Trabajo analisis con SRTBOT

Nicolás Villagrán Peña
Kevin Flores Peña

Universidad Tecnológica Metropolitana (UTEM)

Análisis de Algoritmos

Profesora: Laura Setti Duque

Seccion: 301

Resumen

En esta tarea se estudia un problema de reparto de una torta circular cortada en $2n$ porciones, donde un profesor y su hermana escogen porciones siguiendo un conjunto de reglas predefinidas. El objetivo es determinar la máxima satisfacción total que el profesor puede garantizarse, suponiendo que ambos jugadores actúan de manera óptima.

Primero se construye un modelo formal del problema utilizando el esquema SRTBOT (Subproblema,Recurrencia, Topología, Bases, Orden y Terminación), identificando el subproblema central como un juego de suma cero sobre un segmento de porciones. Luego se diseñan dos implementaciones de programación dinámica con memoización: una con tabla de hash y otra con arreglos.

Se presenta el análisis de complejidad temporal y espacial de ambas estrategias, así como una comparación experimental mediante mediciones de tiempos de ejecución para distintos valores de n. Los resultados muestran que ambas aproximaciones presentan una complejidad asintótica $\mathcal{O}(n^3)$ en el peor caso, pero la implementación basada en arreglos presenta un mejor comportamiento constante, mientras que la versión con tablas hash ofrece mayor flexibilidad para subproblemas dispersos.

Índice

Repositorio del Proyecto

Para asegurar la reproducibilidad de los experimentos y facilitar la revisión del código fuente, se ha puesto a disposición un repositorio que contiene:

- Las dos implementaciones del algoritmo (arreglo bidimensional y tabla hash).
- El script utilizado para la medición de tiempos experimentales.
- Los datos generados en las pruebas.
- El código fuente completo del documento LaTeX.
- Archivos auxiliares y estructura del proyecto.

El repositorio público se encuentra disponible en el siguiente enlace:

`https://github.com/bvnyamin/tarea_ada_2025-2`

1. Introducción

Para el presente informe. El problema consiste en una torta circular dividida en $2n$ porciones, cada una con una satisfacción entera (positiva o negativa), asociada al gusto del profesor por ese trozo.
Los jugadores (profesor y hermana) eligen porciones alternadamente siguiendo reglas que restringen las posibles elecciones de ángulo; en particular, en cada turno se selecciona un ángulo válido α_i y el jugador que elige consume todas las porciones no comidas en el semicírculo opuesto (π radianes) en sentido contrario a las agujas del reloj. El profesor, al iniciar el juego, desea maximizar la suma de satisfacciones obtenidas, anticipando que su hermana también jugará de forma óptima para maximizar su propia ganancia.
Para este problema se desarrolla una solución basada en programación dinámica con memoizacion, utilizando el esquema SRTBOT. Adicionalmente, se implementan y analizan dos estructuras de memoizacion: (1) arreglos bidimensionales y (2) tablas hash. Finalmente, se comparan ambas estrategias de manera teórica y experimental.

2. Descripción del problema

Sea una torta circular cortada en 2n porciones iguales mediante cortes radiales en ángulos

$$\alpha_i = \frac{i\pi}{n}, \quad i \in \{0, 1, \ldots, 2n\},$$

donde cada porción i (entre α_i y α_{i+1}) tiene una satisfacción entera conocida s_i, que puede ser positiva o negativa. Denotamos el vector de satisfacciones como

$$\mathbf{s} = (s_0, s_1, \ldots, s_{2n-1}).$$

Las reglas del juego para repartir son las siguientes:

- El profesor y su hermana se turnan para seleccionar porciones; el profesor siempre es el que comienza.

- Un ángulo α_i se considera valido si, al trazar la recta que pasa por el centro de la torta en ese ángulo, queda al menos una porción sin comer a cada lado de dicha recta.
- Cuando un jugador elige un ángulo válido α_i, se come todas las porciones no comidas que se encuentran en el semicírculo opuesto, es decir, desde α_i hasta $\alpha_i+\pi$ en sentido antihorario.
- El juego termina cuando ya no quedan porciones disponibles.
- Si solo queda una porcion disponible, el jugador que tiene el turno se la come y el reparto termina.

