Práctica 1. Algoritmos devoradores

9 de enero de 2015

Esta práctica consiste en diseñar un algoritmo para la colocación de las defensas en el planeta que siga una estrategia devoradora. Debe ser un algoritmo determinista, esto es, para una misma configuración de entrada debe devolver siempre la misma salida. La función de factibilidad del algoritmo debe ser explícita e independiente de la función de selección.

Antes de proceder a la creación de dicho algoritmo se explicará cómo debe generarse la biblioteca dinámica donde se alojará el algoritmo y también se comentarán algunas de las herramientas que pueden ser utilizadas para la depuración del programa. Sitúese en el directorio BASE/p1 y ejecute la siguiente orden.

```
make test
```

Compruebe que el simulador se ejecuta correctamente y consulte el fichero *Makefile* para entender la orden que acaba de ejecutar.

La estrategia para la colocación de las defensas de la base se encuentra implementada en el fichero *libDefenseStrategy.so*. Antes de sobrescribir este fichero es conveniente realizar una copia del mismo, de tal forma que pueda volver a recuperarse este comportamiento en cualquier momento. Ejecute la siguiente orden.

```
cp libDefenseStrategy.so libDefenseStrategy.default.so
```

Para recuperar de nuevo esta estrategia defensiva solo tendrá que ejecutar las siguientes órdenes.

```
rm libDefenseStrategy.so
cp libDefenseStrategy.random.so libDefenseStrategy.so
```

Cree una copia del fichero *DefenseStrategy.example.cpp* ejecutando la siguiente orden.

```
cp DefenseStrategy.example.cpp DefenseStrategy.cpp
```

Abra el fichero de nombre *DefenseStrategy.cpp* con algún editor de texto y localice la función *placeDefenses*. Deberá implementar su versión del algoritmo dentro de dicha función. Puede añadir al fichero *DefenseStrategy.cpp* tantas funciones auxiliares como considere necesarias pero deben estar contenidas siempre dentro del fichero *DefenseStrategy.cpp*. Ejecute la siguiente orden para generar la nueva versión de la biblioteca.

```
make
```

Tenga en cuenta que la versión del ejemplo no se corresponde con la versión compilada de la biblioteca que ha estado usando hasta el momento. En la versión del ejemplo no se comprueba que la ubicación de una defensa sea válida. En la versión compilada las localizaciones de las defensas son siempre válidas.

La memoria de la práctica deberá realizarse en latex. Para simplificar esta tarea se han creado una serie de documentos de texto y un *script* que se encarga de generar la memoria a partir de ellos. Usted solo deberá editar los documentos de texto correspondientes a cada ejercicio (*e1.tex*, *e2.tex*...). Abra el fichero *e1.tex* en un editor de texto básico y modifique ligeramente su contenido. Ejecute la siguiente orden y compruebe el resultado.

```
make doc
```

Si ha seguido atentamente las instrucciones anteriores, el empaquetado de la memoria para su entrega a través del campus virtual es muy sencillo. Basta con que ejecute la siguiente orden.

```
make pack
```

Renombre el fichero *APELLIDO1_APELLIDO2_NOMBRE-p1.tar.gz* para hacerlo coincidir con su nombre y apellidos.

Por último, es muy recomendable que revise el fichero *BASE/simulador/Asedio.h* aténtamente. Encontrará información que le será de utilidad a la hora de resolver los ejercicios propuestos en esta y otras prácticas.

1. Ejercicios

1. Describa a continuación la función diseñada para otorgar un determinado valor a cada una de las celdas del terreno de batalla para el caso del centro de extracción de minerales.
2. Diseñe una función de factibilidad explicita y descríbala a continuación.
3. A partir de las funciones definidas en los ejercicios anteriores diseñe un algoritmo voraz que resuelva el problema para el caso del centro de extracción de minerales. Incluya a continuación el código fuente relevante.
4. Comente las características que lo identifican como perteneciente al esquema de los algoritmos voraces.
5. Describa a continuación la función diseñada para otorgar un determinado valor a cada una de las celdas del terreno de batalla para el caso del resto de defensas. Suponga que el valor otorgado a una celda no puede verse afectado por la colocación de una de estas defensas en el campo de batalla. Dicho de otra forma, no es posible modificar el valor otorgado a una celda una vez que se haya colocado una de estas defensas. Evidentemente, el valor de una celda sí que puede verse afectado por la ubicación del centro de extracción de minerales.
6. A partir de las funciones definidas en los ejercicios anteriores diseñe un algoritmo voraz que resuelva el problema global. Este algoritmo puede estar formado por uno o dos algoritmos voraces independientes, ejecutados uno a continuación del otro. Incluya a continuación el código fuente relevante que no haya incluido ya como respuesta al ejercicio 3.

Todos los ejercicios tienen la misma puntuación. No es necesario que explique el código fuente en los ejercicios en los que se le solicita que lo incluya. Puede incrustar en el código los comentarios que considere oportunos.