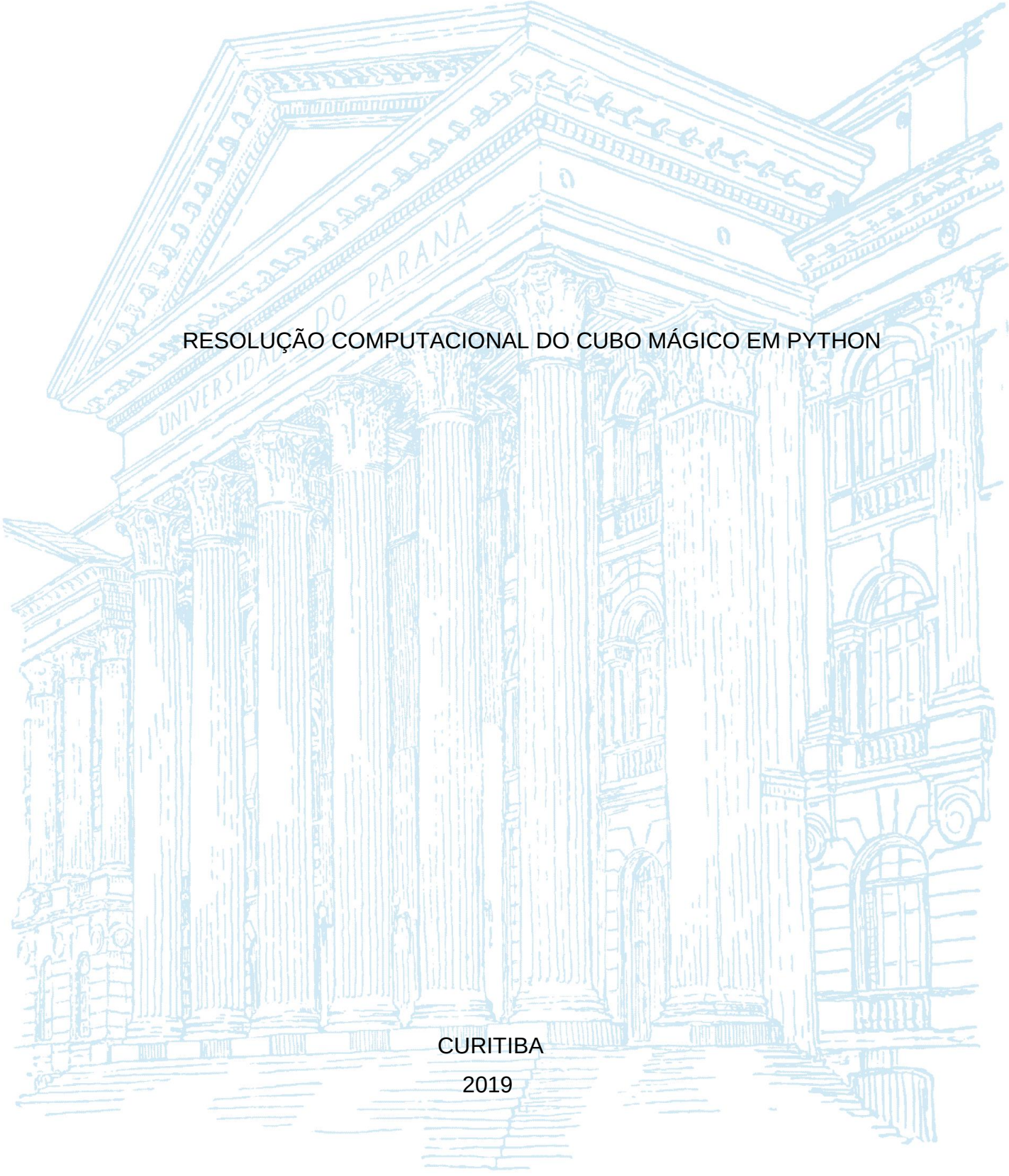
UNIVERSIDADE FEDERAL DO PARANÁ

RAFAEL KURZLOP BRUNKOW




# RESOLUÇÃO COMPUTACIONAL DO CUBO MÁGICO EM PYTHON

CURITIBA

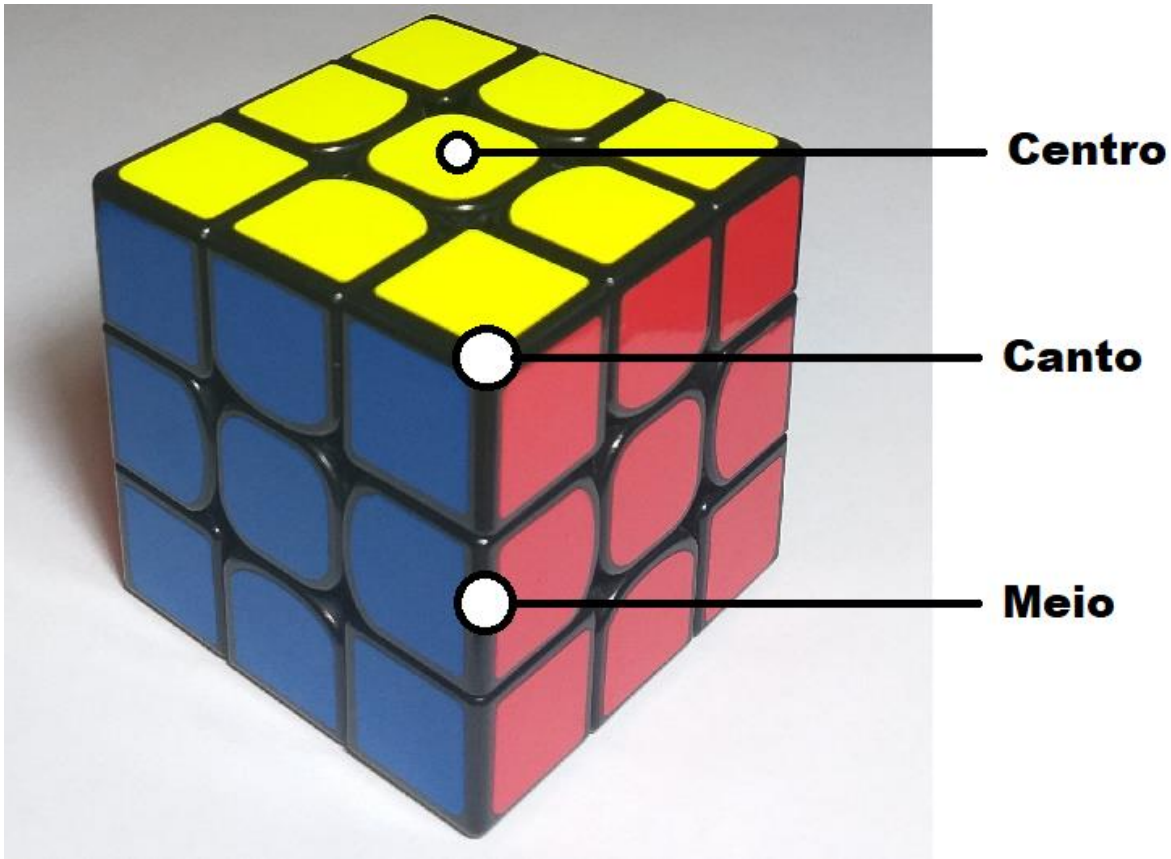
2019

## 2 FUNDAMENTAÇÃO TEÓRICA

### 2.1 CONCEITOS BÁSICOS:

O cubo mágico tem cada uma das suas 6 faces divididas em 9 quadrados menores em que cada um possuí um adesivo colorido. Dessa maneira, o cubo é dividido em 3 tipos de peças: Centros (peças com apenas um adesivo), Meios(peças com dois adesivos) e Cantos (peças com três adesivos).

