# EL BAR

MARCOS M. TIRADOR AND LEANDRO RODRÍGUEZ LLOSA

## 1. Enunciado del Problema

**Problema 1.1.** En un bar se controla la entrada de personas diaria durante una cantidad determinada de días. En lo adelante, la administración del bar se refiere a la cantidad de personas que entran, en relación a la media durante ese período, por ejemplo si un día entran 5 personas más que la media calculada se registra como 5, si entran 10 personas menos que la media ya calculada se registra $-10$. Esta media no se actualiza.

La administración del bar tiene un registro de la asistencia durante $n$ días consecutivos, que ha sido registrada de la forma antes mencionada, y se ha visto que la asistencia al bar en la segunda mitad analizada fue la misma cada día.

El dueño del bar va a hacerle una auditoría a la administración para saber si todo marcha bien. La auditoría se lleva a cabo de la siguiente forma: El administrador le dirá un número $k$, y el dueño obtendrá la suma de las $n-k+1$ posibles secuencias de $k$ días consecutivos. Esto es, para todo $i$ entre 1 y $n-k+1$, obtendrá la suma:

$$a_i + a_{i+1} + ... + a_{i+k-1}.$$

Por ejemplo si se registró como asistencia $[-1, 0, 1, 2, 2]$ (nótese que los dos últimos elementos conforman la segunda mitad, tomando parte entera por debajo, y son todos iguales), para $k = 3$, el dueño tendrá como suma los números 0, 3 y 5.

Si todas las sumas obtenidas resultan positivas con respecto a la media anterior el dueño concluirá que el bar marcha bien, de lo contrario, despide al administrador.

El administrador lo contrata a usted para que lo ayude, como científico que se considera, a determinar qué valor de $k$ debe elegir el administrador para que el dueño del bar no lo despida.

**Problema 1.2.** Se tiene una lista de $n$ números enteros $a_1, a_2, \ldots, a_n$. Se desea saber si existe un $1 \leq k \leq n$ tal que la suma

$$a_i + a_{i+1} + \cdots + a_{i+k-1}$$

es positiva para todo $1 \leq i \leq n-k+1$.

*Date*: April 3, 2023.

## 2. Solución Inocente

Como podemos ver, una vez formulado el problema, podemos encontrar una solución simplemente explorando todos el espacio de búsqueda. Veremos a continuación dicha solución.

**Solución 2.1.** Podemos tomar cada entero positivo $1 \leq k \leq n$ y comprobar si existe un subarray de tamaño $k$ del array dado con suma no positiva, teniendo en caso contrario que $k$ es solución. Si $k$ no es solución para ningún valor entre 1 y $n$, entonces la solución no existe y damos $-1$ como resultado.

Para un $k$ fijado, podemos comprobar si este es solución haciendo todas las posibles sumas de $k$ valores consecutivos del array dado.

**Proposición 2.2.** *La solución dada en 2.1 es correcta.*

*Demostración:* Esto se puede ver directamente del hecho de que exploramos todas las sumas posibles de $k$ valores consecutivos para cada valor de $k$ posible. ■

**Complejidad temporal:** La complejidad temporal de esta solución es $\Theta(n^3)$. Tenemos $n$ valores posibles de $k$ a explorar, y para cada uno de ellos tenemos $n-k+1$ subarrays de tamaño $k$. Por tanto la cantidad de operaciones es del orden de

$$\begin{aligned}\sum_{k=1}^{n} k(n-k+1) &= \sum_{k=1}^{n} kn - \sum_{k=1}^{n}(k^2-k) = n^2(n+1)/2 - \sum_{k=1}^{n}(k^2-k)\\ &= n^2(n+1)/2 - n(n+1)(2n-1)/6 + n(n+1)/2\\ &= n^3(1/2-1/3) + n^2(1-1/6) + n(1/6+1/2) = \Theta(n^3).\end{aligned}$$

Sin embargo nos podemos quitar el factor $k$ de acumular el array de tamaño $k$ si precalculamos en $O(n)$ la suma del prefijo de tamaño $i$, para cada $1 \leq i \leq n$. En este caso la solución sería entonces $\Theta(n^2)$.

**Complejidad espacial:** Necesitaremos a lo sumo 2 arreglos de tamaño $n$ y algunos otros valores enteros, lo cual no es más que $O(n)$ en complejidad espacial.

