PD Claves

Conceptos clave:

1. **Solapamiento de subproblemas**: El problema se puede descomponer en subproblemas más pequeños que se resuelven de manera repetitiva. La programación dinámica aprovecha esto almacenando las soluciones a los subproblemas para evitar recalcularlas.
2. **Optimalidad de subestructuras**: La solución óptima de un problema se puede construir a partir de las soluciones óptimas de sus subproblemas.
3. **Memoización**: Consiste en almacenar los resultados de sub problemas resueltos para no volver a calcularlos. Esto se suele hacer utilizando una estructura de datos como una tabla o matriz.
4. **Tabulación**: Se resuelven los subproblemas de manera ascendente, construyendo las soluciones de los subproblemas más pequeños primero y utilizándose para resolver problemas más grandes. Es un enfoque iterativo, en lugar de recursivo.

Guia para armar ecuaciones de recurrencia:

Para armar una ecuación de recurrencia en programación dinámica, el enfoque se centra en descomponer el problema en subproblemas más pequeños y relacionar la solución de cada subproblema con las soluciones de subproblemas previos. A continuación te proporcionaré una guía para formular una ecuación de recurrencia, tomando como base el problema de la mochila.

Guía para formular una ecuación de recurrencia:

Paso 1: Entender el problema y los subproblemas

El **problema de la mochila** consiste en seleccionar elementos, cada uno con un valor y un peso, de manera que se maximice el valor total de los elementos seleccionados sin exceder un peso máximo W.

Cada subproblema es: ¿Cuál es el valor máximo que se puede obtener considerando los primeros i elementos y un peso máximo de w?

Paso 2: Definir el estado y la función de recurrencia

En este caso, podemos definir la función **OPT(i, w)** como el valor máximo que podemos obtener utilizando los primeros i elementos con un peso máximo de www.

La idea es construir la solución para el subproblema i utilizando:

- La solución sin incluir el elemento i (mantenemos el valor obtenido con los i−1 elementos).
- La solución incluyendo el elemento i, si es que el peso del elemento i es menor o igual a w.

Paso 3: Definir la decisión en cada paso

Para cada elemento i tienes dos opciones:

1. **No incluir el elemento i**: En este caso, el valor máximo que podemos obtener es el mismo que si solo consideramos los primeros i−1 elementos y el mismo peso w. OPT(i,w)=OPT(i−1,w)
2. **Incluir el elemento i**: Si el peso del elemento i es menor o igual a w, podemos considerar el valor de incluirlo. En este caso, obtenemos el valor del elemento i, más el valor óptimo de los primeros i−1 elementos con el peso restante w−peso[i]

   OPT(i,w)=max(OPT(i−1,w),valor[i]+OPT(i−1,w−peso[i]))

Paso 4: Definir los casos base

Los casos base definen las condiciones más simples del problema:

- Si no consideramos ningún elemento (i=0), el valor máximo es 0 independientemente del peso:

  OPT(0,w)=0 para cualquier w

- Si el peso máximo es 0 (w=0), el valor máximo es 0 independientemente del número de elementos:

  OPT(i,0)=0 para cualquier i

Paso 5: Implementación en el código

Ahora, basándonos en estos principios, la implementación de la ecuación de recurrencia en tu código queda clara.

Recapitulación de la ecuación de recurrencia

La **ecuación de recurrencia** para el problema de la mochila es:

$$OPT(i,w)=\begin{cases} 0 & \text{si } i=0 \text{ o } w=0 \\ OPT(i-1,w) & \text{si } peso\_i > w \\ \max(OPT(i-1, w), valor\_i + OPT(i-1, w-peso\_i)) & \text{si } peso\_i \le w \end{cases}$$

Esto te permite calcular el valor máximo que se puede obtener con un conjunto de elementos y un peso límite.