FIGURA 1 – TIPOS DE PEÇAS




É importante perceber que os centros do cubo são fixos. Centros opostos permanecem opostos, no caso do cubo utilizado para esse trabalho são opostos: branco e amarelo, vermelho e laranja, azul e verde.

Dessa forma, se o centro de uma face do cubo possuí uma determinada cor, para a resolução todos os adesivos daquela face devem possuir a mesma cor, e

além disso, os adesivos devem compor a face com as peças adequadas, de forma que os adesivos das faces adjacentes também estejam corretamente posicionados.

FIGURA 2 – RESOLUÇÃO CORRETA

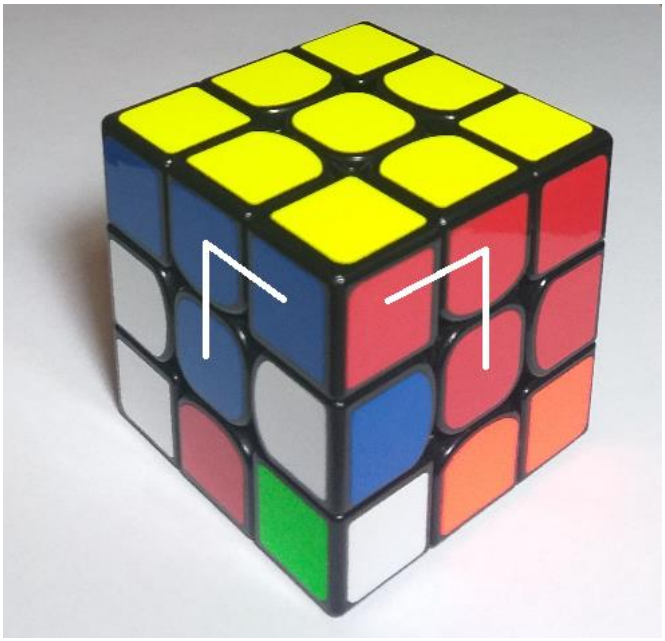
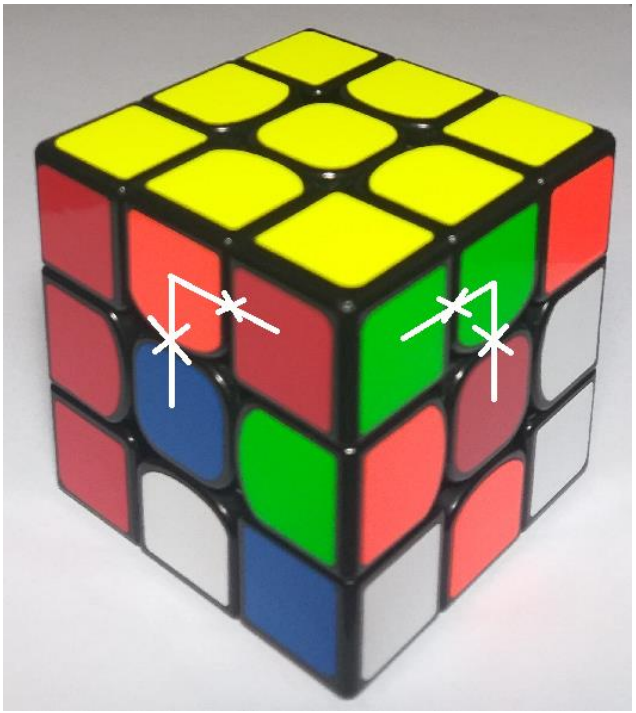


FIGURA 3 – RESOLUÇÃO INCORRETA



## 2.2 MOVIMENTOS E NOTAÇÃO

Para descrevermos de forma eficiente os movimentos do cubo, precisamos de um referencial. No caso deste trabalho, escolhi como padrão a face frontal vermelha e o topo amarelo.

Dessa forma, podemos descrever a rotação de uma face com uma letra maiúscula:

L: esquerda (left)

F: frontal (front)

R: direita (right)

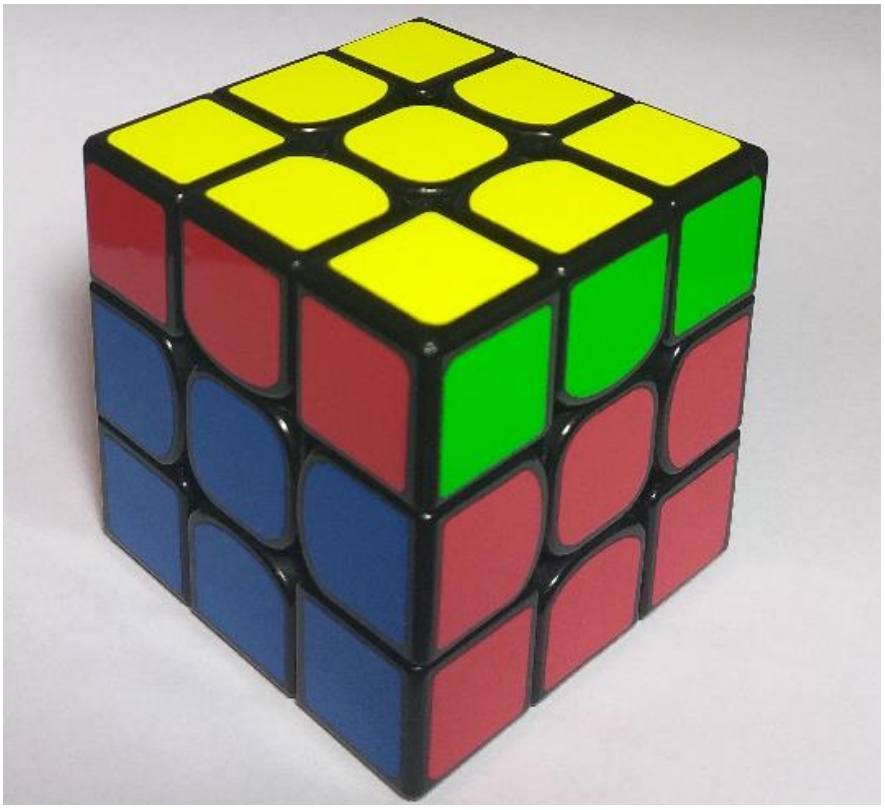
B: traseira (back)

D: inferior (down)

U: superior (up)

Ao ler um algoritmo, entendemos que cada letra indica uma rotação no sentido horário. Para rotações no sentido anti-horário, acrescentamos um apóstrofo, por exemplo: R'

**FIGURA 4 – MOVIMENTO U**



Existem mais notações para descrever movimentos no cubo, por exemplo: M e E, que são rotações das camadas intermediárias. Em tese, não são movimentos

novos. Rotacionar a camada horizontal intermediária (E) é equivalente a rotacionar a face superior e a inferior simultaneamente, porém, essas notações são úteis pois mudam o referencial de faces.

