Problema 2: Uniformando la secuencia

Amanda Marrero Santos

Loraine Monteagudo García

Carlos Rafael Ortega Lezcano

Universidad de La Habana

2018-2019

Resumen

En este documento se analizan varios intentos de solución de un problema que consiste en, dado una lista de l números, determinar la cantidad de formas de igualar todos los números a un valor h sumándole solamente 1 a intervalos de números, con la restricción de que un mismo intervalo no puede ser seleccionado 2 veces. Después de demostrar varias propiedades que deben cumplir estas listas para obtener un resultado válido se desarrolla un algoritmo de fuerza bruta que consiste en generar todos los posibles intervalos que conforman un conjunto dado y después se seleccionarán aquellos a los que se les aplicarán transformaciones, comprobando posteriormente si son solución con un costo de a lo sumo $O(n^2 * 2^{n^2})$. Luego se intentaron varias vías dinámicas de resolver el problema, la dinámica final resultó tener un costo demasiado alto para ser considerado. Finalmente, se desarrolla un algoritmo que, a pesar de ser fallido, se cree que con un enfoque similar se podría resolver el problema.

I. Análisis del Problema

El algoritmo recibirá como entrada:

- Una lista l de n de números.

$$l = \boxed{l_1}\boxed{l_2}\boxed{l_3}\boxed{...}\boxed{l_{n-1}}\boxed{l_n}$$

- Un número h

Se tiene que una transformación consiste en, dado un intervalo $[l, r]$ sumarle 1 a todos sus elementos. Pero existe una restricción: si se realiza una transformación al intervalo $[l_1, r_1]$ y $[l_2, r_2]$ entonces $l_1 \neq l_2$ y $r_1 \neq r_2$, es decir, los extremos deben ser distintos, por lo que nunca se le podrá realizar una misma transformación a un intervalo.

Se quiere generar entonces como salida:

- La cantidad k de formas distintas en que se puede transforma la lista de números para que todos sus elementos sean iguales a h.

Demostraremos a continuación los siguientes lemas que surgieron interpretando las propiedades del problema:

Notemos que existen casos para los cuales no habrán maneras de llegar a la solución. Por ejemplo, si existe un número $l_i \in l$ que sea mayor que h, entonces no existirá forma de transformar la lista en h ya que solo se pueden sumar elementos en la lista. La demostración de esto es trivial: la única operación válida es sumar 1, por lo que después de un conjunto de transformaciones para todo valor l_i, este tendrá un valor $k \geq l_i > h$, por lo que nunca se podrá igualar a h.

Lema 1

Dada una lista $l = \{l_0, l_1, ..., l_{n-1}\}$, un término l_i, donde i denota la posición *i-ésima* en la lista, este aparece en subintervalos distintos $(i+1) * (n-i)$ veces.

Dado un indice i de la secuencia de enteros, $0 \leq i \leq n-1$, se desea conocer la cantidad de intervalos que contiene al elemento i, para ello se toman dos conjuntos de la siguiente forma:

- Los elementos de la secuencia desde 0 hasta i, incluyendo a i: $A = \{l_0, l_1, l_2, ..., l_i\}$
- Los elementos a partir de i hasta la ultima posición, incluyendo i: $B = \{l_i, l_{i+1}, ..., l_{n-1}\}$

Ahora observemos que existe una equivalencia entre la cantidad de intervalos que contienen a i y combinar los elementos en A con los de B, por cada elemento que se elija de A este será el extremo izquierdo del intervalo y si se elije un elemento de B este será el extremo derecho. El elemento l_i está en los dos intervalos, ya que él puede ser un extremo también, y notar que el intervalo que lo contiene solo a él también es contado, se representa como $[l_i, l_i]$. Luego, la cantidad de intervalos que contienen a i es la cantidad de formas de relacionar los elementos de A con los de B, resultando: $|A| * |B| = (i+1) * (n-1-i+1) = (i+1) * (n-i)$

Lema 2

Dada una lista $l = \{l_0, l_2, ..., l_{n-1}\}$ y un valor h, donde a cada elemento l_i se le pueden realizar las transformaciones explicadas, si $\exists l_i \in l$ tal que $l_i + (i+1) * (n-1) < h$, entonces no existe manera de convertir los valores de l en h.

