Índice general

Proyecto Final
Diseño y Análisis de Algoritmos

Belsai Arango Hernández
Daniela Rodríguez Cepero

Universidad de La Habana,
San Lázaro y L. Plaza de la Revolución, La Habana, Cuba
https://github.com/Daroce1012/DAA-problems/tree/Presupuesto-de-la-FEU
http://www.uh.cu

1. Problema

Presupuesto de la FEU.

Deborah es la nueva secretaria de la FEU de MATCOM y una de sus nuevas tareas consiste en hacer un reporte de las ganancias y gastos de los últimos meses. Para cada mes tiene un número entero. Un número positivo significaría que sobró dinero, ese mes, mientras que uno negativo indica que se perdió dinero. El decano quiere saber la suma de ganancias o gastos de algunos meses consecutivos, pero con el apuro no dijo cuántos meses consecutivos ni a partir de qué mes. Por supuesto, Deborah, en un acto de agilidad mental (no corrupción), decide encontrar las secuencias de k meses consecutivos en los que el balance de dinero sea positivo.

Deborah lleva n meses como secretaria y esos son los meses de los que es responsable. A partir del mes $\frac{n}{2}$, fue que logró acostumbrarse al trabajo nuevo y por tanto, a partir del mes siguiente a ese, el reporte de gasto o ganancia se mantuvo constante (siempre fue el mismo).

Ayude a Deborah con un algoritmo que encuentre un K tal que todas las secuencias de meses de tamaño k posibles, tengan balance positivo.

2. Solución

Se tienen n meses y se desea conocer un k, para el cual la suma de todas las secuencias en n de tamaño k son positivas. Conociendo que a partir de $\frac{n}{2}$ el reporte por mes se mantuvo constante.

Para la solución del mismo primero se obtiene un array con la suma acumulativa de las ganancias o gastos, donde la posición i del array representa la suma acumulativa de todos los elementos desde la posición i hasta la posición n.

Luego de esto podemos decir que si la posición 0 del array de las sumas acumulativas es mayor que cero entonces la suma de los gastos o ganancias de los n meses representan una ganancia y por tanto n sería una solución válida.

En caso contrario habría que encontrar un k menor que n. Para eso empezamos a recorrer el array de suma acumulativa desde la posición $n - \frac{n}{2}$ hasta la posición 0 y se guardan los posibles k , que serían los índices en el array que contengan un valor mayor o igual a cero. Es decir para la posición i del array k sería $n - \frac{n}{2} - i$ puesto que el recorrido se hace en orden inverso.

En caso de no encontrar un posible k, finaliza el algoritmo.

Luego de obtener los posibles k que satisfacen el problema se decide comprobar si son válidos o no. Para eso por cada k se verifica si la suma de las ganancias o pérdidas de todas las secuencias de ese tamaño son positiva en caso de serlo se devuelve ese k, si se encuentra al menos una que sea negativa entonces se pasa al próximo k. Así hasta el último k posible.

3. Correctitud

Para la demostración de la correctitud del algoritmo se comenzará desmostrando como con el array de suma acumulativa se pueden descartar posibles k.
Demostración:
Sea N el array que contiene las ganancias o pérdidas de n meses y S el array de sumas acumulativas de n. Supongamos que existe un k que satisface el problema y que la posición k-1 de S contiene un valor negativo.

Por definición de array de suma acumulativa tenemos que la pos i contiene la suma de todos los elementos desde la posición inicial hasta i, incluido i, por tanto en i se guarda la suma de i+1 elementos consecutivos.

Sin pérdida de generalidad escojamos i = k-1, por datos sabemos que esa posición guarda un valor negativo y como pertenece a S, i guarda la suma de k elementos. Luego se encontró una secuencia de tamaño k cuya suma de elementos es negativa. Pero k es válido, entonces la suma de las ganancias o pérdidas de los meses de cada secuencia de tamaño k es positivas y por tanto hemos llegado a una contradicción pues se encontró una secuencia para la cual la suma de las ganancias o pérdidas de los meses es negativa. Por tanto si k es válido entonces la suma acumulativa de k elementos tiene que ser positiva.