FIGURA 5 – MOVIMENTO E

## 2.3 O MÉTODO

Para a resolução do cubo mágico, geralmente se usa o método denominado CFOP. Esse método não demanda tantos movimentos e pode ter execução muito rápida, entretanto, é muito situacional, o que dificulta sua implementação computacional. O método implementado é o que geralmente se usa para resolução ás cegas. Este método demanda muitos movimentos e pode não ser tão rápido, porém, sua execução é mais prática.

Para descrevê-lo, explicarei um problema análogo. Considere um tabuleiro com 6 espaços, cada espaço possui uma carta associada (de ás a 6). Um desses espaços (no exemplo, o espaço do ás) se chama buffer. Agora distribuímos as 6

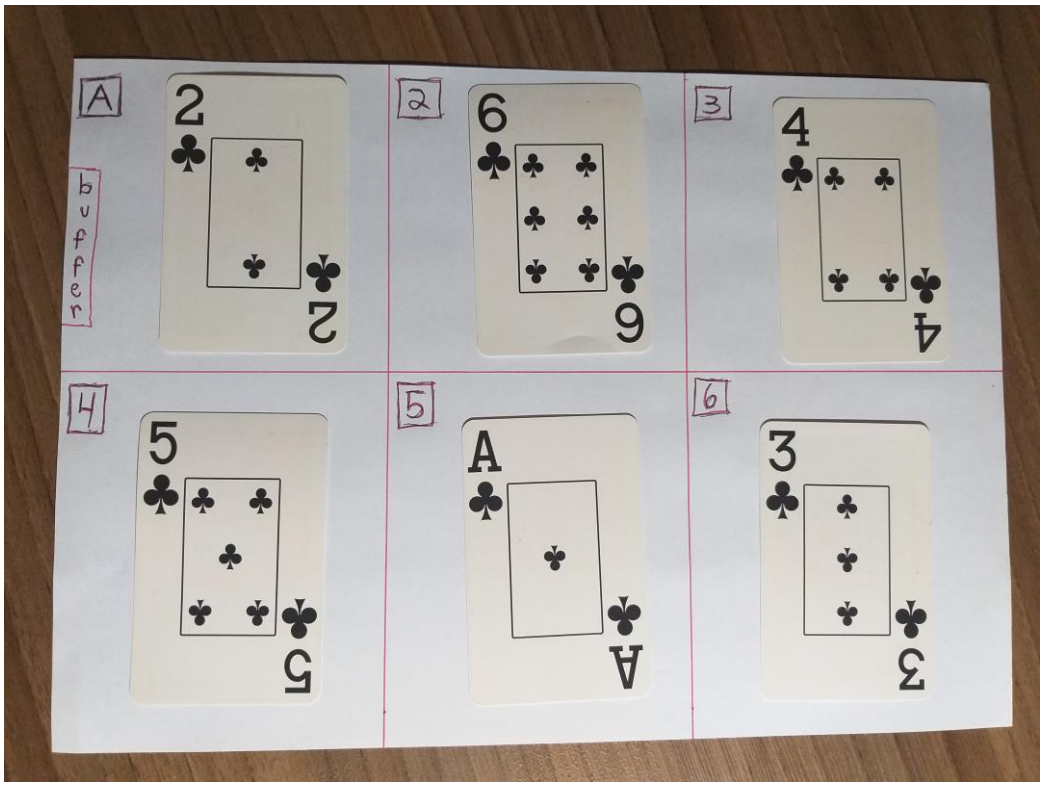
cartas aleatoriamente nos 6 espaços. Nosso objetivo é colocar cada carta em seu espaço associado, seguindo duas regras:

1) Só podemos trocar duas cartas por vez. Não é possível pegar todas as cartas na mão e colocá-las em posição.

2) Toda troca deve envolver a posição buffer.

Se toda troca deve envolver a posição buffer, a maneira de resolver o problema é trocar a carta que está atualmente na posição do buffer com sua posição desejada e repetir o processo.

FIGURA 6.1 – ANALOGIA DAS CARTAS




No exemplo, a carta 2 ocupa a posição do buffer e portanto trocamos o buffer com a posição 2. Dessa forma, a carta 2 passa a estar em sua posição correta. Ao fazer isso, agora temos a carta 6 na posição do buffer, repetimos o processo até que todas as cartas estejam no lugar.

