Naomi Lahera Champagne
Examen Final

Diseño y análisis de algoritmos
Universidad de La Habana

Performing Operations to Maximize Score

1 Definición del problema

Dado un arreglo a de longitud n, un entero k y un arreglo binario b de longitud n, se definen:

- $operation(a)$:
  Selecciona un índice i ($1 \leq i \leq n$) tal que $b_i = 1$. Establece $a_i = a_i + 1$ (es decir, incrementa a_i en 1).
- $mediana(c_i)$:
  Sea c_i el array resultante de eliminar el elemento de la posición i del array a, Se define mediana(c_i) como el $\left\lfloor \frac{|c_i|+1}{2} \right\rfloor$-ésimo elemento más pequeño de c_i [1].
- $score(a)$:
  Sea c_i el array de longitud $n-1$ que se obtiene al eliminar a_i de a. Se define el score como $\max_{i=1}^{n}(a_i + \text{mediana}(c_i))$ [2].

El objetivo del problema es maximizar el $score(a)$ luego de realizar a lo sumo k operaciones.

2 Análisis

Estamos en frente de un problema de optimización. Debemos notar que teniendo un array a de enteros y una permutación de a que denotaremos por a', se cumple que $score(a) = score(a')$. Hagamos que este planteamiento quede claro.

El $score(a)$ depende únicamente de un elemento a_i del array y la mediana asociada al c_i resultante de eliminar a_i de a. Sea a_j el elemento que maximiza $score(a)$ se cumple $a_j \in a$ y por tanto $a_j \in a'$. Como a y a' contienen extactamente los mismo valores se cumple que el $i - ésimo$ menor de a es también el $i - ésimo$ menor de a'. Luego el $\left\lfloor \frac{|c_i|+1}{2} \right\rfloor$-ésimo menor de c_j y ${c_j}'$ es el mismo y por ende $mediana(c_j) = mediana({c_j}')$. Tenemos entonces que $score(a) = score(a')$. **De ahora en adelante asumiremos que el array a de enteros cumple con el orden no decreciente.**

Vamos a esclarecer ahora aspectos relacionados con $mediana(c_i)$. Comencemos en 1 la numeración de las posiciones de los elementos del array a y el por consiguiente todos los c_i, de este

[1] Por ejemplo, mediana($[3, 2, 1, 3]$) = 2 y mediana($[6, 2, 4, 5, 1]$) = 4.

[2] En otras palabras, el score es el valor máximo de a_i+ mediana(c_i para todo i de 1 a n.

modo como c_i también cumple con el orden no decreciente tenemos que $mediana(c_i)$ es el elemento que está en la posición $\left\lfloor \frac{|c_i|+1}{2} \right\rfloor$ de c_i. Que posición ocupaba este elemento en el array a?

Sea $a_{\lfloor \frac{n}{2} \rfloor}$ el $\lfloor \frac{n}{2} \rfloor$ -ésimo menor elemento de a, veamos como varía la mediana del c_i resultante de eliminar un elemento menor o igual que $a_{\frac{n}{2}}$ y al eliminar un elemento mayor que $a_{\frac{n}{2}}$ del array a.

Sea $a_j \in a$, al eliminar a_j de a, el array c_j resultante contiene $n-1$ elementos y su mediana es el $\left\lfloor \frac{(|c_j|+1)}{2} \right\rfloor$-ésimo menor del array que sería $\lfloor \frac{n}{2} \rfloor$-ésimo menor elemento de c_j.

Caso 1: Sea a_j tal que $(\lfloor \frac{n}{2} \rfloor + 1) \leq j \leq n$ se cumple que $a_{\lfloor \frac{n}{2} \rfloor} = c_{\lfloor \frac{n}{2} \rfloor}$ es el $\lfloor \frac{n}{2} \rfloor$-ésimo menor elemento de a y de c_j y por tanto la mediana de c_j.

Caso 2: Sea a_j tal que $1 \leq j \leq \lfloor \frac{n}{2} \rfloor$ tenemos que al eliminar a_j de a, los a_k con $\lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n$ ocupan la posición $k-1$ en el array c_j resultante, luego $a_{\lfloor \frac{n}{2} \rfloor + 1} = c_{\lfloor \frac{n}{2} \rfloor}$ es el $\lfloor \frac{n}{2} \rfloor$-ésimo menor elemento del array y por tanto la mediana de c_j.

Podemos afirmar entonces que la mediana de los c_i resultantes de eliminar a_i de a es $a_{\lfloor \frac{n}{2} \rfloor}$ o $a_{\lfloor \frac{n}{2} \rfloor + 1}$.

Dado que la mediana solo puede tener uno de dos valores veamos que pasa con el *score* que al depender de la media nos hace analizar 2 posibles casos.

1. $\text{med}(c_i) = a_{\lfloor \frac{n}{2} \rfloor}$
   Los valores que hacen que $\text{med}(c_i) = a_{\lfloor \frac{n}{2} \rfloor}$ son los $\{a_i | \lfloor \frac{n}{2} \rfloor + 1 \leq i\}$. De estos valores el mayor valor es a_n [3], siendo este el valor que maximiza el score.

2. $\text{med}(c_i) = a_{\lfloor \frac{n}{2} \rfloor + 1}$
   Los valores que hacen que $\text{med}(c_i) = a_{\lfloor \frac{n}{2} \rfloor + 1}$ son los $\{a_i | i \leq \lfloor \frac{n}{2} \rfloor\}$. De estos valores el mayor valor es $a_{\lfloor \frac{n}{2} \rfloor}$, y por tanto es este el que maximiza el score.

