REALIZAR OPERACIONES PARA MAXIMIZAR LA PUNTUACIÓN

NAOMI LAHERA CHAMPAGNE C411

1. Problema

Sean a, b arrays de enteros y k un entero positivo se define la siguiente operación:

- 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).

Definición 1. *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].

Definición 2. *Sea c_i el array de longitud $n-1$ que se obtiene al eliminar a_i de a.* [2] *Se define el score como*

$$\max_{i=1}^{n}(a_i + mediana(c_i)).$$

Se quiere maximizar el score luego de realizar a lo sumo k operaciones.

2. Solución

Lema 1. *El orden de los elementos no altera el valor del score.*

Proof. Sean a array de enteros y d array de enteros que contiene una permutación de los elementos de a.

Se cumple que i-ésimo menor de un array no se afecta al cambiar su posición en el array, luego se cumple que el $\left\lfloor \frac{|a|+1}{2} \right\rfloor$-ésimo menor de ambos arrays es el mismo, luego la mediana es la misma.

Sea a_i el elemento que maximiza el score de a se cumple que $a_i \in d$, luego al eliminar a_i de a y de d se obtienen también dos arrays c_i y $\overline{c_i}$ con los mismo elementos pero en posiciones distintas y se cumple

[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.

que la mediana de c_i y $\overline{c_i}$ es la misma luego se cumple que a_i también maximiza el score de d.

Podemos afirmar que el score se matiene invariabte ante los cambios de las posiciones de los elementos en el array $\square$

De ahora en adelante asumiremos que el array a de enteros esta ordenado ascendentemente.

Lema 2. *La mediana de los c_i resultantes de eliminar a_i de a es $a_{\lfloor \frac{n}{2} \rfloor}$ o $a_{\lfloor \frac{n}{2} \rfloor + 1}$.*

Proof. 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 que pertenece a 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.

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. $\square$

Lema 3. *El score del array final (después de ordenarlo ascendentemente) es $a_n + med(c_n)$.*

Proof. Como el valor del score depende de la mediana y la media tiene 2 posibles valores tenemos que considerar dos casos.

Caso 1: med$(c_i) = a_{\lfloor \frac{n}{2} \rfloor}$

El array a está ordenado ascendentemente por tanto valor óptimo de i es n, ya que queremos maximizar a_i. Luego, el puntaje en este caso es $a_n + a_{\lfloor \frac{n}{2} \rfloor}$.

Caso 2: $med(c_i) = a_{\lfloor \frac{n}{2} \rfloor + 1}$

Como aclaramos en el caso anterior el array a está ordenado ascendentemente entonces el valor óptimo de i es $\lfloor \frac{n}{2} \rfloor$, ya que este es el valor más grande de i que cambiará la mediana. Esto obtiene un score de $a_{\lfloor \frac{n}{2} \rfloor} + a_{\lfloor \frac{n}{2} \rfloor + 1}$.

El score en el Caso 1 es claramente mayor que en el Caso 2, por lo tanto, es óptimo.

Así, el score se puede definir como "máximo + mediana del resto", o sea $a_n + \mathrm{med}(c_n)$ □

Entonces tenemos que el valor del score depende solamente de el mayor elemento del array y de la mediana del resto, por lo que es conveniente usar las operaciones solo para maximizar el mayor elemento del array o maximizar la mediana.

Lema 4. *Usaremos todas las k operaciones en el elemento que eventualmente se convierte en el elemento máximo en nuestro array, o usaremos todas las operaciones tratando de mejorar $\mathrm{med}(c_n)$ y mantener constante el elemento máximo.*

Proof. Existen 2 posibles casos:

Caso 1 Aplicamos operaciones que no contribuyen al aumento de la mediana ni alaumento del elementos mámimo del array que cumple $b_i = 1$.

Supongamos que realizamos x $(1 \leq x \leq k)$ operaciones en el mayor elemento que eventualmente se convirtió en el máximo.

Entonces, podríamos haber realizado las $k - x$ operaciones restantes en este elemento también, ya que este ya es el elemento máximo y hacer operaciones sobre él mejora nuestro score en 1 cada vez.

Caso 2 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 $bi = 1$, denotemos a ese elemento como a_i.

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. Siendo así el score aumentó en $a_i + m - 1 - a_n + k - m \leq a_i + k - a_n$, siendo este último el aumento del score si se hibiesen aplicado todas las operaciones a a_i. Luego podemos concluir que el máximo no se alcanza consumiendo operaciones para aumentar la meidiana y consuminedo operaciones para aumentar el máximo del array de forma silutánea. □

Solo es necesario analizar 2 casos, o bien aumentamos el máximo, o bien aumentamos la mediana de los demás. Resolveremos el problema considerando ambos casos por separado.

Caso 1: Realizamos operaciones en el elemento que eventualmente se convierte en el máximo.

Solución: Solo deberíamos realizar operaciones en el índice más grande i tal que $b_i = 1$, pues es el elemento mas grande que puede convertirse en máximo del array luego de realizar las k operaciones.

Proof. □

Caso 2: Realizamos operaciones para aumentar la mediana de los demás.

Solución: a_n está fijo como el elemento máximo en este caso, y queremos encontrar la mediana más grande posible usando las k operaciones en los otros $n - 1$ elementos.

Para umentar la mediana del array c_n asociado a a_n debemos asegurarno de que mas de la mitad de los elementos del array c_n son mayores o iguales que la nueva mediana.

Búsqueda binaria:
Supongamos que queremos verificar si podemos hacer que $\text{med}(c_n) \geq x$ o no. Algunos elementos ya son $\geq x$, y no los modificaremos. Algunos de los otros elementos pueden incrementarse para que se conviertan en $\geq x$ también. 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 se convierten en $\geq x$ al final. La verificación es verdadera si $z \geq \lfloor \frac{n+1}{2} \rfloor$.

Proof. □

Luego tenemos demostrado que deben ser destinadas todas las operaciones indistintamente a aumentar el valor del máximo elemento del array o a aumentar el valor de la mediana.

3. Algoritmo

3.1. **Complejidad.** Por lo tanto, el problema se resuelve en $O(N \cdot \log(\text{MAX}))$.

4. Anexos

4.1. **Problema original. C. Perform Operations to Maximize Score**

You are given an array a of length n and an integer k. You are also given a binary array b of length n.

You can perform the following operation at most k times:

- Select an index i $(1 \leq i \leq n)$ such that $b_i = 1$. Set $a_i = a_i + 1$ (i.e., increase a_i by 1).

Your score is defined to be $\max_{i=1}^{n}(a_i + \text{median}(c_i))$, where c_i denotes the array of length $n-1$ that you get by deleting a_i from a. In other words, your score is the maximum value of $a_i + \text{median}(c_i)$ over all i from 1 to n.

Find the maximum score that you can achieve if you perform the operations optimally.

For an arbitrary array p, median(p) is defined as the $\left\lfloor \frac{|p|+1}{2} \right\rfloor$-th smallest element of p. For example, $\text{median}([3,2,1,3]) = 2$ and $\text{median}([6,2,4,5,1]) = 4$.

Input

The first line contains an integer t $(1 \leq t \leq 10^4)$ — the number of test cases.

Each test case begins with two integers n and k $(2 \leq n \leq 2 \times 10^5$, $0 \leq k \leq 10^9)$ — the length of the array a and the number of operations you can perform.

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$ — denoting the array a.

The following line contains n space-separated integers $b_1, b_2, \ldots, b_n$ (b_i is 0 or 1) — denoting the array b.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5.

Output

For each test case, output the maximum value of score you can get on a new line.