Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala

“Algoritmo Voraz para Optimización de Envíos de Café”

Análisis y Diseño de Algoritmos

Dr. Esaú Eliezer Escobar Juárez

Integrantes:

Francisco Rodríguez campo

Rodrigo Paredes Urrusquieta

Leonel Carrillo Gutiérrez

Fecha

04/06/2024

Abstract

Esta práctica explora el diseño e implementación de un algoritmo voraz para optimizar los envíos de café de una compañía. El objetivo es maximizar el valor del café enviado dentro de la capacidad limitada de la camioneta de reparto. El problema se modeló como una variante del problema de la mochila fraccionaria y se evaluaron tres criterios de selección: peso, costo total y costo por kilo. A través del análisis, se demuestra que el criterio de costo por kilo produce el mejor resultado. Se propone un algoritmo voraz utilizando este criterio, que garantiza la maximización del valor del envío. El algoritmo se implementa con una cola de prioridades para seleccionar el siguiente mejor valor, logrando una complejidad de O(n log n). Los resultados confirman la efectividad del enfoque, asegurando soluciones óptimas para las entradas dadas.

Palabras clave: algoritmo voraz, optimización, mochila fraccionaria, costo por kilo, cola de prioridades.

Introducción

La optimización de envíos es un problema crucial en la logística de compañías de distribución. Este problema puede ser modelado como una variante del problema de la mochila fraccionaria, donde el objetivo es maximizar el valor del café enviado dentro de la capacidad limitada de una camioneta. Se evaluarán tres criterios de selección: peso, costo total y costo por kilo para determinar cuál produce el mejor resultado.

Marco Teórico

El problema de la mochila fraccionaria es una variación del clásico problema de la mochila, donde se permite tomar fracciones de los objetos. En este contexto, el objetivo es maximizar el valor total de los objetos seleccionados sin exceder la capacidad de la mochila (o camioneta en este caso). Un algoritmo voraz es adecuado para este tipo de problemas, ya que toma decisiones locales óptimas con la esperanza de encontrar una solución global óptima.

Solución del problema

1. Determinación del criterio de selección que se usará vorazmente para producir una solución óptima.

a) Si se utiliza el peso como criterio, se seleccionaría café en función de la cantidad máxima de peso disponible en la camioneta. En este caso, las cantidades de cada tipo de café seleccionadas serían:

- T5: 50 kg
- T4: 40 kg
- T3: 10 kg
- T2: 0 kg
- T1: 0 kg

El precio total de venta del envío resultante sería 122.

b) Si se utiliza el costo como criterio, se seleccionaría café en función del costo total (peso * costo por kilo) de cada tipo de café. Las cantidades seleccionadas serían:

- T3: 30 kg
- T5: 50 kg
- T4: 20 kg
- T2: 0 kg
- T1: 0 kg

El precio total de venta del envío resultante sería 146.

c) Si se utiliza el costo por kilo como criterio, se seleccionaría café en función del costo por kilo de cada tipo de café. Las cantidades seleccionadas serían:

- T3: 30 kg
- T1: 10 kg
- T2: 20 kg

- T5: 40 kg
- T4: 0 kg

El precio total de venta del envío resultante sería 164.

d) El criterio de selección que produce el mejor resultado es el costo por kilo. En este caso, se obtiene el mayor precio total de venta del envío resultante, que es 164.

2. Pseudocódigo

```
función ordenar_envios_cafe(W, n, weight, costo_por_kilo):
    // Crear un arreglo para almacenar la cantidad de cada tipo de café a enviar
    ship = [0] * n

    // Crear una cola de prioridades para almacenar la relación costo/peso
    cola_prioridad = cola_prioridades()

    // Insertar cada tipo de café en la cola de prioridades con su costo/peso
    para i en rango(n):
        cola_prioridad.insertar((costo_por_kilo[i] / weight[i], weight[i], i))

    valor_total = 0
    capacidad_restante = W

    // Mientras haya capacidad restante y la cola de prioridades no esté vacía
    mientras capacidad_restante > 0 y no cola_prioridad.vacía():
        // Extraer el tipo de café con mayor relación costo/peso
        (ratio, peso, indice) = cola_prioridad.extraer_max()

        si peso <= capacidad_restante:
            // Tomar todo el café de este tipo
            ship[indice] = peso
            valor_total += peso * costo_por_kilo[indice]
```

```
        capacidad_restante -= peso
    sino:
        // Tomar una fracción del café de este tipo que cabe en la capacidad restante
        ship[indice] = capacidad_restante
        valor_total += capacidad_restante * costo_por_kilo[indice]
        capacidad_restante = 0

devolver ship, valor_total
```

3. Verificación del algoritmo

El criterio de costo por kilo es óptimo porque asegura que siempre seleccionamos el tipo de café que ofrece el mayor valor por unidad de peso, maximizando así el valor total del envío en cada paso del proceso. Esto se basa en el principio voraz de hacer la elección localmente óptima en cada paso con la esperanza de que esto lleve a una solución globalmente óptima.

