.UBAfiuba
FACULTAD DE INGENIERÍA

Teoría de Algoritmos
(75.29) Curso Buchwald - Genender

Trabajo Práctico 1
Los Algoritmos Greedy son juegos de niños

20 de septiembre de 2024

Victoria Avalos	108434
Santiago Tissoni	103856
Facundo Monpelat	92716

1. Introducción

Los algoritmos Greedy se caracterizan por seguir un enfoque çodicioso", en el que se toman decisiones locales óptimas con la esperanza de que conduzcan a una solución global óptima. Sin embargo, dicho enfoque no nos garantiza obtener soluciones óptimas en todos los casos. Esta estrategia se utiliza usualmente para resolver problemas que requieren obtener máximos o mínimos. Algunos ejemplos conocidos de algoritmos Greedy son el algoritmo de Dijkstra, los algoritmos de Prim y Kruskal, y el problema de la mochila.

El objetivo de este trabajo práctico es aplicar y analizar el uso de un algoritmo Greedy en un escenario específico: un juego de toma de decisiones en el que se posee una fila de monedas de diferentes valores donde se debe seleccionar una de alguno de los extremos con el objetivo de acumular la mayor puntuación.

A lo largo del trabajo, se evaluará si es posible garantizar una solución óptima utilizando un enfoque Greedy, así como los factores que podrían afectar la optimalidad y la eficiencia del algoritmo propuesto.

1.1. Descripción del Problema

Se dispone de una fila de n monedas, cada una con un valor distinto. En cada turno, el jugador puede elegir solo entre la primera o la última moneda de la fila, removiendo la elegida y cediendo el turno al otro jugador. Los jugadores son dos hermanos, Sophia y Mateo, los cuales poseen 7 y 4 años respectivamente. Sophia, siendo más competitiva y con conocimientos de algoritmos Greedy, toma las decisiones tanto por ella misma como por Mateo, con el objetivo de garantizar que su suma total sea mayor que la de su hermano.

A lo largo de este trabajo, se propone un algoritmo que permite a Sophia seleccionar las monedas de forma tal que asegure la victoria, maximizando el valor acumulado para ella y minimizando el valor que obtiene Mateo.

1.2. Objetivos

Los objetivos del presente trabajo práctico son los siguientes:

- Desarrollar un algoritmo Greedy que obtenga la solución óptima al problema planteado, es decir indicar qué monedas debe ir eligiendo Sophia para sí misma y para Mateo, de tal forma que se asegure de ganar siempre.
- Proveer una demostración de que dicho algoritmo obtiene siempre la solución óptima.
- Describir y justificar la complejidad algorítmica de la solución planteada.
- Analizar la variabilidad de los valores de las diferentes monedas a los tiempos del algoritmo planteado.
- Hacer ejemplos de ejecución y analizar lo encontrado.
- Realizar mediciones de tiempos para verificar la complejidad teórica indicada.

Resolución

2. Algoritmo para que gane siempre Sophia

En este trabajo de ejemplo realizaremos el análisis teórico y empírico de un algoritmo greedy que resuelve el problema de determinar las monedas que escogerán dos jugadores, Sophia y Mateo, en un juego por turnos.

2.1. Algoritmo Greedy

A continuación, mostramos la implementación en Python de un algoritmo que soluciona el problema. Implementamos la función $problema\_monedas(fila)$ que recibe una lista de enteros fila, representando los valores de las monedas.
Esta función devolverá tres listas: una con los valores de las monedas que elige Sophia, otra con los valores que elige Mateo y un una que contiene un mensaje de qué lado extrajo Sophia o Mateo las monedas según sus turnos.

```python
def problema_monedas(fila):
    sofia = []
    mateo = []
    esperados = []

    fila = deque(fila)

    while len(fila) > 0:
        # Turno de Sofia
        if fila[0] > fila[-1]:
            sofia.append(fila[0])
            fila.popleft()
            esperados.append("Primera moneda para Sophia")
        else:
            sofia.append(fila[-1])
            fila.pop()
            esperados.append(" ltima  moneda para Sophia")

        if len(fila) == 0:
            break

        # Turno de Mateo
        if fila[0] > fila[-1]:
            mateo.append(fila[-1])
            fila.pop()
            esperados.append(" ltima  moneda para Mateo")
        else:
            mateo.append(fila[0])
            fila.popleft()
            esperados.append("Primera moneda para Mateo")

    return sofia, mateo, esperados
```

2.2. ¿Por qué es un algoritmo Greedy?

