Implementar un algoritmo de optimización.

Elaborado por:
Lcdo. Diego Medardo Saavedra García, Mgtr.
Profesor Ocasional de la Universidad de las Fuerzas Armadas ESPE
Departamento de Ciencias de la Computación.

Ejemplo práctico: Problema de la mochila (Knapsack Problem)

Introducción: El problema de la mochila es un clásico problema de optimización que consiste en seleccionar un subconjunto de elementos de manera que se maximice el valor total, sin exceder una capacidad dada. En este ejemplo, implementaremos un algoritmo de optimización para resolver el problema de la mochila.

Objetivo General: Diseñar e implementar un algoritmo de optimización que resuelva el problema de la mochila, maximizando el valor total de los elementos seleccionados sin exceder la capacidad máxima de la mochila.

Objetivos Específicos:

1. Definir el problema de la mochila y sus restricciones.
2. Diseñar un algoritmo de optimización utilizando la técnica de programación dinámica.
3. Implementar el algoritmo en el lenguaje de programación Python.
4. Realizar pruebas exhaustivas para verificar la eficacia y eficiencia del algoritmo.

Marco Teórico: El problema de la mochila se puede formular de la siguiente manera: se tiene una mochila con una capacidad máxima y una serie de elementos, cada uno con un valor y un peso asociado. El objetivo es seleccionar los elementos de manera que se maximice el valor total sin exceder la capacidad de la mochila.

Propuesta de Valor: El algoritmo de optimización de la mochila es ampliamente utilizado en la industria y la investigación, ya que tiene aplicaciones en diversos campos, como la logística, la planificación de recursos y la gestión de inventarios.

Equipo Técnico: Para este proyecto, se requiere un equipo de desarrollo compuesto por programadores con conocimientos en algoritmos y programación dinámica.

Recursos:

- Lenguaje de programación: Python
- IDE o editor de código: Recomendado utilizar PyCharm, Visual Studio Code u otro de tu preferencia.

- Conjunto de datos de prueba: Selecciona un conjunto de elementos con sus respectivos valores y pesos.

Tiempo de Desarrollo: El tiempo estimado para el desarrollo de este proyecto es de 2 meses.

Herramientas:

- Sistema de gestión de versiones: Git
- Repositorio en línea: GitHub

Explicación paso a paso:

1. Definición del problema: Selecciona un conjunto de elementos con sus respectivos valores y pesos. Establece la capacidad máxima de la mochila.

Ejemplo: Conjunto de elementos:

- Elemento 1: Valor = 60, Peso = 10
- Elemento 2: Valor = 100, Peso = 20
- Elemento 3: Valor = 120, Peso = 30

Capacidad máxima de la mochila: 50

2. Diseño del algoritmo de optimización: Utilizaremos la técnica de programación dinámica para resolver el problema de la mochila. El algoritmo consistirá en construir una matriz de tamaño (n+1) x (W+1), donde n es el número de elementos y W es la capacidad máxima de la mochila. Cada celda de la matriz representará el valor máximo alcanzado para una capacidad y un número determinado de elementos.
3. Implementación del algoritmo en Python:

python
```
def knapsack(items, capacity):
    n = len(items)
    matrix = [[0] * (capacity + 1) for _ in range(n + 1)]

    for i in range(1, n + 1):
        value, weight = items[i - 1]
        for w in range(1, capacity + 1):
            if weight <= w:
                matrix[i][w] = max(matrix[i - 1][w], matrix[i - 1][w - weight] + value)
            else:
                matrix[i][w] = matrix[i - 1][w]

    return matrix[n][capacity]

# Ejemplo de uso
```

```
items = [(60, 10), (100, 20), (120, 30)]
capacity = 50

max_value = knapsack(items, capacity)
print("Valor máximo obtenido:", max_value)
```

4. Pruebas y depuración: Verifica que el algoritmo produce los resultados esperados. Realiza pruebas utilizando diferentes conjuntos de elementos y capacidades máximas de la mochila.
5. Optimización: Analiza el rendimiento del algoritmo y busca posibles mejoras para aumentar su eficiencia, como la implementación de técnicas de poda o la utilización de estructuras de datos más eficientes.

Entregables:

- Código fuente del algoritmo implementado en Python.
- Documentación que explique el funcionamiento del algoritmo, los resultados obtenidos y cualquier mejora realizada.

Recuerda utilizar un sistema de control de versiones como Git y almacenar el código y la documentación en un repositorio de GitHub para facilitar la colaboración y la gestión del proyecto en un entorno de trabajo en equipo.