Teoría de Algoritmos
(75.29) Curso Buchwald - Genender

Trabajo Práctico 0
Lineamientos sobre informes

12 de noviembre de 2024

Alan Turing
101010

Barbara Liskov
111111

1. Introducción

Los hermanos Sophia y Mateo comienzan un juego creado por su padre, cuyo objetivo es obtener el mayor valor de monedas de una fila dispuesta. En este juego, los jugadores pueden tomar monedas únicamente desde el inicio o el final de la fila. Ambos hermanos se turnan para seleccionar una moneda en cada ronda.

Sophia es muy competitiva y está decidida a ganar, a pesar de que Mateo no entiende comó funciona el juego. A lo largo de la partida, Sophia intentará ayudar a Mateo, aunque su objetivo principal será hacerlo perder y asi, quedarse con la victoria.

2. Análisis del problema

El problema consiste en modelar una estrategia para que Sophia, elija las monedas de tal forma que tenga mayor valor acumulado que las de Mateo. Para esto, consideramos que hay n monedas dispuestas en una fila, numeradas de 0 a $n-1$, y que cada jugador puede tomar una moneda ya sea del inicio o del final de la fila en su turno.

Dado que Sophia puede tomar una moneda tanto en su turno como determinar cuál tomará su hermano Mateo en el siguiente turno, ella seleccionará la moneda de mayor valor en su turno, y durante el turno de Mateo, elegirá la opción que le deje la moneda de menor valor.

Con esto, garantizamos que al final del juego, la suma de los valores acumulados de las monedas que eligió Sophia sea mayor que las de Mateo.

2.1. Estrategia de Sophia

Sophia busca maximizar el valor acumulado en cada uno de sus turnos. La elección de Sophia se puede formalizar como:

$$S(i,j) = \text{máx}\,(v_i, v_j)$$

2.2. Estrategia de Mateo

Mateo, en su turno, seleccionará la moneda que tenga menor valor. La elección de Mateo se puede representar como:

$$M(i,j) = \text{mín}(v_i, v_j)$$

2.3. Notación

Donde:

- $S(i,j)$ representa el valor máximo que puede obtener Sophia al elegir entre las monedas posicionadas en i y j.
- $M(i,j)$ representa el valor mínimo que puede obtener Mateo al elegir entre las monedas posicionadas en i y j.
- v_i y v_j son los valores de las monedas en las posiciones i y j respectivamente.

3. Algoritmo Greedy

Proponemos el siguiente algoritmo greedy, que se basa en aplicar una regla sencilla en cada iteración. Esta regla establece que en cada turno, Sophia seleccionará la moneda de mayor valor disponible, mientras que Mateo elegirá la moneda de menor valor disponible.

Al seguir esta regla, se busca obtener un mayor valor acumulado de las monedas obtenidas por Sophia en comparación con las monedas de Mateo en cada paso del juego, lo que representa un óptimo local. Si esta regla se aplica de manera consistente hasta que todas las monedas hayan sido seleccionadas, se logrará un óptimo global en términos de la suma total de los valores de las monedas acumuladas por Sophia al final del juego.

3.1. Implementacion

Este algoritmo alterna turnos entre Sophia y Mateo, determinados por el booleano `es_turno_sophia`, quienes eligen entre las monedas en los extremos de la deque `monedas` (representadas como `primera_moneda` y `ultima_moneda`).

En la función `agarrar_moneda`, se comparan los valores de estas dos monedas: Sophia elige la de mayor valor y Mateo la de menor, mientras que la moneda no elegida se reintegra a `monedas`. La elección realizada se guarda en la lista `resultados` ya sea la primera o última moneda elegida.