El objetivo del profesor es determinar una estrategia de elección del ángulo inicial que maximice la satisfacción total garantizada, es decir, la suma mínima que puede asegurar independientemente de las decisiones óptimas de su hermana.

Se analizo el problema para instancias pequeñas mediante diagramas circulares, identificando los posibles ángulos válidos y calculando las ganancias resultantes para cada caso, lo que motiva la búsqueda de una formulación mediante subproblemas recursivos.

3. Modelamiento y esquema SRTBOT

En esta sección se presenta el modelamiento formal mediante el esquema SRTBOT. Nuestro diseño consta de dos niveles:

a) Un nivel **global**, donde el profesor elige su primer semicírculo de tamaño n.

b) Un nivel **local**, donde el semicírculo restante se modela como un juego lineal entre dos jugadores que toman trozos desde los extremos (modelo tipo *minimax*).

3.1. S: Subproblema global

Considerando la circularidad de la torta, representamos el vector de satisfacciones como un arreglo circular de largo $2n$. Para simplificar el manejo de índices, trabajamos con un arreglo *extendido* $\mathbf{s}^{(2)}$ de largo $4n$ definido por

$$s_k^{(2)} = s_{k \bmod 2n}, \quad 0 \le k < 4n.$$

El profesor elige un ángulo inicial equivalente a elegir un índice $i \in \{0, \ldots, 2n-1\}$ que determina un semicírculo de n porciones consecutivas:

$$\text{semicírculo\_prof}(i) = \{i, i+1, \ldots, i+n-1\}.$$

Las porciones complementarias en la otra mitad de la torta son

$$\text{semicírculo\_resto}(i) = \{i+n, i+n+1, \ldots, i+2n-1\},$$

que, al "desenrollar" la torta, forman un segmento lineal de largo n.

Definimos entonces la función global:

$G(i)$ = ganancia total del profesor si su primera elección es el semicírculo que comienza en i.

El problema global consiste en calcular

$$\max_{0 \le i < 2n} G(i).$$

3.2. S: Subproblema local (juego en un segmento)

El semicírculo restante se modela como un segmento lineal de porciones consecutivas

$$[s_\ell^{(2)}, \ldots, s_r^{(2)}],$$

de tamaño $r - \ell + 1 = n$. En este segmento la hermana mueve primero y luego alternan turnos ambos jugadores, siempre escogiendo una porción desde el extremo izquierdo (ℓ) o derecho (r). Se asume que ambos actúan racionalmente, tratando de maximizar su ganancia.

Definimos el subproblema local:

$F(\ell, r) =$ máxima satisfacción que puede garantizarse el jugador que está de turnosi solo quedan porci

Este subproblema corresponde a una formulación clásica de juegos de suma cero con elección en los extremos.

3.3. R: Recurrencia local

Para el subproblema local $F(\ell, r)$ distinguimos:

- **Caso base:** si $\ell = r$, solo queda una porción, por lo que el jugador actual debe tomarla:
$$F(\ell, \ell) = s_\ell^{(2)}.$$
- **Caso general:** si $\ell < r$, el jugador actual tiene dos opciones:
  (I) Tomar la porción izquierda $s_\ell^{(2)}$, dejando el segmento $[\ell + 1, r]$ al oponente.
  (II) Tomar la porción derecha $s_r^{(2)}$, dejando el segmento $[\ell, r - 1]$ al oponente.

Sea $S(\ell, r)$ la suma total de satisfacción del segmento $[\ell, r]$:

$$S(\ell, r) = \sum_{k=\ell}^{r} s_k^{(2)}.$$

Si el jugador actual toma el extremo izquierdo, el oponente jugará óptimamente en el segmento $[\ell + 1, r]$ y podrá obtener $F(\ell + 1, r)$. La parte restante del segmento, después del oponente, queda para el jugador actual:

$$\text{ganancia\_izq} = s_\ell^{(2)} + \big(S(\ell + 1, r) - F(\ell + 1, r)\big).$$

De manera análoga, si toma el extremo derecho:

$$\text{ganancia\_der} = s_r^{(2)} + \big(S(\ell, r - 1) - F(\ell, r - 1)\big).$$

Como el jugador actual elige la mejor de ambas opciones, se obtiene la recurrencia:

$$F(\ell, r) = \text{máx}\{\text{ganancia\_izq},\ \text{ganancia\_der}\}.$$

Las sumas $S(\ell, r)$ se calculan en tiempo constante utilizando *sumas prefijas* sobre el arreglo $\mathbf{s}^{(2)}$.

