Práctica 2. Programación dinámica

Iván Alba Gómez
ivan.albagomez@alum.uca.es
Teléfono: XXXXXXXXX
NIF: 49302616T

23 de noviembre de 2021

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

$$f(radio, rango, salud) = (radio + rango) - 1 + rango^2 + 3\frac{radio}{salud}$$

La función que determina un valor para cada tipo de defensa la he llamado defenseValue. Dicha función recibe una defensa y devuelve un valor siguiendo mi criterio. El criterio utilizado es el siguiente:

Mi criterio de importancia (0-100 %):

range ———————— 5 %

dispersion ———————— 15 %

damage ————————- 30 %

attacksPerSecond ——— 20 %

health ————————- 30 %

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

Para representar la tabla de subproblemas resueltos he utilizado un vector de vectores.

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 mochila(float** tsp, unsigned int ases, std::list<Aux> myDefenses) {
    std::list<Aux>::iterator it = myDefenses.begin();
    for(int j = 0; j <= ases && it != myDefenses.end(); j++) {
        if(j < it->getCost()) tsp[0][j] = 0;
        else tsp[0][j] = it->getValue();
        it++;
    }
    it = myDefenses.begin(); it++;
    for(int i = 1; i <= myDefenses.size() && it != myDefenses.end(); i++) {
        for(int j = 0; j <= ases; j++) {
            if(j < it->getCost()) tsp[i][j] = tsp[i-1][j];
            else tsp[i][j] = std::max(tsp[i-1][j], tsp[i-1][j-it->getCost()] + it->getValue()
                );
        }
        it++;
    }
}
```

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 selectIds(float** tsp, unsigned int ases, std::list<Aux> myDefenses, std::list<int> &
    selectedIDs) {
    std::list<Aux>::iterator it = myDefenses.end()--;   // end() devuelve la siguiente
        posici n a la  ltima , por eso el --
    int j = ases;
    for(int i = myDefenses.size()-1; i > 0; i--, it--) {
        if(tsp[i][j] != tsp[i-1][j]) {
            selectedIDs.push_back(it->getId());
            //std::cout<<it->getId()<<std::endl;
            j = j - it->getCost();
        }
    }
    if(tsp[0][j] != 0) {
        selectedIDs.push_back(myDefenses.begin()->getId());
    }
}
```

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.