta ejemplo 3

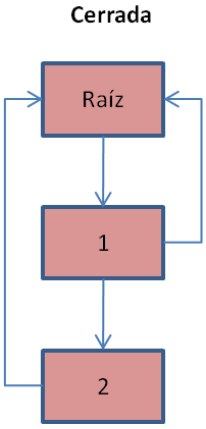



Ilustración 119: Cerrada ejemplo 3

Comprobamos si el nodo 1 está en lista cerrada, es decir, si ese estado contiene un cubo que ya hemos explorado, no lo tiene. Pasamos el nodo 3 de lista abierta a lista cerrada, expandimos sus hijos y comprobamos si alguno es solución. Sí que hay uno que es solución, lo pasamos directamente a la lista cerrada y damos por finalizado el proceso.

El proceso seguido para comprobar si un estado es solución es, primero comprobar si es solución, si no lo es, insertarlo en lista abierta y repetir el proceso en un nodo hermano. De ser solución se para el proceso. Por eso el nodo 8 se inserta en la lista abierta antes de dar con la solución.

Recorremos la lista cerrada al revés para saber el camino a la solución óptima.

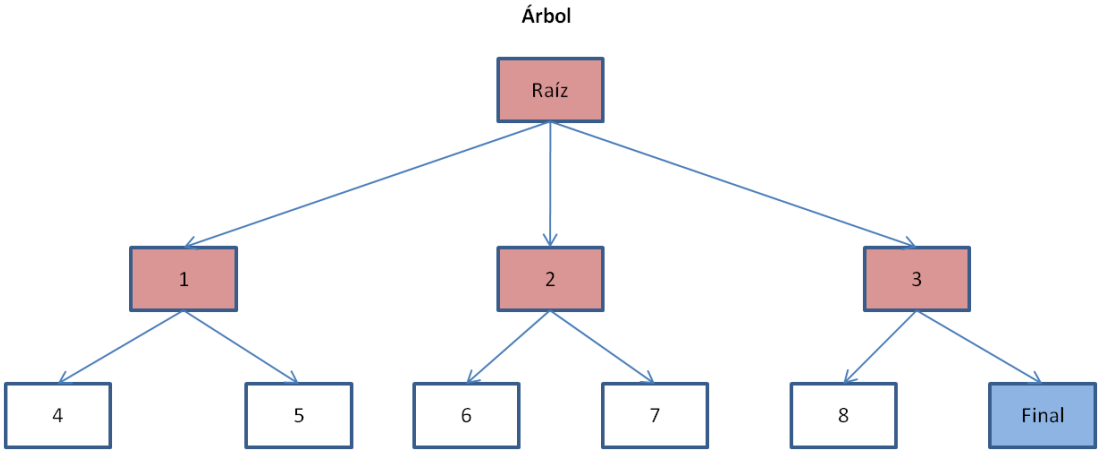



Ilustración 120: Árbol ejemplo 4

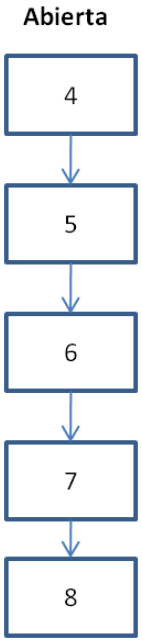



Ilustración 121: Abierta ejemplo 4

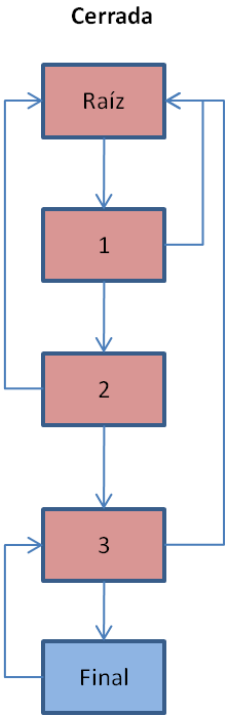



Ilustración 122: Cerrada ejemplo 4

## 6.6. Observaciones

El cubo de 2x2x2 tiene muchos menos estados que el de 3x3x3, por lo que la tarea de encontrar la solución óptima se simplifica mucho. La búsqueda es no informada en amplitud, esto nos garantiza encontrar la solución óptima para cada caso.

Se observa un notable aumento del tiempo de resolución con respecto al cubo de 3x3x3, a pesar de ser un problema más sencillo. Esto se debe al tipo de búsqueda realizado, mientras que en el de 3x3x3 se buscaba una resolución rápida, aquí se ha buscado una solución óptima.

# 7. Pruebas y estadísticas

Ejemplos de ejecución de ambos sistemas, estadística del cubo de rubik de 3x3x3 y comparativa entre ambos.

## 7.1. Pruebas

Se generan casos de prueba para los dos sistemas creados, girando aleatoriamente un cubo resuelto, y se estudian los resultados obtenidos.

### 7.1.1. Cubo de 3x3x3

Ejemplo de ejecución del cubo de 3x3x3. El propio programa genera un cubo de 3x3x3 de manera aleatoria, girando 25 veces un cubo armado. Se genera un archivo de salida llamado “salida.txt” con los movimientos y los pasos dados hasta solventar el cubo.

```
----------paso 1---------------
movimientos: 5
----------paso 2---------------
movimientos: 28
----------paso 3---------------
movimientos: 54
----------paso 4---------------
movimientos: 60
----------paso 5---------------
movimientos: 68
----------paso 6---------------
movimientos: 79
----------paso 7---------------
movimientos: 95
$$$$$$
         11 46 45
         22    44
         36 25 33
10 21 47 37 26 42 34 35 28 39 38 5
9     6  12    15 4     20 27    3
40 43 16 2  41 13 7  23 8  19 18 31
         17 32 14
         29    24
         30 1  0
permutacion:
giro 0
movimientos: 0
-----------------
         11 46 45
         22    44
         34 4  7
10 21 33 42 15 13 14 35 28 39 38 5
9     25 26    41 32    20 27    3
40 43 36 37 12 2  17 23 8  19 18 31
         47 6  16
         29    24
         30 1  0
permutacion: F
giro 3
movimientos: 1
-----------------
```

```
          45 44 7
          46    4
          11 22 34
39 38 5   10 21 33 42 15 13 14 35 28
9      25 26    41 32    20 27    3
40 43 36 37 12 2  17 23 8  19 18 31
          47 6  16
          29    24
          30 1  0
permutacion: U
giro 3
movimientos: 2
-----------------
          45 44 7
          46    4
          16 6  47
39 38 17 2  12 37 36 15 13 14 35 28
9      32 41    26 25    20 27    3
40 43 42 33 21 10 5  23 8  19 18 31
          34 22 11
          29    24
          30 1  0
permutacion: F
giro 2
movimientos: 3
-----------------

[.............]

-----------------

          0  1  2
          3     4
          5  6  7
8  9  10 11 12 13 14 15 16 17 18 19
20    21 22    23 24    25 26    27
28 29 30 31 32 33 34 35 36 37 38 39
          40 41 42
          43    44
          45 46 47
permutacion: E
giro 2
B3D2BD2B2R3U3B3LB2L3URD3B3D3
movimientos: 95
-----------------
#######
```

Al principio de la traza aparece una lista con los pasos del algoritmo para principiantes, indicando el número de operaciones sencillas totales al finalizar dicho paso, por lo tanto en el paso 7 podemos ver el número de operaciones totales necesarias.