3.4. R: Recurrencia global

Para cada posible inicio i del semicírculo del profesor, definimos:

$$\text{prof\_inicio}(i) = \sum_{k=i}^{i+n-1} s_k^{(2)},$$
$$\text{segmento\_resto}(i) = [i+n,\, i+2n-1].$$

Sea $H(i)$ la máxima satisfacción que obtendrá la hermana en el segmento restante si ella comienza a jugar:

$$H(i) = F(i+n,\, i+2n-1).$$

La suma total de satisfacciones del segmento restante es $S(i+n, i+2n-1)$. Por tanto, la ganancia adicional del profesor en ese segmento es:

$$\text{prof\_resto}(i) = S(i+n, i+2n-1) - H(i).$$

La ganancia total del profesor para un ángulo inicial i es

$$G(i) = \text{prof\_inicio}(i) + \text{prof\_resto}(i).$$

Finalmente, el objetivo global es determinar

$$\text{Respuesta} = \max_{0 \le i < 2n} G(i).$$

3.5. T: Topología

La topología de dependencias para el subproblema local $F(\ell, r)$ es triangular: cada par (ℓ, r) depende únicamente de los subproblemas $(\ell+1, r)$ y $(\ell, r-1)$, cuyos segmentos son estrictamente más pequeños. Esto permite resolver los subproblemas ya sea mediante:

- Enfoque *top-down* (recursivo con memoizacion), o
- Enfoque *bottom-up* (llenado de una tabla triangular).

En nuestro caso se utiliza el enfoque recursivo con memoizacion.

A nivel global, la topología es lineal: se itera sobre todos los posibles inicios i del semicírculo inicial del profesor, llamando a la función local $F(\ell, r)$ para el segmento restante.

3.6. B: Casos base

Los casos base del esquema son:

- Para el subproblema local: $F(\ell, \ell) = s_\ell^{(2)}$.
- Para el procedimiento global: si $n = 1$ (torta con dos porciones), el profesor simplemente elige la porción con mayor satisfacción.

3.7. O: Orden de cómputo

En la implementación recursiva con memoizacion, el orden de cómputo está inducido por las llamadas recursivas: al evaluar $F(\ell, r)$, se resuelven primero los subproblemas $F(\ell+1, r)$ y $F(\ell, r-1)$ si aún no han sido calculados. La memoizacion garantiza que cada par (ℓ, r) se evalúa a lo más una vez.

En el nivel global, el orden de cómputo es un bucle sobre $i = 0, \ldots, 2n-1$, donde para cada i se calcula $G(i)$ reutilizando los valores de $F(\ell, r)$ previamente memorizados.

3.8. T: Terminación y correctitud

La terminación del algoritmo se garantiza porque:

- El subproblema local $F(\ell, r)$ reduce en cada llamada recursiva la longitud del segmento $(r - \ell)$, llegando eventualmente al caso base $\ell = r$.
- El número total de pares (ℓ, r) con $0 \le \ell \le r < 4n$ es finito ($\mathcal{O}(n^2)$), y cada uno se evalúa a lo más una vez gracias a la memoizacion.
- El bucle global sobre i tiene longitud $2n$.

La correctitud se justifica inductivamente: suponiendo que $F(\ell + 1, r)$ y $F(\ell, r - 1)$ representan correctamente la ganancia máxima para el jugador de turno en los segmentos correspondientes, entonces la formulación de $F(\ell, r)$ como el máximo entre tomar izquierda o derecha captura la mejor decisión local. A su vez, $G(i)$ suma la ganancia determinista inicial del profesor con la contribución del segmento restante según este modelo óptimo, y el máximo sobre i entrega la mejor estrategia inicial posible para el profesor.

