Práctica 1. Algoritmos devoradores

Carmen del Mar Ruiz de Celis
carmen.ruizdecelis@alum.uca.es
Teléfono: xxxxxxxx
NIF:49565250C

17 de noviembre de 2019

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.

   El criterio que he elegido para la colocación de la primera defensa depende de dos factores: el número de obstáculos que haya alrededor y la distancia al centro del mapa. He considerado que el concepto "alrededor" es todo obstáculo a una distancia menor o igual al rango de alcance de la defensa. La función devuelve el número de obstáculos menos la distancia entre el centro y la posición, ya que: - La función de selección devuelve el mayor valor del mapa - Nos interesa que se valore más que haya poca distancia respecto al centro - La distancia siempre será mayor que el número de obstáculos

   En este orden de los operandos, tendremos un número negativo, haciendo que la función de selección valore la poca cercanía y la cantidad de obstáculos

2. Diseñe una función de factibilidad explicita y descríbala a continuación.

   La función de factibilidad comprueba si la defensa se puede colocar en una posición dadas la fila y la columna. Luego, es necesario comprobar si colisionaría con algún obstáculo u defensa y si no excedería los límites del mapa.

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.

```
void DEF_LIB_EXPORTED placeDefenses(bool** freeCells, int nCellsWidth, int nCellsHeight,
    float mapWidth, float mapHeight
              , std::list<Object*> obstacles, std::list<Defense*> defenses) {


    float cellWidth = mapWidth / nCellsWidth;
    float cellHeight = mapHeight / nCellsHeight;
    int maxAttemps = 1000; //N mero de intentos en el que debe encontrar una posici n
        factible

    //inicializo el mapa de valores
    std::vector<std::vector<float>> MapValue(nCellsHeight, std::vector<float>(nCellsWidth));
        //Mapa para el resto de defensas
    std::vector<std::vector<float>> MapValueFirstD(nCellsHeight, std::vector<float>(
        nCellsWidth)); //Mapa para para la primera

    std::list<Defense*> defensescopy = defenses; //Hago una copia de la lista de defensas
        para ordenarlas
    defensescopy.erase(defensescopy.begin()); //Borro la primera defensa
    defensescopy.sort(compare()); //Las ordeno seg n una puntuaci n de sus caracter sticas

    // Le doy valores a los mapas
    for(int i = 0; i < nCellsHeight; i++){
        for(int j = 0; j < nCellsWidth; j++){
            MapValue[i][j] = cellValue(i, j, nCellsWidth, nCellsHeight, mapWidth, mapHeight,
                obstacles, *defenses.begin());
            MapValueFirstD[i][j] = FirstDefensecellValue(i, j, nCellsWidth, nCellsHeight,
                mapWidth, mapHeight, obstacles, defenses);
```

```
        }
    }

   int row, col; //variables para obtener la fila y la columna a partir de positiontoCell
   Vector3 v; //Guardar  la posici n devuelta por selection (la mejor del mapa)
   bool firstTime = false; //Variable que indicar  si la primera defensa se ha colocado o no
   for( List<Defense*>::iterator currentDefense = defensescopy.begin(); currentDefense !=
       defensescopy.end() && maxAttemps > 0; maxAttemps--){
        if(!firstTime){ //Como estoy recorriendo la copia sin la primera defensa, tengo que
            hacer este bloque aparte para no utilizar
                        //la lista original en el bucle for
            v = selection(MapValueFirstD, cellHeight, cellWidth, nCellsWidth, nCellsHeight,
                defenses); //obtengo la mejor posici n del mapa
             positionToCell(v, row, col, cellWidth, cellHeight); //obtengo la fila y la
                 columna de la mejor posici n
            if(feasibility((*defenses.begin())->id, row, col, nCellsWidth, nCellsHeight,
                mapWidth, mapHeight, obstacles, defenses)){
                maxAttemps = 1000;
                (*defenses.begin())->position = v;
                firstTime=true;
                //Si la posici n hayada por selection es factible, se resetean los intentos,
                     se le asigna la posici n a la defensa
                //y se pasa a la siguiente. En este bloque se hace lo mismo que en el del
                    else, s lo que las funciones reciben
                //par metros distintos (el mapa y el id de la primera defensa, que requiere
                    de la lista original). Adem s, necesita
                //indicar que la primera defensa se ha colocado con la variable firstTime.
             }

        }else{
          // En este bloque se hace lo mismo que arriba, s lo que
            v = selection(MapValue, cellHeight, cellWidth, nCellsWidth, nCellsHeight,
                defenses);
            positionToCell(v, row, col, cellWidth, cellHeight);
            if(feasibility((*currentDefense)->id, row, col, nCellsWidth, nCellsHeight,
                mapWidth, mapHeight, obstacles, defenses)){
                maxAttemps = 1000;
                (*currentDefense)->position = v;
                currentDefense++;
            }
        }
   }

}

struct compare {
    bool operator()(Defense * d1, Defense * d2) {return ((d1->health*1/5 + d1->
        attacksPerSecond*1/5 + d1->range*1/5 + d1->damage*1/5
                                                    + d1-> dispersion*1/5) > (d2->health
                                                        *1/5 + d2->attacksPerSecond*1/5 +
                                                         d2->range*1/5 + d2->damage*1/5
                                                    + d2-> dispersion*1/5));}
};

//Estructura usada como criterio de ordenaci n en la copia de la lista de defensas
    bas ndose en la puntuaci n de sus caracter sticas.
```

4. Comente las características que lo identifican como perteneciente al esquema de los algoritmos voraces.

   Se distinguen: - Un conjunto de candidatos: los mapas. - Un conjunto de candidatos seleccionados: aquellos elegidos por selection (los de mayor valor) - Una función de selección que indica el candidato más prometedor de los que quedan - Una función de factibilidad que comprueba si un candidato seleccionado es válido - Una función objetivo: que todas las defensas sean colocadas

   Además, sigue el esquema de la estructura de un algoritmo voraz

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.

Para el resto de defensas, he considerado que tenga en cuenta: - El número de obstáculos alrededor, donde "alrededor" significa en un radio máximo equivalente a la diagonal de una celda del mapa - La distancia a la primera defensa. - La distancia al centro

De igual manera que el valor retornado por la función que da valores al terreno para la primera defensa, los signos de la operación devuelta están dispuestos de manera que premia tener muchos obstáculos alrededor y poca distancia al centro y a la primera defensa, haciendo hincapié en esta última.

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.

```
El algoritmo voraz, tanto para el centro de extracci n como para el resto de defensas, se
    hacen conjuntamente en el mismo c digo, ya comentado en el ejercicio 3.
```

Todo el material incluido en esta memoria y en los ficheros asociados es de mi autoría o ha sido facilitado por los profesores de la asignatura. Haciendo entrega de este documento confirmo que he leído la normativa de la asignatura, incluido el punto que respecta al uso de material no original.