El algoritmo presentado sigue una regla Greedy diferente para cada jugador: Sophia siempre toma la decisión que maximiza su ganancia inmediata, mientras que Mateo elige de manera que minimiza su propia ganancia. Busca el óptimo local para cada instancia del juego sin considerar configuraciones futuras.

2.3. Análisis de optimalidad

Observamos que Sophia elige maximizando sus ganacias con un óptimo local que sea el mayor número de las monedas a escoger, mientras que Mateo hace la inversa minimiza sus ganancias obteniendo el mínimo en cada turno. El algoritmo podría no ser óptimo en caso de haber una cantidad par de monedas de mismo valor, pero dicho escenario se encuentra descartado por enunciado. Además, como ambos jugadores tienen la misma posibilidad de tomar monedas de cualquiera de los extremos y Sophia siempre comienza, ella tiene ventaja ya que puede escoger la mayor al principio del juego de los primeros extremos.

A continuación, profundizaremos el análisis con un enfoque inductivo.

Caso base: 1 moneda Cuando hay solo una moneda en la fila, Sophia la agarra, ya que es su turno, y como no hay más monedas disponibles, Sophia gana automáticamente. Esto establece la base de la inducción para n=1.

También se podría tomar como base el caso de n=2, donde por la naturaleza del problema siempre habrá una moneda de valor mayor a la otra (el caso de una cantidad par de monedas del mismo valor está descartado por el enunciado del problema). El algoritmo elegirá aquella de mayor valor para Sophia, y la restante (inevitablemente de menor valor) será elegida por Mateo. Para n=2, Sophia siempre gana.

Hipótesis de inducción: Supongamos que el algoritmo garantiza que Sophia siempre gana para cualquier fila con n monedas, es decir, cuando la fila tiene n monedas, Sophia sigue la estrategia de elegir siempre la moneda de mayor valor y al final tendrá más puntos que Mateo.

Ahora probaremos que Sophia también ganará cuando la fila tenga n+1 monedas. Consideremos que n ¿= 3, ya que el caso de n=2 ya lo analizamos.

Como comienza Sophia, ella tiene dos opciones: elegir la moneda al inicio o al final de la fila. Según nuestra regla Greedy, ella selecciona la moneda de mayor valor entre los dos extremos. La fila restante tendrá n monedas.

Ahora es el turno de Mateo, quien sigue la regla Greedy opuesta y selecciona la moneda de menor valor en su turno. Inevitablemente, dicha moneda debe valer menos que la seleccionada por Sophia. Para demostrar esto último, asumamos que Sophia eligió la moneda del extremo izquierdo, digamos de valor q, en el turno anterior. Ahora Mateo tiene dos opciones:

- Elegir el extremo derecho: no puede ser mayor que q, ya que en caso contrario habría sido el elegido por Sophia. Por lo tanto, ella mantiene su ventaja.
- Elegir el extremo izquierdo: si es más grande que q, entonces Mateo eligirá el lado derecho, que ya quedó demostrado ser menor a q. Y si el extremo izquierdo es menor que el derecho, entonces Mateo lo seleccionará. Al ser menor que el lado derecho, el cual era menor que q, por transitividad también resulta más pequeño que q. En consecuencia, Sophia mantiene la ventaja.

La fila ahora tiene n-1 monedas.

Sophia vuelve a elegir siguiendo la misma regla de maximización. Por la hipótesis de inducción, sabemos que para cualquier fila de n monedas, Sophia siempre obtendrá una mayor suma que Mateo. Como ahora estamos en una fila de tamaño n-1 después de dos elecciones, podemos aplicar el mismo razonamiento.

Dado que Sophia siempre selecciona la moneda de mayor valor en cada turno, y cada vez que elige maximiza su ganancia frente a las opciones de Mateo, la suma total de Sophia será mayor que la de Mateo al final del juego, independientemente de cuántas monedas haya al principio.

2.4. Complejidad

Nuestro algoritmo recorre una vez la lista de monedas de forma iterativa, realizando en cada iteración operaciones de comparación y eliminación del primer o último elemento de la lista. Para que dichas operaciones se realicen en una complejidad O(1) utilizamos una librería llamada *deque*, la cual asegura que quitar el primer o último elemento se realice en tiempo constante.

En consecuencia, el algoritmo resulta ser de complejidad **O(n)**.

2.5. Variabilidad del valor de las monedas

Con respecto a la variabilidad de las monedas, teniendo en cuenta que no puede haber una cantidad par de monedas del mismo valor, no afecta a la optimalidad del algoritmo. Esto se deriva de la demostración de optimalidad previamente desarrollada, ya que en ningún momento se precisa de variaciones en los valores de las monedas más allá de si una vale más que otra. En resumen, solamente se necesita comparar dos valores entre sí en cada iteración, importando solamente cuál es el mayor o menor. Como el algoritmo Greedy es óptimo, y no afecta el rango de valores de las monedas, se puede concluir que no influye dicha variabilidad.

