Formulation : Missioner's and Cannibals

States

3 Missioners and 3 cannibals a boat and two sides of a river

Initial state

The 3 missioners, cannibals and the boat in one side of the river

Goal state

The 3 missioners, cannibals and the boat in the other side of the river

Succession function

Two persons get in the boat to cross the river and one of them come back to repeat the process until everyone can cross to the other side.

Formulation: Rubik Cube

States

Permutations of 6 sides in pieces of 3x3

Initial state

Disorder of the colors in each of the sides

Goal state

Each one of the sides it’s complete from a uniform color

Succession function

Turn the axis of the cube N quantity of times to different sides until we get the goal state, the possible movements are Left, Right, Up and Down.

Formulation: Route Monterrey - CDMX

States

Different Ways or routes, car or bus (depending of the conditions)

Initial state

Somewhere in Monterrey N.L. (Depending of the conditions the exact place)

Goal state

Somewhere in CDMX (depending of the conditions the exact place)

Succession function

Take the ways or routes highways, etc that will make us accomplish the goal state.

```
public class
MisionerosY
Canibales {
                    public static String M = "M";   //se mueve un misionero
                public static String MM = "MM"; //se mueven dos misioneroos
                public static String C = "C";   //se mueve un canibal
                public static String CC = "CC"; //se mueven dos canibales
                public static String MC = "MC"; //se mueve un misionero y un canibal

                private String ultMov; //guarda el ultimo movimiento, para no generar
            ciclos

                //Numero maximo de personas que puede trasladar
                private final int capacidadBarca = 2;
                public int getCapacidadBarca() {
                    return capacidadBarca;
                }



                // [misioneros,canibales,barca] representa la orilla izquierda
                private int misioneros;
                private int canibales;
                private int barca; //1 orilla izquierda, 0 orilla derecha

                    public MisionerosYCanibales(){
                           //3 misioneros, 3 canibales y la barca en la orilla
            izquierda
                    misioneros = 3;
                    canibales = 2;
                    barca = 1;
                    ultMov = " ";
                   }
                   public MisionerosYCanibales(MisionerosYCanibales estado){
                    ultMov = estado.ultMov;
                    misioneros = estado.misioneros;
                    canibales = estado.canibales;
                    barca = estado.barca;
                }

                    public int getMisioneros() {
                    return misioneros;
                }
```

```
    public int getCanibales() {
        return canibales;
    }


    public int getBarca() {
        return barca;
    }

    public int[] dameEstado(){
        int[] est = new int[3];
        est[0] = getMisioneros();
        est[1] = getCanibales();
        est[2] = getBarca();
        return est;
    }

    public String imprimirEstado() {
       int[] estado= dameEstado();
       return
"Misioneros:"+Math.abs(estado[0])+",Canibales:"+Math.abs(estado[1])+",B
ote:"+Math.abs(estado[2])+"____"+"Misioneros:"+(3-
estado[0])+",Canibales:"+(2-estado[1])+",Bote:"+(1-estado[2]);
       }
  //OPERADORES
    public void mueveM(){
        if (getBarca() == 1){
            barca = 0;
            misioneros--;
        }
        else{
            barca = 1;
            misioneros++;
        }
        ultMov = M;
    }

    public void mueveMM(){
        if (getBarca() == 1){
            barca = 0;
            misioneros = getMisioneros() - 2;
        }
```

```
        else{
            barca = 1;
            misioneros = getMisioneros() + 2;
        }
        ultMov = MM;
    }

    public void mueveC(){
        if (getBarca() == 1){
            barca = 0;
            canibales--;
        }
        else{
            barca = 1;
            canibales++;
        }
        ultMov = C;
    }

    public void mueveCC(){
        if (getBarca() == 1){
            barca = 0;
            canibales = getCanibales() - 2;
        }
        else{
            barca = 1;
            canibales = getCanibales() + 2;
        }
        ultMov = CC;
    }

    public void mueveMC(){
        if (getBarca() == 1){
            barca = 0;
            misioneros--;
            canibales--;
        }else{
            barca = 1;
            misioneros++;
            canibales++;
        }
        ultMov = MC;
    }
```

```
    public boolean puedeMover(String mov){
        if (ultMov.equals(mov)) return false; //evito repetir ultimo
movimiento
        if (mov.equals(M)){
            if (getBarca() == 1)
                return getMisioneros() >= 1 &&
                       !estadoPeligroso(misioneros-1,canibales);
            else
                return 3-getMisioneros() >= 1 &&
                       !estadoPeligroso(misioneros+1,canibales);
        }
        if (mov.equals(MM)){
            if (getBarca() == 1)
                return getMisioneros() >= 2 &&
                        !estadoPeligroso(misioneros-2,canibales);
            else
                return 3-getMisioneros() >= 2 &&
                        !estadoPeligroso(misioneros+2,canibales);
        }
        if (mov.equals(C)){
            if (getBarca() == 1)
                return getCanibales() >= 1 &&
                        !estadoPeligroso(misioneros,canibales-1);
            else
                return 3-getCanibales() >= 1 &&
                        !estadoPeligroso(misioneros,canibales+1);
        }
        if (mov.equals(CC)){
            if (getBarca() == 1)
                return getCanibales() >= 2 &&
                        !estadoPeligroso(misioneros,canibales-2);
            else
                return 3-getCanibales() >= 2 &&
                        !estadoPeligroso(misioneros,canibales+2);
        }
        if (mov.equals(MC)){
            if (getBarca() == 1)
                return getMisioneros() >= 1 && getCanibales()  >= 1 &&
                        !estadoPeligroso(misioneros-1,canibales-1);
            else
                return 3-getMisioneros() >= 1 && 3-getCanibales() >= 1
&&
```

```
                        !estadoPeligroso(misioneros+1,canibales+1);

