Trabajo Práctico 1
Los Algoritmos Greedy son juegos de niños

Teoría de Algoritmos
(75.29) Curso Buchwald - Genender

Nombre	Padrón
Martin Alejo Polese	106808
Nicolas Agustin Riedel	102130
Denise Dall'Acqua	108645

Índice

Consigna

Introducción y primeros años

Cuando Mateo nació, Sophia estaba muy contenta. Finalmente tendría un hermano con quien jugar. Sophi tenía 3 años cuando Mateo nació. Ya desde muy chicos, ella jugaba mucho con su hermano.

Pasaron los años, y fueron cambiando los juegos. Cuando Mateo cumplió 4 años, el padre de ambos le explicó un juego a Sophia: Se dispone una fila de n monedas, de diferentes valores. En cada turno, un jugador debe elegir alguna moneda. Pero no puede elegir cualquiera: sólo puede elegir o bien la primera de la fila, o bien la última. Al elegirla, la remueve de la fila, y le toca luego al otro jugador, quien debe elegir otra moneda siguiendo la misma regla. Siguen agarrando monedas hasta que no quede ninguna. Quien gane será quien tenga el mayor valor acumulado (por sumatoria).

El problema es que Mateo es aún pequeño para entender cómo funciona esto, por lo que Sophia debe elegir las monedas por él. Digamos, Mateo está "jugando". Aquí surge otro problema: Sophia es muy competitiva. Será buena hermana, pero no se va a dejar ganar (consideremos que tiene 7 nada más). Todo lo contrario. En la primaria aprendió algunas cosas sobre algoritmos greedy, y lo va a aplicar.

Consigna

1. Hacer un análisis del problema, y proponer un algoritmo greedy que obtenga la solución óptima al problema planteado: Dados los n valores de todas las monedas, indicar qué monedas debe ir eligiendo Sophia para si misma y para Mateo, de tal forma que se asegure de ganar siempre. Considerar que Sophia siempre comienza (para sí misma).

2. Demostrar que el algoritmo planteado obtiene siempre la solución óptima (desestimando el caso de una cantidad par de monedas de mismo valor, en cuyo caso siempre sería empate más allá de la estrategia de Sophia).

3. Escribir el algoritmo planteado. Describir y justificar la complejidad de dicho algoritmo. Analizar si (y cómo) afecta la variabilidad de los valores de las diferentes monedas a los tiempos del algoritmo planteado.

4. Analizar si (y cómo) afecta la variabilidad de los valores de las diferentes monedas a la optimalidad del algoritmo planteado.

5. Realizar ejemplos de ejecución para encontrar soluciones y corroborar lo encontrado. Adicionalmente, el curso proveerá con algunos casos particulares que deben cumplirse su optimalidad también.

6. Hacer mediciones de tiempos para corroborar la complejidad teórica indicada. Agregar los casos de prueba necesarios para dicha corroboración. Esta corroboración empírica debe realizarse confeccionando gráficos correspondientes, y utilizando la técnica de cuadrados mínimos. Para esto, proveemos una explicación detallada, en conjunto de ejemplos.

7. Agregar cualquier conclusión que les parezca relevante.

Resolución

1. Algoritmo greedy para la resolución del problema

El problema que se nos presenta en resumidas cuentas es el siguiente:

En un juego por turnos, se nos da una fila con monedas y solo puedo sacar una de uno de los extremos de la fila por turno. El juego termina cuando no quedan más monedas, y gana el jugador que haya acumulado la mayor ganancia.

Recordemos que un algoritmo greedy, es aquel que al aplicar una regla sencilla, nos permita obtener el óptimo local a mi estado actual, y aplicando iterativamente esa regla, llegar a (idealmente) un óptimo global.

Entonces, un algoritmo greedy para resolver el problema, podría ser el siguiente: Vemos las monedas de ambos extremos (mi información actual), y me quedo con la de mayor valor cuando es el turno de Sophia, y la de menor valor cuando es el turno de Mateo (está seria nuestra regla sencilla), la cual aplicamos reiterativamente (hasta que no haya monedas) para llegar a una solución óptima (que Sophia gane siempre).

