GRUPO C1-04

  SISTEMAS INTELIGENTES

documentación de la práctica

# ÍNDICE



## Representación del problema y creación de un terreno.

## Lectura y escritura de un terreno desde/hacia un fichero.

## Generación de las acciones posibles.

## Definición del nodo y frontera del árbol de búsqueda.

# Representación del problema y creación de un terreno.

Sea un terreno T de dimensiones CxF, en cuyas casillas puede haber una cantidad de arena no mayor que una cantidad MAX (max >= k+s), y un pequeño tractor en una casilla (Xt, Yt) determinada.

En este apartado lo que queremos es representar dicho enunciado en nuestro programa .java donde tendremos que crear la clase terreno, la cual tendrá todos los datos necesarios para crear dicho terreno, que serán:

Xt: posición del tractor, coordenada X.

Yt: posición del tractor, coordenada Y.

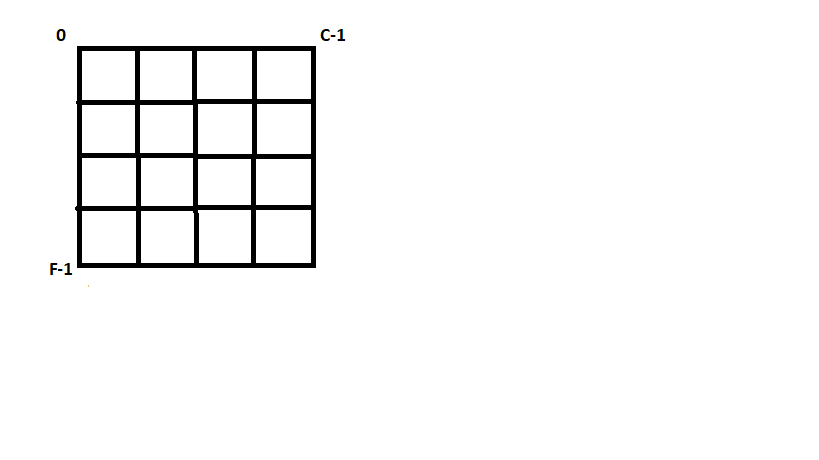
K: cantidad de arena deseada en cada casilla.

Max: arena máxima que puede alojar cada casilla.

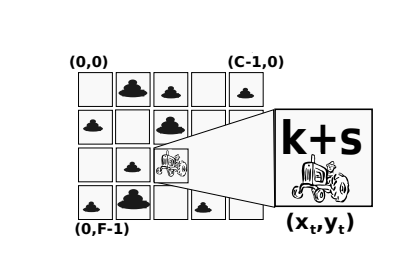
C: número de columnas de la matriz terreno.

F: número de filas de la matriz terreno.

Matriz terreno: matriz que rellenaremos con la arena correspondiente.

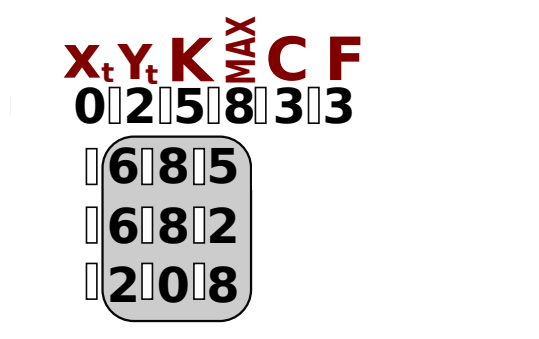


Esta será nuestra matriz terreno la cual tendremos que rellenar con la arena correspondiente.

Dicha matriz la rellenaremos en el método inicializarTerreno() de la clase Terreno el cual irá asignando arena aleatoriamente dentro de los limites establecidos (0…Max), hasta que la cantidad de arena a repartir V= C\*F\*K sea igual a 0.

# Lectura y escritura de un terreno desde/hacia un fichero.

En este apartado nuestro objeto Terreno, que ya hemos creado anteriormente, será necesario guardarlo en un fichero.