Luego aparece un cubo de rubik representado de forma desplegada, ese cubo es el inicial, resultado de haber aplicado 25 operaciones simples al cubo sin desordenar. Por ello la información que aparece debajo está en blanco.

Los cubos siguientes son el resultado de aplicar el operador o macrooperador indicado por "permutación" y "giro" sobre el estado inmediatamente anterior. En caso de ser un macrooperador aparecerá su traducción en operaciones simples justo debajo de "giro".

"Movimientos" indica el total de movimientos después de aplicar el operador o macrooperador.

Las cadenas "#####" y "$$$$$" son separadores usados por el programa encargado de obtener los datos estadísticos.

*Observaciones:*

- Se han realizado 1000 ejecuciones, cada ejecución escribía al final del fichero de salida.
- Se tienen 20 archivos de salida (con 1000 ejecuciones cada uno) para las estadísticas. Un total de 20000 ejecuciones.
- Cada tanda de 1000 ejecuciones tarda aproximadamente 5'25 segundos en completarse, lo que quiere decir que, de media, un cubo lo hace en 0,00525 segundos.

### 7.1.2. Cubo de 2x2x2

Salida de la ejecución de un cubo de 2x2x2. El programa genera un cubo girando aleatoriamente 12 veces un cubo en su estado final. La salida se muestra por pantalla.

```
************************
orientador de cubo 2x2x2
  U
L F R B
  D
************************
*****INSTRUCCIONES******
La disposición de las caras es la mostrada
en el diagrama anterior.
Una letra mayúscula indica que se gira en el
sentido de las agujas del reloj esa cara (90°)
Si a la letra le sigue un 2 el giro es de 180°
Si le sigue un 3 el giro es de 270°

Final con F, 2
      7  15
      6  14
3  2  5  13 20 21 16 8
1  0  4  12 22 23 17 9
      11 19
      10 18
g: 5
F - 2
----------------
```

```
       7  15
       19 11
3  22 12 4  0  21 16 8
1  20 13 5  2  23 17 9
       14 6
       10 18
g: 4
L - 3
----------------

[..............]
----------------
       16 8
       2  9
21 5  6  10 1  3  7  15
20 11 0  19 12 23 17 14
       4  22
       13 18
g: 0
0 - 0
----------------
```

Se muestra la traza de la solución a la inversa, primero se muestra el estado final y por último el estado inicial. La g indica la profundidad en el árbol, que a su vez nos dice el número de movimientos totales necesarios para resolverlo.

Debajo se puede observar el movimiento básico necesario para llegar a un estado desde el anterior. Como no hay macrooperadores ya que no hace falta mostrar nada más. El estado inicial, para evitar confusiones, va marcado con “0 - 0”.

### 7.1.3. Comparaciones

Una comparativa del rendimiento de las dos aplicaciones creadas. Se aprecia un mayor rendimiento en la aplicación del cubo de 3x3x3.

*Cubo de 3x3x3:*

- Aplica el algoritmo de principiantes y resuelve el cubo en aproximadamente 90 movimientos.
- Tarda el resolver un cubo aproximadamente 0,00525 segundos.
- No se aprecia una variación significativa de tiempo en función de los movimientos totales necesarios para resolverlo. Tampoco en función de los pasos aplicados.
- En la aplicación del primer paso, que se usa búsqueda informada, no se aprecia una variación de tiempo en función de la profundidad. Pero si se ha apreciado una mejora significativa al aplicar distintas medidas para reducir el número de estados, eliminando estados repetidos, mejorando la heurística y el tratamiento de las listas.

*Cubo de 2x2x2:*

- Resuelve el cubo en el número óptimo de movimientos: 11 o menos.

- El tiempo varía mucho en función del número de movimientos necesarios para resolverlo, crece mucho en función de la profundidad del árbol.
    - Con 4 operaciones o menos tarda menos de un segundo.
    - Entre 4 y 6 varios segundos
    - Entre 6 y 8 varios minutos. Se observa variación entre dos casos con los mismos movimientos, ya que el tiempo en explorar el árbol será mayor si la solución está más alejada dentro de un mismo nivel. Se puede llegar a 5 minutos.
    - De 8 a 9 varios minutos, pudiendo llegar a 15 en muchos casos.
    - 10 operaciones: llega en muchos casos a 1 hora.
    - 11 operaciones: puede tardar varias horas, en las pruebas hechas se han observado hasta 4 horas.
- Al no poder tener una posición fijada como en el cubo de 3x3x3 las operaciones de comprobar el estado final, y comparar dos estados para ver si son iguales se complican.
- Se aprecia una mejora notable de rendimiento al aplicar operaciones de optimización como quitar estados repetidos y verificar el estado final anticipadamente.

### 7.1.4. Estadísticas

Para realizar las estadísticas del cubo de 3x3x3 se ha hecho una aplicación que lee los datos de salida de dicho cubo, los analiza y muestra una serie de resultados.

Se ha ejecutado el programa principal en tandas de 1.000 con la ayuda de un *script*, y la salida de las ejecuciones se almacena en un fichero. La salida de una de las pruebas es la siguiente:

```
el número medio total es: 502
el número medio de 502 pruebas con 7 pasos es: 91
el número medio de pasos es: 5
distribución de número de pasos-----
    1 pasos: 167 veces con una media de 2 movimientos
    2 pasos: 66 veces con una media de 4 movimientos
    3 pasos: 15 veces con una media de 12 movimientos
    4 pasos: 12 veces con una media de 32 movimientos
    5 pasos: 49 veces con una media de 69 movimientos
    6 pasos: 189 veces con una media de 79 movimientos
    7 pasos: 502 veces con una media de 91 movimientos
Media de movimientos por paso-----
    1º paso con 995 veces: 4
    2º paso con 756 veces: 16
    3º paso con 715 veces: 31
    4º paso con 633 veces: 6
    5º paso con 793 veces: 7
    6º paso con 678 veces: 9
    7º paso con 715 veces: 13
```

El primer número hace referencia a las pruebas en las que se han tenido que realizar los 7 pasos. En este ejemplo vemos que necesitamos 7 pasos para 502 de los 1.000 cubos

resueltos. En la siguiente línea tomamos el número medio de movimientos necesarios para resolver esos 502 cubos de 7 pasos. Después indicamos que el número medio de pasos es 5. Eso no quiere decir que sean los pasos del 1 al 5.

