Universidad Tecnológica Metropolitana

Departamento de Informática y Computación

Análisis de Algoritmos

Tarea 1: Subproblemas y un caso de prueba

Vicente Navarro, Benjamin Gomez y Joaquin Troncoso

10 de noviembre de 2025

1. Descripción del Problema

El problema consiste en un juego de suma cero entre un Profesor y su hermana, quienes se reparten una torta circular dividida en $2n$ porciones. Cada porción i tiene un valor de satisfacción asociado st_i.

El juego comienza cuando el Profesor selecciona un ángulo de corte α_i, tomando para sí las n porciones contiguas en sentido antihorario hasta el ángulo $\alpha_i + \pi$. A partir de ese momento, ambos jugadores alternan turnos seleccionando porciones de los extremos del segmento restante de la torta.

El objetivo es encontrar el ángulo α_i inicial que maximice la ganancia total del Profesor, asumiendo que su hermana juega de manera óptima para minimizar dicha ganancia.

2. Estrategia de Solución

Para resolver este problema de optimización, se utilizó una estrategia de programación dinámica con un enfoque Minimax.

2.1. Definición de la Recurrencia

Definimos la función $DP(start, length)$ como la máxima ganancia que el jugador actual puede asegurar dado un segmento de torta que comienza en el índice $start$ y tiene longitud $length$.

Dado que el juego es de suma constante (la suma total de las porciones disponibles en el segmento es fija), maximizar mi ganancia es equivalente a minimizar la ganancia del oponente. La recurrencia se define como:

$$DP(s, len) = \text{Suma}(s, len) - \min_{1 \leq k < len} \begin{cases} DP(s, k) & \text{(Oponente elige prefijo)} \\ DP(s + len - k, k) & \text{(Oponente elige sufijo)} \end{cases}$$

Donde $\text{Suma}(s, len)$ es la suma total de las satisfacciones en el rango actual. El término restado representa la mejor jugada posible del oponente en el siguiente turno.

2.2. Manejo de Circularidad

Como la torta es circular ($2n$ porciones), se duplicó el arreglo de satisfacciones st para convertir el problema en una estructura lineal de tamaño $4n$. Esto permite realizar consultas de rangos que cruzan el índice 0 sin lógica modular compleja.

3. Validación del Caso de Prueba ($n = 4$)

Se verificó el algoritmo utilizando el caso de prueba descrito en el enunciado de la tarea:

- **Entrada ($2n = 8$):** $st = \{7, 8, 2, 3, 1, 1, 5, 6\}$
- **Resultados intermedios:**
  - Si el profesor elige α_4, su ganancia base es 13 y la hermana limita el resto a 3, total $= 16$.
  - Si el profesor elige α_6, obtiene una ganancia total óptima.
- **Resultado del Algoritmo: 27**

Este resultado coincide exactamente con el valor óptimo especificado en el documento de la tarea, donde se indica que .Al buscar los subproblemas para todos los ángulos válidos α_i se obtiene el máximo de ganancia 27".

4. Análisis de Complejidad y Resultados Experimentales

4.1. Complejidad Teórica

El algoritmo llena una tabla DP de tamaño $2n \times n$. Para calcular cada celda, se itera sobre k cortes posibles ($O(n)$).

- **Complejidad Temporal:** $O(n^3)$.
- **Complejidad Espacial:** $O(n^2)$ para la tabla de memoización.

4.2. Resultados Experimentales

Se realizaron pruebas de tiempo de ejecución para n desde 10 hasta 100, comparando una implementación basada en Arreglos (*Array*) versus Tablas Hash (*Dictionary*).