Si existe un k válido entonces $k > \frac{n}{2}$ ó $k = \frac{n}{2}$.
Demostración:

Sea la suma de todos los elementos del array N , negativa y sea el valor constante de los últimos $\frac{n}{2}$ meses $\alpha > 0$. Supongamos que existe k diferente de n y $k < \frac{n}{2}$ solución del problema.
Si k es solución del problema entonces la suma de los elementos de todas las secuencia de tamaño k son positivas y por tanto la suma de todas las secuencias, como $k < \frac{n}{2}$, entonces queda un conjunto de elementos de tamaño menor que k que tienen como valor α, como la suma de todos los elementos del array es negativa, entonces la suma de esos elementos es negativa, pero todos esos elementos son iguales y positivos por los datos, entonces la suma es positiva y hemos llegado a una contradicción. Por tanto $k > \frac{n}{2}$ ó $k = \frac{n}{2}$.

Sea la suma de todos los elementos del array N , negativa y sea el valor constante de los últimos $\frac{n}{2}$ meses $\alpha < 0$. Supongamos que existe k diferente de n y $k < \frac{n}{2}$ solución del problema.
Como $k < \frac{n}{2}$, entonces siempre se puede escoger un conjunto de elementos de tamaño menor que k que tienen como valor α y como $\alpha < 0$ entonces la suma es negativa y por tanto existe una secuencias de tamaño k cuya suma de sus elementos es negativa.
Por tanto $k > \frac{n}{2}$ ó $k = \frac{n}{2}$.

Hasta ahora se ha demostrado que si existe un k que satisface el problema, este es mayor o igual que $\frac{n}{2}$ y que la suma acumulativa de k elementos tiene que ser positiva sino k no es válido.

Luego de tener el array de sumas acumulativas como se sabe que $k > \frac{n}{2}$ se hace un recorrido a partir del índice que contiene la suma de los $\frac{n}{2}$ elementos hasta el de la suma de los n elementos. Por este recorrido se van guardando los índices de lo que contienen suma acumulativa positiva puesto que se demostró que de no ser así no son válidos esos k que representan. En caso de no encontrar ningún k el algoritmo termina aquí.

Por cada k obtenido del recorrido en el array de la suma acumulativa se comienza un proceso de validación que consiste en comprobar que la suma de los elementos de cada secuencia de tamaño k es positiva. Para obtener la suma de cada intervalo de tamaño k se utiliza la suma acumultiva a partir del índice i=0 que representa a k = n y se itera hasta que k+i sea mayor que n o lo que es lo mismo hasta la posición n-k. No tiene sentido continuar luego de ese índice ya que no se encontrará una secuencia de tamaño k sino una menor y por tanto no nos interesa analizarla. Luego de esta iteración habremos obtenido la suma de los elementos de cada secuencia de tamaño k.

Por cada índice se obtiene la suma del k que representa y se le resta el valor que contiene la pos i+k, puesto que al restar el valor que contiene esa posición estaríamos eliminando las suma del resto de los elementos que no pertenecen a la secuencia de tamaño k que inicia en i y obtendríamos la suma de los k elementos consecutivos del array original iniciados por i. Esto es posible ya que la suma acumulativa se hizo en orden inverso, es decir en vez de i representar la suma de los elementos de 0 hasta i, representa la suma de los elementos desde i hasta n o de n hasta i y por tanto la suma de k elementos iniciados por i sería la suma desde i hasta i+k lo que es lo mismo que la suma de todos los elementos desde i hasta n (guardado en i) menos la suma de los elementos desde i+k hasta n(guardados en i+k) en vez de el valor que guarda i+k menos el valor que guarda i.

Esto se aplica a cada i durante la iteración, así con todos los k posibles, si se detecta un resultado negativo en la restas durante la iteración se descarta ese k, puesto que se ha detectado que la suma de una secuencia de tamaño k tiene valor negativo, entonces k no satisface el problema y por tanto se inicia con el próximo k. En caso contrario, y si se llega al último índice n-k y su resta es positiva entonces eso significa que la suma de todos los elementos de cada secuencia de tamaño k ha sido positiva y por tanto k resuelve el problema y es una solución válida, por tanto se retorna el valor de k.