FIGURA 6.2 – ANALOGIA DAS CARTAS

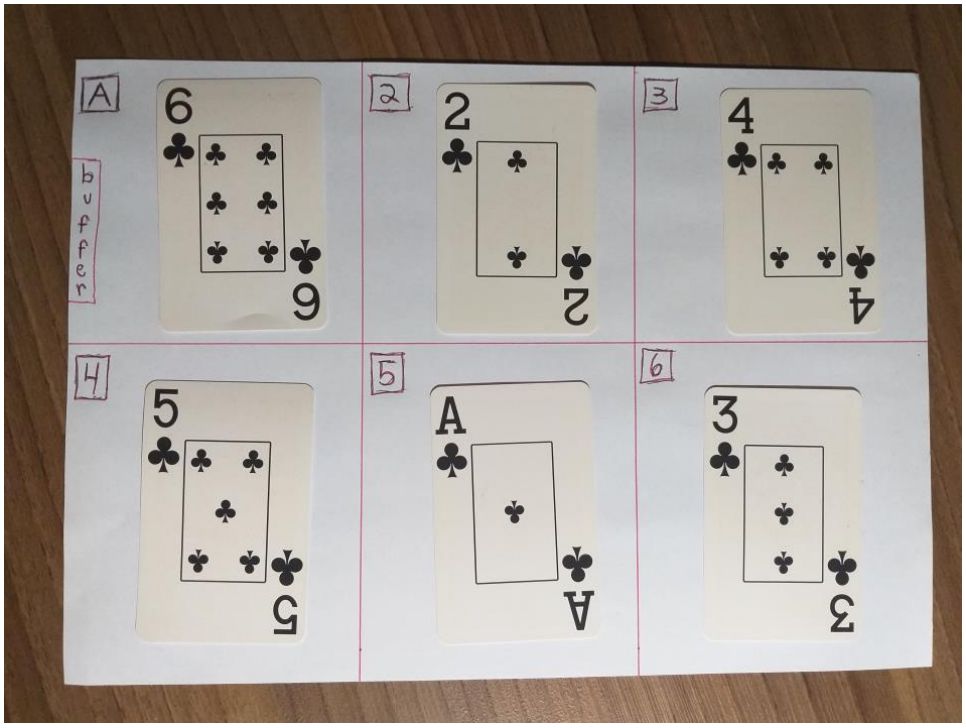



FIGURA 6.3 – ANALOGIA DAS CARTAS

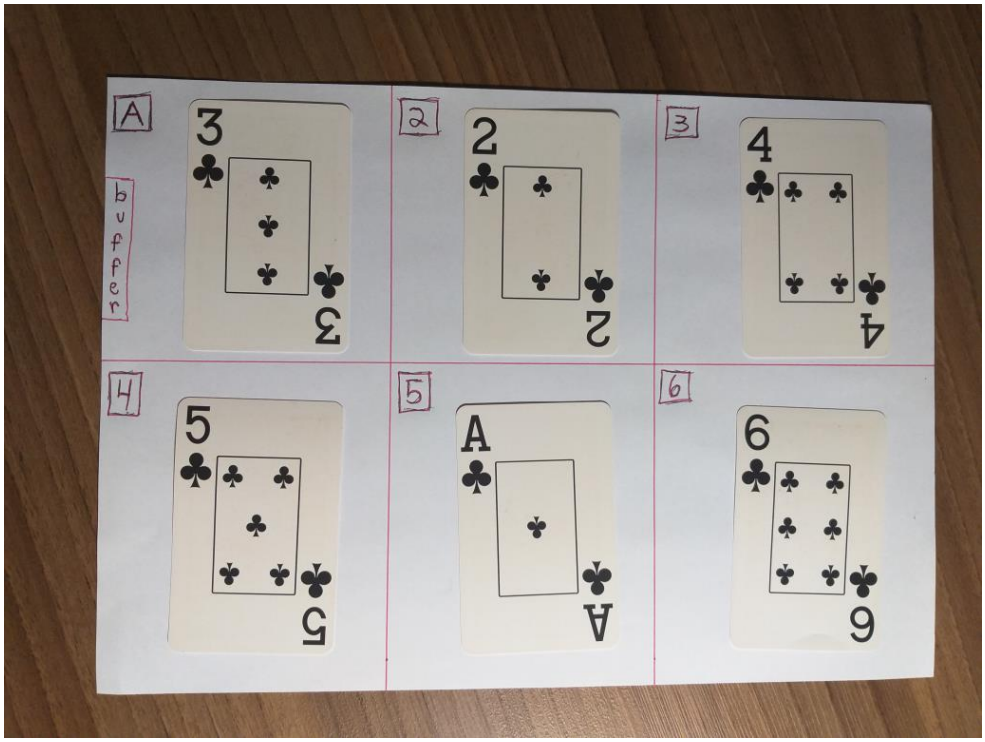


FIGURA 6.3 – ANALOGIA DAS CARTAS

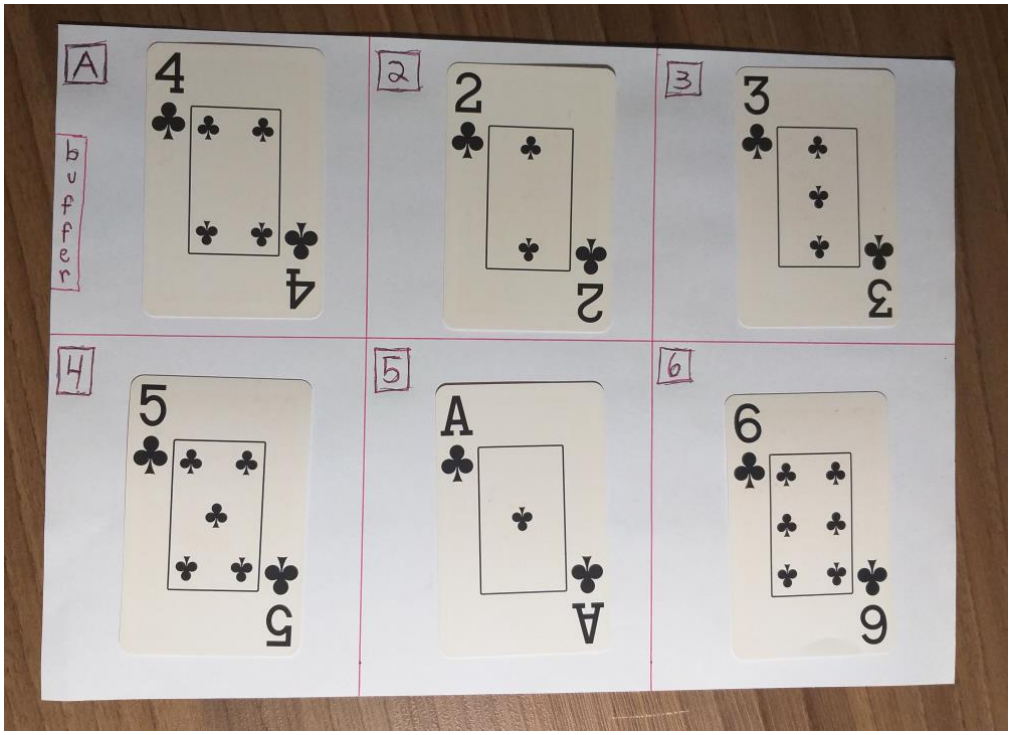



FIGURA 6.4 – ANALOGIA DAS CARTAS

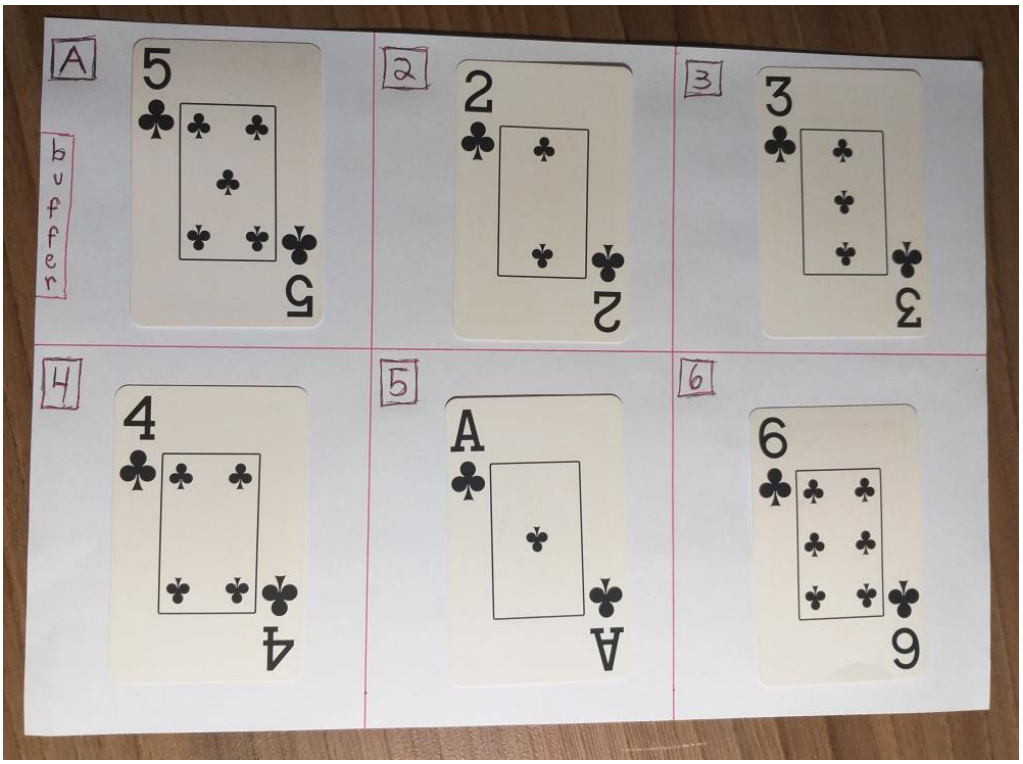
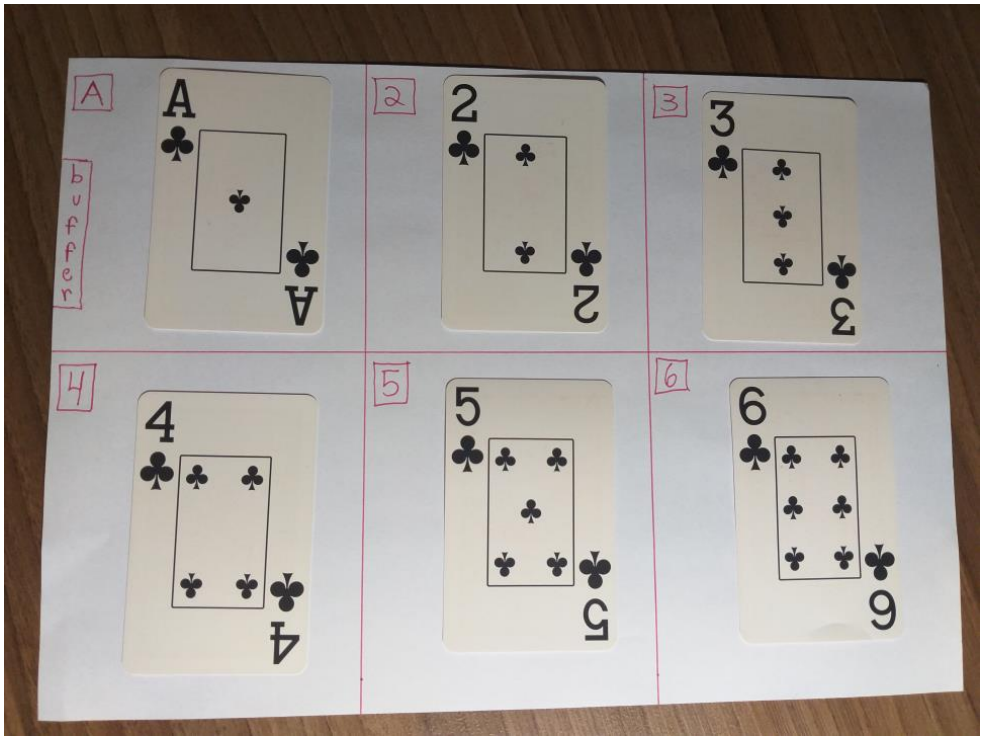