            }
            return false;
        }

        private boolean estadoPeligroso(int m,int c){
            return (m < c && m != 0) || (m > c && m != 3);
        }

        public int heuristica() {
              int hVal = getMisioneros() +getCanibales();
              return hVal;
         }

           public static void main(String[] args){
                 MisionerosYCanibales        algoritmo        =        new
MisionerosYCanibales();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveCC();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveC();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveCC();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveC();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveMM();
                 System.out.println(algoritmo.imprimirEstado());
                 System.out.println("Numero      de      personas      sin
trasladar:"+algoritmo.heuristica());
                 algoritmo.mueveMC();
                 System.out.println(algoritmo.imprimirEstado());
```

```
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
                algoritmo.mueveMM();
                System.out.println(algoritmo.imprimirEstado());
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
                algoritmo.mueveC();
                System.out.println(algoritmo.imprimirEstado());
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
                algoritmo.mueveCC();
                System.out.println(algoritmo.imprimirEstado());
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
                algoritmo.mueveC();
                System.out.println(algoritmo.imprimirEstado());
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
                algoritmo.mueveCC();
                System.out.println(algoritmo.imprimirEstado());
                System.out.println("Numero de personas sin
trasladar:"+algoritmo.heuristica());
        }



}
```

```
import java.util.Scanner;

public class InfoCubo {
	public static void main (String[] args) {

		Scanner escáner = new Scanner(System.in);

		System.out.println("Bienvenido a InfoCubo, aquí te proporcionaremos informcaión sobre las piezas del puzle que usted quiera.\n");
		System.out.println("¿Su puzle es un cubo o un cuboide?\nSi es un cubo escriba \"cubo\" y si es un cuboide \"cuboide\".");
		boolean again, again1; char seguir;

		do {
			again1 = false;
			String c = escáner.next();

			if (c.equals("cubo")){

				System.out.println("\nIntroduzca el número de piezas por fila que tiene su cubo.");
				int p = escáner.nextInt();

				do {
					again = false;
					if (p > 1) {

						System.out.println("Cubo: " + p + "x" + p + "x" + p + "\n\nPIEZAS\nNúmero de piezas en total: " + (p*p*p - (p-2)*(p-2)*(p-2)) );
						System.out.println("Número de esquinas: 8");
						System.out.println("Número de aristas: " + 12*(p-2));
						System.out.println("Número de centros: " + 6*(p-2)*(p-2));
```

```
                int total = 0;
                int centros = 0;

                for (int i = 1; i <= (p+1)/2; i++) {
                        total = total + i;
                }

                for (int i = 1; i <= (p-1)/2; i++) {
                        centros = centros + i;
                }

                System.out.println("\nTIPOS DE PIEZAS\nNúmero de tipos de piezas en total: " + total);
                System.out.println("Número de tipos de esquinas: 1");
                System.out.println("Número de tipos de aristas: " + (((p+1)/2)-1));
                System.out.println("Número de tipos de centros: " + centros);

            } else{
                if (p == 1) {
                        System.out.println("Un 1x1x1 solo tiene una pieza, que al carecer de forma de esquina, centro o arista, se podría calificar como núcleo.");
                } else{

        System.out.println("Número carente de sentido en este contexto, prueba de nuevo");
                        p = escáner.nextInt();
                        again = true;
                }
            }
```

```
                                        boolean inválido = false;
                                    System.out.println("\n\n¿Quieres    probar
con otro puzle? (Podrás elegir el tipo)\nSi la respuesta es afirmativa escriba 's'
o 'S'; si es negativa 'n' o 'N'");
                                        do {
                                                seguir                          =
escáner.next().charAt(0);
                                                if (seguir == 's' || seguir ==
'S'){
                                                        again1 = true;

        System.out.println("\n\n¿Su puzle es un cubo o un cuboide?\nSi es un cubo
escriba \"cubo\" y si es un cuboide \"cuboide\".");
                                                } else if (!(seguir == 'n' ||
seguir == 'N')){

        System.out.println("Comando inválido,  prueba de nuevo.");
                                                        inválido = true;
                                                }
                                        } while(inválido);

                                    }while(again);

                            } else if (c.equals("cuboide")){

                                    System.out.println("\nEscriba      las      tres
dimensiones del cuboide separando cada una con un intro. Cuando haya escrito las
tres, dé otro intro.");
                                    int[] ps = new int[3];
                                    for (int i = 0; i < 3; i++){
                                            ps[i] = escáner.nextInt();
                                    }
                                    System.out.println("Cuboide:  " + ps[0] + "x" +
ps[1] + "x" + ps[2]);
```

```
                                boolean np = false;
                                for (int i = 0; i < 3; i++){
                                        if (ps[i]==1) {
                                                np = true;
                                        }
                                }

                                System.out.print("\nPIEZAS\nNúmero  total  de
piezas: ");
                                if (np==false){
                                        System.out.println((ps[0]*ps[1]*ps[2]-
(ps[0]-2)*(ps[1]-2)*(ps[2]-2)) );
                                        System.out.println("Número de esquinas:
8");
                                } else{
                                        System.out.println(ps[0]*ps[1]*ps[2]);