2. Demostración del Algortimo

El algortimo desarrollado anteriormente nos afirma que nos va a dar la solución optima, osea que Sophia gana todas las partidas de su nuevo juego, no importa el orden de los valores de las monedas, dejando a Mateo como el perdedor tanto de cada ronda como del juego en si.

A continuación, se detallará el paso a paso de la demostración que hemos realizado para afirmar la teoria:

Sophie siempre ganas las partidas, por lo tanto también el juego

Utilizamos el método por inducción:

Siendo n la cantidad de monedas:

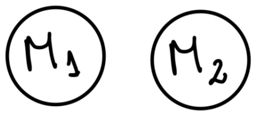
Si $n = 1$

$valorSofiaMonedas = M_1$

$valorMateoMonedas = 0$

Gana Sofia ya que siempre el primer turno es para ella, y solo hay una moneda.

Si $n = 2$



Siendo, $M_1 > M_2$:

$valorSofiaMonedas = M_1$

$valorMateoMonedas = M_2$

Al empezar sofía, ella agarra la moneda más grande y gana

Si $n = 3$

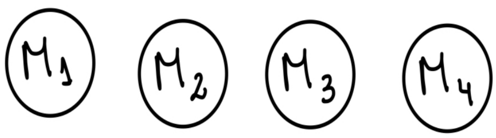
Siendo, $M_1 < M_2 < M_3$:

$valorSophiaMonedas = M_3 + M_2$

$valorMateoMonedas = M_1$

Sophia agarra M_3 al ser el más grande, Mateo agarra (en realidad lo elige Sofía) el M_1 que es el valor más pequeño, y por último Sophia agarra la moneda M_2 que es la última que quedaba. Sophia gana al tener las monedas de mayor valor.

Si $n = 4$



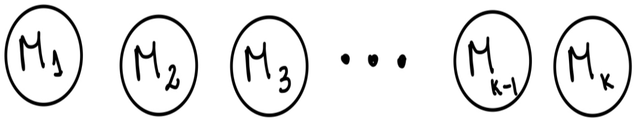
Siendo, $M_4 < M_3 < M_1 < M_2$:

$valorSofiaMonedas = M_1 + M_2$

$valorMateoMonedas = M_4 + M_3$

Sofía agarra M_1 al ser la mas grande en comparación a M_4; Mateo agarra la moneda M_4 al ser la mas pequeña entre M_4 y M_2; Sofá agarra M_2 y por último por descarte Mateo se queda con la moneda M_3. Gana Sofía por tener el mayor valor de monedas.

Si $n = k$



No podemos establecer quien es mayor/menor al haber k valores, ya que en la fila los valores no estan ordenados ascendente o descendentemente.

Para poder analizarlo vamos a hacer una función partida para ver cuales son nuestras opciones para cada uno:

$$ValorSofiaTurno = \begin{cases} M_k \quad si \quad M_k > M_1, ValorMateoTurno = \begin{cases} M_{k-1} & si \quad M_{k-1} < M_1 \\ M_1 & si \quad M_{k-1} > M_1 \end{cases} \\ \\ M_1 \quad si \quad M_k < M_1, ValorMateoTurno = \begin{cases} M_2 & si \quad M_2 < M_k \\ M_k & si \quad M_k < M_2 \end{cases} \end{cases}$$

En consecuencia del turno de Sofía, Mateo va a tener 2 posibilidad por cada elección.

Como podemos observar, siempre hay un patrón: Si los turnos empezaran desde el 0, Sofia siempre tiene los turnos pares ,en los cuales agarra siempre el valor de moneda más grande, y Mateo tiene los turno impares donde agarra el valor los valores más pequeños.

$$valorSofiaTotal = \sum_{k=1}^{n} M_k \text{ siendo } \begin{cases} k = i_{inicial\_actual} & si \quad i_{inicial\_actual} > j_{final\_actual} \\ k = j_{final\_actual} & si \quad i_{inicial\_actual} < j_{final\_actual} \end{cases}$$

$$valorMateoTotal = \sum_{k=1}^{n} M_k \text{ siendo } \begin{cases} k = i_{inicial\_actual} & si \quad i_{inicial\_actual} < j_{final\_actual} \\ k = j_{final\_actual} & si \quad i_{inicial\_actual} > j_{final\_actual} \end{cases}$$