FIGURA 6.5 – ANALOGIA DAS CARTAS




Porém, esse processo possui um problema. Em algum momento, a posição buffer pode estar ocupada por sua carta correspondente, no caso o ás, porém, não necessariamente o tabuleiro está resolvido. Nesse caso, trocamos a carta do buffer (mesmo que já esteja correta) com qualquer outra carta em posição incorreta e continuamos o processo.

FIGURA 6.6 – ANALOGIA DAS CARTAS

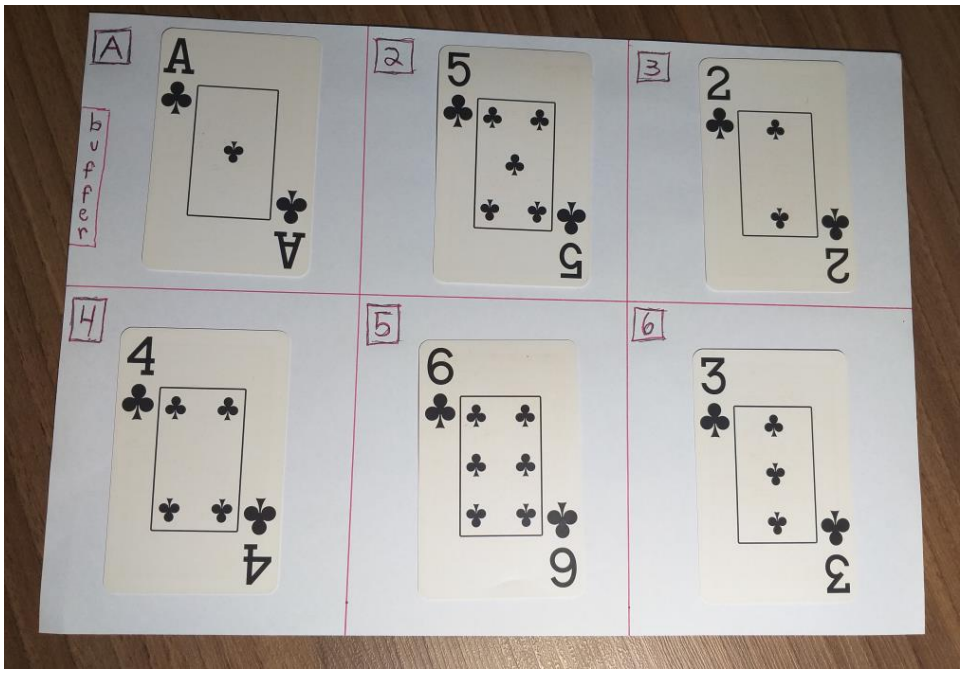
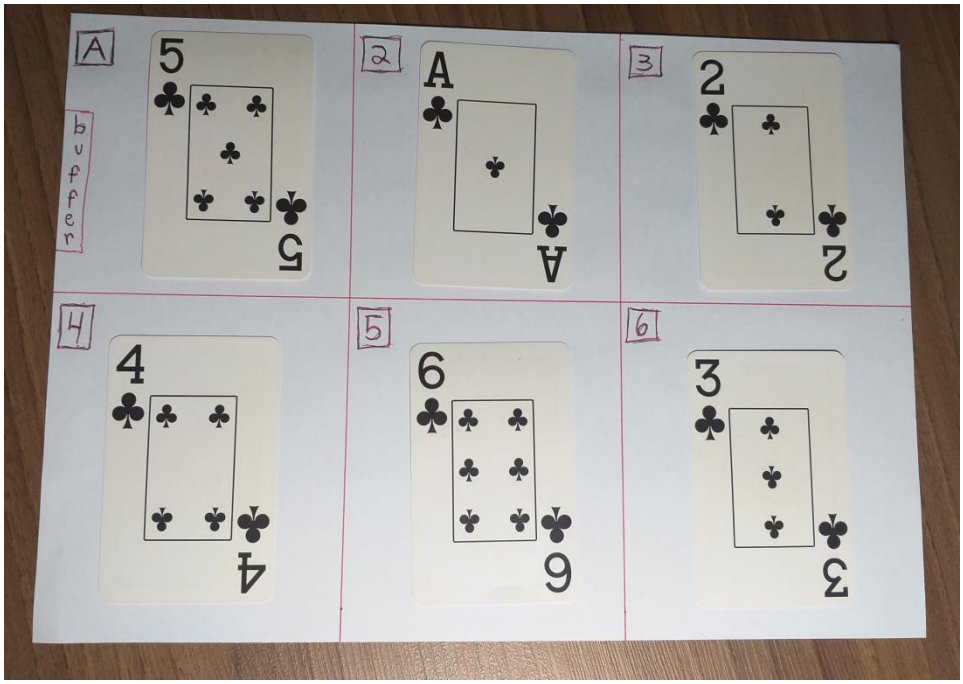


FIGURA 6.7 – ANALOGIA DAS CARTAS




Voltando ao cubo. A resolução é feita em duas etapas quase idênticas. Primeiro resolvemos todas as peças de tipo “meio” e depois resolvemos todas as peças de tipo "canto". Utilizaremos 3 algoritmos fundamentais: permutação de meios, permutação de cantos e resolução de paridade (futuramente explicado).