Dicha acción la representaremos en el método GuardarTerreno() de la clase Auxiliar, al cual le pasaremos como parámetro nuestro objeto terreno, que guardara toda la cabecera del mismo (sus atributos) y después la matriz terreno de la siguiente manera:

La lectura desde fichero la realizaran dos métodos, el método leerCabecera() y el método leerTerreno() que englobará al anterior, ambos pertenecientes a la clase Auxiliar. A estos dos métodos será necesario pasarles como parametro la ruta donde se encuentra nuestro archivo donde se encuentra el terreno que vamos a leer.

El método leerCabecera() creará una cadena de caracteres con todos los atributos de nuestro objeto Terreno, el cual crearemos en el método leerTerreno() y crearemos/rellenaremos la matriz terreno con su correspondiente arena en cada casilla.

# Generación de las acciones posibles

Para generar todas las combinaciones de desplazar arena se ha seguido una estrategia de backtracking.

La estrategia consta de 4 etapas. En cada una se evaluará trasladar una cantidad de arena ***a*** que será menor que la arena sobrante en la casilla que se encuentra el tractor a cada uno de los puntos cardinales (Norte, Sur, Este y Oeste).

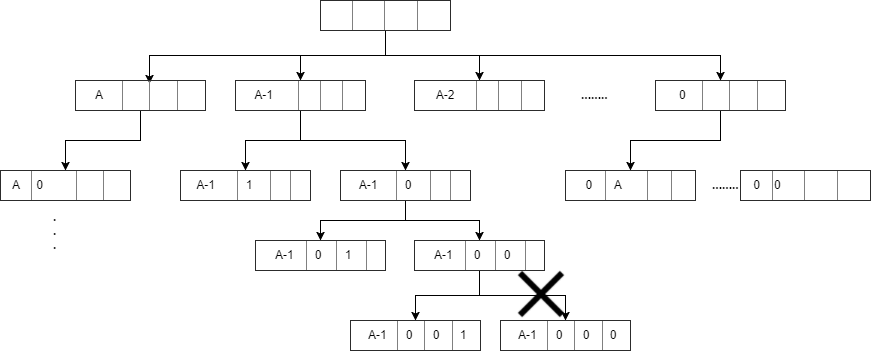
La forma de la solución es una lista de vectores de cuatro posiciones cada uno en los que cada posición representa un punto cardinal (N, S, E, O) y el valor en dicha posición es la cantidad de arena que se traslada a dicho punto cardinal

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 0 | 1 |

Suponiendo que en la casilla del tractor hay 4 montones de arena para repartir, un posible desplazamiento de dicha arena podría ser:

N S E O

El árbol generador de desplazamientos es este:



Los nodos finales no se añaden al conjunto de soluciones si no se desplaza toda la arena disponible.

Para completar todos los movimientos posibles, a los desplazamientos generados se les asigna el desplazamiento del tractor a una casilla (nuevamente NSEO) por lo que todos los movimientos posibles serán todos los desplazamientos de arena por 4.

Se ha usado la forma de vector por su simpleza para trabajar y generar los nodos del árbol.

# Definición del nodo y la frontera del árbol de búsqueda.

Vamos a definir los nodos de nuestro árbol de búsqueda con las propiedades que vienen en las diapositivas, por tanto cada nodo tendrá las siguientes características:

1. Nodo del padre: hace referencia a al nodo del cual viene nuestro nodo.
2. Matriz terreno: que nos engloba el estado actual de nuestro problema.
3. Costo: cuanto nos cuesta llegar desde el nodo inicial al actual.
4. Acción: movimiento que vamos a realizar, donde y cuanta arena vamos a distribuir.
5. Valor: es el valor por el cual se va a ir ordenando la frontera.

Cuando ya tenemos nuestros nodos creados, necesitamos una frontera que definiremos como un Array List e iremos metiendo en dicha frontera y por orden de valor, en nuestro caso de menor a mayor, cada nodo que hayamos definido y tengamos que recorrer posteriormente.

Tanto para insertar, eliminar y comprobar si dicha frontera tiene algún elemento serán métodos de la propia clase Frontera.