Práctica Individual 1

Problema Número 1

Análisis y Diseño de Datos y Algoritmos
Grado de Ingeniería del Software
Curso 2º

Francisco Javier Huertas Vera (javiihv93@gmail.com)

Tutor: Fernando de la Rosa Troyano

Índice

1 Enunciado

Un mapa de un juego está modelado a través de una matriz de casillas, y cada casilla posee dos propiedades: tipo de casilla, e inclinación. Tipo de casilla es un tipo enumerado que puede ser ASFALTO, PIEDRA o BARRO, mientras que la inclinación es un valor entero positivo que describe la altura de dicha casilla. En dicha matriz de casillas, se desea realizar un recorrido. En dicho recorrido, el hecho de pasar de una casilla a otra, puede tener diferentes recompensas ó penalizaciones. Para ser más concretos, si la transición es de una casilla a otra del mismo tipo y las casillas tienen una inclinación positiva (es decir, la inclinación de la casilla origen es mayor que la casilla destino) entonces tendremos una recompensa de +1 en experiencia. Por otro lado, si la transición es entre casillas diferentes y la inclinación es positiva tendremos una penalización de -1 en experiencia. En cualquier otro caso, el hecho de pasar de una casilla a otra no tendrá ni recompensa ni penalización. Se quiere implementar la función calculaRecompensas, que dada una matriz de casillas como las descritas anteriormente permita calcular el total de recompensas/penalizaciones que se obtendría de recorrer dicha matriz en zig-zag , es decir, en primer lugar la primera fila de izquierda a derecha, tras esto la segunda fila de derecha a izquierda, tras esto la tercera fila de izquierda a derecha, etc. Veamos un ejemplo de matriz de casillas, es decir, la entrada del problema:

1 A	2 A	2 A	3 A	2 A	5 A	6 A	4 A	5 A	7 A	6 A	8 A	9 A
9 A	8 A	6 A	7 B	5 A	4 P	6 A	5 A	2 A	3 A	2 A	2 A	1 A
1 A	2 A	2 A	3 B	2 P	5 A	6 A	4 P	5 A	7 P	6 B	8 A	9 A

Ejemplo donde la función calculaRecompensas es un resultado de -1.
No se tendrá en cuenta el salto de una fila a otra en el cálculo de penalizaciones o recompensas, es decir, el salto de la casilla 13 de la primera fila a la primera casilla de la segunda fila, no produce ni recompensa ni penalización. Cada casilla tiene una inclinación y una sigla correspondiente al tipo de celda. En este caso las casillas 4 y 5 de la primera fila tienen una inclinación positiva pero son del mismo tipo por lo tanto producen una recompensa positiva, sin embargo las casillas 7 y 8 de la segunda fila tienen una inclinación positiva pero son de tipos diferentes con lo que produce una penalización.

1.1 Cuestiones a resolver

1. Defina la función recursiva no final calculaRecompensas.

2. Implemente una función recursiva no final empleado el lenguaje de programación Java.
3. Implemente una función recursiva final a partir de la definición anterior.
4. A partir de las funciones anteriores, diseñar e implementar un algoritmo iterativo siguiendo las plantillas vistas en clases.
5. Indique el tamaño del problema, T(n) y la complejidad del algoritmo.

1.2 Lenguaje de programación asignado

Lenguaje de programación: Java
Entorno de programación: Eclipse

2 Seguimiento

2.1 Control de seguimiento y tutorías

El alumno debe listar y detallar las visitas a tutoría:

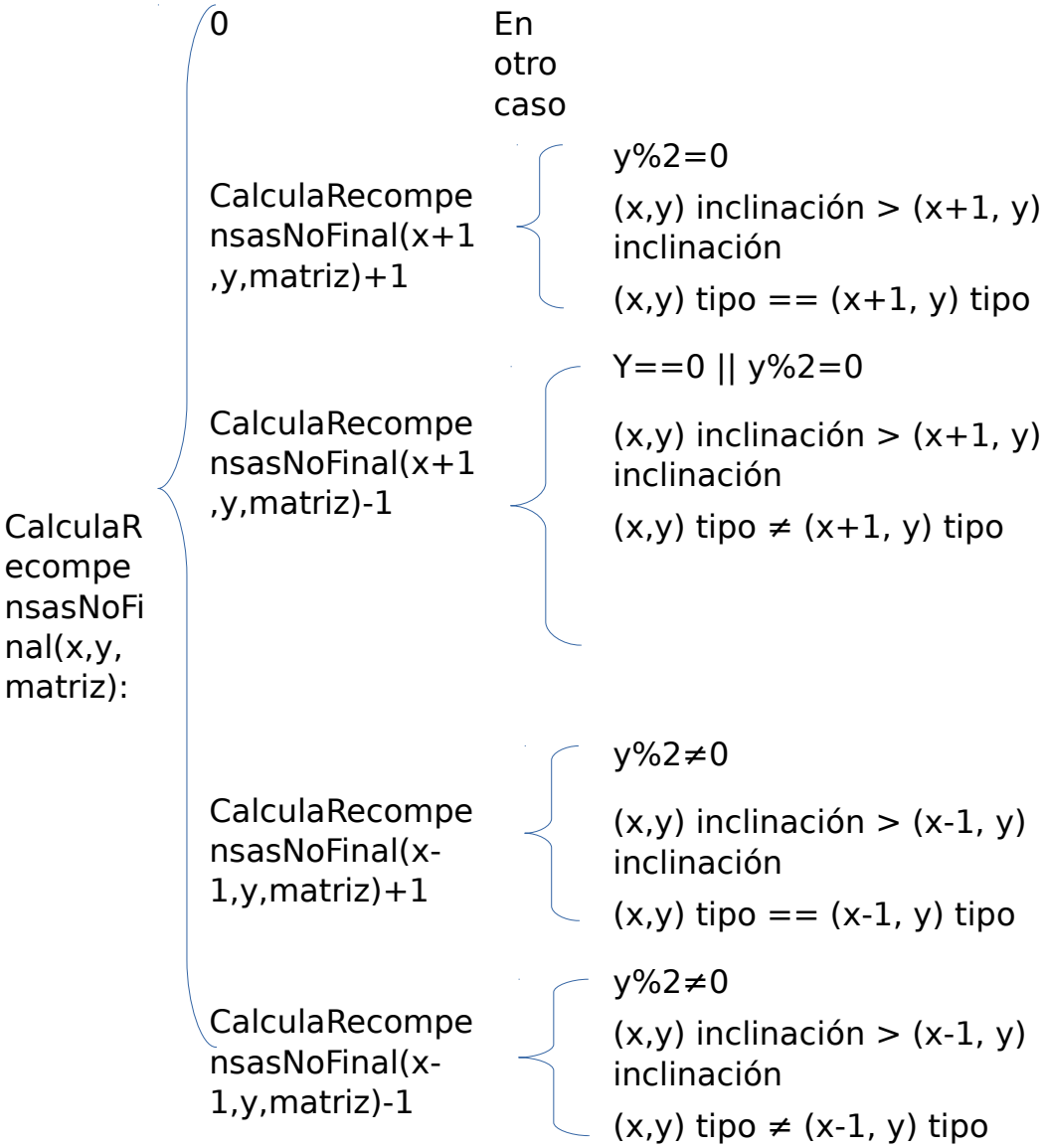
Fecha	Profesor	Detalles y dudas resueltas
11/09/16	Fernando de la Rosa Troyano	- Problemas con la traza.

2.2 Portafolio

Agenda de seguimiento en la que el alumno deberá registrar toda la actividad que realice para resolver el ejercicio, así como los problemas encontrados.