En la próxima sección veremos algunos ejemplos de configuraciones de filas de monedas que poseen variabilidad en los valores, y se podrá apreciar que el algoritmo siempre asegura que gane Sophia.

2.6. Ejemplos de ejecución

A continuación mostraremos algunos de los resultados obtenidos de ejemplos de ejecución para diferentes configuraciones de filas de monedas. Los mismos corresponden a tests con diversos valores de monedas.

Ejemplo 1: $[20, 30, 2, 10, 50]$

Jugador	Monedas	Total de Puntos
Sophia	[50, 20, 30]	100
Mateo	[10, 2]	12

Ejemplo 2: $[10, 15, 50, 40, 60, 1000, 20, 1500, 22]$

Jugador	Monedas	Total de Puntos
Sophia	[22, 1500, 50, 1000, 60]	2632
Mateo	[10, 15, 20, 40]	85

Ejemplo 3: $[4, 2, 9, 5, 10, 1, 3, 6, 8, 7]$

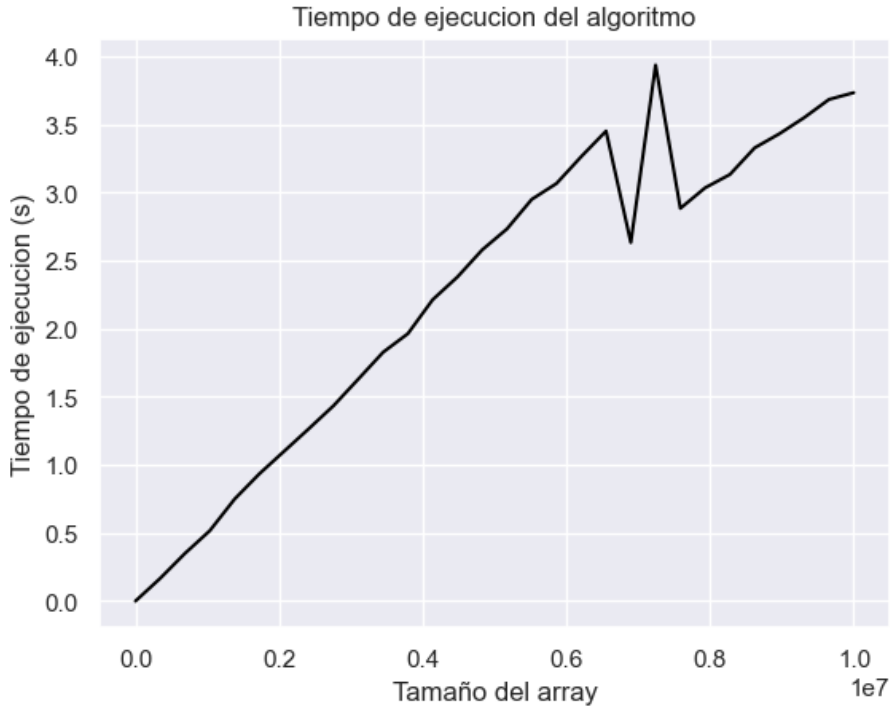
Jugador	Monedas	Total de Puntos
Sophia	[7, 8, 9, 10, 6]	40
Mateo	[4, 2, 5, 1, 3]	15

Se puede apreciar que a pesar de los distintos rangos de valores utilizados para las monedas, en todos los casos gana Sophia.