Este proceso continúa hasta que solo queda una moneda, en cuyo caso se efectúa el último turno. Si desde el inicio solo hay una moneda, esta se considera en el primer y único turno.

```
1
2  primera_moneda_sophia ="Primera moneda para Sophia"
3  ultima_moneda_sophia = " ltima  moneda para Sophia"
4  primera_moneda_mateo = "Primera moneda para Mateo"
5  ultima_moneda_mateo = " ltima  moneda para Mateo"
6
7  def agarrar_moneda(monedas, resultado, es_sophia):
8      primera_moneda = monedas.popleft()
9      ultima_moneda = monedas.pop()
10
11     if es_sophia: # Turno de Sophia, elije siempre la mayor
12         if primera_moneda > ultima_moneda:
13             resultado.append(primera_moneda_sophia)
14             monedas.append(ultima_moneda)
15         else:
16             resultado.append(ultima_moneda_sophia)
17             monedas.appendleft(primera_moneda)
18     else: # Turno de Mateo, elije siempre la menor
19         if primera_moneda > ultima_moneda:
20             resultado.append(ultima_moneda_mateo)
21             monedas.appendleft(primera_moneda)
22         else:
23             resultado.append(primera_moneda_mateo)
24             monedas.append(ultima_moneda)
25
26 def algoritmo_greedy(monedas):
27     resultado = []
28     es_turno_sophia = True
29     while len(monedas) >= 2:
30         agarrar_moneda(monedas, resultado, es_turno_sophia)
31         es_turno_sophia = not es_turno_sophia  # Cambio de turno
32
33     # Ultimo turno
34     if es_turno_sophia:
35         resultado.append(ultima_moneda_sophia)
36     else:
37         resultado.append(ultima_moneda_mateo)
38
39     return resultado
```

3.2. Complejidad

Observamos que nuestro algoritmo, en `algoritmo_greedy`, itera sobre todas las monedas disponibles. Si llamamos n a la cantidad de monedas, obtenemos una complejidad temporal de $O(n)$ para esta parte del algoritmo, ya que solo hay asignaciones y condicionales adicionales restantes, los cuales no aumentan la complejidad.

En cada iteración del algoritmo principal, se invoca a la funcion `agarrar_moneda` que tambien tenemos que analizar su complejidad.

En esta función, independientemente de si es el turno de Sophia o de Mateo, se extraen las monedas de los extremos y se comparan, seguidas de operaciones de almacenamiento y reinserción.

Cabe mencionar que las operaciones `append` y `appendleft` de una deque, así como `append` en una lista de Python, tienen complejidad $O(1)$, por lo que no afectan la complejidad temporal general.

Uniendo todo esto, concluimos que la complejidad total es $O(n)$.

3.3. ¿Es óptimo?

A continuación, demostraremos que el algoritmo es óptimo mediante el principio de inducción.

Sean S (Sophia), M (Mateo) y k la cantidad de turnos de cada uno:

Proposición: En cada turno t_i, S toma la moneda de mayor valor disponible entre los dos extremos, y esta moneda siempre será mayor que la moneda que M pueda tomar en el siguiente turno.

Esto implica que todo el valor acumulado de las monedas que recoge S será mayores que la de M, independientemente del orden en el cual se presenten las monedas (exceptuando el caso en que todos los valores sean iguales).

Caso base (primer turno): En el primer turno, S elige entre las dos monedas de los extremos. Como S siempre toma la moneda de mayor valor en su turno, elige esa misma. Como M tomará una moneda en el siguiente turno, sera la otra moneda de los extremos (que es menor en comparación con la que tomó S). Por lo tanto, la proposición se cumple en este turno inicial.

Paso inductivo: Supongamos que hasta el turno k, la proposición se cumple: cada moneda tomada por S ha sido mayor que cualquiera de las monedas que M ha tomado en cada turno.

Ahora, en el turno $k+1$, S debe elegir nuevamente entre las dos monedas de los extremos, y siguiendo la regla greedy, tomará la de mayor valor. En el siguiente turno de M elegirá la moneda de menor valor entre los dos extremos restantes. Esta elección de M garantiza que su moneda es menor que la moneda que acaba de tomar S.

Por lo tanto, la proposición se mantiene para el turno $k+1$: la moneda tomada por S sigue siendo mayor que la de M.

De esta forma, se concluye por inducción que siempre S tendrá mayor valor acumulado de monedas en comparación con M, independientemente del orden de las monedas (excepto cuando todas las monedas sean de igual valor).

4. Mediciones

Aca irian las mediciones del trabajo :)

5. Conclusiones

Acá irían las conclusiones de todo nuestro trabajo :)