Después se muestra una distribución de los cubos en función del número de pasos necesarios para resolverlo. Se ve que la mayoría de los cubos, con mucha diferencia, requieren los 7 pasos.

Al final se muestra una lista con el número de veces que se ha aplicado un paso y la media de movimientos de cada paso. Para contabilizar la media de movimientos por cada paso se contabilizan sólo las veces que se ha usado dicho paso. Se observa una distribución bastante uniforme con respecto a las veces que se ha usado un paso, si bien el primer paso destaca por ser el más usado.

La media de movimientos de cada paso sube hasta el paso 3º, baja en el 4º y luego vuelve a subir. Se ve que el paso 3º es el que más movimientos requiere de media, debido a la preparación que hay que hacer antes de aplicar un macrooperador.

Una serie de gráficos realizados con las 20.000 pruebas, muestran la relación entre los distintos valores obtenido en las estadísticas.

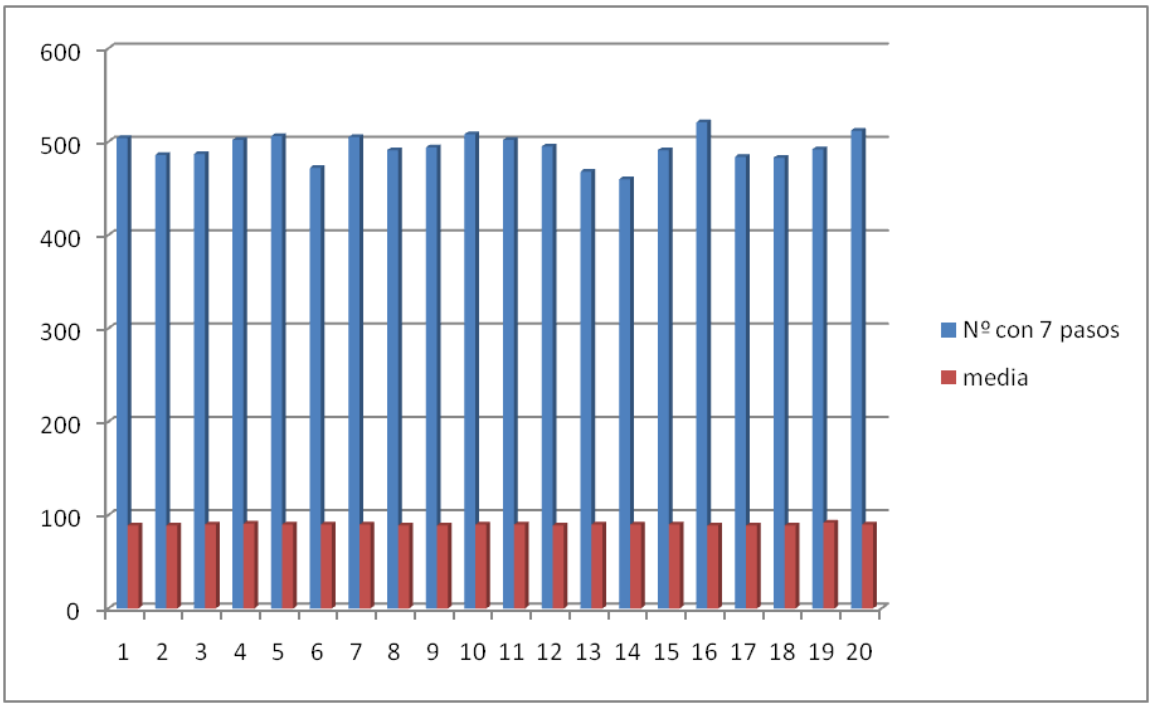



*Gráfico que muestra el número de cubos que han necesitado 7 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 123: Cubos 7 pasos**

Se observa una distribución bastante uniforme. Cerca de la mitad de los cubos necesitan los 7 pasos para resolverse, con un total de aproximadamente 90 movimientos para ello.

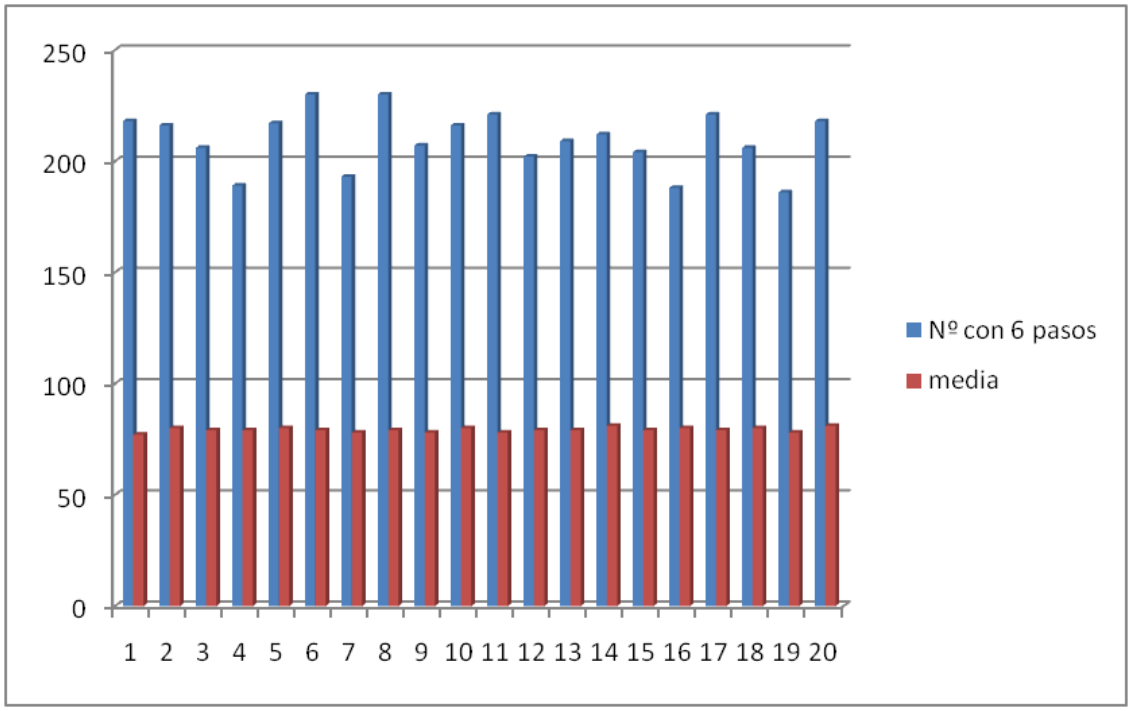



*Gráfico que muestra el número de cubos que han necesitado 6 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 124: Cubos 6 pasos**

Se observa una distribución bastante uniforme, aunque no tanto como en el anterior caso. Tenemos que menos de la cuarta parte de los cubos requieren sólo 6 pasos para resolverse, con una media aproximada de 80 movimientos.