Una transformación válida es sumar 1 a cada uno de los elementos de un conjunto de intervalos de l, la máxima cantidad de bloques en los que puede aparecer un elemento $l_i \in l$ es $(i+1) * (n-i)$, por lo que este es la cantidad máxima que se puede aumentar l_i, por lo tanto, si $l_i + (i+1) * (n-1) < h$, entonces l_i nunca alcanzará el valor de h, por lo que la secuencia no se podrá uniformar.

II. Fuerza Bruta

Este algoritmo fuerza bruta consiste en generar todos los posibles intervalos que conforman un conjunto dado, de forma que al seleccionar un subconjunto del total de intervalos y sumarle 1 a sus elementos se logre que todos los elementos de la lista l original alcancen el valor h.

Para simular la creación de los intervalos no se generan todos los números que conforman un intervalo, sino en su lugar se generan todos los posibles pares de elementos del conjunto, representado estos solo los extremos de dicho intervalo.

La selección de los intervalos que podrían formar parte de un grupo solución, se realiza a través de una mascara de bits, donde el hecho de escoger un elemento o no es representado por los bits que identifican a un número, ya que todo número en binario se representa de manera única.

Después de tener todas las agrupaciones posibles, la validez de un subconjunto se determina sumando cada uno de los elementos de la lista l que estén en el rango de los extremos de cada uno de los intervalos; y a continuación se comprueba que todos los integrantes de la lista original l hallan llegado al valor h deseado.

Se lleva además un contador *count* que contará cuantos de los subconjuntos de intervalos formados son válidos. Este contador aumentará de encontrar un conjunto de intervalos que sumen h, ya que se habrá encontrado una manera de transformar los números en h y será el que se retornará como resultado final.

Complejidad

Inicialmente se generan todos los posibles pares de números que conformaran los extremos de los intervalos, como además un intervalo que contiene un solo , en un conjunto de n elementos habrán $\frac{n*(n-1)}{2} + n$ maneras de realizar esto, por lo que en este procedimiento se realizarán $O(n^2)$ operaciones.

Para crear todos los subconjuntos del conjunto total de intervalos se efectúan $O(2^{n^2})$ operaciones, ya que para cada uno de los elementos de los intervalos se exploran 2 opciones: sumarle a sus elementos 1 o no hacerlo, es decir, añadirlo o no al conjunto solución.

Por lo que finalmente, el costo final del algoritmo es $O(n^2 * 2^{n^2})$, determinado por el hecho de recorrer todos los subconjuntos y por cada uno de ellos iterar sobre los intervalos que lo conforman, y después por cada uno de estos últimos adicionar 1 a sus elementos, operación que tiene una complejidad de $O(n^2)$.

Correctitud

La correctitud del algoritmo es muy simple: se exploran todas las posibles opciones, así que entre las vías generadas está la solución. Se generarán conjuntos que no serán solución, lo cual consumirá tiempo, pero la veracidad de la solución se asegura.

III. Otros intentos de solución

I. Primer intento dinámico

Para mejorar el costo de la fuerza bruta se intentó un primer enfoque dinámico, que tendría el siguiente esquema:

dp[i, j]: cantidad de maneras de llevar un intervalo $[i, j]$ a h

Notemos entonces que todo elemento de la diagonal tendría valor 1 si $l_i = h$ o $l_i = h - 1$ y 0 en caso contrario.

La idea de llevar una matriz dp es para iterar por los distintos grupos de intervalos y recurrir a elementos previamente calculados.

