Práctica 2. Programación dinámica

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

8 de diciembre de 2019

1. Formalice a continuación y describa la función que asigna un determinado valor a cada uno de los tipos de defensas.

$$f(damage, attacksPerSecond, range, dispersion, health) =$$
$$(damage/attacksPerSecond) * 4/5 + (range * dispersion * health) * 1/5$$

   El criterio de evaluación de las defensas consiste en una suma de dos partes, que se desglosa en:

   - El daño por segundo, de la relación del daño entre los ataques por segundo, en el que recae la mayor parte de la nota al estar ponderado con un 4/5. Poco importa la salud, el rango o la dispersión si la defensa es superior en lo que respecta a daño.
   - El producto del rango, la dispersión y la salud. Claro está que el rango y la dispersión están estrechamente ligadas, pero, al tener la salud un gran papel ese 1/5, lo más correcto es utilizarlo como "potenciador" del resultado anterior (rango y dispersión).

2. Describa la estructura o estructuras necesarias para representar la tabla de subproblemas resueltos.

   Como estructura he elegido una matriz de tipo *float* (adaptado al tipo que devuelve mi criterio de evaluación de las defensas), de dimensiones: *n^o de defensas*coste máximo*; ya que, aplicaré el algoritmo propuesto para *el problema del vendedor viajero* o *TSP* visto en clase, que permite calcular las mejores defensas a las que podemos optar en función de un coste máximo dado.

3. En base a los dos ejercicios anteriores, diseñe un algoritmo que determine el máximo beneficio posible a obtener dada una combinación de defensas y *ases* disponibles. Muestre a continuación el código relevante.

```
void selectDefenses(std::list<Defense*> defenses, unsigned int ases, std::list<int> &
    selectedIDs
            , float mapWidth, float mapHeight, std::list<Object*> obstacles){
    //Tengo en cuenta primero la primera defensa
    selectedIDs.push_front((*defenses.begin())->id);
    ases -= (*defenses.begin())->cost;
    defenses.pop_front(); // La elimino de la lista de defensas ya que debe quedar fuera de
        la b squeda de la mejor combinaci n
    std::vector<std::vector<float>> m(defenses.size(), std::vector<float>(ases + 1)); //tabla
         de subproblemas

    cellvalue(m, defenses, ases); //relleno la tabla
    topDefenses(m, selectedIDs, defenses, ases); //interpreto la tabla y extraigo las mejores
         defensas

}

//algoritmo TSP
void cellvalue(std::vector<std::vector<float>> &m, std::list<Defense*> defenses, unsigned int
     ases){
    std::list<Defense*>::iterator it = defenses.begin();
    for(int i = 0; i < ases + 1; i++){
        if(i < (*it)->cost ) m[0][i] = 0;
```

```
        else m[0][i] = DefenseValue(*it);
    }

    for(int j =1; j < defenses.size(); j++, it++){
        for(int k = 0 ; k < ases + 1; k++){
            if(k < (*it)->cost) m[j][k] = m[j-1][k];
            else m[j][k] = std::max(m[j-1][k], m[j-1][k - (*it)->cost] + DefenseValue(*it));
        }
    }

}
```

4. Diseñe un algoritmo que recupere la combinación óptima de defensas a partir del contenido de la tabla de subproblemas resueltos. Muestre a continuación el código relevante.

```
void topDefenses(std::vector<std::vector<float>> &m, std::list<int> &selectedIDs, std::list<
    Defense *> defenses, unsigned int ases){

    std::list<Defense *>::iterator itDefe = defenses.end();
    --itDefe;
    int j = ases;
    for (int i = defenses.size()-1; i > 0 && ases > 0; --i, --itDefe){
        if (m[i][j] != m[i-1][j]){
            selectedIDs.push_back((*itDefe)->id);
            ases -= (*itDefe)->cost;
            j-=(*itDefe)->cost;
        }
    }
    if(ases >= (*defenses.begin())->cost){
            selectedIDs.push_back((*defenses.begin())->id);
            ases -= (*defenses.begin())->cost;
    }

}
```

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.