3 Implementación

1. Definición de la función recursiva no final:



2. Función recursiva no final:

```
public static Integer calculaRecompensasNoFinal(Integer x,
      Integer y, Casilla[][] matriz){

      Integer res = 0;

      if(y<matriz[x].length){
            if(y%2==0){
                  if(x==matriz.length-1){
res+=calculaRecompensasNoFinal(x,y+1,matriz);

                  }else if(matriz[x][y].getInclinacion()>
                        matriz[x+1][y].getInclinacion() &&
                        matriz[x][y].getTipo().equals(
                              matriz[x+1][y].getTipo())){
res+=calculaRecompensasNoFinal(x+1, y, matriz)+1;

                  }else if(matriz[x][y].getInclinacion()>
                        matriz[x+1][y].getInclinacion() &&
                        !(matriz[x][y].getTipo().equals(
                              matriz[x+1][y].getTipo()))){
res+=calculaRecompensasNoFinal(x+1, y, matriz)-1;

                  }else{
res+=calculaRecompensasNoFinal(x+1, y, matriz);

                  }
            }
            if(y%2!=0){
                  if(x==0){
res+=calculaRecompensasNoFinal(x,y+1,matriz);

                  }else if(matriz[x][y].getInclinacion()>
                        matriz[x-1][y].getInclinacion()
                        && matriz[x][y].getTipo().equals(
                              matriz[x-1][y].getTipo())){
res+=calculaRecompensasNoFinal(x-1, y, matriz)+1;

                  }else if(matriz[x][y].getInclinacion()>
                        matriz[x-1][y].getInclinacion()
                        && !(matriz[x][y].getTipo().equals(
                              matriz[x-1][y].getTipo()))){
res+=calculaRecompensasNoFinal(x-1, y, matriz)-1;
```

```
                        }else{
      res+=calculaRecompensasNoFinal(x-1, y, matriz);

                        }
                  }
            }
            return res;
      }
```

3. Función Recursiva Final:

```
 public static Integer calculaRecompensasFinal(Integer x, Integer y,
Casilla[][] matriz, Integer acum){
            Integer res = acum;
            if(y<matriz[x].length){
                  if(y%2==0){
                        if(x==matriz.length-1){
      res=calculaRecompensasFinal(x,y+1,matriz, acum);

                        }else if(matriz[x][y].getInclinacion()>
                        matriz[x+1][y].getInclinacion()
                        && matriz[x][y].getTipo().equals(
                              matriz[x+1][y].getTipo())){
      res=calculaRecompensasFinal(x+1, y, matriz, acum+1);

                        }else if(matriz[x][y].getInclinacion()>
                        matriz[x+1][y].getInclinacion()
                        && !(matriz[x][y].getTipo().equals(
                              matriz[x+1][y].getTipo()))){
      res=calculaRecompensasFinal(x+1, y, matriz,acum-1);

                        }else{
      res=calculaRecompensasFinal(x+1, y, matriz,acum);
                        }
                  }
                  if(y%2!=0){
                        if(x==0){
      res=calculaRecompensasFinal(x,y+1,matriz, acum);

                        }else if(matriz[x][y].getInclinacion()>
                        matriz[x-1][y].getInclinacion()
                        && matriz[x][y].getTipo().equals(
                              matriz[x-1][y].getTipo())){
      res=calculaRecompensasFinal(x-1, y, matriz, acum+1);

                        }else if(matriz[x][y].getInclinacion()>
                        matriz[x-1][y].getInclinacion()
                        && !(matriz[x][y].getTipo().equals(
                              matriz[x-1][y].getTipo()))){
      res=calculaRecompensasFinal(x-1, y, matriz, acum-1);

                        }else{
```

```
		res=calculaRecompensasFinal(x-1, y, matriz, acum);
                        }


                }
        }
        return res;
}
```

4. Función Iterativa:

```
public static Integer CalculaRecompensasIterativo(Casilla[][] matriz){
	Integer res=0;

	for(int i=0;i<matriz.length-1;i++){
		for(int j=0;j<matriz[i].length;j++){
			if(j==0 || j%2==0){
				int i2=i+1;
				if(i2==matriz.length){
					res+=0;
				}
				if(matriz[i][j].getInclinacion()>
					matriz[i+1][j].getInclinacion() &&
					matriz[i][j].getTipo().equals(
						matriz[i+1][j].getTipo())){
					res+=1;
				}
				if(matriz[i][j].getInclinacion()>
					matriz[i+1][j].getInclinacion() &&
					!(matriz[i][j].getTipo().equals(
						matriz[i+1][j].getTipo()))){
					res+=-1;
				}
			}

		}
	}
	for(int i=matriz.length-1;i>0;i--){
		for(int j=0;j<matriz[i].length;j++){
			if(j%2!=0){
				int i3=i-1;
				if(i3<0){
					res+=0;
				}
				if(matriz[i][j].getInclinacion()>
					matriz[i3][j].getInclinacion() &&
					matriz[i][j].getTipo().equals(
						matriz[i3][j].getTipo())){
					res+=1;
				}
				if(matriz[i][j].getInclinacion()>
```

```
                                matriz[i3][j].getInclinacion() &&
                                !(matriz[i][j].getTipo().equals(
                                      matriz[i3][j].getTipo()))){
                                res+=-1;
                          }
                    }
              }
        }
        return res;
}
```

