Práctica 2. Programación dinámica

David Luna Jurado
david.lunajurado@alum.uca.es
Teléfono: 663590384
NIF: 32098835N

26 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.

$$f(h, d, aps, r, disp) = h * 0.4 + (\frac{d}{aps}) * 0.4 + r * 0.15 + disp * 0.05$$

   Donde h es la salud de la defensa, d el daño, aps los ataques por segundo, r el rango y $disp$ la dispersión.

   La función es una ponderación arbitraria de las diferentes características de la defensa, se ha considerado esta estrategia debido a que se obtenía el mejor resultado en las pruebas realizadas

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

   Para representar la tabla de problemas resueltos he utilizado dos estructuras.
   La primera es un vector de valores que es donde están almacenados los valores de las defensas.
   La segunda estructura es un vector de punteros a defensa. Podría ser un vector de costes donde se almacenen los costes de las defensas, pero como son necesarias las ids de las defensas para rellenar la lista de las defensas seleccionadas he considerado como lo mejor almacenar todas las direcciones de las defensas en un vector en lugar de coste e id por separado.

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)
{
    int coste = 0;
    int tam = defenses.size();
    Defense *defensas[tam];
    float valor[tam];
    int i = 0;

    if (ases >= defenses.front()->cost)
    {
        ases -= defenses.front()->cost;

        for (Defense *d : defenses)
        {
            if (d != defenses.front())
            {
                defensas[i] = d;
                valor[i] = defenseValue(d);
                i++;
            }
        }

        float tabla[tam][ases + 1];
        for (size_t i = 0; i < tam; i++)
        {
```

```
                for (size_t j = 0; j <= ases; j++)
                {
                    if (i == 0)
                    {

                        if (j < defensas[i]->cost)
                        {
                            tabla[i][j] = 0;
                        }

                        else
                        {
                            tabla[i][j] = valor[i];
                        }
                    }
                    else // i > 0
                    {
                        if (j < defensas[i]->cost)
                            tabla[i][j] = tabla[i - 1][j];
                        else // j >= costes[i]
                        {
                            tabla[i][j] = std::max(tabla[i - 1][j], tabla[i - 1][j - defensas
                                [i]->cost] + valor[i]);
                        }
                    }
                }
            }

            int valorOptimo = tabla[tam - 1][ases];
        }
    }
```

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 reconstruirCamino(int** tabla, Defensa* defensas, int ases, int tam, list<Defensa*>
    defenses)
{

    // Reconstruir camino
    int i = tam - 1;
    int j = ases;
    while (i >= 0 && j > 0)
    {
        if (i > 0)
        {
            if (tabla[i][j] != tabla[i - 1][j])
            {
                selectedIDs.push_front(defensas[i]->id);

                j -= defensas[i]->cost;
            }
        }
        if (i == 0 && tabla[i][j] > 0)
        {
            selectedIDs.push_front(defensas[i]->id);
        }
        i--;
    }
    selectedIDs.push_front(defenses.front()->id); //Agregamos el centro de extraccion a la
        lista, no estaba en la tabla de subproblemas resueltos porque siempre se coloca

}
```

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.