Unidad Profesional Interdisciplinaria de Ingeniería campus Zacatecas

Problema Mochila Dinamico

Carrera: Sistemas Computacionales.

Grupo: 2CM2.

Alumnas: Mónica Yoselin Gallardo Galván

Profesor: Roberto Oswaldo Cruz Leija.

Materia: Análisis de Algoritmos

Planteamiento del problema.

El problema modela una situación análoga al llenar una mochila, incapaz de soportar mas de un peso determinado, con todo o parte de un conjunto de objetos (los cuales llamamos ítems) cada uno con un peso y un valor específico.

Los objetos que coloquemos en la mochila deben maximizar el valor total sin exceder el peso máximo.

Pseudocódigo (Solución).

Se creó la clase llamada ítem para facilitar el manejo de los mismos.

```
public class Item {
    private double valor;
    private int peso;

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

    public double getValor() {
        return valor;
    }

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

    public int getPeso() {
        return peso;
    }

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

    @Override
    public String toString() {
        String aux ="";
        aux+=this.peso+" - "+this.valor;
        return aux; //To change body of genera
    }
```

De igual manera cree la clase mochila para aquí manejar los procesos que se necesitan

```
blic class MochilaDinamico{

  private ArrayList<Item> items;
  private ArrayList<Item> itemsSolucion;
  private double[][] mBeneficios;
  private int _W;
  private int maxBenefit;

  public MochilaDinamico(ArrayList<Item> items, int _W) {
      this.items = items;
      this._W = _W;
      construirMatrizBeneficios();

  }

  public static void main (String args[]){
      int n = 5;//cantidad de items
      int p = 5;//peso maximo
      int v = 30;//valor maximo
      ArrayList<Item> items= new ArrayList<>();
      for(int i=0; i<n; i++){
          Random rndp = new Random();
          Random rndv = new Random();
          Item it= new Item(rndv.nextInt(v)+1,rndp.nextInt(p)+1);
          items.add(it);
      }
     // Herramientas.guardar(items);

      MochilaDinamico m = new MochilaDinamico(items,8);
     // m.construirMatrizBeneficios();
```

Para la solución del problema se crea una matriz de solución donde se irán colocando los valores totales que se van sumando al ingresar ítems a la mochila, esta se construye iniciando en 0:

```
  private void construirMatrizBeneficios() {
      // construimos la matriz de beneficios
      this.mBeneficios = new double[this.items.size()+1][this._W+1];
      // agregar en la primer columna puros ceros
      for (int x=0;x <= this.items.size();x++)
          this.mBeneficios[x][0] = 0;
      // agregar en la primer fila puros ceros

      for (int x=0;x <= this._W;x++)
          this.mBeneficios[0][x] = 0;
```

Y el siguiente es el que resume todo el proceso de solución

```
public void buscarSolucion(){

        // calculamos la matriz de beneficios
  for (int i=1;i <= this.items.size();i++)
      for(int w=0; w<= this._W;w++){
      // verificamos si el item puede ser parte de la solucion
          if  (this.items.get(i-1).getPeso()<= w){

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

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

              }else{

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

              }

          }else{
          this.mBeneficios[i][w] = this.mBeneficios[i-1][w];
          }

      }
  this.maxBenefit = (int)this.mBeneficios[items.size()][_W];
  this.itemsSolucion = new ArrayList<>();
  // calcular los elementos utilizados para _W
```

```
  // calcular los elementos utilizados para _W

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

  while (i > 0 && j > 0){
      double val = this.mBeneficios[i][j];
    if( val != this.mBeneficios[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--;
    }

  }

  System.out.println("Matriz de Beneficios empleada en la respuesta ");
  for(int x=0;x<=this.items.size();x++){
      for(int y=0;y<=this._W;y++){
          System.out.print(this.mBeneficios[x][y]+"   ");
      }
      System.out.println();
   }
  System.out.println("********************************************** ");

  System.out.println();
   }
```

Donde agregue unas líneas para poder visualizar los resultados en la ejecución

Resultados

Cree dos métodos uno para generar en aleatorios ítems y uno mas para poder leer de un archivo ítems asignados por nosotros.

