FACULTAD DE
INGENIERÍA
UNIVERSIDAD DE SANTIAGO DE CHILE

TAREA 1: ALGORITMO A*

TALLER DE PROGRAMACIÓN

Sección: 13315-0-A-1

Profesor: Pablo Roman Asenjo

Ayudante: Christopher Torres Aceituno

Autor: Cristóbal Marchant Osorio

1. Descripción del problema

El Cubo Rubik es un rompecabezas mecánico el cual consiste en un cubo tridimensional en donde sus 6 caras poseen un color distinto cada una. Sobre este cubo se podrán realizar movimientos en cada cara de este cubo, de esta manera se podrán combinar colores que existen en cada cara y de esta manera encontrar una solución.

Se debe construir un programa codificado en C++, el cual tenga la implementación del algoritmo A*, el cual pueda encontrar la manera más corta de resolver el Cubo Rubik, calculando adicionalmente el tiempo en que se demora en encontrar dicha solución y utilizando estructuras de datos auxiliares más una heurística que permita acelerar el algoritmo.

2. Solución propuesta (Estrategia a utilizar)

Tener en consideración que para la resolución de este problema de momento se necesitaran 6 clases importantes:
- Clase Rubik: Clase en donde se encontrar nuestro cubo y las funciones de giro.
- Clase Rubik Solver: En donde estará la resolución de Cubo Aplicando el algoritmo A*.
- Clase Idea: Representación de todos los estados que ha adquirido nuestro Cubo, básicamente es un heap, al que llame idea.
- Clase State: Clase que representa el estado del cubo y el previo.
- Clase Path: Clase que almacena las operaciones.
-Clase Rotate: Clase que implementa la rotación de las caras.
Para la solución planeada dispone de 6 funciones (1 por cada cara) que nos permiten girar una cara junto la primera fila que posea como vecino esta cara (Figura 1), de esta manera como se puede ver en la figura tendremos un total de 12 movimientos, D: Inferior; U: Superior; R: Derecha; L: Izquierda; B: Trasera; F: Frontal. En donde el sentido de estos se representa con un + o un -.

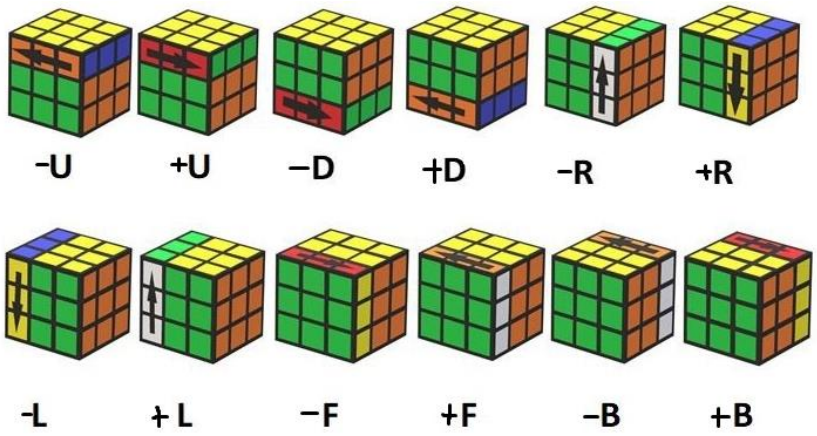

Figura 1: Giro de las caras con su respectiva perspectiva.

La Solución se basa en que al tener un Cubo desordenado (previamente generado) como estado inicial, a la cual se le aplicará las funciones, guardando el estado de las caras (State), de este modo ver que no se repita esa forma (no se repitan combinaciones) y no se genere un bucle, trabajando junto a “Path” que será el encargado de almacenar las operaciones realizadas por el algoritmo A*.

La heurística aplicada será dividida en pasos para luego ser sumada según la condición del cubo, esto quiere decir que cuando este buscando la cara blanca esta estará en condición 0, la cual solo podrá usar una parte de la heurística total, cuando la encuentre pasará a condición 1, la cual calculará la heurística usando la parte 1 y 2 de esta, de esta manera hasta completar el cubo.
Cada parte de la heurística consiste en que, si en cierta posición la pieza no es la correcta, aumenta en cierto tamaño la heurística, ya que esto equivaldría a la cantidad de movimientos para que alcance la posición esperada.

3. Ejecución del código

Para la ejecución del código, es necesario tener instalado g++ y make, los cuales utilizaremos para compilar nuestro código, posteriormente nos situamos en la carpeta del código y abriremos una terminal en la cual escribiremos “make” para que compile todo nuestro código, posteriormente procederemos a escribir “./namefile” para abrir algún programa ya compilado.
Para esto tener un main el cual deberá ser ejecutado, y para probar el funcionamiento de las 6 clases mencionadas anteriormente, será por “test_name”.

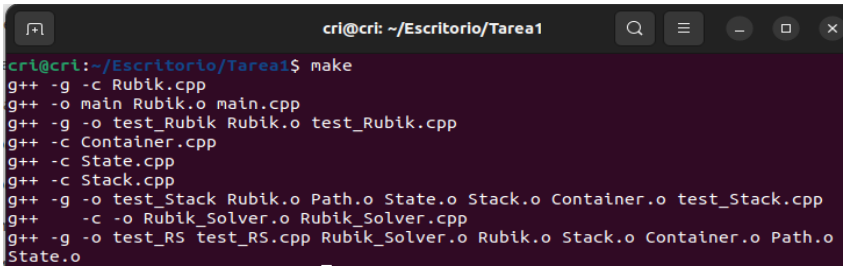

Figura 2: Terminal make

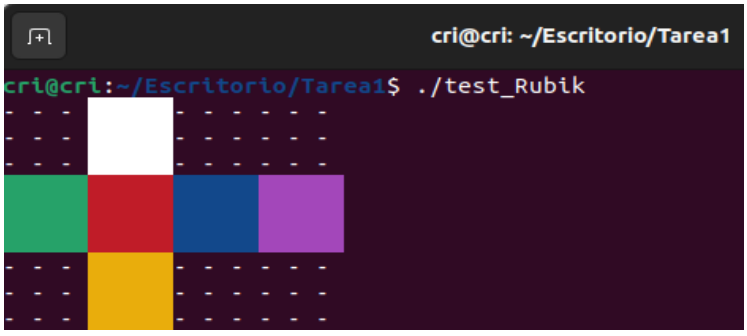

Figura 3: test_Rubik (con visualización matricial del cubo)

4. conclusión

En conclusión, se cumplió con los objetivos de realizar todas las clases importantes e implementar una heurística adecuada al tema (por pasos), pero para números grandes es probable que se demore mucho en encontrar una solución y esta probablemente no sea la más eficiente, debido el planteamiento de la heurística. Cabe mencionar que el desarrollo de este proyecto se trabajo como pionero del lenguaje C++, en donde se ha aprendido bastante y se propone seguir trabajando con el mismo entusiasmo como en este proyecto.