Explicación de Algoritmo Voraz orientado a Problema de la mochila

José Luis Sandoval Pérez

Teoría de la complejidad Computacional

Código

```
import random


def knapsack_greedy(values, weights, capacity):
    n = len(values)
    items = list(range(n))
    random.shuffle(items)  # Mezcla aleatoriamente los índices de los objetos

    total_value = 0
    total_weight = 0
    knapsack = []

    for index in items:
        if total_weight + weights[index] <= capacity:
            total_value += values[index]
            total_weight += weights[index]
            knapsack.append(index)
            print(f"Objeto {index}: Valor = {values[index]}, Peso = {weights[index]}")
        if total_weight == capacity:
            break

    return total_value, knapsack
```

```
# Generar valores y pesos aleatorios para los objetos
num_objects = 20
values = [random.randint(10, 100) for _ in range(num_objects)]
weights = [random.randint(5, 30) for _ in range(num_objects)]
capacity = 100

total_value, selected_items = knapsack_greedy(values, weights,
capacity)
print("\nValor total en la mochila:", total_value)
print("Objetos seleccionados:", selected_items)
```

1. Indicar restricciones de entrada para el algoritmo
    a. Se eligen los objetos con mayor valor, sin importar el peso de tienen
    b. Se seleccionan objetos en una lista de 20 generada de manera aleatoria.
2. Explicación como funciona:
   Inicialización:
   Se establece el número de objetos (num_objects) y se generan valores y pesos aleatorios para los objetos.
   Función knapsack_greedy:
   La función knapsack_greedy toma tres parámetros: values, weights y capacity, que representan los valores, los pesos y la capacidad de la mochila, respectivamente.
   Se crea una lista de índices de objetos items, que son mezclados aleatoriamente utilizando random.shuffle(). Esto introduce aleatoriedad en el orden de selección de los objetos.
   Se inicializan las variables total_value y total_weight para realizar el seguimiento del valor total y el peso total de los objetos seleccionados.

Se itera sobre los índices de objetos aleatorios:

- Se verifica si agregar el objeto a la mochila no excede la capacidad máxima. Si es así, se agrega el objeto a la mochila, y se incrementan total_value y total_weight.
- Se imprime el valor y el peso del objeto seleccionado.
- Si se alcanza la capacidad máxima de la mochila, se detiene la iteración.
- Se devuelve el valor total y los índices de los objetos seleccionados.

Impresión de Resultados:

Se imprime el valor total en la mochila y los índices de los objetos seleccionados.

Recordemos que el algoritmo voraz toma decisiones en cada paso basándose en una regla de elección local óptima, con la esperanza de llegar a una solución global óptima. En este caso, el algoritmo selecciona los objetos en un orden aleatorio y los agrega a la mochila si no exceden la capacidad máxima, priorizando los objetos de mayor valor. Esto se hace iterativamente hasta que la capacidad máxima de la mochila se alcanza. El algoritmo no considera las consecuencias a largo plazo de cada decisión, sino que solo se enfoca en tomar la mejor decisión en cada paso individual.

Resultados:

```
Objeto 17: Valor = 23, Peso = 23
Objeto 9: Valor = 39, Peso = 20
Objeto 3: Valor = 62, Peso = 13
Objeto 10: Valor = 70, Peso = 27
Objeto 2: Valor = 59, Peso = 6
Objeto 5: Valor = 92, Peso = 10

Valor total en la mochila: 345
Objetos seleccionados: [17, 9, 3, 10, 2, 5]
```

Podemos ver que se seleccionan los objetos con mayor valor, sin importar el peso, dándonos como solución un valor total en la mochila de 345. Sin exceder el peso total de 100.

```
Objeto 0: Valor = 76, Peso = 13
Objeto 6: Valor = 14, Peso = 8
Objeto 14: Valor = 72, Peso = 25
Objeto 18: Valor = 78, Peso = 28
Objeto 19: Valor = 80, Peso = 20
Objeto 11: Valor = 52, Peso = 6

Valor total en la mochila: 372
Objetos seleccionados: [0, 6, 14, 18, 19, 11]
```

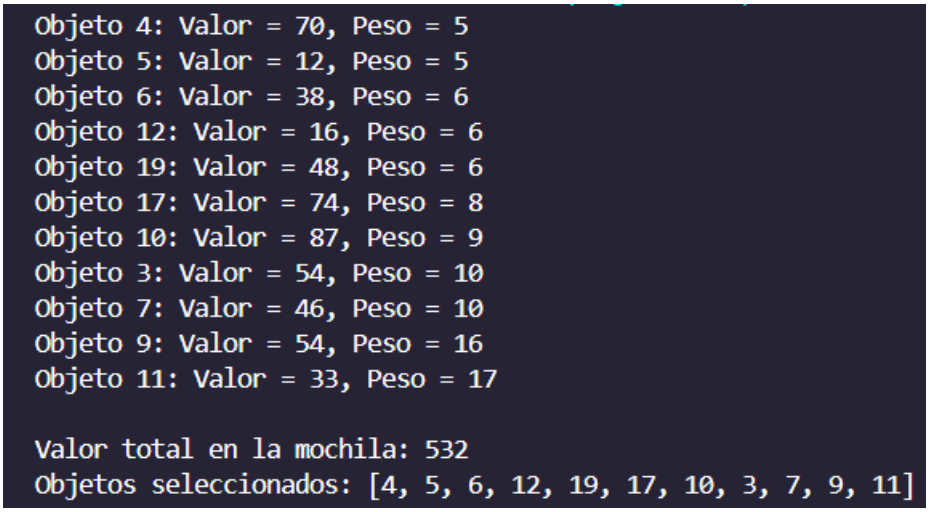
Ejecución adicional

```
Objeto 5: Valor = 81, Peso = 24
Objeto 11: Valor = 19, Peso = 7
Objeto 13: Valor = 51, Peso = 13
Objeto 0: Valor = 72, Peso = 24
Objeto 1: Valor = 95, Peso = 6
Objeto 12: Valor = 49, Peso = 14
Objeto 18: Valor = 78, Peso = 8

Valor total en la mochila: 445
Objetos seleccionados: [5, 11, 13, 0, 1, 12, 18]
```

Ejecución adicional

Ahora se adjuntan soluciones pero con una restricción diferente, la cual es seleccionar los objetos de menor peso.



Podemos notar que el valor total de la mochila aumenta y es mayor en comparación de seleccionar siempre el de mayor valor. Claro esto no siempre es así debido a que los números que se generan son aleatorios y el peso y valor cambia en cada ejecución de código.