UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA CAMPUS ZACATECAS

ANALISIS DE ALGORITMOS

CRUZ LEIJA ROBERTO OSWALDO

ROBERTO COVARRUBIAS LUNA

PROBLEMA DE LA MOCHILA

Introducción:

En algoritmia, el problema de la mochila, comúnmente abreviado por KP (del inglés Knapsack problem) es un problema de optimización combinatoria, es decir, que busca la mejor solución entre un conjunto finito de posibles soluciones a un problema. Modela una situación análoga al llenar una mochila, incapaz de soportar más de un peso determinado, con todo o parte de un conjunto de objetos, cada uno con un peso y valor específicos. Los objetos colocados en la mochila deben maximizar el valor total sin exceder el peso máximo.

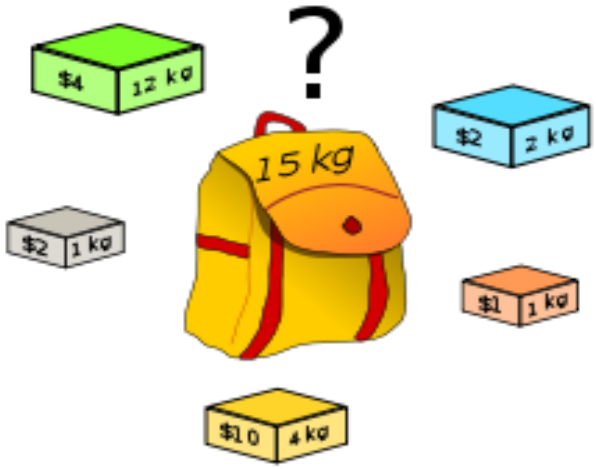
Instrucciones:

1.- No pasar el peso de la mochila

2.- No repetir el mismo objeto

3.- Llevar el máximo peso posible

Ejemplo:



Ejemplo del problema de la mochila: dada una mochila con una capacidad de 15 kg que puedo llenar con cajas de distinto peso y valor, ¿qué cajas elijo de modo de maximizar mis ganancias y no exceder los 15 kg de peso permitidos?

Código:

Clase Mochila:

```
package Mochila;

import java.util.ArrayList;


public class MochilaEnteraDinamica{

    private ArrayList<Articulos> items;
    private ArrayList<Articulos> itemsSolucion;
    private double[][] MaxBen;
    private int W;
    private int maxBenefit;

    public MochilaEnteraDinamica(ArrayList<Articulos> items, int _W) {

        this.items = items;
        this.W = _W;
        construirMatrizBeneficios();

    }

    private void construirMatrizBeneficios() {

        this.MaxBen = new double[this.items.size()+1][this.W+1];

        for (int x=0;x <= this.items.size();x++)
            this.MaxBen[x][0] = 0;


        for (int x=0;x <= this.W;x++)
            this.MaxBen[0][x] = 0;

    }


    public void buscarSolucion(){

       for (int i=1;i <= this.items.size();i++)
           for(int w=0; w<= this.W;w++){

               if  (this.items.get(i-1).getPeso()<= w){
```

```
                if ((this.items.get(i-1).getValor()+
                        this.MaxBen[i-1][w-this.items.get(i-
1).getPeso()])
                        >this.MaxBen[i-1][w]){

                    this.MaxBen[i][w] = this.items.get(i-1).getValor()+
                            this.MaxBen[i-1][w-this.items.get(i-
1).getPeso()];

                }else{

                     this.MaxBen[i][w] = this.MaxBen[i-1][w];

                }

              }else{
              this.MaxBen[i][w] = this.MaxBen[i-1][w];
              System.out.print(this.MaxBen[i][w]+".");
              }
           System.out.println();
           }
       this.maxBenefit = (int)this.MaxBen[items.size()][W];
       this.itemsSolucion = new ArrayList<>();

       int i = this.items.size();
       int j = this.W;

       while (i > 0 && j > 0){
           double val = this.MaxBen[i][j];
          if( val != this.MaxBen[i-1][j]){
              this.itemsSolucion.add(this.items.get(i-1));
              // imprimir el articulo
              String aux =this.items.get(i-1).toString();
              System.out.println(aux);
              i--;
              j = j - this.items.get(i).getPeso();
          } else {
            i--;
          }

       }

        }

    public static void main(String args[]){
```

```
        ArrayList<Articulos> articulos = new ArrayList<>();
        articulos.add(new Articulos(10,1));
        articulos.add(new Articulos(13,2));
        articulos.add(new Articulos(50,7));
        articulos.add(new Articulos(80,5));
        articulos.add(new Articulos(35,3));
        articulos.add(new Articulos(47,1));
        articulos.add(new Articulos(8,2));
        articulos.add(new Articulos(20,8));

        MochilaEnteraDinamica md= new MochilaEnteraDinamica(articulos,5);
        md.buscarSolucion();
    }



}
```

Clase Artículos:

```
package Mochila;


import java.util.ArrayList;
import java.util.Random;


public class Articulos {
    private double valor;
    private int peso;

    public Articulos(double valor, int peso) {
        this.valor = valor;
        this.peso = peso;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
```

```java
    }

    public int getPeso() {
        return peso;
    }

    public void setPeso(int peso) {
        this.peso = peso;
    }

    @Override
    public String toString() {
        String aux ="Tamaño: "     ;
        aux+=this.peso+" <--------> Beneficio: "+this.valor;
        return aux;
    }



    public static ArrayList<Articulos> geraraitems(int n, int v, int p){
        ArrayList<Articulos> items= new ArrayList<>();
        for(int i =0; i<n; i++){
             Random rndp = new Random();
             Random rndv = new Random();
             Articulos it= new Articulos(rndv.nextInt(v)+1,rndp.nextInt(p)+1
);
             items.add(it);
        }
        return items;
    }

}
```

Capturas:

En el ejemplo se ponen los siguientes artículos:

Beneficio	10	13	50	80	35	47	8	20
Peso	1	2	7	5	3	1	2	8

Con una Mochila de peso de 10


```
Tamaño: 1 <--------> Beneficio: 47.0
Tamaño: 3 <--------> Beneficio: 35.0
Tamaño: 5 <--------> Beneficio: 80.0
Tamaño: 1 <--------> Beneficio: 10.0
BUILD SUCCESSFUL (total time: 2 seconds)
```

Beneficio	10	13	50	80	35	47	8	20
Peso	1	2	7	5	3	1	2	8

Con una mochila de peso 5


```
92.0.
Tamaño: 1 <--------> Beneficio: 47.0
Tamaño: 3 <--------> Beneficio: 35.0
Tamaño: 1 <--------> Beneficio: 10.0
BUILD SUCCESSFUL (total time: 1 second)
```