## 3. Solución Propuesta

En esta sección presentaremos una mejor solución del problema que la inocente. Además, demostraremos su correctitud y calcularemos su complejidad temporal. Finalmente demostraremos que es óptima.

**Solución 3.1.** Sea $a_1, a_2, \ldots, a_n$ la lista dada. Si $a_n$ es no negativo, la solución es $n$ si $a_1 + \cdots + a_n$ es positivo y $-1$ (no existe el $k$ buscado) en otro caso.

Supongamos entonces que $a_n$ es negativo. Lo que hacemos es acumular para cada $1 \leq i \leq n$ el sufijo que comienza en $i$ de la lista dada, esto es $s_i := a_i + a_{i+1} + \cdots + a_n$. Si $s_i$ es no positivo, descartamos como posibles valores de $k$, todos los números entre $n - i + 1 - \min(\lfloor s_i/a_n \rfloor, \lfloor n/2 \rfloor)$ y $n - i + 1$. Luego si existe un $\lfloor n/2 \rfloor < k \leq n$ que no haya sido descartado, este es la solución. En otro caso no existe la solución.

**Teorema 3.2.** *La Solución 3.1 es correcta.*

*Demostración:* Primero demostremos que si $k$ es solución, con $k \leq \lfloor n/2 \rfloor$, entonces $2k$ también. Supongamos por el contrario que existe un $1 \leq i \leq n-2k+1$ tal que $a_i + \cdots + a_{i+2k-1} \leq 0$. Entonces debe ocurrir que $a_i + \cdots + a_{i+k-1} \leq 0$ o $a_{i+k} + \cdots + a_{i+2k-1} \leq 0$, ya que de lo contrario la suma entera fuera positiva. Sin embargo, ambas sumas están compuestas por $k$ elementos consecutivos lo cual contradice que $k$ es solución. Por tanto si existe una solución entonces existe una tal que $k > \lfloor n/2 \rfloor$.

Veamos que si $a_n \geq 0$ entonces existe una solución si y solo si $s_1 := a_1 + \cdots + s_n > 0$. La implicación reversa es evidente. Supongamos entonces que existe una solución $k$. Entonces la suma $a_1 + a_2 + \cdots + a_k > 0$. Dado que por el resultado anterior podemos asumir que $k > \lfloor n/2 \rfloor$, entonces $a_{k+1} = a_{k+2} = \cdots = a_n \geq 0$. Por tanto $a_1 + a_2 + \cdots + a_n = (a_1 + a_2 + \cdots + a_k) + a_{k+1} + a_{k+2} + \cdots + a_n > 0 + (n-k)a_n \geq 0$, lo cual concluye nuestra demostración para este caso.

Para el otro caso primero demostremos que si existe una $k > \lfloor n/2 \rfloor$ tal que $k$ no es solución, entonces $k$ fue descartado en la solución. Sea $i$ tal que $a_i + \cdots + a_{i+k-1} \leq 0$. Veamos que

$$\begin{aligned} -s_i/a_n = -\frac{a_i + \cdots + a_{i+k-1} + a_{i+k} + \cdots + a_n}{a_n} &\leq -\frac{a_{i+k} + \cdots + a_n}{a_n} \\ &= -\frac{(n-i-k+1)a_n}{a_n} \\ &= (k+i-n-1). \end{aligned}$$

Por tanto podemos ver que

$$\begin{aligned} l_i = \max(n-i+1-\lfloor s_i/a_n \rfloor, n-i+1-\lfloor n/2 \rfloor) &= \max(n-i+1-s_i/a_n+\delta, n-i+1-\lfloor n/2 \rfloor) \\ &\leq \max(n-i+1+(k+i-n-1)+\delta, \lfloor n/2 \rfloor + 1) \\ &= \max(k+\delta, \lfloor n/2 \rfloor + 1) < k+1, \end{aligned}$$

donde $s_i/a_n = \lfloor s_i/a_n \rfloor + \delta$. Por otro lado $r_i = (n-i+1)$, de donde obtenemos $l_i \leq k \leq r_i$. Podemos decir entonces que este valor de $k$ fue correctamente descartado.

