Práctica 2. Programación dinámica

Luis José Quintana Bolaño
luisjquintana@gmail.com
Teléfono: 956535843
NIF: 49073584w

7 de diciembre de 2014

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

$$f(damage, attackspersecond, health) = \frac{damage * attackspersecond}{5} + \frac{health}{500}$$

   Para calcular el valor de cada defensa sumamos el daño por segundo que esta causa y su salud, ambos divididos por el valor considerado por defecto en Defense.h

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

   La tabla de subproblemas resueltos es representada mediante una matriz con columnas del 0 al máximo número de ases disponibles, y un número de filas igual al de defensas disponibles.

   La posición de la fila equivale al indice de cada defensa en la lista, y al del valor de esta en el array de valores previamente calculado.

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.

```
/*
 * Function that returns the table of optimal solutions for each subproblem.
 */
std::vector<std::vector<float> > subproblemTable(std::list<Defense*> defenses, std::vector<
    float> values, unsigned int ases) {
    std::vector<std::vector<float> > subTable(defenses.size(),std::vector<float>(ases+1));

    std::list<Defense*>::iterator it = defenses.begin();
    for(int j=0;j<=ases;++j)
        subTable[0][j]=(j<(*it)->cost)? 0:values[0];
    ++it;
    for(int i=1;i<defenses.size();++i){
        for(int j=0;j<=ases;++j){
            subTable[i][j]=(j<(*it)->cost)? subTable[i-1][j]
                                          :std::max(subTable[i-1][j],subTable[i-1][j-(*
                                              it)->cost]+values[i]);
        }
        ++it;
    }

    return subTable;
}
```

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.

```
/*
 * Function that selects the defenses to be placed
 */
void DEF_LIB_EXPORTED selectDefenses(std::list<Defense*> defenses, unsigned int ases, std::
    list<int> &selectedIDs
```

```
            , float mapWidth, float mapHeight, std::list<Object*> obstacles) {

    //The first element always gets selected, and then is removed from the list
    selectedIDs.push_back(defenses.front()->id);
    ases-=defenses.front()->cost;
    defenses.pop_front();
    //The table of subproblems is calculated
    std::vector<float> values(defenses.size());
    values=evaluateDefenses(defenses);
    std::vector<std::vector<float> > subTable(defenses.size(),std::vector<float>(ases+1));
    subTable=subproblemTable(defenses,values,ases);

    //The list of objects included in the optimal solution is retrieved
    std::list<Defense*>::iterator it = defenses.begin();
    for (int i=defenses.size()-1;i>0;--i){
        if(subTable[i][ases]!=subTable[i-1][ases]){
            selectedIDs.push_back((*it)->id);
            ases-=(*it)->cost;
        }
        ++it;
    }
    if(subTable[0][ases]!=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.