4 Análisis y complejidad

Tamaño del problema, T(n) y complejidad:

-Tamaño del problema: El tamaño del problema viene dado por el tamaño de la matriz, ya que tenemos que recorrer la matriz celda a celda. El tamaño sería:

n = j x i

i: nº de columnas de la matriz.

j: nº de filas de la matriz.

-T(n): Complejidad del algoritmo iterativo:

$T(n) = T(n-1)+\theta(n^2)$

a=1
b=1 como c=1 y a=1: la complejidad es de: $\theta(n^3)$
c=1
d=2

5 Pruebas y traza

- Adjuntar los test de prueba realizados para comprobar el funcionamiento de los algoritmos desarrollados en cada apartado indicando distintos ejemplos de uso.
- Adjuntar trazas de los algoritmos desarrollados. La **traza** de un algoritmo indica la secuencia de acciones (instrucciones) de su ejecución, así como, el valor de las variables del algoritmo (o programa) después de cada acción (instrucción).

```
public class Test {

      public static void main(String[] args) {
            // TODO Auto-generated method stub
            Casilla matriz[][] = new Casilla[13][3];
            matriz[0][0] = new CasillaImpl(TipoCasilla.ASFALTO,1);
            matriz[1][0] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[2][0] = new CasillaImpl(TipoCasilla.ASFALTO,2);
```

```
            matriz[3][0] = new CasillaImpl(TipoCasilla.ASFALTO,3);
            matriz[4][0] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[5][0] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[6][0] = new CasillaImpl(TipoCasilla.ASFALTO,6);
            matriz[7][0] = new CasillaImpl(TipoCasilla.ASFALTO,4);
            matriz[8][0] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[9][0] = new CasillaImpl(TipoCasilla.ASFALTO,7);
            matriz[10][0] = new CasillaImpl(TipoCasilla.ASFALTO,6);
            matriz[11][0] = new CasillaImpl(TipoCasilla.ASFALTO,8);
            matriz[12][0] = new CasillaImpl(TipoCasilla.ASFALTO,9);

            matriz[0][1] = new CasillaImpl(TipoCasilla.ASFALTO,9);
            matriz[1][1] = new CasillaImpl(TipoCasilla.ASFALTO,8);
            matriz[2][1] = new CasillaImpl(TipoCasilla.ASFALTO,6);
            matriz[3][1] = new CasillaImpl(TipoCasilla.BARRO,7);
            matriz[4][1] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[5][1] = new CasillaImpl(TipoCasilla.PIEDRA,4);
            matriz[6][1] = new CasillaImpl(TipoCasilla.ASFALTO,6);
            matriz[7][1] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[8][1] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[9][1] = new CasillaImpl(TipoCasilla.ASFALTO,3);
            matriz[10][1] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[11][1] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[12][1] = new CasillaImpl(TipoCasilla.ASFALTO,1);

            matriz[0][2] = new CasillaImpl(TipoCasilla.ASFALTO,1);
            matriz[1][2] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[2][2] = new CasillaImpl(TipoCasilla.ASFALTO,2);
            matriz[3][2] = new CasillaImpl(TipoCasilla.BARRO,3);
            matriz[4][2] = new CasillaImpl(TipoCasilla.PIEDRA,2);
            matriz[5][2] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[6][2] = new CasillaImpl(TipoCasilla.ASFALTO,6);
            matriz[7][2] = new CasillaImpl(TipoCasilla.PIEDRA,4);
            matriz[8][2] = new CasillaImpl(TipoCasilla.ASFALTO,5);
            matriz[9][2] = new CasillaImpl(TipoCasilla.PIEDRA,7);
            matriz[10][2] = new CasillaImpl(TipoCasilla.BARRO,6);
            matriz[11][2] = new CasillaImpl(TipoCasilla.ASFALTO,8);
            matriz[12][2] = new CasillaImpl(TipoCasilla.ASFALTO,9);


            Integer recompensasNoFinal =
Utilidad.CalculaRecompensasNoFinal(0,0,matriz);
            System.out.println("Recompensas No Final:
"+recompensasNoFinal);
            Integer recompensasFinal =
Utilidad.CalculaRecompensasFinal(0,0,matriz,0);
            System.out.println("Recompensas Final:
"+recompensasFinal);
            Integer recompensasIterativo =
Utilidad.CalculaRecompensasIterativo(matriz);
            System.out.println("Recompensas Iterativo:
"+recompensasIterativo);
}
```