Solo resta demostrar que si existe un valor de $k > \lfloor n/2 \rfloor$ que es solución, entonces este valor no es descartado por el algoritmo. Supongamos por el contrario que existe un $i$ tal que $l_i \leq k \leq r_i$. Veamos primero que $i + l_i = i + n - i + 1 - \min(\lfloor s_i/a_n \rfloor, \lfloor n/2 \rfloor) \geq n + 1 - \lfloor n/2 \rfloor > \lfloor n/2 \rfloor$ y por tanto $a_{i+l_i} = a_{i+l_i+1} = \cdots = a_n < 0$. Tenemos entonces que

$$\begin{aligned} a_i + a_{i+1} + \cdots + a_{i+k-1} = s_i - (a_{i+k} + \cdots + a_n) &\leq s_i - (a_{i+l_i} + \cdots + a_n) \\ &= s_i - (n - l_i - i + 1)a_n \\ &\leq s_i - \big(n - i + 1 - (n - i + 1 - \lfloor s_i/a_n \rfloor)\big)a_n \\ &= s_i - \lfloor s_i/a_n \rfloor a_n \leq 0, \end{aligned}$$

de donde arribamos a una contradicción, ya que existe un subarray de tamaño $k$ de suma no positiva, pero $k$ era solución.

■

**Complejidad temporal:**

En el caso de que $a_n$ sea positivo solamente debemos acumular la lista, lo cual es $\Theta(n)$. En el otro caso, en cada iteración (cada valor de $i$ desde 1 hasta $n-k+1$) podemos hallar el valor de $s_i$ en $O(1)$ haciendo $s_i = s_{i-1} - a_{i-1}$, donde $s_1$ se calcula en $\Theta(n)$ como en el caso anterior.

Sean $l_i$ y $r_i$ tales que en la iteración $i$ se determinó que debemos descartar como posibles valores de $k$ todos los enteros entre $l_i$ y $r_i$. Entonces creamos un array de tamaño $n+1$, inicialmente con todos los valores iguales a 0. Luego por cada $i$, marcamos la posición $l_i$ del array con valor 1, y la posición $r_i + 1$ con valor $-1$. Luego pasamos acumulando el array desde la posición $\lfloor n/2 \rfloor + 1$ hasta la $n$ y si en algún momento la suma acumulada es 0, entonces esa posición corresponde a un valor que

no fue descartado. Note que esto es similar al conocido *factor de balance*. Además véase que en cada iteración $i$, se hacen un número constante de operaciones, lo cual representa una complejidad total de $\Theta(n)$ y luego pasar por el array del factor de balance es $\Theta(n)$ también. Finalmente la complejidad temporal del algoritmo es $\Theta(n)$.

**Complejidad espacial:** Necesitaremos a lo sumo 3 arreglos de tamaño $n$ y algunos otros valores enteros, lo cual no es más que $O(n)$ en complejidad espacial.

Como resultado final de nuestro trabajo veremos que no existe una solución mejor al problema estudiado.

**Teorema 3.3.** *La complejidad de la Solución 3.1 es óptima.*

*Demostración:* Para esto demostramos que para cualquier algoritmo de solución del problema, se debe realizar al menos una operación (aunque sea de lectura) con cada uno de los primeros $\lceil n/2 \rceil$ elementos. Suponga por el contrario que existe un algoritmo solución que ignora a algún elemento de los primeros $\lceil n/2 \rceil$ de la lista lista, o sea que puede dar una respuesta sin conocer el valor de una determinada posición del array. Supongamos que la respuesta del algoritmo es $-1$ para una entrada determinada donde ignora el número en la posición $i$. Esto significa que sin importar el valor de $a_i$ la solución seguirá siendo $-1$. Sin embargo, si $a_i > \left|\sum_{j \neq i} a_j\right|$, entonces $k = n$ es claramente una solución, lo cual es una contradicción. Por tanto el algoritmo nunca podrá determinar cuando la solución es $-1$ sin analizar los primeros $\lceil n/2 \rceil$ elementos del array, de donde cualquier algoritmo que resuelva el problema tiene complejidad $\Omega(n)$, y concluimos que la Solución 3.1 es óptima. ■

Facultad de Matemática y Computación, Universidad de La Habana, Ciudad de La Habana, Cuba
*Email address*: marcosmath44@gmail.com

Facultad de Matemática y Computación, Universidad de La Habana, Ciudad de La Habana, Cuba
*Email address*: leandro_driguez@outlook.com