Lo generado aleatoriamente se designaron 20 objetos, con un rango de valor hasta 100 y un rango de peso de 8, la evaluación se hizo un peso máximo de 12. Los resultados fueron los siguientes.

```
Items Creados Aleatoriamente
4 - 5.0
3 - 95.0
6 - 51.0
5 - 71.0
8 - 8.0
5 - 26.0
6 - 69.0
6 - 74.0
8 - 1.0
5 - 66.0
6 - 28.0
7 - 94.0
2 - 10.0
7 - 48.0
2 - 69.0
5 - 1.0
4 - 100.0
5 - 89.0
5 - 69.0
2 - 68.0
```

```
Matriz de Beneficios empleada en la respuesta
0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.0   5.0   5.0   5.0   5.0   5.0   5.0   5.0   5.0   5.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   100.0   100.0   100.0   100.0   100.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   100.0   146.0   146.0   146.0   146.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   166.0   166.0   166.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   166.0   166.0   166.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   166.0   166.0   166.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   166.0   166.0   166.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   169.0   169.0   169.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   169.0   169.0   169.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   169.0   169.0   169.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   169.0   169.0   169.0   171.0
0.0   0.0   0.0   95.0   95.0   95.0   95.0   100.0   166.0   169.0   189.0   189.0   189.0
0.0   0.0   10.0   95.0   95.0   105.0   105.0   105.0   166.0   169.0   189.0   189.0   199.0
0.0   0.0   10.0   95.0   95.0   105.0   105.0   105.0   166.0   169.0   189.0   189.0   199.0
0.0   0.0   69.0   95.0   95.0   164.0   164.0   174.0   174.0   174.0   235.0   238.0   258.0
0.0   0.0   69.0   95.0   95.0   164.0   164.0   174.0   174.0   174.0   235.0   238.0   258.0
0.0   0.0   69.0   95.0   100.0   164.0   169.0   195.0   195.0   264.0   264.0   274.0   274.0
0.0   0.0   69.0   95.0   100.0   164.0   169.0   195.0   195.0   264.0   264.0   274.0   284.0
0.0   0.0   69.0   95.0   100.0   164.0   169.0   195.0   195.0   264.0   264.0   274.0   284.0
0.0   0.0   69.0   95.0   137.0   164.0   169.0   232.0   237.0   264.0   264.0   332.0   332.0
**************************************************
```

```
Items Respuesta
2 - 68.0
4 - 100.0
2 - 69.0
3 - 95.0
```

Obteniendo un peso de 11 y un beneficio total de 332

En el otro caso, utilizamos el ejercicio prueba hecho en el salón, pero ahora poniendo en práctica el método de lectura de archivos, los datos leídos son los siguientes:

34	3
28	6
90	6
23	1
11	9
19	1
700	11

Se considero el peso de la mochila en 8, los resultados fueron los siguientes:

```
Matriz de Beneficios empleada en la respuesta
0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
0.0   0.0   0.0   34.0   34.0   34.0   34.0   34.0   34.0
0.0   0.0   0.0   34.0   34.0   34.0   34.0   34.0   34.0
0.0   0.0   0.0   34.0   34.0   34.0   90.0   90.0   90.0
0.0   23.0   23.0   34.0   57.0   57.0   90.0   113.0   113.0
0.0   23.0   23.0   34.0   57.0   57.0   90.0   113.0   113.0
0.0   23.0   42.0   42.0   57.0   76.0   90.0   113.0   132.0
0.0   23.0   42.0   42.0   57.0   76.0   90.0   113.0   132.0
*****************************************************

Items Respuesta
1 - 19.0
1 - 23.0
6 - 90.0
```

Conclusión.

El problema de la mochila puede ser analizado de diferentes maneras y con esto encontrar otros métodos de solución, no solo el empleado en este trabajo.

La codificación puede darse en distintos lenguajes, no se cierra a uno solo, así que el empleado en este trabajo es elegido por facilidad del investigador.