Por contraejemplo podemos argumentar que el algoritmo seleccione inicialmente el tipo de café con el mayor valor por unidad de peso, podría existir otro tipo de café en la lista de prioridades que, al sustituir a uno de los tipos de café seleccionados, mejore el valor total de la carga. Sin embargo, este nuevo tipo de café tendría un valor por unidad de peso menor que el del café sustituido en la lista. Como resultado, la nueva carga total tendría un valor total menor. Por lo tanto, se garantiza que el algoritmo proporciona una solución óptima.

4. Complejidad del algoritmo

El tiempo de ejecución del algoritmo depende principalmente de dos factores:

1. La creación e inserción de elementos en la cola de prioridades.

2. El bucle principal que recorre la cola de prioridades y realiza operaciones de extracción y asignación.

- La creación e inserción de elementos en la cola de prioridades tiene una complejidad de **O(n log n)** debido a las operaciones de inserción para cada uno de los n elementos.
- El bucle principal que recorre la cola de prioridades y realiza operaciones de extracción y asignación también tiene una complejidad de **O(n log n),** ya que en cada iteración se realiza una operación de extracción de la cola de prioridades, que tiene una complejidad de **O(log n)**, y el bucle se ejecuta n veces en el peor de los casos.

Por lo tanto, la función detallada de tiempo de ejecución **T(n**) para el algoritmo es **T(n) = O(n log n).**

Para la función g(n) tal que **T(n) = O(g(n))**, podemos elegir **g(n) = n log n**. Esto se debe a que la complejidad del algoritmo es **O(n log n)**, por lo que **g(n) = n log n** es una cota superior asintótica válida para **T(n)**.

Conclusión

El desarrollo del algoritmo voraz para la optimización de envíos de café ha demostrado claramente que el criterio de costo por kilo es el más eficiente y efectivo. Este enfoque maximiza el valor del café enviado dentro de la capacidad limitada de la camioneta, logrando un costo total de 164 unidades monetarias, el más alto en comparación con los otros criterios evaluados.

El criterio de peso, que selecciona los tipos de café en función de la cantidad máxima de peso, resultó ser menos eficiente, alcanzando un costo total de solo 120 unidades monetarias. Por otro lado, el criterio de costo total, que selecciona en función del valor absoluto del café, mejoró el resultado, obteniendo un costo total de 146 unidades monetarias. Sin embargo, ninguno de estos enfoques superó al criterio de costo por kilo.

El uso de una cola de prioridades en el algoritmo asegura que se seleccione el tipo de café con el mayor valor por unidad de peso en cada paso, garantizando una solución óptima de manera eficiente. La complejidad del algoritmo, O(n log n)

El estudio de este caso específico nos ha permitido comprender cómo estructurar problemas para que sean susceptibles de ser resueltos mediante un enfoque voraz. Identificar el criterio de selección adecuado es fundamental para garantizar que las decisiones locales conduzcan a un resultado óptimo global.

Con este conocimiento, estamos mejor preparados para enfrentar problemas similares en el futuro, ya sea en logística, finanzas u otros campos que requieran optimización bajo restricciones. El enfoque voraz proporciona una herramienta poderosa para desarrollar soluciones eficientes, y la habilidad para identificar y aplicar el criterio de selección correcto es crucial para aprovechar al máximo este método de programación.

Referencias

GeeksforGeeks. (2023, 3 abril). *Fractional knapsack problem*. GeeksforGeeks. https://www.geeksforgeeks.org/fractional-knapsack-problem/?ref=gcse

CSBreakdown. (2015, 14 abril). *The Fractional Knapsack - greedy algorithms* [Vídeo]. YouTube. https://www.youtube.com/watch?v=kFUs5VUxO-s

GeeksforGeeks. (2020, 19 julio). *C Program for the Fractional Knapsack Problem*. GeeksforGeeks. https://www.geeksforgeeks.org/c-program-for-the-fractional-knapsack-problem/