4. Complejidad Temporal

El algoritmo lee la entrada del problema y se ejecuta una operación de costo constante $O(1)$. Luego se obtiene el array de suma acumulativa a partir del array de entrada, dicha operación tiene un costo de $O(n)$, puesto que se recorre todo el array de entrada y por cada iteración se realiza una cantidad constante de operaciones matemáticas elementales $O(1)$. Seguido se realizan tres operaciones elementales de costo $O(1)$ cada una. Luego se realiza una iteración sobre el array de suma acumulativa para guardar posibles k que solucionan el problema, estos se guardan en una lista, el orden de realizar dicha operación es $O(n)$. Seguido a esto se recorre la lista de posibles k, siendo m la cantidad de posibles k en la lista, esto tiene un costo $O(m)$ y por cada k se itera sobre el array de suma acumulativa para obtener la suma de los elementos de cada secuencia de tamaño k para verificar que esta sea mayor que cero y continuar o en caso contrario descartar a ese k, esto tiene un costo $O(\frac{n}{2})$. Por regla de la multiplicación dicho proceso tiene una complejidad temporal de $O(mn)$. Por regla de la suma la complejidad temporal del algoritmo es de $O(mn)$.

5. Tester

```
def test_solution(months):
    n = len(months)
    list_k = []
    for k in range(1,n+1):
        for i in range(0,n-k+1):
            result = sum(months[i:i+k])
            if result < 0:
                break
            elif i + k >= n:
                list_k.append(k)
                break
    return list_k
```

El método tiene como parámetro de entrada el array con los meses y sus ganancias.
líneas 1-2: se guarda la cantidad de meses y se inicializa la lista que guarda los k positivos.
líneas 3 - 4 se toma por cada k posible (desde 1 hasta n) todas las secuencias válidas, desde 0 hasta $n-k+1$, no se toma desde 0 hasta n porque se cumple que a partir del índice que tiene el mismo valor de k en adelante no existe ninguna secuencia de tamaño igual o mayor.
línea 5 : se obtiene la suma de la secuencia desde la posición en la que se encuentra hasta la posición más el k actual.
líneas 6 - 10: se verifica si el resultado es negativo, si es así, el k actual no es positivo por lo tanto no es válido.
En otro caso comprueba si el valor i+k no se pasa del rango del array, si se cumple significa que ya no existe otra secuencia de ese tamaño por lo tanto se obtiene un k válido.
línea 11: se retorna todos lo k válidos.

6. Ocurrencia de Ideas

Para la solución del problema la primera idea que se nos ocurrió fue la utilizada en el tester con la diferencia de que en vez de retornar todos los k, retornara el primero que encontrara. Luego nos dimos cuenta de que habían datos que no estábamos utilizando y empezamos a pensar en como podríamos utilizarlos, con los días nos dimos cuenta de que si existe un k válido entonces $k > \frac{n}{2}$ ó $k = \frac{n}{2}$, pero solo teníamos la seguridad cuando el valor constante de los últimos $\frac{n}{2}$ meses $\alpha < 0$ por tanto decidimos abandonar la idea. Luego mientras el profesor Piad daba su conferencia de dinámica se nos ocurrió la idea de utilizar la suma acumulativa para descartar posibles k, pero el algoritmo seguía teniendo un costo alto. Al cabo de los días luego de seguir pensando en como usar los datos se nos ocurrió como demostrar que si existe un k válido entonces $k > \frac{n}{2}$ ó $k = \frac{n}{2}$ para cuando $\alpha > 0$, entonces decidimos mezclar ambas ideas para podar aún más el algoritmo quitandonós $\frac{n}{2}$ iteraciones. Tiempo despúes mientras se escribía el código del algoritmo si teníamos la suma acumulativa en orden inverso no tendríamos que indexar en el array dos veces para obtener la suma xq se sabría que el número a restar siempre sería un múltiplo de α y por último decidimos juntar todas estas ideas y así fue como se obtuvo el algoritmo final.