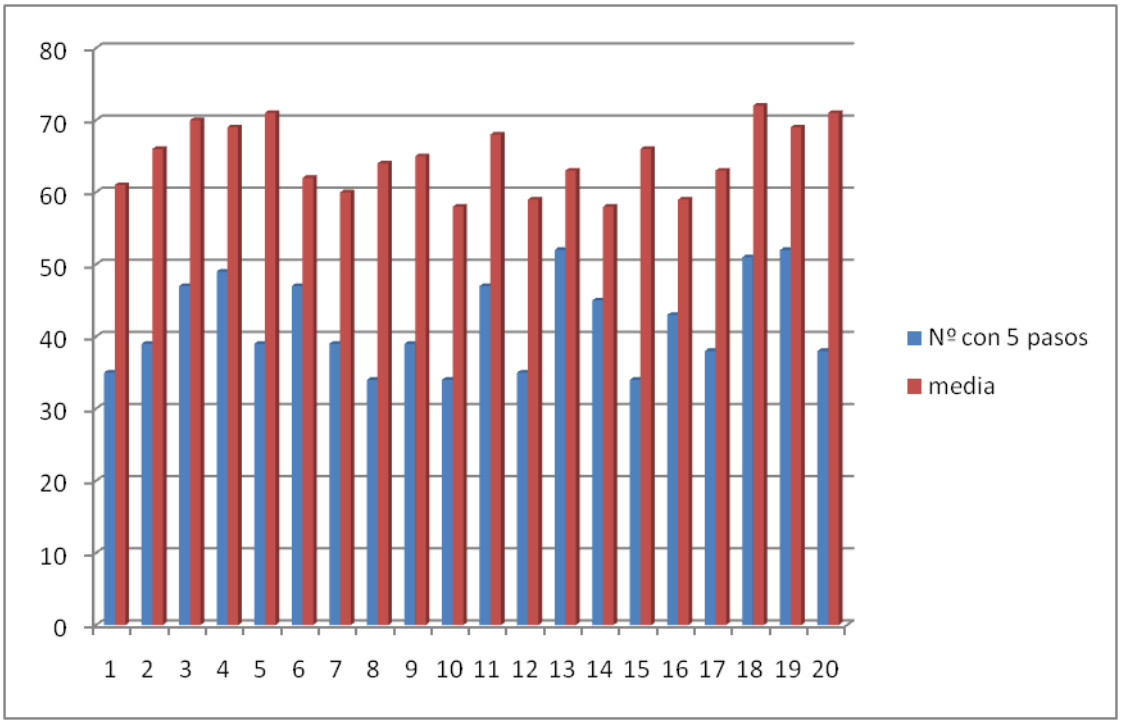



*Gráfico que muestra el número de cubos que han necesitado 5 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 125: Cubos 5 pasos**

Como dato curioso, se observa que el número de movimientos para resolver el cubo es superior al número de casos que han necesitado sólo 5 pasos para resolverse. La oscilación de los valores también es superior que en casos anteriores. Tenemos que aproximadamente 40 casos necesitan 5 pasos, con unos 65 movimientos.

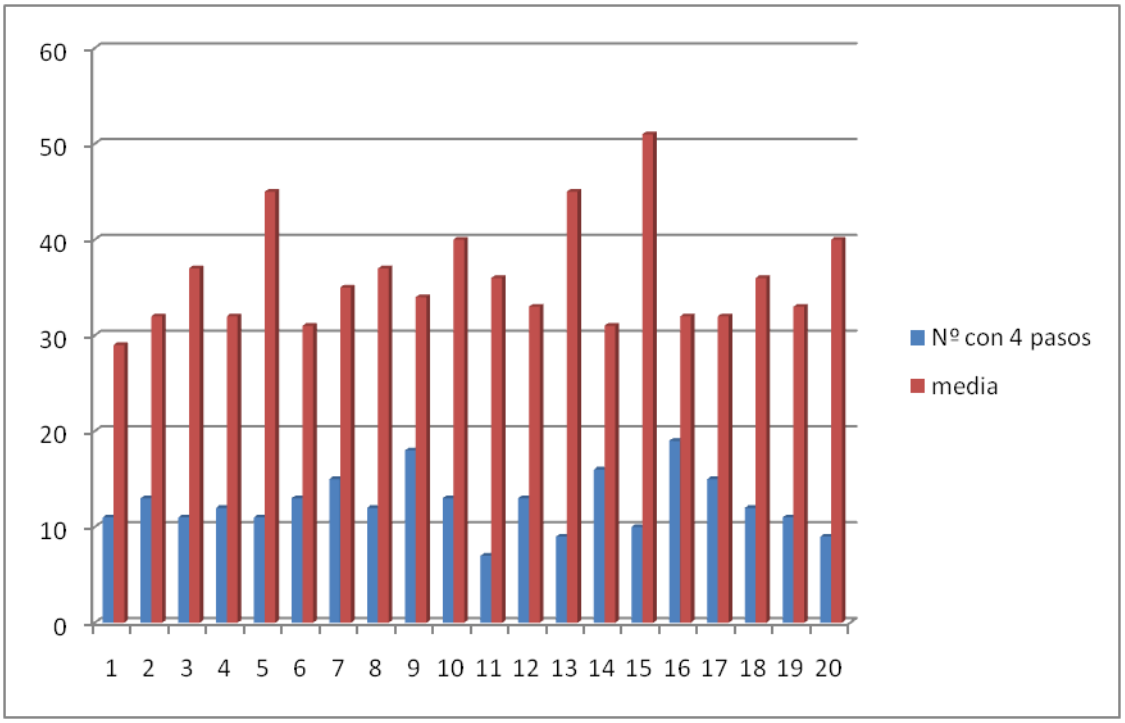



*Gráfico que muestra el número de cubos que han necesitado 4 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 126: Cubos 4 pasos**

Tenemos un caso parecido al anterior, pero más acentuado. Las variaciones son mayores, tanto en el número de casos con en el número de operaciones. Vemos que algo más de 12 casos requieren 4 pasos para resolverse con aproximadamente 30 movimientos.

