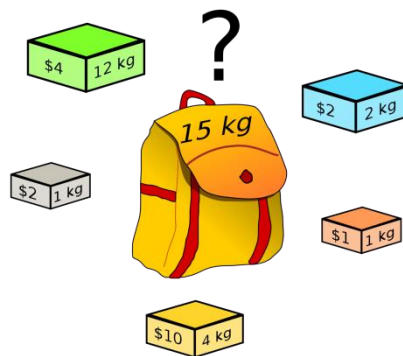




Análisis de Algoritmos

“Mochila”

Solución Mochila (Dinámico)



Docente: Roberto Oswaldo Cruz Leíja.

Alumna: Paulina Medrano Hurtado.

06/11/20

Introducción

El problema de la mochila consiste en un excursionista que debe preparar su mochila, la cual tiene una capacidad limitada y por tanto no le permite llevar todos los artículos que quisiera tener en la excursión. Cada artículo que el excursionista puede incluir en la mochila le reporta una determinada utilidad. Luego el problema consiste en seleccionar un subconjunto de objetos de forma tal que se maximice la utilidad que el excursionista obtiene, pero sin sobrepasar la capacidad de acarrear objetos. Este problema puede ser resuelto de diferentes métodos a implementar pero en esta práctica se pretende dar solución dinámicamente.

Descripción del problema

“Knapsack problem”

Sea n objetos no fraccionables de pesos p_i y beneficios b_i . El peso máximo que puede llevar la mochila es C . Queremos llenar la mochila con objetos, tal que se maximice el beneficio.

El problema de la mochila es un problema clásico en los problemas denominados COP (por sus siglas en inglés Combinatorial Optimization Problem – Problemas de Optimización Combinatoria) de Inteligencia Artificial. Este problema es considerado NP (Non Probabilistic Problem) ya que existe una combinación exponencial de instancias que, en su totalidad, no pueden ser resueltas. Existen variantes relacionadas con este problema: problema con cantidad de productos limitada, problema con cantidad de productos ilimitada, elección múltiple, elección de un producto de diferentes categorías, como un problema relacionado con el peso de los productos, como un problema relacionado con el monto económico, entre otros.

Programación Dinámica

La programación dinámica es un método ascendente. Se resuelven primero los subejemplares más pequeños y por tanto más simples. Combinando las soluciones se obtienen las soluciones de ejemplares sucesivamente más grandes hasta llegar al ejemplar original.

Frecuentemente para resolver un problema complejo se tiende a dividir este en subproblemas, más pequeños, resolver estos últimos (recurriendo posiblemente a nuevas subdivisiones) y combinar las soluciones obtenidas para calcular la solución del problema inicial. Puede ocurrir que la división natural del problema conduzca a un gran número de subejemplares idénticos. Si se resuelve cada uno de ellos sin tener en cuenta las posibles repeticiones, resulta un algoritmo ineficiente; en cambio sí se resuelve cada ejemplar distinto una sola vez y se conserva el resultado, el algoritmo obtenido es mucho mejor.

Esta es la idea de la programación dinámica: no calcular dos veces lo mismo y utilizar normalmente una tabla de resultados que se va rellenando a medida que se resuelven los subejemplares.

Descripción de implementación tomándolo como ejemplo: La solución es usar programación dinámica, vamos a aplicar el enfoque top-down. Lo primero es expresar la solución de forma recursiva.

```
def mochila(n,c):
    if n == 0 or c == 0:
        # solucion óptima para cuando no quedan elementos o la capacidad disponible es 0
        return 0
    elif datos[n].peso > c:
        # no metemos el elemento
        return mochila(n-1,c)
    else:
        #sin meter el elemento
        a = mochila(n-1,c)
        # metiendo el elemento
        b = datos[n].valor + mochila(n-1,c-datos[n].peso)
        return max(a,b)
```

El primer if es la solución óptima, que es trivial si sabemos que ya no quedan elementos por revisar o si la capacidad que queda en la mochila es cero. El elif y else siguientes nos dan las soluciones óptimas también, pero de forma recursiva. En el primer caso, si el elemento es demasiado grande, solo se puede continuar probando con otro elemento, en el otro caso, hacemos los dos cálculos. Miramos si obtenemos más valor metiendo el elemento o sin meterlo y devolvemos el máximo.