El resultado del Test sería:

Trazas:

En las siguientes tablas se muestra cada iteración del algoritmo y el valor de cada variable. En la clase Trazas están los métodos No Final y Final modificados para poder ver el valor de cada variable en cada iteración. Faltaría la iteración nº13, que equivale las variables a x=12 e y=0, en este caso no se calcula recompensa ya que es el final de la fila y lo que hay que hacer es incrementar la y para pasar a la siguiente fila, también falta la iteración nº26, en la que las variables serían x=0 e y=1 que sería la primera casilla de la fila 1 y no hay que calcular recompensa si no que incrementar la y.

Traza Algoritmo recursivo No Final:

Iteración	Variables
Iteracion número: 1	Variable res: -1 Variable x: 0 Variable y: 0
Iteracion número: 2	Variable res: -1 Variable x: 1 Variable y: 0
Iteracion número: 3	Variable res: -1 Variable x: 2 Variable y: 0
Iteracion número: 4	Variable res: -1 Variable x: 3 Variable y: 0
Iteracion número: 5	Variable res: -2 Variable x: 4 Variable y: 0
Iteracion número: 6	Variable res: -2 Variable x: 5 Variable y: 0
Iteracion número: 7	Variable res: -2 Variable x: 6

	Variable y: 0
Iteracion número: 8	Variable res: -3 Variable x: 7 Variable y: 0
Iteracion número: 9	Variable res: -3 Variable x: 8 Variable y: 0
Iteracion número: 10	Variable res: -3 Variable x: 9 Variable y: 0
Iteracion número: 11	Variable res: -4 Variable x: 10 Variable y: 0
Iteracion número: 12	Variable res: -4 Variable x: 11 Variable y: 0
Iteracion número: 14	Variable res: -4 Variable x: 12 Variable y: 1
Iteracion número: 15	Variable res: -4 Variable x: 11 Variable y: 1
Iteracion número: 16	Variable res: -4 Variable x: 10 Variable y: 1
Iteracion número: 17	Variable res: -4 Variable x: 9 Variable y: 1
Iteracion número: 18	Variable res: -5 Variable x: 8 Variable y: 1
Iteracion número: 19	Variable res: -5 Variable x: 7 Variable y: 1
Iteracion número: 20	Variable res: -5 Variable x: 6 Variable y: 1
Iteracion número: 21	Variable res: -4 Variable x: 5 Variable y: 1
Iteracion número: 22	Variable res: -4 Variable x: 4 Variable y: 1
Iteracion número: 23	Variable res: -4 Variable x: 3 Variable y: 1
Iteracion número: 24	Variable res: -3 Variable x: 2 Variable y: 1

Iteracion número: 25	Variable res: -3 Variable x: 1 Variable y: 1
Iteracion número: 27	Variable res: -3 Variable x: 0 Variable y: 2
Iteracion número: 28	Variable res: -3 Variable x: 1 Variable y: 2
Iteracion número: 29	Variable res: -3 Variable x: 2 Variable y: 2
Iteracion número: 30	Variable res: -3 Variable x: 3 Variable y: 2
Iteracion número: 31	Variable res: -2 Variable x: 4 Variable y: 2
Iteracion número: 32	Variable res: -2 Variable x: 5 Variable y: 2
Iteracion número: 33	Variable res: -2 Variable x: 6 Variable y: 2
Iteracion número: 34	Variable res: -1 Variable x: 7 Variable y: 2
Iteracion número: 35	Variable res: -1 Variable x: 8 Variable y: 2
Iteracion número: 36	Variable res: -1 Variable x: 9 Variable y: 2
Iteracion número: 37	Variable res: 0 Variable x: 10 Variable y: 2
Iteracion número: 38	Variable res: 0 Variable x: 11 Variable y: 2

Traza Algoritmo recursivo Final:

Para el la solución de la traza del algoritmo recursivo Final he hecho lo mismo que para el No Final. El resultado en la siguiente tabla.

