Práctica 2. Programación dinámica

Manuel Diaz Gil
manuel.diazgil@alum.uca.es
Teléfono: 667361517
NIF: 45382945N

3 de diciembre de 2017

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

$$f(valorar) = (rango/30) * 5 + dispersion * 5 + ((daño * ataquesPorSegundo)/5) * 15 + (salud/500) * 20$$

El valor de la defensa se asignna en funcion del rango, el cual se divide por 30 para que quede mas menos alrededor de 1, la dispersion, el dañor multiplicado por los ataques por segundo, para asi obtener el daño por segundo de la torreta, y la salud, divida entre 500 para que este alrededor de 1.

Las multiplicaciones por numeros constantes se dan segun el orden de importancia. De esta manera le damos mas prioridad a la salud y al daño por segundo.

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

Las estructuras necesarias son una matriz, de tipo double con dimension numero de defensas por los recursos iniciales, y dos vectores de tipo double con dimension numero de defensas, para los costes de cada defensa y el valor que se le da segun nuestra funcion.

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.

```
double tabla[defenses.size()][ases];
   double valor[defenses.size()];
   double costes[defenses.size()];


   std::list<Defense*>::iterator it = defenses.begin();
   int i = 0;
   for(it = defenses.begin(); it!=defenses.end();++it){
       valor[i] = valorar(*(*it));
       costes[i]=(*it)->cost;
       i++;
   }

   for (int j = 0; j < ases; ++j)
   {
       if(j<costes[0]){
           tabla[0][j] = 0;
       }else{
           tabla[0][j] = valor[0];
       }
   }

   for (i = 1; i < defenses.size(); i++)
   {
       for (int j = 0; j < ases; j++)
       {
           if(j < costes[i]){
               tabla[i][j] = tabla[i-1][j];
           }else{
```

```
                tabla[i][j] = std::max(tabla[i-1][j],tabla[i-1][j-(int)costes[i]]+valor[i]);
            }

        }

    }
```

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.

```
    it = defenses.begin();
    int currentases = ases;
    selectedIDs.push_back((*it)->id);
    currentases -= (*it)->cost;

    it = defenses.end();
    for (i = defenses.size()-1; i > 0; --i)
    {
        --it;
        if (tabla[i][currentases]!=tabla[i-1][currentases])
        {
            selectedIDs.push_back((*it)->id);
            currentases -= (*it)->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.