Código

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Mochila;

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

/**
 *
 * @author HP
 */
public class Item {
    private double beneficio;
    private int peso;

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

```

        this.peso = peso;
    }

    public double getValor() {
        return beneficio;
    }

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

    public int getPeso() {
        return peso;
    }

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

    @Override
    public String toString() {
        String aux = "";
        aux+=this.peso+"-"+this.beneficio;
        return aux;
    }

    public static ArrayList<Item> geraraitems(int n, int v, int p){
        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);
        }
        return items;
    }
}

```

```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Mochila;

/**
 *
 * @author HP
 */
import java.util.ArrayList;
import java.util.Random;
public class MochilaD {

    private ArrayList<Item> items;
    private ArrayList<Item> itemsSol;
    private double[][] MATBen;
    private int MP;
    private int maxBenefit;

    public MochilaD(ArrayList<Item> items, int MP) {
        this.items = items;
        this.MP = MP;
        construirMatrizBen();
    }

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

    public void buscarSol(){

        // calculamos la matriz de beneficios
        for (int i=1;i <= this.items.size();i++)
            for(int w=0; w<= this.MP;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.MATBen[i-1][w-this.items.get(i-1).getPeso()])
        >this.MATBen[i-1][w]){

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

        }else{

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

        }

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

    System.out.print(this.MATBen[i-1][w]);
    }
    System.out.println("la matriz es:");
    System.out.println();
    this.maxBenefit = (int)this.MATBen[items.size()][MP];
    this.itemsSol= new ArrayList<>();
    // calcular los elementos utilizados para _W

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

    while (i > 0 && j > 0){
        double val = this.MATBen[i][j];
        if( val != this.MATBen[i-1][j]){
            this.itemsSol.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();

}

public static void main(String[]arg){

```

```

int n =8;
int p= 10;
int v=15;

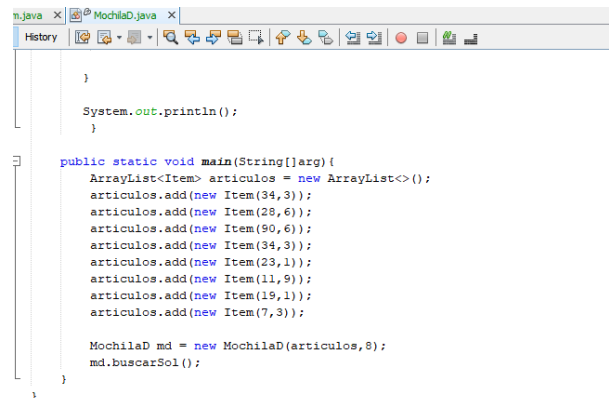
ArrayList<Item> items = new ArrayList<>();
for(int i=0; i<n; i++){
    Random rndp= new Random();
    Random rndv = new Random();
    Item it = new Item(rndv.nextInt(v)+1,
rndp.nextInt(p)+1);
    items.add(it);
}
MochilaD md = new MochilaD(items,8);
md.construirMatrizBen();
md.buscarSol();


//      ArrayList<Item> articulos = new ArrayList<>();
//      articulos.add(new Item(34,3));
//      articulos.add(new Item(28,6));
//      articulos.add(new Item(90,6));
//      articulos.add(new Item(34,3));
//      articulos.add(new Item(23,1));
//      articulos.add(new Item(11,9));
//      articulos.add(new Item(19,1));
//      articulos.add(new Item(7,3));
//
//      MochilaD md = new MochilaD(articulos,8);
//      md.buscarSol();
    }
}

```

Resultados

Utilizando los siguientes ítems que ya habíamos utilizado en clase.



