Práctica 2. Programación dinámica

AGUSTIN ENEAS HARISPE LUCARELLI
correo@servidor.com
Teléfono: xxxxxxxx
NIF: 55080402W

27 de noviembre de 2022

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

$$f(defensa) = \frac{2 \cdot vida + 1000 \cdot damage + 5000 \cdot aps + 30 \cdot rango}{coste}$$

   Para determinar el valor defensivo de nuetras defensas debemos entender o presuponer cuales son los atributos positivos y negativos para las mismas, además de conocer sus valores medios o por defecto para entender su escala. Dicho esto la formula consiste en una división donde como numerador encontramos la suma de atributos que aportan valor defensivo y como denomiador la suma de atributos que restan, siendo este el coste. Los valores con aleatoriedad 0 de cada atributo son: Vida = 500, Daño = 5, AtaquePorSegundo = 1, Rango = 30, Coste = 100 Los atributos del numerador son escalados por una constante multiplicativa equiparandolos entre ellos y dando más valor a los atributos que considero más importantes que en el siguiente orden son:

$$Damage = APS > Vida > Rango$$

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

   Necesitamos una matriz de tamaño [nº defensas][ases] para dicha tabla y un dos vectores tamaño [nº defensas] para el valor y costes de las mismas respectivamente.

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.

```
//funcion que da valor a las defensas
int DefensesValue(List<Defense*>::iterator defensa){
    float rango = (*defensa)->range;
    float dispersion = (*defensa)->dispersion;
    float dmg = (*defensa)->damage;
    float dps = (*defensa)->attacksPerSecond;
    float vida = (*defensa)->health;
    float coste = (*defensa)->cost;

    float resultado = (vida*2+dmg*1000+dps*5000+rango*30);

    return static_cast<int>(resultado);
}
```

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.

```
//recuperacion de combinacion optima
    int i = size-1;
    int j = dinero-1;
    while(i>0 && j>0){
        if(tabla[i][j] == tabla[i-1][j]);
        else {
```

```
            if(j-costes[i] > 0){
                selectedIDs.push_back(i+1);
                j-= costes[i];
            }
        }
        i--;
    }
```

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.