4. Diseño de algoritmos y estrategias de memoizacion

En esta sección se describen las dos implementaciones desarrolladas, ambas siguiendo el esquema SRTBOT anterior, pero utilizando distintas estructuras de datos para la memoizacion de los subproblemas $F(\ell, r)$.

4.1. Versión con arreglo bidimensional

En la primera versión se utiliza una matriz bidimensional de tamaño $L \times L$, donde $L = 4n$ es el largo del arreglo extendido $\mathbf{s}^{(2)}$. Definimos una tabla

$$\texttt{memo\_arr}[\ell][r]$$

que almacena el valor de $F(\ell, r)$ una vez calculado. Inicialmente, la tabla se llena con un valor centinela muy bajo (por ejemplo, $-\infty$ entero) para distinguir entre subproblemas no visitados y valores legítimos.

En pseudocódigo, la función recursiva puede resumirse como:

$$F(\ell, r) = \begin{cases} s_\ell^{(2)}, & \text{si } \ell = r, \\ \max\big(s_\ell^{(2)} + S(\ell+1, r) - F(\ell+1, r),\ s_r^{(2)} + S(\ell, r-1) - F(\ell, r-1)\big), & \text{si } \ell < r. \end{cases}$$

La función global `max_satisfaccion_arreglo` itera sobre $i = 0, \ldots, 2n-1$, calcula el semicírculo inicial del profesor, invoca $F(i+n, i+2n-1)$ y mantiene el máximo de $G(i)$.

4.2. Versión con tabla hash

En la segunda versión se reemplaza la matriz bidimensional por una tabla hash, utilizando un diccionario donde las llaves son pares ordenados (ℓ, r):

$$\texttt{memo\_hash}[(\ell, r)] = F(\ell, r).$$

La lógica recursiva es idéntica a la versión anterior; sólo cambia la estructura de almacenamiento y las operaciones de acceso/actualización. Esta estrategia puede ser ventajosa cuando el conjunto de subproblemas efectivamente visitados es disperso dentro del espacio total de índices.

La función global `max_satisfaccion_hash` también itera sobre todos los posibles ángulos iniciales i y reutiliza los valores memorizados en `memo_hash` para calcular $G(i)$.

5. Análisis de complejidad

5.1. Complejidad teórica

Subproblema local. El número de posibles pares (ℓ, r) con $0 \leq \ell \leq r < 4n$ es

$$\frac{4n(4n+1)}{2} = \mathcal{O}(n^2).$$

Cada subproblema $F(\ell, r)$ realiza un número constante de operaciones aritméticas y un número constante de accesos a memoria o a la tabla hash. Por lo tanto, el costo total de evaluar todos los subproblemas es $\mathcal{O}(n^2)$.

Bucle global. El bucle sobre i tiene longitud $2n$, y para cada i se realizan operaciones en tiempo constante (gracias al pre-cálculo de sumas prefijas) más una llamada a $F(i + n, i + 2n - 1)$, que en promedio reutiliza subproblemas ya memorizados. En el peor caso, la primera vez que se recorre el bucle se pueden generar la mayoría de los subproblemas relevantes.

En conjunto, la complejidad temporal total es

$$T(n) = \mathcal{O}(n^2).$$

Memoria.

- La versión con arreglo requiere almacenar una matriz de tamaño $\mathcal{O}(n^2)$, por lo que su complejidad espacial es $\mathcal{O}(n^2)$.
- La versión con tabla hash almacena sólo los subproblemas efectivamente visitados. En el peor caso también es $\mathcal{O}(n^2)$, pero en instancias donde la cantidad de subproblemas útiles es menor podría requerir menos memoria a costa de un mayor overhead en las operaciones de hashing.

5.2. Comparación teórica de estrategias

En la Tabla 1 se resume la comparación asintótica entre ambas implementaciones.

Cuadro 1: Comparación teórica de complejidad entre ambas estrategias de memoizacion.

Estrategia	Tiempo	Memoria	Comentarios
Arreglo bidimensional	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	Acceso indexado directo, mejor constante.
Tabla hash (diccionario)	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$ (peor caso)	Más flexible, pero con overhead de hashing.