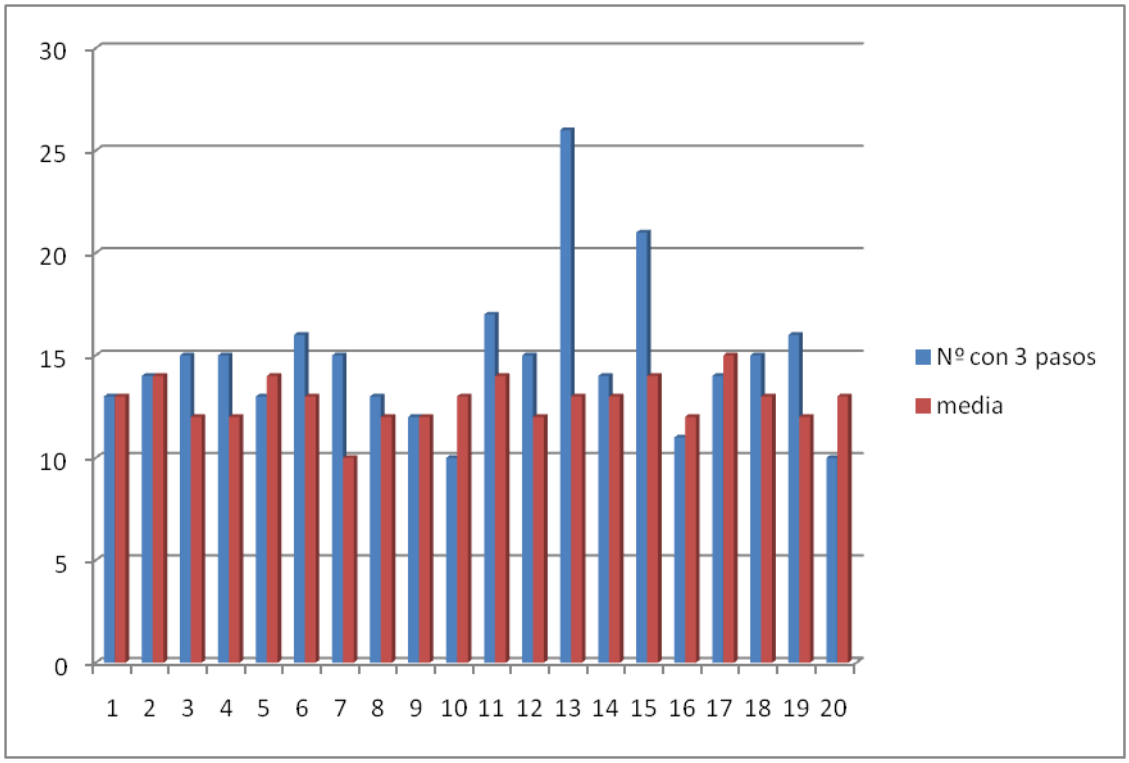



*Gráfico que muestra el número de cubos que han necesitado 3 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 127: Cubos 3 pasos**

Se observa que una de las pruebas tiene un pico con 26 casos, pero si lo excluimos, vemos que las variaciones son similares a casos anteriores. 14 casos necesitan 3 pasos para resolverse, con una media de 13 movimientos. Se observa un pequeño repunte en el número de casos con respecto al gráfico anterior.

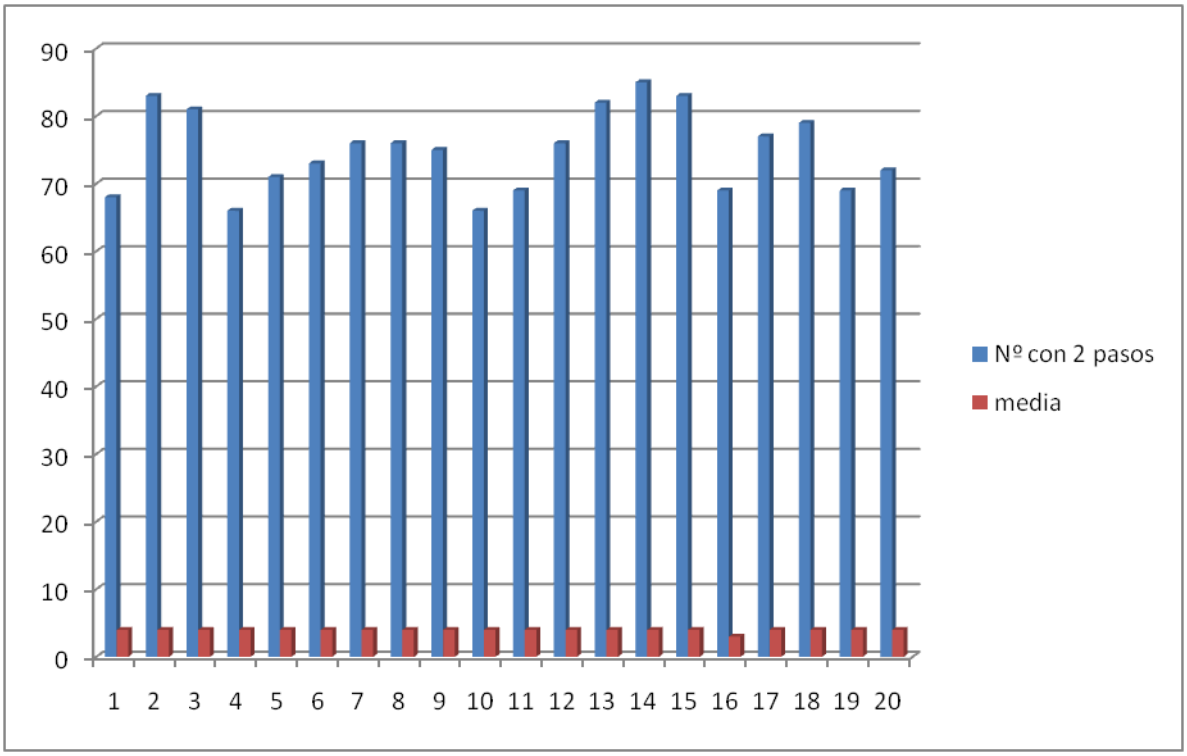



*Gráfico que muestra el número de cubos que han necesitado 2 pasos para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 128: Cubos 2 pasos**

Se observa que el número de casos ha subido considerablemente, pero sigue oscilando bastante. Por otro lado, el número de movimientos necesarios se mantiene constante. Aproximadamente 70 casos necesitan 2 pasos para resolverse con una media de 4 movimientos necesarios.