NOTA: $i_{inicial\_actual}$ es el principio de nuestra fila de monedas por turnos, y j_{final_actual} en el final de nuestra fila por turnos

Y para corroborar que nadie está haciendo trampa:

$$valorTotalDeMonedas = valorSofiaTotal + valorMateoTotal$$

3. Algoritmo planteado y complejidad

El algoritmo que decidimos utilizar para resolver el problema, es el siguiente:

- Mientras hayan monedas para elegir:
  - Vemos las monedas que se encuentran en los dos extremos de la fila, y las comparamos:

- Si el turno es de Sophia, se elige la moneda de mayor valor.
- Si el turno es de Mateo, se elige la moneda de menor valor.

- Devolvemos la ganancia acumulada de Sophia y Mateo.

Lo que estamos haciendo, es recorrer toda la fila de monedas (con dos índices, uno para cada extremo), y en cada iteración, comparamos las monedas, y acumulamos la ganancia para Sophia o Mateo (según corresponda el turno), y se agrega el movimiento que se realizó, hasta que finalmente ya no quedan monedas. Por lo tanto, siendo "n"las monedas de la fila, nuestro algoritmo es lineal: O(n), ya que solamente recorremos ese arreglo de monedas, y en cada iteración hacemos operaciones de tiempo constante: O(1).

En conclusión, la complejidad algorítmica es: **O(n)**.

4. Variabilidad

En relación a la complejidad algorítmica y la optimalidad del algortimo, la variabilidad de los valores de las monedas **no** afectan a los tiempos del algoritmo planteado, ya que lo único que se realiza son comparaciones entre las monedas, y adiciones sobre el acumulado.

Sobre la optimalidad del algortimo, la variablidad de los valores de las monedas no va a cambiar la optimalidad ya que gracias a este algoritmo, que hemos demostrado en la sección Demostración, no importa qué valores tengan ni cuantás monedas haya, siempre se va a cumplir que Sofía va a ser la ganadora turno a turno. Es decir siempre se va a dar el óptimo local para nuestor problema que nos va a guiar a nuestro óptimo global que es que Sofía gane el juego.

Sin embargo, lo que si afecta al tiempo, es la cantidad de monedas, ya que por cada moneda se debe realizar una iteración más (puesto que es un turno más del juego), aunque esto no cambia la complejidad del algoritmo, que siempre se mantiene lineal.

5. Ejemplos de ejecución

Se pueden encontrar múltiples ejemplos de ejecución dentro de la carpeta *ejemplos* en el repositorio. Veamos alguno:

Supongamos que tenemos 6 monedas:

```
[3, 7, 1, 12, 9, 5]
```

Recordando lo que vimos en la sección 3 de Algoritmo planteado y complejidad, y que el primer turno es de Sophia:

Vemos las monedas de ambos extremos: 3 y 5. Como el turno es de Sophia, nos quedamos con la moneda de mayor valor, en este caso 5. Entonces: **Ganancia de Sophia = 5**

Nuestras monedas quedarían ahora:

```
[3, 7, 1, 12, 9]
```

Ahora las monedas de ambos extremos son: 3 y 9. Como ahora es el turno de Mateo, nos quedamos con la moneda de menor valor: 3. Entonces: **Ganancia de Mateo = 3**

Nuestras monedas quedarian ahora:

```
[7, 1, 12, 9]
```

Siguiendo el algoritmo hasta el final (hasta que ya no queden monedas), obtenemos finalmente:

Ganancia de Sophia = 26 (5 + 9 + 12)

Ganancia de Mateo = 11 (3 + 7 + 1)

Como la ganancia de Sophia es mayor que la de Mateo ($26 > 11$), Sophia resulta la ganadora del juego.

