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

- Espacio de estados
- Problema
- Árbol de búsqueda

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. Espacio de estados

Hemos utilizado los métodos indicados de acciones y estados

```
public class EspacioDeEstados {

    public EspacioDeEstados(){}

    public List<Sucesor> getSucesores(Estado estado){
        List<Sucesor> sucesores = new ArrayList();
        List<Accion> acciones = new ArrayList(estado.getAcciones());
        for(Accion accion : acciones){
            sucesores.add(new Sucesor (accion,estado.getEstado(accion),1));
        }
        return sucesores;
    }

}
```

2. Frontera

Para crear la frontera hemos creado los métodos necesarios para insertar y eliminar nodos, a la vez que también vemos cuando esta vacía

```
public class Frontera {

    private PriorityQueue<NodoArbol> colaNodos = new PriorityQueue<NodoArbol>();

    public void CrearFrontera (){
        colaNodos.clear();
    }
    public void Insertar (NodoArbol n){
        colaNodos.add(n);
    }
    public void Eliminar (){
        colaNodos.remove();
    }
    public boolean EstaVacia(){
        return colaNodos.isEmpty();
    }

}
```

3. Estado

Para esta clase estado hemos implementado varios métodos, el primero para duplicar todas las caras del cubo, incluyendo un mensaje de error en caso de que falle.

Otro método para leer el fichero JSON, que va añadiendo poco a poco cada cara del cubo.

Despues una llamada getID, con todas las posiciones para poder identificarlas, “getMD5”

A continuación implementamos un método para cada movimiento que queramos hacer: moveL, movel, moveD, moved, moveB, moveb, rotarMatrizIzq, rotarMatrizDer. Y otras tantas para conseguir el estado, el objetivo y las acciones.

Al final hemos incluido otra para poder escribir todo lo anterior: toString()

4. NodoArbol

Con esta clase queremos acceder a todos los nodos del Árbol que genera el padre con toda la información que nos pide el dominio

```
public NodoArbol(NodoArbol padre, Estado estado, double coste, Accion accion, int p, double f) {
    this.padre = padre;
    this.estado = estado;
    this.coste = coste;
    this.accion = accion;
    this.p = p;
    this.f = f;
}
```