Começarei explicando a resolução dos meios. Nosso objetivo é realizar no cubo o mesmo processo da analogia com cartas. O tabuleiro é análogo ao cubo (considerando apenas peças de tipo "meio"). Os espaços do tabuleiro são as posições das peças, associadas com a respectiva peça correta para aquela posição. Existe uma posição buffer e desejamos trocar duas peças por vez até que os meios estejam resolvidos.

Para realizarmos a troca de duas peças, utilizaremos o algoritmo "permutação de meios". Infelizmente nos deparamos com dois problemas:

1) Não conseguimos trocar a peça buffer com qualquer peça do cubo (não diretamente). Trocaremos a peça buffer com uma peça específica que chamarei de "posição de troca".

2) O algoritmo de permutação de meios indesejadamente também permuta outras duas peças, o que chamarei de "troca acidental".

Sobre (2), se tomarmos o devido cuidado, a troca acidental não nos atrapalha, afinal, ao executar um número par de permutações, elas voltam ao seu lugar.

Sobre (1), não podemos trocar a peça buffer com qualquer peça, mas podemos executar um algoritmo de "preparação" que coloque a peça que desejamos trocar na posição de troca, então executamos a permutação, e depois realizamos o inverso do algoritmo de preparação para que tudo volte ao seu lugar. O cuidado que devemos tomar é que o algoritmo de preparação não altere a posição das peças que são trocadas acidentalmente.

FIGURA 7.1 – PERMUTAÇÃO DOS MEIOS

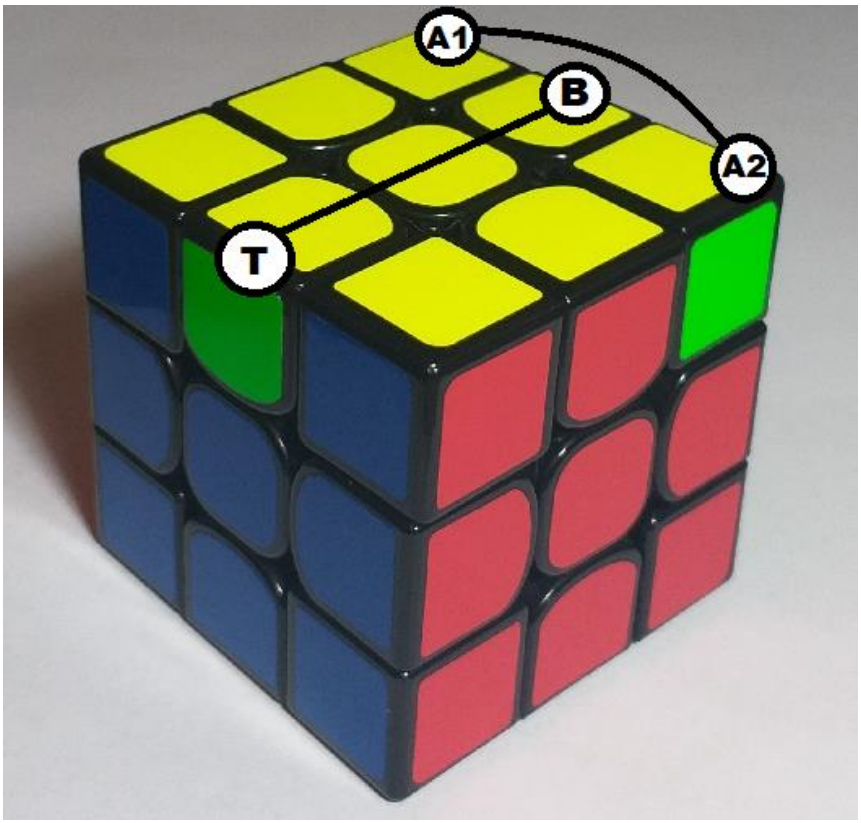
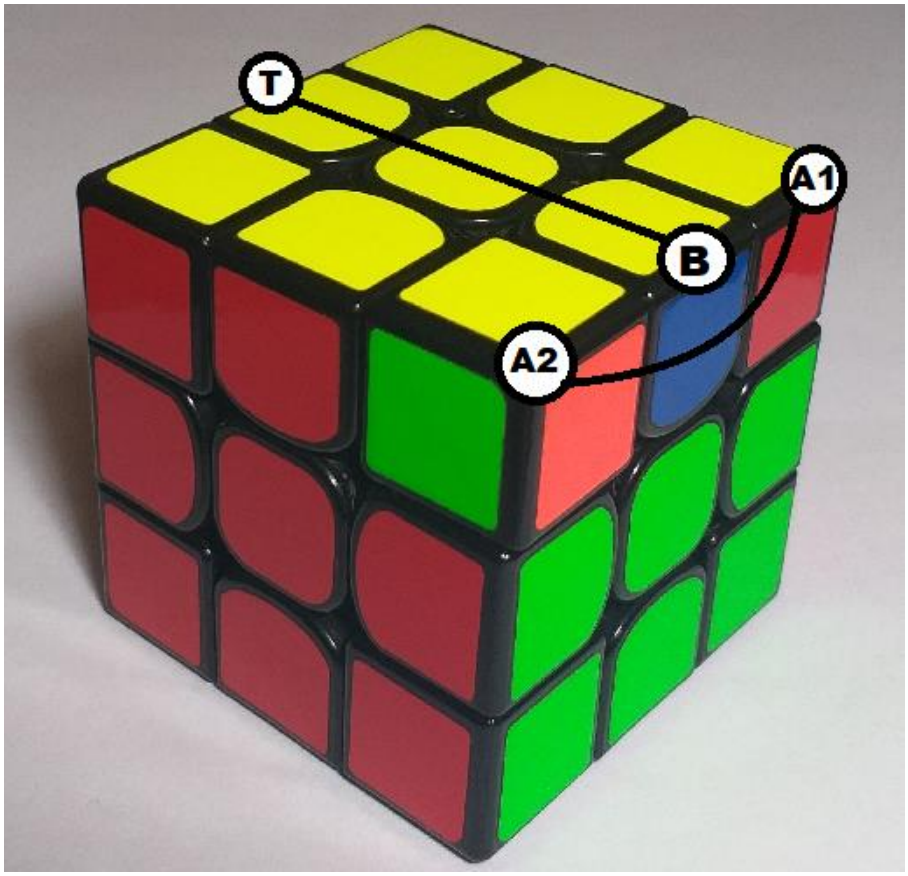


FIGURA 7.2 – PERMUTAÇÃO DOS MEIOS




De maneira resumida, nosso processo consiste em:

1) Verificar para que posição a peça que está no buffer deve ir.

2) Executar um algoritmo de preparação que posicione a peça a ser trocada na posição de troca. Tomando cuidado para não alterar a posição das peças que são trocadas acidentalmente.

3) Executar o algoritmo de permutação de meios.

4) Executar o inverso do algoritmo de preparação, para que tudo volte ao lugar correto.