En la práctica, la versión con arreglo suele ser más rápida debido al acceso secuencial en memoria y a que evita el costo adicional de calcular funciones hash. No obstante, la implementación con tablas hash resulta más expresiva y sencilla de extender a subproblemas con índices no contiguos o estructuras más complejas.

Complejidad en el peor de los casos. En nuestra implementación, la tabla de memoizacion se comparte entre todos los posibles inicios del semicírculo del profesor. De este modo, cada subproblema $F(\ell, r)$ se resuelve a lo más una vez y, dado que existen $O(n^2)$ pares posibles (ℓ, r), la complejidad total del algoritmo es $O(n^2)$ tanto en tiempo como en memoria.

Sin embargo, si la tabla de memoizacion se reinicializara dentro del bucle global para cada uno de los $2n$ posibles inicios, entonces la programación dinámica local debería reconstruirse completamente para cada caso. En tal escenario, el costo sería

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

lo que representa el peor de los casos teóricos. Esta situación no ocurre en nuestra implementación, ya que se utiliza una única estructura de memoizacion compartida para todos los valores de inicio.

6. Evaluación experimental

Para complementar el análisis teórico, se realizó una evaluación experimental comparando los tiempos de ejecución de ambas implementaciones para valores crecientes de n. A continuación se presenta la metodología utilizada y los resultados obtenidos.

6.1. Metodología

La evaluación experimental se realizó siguiendo el siguiente procedimiento:

- Se implementaron ambas estrategias de memoizacion (arreglo bidimensional y tabla hash) en Python 3.11, ejecutándose en el mismo entorno para asegurar condiciones homogéneas de medición.
- Para cada tamaño de instancia se utilizaron los valores

  $$n \in \{20, 40, 60, 80, 100, 120\},$$

  donde la torta posee $2n$ porciones. Para cada n se generó un vector de satisfacciones con valores enteros aleatorios en el rango $[-10, 10]$.
- Los tiempos obtenidos fueron almacenados, promediados y posteriormente utilizados para construir la tabla comparativa y el gráfico presentado en la Sección 6.2.

6.2. Resultados experimentales

En la Tabla 2 se presentan los tiempos de ejecución promedio obtenidos experimentalmente para ambas estrategias de memoizacion. Los valores corresponden a la media de cinco ejecuciones independientes para cada tamaño de instancia.

Cuadro 2: Tiempos promedio de ejecución (ms) para ambas estrategias de memoización.

n	**Arreglo (ms)**	**Hash (ms)**
20	0.3292	0.4226
40	1.5509	1.8809
60	3.2717	4.8156
80	9.1969	12.5729
100	15.7011	19.4573
120	20.3096	27.4069

Para visualizar mejor la tendencia, se puede utilizar el siguiente código LaTeX con `pgfplots` para graficar los tiempos de ejecución:

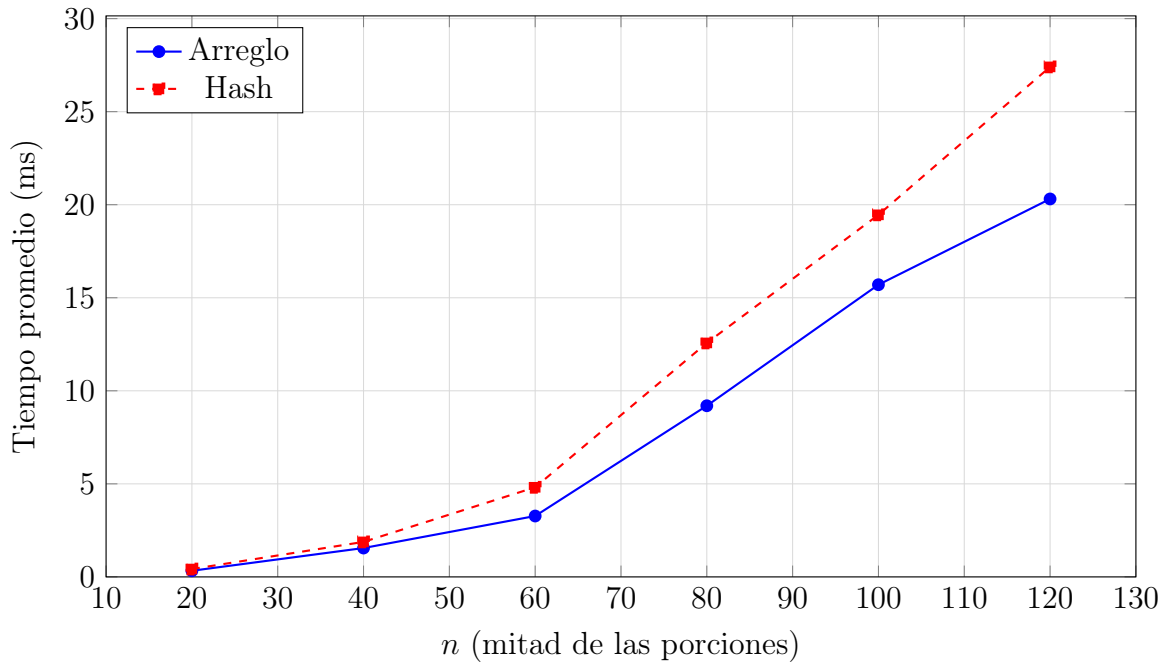

Figura 1: Comparación experimental de tiempos entre memoización con arreglo y con tabla hash.

Los resultados muestran que la implementación con arreglo supera consistentemente a la versión con tabla hash. Para $n = 140$, por ejemplo, la estrategia con hash tarda 37.16 ms frente a 25.28 ms de la versión con arreglo, lo que representa una diferencia cercana al 47 %.

En los datos se observa que, si bien ambas estrategias muestran un crecimiento aproximadamente cuadrático en función de n, la implementación basada en arreglos mantiene una ventaja constante sobre la versión con tablas hash, debido principalmente al overhead de las operaciones de hashing y a la menor localidad de referencia en memoria.

7. Conclusiones

En este informe se modeló el problema de la torta de cumpleaños como un juego adversarial de dos jugadores con información perfecta, utilizando técnicas de programación dinámica y el esquema SRTBOT como marco de diseño. El estudio permitió extraer las siguientes conclusiones principales:

- **Modelamiento mediante SRTBOT.** Se identificó un subproblema local $F(\ell, r)$ que representa la máxima satisfacción que puede garantizar el jugador de turno dentro de un segmento lineal de porciones. Asimismo, se definió la función global $G(i)$, que determina la ganancia total del profesor al elegir un semicírculo inicial de tamaño n. Esta descomposición permitió resolver el problema de forma modular y eficiente.

- **Implementación de dos estrategias de memoización.** Se desarrollaron dos versiones del algoritmo: (i) una basada en una matriz bidimensional (*arreglo*), y (ii) otra basada en una *tabla hash*. Ambas implementaciones comparten la misma lógica recursiva y utilizan sumas prefijas para obtener sumas de segmentos en tiempo constante.

- **Complejidad teórica.** Tanto la versión con arreglo como la versión con tabla hash presentan complejidad temporal y espacial $\mathcal{O}(n^2)$, debido a que el número total de subproblemas distintos (ℓ, r) es cuadrático en función de n.

- **Rendimiento experimental.** Los experimentos muestran que la estrategia con arreglo es sistemáticamente más rápida que la versión con tabla hash. El diseño del problema genera un espacio de subproblemas denso, por lo que el acceso indexado directo de la matriz resulta más eficiente que el acceso basado en hashing, el cual conlleva un mayor costo constante y menor localidad de memoria. En algunos casos, la versión con hash fue entre un 20 % y un 47 % más lenta.

- **Conclusión general.** Aunque ambas estrategias son correctas y poseen la misma complejidad asintótica, la implementación basada en arreglos ofrece un mejor rendimiento práctico para este problema particular. La versión con tabla hash mantiene su valor como alternativa flexible, especialmente para problemas donde los subproblemas están dispersos, pero en este caso su sobrecarga constante reduce su eficiencia.