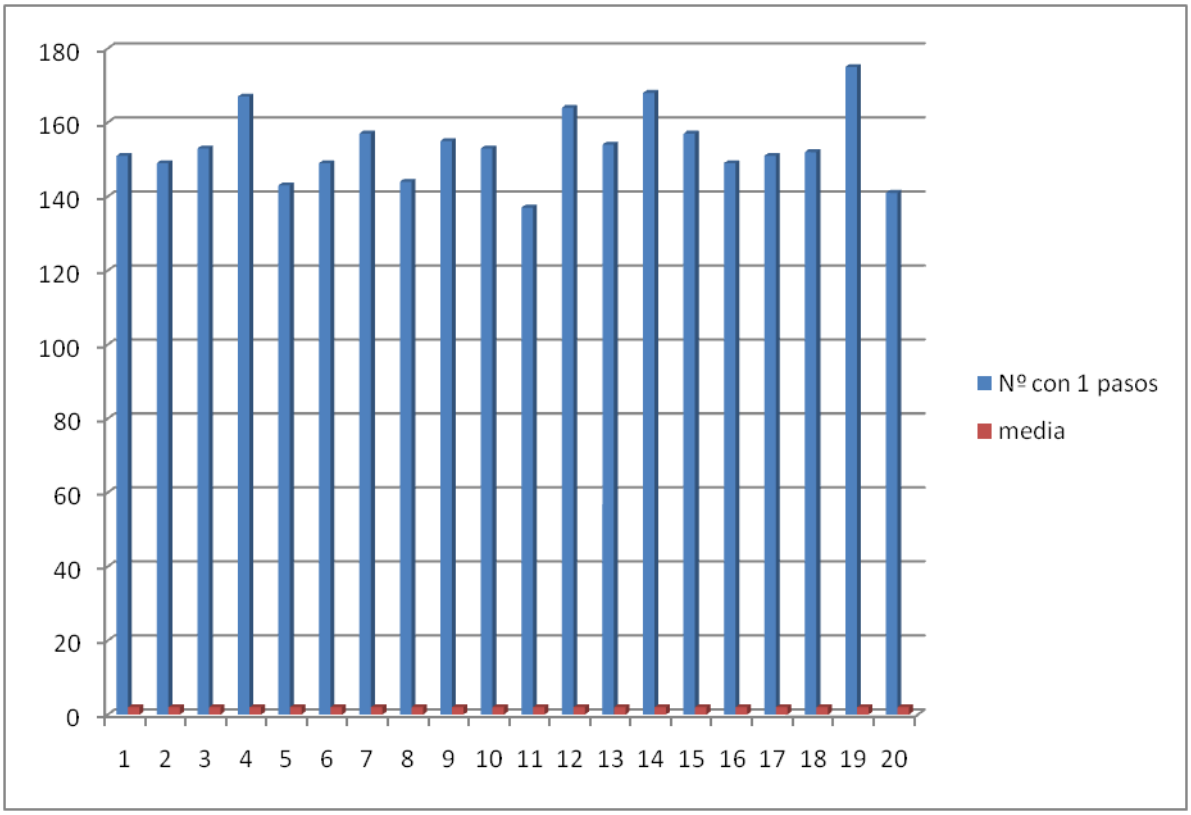



*Gráfico que muestra el número de cubos que han necesitado 1 paso para resolverse (azul) y el número medio de movimientos necesarios para resolverlos.*

**Ilustración 129: Cubos 1 pasos**

Se observa un importante repunte en el número de casos, asimismo la variación en el número de casos se reduce. La media de movimientos necesarios para resolverlo permanece constante en todas las pruebas. Aproximadamente 150 casos requieren 1 paso para resolverse, con 2 movimientos.

En general, se puede ver una distribución en forma de U en el número de casos. La mitad de los casos requerirán 7 pasos. Luego esta cifra se va reduciendo hasta que llegamos a los 3 pasos, que tienen ligeramente más casos. Pero esta diferencia se acentúa mucho en los casos en que son necesarios 2 o 1 pasos.

## 8. Líneas futuras

Se ha podido resolver el cubo de rubik de manera rápida adaptando el método de resolución de principiantes, y ello se puede aplicar a otros puzles similares.

Se pueden implementar metodologías de resolución pensadas para personas para resolver cubos de dimensiones superiores, que existen hasta de 11x11x11 pero de manera teórica pueden ser de cualquier dimensión.

Teniendo en cuenta que el espacio de estados crece exponencialmente al aumentar las dimensiones implementar métodos pensados para personas puede hacer que cubos muy grandes se resuelvan en un tiempo razonable

Los métodos para resolver cubos de dimensiones superiores siempre simplifican al cubo de rubik de 3x3x3 para resolverlo, así que se podría implementar una aplicación que resolviera cualquier cubo de nxnxn, puesto que hay similitud en el método y en los casos especiales (problemas de paridad), así que se podría generalizar.

Para hacer la aplicación más amigable se podría desarrollar una interfaz gráfica que leyera los datos de entrada y representara los movimientos en un cubo en 3 dimensiones.

Siguiendo esta línea se podría hacer que los datos de entrada de dicha interfaz pudieran ser imágenes tomadas con una cámara, con lo que sólo habría que hacer una foto del cubo de entrada (una foto por cara) y que de esta forma lo resolviera.

Se podría implementar el método existente para expertos, y comprobar qué método tarda menos, y si se reduce el número de movimientos necesarios.

Se pueden usar otras métricas para medir el coste de hacer el cubo.

También se pueden resolver otras figuras similares, como el megaminx y sus semejantes (petaminx, teraminx…) adaptando los métodos mencionados.

Sería interesante adaptar todas las aplicaciones mencionadas a dispositivos móviles (*Smart phones* y *tablets*) que cuentan con cámara y una capacidad de cálculo suficiente como para mantener un tiempo de ejecución bajo.

# 9. Planificación y presupuesto

A continuación se especifican las partes de las que consta este proyecto, la duración estimada de las mismas y la duración real, justificando la diferencia que haya entre ambas. También se hace un estudio económico del proyecto con el fin de hacer un presupuesto del mismo.

## 9.1. Planificación

Análisis de las tareas en las que se divide el proyecto, incluyendo duración estimada y real, diagramas de Gantt para ver la precedencia de tareas y una justificación de la desviación entre lo estimado y lo real. Las tareas que componen el proyecto son:

- *Definición del problema:* Analizar el entorno del proyecto, ver opciones y definir los límites del mismo. Se definen las características del proyecto, y se establece un objetivo.
- *Diseño:* Se propone una metodología a seguir, y se define el tipo de búsqueda que se va a usar, las heurísticas y las estructuras de datos principales.
- *Implementación:* Se implementa en lenguaje C el diseño propuesto, adaptando los elementos definidos a las características del lenguaje usado.
- *Pruebas:* Se realizan múltiples pruebas para verificar la integridad del sistema creado, comprobar que los resultados obtenidos son correctos y son los esperados.
- *Estadísticas:* Se analizan los resultados obtenidos para hacer un estudio del sistema. Se comparan los resultados de los dos sistemas implementados.
- *Documentación:* Se redacta esta documentación. Se realiza durante toda la duración del proyecto.
- *Presentación:* Se diseña la presentación del proyecto. Se decide el material que se presenta y el formato de la presentación.

