Actividad 1.2

Implementación de la técnica de programación "Programación dinámica" y "algoritmos avaros" Bernardo Santiago Marín A01638915

Enfoque del Código

El código implementa una solución al problema de la mochila utilizando programación dinámica. Este problema consiste en seleccionar un subconjunto de elementos, cada uno con un peso y un valor, de manera que se maximice el valor total sin exceder un peso máximo permitido.

El enfoque utiliza una tabla bidimensional (`table`) para almacenar los valores máximos que se pueden obtener con un peso límite dado y un conjunto de elementos. La tabla se llena iterativamente considerando cada elemento y cada capacidad posible de la mochila.

Algoritmo Implementado

Algoritmo original hecho por William Fiset en YouTube. Consiste en crear una tabla donde las columnas representen la capacidad (desde 0 hasta `W`) y las filas representen el valor y peso de cada uno de los objetos, de manera que se guarda en cada celda de la tabla el valor máximo que se puede obtener con dicho objeto en la fila.

1. Inicialización de la Tabla:
   - Se crea una tabla `table` de tamaño `(n+1) x (W+1)` , donde `n` es el número de elementos y `W` es la capacidad máxima de la mochila. Decidí tomar este tamaño para poder recorrerlo hacia atrás sin riesgo de salirnos del rango.
   - Cada celda `table[i][j]` representa el valor máximo que se puede obtener con los primeros `i` elementos y una capacidad de `j` .
2. Llenado de la Tabla:
   - Para cada elemento `i` y cada capacidad `cap` :
     - Si no se incluye el elemento `i` , el valor es el mismo que el de la fila anterior (`table[i-1][cap]`).
     - Si se incluye el elemento `i` , se suma su valor al mejor resultado posible con el peso restante (`table[i-1][cap - currentWeight]`).
     - Se toma el máximo de estas dos opciones.
3. Backtracking:
   - Una vez completada la tabla, se realiza un seguimiento hacia atrás para determinar qué elementos fueron seleccionados.
   - Si el valor en `table[i][remainingWeight]` es diferente al de `table[i-1][remainingWeight]` , significa que el elemento i-1 fue incluido. Es decir, si entre una columna hay un cambio de valor entre una fila a otra, es porque se incluyó el elemento de la última fila.
4. Resultado:
   - Se devuelve una lista con los índices de los elementos seleccionados.

Ejemplos con Entradas y Salidas

Ejemplo 1

- Entrada:

```
W = 4;
weights = {1, 2, 3};
values = {1, 2, 3};
```

- Salida:

```
Selected items: [2, 0]
Total value: 4
```

- Explicación: Se seleccionan los elementos con índices 2 y 0 (pesos 3 y 1, valores 3 y 1), maximizando el valor total sin exceder el peso máximo.

Ejemplo 2

- Entrada:

```
W = 5
weights = {2, 3, 4, 5}
values = {3, 4, 5, 6}
```

- Salida:

```
Selected items: [1, 0]
Total value: 7
```

- Explicación: Se seleccionan los elementos con índices 1 y 0 (pesos 3 y 2, valores 4 y 3), maximizando el valor total.

Ejemplo 3

- Entrada:

```
W = 10
weights = {1, 4, 8}
values = {10, 40, 50}
```

- Salida:

```
Selected items: [2, 0]
Total value: 60
```

- Explicación: Se seleccionan los elementos con índices 2 y 0 (pesos 8 y 1, valores 50 y 10), maximizando el valor total.

Ejemplo 4

- Entrada:

```
W = 7
weights = {3, 4, 5}
values = {30, 40, 50}
```

- Salida:

```
Selected items: [1]
Total value: 40
```

- Explicación: Se selecciona el elemento con índice 1 (peso 4, valor 40), ya que no es posible incluir más elementos sin exceder el peso máximo.

Resultados Obtenidos

El algoritmo es eficiente para resolver el problema de la mochila con entradas pequeñas y medianas. Las complejidades temporal y espacial son ambas $O(n \cdot W)$, donde (`n`) es el número de elementos y (`W`) es la capacidad máxima de la mochila. Esto lo hace adecuado para problemas donde (`n`) y (`W`) no son excesivamente grandes. Esto es porque, en la tabla dinámica, se recorre `n` veces los pesos `W`.

Conclusión

Me gustó mucho implementar este código para el problema de la mochila con programación dinámica. La tabla es útil para calcular de manera eficiente el valor máximo posible, y con el uso del backtracking podemos identificar los elementos seleccionados. Aunque me costó trabajo entender el método el principio, creo que con un poco más de práctica podré identificar estos problemas más facilmente. Ahora que conozco como puedo usar tablas dinámicas para guardar resultados previos, puedo verlo como una herramienta más al momento de resolver problemas de este tipo.