                                        int núcleo = 0;
                                        int centros1 = 0;
                                        int centros2 = 0;
                                        int aristas1 = 0;
                                        int aristas2 = 0;
                                        int esquinas = 0;
                                        int nd = 0;
                                        for (int i = 0; i < 3; i++) {
                                                if (ps[i] == 1){
                                                        nd++;
                                                }
                                        }
                                        switch (nd){

                                        case 1:
                                                núcleo  =  -((ps[0]-2)*(ps[1]-
2)*(ps[2]-2));
                                                aristas2 = 4;
```

```
                                        centros2                        =
2*(ps[0]+ps[1]+ps[2]-5);

                                        esquinas = 4;
                                        aristas1 = centros2;
                                        centros1 = núcleo;
                                        break;

                                case 2:
                                        núcleo    =    (ps[0]-2)*(ps[1]-
2)*(ps[2]-2);
                                        centros2 = 2;
                                        aristas2 = 0;

                                        esquinas = 2;
                                        for (int i = 0; i < 3; i++){
                                                if (ps[i]/2*2==ps[i]){
                                                        centros1 = 0;
                                                        break;
                                                } else{
                                                        centros1 = 1;
                                                }
                                        }
                                        aristas1  =  (ps[0]*ps[1]*ps[2]-
2)/2*2;
                                        break;

                                default://1x1x1
                                        núcleo = 1;
                                        aristas2 = 0;
                                        centros2 = 0;

                                        centros1 = 1;
                                        aristas1 = 0;
                                        esquinas = 0;
                                        break;
```

```
                                        }

                                        System.out.println("Los criterios de calificación de piezas son ambiguos en este tipo de cuboides."
                                                + "\nLea la información que le corresponde a este tema en la descripción del programa marcada con un asterisco (*) para responder a la siguiente pregunta.");
                                        System.out.println("¿Seguirá el primer criterio (marque 1), el segundo (marque 2), o quiere que se muestren ambos (marque 3)?");
                                        int criterio = escáner.nextInt();
                                        switch (criterio){
                                        case 1:
                                                System.out.println("Número de esquinas: " + esquinas);
                                                System.out.println("Número de aristas: " + aristas1);
                                                System.out.println("Número de centros: " + centros1);
                                                break;

                                        case 2:
                                                System.out.println("Número de esquinas: 0");
                                                System.out.println("Número de aristas: " + aristas2);
                                                System.out.println("Número de centros: " + centros2);
                                                System.out.println("Número de núcleos: " + núcleo);
                                                break;

                                        case 3:
                                                System.out.println("Criterio 1 (físico):");
```

```
                                                System.out.println("Número   de
esquinas: " + esquinas);
                                                System.out.println("Número   de
aristas: " + aristas1);
                                                System.out.println("Número   de
centros: " + centros1);

                                                System.out.println("\nCriterio
2 (comportamiento):");
                                                System.out.println("Número   de
esquinas: 0");
                                                System.out.println("Número   de
aristas: " + aristas2);
                                                System.out.println("Número   de
centros: " + centros2);
                                                System.out.println("Número   de
núcleos: " + núcleo);
                                                break;

                                        }
                                }

                                System.out.println("\n\n¿Quieres probar con otro
puzle? (Podrás elegir el tipo)\nSi la respuesta es afirmativa escriba 's' o 'S';
si es negativa 'n' o 'N'");
                                seguir = escáner.next().charAt(0);
                                if (seguir == 's' || seguir == 'S'){
                                        again1 = true;
                                        System.out.println("\n\n¿Su puzle es un
cubo o un cuboide?\nSi es un cubo escriba \"cubo\" y si es un cuboide \"cuboide\".");
                                } else if (!(seguir == 'n' || seguir == 'N')){
                                        System.out.println("Comando   inválido,
prueba de nuevo.");
                                }

                        } else{
```

```
                                    again1 = true;
                                    System.out.println("Comando inválido, prueba de
nuevo.");
                            }


                }while(again1);

                System.out.println("\nFin del programa.");
                escáner.close();
        }
}
```