Estas son las partes de las que se compone el proyecto, y muchas se pueden desarrollar en paralelo, mientras que para otras se tienen que cumplir unos requisitos.

### 9.1.1. Planificación estimada

Lo que se estimaba que duraba cada parte del proyecto. Se adjunta un gráfico de Gantt para mostrar las dependencias de las distintas fases de desarrollo.

| Tarea | Duración | Fecha de inicio | Fecha de fin |
|---|---|---|---|
| Definición del problema | 4 semanas | 15/02/11 | 15/03/11 |
| Diseño | 4 semanas | 15/03/11 | 15/04/11 |
| Implementación | 13 semanas | 15/04/11 | 20/07/11 |
| Pruebas | 14 semanas | 15/04/11 | 27/07/11 |
| Estadísticas | 2 semanas | 27/07/11 | 11/08/11 |
| Documentación | 24 semanas | 15/03/11 | 20/09/11 |
| Presentación | 2 semanas | 20/09/11 | 7/10/11 |

Tabla 1: Planificación estimada

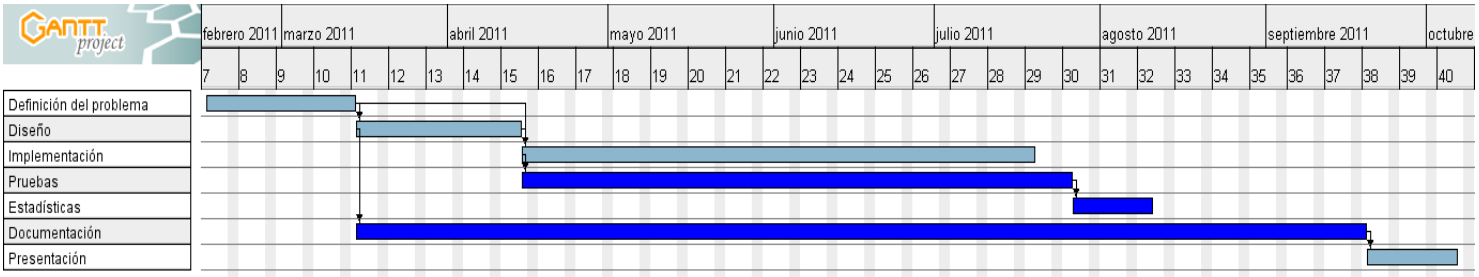



Tabla 2: Gantt estimado

### 9.1.2. Planificación real

Duración real de las tareas

| Tarea | Duración | Fecha de inicio | Fecha de fin |
|---|---|---|---|
| Definición del problema | 4 semanas | 1/03/11 | 1/04/11 |
| Diseño | 6 semanas | 1/04/11 | 15/05/11 |
| Implementación | 20 semanas | 15/05/11 | 15/10/11 |
| Pruebas | 21 semanas | 15/05/11 | 22/10/11 |
| Estadísticas | 1 semana | 22/10/11 | 29/10/11 |
| Documentación | 31 semanas | 1/04/11 | 1/12/11 |
| Presentación | 2 semanas | 1/12/11 | 15/12/11 |

Tabla 3: Plan real

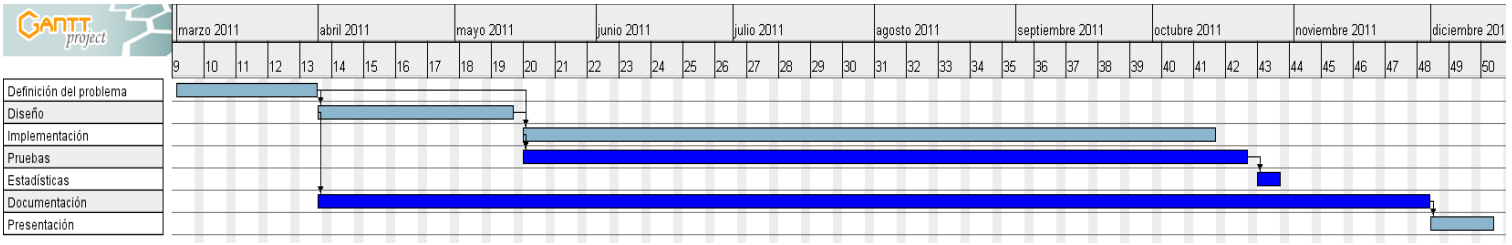



Tabla 4: Gantt real

Se observa un aumento de muchas tareas. El diseño fue un paso que entrañó alguna dificultad para buscar un algoritmo que se ajustara bien al objetivo del proyecto, por lo que hubo que desechar alguna idea y alagar el proceso. La implementación ha sido una tarea mucho más costosa de lo que se pensó inicialmente, hubo complicaciones a la hora de programar los diseños propuestos, además la codificación de macrooperadores fue más costosa de lo esperado.

Las pruebas se han ampliado porque se realizan durante todo el proceso de implementación, así que tienen que durar como poco lo mismo que dura la implementación. Pasa lo mismo con la documentación, se va generando documentación a lo largo de todo el proyecto, y si cualquier otra tarea se alarga, ésta también.

Las estadísticas se han generado en la mitad del tiempo esperado.

## 9.2. Hardware y software usado

Lo que se ha necesitado para el desarrollo del proyecto.

### 9.2.1. Hardware

- Macbook 2.2 GHz Dual core. 4 GB RAM DDR3
- PC 3.2 GHz Dual core. 3 GB RAM DDR2

### 9.2.2. Software

*Sistemas operativos:*

- Mac OS Snow leopard
- Microsoft Windows XP
- Debian Squeeze

*Transferencia de archivos*

- SSH secure shell
- Fugu
- Cyber duck

*Editores de texto*

- Notepad ++
- Xcode (usado como editor)

*Compilador*

- Gcc compilador usual

*Creación y edición de imágenes*

- Blender
- Gimp

*Programas de ofimática*

- Microsoft Office word
- Microsoft Office power point
- Microsoft Office Excel
- GanttProject

## 9.3. Análisis económico

Análisis del coste del proyecto. Se desglosa el coste del proyecto, separado según el tipo de recurso utilizado, luego se hace un coste total. Se muestra el coste estimado (según la planificación estimada) y el coste real.

### 9.3.1. Recursos humanos

Personas que han intervenido en el proceso de desarrollo del proyecto.

