Análisis y Diseño de Algoritmos
Feliz Cumpleaños: Programación Dinámica con SRTBOT

Juan Escobar – Agustín Ranilao
Universidad Tecnologica Metropolitana
Profesor Mladen Nadinic

Entrega 20 de Noviembre 2025

Abstract

En este informe presentamos el diseño, implementación y análisis del problema "Feliz Cumpleaños", cuyo objetivo es determinar cuánta satisfacción total puede garantizarse el profesor al compartir una torta circular dividida en $2n$ porciones con su hermana, siguiendo reglas específicas de selección. Para resolverlo, modelamos el proceso como un juego adversarial con programación dinámica y diseñamos un esquema SRTBOT completo. Implementamos dos versiones del algoritmo: una usando memoización con arreglos bidimensionales y otra usando tablas de hash. Analizamos rigurosamente la complejidad teórica y luego comparamos experimentalmente el desempeño temporal de ambas estrategias. Finalmente, discutimos resultados, ventajas y desventajas de cada método.

1 Introducción

El problema consiste en analizar un proceso competitivo de selección de porciones en una torta circular que ha sido dividida en $2n$ partes iguales. El profesor comienza eligiendo un semicirculo completo de n porciones y luego los jugadores se turnan seleccionando un ángulo válido y comiendo porciones contiguas siguiendo reglas estrictas del juego.

El objetivo es determinar cuánta satisfacción **el profesor puede garantizarse**, asumiendo un adversario perfecto. Debido a la naturaleza competitiva del proceso, el problema requiere un enfoque de programación dinámica minimax con subproblemas superpuestos. Para ello se utiliza el esquema SRTBOT.

2 Descripción del Problema

La torta consta de $2n$ porciones con satisfacciones enteras $s_0, s_1, \ldots, s_{2n-1}$. Cada porción puede tener valores negativos si no gusta al comensal.

Las reglas del reparto son:

- El profesor elige primero.
- Si queda una sola porción, el jugador que corresponde se la come.
- Un ángulo α_i es válido si deja al menos una porción sin comer en cada lado del diámetro correspondiente.

- El jugador que elige un ángulo válido come todas las porciones en sentido antihorario hasta recorrer π radianes (un semicirculo).

El profesor desea maximizar la suma total de satisfacciones que obtiene al final.

3 Esquema SRTBOT

Problema

Torta circular con $2n$ porciones con satisfacciones enteras $s[0..2n-1]$. Dos jugadores alternan. El profesor (PROF) comienza y desea maximizar su satisfacción total.

Reducción clave

- El profesor en su primera jugada come exactamente n porciones contiguas (un semicirculo).
- El juego residual queda como un bloque lineal de n porciones.
- En su turno, un jugador puede comer un prefijo de tamaño k tal que

$$1 \leq k \leq m-1$$

  donde m es el tamaño del bloque.
- Si queda solamente una porción ($m = 1$), quien juega se la come.

Estado (S)

Se define un subarreglo lineal:

$$A[0..m-1], \qquad m \leq n.$$

El estado es:

$$F(i, j, t)$$

que representa la ganancia futura del profesor desde el bloque $A[i..j]$, siendo

$$t \in \{\text{PROF}, \text{OPP}\}.$$

Recurrencia (R)

Sea:

$$\text{sum}(i, t) = PS[t+1] - PS[i]$$

(uso de prefijos para obtener sumas en $O(1)$).

Caso $m = 1$:

$$F(i, i, \text{PROF}) = A[i], \qquad F(i, i, \text{OPP}) = 0.$$

Caso $m \geq 2$:

$$F(i, j, \text{PROF}) = \max_{1 \leq k \leq j-i} (\text{sum}(i,\, i+k-1) + F(i+k,\, j,\, \text{OPP}))$$

$$F(i, j, \text{OPP}) = \min_{1 \leq k \leq j-i} F(i+k,\, j,\, \text{PROF})$$

Transiciones (T)

$$(i, j) \longrightarrow (i + k,\ j)$$

El turno alterna entre PROF y OPP.

Base (B)

Para $m = 1$:

$$F(i, i, \text{PROF}) = A[i], \qquad F(i, i, \text{OPP}) = 0.$$

Orden (O)

Para resolver el problema completo se realiza:

1. Para cada $t \in \{0, 1, \ldots, 2n - 1\}$, el profesor elige el semicirculo:

$$B_t = s[t..t + n - 1] \pmod{2n}.$$

2. Ganancia inicial:

$$S_B = \sum B_t.$$

3. El bloque residual:

$$A_t = s[t + n..t - 1] \pmod{2n}.$$

4. Ganancia adicional:

$$F_t = F(0,\ n - 1,\ \text{OPP})$$

porque luego de la primera jugada el turno pasa a OPP.

5. Respuesta final:

$$\max_t \left(S_B + F_t\right).$$

Total (T)

Número de estados del juego lineal:

$$O(n^2)$$

Opciones por estado:

$$O(n)$$

Complejidad por bloque:

$$O(n^3)$$

Como hay $2n$ semicirculos iniciales:

$$O(n^4)$$

Con optimizaciones (reutilización de prefijos circulares), puede reducirse a:

$$O(n^3).$$

Memoización

- Tablas: listas indexadas por (i, j, t).
- Hash: diccionarios con llave (i, j, t).