El problema con esta idea es que para computar todas las maneras en que, por ejemplo, se calcula $[0, 1]$ tendríamos que determinar: todas las formas de que $[0, 0]$ sea h multiplicado por todas las formas en que $[1, 1]$ sea llevado a este mismo valor adicionado a todas las formas en que $[0, 0]$ sea $h - 1$ más todas las maneras en que $[1, 1]$ sea $h - 1$, notemos que estos últimos valores son totalmente desconocidos para nosotros, con lo cual este enfoque no nos sirvió.

II. Segundo intento dinámico

Basándonos en lo anterior pensamos en una dinámica ahora de tres estados, que cumplirá con lo siguiente:

dp[i, j, k]: cantidad de formas en la que el intervalo $[i, j]$ puede ser llevado a k.

Notemos que dp tendrá tamaño $n * n * (h + 1)$ y que el resultado final estaría en $dp[0, n - 1, h]$

Como caso base, se tendrían las posiciones $dp[i, i, k]$, con $k = \overline{0, h - 1}$. No será posible que estos elementos alcancen k si $k > l_i + 1$, por lo que se retornará 0 en ese caso y 1 en el caso contrario.

La manera de recorrer la matriz sería con el mismo enfoque de la dinámica anterior, es decir, recorreríamos primero los intervalos de tamaño 2, para los cuales calcularíamos la cantidad de formas de llegar a todos los posibles valores de k; luego los de tamaño 3, etc. Por lo cual, los elementos se recorrerán diagonalmente por i, j y verticalmente por k.

Para ver el problema con esta dinámica analicemos cuál sería la dependencia de $[0, 2]$:

$$dp[0,2,0] = dp[0,0,0] * dp[1,1,0] * dp[2,2,0] + dp[0,1,0] * dp[2,2,0] + dp[0,0,0] * dp[1,2,0]$$

Otro ejemplo:

$$\begin{aligned} dp[0,2,2] = {} & dp[0,0,2] * dp[1,1,2] * dp[2,2,2] + dp[0,1,2] * dp[2,2,2] + dp[0,0,2] * dp[1,2,2] + \\ & dp[0,0,1] * dp[1,2,1] + dp[0,1,1] * dp[2,2,1] + dp[0,0,1] * table[1,1,1] * table[2,2,1] \end{aligned}$$

La dificultad de, enfoque es que para determinar la manera de relacionarse los elementos nos queda una expresión complicada. Esta consiste en que para cada valor de la matriz $dp[i, j, k]$ se debe expresar la longitud del intervalo como sumandos. En intervalos de tamaño 1 una primera forma seria: $1 + +1 + ... + 1 = j - i + 1$, en donde esto representa todas las formas multiplicadas en que cada elemento por separado llegue a h, lo que nos conllevaría a hacer además todas las formas en que cada elemento por separado llega a $h - 1$, siempre que $h - 1 >= 0$. Otra manera de dividir en sumandos es: $2 + 1 + ... + 1 = j - i + 1$, lo cual significa todas las formas de que un intervalo de longitud 2 llegue a h multiplicado por todas las formas en que cada elemento que no fue escogido en ese intervalo de tamaño 2 llegue a h, adicionado por todas las formas de que un intervalo de longitud 2 llegue a $h - 1$ por todas las formas en que cada elemento que no fue escogido en ese intervalo de tamaño 2 llegue a $h - 1$.

Por lo tanto, como dicha dependencia resulta bastante costosa resolvimos que sería una mala elección seguir por este camino.

III. Analizando los intervalos

En este enfoque se hará un análisis más exhaustivo de las características del problema para empezar el desarrollo de una posible solución recursiva.

