Práctica 2. Programación dinámica

Adrian Reyes Alba
adrian.reyesalba@alum.uca.es
Teléfono: 647243014
NIF: 49074565V

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.

   Mi estrategia ha sido sacar la media de los valores de las caracteristicas de las defensas.

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

   1. Tenemos una matriz donde el tamaño de las filas sería el tamaño de la lista de defensas y las columnas el tamaño de los ases 2. Tenemos también un vector de medias que serán los valores que le asociamos a las defensas 3. Y otro vector de costes, que tiene los costes de las 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 getVectors(std::list<Defense*> defenses, unsigned int* costs, float* media) {
        std::list<Defense*>::iterator itDefenses = defenses.begin();
        int i = 0;

        while(itDefenses != defenses.end()){
                        media[i] = (((*itDefenses)->range +(*itDefenses)->dispersion + (*
                            itDefenses)->damage + (*itDefenses)->attacksPerSecond + (*
                            itDefenses)->health) / 5);
                        costs[i] = (*itDefenses)->cost;
                        ++itDefenses;
                        ++i;
        }

}


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

        std::list<Defense*>::iterator extractor = defenses.begin();
    selectedIDs.push_back((*extractor)->id);
    ases -= (*extractor)->cost;
    defenses.erase(extractor);

        float* media = new float[defenses.size()];
        unsigned int* costs = new unsigned int[defenses.size()];


    float** matriz = new float*[defenses.size()];
    for(int j = 0; j < defenses.size(); ++j)
        matriz[j] = new float[ases];

   getVectors(defenses,costs,media);

   for(unsigned int j = 0; j < ases; j++) {
                if(j < costs[0])
                        matriz[0][j] = 0;
                else
```

```
                        matriz[0][j] = media[0];
        }


        for(unsigned int i = 1; i < defenses.size(); ++i)
                for(unsigned int j = 0; j < ases; ++j){
                        if(j < costs[i])
                                matriz[i][j] = matriz[i-1][j];
                        else
                                matriz[i][j] = std::max(matriz[i-1][j], matriz[i-1][j - costs
                                    [i]] + media[i]);
                }

        int i = defenses.size() - 1;
        unsigned int j = ases-1;
    std::list<Defense*>::iterator it = --defenses.end() ;

    while(i > 0 && j > 0) {

        if(matriz[i][j] != matriz[i-1][j]) {
            selectedIDs.push_back((*it)->id);
            j -= (*it)->cost;
            i--;
            it--;
        }else{
            i--;
            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.

```
        int i = defenses.size() - 1;
        unsigned int j = ases-1;
    std::list<Defense*>::iterator it = --defenses.end() ;

    while(i > 0 && j > 0) {

        if(matriz[i][j] != matriz[i-1][j]) {
            selectedIDs.push_back((*it)->id);
            j -= (*it)->cost;
            i--;
            it--;
        }else{
            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.