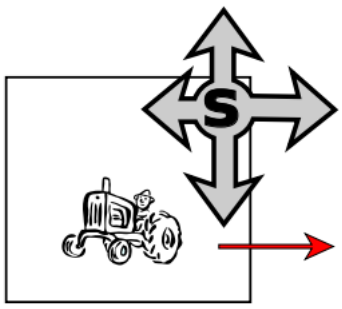
SISTEMAS INTELIGENTES

Práctica 1: Implementación del artefacto terreno

Adrián Muñoz Llano
Ángel Sánchez González
Javier Monescillo Buitrón
Repositorio: https://github.com/javirmones/SistemasInteligentes

BC1_06

Índice



-Introducción al problema

La primera tarea que se desarrolla en las prácticas de sistemas inteligentes consiste en generar un terreno T de dimensiones CxF ([0..C-1] x [0..F-1])**,** en cuyas casillas puede haber una cantidad de arena no mayor que una cantidad MAX tal que ($max \geq k + s$), y también un tractor situado en una casilla (*xt, yt*) determinada, donde:

k: es la cantidad deseada en cada casilla

s: es la cantidad a redistribuir entre las casillas vecinas

MAX: es la cantidad máxima existente en una casilla

C: son las columnas que tendrá nuestro terreno

F: el número de filas que tendrá el terreno

V: será el peso total a repartir en todas las columnas, donde *V=C*F*k*

Xt: posición absoluta del tractor en X.

Yt: posición absoluta del tractor en Y.

Terreno[][]: la matriz que representará el terreno que estará definido por C y F.

--Conjunto de operaciones a realizar:

-Creación del terreno

--Creación del terreno a través de teclado

--Creación del terreno a través de un fichero

-Generación de todas las acciones posibles a partir de un terreno con el tractor en la casilla (Xt, Yt)

--Generación de los vecinos asociados a la posición del tractor

-Obtención de un nuevo terreno tras aplicar una acción a uno dado.

-Estructuras e implementación de las operaciones

-Creación del terreno:

```
public static Terreno cargarDatosTeclado() throws Exception {
```

Para poder generar el terreno podemos utilizar o bien el método *cargarDatosTeclado(),* cuya salida será un objeto terreno con todas las propiedades definidas anteriormente.

```
public static Terreno cargarDatosFichero() throws Exception {
```

O bien podemos utilizar este método para poder cargar mediante el fichero *FicheroPrueba.txt*

El objeto terreno se rellenará automáticamente con números aleatorios, lógicamente con los parámetros correctamente indicados con el método

```
public static void mostrarTerreno(Terreno t){
```

-Generación de todas las acciones posibles de un terreno:

--Obtención de un nuevo terreno tras aplicar una acción a uno dado

Para comenzar generamos todos los vecinos asociados a la posición (Xt,Yt) del tractor, como máximo puede tener cuatro vecinos, como mínimo dos, para ello empleamos el siguiente método, que devolverá un ArrayList con los vecinos generados En este método lo que hacemos es crear un objeto vecino para cada vecino que tenga el tractor, el cual posee la cantidad de arena, la arena máxima, la posición en X e Y y la cantidad de arena que se le distribuye:

```
public static ArrayList generarVecinos(Terreno t){
```

Cuyos parámetros son el objeto Terreno t, y cuya salida será un vector con el valor de los vecinos.

Para encargarnos de la distribución utilizaremos:

```
public static void distribucion(int etapa, int k, int actual, ArrayList<Vecino> vecinos,
        ArrayList<ArrayList> todasDistribuciones){
```

Donde el parámetro k es equivalente a s: que es la cantidad por redistribuir entre las casillas vecinas**,** max como el máximo que soporta cada casilla, la lista de vecinos, y la lista final con todas las posibles distribuciones.

De esto se encarga un algoritmo recursivo de backtracking que no hemos conseguido finalizar su implementación, ya que no sale el resultado esperado. Para la obtención de un nuevo terreno simplemente se utilizaría la creación de

una nueva matriz, dentro de un propio objeto terreno y la previa utilización del anterior algoritmo.