SISTEMAS INTELIGENTES: CUBO DE RUBIK. EQUIPO EC1-8.

Integrantes del equipo:

Ángel Villaseñor Gómez

Álvaro Rojas Parra

Manuel Morales Rodríguez

Definición del problema e implementación de la frontera

- Algoritmo básico de búsqueda
- Resultados

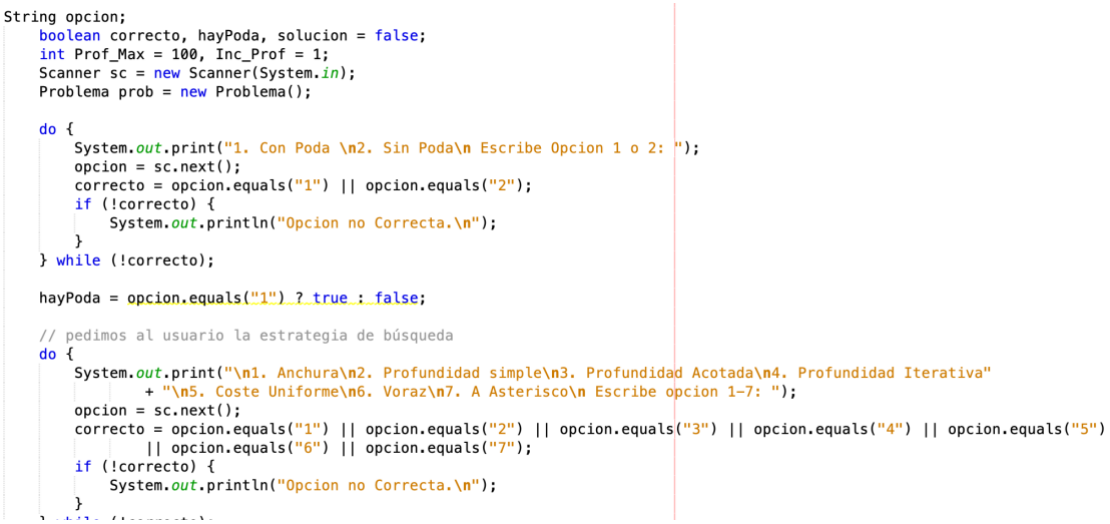
El proyecto va a ser almacenado en un repositorio de GitHub, llamado EC1-8, el cual puede ser encontrado en el siguiente enlace:

https://github.com/angelvillago15/EC1-8

1. Algoritmo básico de búsqueda

Hemos creado un menú inicial en el que primero se elige si se quiere el algoritmo con poda o sin ella. Después se elige el algoritmo de búsqueda y en función del algoritmo elegido se pide una información de partida como por ejemplo la profundidad máxima, el incremento de la profundidad iterativa, etc. Además hemos implementado una serie de controles para el menú y asegurarnos de que se utilice correctamente.

Se ha implementado los algoritmos de búsqueda para anchura, todos los tipos de profundidad y coste uniforme. Adicionalmente hemos añadido Voraz y A*.



```
String opcion;
    boolean correcto, hayPoda, solucion = false;
    int Prof_Max = 100, Inc_Prof = 1;
    Scanner sc = new Scanner(System.in);
    Problema prob = new Problema();

    do {
        System.out.print("1. Con Poda \n2. Sin Poda\n Escribe Opcion 1 o 2: ");
        opcion = sc.next();
        correcto = opcion.equals("1") || opcion.equals("2");
        if (!correcto) {
            System.out.println("Opcion no Correcta.\n");
        }
    } while (!correcto);

    hayPoda = opcion.equals("1") ? true : false;

    // pedimos al usuario la estrategia de búsqueda
    do {
        System.out.print("\n1. Anchura\n2. Profundidad simple\n3. Profundidad Acotada\n4. Profundidad Iterativa"
                + "\n5. Coste Uniforme\n6. Voraz\n7. A Asterisco\n Escribe opcion 1-7: ");
        opcion = sc.next();
        correcto = opcion.equals("1") || opcion.equals("2") || opcion.equals("3") || opcion.equals("4") || opcion.equals("5")
                || opcion.equals("6") || opcion.equals("7");
        if (!correcto) {
            System.out.println("Opcion no Correcta.\n");
        }
```

2. Información de partida

Todos los algoritmos de búsqueda menos el de profundidad iterativa han sido implementados por un único método llamado “Busqueda_Acotada” al cual, en función de la opción elegida y la información proporcionada en el menú el método se ejecuta para cada tipo de algoritmo.

El algoritmo de profundidad iterativa ha sido implementado por un método exclusivo para él llamado “Busqueda” ya que este algoritmo de búsqueda tiene un parámetro adicional llamado “Inc_Prof” el cual hace referencia al incremento de la profundidad iterativa y es exclusivo de este algoritmo de busqueda

```
switch (opcion) {
    case "1":
        solucion = Busqueda_Acotada(prob, "Anchura", Prof_Max, hayPoda); //anchura
        break;
    case "2":
        solucion = Busqueda_Acotada(prob, "ProfundidadSimple", 10000, hayPoda); //Profundidad simple, utilizada prof infinita
        break;
    case "3":
        solucion = Busqueda_Acotada(prob, "ProfundidadAcotada", Prof_Max, hayPoda); // Profundidad acotada
        break;
    case "4":
        solucion = Busqueda(prob, "ProfundidadIterativa", Prof_Max, Inc_Prof, hayPoda); // Profundidad iterativa
        break;
    case "5":
        solucion = Busqueda_Acotada(prob, "CosteUniforme", Prof_Max, hayPoda); // Coste uniforme
        break;
    case "6":
        solucion = Busqueda_Acotada(prob, "Voraz", Prof_Max, hayPoda); // Voraz
        break;
    case "7":
        solucion = Busqueda_Acotada(prob, "A", Prof_Max, hayPoda); // A asterisco
        break;

}
```

Algoritmo básico de búsqueda en la clase problema

La clase problema tiene una instancia del espacio de estados y un estado inicial obtenido del espacio de estados proporcionado anteriormente

```
public class Problema {
    private EspacioDeEstados espacioDeEstados;
    private Estado estadoInicial;

    public Problema() throws IOException, FileNotFoundException, ParseException {
        this.espacioDeEstados = new EspacioDeEstados("cube.json");
        this.estadoInicial = espacioDeEstados.getEstado();
    }
```