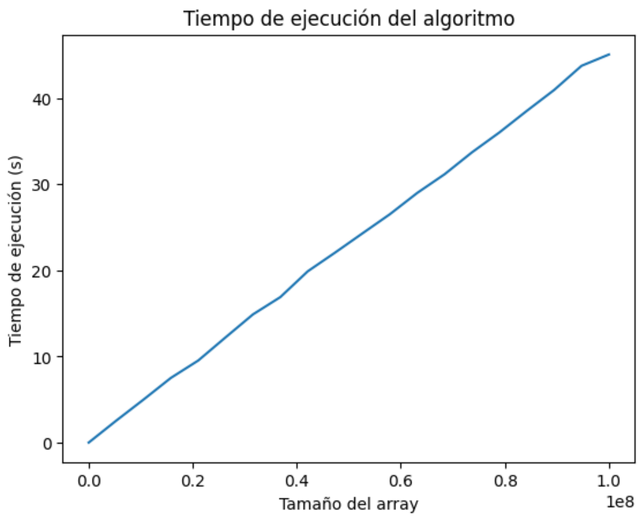
Veamos lo que obtenemos al ejecutar el codigo:

```
martin@Ubuntu:~/TDA/TP1$ python3 main.py ./ejemplos/6.txt
Suma total de monedas: 37
Ganancia de Sophia: 26
Ganancia de Mateo: 11
Movimientos: Última moneda para Sophia; Primera moneda para Mateo; Última moneda para Sophia; Primera moneda para
 Mateo; Última moneda para Sophia; Última moneda para Mateo
```

6. Medición empírica

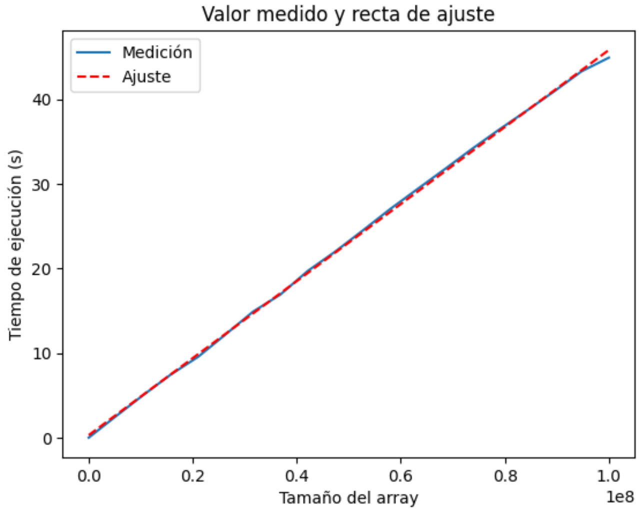
Para comprobar empíricamente la complejidad $\mathbf{O(n)}$ del algoritmo, se decidió ejecutar el mismo con distintos tamaños de entrada y medir el tiempo de ejecución. Se generaron muestras de tamaño n, las cuales varían desde 100 hasta 100 millones.

Para cada muestra se registró el tiempo de ejecución, obteniendo el siguiente gráfico:



A simple vista se puede observar un crecimiento lineal. Para confirmar esto, vamos a ajustar los datos a una recta mediante cuadrados mínimos. Esto lo realizamos con Python y la función *optimize.curve_fit* de la biblioteca *scipy*.

Obtenemos que el gráfico se puede ajustar a la recta $y = 4{,}54e^{-07}x + 0{,}25$, con un error cuadrático medio de 0.06. Por lo tanto, podemos verificar lo que ya vimos en la seccion 3, que el orden es lineal $\mathbf{O(n)}$.



7. Conclusiones

Pudimos observar y verificar lo siguiente:

- Utilizando un algoritmo greedy, pudimos resolver nuestro problema buscando iterativamente optimos locales, hasta finalmente alcanzar un optimo global: que Sophia gane el juego.
- Tras realizar un analisis de complejidad, tanto haciendo un seguimiento del algoritmo planteado, como realizando pruebas empiricas, se concluyo que el algoritmo es de orden lineal: O(n).
- Al realizar pruebas, tanto las otorgadas por la catedra como las nuestras, se pudo llegar a la conclusion de que no importa la variabilidad de los valores de las monedas, siempre se llega al optimo global.

En conclusion, el presente trabajo permitio afianzar los conocimientos adquiridos en la materia de una manera practica, donde desarrollamos un algoritmo greedy para resolver el problema planteado, con una complejidad lineal, alcanzando de esta manera su optimo global (Sophia siempre gana el juego).