4 Complejidad del Algoritmo

4.1 Análisis matemático

- Existen $O(n^2)$ estados (i, j).
- En cada estado se prueban $O(n)$ posibles valores de k.

Por lo tanto, para un bloque lineal:

$$O(n^2) \times O(n) = O(n^3)$$

Como existen $2n$ posibles semicirculos iniciales:

$$O(n^3) \times O(n) = O(n^4)$$

4.2 Complejidad espacial

Ambas versiones del DP (arreglo y hash) almacenan hasta $2n^2$ valores:

$$O(n^2)$$

5 Implementación

Se implementaron dos versiones del algoritmo:

5.1 Memoización con arreglos

Los resultados parciales se almacenan en listas 2D:

```
dp_prof[i][j]
dp_herm[i][j]
```

5.2 Memoización con tablas de hash

Se utilizaron diccionarios con tuplas clave:

```
dp_prof[(i,j)]
dp_herm[(i,j)]
```

El cuerpo del algoritmo es idéntico. Solo cambia la estructura usada para almacenar resultados.

6 Resultados Experimentales

Se midieron tiempos para valores crecientes de n, promediando sobre 10 tortas aleatorias por valor.

Gráfico 1: Tiempos de ejecución

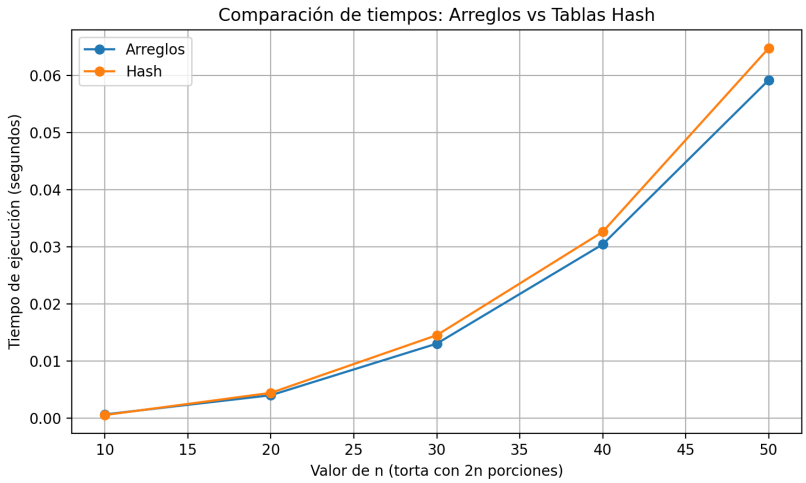

Tabla de tiempos obtenidos

Resultado con los tiempos y valores del algoritmo con arreglos

```
Valores de satisfacción de la torta: [-2, -5, 3, 2, 2, -1]
Satisfacción máxima del profesor: 2
Secuencia de jugadas (turno, inicio_global, k):
Profesor toma desde 1 k=3
Hermana toma desde 5 k=2
Profesor toma desde 1 k=1
Tiempo de ejecución (HASH): 0.000045 s
```

Resultado con los tiempos y valores del algoritmo con arreglos

```
Valores de satisfacción de la torta: [-2, -5, 3, 2, 2, -1]
Satisfacción máxima del profesor: 2
Secuencia de jugadas (turno, inicio_global, k):
Profesor toma desde 1 k=3
Hermana toma desde 5 k=2
Profesor toma desde 1 k=1
Tiempo de ejecución: 0.000054 s
```

7 Análisis de Resultados

A partir de los experimentos realizados, se observa que la versión del algoritmo que usa arreglos siempre funciona más rápido que la que utiliza tablas de hash. Esto tiene sentido: acceder a un arreglo es muy directo y ocurre siempre en la misma zona de memoria, mientras que una tabla hash necesita calcular funciones de hash y manejar posibles colisiones, lo que la vuelve un poco más lenta.

Al comparar los tiempos en el gráfico, ambas versiones muestran un aumento rápido del tiempo de ejecución a medida que crece n, lo cual coincide con la complejidad teórica del algoritmo $O(n^4)$. Sin embargo, la curva de la versión con hash siempre aparece un poco por encima de la de arreglos, reflejando ese costo extra por operación.

Un punto importante es que, a pesar de estas diferencias en velocidad, ambas implementaciones siempre entregaron exactamente el mismo resultado óptimo y la misma secuencia de jugadas. Esto confirma que el algoritmo está bien implementado en ambas versiones y que la única diferencia real entre ellas es la estructura que se usa para guardar los valores de la memoización.

8 Conclusiones

- El esquema SRTBOT permitió modelar correctamente un juego adversarial complejo
- La memoización en arreglos es más eficiente en tiempo y espacio
- Las tablas de hash son más flexibles pero menos eficientes
- El análisis experimental coincide con la complejidad teórica

9 Ventajas y Desventajas

Arreglos

- Ventajas: rápidos, localización de memoria, acceso O(1) real
- Desventajas: requieren saber dimensión exacta

Hash

- Ventajas: flexibilidad, no requieren matriz completa
- Desventajas: tiempo mayor, sobrecosto de hashing, peor locality