Definición: Sea X un arreglo tal que se cumpla el **Lema 2** con respecto a h, se define $Dif(X, h)$ al arreglo que en su posición i-ésima es: $h - X[i]$

Lema 3

Sea n el número de intervalos seleccionados tal que cada elemento de A se quiere transformar en h, la transformación de $diff(X, h)$ a 0 se puede realizar con los mismos n intervalos

Sea la k-ésima operación para convertir X en h, la única operación válida es sumar 1 a todos los elementos del intervalo, por lo tanto se cumplirá para $i \in [l, r]$, donde $[l, r]$ denota el intervalo seleccionado, que:

$$A_{k+1}[i] = A_k[i] + 1$$

La expresión para $diff_{k+1}(X, h)$ será:

$$diff_{k+1}(X,h) = h - A_{k+1}[i]$$

Sustituyendo, resulta que la operación que corresponderá a aplicar $diff_k(X,h)$ para que se transforme en 0 será restar 1 a los elementos del intervalo:

$$diff_{k+1}(X,h) = h - (A_k[i]+1) = (h - A_k[i]) - 1 = diff_k(X,h) - 1$$

Luego se demuestra que para la misma secuencia de intervalos es posible transformar los elementos de $diff(X,h)$ en 0, restando 1 a los elementos contenidos en cada intervalo

Por lo tanto, apoyándonos en este lema, en el algoritmo que se presentará a continuación en vez de igualar todos los elementos de l a h, se usará el arreglo $diff$ y se transformarán en 0 todos sus elementos.

Notemos que la cantidad de intervalos en los que un elemento debe estar contenido para llegar a h es $diff[i]$. Haciendo uso de esta información se desarrolla un algoritmo casuístico que intenta contar todas las posibles maneras en las que un número puede estar contenido en un intervalo. A pesar de que no se obtuvieron buenos resultados, se cree que constituye un método válido para resolver el problema que necesita ser pulido.

Entonces, algo que podríamos hacer es crear una función que recibirá una posición y un entero k.Este entero representará la cantidad de intervalos que las posiciones hacia la derecha están obligados a cerrar de alguna manera. Esta función computará el número de formas que tiene el intervalo de *pos* a $n-1$ de garantizar cerrar k intervalos.

Definamos las siguientes posibilidades para una determinada posición:

1. Se pueden abrir 1 o varios intervalos.
2. Se pueden cerrar 1 o varios intervalos.
3. Se puede abrir y cerrar a la vez, y esto representaría el intervalo que abre y cierra en tí. Notar que esta opción no se puede hacer más de una vez, pues estaríamos repitiendo intervalos.
4. Se puede no hacer nada, en cuyo caso este elemento ya pertenecerá a todos los intervalos necesarios para sumar h, o bien no será extremo

Expliquemos brevemente como funcionaría la función para una posición i y un k determinado:

- Si $diff[i] = k$: Tendremos 4 opciones:
    1. No podemos abrir intervalos: esto es debido a que nuestra solución estaría en $F(i+1, k+t)$ donde t es la cantidad de intervalos que se decidieron abrir, entonces $F(i+1, k+t)$ retornará todas las formas, garantizando que se van a cerrar $k+t$ intervalos por lo que ahora sería como si el elemento en la posición pos perteneciera a $k+t$ intervalos lo que es una contradicción ya que este solo pertenece a k.
    2. Se pueden cerrar 1 o varios intervalos en pos: el resultado estará en $F(i+1, k-t)$, donde t es la cantidad de intervalos que se decidieron cerrar. Notar que el resultado de $F(i+1, k-t)$ hay que multiplicarlo por t y por k, ya que cada cierre de intervalo puede haber cerrado cualquiera de los k intervalos abiertos.

3. Abrir y cerrar en él: no se puede aplicar este caso ya que la solución estaría en $F(i+1,k)$, como la función me garantiza que ella va a cerrar k intervalos y yo cierro otro más, esto implicaría entonces que pos pertenecerá a $k+1$ intervalos, lo que seria una contradicción.
4. Puedo no hacer nada y delego en $i+1$, el problema de cerrar k intervalos.