5) Repetir o processo.

*Caso o buffer já esteja em sua posição (ainda que não orientado corretamente), realizamos o mesmo processo para trocar o buffer com qualquer peça que ainda não esteja resolvida, assim como na analogia das cartas.

**Ao descrever o processo, falamos muito sobre a permutação de peças do cubo, mas na prática, é necessário atentar-se também em qual adesivo da peça estamos trocando. Dessa forma, além de existir a peça buffer, estamos nos concentrando no adesivo buffer.

Vejamos um exemplo. Para facilitar a compreensão, neste exemplo a permutação que precisamos realizar é a última que falta para que o cubo fique solucionado.

FIGURA 8.1 – EXEMPLO DE SOLUÇÃO DE MEIOS

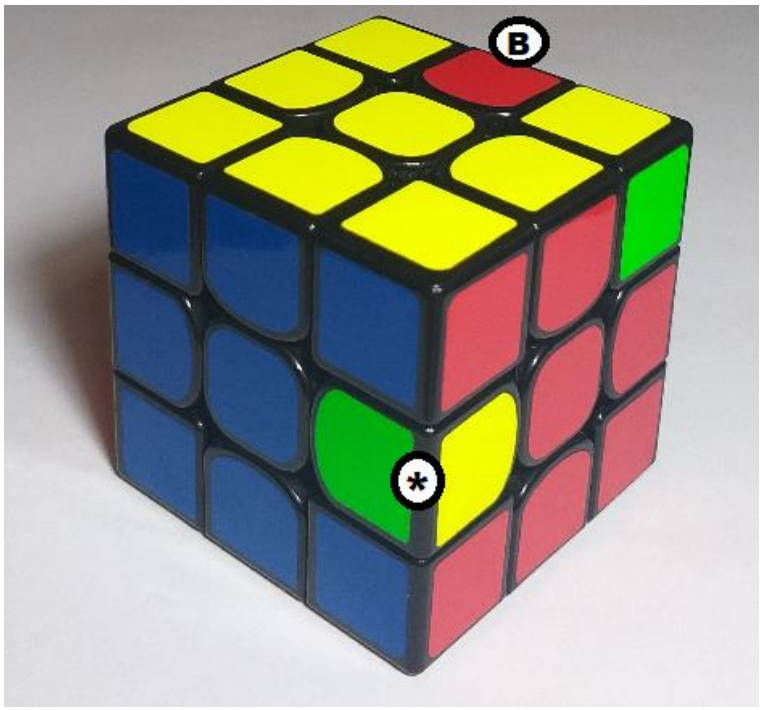



FIGURA 8.2 - EXEMPLO DE SOLUÇÃO DE MEIOS

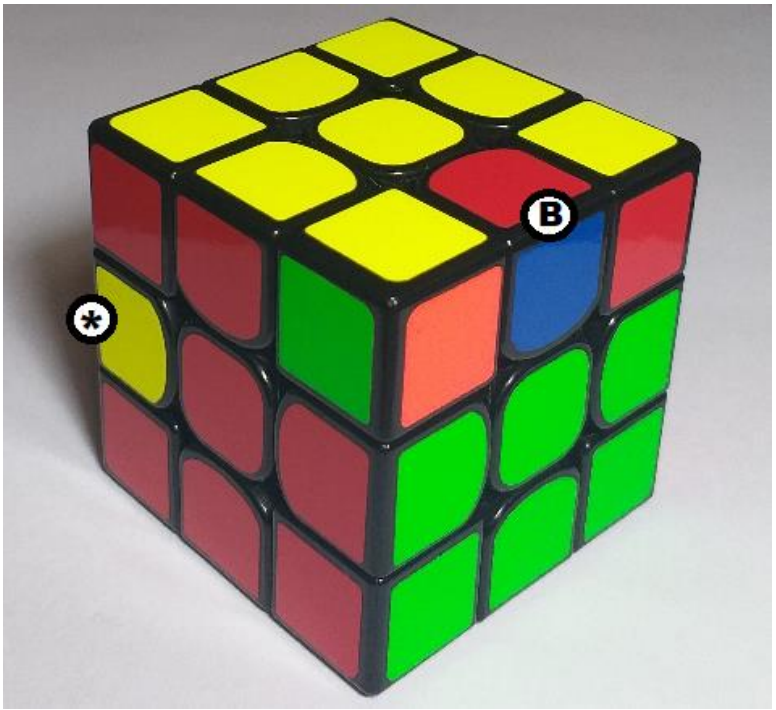


FIGURA 8.3 – EXEMPLO DE SOLUÇÃO DE MEIOS

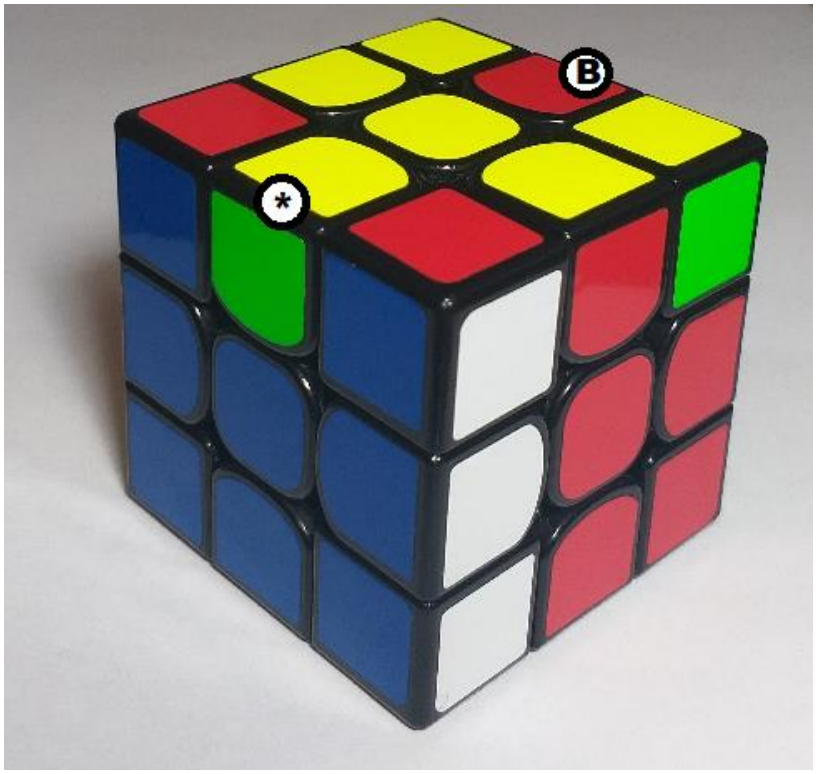
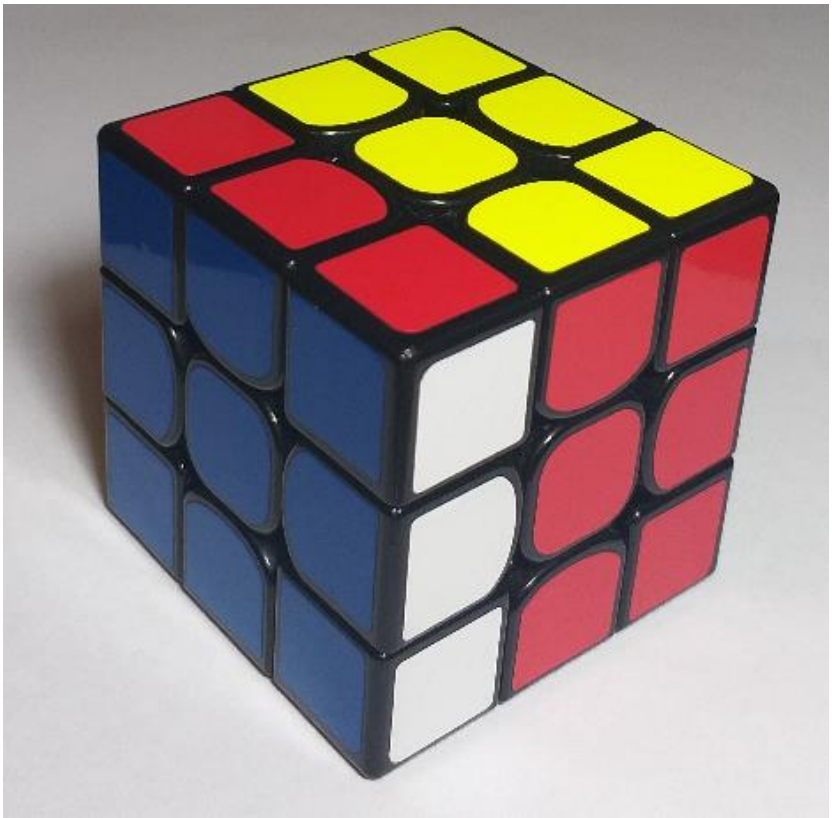