Como $a_{\lfloor \frac{n}{2} \rfloor} + a_n \geq a_{\lfloor \frac{n}{2} \rfloor + 1} + a_{\lfloor \frac{n}{2} \rfloor}$ se cumple que *score* en el Caso 1 es claramente mayor que en el Caso 2, por lo tanto, es óptimo. **Podemos afirmar entonces que** $score(a) = a_n + \textbf{med}(c_n)$.

Dado que el score se puede definir como "máximo + mediana del resto", o sea $a_n + \text{med}(c_n)$ y solo depende de estos dos valores usemos las k operaciones para incrementar la mediana o incrementar el mayor elemento del array a, pero nuca los dos al mismo tiempo, pues de lo contrario habremos desperdiciado las operaciones. Veamos el por qué.

Supongamos que incrementamos la mediana del array y quedaron operaciones que maximizan el valor del máximo elemento del array. Supongamos que para aumentar el valor de la mediana se consumieron $m \leq k-1$ operaciones y se quieren emplear las $k-m$ operaciones restantes para aumentar el valor del máximo elemento del array que cumple $b_i = 1$ [4], denotemos a ese elemento como a_i. Al terminar de realizar las operaciones el nuevo máximo y el anterior difieren en a los sumo $a_i + k - m - a_n$ unidades.

Caso 1: Si al menos 2 de los elementos del array tuvieron que ser modificados para aumentar el $\lfloor \frac{n}{2} \rfloor$-ésimo menor elemento entonces la mediana aumentó a lo sumo $m-1$ unidades pues fue necesario como mínimo realizar 1 operación a cada uno de los elementos modificados.

[3] Recordemos que los arrays matienen orden no decreciente.

[4] Recordemos que solo se pueden aplicar operaciones al elemento a_i si $b_i = 1$.

Siendo así el score aumentó en $a_i + m - 1 - a_n + k - m < a_i + k - a_n$, siendo este último valor el aumento del score si se hibiesen aplicado todas las operaciones a a_i.

Caso 2: En cambio si no fue necesario incrementar más de un valor del array a para aumentar la mediana el resultado de consumir la $k - m$ operaciones restantes en a_i es equivalente a haber consumido las k operaciones en a_i pues en ambos casos el score aumentaría en $a_i + k - a_n$ unidades.

Luego podemos concluir que el máximo puede alcanzarse consumiendo operaciones para aumentar la meidiana o consuminedo operaciones para aumentar el máximo del array sin necesidad de ambas opciones de forma simultánea. Resolveremos el problema considerando ambos casos por separado.

Si queremos maximizar el máximo elemento del array a basta con encontrar el mayor a_i tal que $b_i = 1$ y aplicarle las k operaciones. Si se cumple que $a_n < a_i + k$ entonces habremos aumentado el *score*. Denotemos a este valor del score como s_1.

Si logramos aumentar el valor de la mediana del array entonces llamémole s_2 a ese valor del *score*. La respues del ejercicio será el máximo valor entre s_1 y s_2.

Veamos ahora como aumentar el valor de la mediana del array. Debemos aumentar la mediana del array resultante de eliminar a_n, o sea la mediana de c_n. Supongamos que queremos que la nueva mediana sea al menos x, tendríamos entonces que lograr que más de la mitad de c_n sea mayor o igual que x. Para esto haremos una búsqueda binaria.

3 Búsqueda Binaria

La búsqueda binaria se hará en $mediana(c_i)$ o sea en x. Notemos que puede existir $y \in c_i$ tal que $x \leq c_i$. Los elementos que cumplan con esta condición no necesitan consumir operación alguna para contribuir al aumento de la mediana. El resto de los elemento podrían o no incrementarse para que $x \leq mediana(c_i)$. Intuitivamente deberíamos elegir los índices más grandes i tales que $a_i < x$ y $b_i = 1$, e incrementarlos de manera voraz tanto como sea posible. Sea z el número máximo de elementos que pasan a ser mayores que x al final, la verificación es verdadera si $z \geq \lfloor \frac{n+1}{2} \rfloor$. Solo en ese caso la mediana de c_i puede ser al menos x, lo que nos llevaría a intentar con un mayor valor de x. Si $z \leq \lfloor \frac{n+1}{2} \rfloor$ entonces tendría que buscar un menor valor de x.

4 Complejidad temporal y espacial

- **Temporal:**
  - Ordenar el array $O(n * log(n))$.
  - Aumentar en k el maáximo valor de a que cumpla $b_i = 1$. Tengo que busacar dicho elemento en el array. $O(n)$.
  - Busqueda binaria. La búsqueda binaria se hace solo si se va a mantener fijo el máximo elemento del array a. Siendo así el mayor valor que puede tomar la mediana es $max(a)$. En cada valor de x que se analiza de puede que ser recorra todo el array c_n al tratar de incrementar los elementos del array, ya que es posibl eque entre los $\lfloor \frac{n-1}{2} \rfloor + 1$ mayores

elementso del array exista alguno con $b_i = 0$ obligándonos a incrementar uno de los restantes $\lfloor \frac{n-1}{2} \rfloor$ - 1 elementos. $O(n * log(max(a))$.

Sea $m = max(max(a), n)$ tenemos que la complejidad temporal es $O(n * log(m))$

- **Espacial:**
  - Tamaño los arrays. $O(n)$.

Implementación disponible en: GitHub