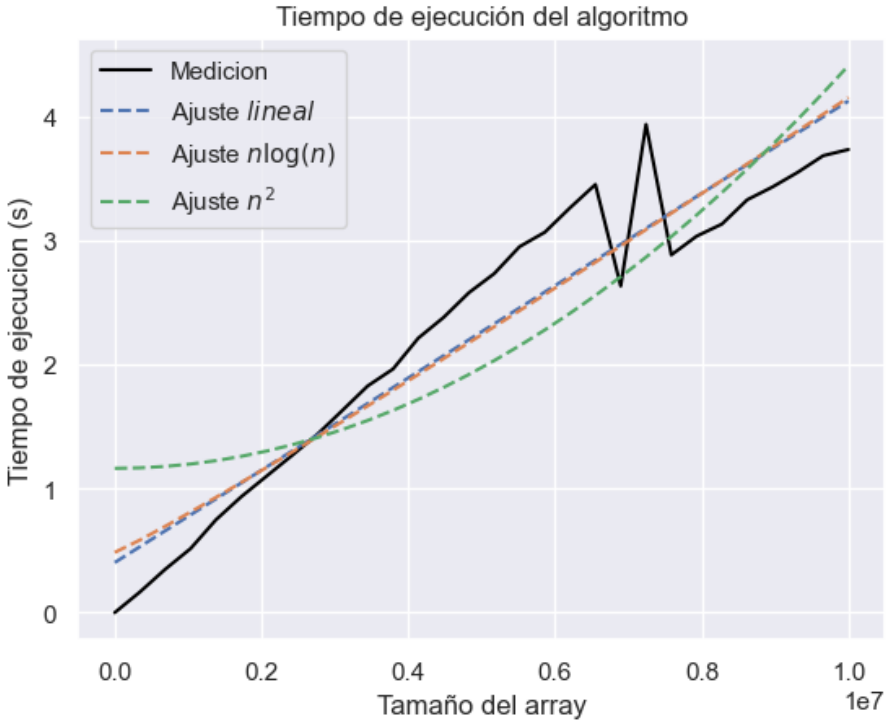
3. Mediciones

Para estimar la complejidad de nuestro algoritmo, hemos hecho 30 ejecuciones de diferente tamaño, yendo desde partidas con 100 monedas, hasta partidas con 10

millones de monedas. Además, cada partida es ejecutada 10 veces para obtener una estimación más precisa de los tiempos de ejecución del programa, a partir de la media de tiempos tomados por cada partida.



Como se puede apreciar, el tiempo de ejecución del algoritmo crece de forma lineal a medida que aumentamos el tamaño del arreglo de entrada del algoritmo. Para concluir con mas precisión esta idea, podemos ajustar diferentes curvas por cuadrados mínimos y ver qué tipo de ajuste se aproxima más a la curva graficada.



Hicimos un ajuste lineal, uno $n * log(n)$ y uno n^2 . Como podemos ver el ajuste lineal es el que mejor se aproxima a nuestra curva de complejidad, tal como lo habíamos argumentado previamente en este informe. La suma de los errores cuadráticos es el menor para el ajuste lineal, le sigue de cerca el ajuste logarítimico. El ajuste cuadrático en cambio presenta un error mucho mayor, ya que esa curva es la que peor se ajusta a los datos.

Ajuste	**Parámetros**	**Suma del error cuadrático**
Lineal	$c_1 : 3{,}721e-07, c_2 : 0{,}403$	3.873
Logarítmico	$c_1 : 2{,}275e-08, c_2 : 0{,}485$	4.259
Cuadrático	$c_1 : 3{,}246e-14, c_2 : 1{,}163$	10.711

De esta forma, podemos apreciar de forma empírica que el algoritmo tiene complejidad $O(n)$, ya que el tiempo de ejecución del algoritmo crece de forma lineal con respecto al tamaño de la entrada.

4. Conclusiones

El análisis del problema y la implementación del algoritmo Greedy propuesto han demostrado ser efectivos para garantizar que Sophia gane siempre. A lo largo de las pruebas realizadas, se observó que la variabilidad en los valores de las monedas no afecta el resultado final del algoritmo. Esto se debe a que la estrategia aplicada siempre selecciona la moneda de mayor valor disponible, independientemente de cómo estén distribuidos los valores restantes. Al elegir de manera óptima en cada turno, Sophia asegura una ventaja constante sobre Mateo, sin importar si las monedas tienen valores muy distintos o cercanos entre sí.

Además, la optimalidad del algoritmo queda demostrada, ya que cada decisión tomada por Sophia maximiza su ganancia inmediata, lo que le asegura la victoria en todos los casos. Aunque podrían existir otras estrategias alternativas que conduzcan a resultados similares, el enfoque Greedy aplicado es suficiente para garantizar que Sophia gane en cada partida sin dejarse vencer, cumpliendo así con el objetivo propuesto.

En cuanto a la complejidad del algoritmo, las mediciones empíricas realizadas corroboraron que su tiempo de ejecución es lineal, es decir, *O(n)*. Este comportamiento eficiente se mantuvo incluso al incrementar el tamaño de la fila de monedas, y los gráficos generados mediante la técnica de mínimos cuadrados donde confirmaron el ajuste teórico propuesto. Por lo tanto, el algoritmo no solo es óptimo en términos de garantizar la victoria de Sophia, sino también en cuanto a su complejidad temporal,

permitiendo una ejecución eficiente incluso para filas grandes de monedas.

Finalizando, el algoritmo Greedy en este caso resultó en una solución óptima. Sin embargo, es importante destacar que, en general, los algoritmos Greedy no siempre garantizan una solución óptima para todos los problemas.