Práctica 2. Programación dinámica

Miguel Ángel Fernández JImínez
miguel.ferjimenez@alum.uca.es
Teléfono: 601101415
NIF: 77497367V

November 27, 2022

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

   Escriba aquí su respuesta al ejercicio 1.

   valor = (def.damage*def.attacksPerSecond*def.health*def.range) / (def.dispersion);

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

   Escriba aquí su respuesta al ejercicio 2.

   Una matriz reprensentada con un vector de vectores de tipo float, de la siguiente forma, std::vector¡std::vector¡float¿¿ TSP, cada vector representará uno los valores y otro los costes de las defensas, respectivamente.

   Una lista donde están almacenadas las defensas, que nos permite recorrerlas con mayor facilidad. Nos permite también las inserciones y borrados con mayor rapidez.

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.

```
// sustituya este codigo por su respuesta
void beneficioMaximo(std::vector<std::vector<float>>& spTable, std::list<Defense*> defenses,
    unsigned int ases)
{
    float values[defenses.size() - 1];
    int costs[defenses.size() - 1];
    List<Defense*>::iterator it = defenses.begin();
    it++;
    //Relleno los vectores
    for(int i = 0; i < defenses.size()-1; i++)
    {
        costs[i] = (*it)->cost;
        values[i] = valDefensa((**it));
        it++;
    }

    ases = ases - (*defenses.begin())->cost;
    for(int j = 0; j < ases+1; j++)
    {
        if (j < costs[0])
            spTable[0][j] = 0;
        else
            spTable[0][j] = values[0];
    }
    for(int i = 1; i < defenses.size()-1; i++){
        for(int j = 0; j < ases+1; j++)
        {
            if(j < costs[i])
                spTable[i][j] = spTable[i-1][j];
            else
                spTable[i][j] = std::max(spTable[i-1][j], spTable [i-1][j-costs[i]] + values[
                    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.

```
// sustituya este codigo por su respuesta
void defensasOptimas(std::list<Defense*> defenses, unsigned int ases, std::list<int>&
    selectedIDs, const std::vector<std::vector<float>>& spTable)
{
    int i = defenses.size() - 2, j = ases - (*defenses.begin())->cost;
    List<Defense*>::iterator it = defenses.end();

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

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.