- *Supervisor:* Encargado de seguir el desarrollo del proyecto y de guiar al desarrollador. También se encarga de corregir los documentos y ayudar a hacer una presentación.
- *Desarrollador:* Encargado de hacer el proyecto, siguiendo las guías del supervisor. Implementa el código y genera la documentación.

Tomamos que se trabaja una media de 35 horas/semana para el desarrollador y dos horas /semana para el supervisor

*Coste estimado*

| Recurso | Coste por hora | Horas dedicadas | Coste total |
|---|---|---|---|
| Supervisor | 30€ | 50 | 1500€ |
| Desarrollador | 20€ | 700 | 14000€ |
| | | **Total** | 15500€ |

Tabla 5: Recursos estimados

*Coste real*

| Recurso | Coste por hora | Horas dedicadas | Coste total |
|---|---|---|---|
| Supervisor | 30€ | 58 | 1740€ |
| Desarrollador | 20€ | 1015 | 20300€ |
| | | **Total** | 22040€ |

Tabla 6: Recursos real

Como la duración del proyecto ha sido mayor que la estimada, también lo han sido los costes estimados.

### 9.3.2. Recursos de hardware

Los ordenadores y otros elementos físicos que se han usado durante el proyecto, que se detallaron en el punto anterior. Se estima su coste según su amortización y la duración del proyecto.

*Coste estimado*

| Recurso | Coste | Vida útil (meses) | Tiempo de uso (meses) | Coste total |
|---|---|---|---|---|
| PC | 700€ | 36 | 6 | 116'70€ |
| Macbook | 1080€ | 60 | 6 | 108€ |
| | | | **Total** | 224'70€ |

Tabla 7: Hardware estimado

*Coste real*

| Recurso | Coste | Vida útil (meses) | Tiempo de uso (meses) | Coste total |
|---|---|---|---|---|
| PC | 700€ | 36 | 7 | 136'10€ |
| Macbook | 1080€ | 60 | 7 | 126€ |
| | | | **Total** | 262'10€ |

Tabla 8: Hardware real

### 9.3.3. Recursos de software

Los programas que han sido utilizados durante el desarrollo del proyecto, que se detallaron en el punto anterior. Se estima su coste según su amortización y la duración del proyecto.

*Coste estimado*

| Recurso | Coste | Tiempo útil (meses) | Tiempo de uso (meses) | Coste total |
|---|---|---|---|---|
| Mac OS Snow Leopard | 110€ | 24 | 6 | 27'50€ |
| Windows XP | 200€ | 36 | 6 | 33'30€ |
| SSH secure Shell | 0€ | 24 | 6 | 0€ |
| Fugu | 10€ | 24 | 6 | 2'50€ |
| Cyber duck | 10€ | 24 | 6 | 2'50€ |
| Notepad ++ | 0€ | 24 | 6 | 0€ |
| Xcode | 0€ | 24 | 6 | 0€ |
| Gcc compilador | 0€ | 24 | 6 | 0€ |
| Blender | 10€ | 24 | 6 | 2'50€ |
| Gimp | 10€ | 24 | 6 | 2'50€ |
| Microsoft Office word | 50€ | 24 | 6 | 12'50€ |
| Microsoft Office power point | 50€ | 24 | 6 | 12'50€ |
| Microsoft Office excel | 50€ | 24 | 6 | 12'50€ |
| GanttProject | 0€ | 24 | 6 | 0€ |
| | | | **Total** | 108'30€ |

Tabla 9: Software estimado

*Coste real*

| Recurso | Coste | Tiempo útil (meses) | Tiempo de uso (meses) | Coste total |
|---|---|---|---|---|
| Mac OS Snow Leopard | 110€ | 24 | 7 | 32'08€ |
| Windows XP | 200€ | 36 | 7 | 38'90€ |
| SSH secure Shell | 0€ | 24 | 7 | 0€ |
| Fugu | 10€ | 24 | 7 | 2'91€ |
| Cyber duck | 10€ | 24 | 7 | 2'91€ |
| Notepad ++ | 0€ | 24 | 7 | 0€ |
| Xcode | 0€ | 24 | 7 | 0€ |
| Gcc compilador | 0€ | 24 | 7 | 0€ |
| Blender | 10€ | 24 | 7 | 2'91€ |
| Gimp | 10€ | 24 | 7 | 2'91€ |
| Microsoft Office word | 50€ | 24 | 7 | 14'58€ |
| Microsoft Office power point | 50€ | 24 | 7 | 14'58€ |
| Microsoft Office excel | 50€ | 24 | 7 | 14'58€ |
| GanttProject | 0€ | 24 | 7 | 0€ |
| | | | **Total** | 126'36€ |

Tabla 10: Software real

### 9.3.4. Resumen

*Coste estimado*

| Recurso | Coste |
|---|---|
| Recursos humanos | 15500€ |
| Recursos de hardware | 224'70€ |
| Recursos de software | 108'30€ |
| **Total** | **15833€** |

Tabla 11: Resumen estimado

*Coste real*

| Recurso | Coste |
|---|---|
| Recursos humanos | 22040€ |
| Recursos de hardware | 262'10€ |
| Recursos de software | 126'36€ |
| **Total** | **22482'40€** |

Tabla 12: Resumen real

Existe una desviación con respecto a lo estimado porque el proyecto se ha alargado más de lo esperado en alguna de sus fases.

## 10. Bibliografía

*Método de resolución para principiantes:* http://www.rubikaz.com

*Demostración del número de combinaciones:* http://www.rubikaz.com/democomb.php

*Página oficial de competiciones con el cubo de rubik:* http://worldcubeassociation.org/

*Página oficial de resolución en 20 pasos o menos:* http://www.cube20.org/

*Simetrías en un cubo:* http://geometriadinamica.es/Geometria/Cuerpos/Simetrias-del-cubo.html

*Stomachion:* http://mathandarte.blogspot.com/2010/10/stomachion-el-puzzle-mas-antiguo-del.html

*Torres de Hanoi:* http://www.rodoval.com/heureka/hanoi/index.html

Panqueques: http://www.lnds.net/blog/2010/07/los-panqueques-de-bill-gates.html

*Bill Gates panqueques:* http://www.cs.berkeley.edu/~christos/papers/Bounds%20For%20Sorting%20By%20Prefix%20Reversal.pdf

*Topspin:* http://www.jaapsch.net/puzzles/topspin.htm

*15 puzle:* Archer, Aaron F. (1999), "A modern treatment of the 15 puzzle"

*Sokoban:* M. Fryers and M.T. Greene (1995). "Sokoban". *Eureka* (54).

*Rush hour*: http://www.puzzles.com/products/rushhour.htm

*M12:* http://www.scientificamerican.com/media/inline/2008-07/puzzles/m12.html