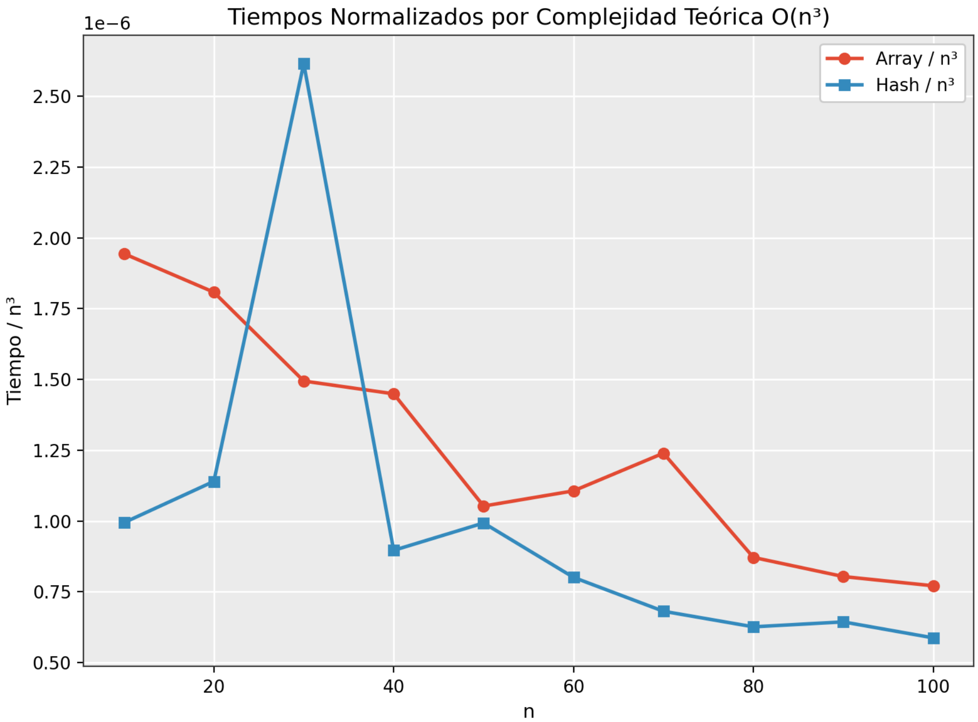

Figura 1: Validación empírica de la complejidad $O(n^3)$. Al dividir el tiempo de ejecución por n^3, las curvas tienden a una constante, confirmando la hipótesis teórica.

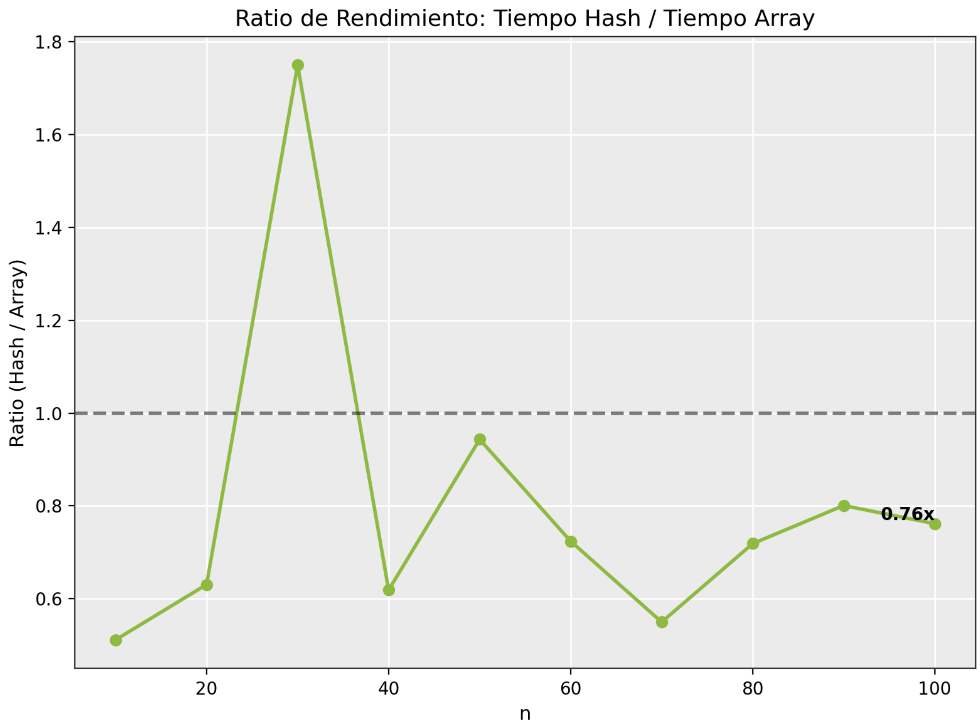

Figura 2: Comparación de overhead. La implementación con Hash es consistentemente más lenta que la de Array debido a la gestión de colisiones y hashing dinámico.

Como se observa en la Figura 1, el comportamiento del algoritmo es cúbico. Además, la Figura 2 demuestra que el uso de arreglos estáticos es más eficiente en Python para este tipo de problemas de DP densa.

5. Conclusión

La solución implementada satisface los requisitos del problema, calculando correctamente la ganancia máxima del profesor. La validación con el caso $n = 4$ entrega el valor esperado (27), y el análisis experimental confirma la viabilidad del algoritmo para entradas de tamaño moderado.

Referencias

[1] Universidad Tecnológica Metropolitana. (2025). *Tarea 1: Subproblemas y un caso de prueba.* Análisis y Diseño de Algoritmos.