- Si $diff[i] < k$: Esto significa que la posición i pertenece a al menos de k intervalos, hagamos entonces lo siguiente: sea $d = k - diff[i]$, cerraremos k intervalos en la posición $i-1$, ya que sino i pertenecería a más intervalos de los que puede. Analicemos las posibles opciones para este caso:
  1. Se puede abrir intervalos: la respuesta estaría bien calculada en $F(i+1,k+t)$, donde t representa la cantidad de intervalos que se deciden abrir en i.
  2. No se pueden cerrar intervalos: de ser así entonces el resultado estaría en $F(i+1,k-t)$, donde t es la cantidad de intervalos que se decidieron cerrar, entonces dicho llamado resolvería el hecho de cerrar k intervalos con lo que el elemento actual aparecería en más intervalos de los que puede estar.
  3. Se podrán abrir y cerrar intervalos en la posición actual: En caso de que $diff[i] - 1 = k$, entonces la solución estará bien calculada en $F(i+1,k)$, es decir, se tendrán k intervalos que contienen a l_i y el que se crea abriendo y cerrando intervalos, entonces la posición i pertenecerá a $k+1$ intervalos. Notar que como se cierra un intervalo, entonces existen k posibles inicios para los cuales este es el otro extremo, luego, $F(i+1,k) * k$ cuenta las posibles formas de crear los intervalos.
  4. No se puede hacer nada: por la misma razón que en el caso 2 el elemento en la posición i aparecerá en más intervalos de los que puede.

- Si $diff[i] > k$: Esto significa que la posición i necesita pertenecer a más intervalos que k, de manera análoga al caso anterior, abriremos en l_i d intervalos con $d = diff[i] - k$.
  1. No se podrá abrir más intervalos, ya que si no, el elemento i-ésimo pertenecerá a más intervalos de los que puede.
  2. Se pueden cerrar intervalos, pero a lo sumo 1 se pueden cerrar en pos, porque sino repetiríamos intervalos, por lo que la solución sería $F(i+1, diff[i]-t) * (k+1)$
  3. No se puede abrir y cerrar en la misma posición, pues entonces la respuesta estará en $F(i+1, diff[i])$, pero como estaríamos abriendo un nuevo intervalo, entonces el elemento i-ésimo formara parte de $diff[i]+1$, intervalos lo que seria una contradicción.
  4. Se puede no hacer nada, entonces la solución está bien calculada en $F(i+1, diff[i])$

Como caso base se tendrá que $i = n-1$, es decir, se llega a la última posición, en ese caso existen distintas posibilidades dependiendo del valor de k, es decir, de la cantidad de intervalos que se deben cerrar:

- Para k = 0:
  - $diff[n-1] = 0$ se retornará 1 ya que si el último elemento no necesita pertenecer a ningún intervalo, entonces no hay nadie que que necesite cerrar un intervalo, por lo que existe una manera de llegar a 0, lo que significa que se creo unos intervalos válidos.
  - $diff[n-1] = 1$ se retornará 1, ya que se puede resolver aplicando la forma 3 (abriendo y cerrando) y de esta manera se mantiene $k = 0$

- En cualquier otro caso se retornará 0, ya que como no se puede cerrar intervalos, no se cumplirá que $diff[i] = 0$.

- Para k = 1:
  - $diff[n-1] = 0$ retorna 0, ya que si no el elemento se incluiría en un intervalo que no le funciona para llegar a h.
  - $diff[n-1] = 1$ retorna 1 ya que se cierra el intervalo, se decrementa $diff[i]$ igualándolo a 0 y se deja el cierre para todo los intervalos que necesiten cerrarse.
  - $diff[n-1] = 2$ retorna 1, ya que se puede disminuir en uno $diff[i]$ y luego cerrar un intervalo, resolviéndose el elemento actual y cumpliendo con el llamado.
  - En otro caso retorna 0, ya que no se puede realizar lo mismo que en el caso anterior, entonces faltarían intervalos a los que el elemento en la posición actual debe pertenecer.
- En otro caso:
  - $diff[n-1]! = k$ retorna 0 ya que el elemento pertenece a todos los intervalos necesarios, entonces esto provocará que $diff[i]$ se haga negativo, lo que no puede pasar.
  - En otro caso retorna 1, ya que es la única forma de no incluir al elemento en l_i en más intervalos de los que le corresponden