Explicación del Problema: 2^sorted

En este documento se presenta una explicación clara del enunciado del problema “2 elevado a la sorted”, proveniente de la plataforma Codeforces. El objetivo es comprender qué pide exactamente el problema, cómo interpretar sus restricciones y cuál es la estrategia para resolverlo de manera eficiente sin caer en complejidades innecesarias.

Interpretación del Enunciado

Se nos proporciona:

- Un arreglo A de longitud n.
- Un entero k.

El objetivo es contar cuántos índices i (siendo $1 \le i \le n - k$) cumplen la siguiente condición:

Propiedad del subarreglo:

Se toma el subarreglo:

Ai, Ai+1, Ai+2, …, Ai+k

(de longitud k+1), y cada elemento se multiplica por una potencia de 2, de forma consecutiva:

- Primer elemento → $\times 2^0$
- Segundo elemento → $\times 2^1$
- Tercero → $\times 2^2$
- …
- Último → $\times 2^k$

Después de estas multiplicaciones, el subarreglo debe quedar en orden estrictamente creciente. Es decir:

$Ai \cdot 2^0 < Ai+1 \cdot 2^1 < Ai+2 \cdot 2^2 < \cdots < Ai+k \cdot 2^k$

Si esto se cumple, entonces el índice i es válido.

El programa debe devolver la cantidad total de índices válidos.

Datos de entrada:

-dos enteros n y k

- $n < 2 \times 10^5$
- $1 \le k < n$

-n enteros que representan los elementos del arreglo.

Salida:

Se debe imprimir la mayor suma posible obtenida usando hoteles consecutivos, sin exceder M.

Ejemplo:

- n = 7
- k = 2
- A = [11, 12, 4, 8, 16, 32, 64]

Debemos analizar subarreglos de longitud 3 (k+1).

Para cada subarreglo, multiplicamos por potencias de 2 consecutivas y verificamos si queda estrictamente creciente.

Al revisar todos los subarreglos posibles, encontramos que 3 de ellos cumplen la propiedad.

Por lo tanto, la respuesta es 3.

Problema de eficiencia:

La forma directa de resolver el problema sería:

- Para cada índice i, multiplicar k+1 valores por potencias de 2.
- Verificar si los k+1 elementos resultantes están en orden ascendente.

Esto genera un enfoque de complejidad $O(n \cdot k)$.

El inconveniente es que:

- k puede ser hasta n−1.
- En el peor caso, la complejidad llega a $O(n^2)$.
- Las potencias de 2 podrían ser demasiado grandes para caber en tipos de datos comunes.

Claramente, esta no es una solución viable para n = 200,000.

Observación clave para optimizar

En realidad no hace falta multiplicar nada.

Observemos:

Si queremos que:

Ai·2^x<Ai+1·2^x+1

Esto es equivalente a:

Ai<2·Ai+1

El exponente exacto no importa, porque lo único que estamos verificando es que:

- Cada elemento multiplicado por su potencia sea menor que el siguiente multiplicado por la potencia siguiente.

- Y entre potencias consecutivas siempre se cumple que:

$2^{x+1}=2\cdot 2^{x}$

Conclusión:

Para verificar el orden creciente del subarreglo, no necesitamos calcular potencias. Basta con comprobar:

$A_j<2\cdot A_{j+1}$

para cada par consecutivo dentro del subarreglo.

Solución óptima (O(n)):

Podemos recorrer el arreglo una sola vez manteniendo una variable llamada válido, que cuenta cuántos pares consecutivos cumplen:

$A_j<2A_{j+1}$

- Si válido ≥ k → tenemos un subarreglo de k+1 elementos que cumple la propiedad. Sumamos 1 a la respuesta.
- Si algún par rompe la propiedad, válido se reinicia a 0.

De esta forma, evitamos repetir comparaciones entre subarreglos y eliminamos las potencias, logrando una solución lineal.

Conclusión:

Al analizar correctamente la propiedad requerida, se descubre que todo puede reducirse a una simple comparación entre pares consecutivos: verificar si cada elemento es menor que el doble del siguiente. Esto permite usar un método lineal, eficiente y adecuado para n de hasta 200,000 elementos.