Práctica 2. Programación dinámica

Alejandro Segovia Gallardo
alejandro.segoviagallardo@alum.uca.es
Teléfono: 608842858
NIF: 32083695Y

1 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.

$$f(defensa) = daño * ataquesporsegundo * dispersion * rango * salud$$

   Mi estrategia de puntuacion de las defensas se ha basado en la multiplicacion de todas sus características principales mas significativas(todas menos el radio de la propia defensa ya que no considero que sea influyente en la defensa).

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

   Para representar la tabla de subproblemas resueltos he usado una matriz dinámica de tantas filas como defensas dispongamos y tantas columnas como cantidad de presupuesto(ases) que recibamos de entrada. En la eleccion de la matriz como estructura que soporte la tabla de subproblemas resueltos ha sido determinante la facilidad que aporta esta misma para la comprobación de las variaciones al añadir nuevas defensas.

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 algoritmo_mochila(unsigned int ases, std::list<Defense*> defenses,
                       float rendimiento[], unsigned int coste[],float** matDef)
{

  for(int i=0;i<=ases;i++){
    if(i<coste[0])
    {
      matDef[0][i]=0;
    }
    else
    {
      matDef[0][i]=rendimiento[0];
    }
  }

  for(int i=1;i<defenses.size();i++){
  for(int j=0;j<=ases;j++)
    if(j<coste[i])
      matDef[i][j]=matDef[i-1][j];

    else
      matDef[i][j]=std::max(matDef[i-1][j], matDef[i-1][j-coste[i]] +
                                            rendimiento[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.

```
void DEF_LIB_EXPORTED selectDefenses(std::list<Defense*> defenses,
                                     unsigned int ases,
                                     std::list<int> &selectedIDs,
                                     float mapWidth, float mapHeight,
                                     std::list<Object*> obstacles){

    //Contenido previo

    if(ctotal <= ases)
      for(std::list<Defense*>::iterator it=defenses.begin();it!= defenses.end();
          it++)
            selectedIDs.push_back((*it)->id);

    else{
    //Contenido previo

    while(i>=0 && j>0){
      if(i>0 && matDef[i][j] == matDef[i-1][j]){
        i--;
        it--;
      }
      else{
        selectedIDs.push_back((*it)->id);
        j = j - (*it)->cost;
        i--;
        it--;
      }
    }
  }
}
```

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.