Tecnológico de Monterrey

Actividad 1.2 Implementación de la técnica de programación "Programación dinámica" y "Algoritmos Avaros"

Patricio Blanco Rafols - A01642057

31 de Agosto del 2025

Algoritmos Avanzados

Profesor Adolfo Arroyo

Tecnológico de Monterrey Campus GDA

Informe sobre la resolución del Problema de la Mochila con el uso de Algoritmos Avaros y Programación dinámica

Introducción

El problema de la mochila es un tipo de problema con un enfoque de optimización, este se basa en llenar una mochila con objetos los cuales tienen un peso y un valor maximizando el valor total sin exceder la capacidad de carga de la mochila.

Para la resolución de este problema utilizaremos los principios de Programación Dinámica e Algoritmos Avaros en 2 modalidades del problema:

- **Programación Dinámica - Mochila 0/1:** cada objeto se toma completo o no se toma.

- **Algoritmos Avaros - Mochila fraccionaria:** es posible tomar fracciones de un objeto.

Para la implementación de nuestros algoritmos yo escogí la resolución en el lenguaje de Python.

Lógica de los Algoritmos:

Programación Dinámica - Mochila 0/1:

El objetivo es calcular el valor máximo posible usando cada objeto solo una vez. El código usa una lista mejorValor, donde cada posición representa la mejor solución posible para una capacidad determinada.

Cómo funciona:

1. Se crea una lista mejorValor inicializada en ceros con tamaño igual a la CapacidadMax + 1.

2. Para cada objeto: Se revisan las capacidades desde CapacidadMax hasta el peso del objeto (p), en orden descendente.Se actualiza el valor máximo que se puede obtener si se incluye ese objeto.

3. Al final mejorValor[CapacidadMax] contiene el valor máximo total.

Cada posición de mejorValor guarda el mejor valor que puedo obtener con esa capacidad. Al recorrer en orden descendente evitamos usar el mismo objeto más de una vez.

Algoritmos Avaros - Mochila fraccionaria

En esta variante se pueden tomar fracciones de los objetos por lo que la estrategia es ordenar los objetos por su relación valor/peso y llenar la mochila en ese orden.

Cómo funciona:

1. Se calcula el valor por unidad de peso (valor / peso) de cada objeto.
2. Se ordenan los objetos de mayor a menor relación.
3. Se llena la mochila:

   Si el objeto completo cabe se toma completo.

   Si no cabe se toma la fracción que quepa.
4. Se suma el valor correspondiente al total.

El objeto más rentable por unidad siempre deberá entrar primero. Esta técnica asegura el máximo valor posible de forma rápida.

Resultados:

Programación Dinámica - Mochila 0/1

Inputs:

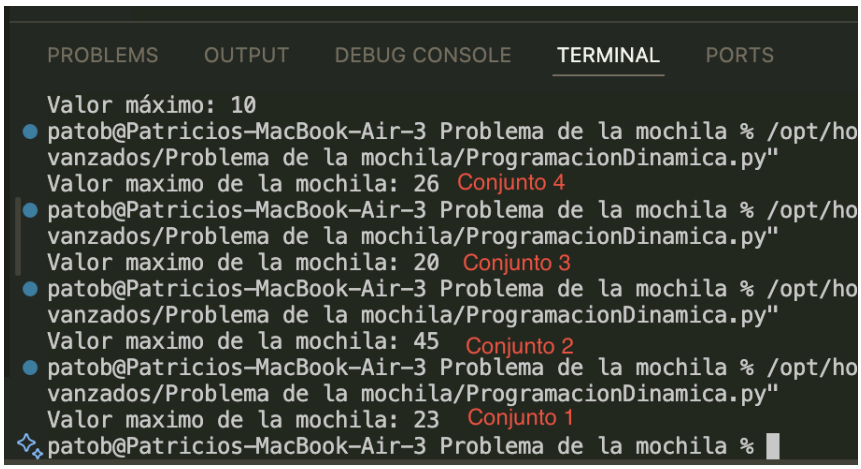
```python
# Conjunto 1
peso = [4, 5, 2, 8, 1]
valor = [12, 9, 3, 16, 2]
CapacidadMax = 10

# Conjunto 2
peso = [1, 3, 5, 4, 7]
valor = [5, 10, 15, 7, 20]
CapacidadMax = 15

# Conjunto 3
peso = [6, 2, 3, 5, 4]
valor = [8, 3, 5, 9, 6]
CapacidadMax = 12

# Conjunto 4 (más elementos)
peso = [3, 2, 5, 7, 1, 4, 6]
valor = [5, 3, 8, 12, 2, 7, 10]
CapacidadMax = 15
```

Outputs:



Algoritmos Avaros - Mochila fraccionaria

Inputs:

```
pesos1 = [10,20, 30, 40, 50]
valores1 = [60, 100,120, 140,150]
capacidad1 = 100

pesos2 = [2, 5, 10,3, 8]
valores2 = [20, 30, 50, 40,45]
capacidad2 = 15

pesos3 = [8, 3, 5, 2, 4, 9,1]
valores3 = [16, 9, 20,6, 10,24, 5]
capacidad3 = 20
```

Output:

```
patob@Patricios-MacBook-Air-3 Probl
Users/patob/Downloads/Algoritmos Av
py"
Resultado conjunto 1: 420
Resultado conjunto 2: 118.125
Resultado conjunto 3: 64
patob@Patricios-MacBook-Air-3 Probl
```

Conclusión:

Para finalizar con esta actividad me gustaría recalcar un gran aprendizaje que genere en la resolución de esta actividad la cual es la importancia de implementar el algoritmo correcto en la situación adecuada. En la resolución del problema de la Mochila podemos ver como una variante del mismo problema puede ser optimizada con algoritmos diferentes:

- La programación dinámica es ideal para la versión 0/1 porque explora todas las combinaciones posibles de forma eficiente y garantiza el valor máximo.

- El algoritmo codicioso en cambio es más ligero y rápido y es perfecto para la versión fraccionaria porque su lógica de priorizar el valor por unidad de peso siempre genera el valor máximo.

Ambos métodos de resolución muestran que una buena estrategia que simplifica problemas complejos y permite soluciones eficientes y confiables.