FIGURA 8.4 – EXEMPLO DE SOLUÇÃO DE MEIOS



Após a resolução dos meios, fazemos a resolução dos cantos.

O processo é essencialmente o mesmo, a única diferença é quais são as peças: buffer, posição de troca e as duas que serão trocadas acidentalmente.

Vejamos o resultado da permutação dos cantos em um cubo resolvido:

FIGURA 9.1 – PERMUTAÇÃO DOS CANTOS

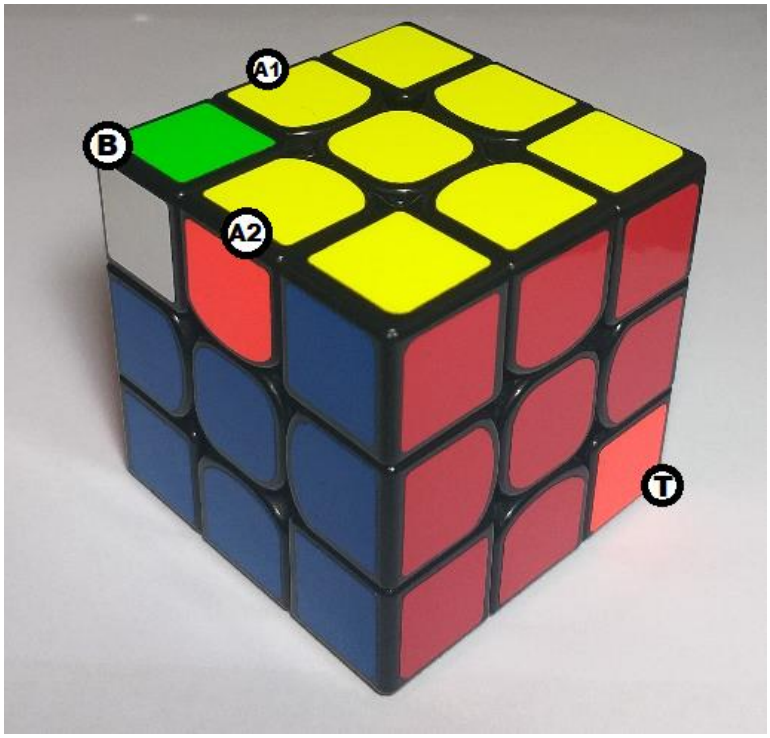
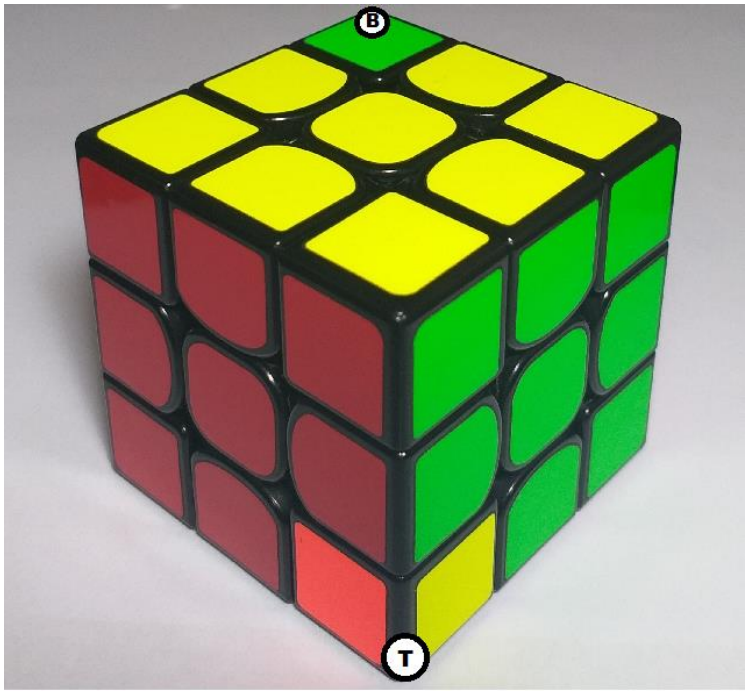



FIGURA 9.2 – PERMUTAÇÃO DOS CANTOS



Com as duas etapas explicadas, agora precisamos falar sobre resolução de paridade.

Se o número de permutações de meios for par, o número de permutações de cantos também será. Se o número de permutações de meios for ímpar, o número de permutações de cantos também será.

Quando resolvemos todos os meios, se tivermos um número ímpar de permutações, a troca acidental não será desfeita, o que tornaria a resolução dos cantos errada. Por isso, executamos um algoritmo de "resolução de paridade". Esse algoritmo se encarrega de inverter as trocas acidentais tanto dos meios quanto dos cantos simultaneamente.

Esse algoritmo deve ser executado após a resolução dos meios e antes da resolução dos cantos.

# 3 METODOLOGIA

## 3.1 IMPLEMENTAÇÃO

A representação do cubo em Python é a seguinte: existe uma lista chamada pos (posição) e seus elementos são as faces do cubo. As faces do cubo são matrizes 3x3 em que cada elemento é um número inteiro de 0 a 5.

As faces do cubo na lista pos seguem a seguinte ordem: esquerda, frente, direita, trás, baixo, cima. Em notação: L,F,R,B,D,U.

Dessa forma, como o referencial estabelecido foi face frontal vermelha e face superior amarela, temos que cada número é associado a uma cor.

0: AZUL

1: VERMELHO

2: VERDE

3: LARANJA

4: BRANCO

5: AMARELO

Assim, cada adesivo do cubo tem uma representação do tipo pos[face][linha][coluna] e carrega um número de 0 a 5 informando a cor do adesivo.