PRACTICA 2

Diego Viñals Lage

Carlos Vega Colado

1. Algoritmos Empleados

1.1 Voraz

```
using System;
using System.Collections.Generic;
using System.Text;

namespace Practica2COCO
{
    class VersionVoraz
    {
        public VersionVoraz()
        {

        }

        public float[] mochilavoraz(Objeto[] objetos, int pesomochila)
        {
            float[] x = new float[objetos.Length];
            float peso_acumulado = 0;
            int i = 0;
            while (peso_acumulado < pesomochila && i < objetos.Length)
            {
                if (objetos[i].pesoObjeto() + peso_acumulado < pesomochila)
                {
                    x[i] = 1;
                }
                else
                {
                    x[i] = (pesomochila - peso_acumulado) / objetos[i].pesoObjeto();
                }
                peso_acumulado = (int)(peso_acumulado + (x[i] * objetos[i].pesoObjeto()));
                i++;
            }
            return x;
        }


        public float[] mochilavorazNoDiv(Objeto[] objetos, int pesomochila)
        {
            float[] x = new float[objetos.Length];
            int peso_acumulado = 0;
            int i = 0;
            while (peso_acumulado < pesomochila && i < objetos.Length)
            {
                if (objetos[i].pesoObjeto() + peso_acumulado < pesomochila)
                {
                    x[i] = 1;
                }
                else
                {
                    i = objetos.Length;
                }
                if (i < objetos.Length)
                {
                    peso_acumulado = (int)(peso_acumulado + (x[i] * objetos[i].pesoObjeto()));
                    i++;
                }
            }
            return x;
        }
    }
}
```

1.2 Dinámica

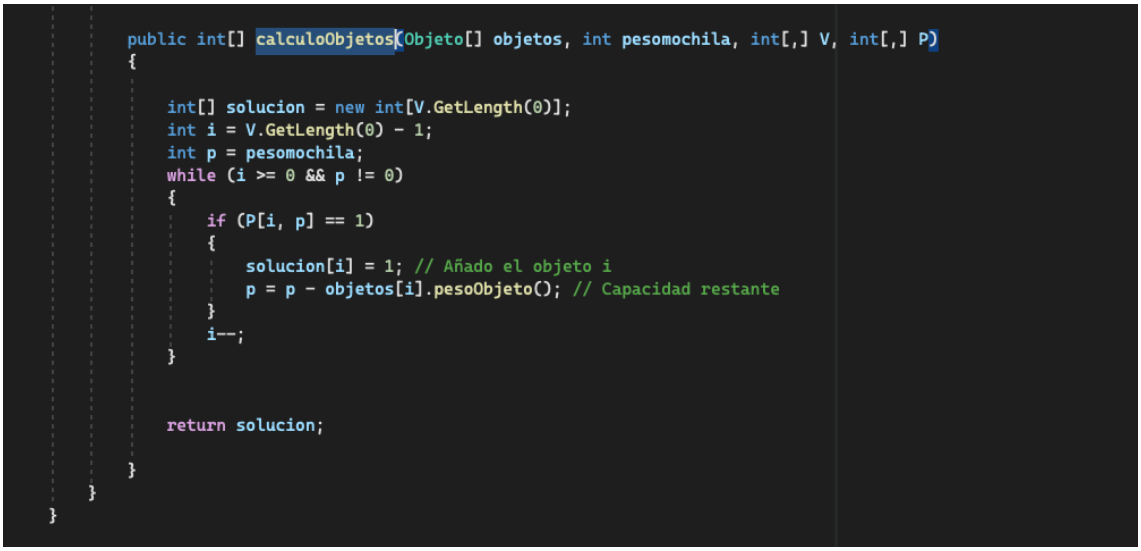
```
using System;
using System.Collections.Generic;
using System.Text;

namespace Practica2COCO
{
    class VersionDinamica
    {
        public VersionDinamica(Objeto[] objetos, int pesomochila, int[,] V, int[,] P)
        {
            //Iniciamos las matrices V y P a 0
            for (int i = 0; i < V.GetLength(0); i++)
            {
                for (int j = 0; j <= pesomochila; j++)
                {
                    V[i, j] = 0;
                    P[i, j] = 0;
                }
            }

            for (int i = 0; i < V.GetLength(0); i++)
            {
                for (int p = 0; p <= pesomochila; p++)
                {
                    if (i == 0 && p < objetos[i].pesoObjeto())
                    {
                        V[i, p] = 0;
                        P[i, p] = 0;
                    }
                    else if (i == 0)
                    {
                        V[i, p] = objetos[i].valorObjeto();
                        P[i, p] = 1;
                    }
                    else if (p < objetos[i].pesoObjeto())
                    {
                        V[i, p] = V[i - 1, p]; // Copiamos la celda superior
                        P[i, p] = 0;
                    }
                    else if (V[i - 1, p] > V[i - 1, p - objetos[i].pesoObjeto()] + objetos[i].valorObjeto())
                    {
                        V[i, p] = V[i - 1, p];
                        P[i, p] = 0;
                    }
                    else
                    {
                        V[i, p] = V[i - 1, p - objetos[i].pesoObjeto()] + objetos[i].valorObjeto();
                        P[i, p] = 1;
                    }
                }
            }
        }


        public int[] calculoObjetos(Objeto[] objetos, int pesomochila, int[,] V, int[,] P)
        {

            int[] solucion = new int[V.GetLength(0)];
            int i = V.GetLength(0) - 1;
            int p = pesomochila;
            while (i >= 0 && p != 0)
            {
                if (P[i, p] == 1)
                {
                    solucion[i] = 1; // Añado el objeto i
                    p = p - objetos[i].pesoObjeto(); // Capacidad restante
```



```
        public int[] calculoObjetos(Objeto[] objetos, int pesomochila, int[,] V, int[,] P)
        {

            int[] solucion = new int[V.GetLength(0)];
            int i = V.GetLength(0) - 1;
            int p = pesomochila;
            while (i >= 0 && p != 0)
            {
                if (P[i, p] == 1)
                {
                    solucion[i] = 1; // Añado el objeto i
                    p = p - objetos[i].pesoObjeto(); // Capacidad restante
                }
                i--;
            }


            return solucion;

        }
    }
}
```

2. Complejidad

La complejidad del algoritmo utilizado con la programación voraz es de n*log(n) que en nuestro caso n = 6

La complejidad del algoritmo utilizado con la programación dinámica es de n*log(n), la misma que la programación voraz, y en este caso también n = 6