Iteraciones	Variables
Iteración número: 1	Variable res: -1 Variable x: 0

	Variable y: 0 Variable acum: 0
Iteración número: 2	Variable res: -1 Variable x: 1 Variable y: 0 Variable acum: 0
Iteración número: 3	Variable res: -1 Variable x: 2 Variable y: 0 Variable acum: 0
Iteración número: 4	Variable res: -1 Variable x: 3 Variable y: 0 Variable acum: 0
Iteración número: 5	Variable res: -1 Variable x: 4 Variable y: 0 Variable acum: 1
Iteración número: 6	Variable res: -1 Variable x: 5 Variable y: 0 Variable acum: 1
Iteración número: 7	Variable res: -1 Variable x: 6 Variable y: 0 Variable acum: 1
Iteración número: 8	Variable res: -1 Variable x: 7 Variable y: 0 Variable acum: 2
Iteración número: 9	Variable res: -1 Variable x: 8 Variable y: 0 Variable acum: 2
Iteración número: 10	Variable res: -1 Variable x: 9 Variable y: 0 Variable acum: 2
Iteración número: 11	Variable res: -1 Variable x: 10 Variable y: 0 Variable acum: 3
Iteración número: 12	Variable res: -1 Variable x: 11 Variable y: 0 Variable acum: 3
Iteración número: 14	Variable res: -1 Variable x: 12 Variable y: 1 Variable acum: 3

Iteración número: 15	Variable res: -1 Variable x: 11 Variable y: 1 Variable acum: 3
Iteración número: 16	Variable res: -1 Variable x: 10 Variable y: 1 Variable acum: 3
Iteración número: 17	Variable res: -1 Variable x: 9 Variable y: 1 Variable acum: 3
Iteración número: 18	Variable res: -1 Variable x: 8 Variable y: 1 Variable acum: 4
Iteración número: 19	Variable res: -1 Variable x: 7 Variable y: 1 Variable acum: 4
Iteración número: 20	Variable res: -1 Variable x: 6 Variable y: 1 Variable acum: 4
Iteración número: 21	Variable res: -1 Variable x: 5 Variable y: 1 Variable acum: 3
Iteración número: 22	Variable res: -1 Variable x: 4 Variable y: 1 Variable acum: 3
Iteración número: 23	Variable res: -1 Variable x: 3 Variable y: 1 Variable acum: 3
Iteración número: 24	Variable res: -1 Variable x: 2 Variable y: 1 Variable acum: 2
Iteración número: 25	Variable res: -1 Variable x: 1 Variable y: 1 Variable acum: 2
Iteración número: 27	Variable res: -1 Variable x: 0 Variable y: 2 Variable acum: 2
	Variable res: -1

Iteración número: 28	Variable x: 1 Variable y: 2 Variable acum: 2
Iteración número: 29	Variable res: -1 Variable x: 2 Variable y: 2 Variable acum: 2
Iteración número: 30	Variable res: -1 Variable x: 3 Variable y: 2 Variable acum: 2
Iteración número: 31	Variable res: -1 Variable x: 4 Variable y: 2 Variable acum: 1
Iteración número: 32	Variable res: -1 Variable x: 5 Variable y: 2 Variable acum: 1
Iteración número: 33	Variable res: -1 Variable x: 6 Variable y: 2 Variable acum: 1
Iteración número: 34	Variable res: -1 Variable x: 7 Variable y: 2 Variable acum: 0
Iteración número: 35	Variable res: -1 Variable x: 8 Variable y: 2 Variable acum: 0
Iteración número: 36	Variable res: -1 Variable x: 9 Variable y: 2 Variable acum: 0
Iteración número: 37	Variable res: -1 Variable x: 10 Variable y: 2 Variable acum: -1
Iteración número: 38	Variable res: -1 Variable x: 11 Variable y: 2 Variable acum: -1