PRÁCTICA 4: ALGORITMOS DEVORADORES

Gabriel García Martínez L6

ALGORITMIA

Las mediciones han sido realizadas con un ordenador con las siguientes características:

Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz

8,00 GB (7,87 GB usable)

HEURÍSTICO

El menor de los mayores que 1 de los coeficientes defensa/enemigos. En caso de ser todos menores, el menor de los valores del array de defensores.

COMPLEJIDAD:

La complejidad del algoritmo es $O(n^2)$. Se podría cambiar el diseño para mejorar la complejidad, ordenando ambos arrays y comparando las posiciones homólogas. Si es mayor la de la defensa entonces se asigna, si es menor se asigna a la primera que sea mayor. Para ello, sería necesario una variable auxiliar para saber a cuál se tendrían que asignar las siguientes.

```
public int[] asignar()
{
    // Solución voraz para este problema partiendo de los datos que ya tenemos
    // se pueden crear los mi·todos privados que se consideren necesarios
    for (int i= 0; i < enemigos.length; i++) {  //Recorremos la lista de enemigos
        double coef = 0.0;       //Inicializamos el coeficiente
        int pos = -1;       //Inicializamos la posición
        for (int j = 0; j < grupoDefensaRapida.length;j++) {//Recorremos la lista de defensas
            if(asignados[j] == false) {      //Si ese grupo de defensa no está asignado
                double auxCoef = (grupoDefensaRapida[j]*1.0) / (enemigos[i]*1.0);//Obtenemos nuevo coef
                if(coef >= 1.0) {    //Si el coef es mayor que 1
                    if (auxCoef > 1.0 && auxCoef < coef) {  //Comprobamos si el coef nuevo es mayor
                        coef = auxCoef; //que 1 y si lo es y el coef anterior es mayor que el nuevo
                        pos = j;     // entonces asignamos el nuevo coef y la posición
                    }
                } else {    //Si el coef es menor que 1
                    coef = auxCoef;      //Asignamos el nuevo coef y la posición de esta
                    pos = j;
                }
            }
        }
        if(coef >= 1.0) {    //Tras recorrer el array de defensas, si el coef es mayor que 1,
            asignacion[i] = pos;     //Asignamos en el array solución la pos
            asignados[pos] = true;   //Asignamos el array auxiliar a true
            numVictorias++;          //Aumentamos el número de victorias
        } else {    //Si el coef es menor que 1
            pos = calculaMinimo(this.grupoDefensaRapida);    //Obtenemos el mínimo del array defensas
            asignacion[i] = pos;         //Asignamos en el array solución la pos(el mínimo)
            asignados[pos] = true;       //Asignamos el array auxiliar a true
        }
    }
    return asignacion;
}
```

Como hay dos bucles anidados con complejidad O(n) cada uno, la complejidad del algoritmo es $O(n^2)$.

TIEMPOS:

N	t(μs((10^{-6}s))	Límite
10	0,14	1000000
20	0,45	1000000
40	1,33	1000000

80	5,07	1000000
160	19,81	1000000
320	81,00	1000
640	294,00	1000
1280	1.080,00	1000
2560	4.167,00	1000
5120	16.583,00	1000
10240	66.242,00	1000
20480	306.650,00	100
40960	1.656.600,00	100
81920	31.104.000,00	1
163840	116.260.000,00	1
327680	482.590.000,00	1
655360	2.099.028.000,00	1

Demostración complejidad:

n_1=320 n_2=640 t_1=81 (10^{-6} s)

t_2=$640^2/320^2$ * 81 = 324(10^{-9} s) ≈ 294(10^{-9} s)