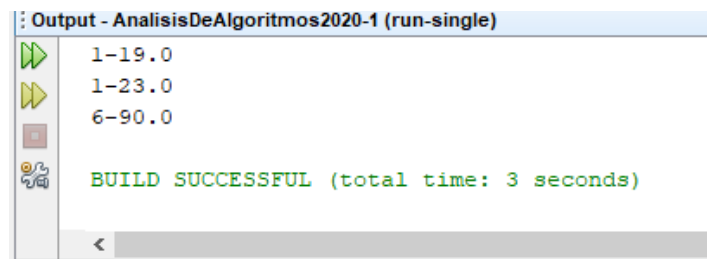
```
    }

    System.out.println();
}

public static void main(String[] arg) {
    ArrayList<Item> articulos = new ArrayList<>();
    articulos.add(new Item(34, 3));
    articulos.add(new Item(28, 6));
    articulos.add(new Item(90, 6));
    articulos.add(new Item(34, 3));
    articulos.add(new Item(23, 1));
    articulos.add(new Item(11, 9));
    articulos.add(new Item(19, 1));
    articulos.add(new Item(7, 3));

    MochilaD md = new MochilaD(articulos, 8);
    md.buscarSol();
}
```

Se obtuvo.

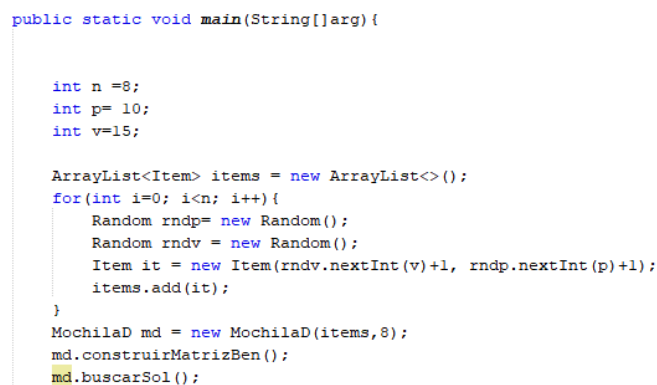


```
Output - AnalisisDeAlgoritmos2020-1 (run-single)

1-19.0
1-23.0
6-90.0

BUILD SUCCESSFUL (total time: 3 seconds)
```

Con ítems aleatorios.



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

    int n = 8;
    int p = 10;
    int v = 15;

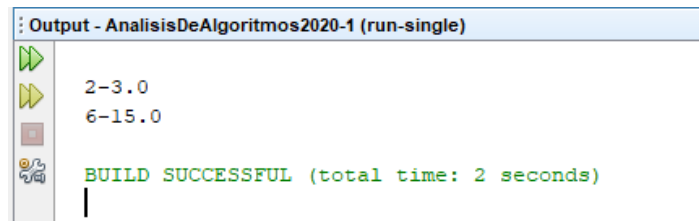
    ArrayList<Item> items = new ArrayList<>();
    for(int i=0; i<n; i++){
        Random rndp = new Random();
        Random rndv = new Random();
        Item it = new Item(rndv.nextInt(v)+1, rndp.nextInt(p)+1);
        items.add(it);
    }
    MochilaD md = new MochilaD(items, 8);
    md.construirMatrizBen();
    md.buscarSol();
}
```

n = items

p = peso

v = valor

Se obtuvo.



```
Output - AnalisisDeAlgoritmos2020-1 (run-single)
2-3.0
6-15.0
BUILD SUCCESSFUL (total time: 2 seconds)
```

Conclusiones

En esta práctica el desarrollo de la solución del TCP dinámico, fue algo retador aunque ya que se sabía el contexto del problema y ya se había dado una solución que fue con fuerza bruta, pero esta vez sería distinto y para poder lograrlo estuve investigando al respecto para tener una idea más clara. A lo que encontré fue información que si me ayudó mucho y así poder implementar esta solución ya presentada. Con el desarrollo de esta práctica adagüe más respecto a la programación dinámica dejando más claro dicho tema.

Implemente la solución son la matriz que antes ya habíamos trabajado con fuerza bruta todo el grupo y los resultados si fueron favorables o bien correctos.

Bibliografía

https://thales.cica.es/rd/Recursos/rd99/ed99-0033-04/din_introd.html

<https://www.uaeh.edu.mx/scige/boletin/tlahuelilpan/n6/e2.html>