Explicación del Código: 2^sorted

En este documento se analiza la implementación en Java de dos enfoques distintos para resolver el problema “2 elevado a la sorted”: el método ingenuo (cándido) y el método óptimo.

El objetivo es comprender la lógica detrás de cada implementación, su complejidad y por qué uno de ellos es significativamente más eficiente que el otro.

1. Método Cándido (Enfoque Básico)

El método cándido sigue la lógica más directa posible:

evaluar cada subarreglo por completo, comparando todos los pares de elementos que lo componen.

1.1 Inicialización

Se crean dos variables:

- valido: cuenta cuántas comparaciones consecutivas cumplen la propiedad.
- respuesta: cuenta cuántos subarreglos son válidos.

respuesta se inicializa en 0.

1.2 Doble ciclo for

El primer ciclo recorre las posiciones posibles donde puede iniciar un subarreglo de longitud k+1:

for i desde 0 hasta n - k - 1

Esto evita salirse de los límites del arreglo.

Dentro de este ciclo externo, se incluye un segundo ciclo interno, cuyo objetivo es revisar todas las comparaciones dentro del subarreglo actual:

for j desde i hasta i + k - 1

En cada paso, se compara:

arreglo[j] < 2 * arreglo[j + 1]

Si la condición se cumple, se incrementa valido.

1.3 Verificación del subarreglo

Al finalizar el ciclo interno, si:

valido == k

entonces el subarreglo completo cumple la propiedad, así que:

respuesta++

Posteriormente, valido se reinicia para analizar el siguiente subarreglo.

1.4 Complejidad

Debido al doble ciclo, este método tiene complejidad:

O(n·k)

En el peor caso, k ≈ n, lo que convierte el algoritmo en:

O(n^2)

Lo cual es demasiado lento para n = 200 000.

2. Método Óptimo

El método óptimo evita repetir comparaciones innecesarias y elimina el ciclo interno.

En lugar de analizar cada subarreglo desde cero, aprovecha la continuidad de las desigualdades.

2.1 Inicialización

Se declaran nuevamente:

- valido = 0
- respuesta = 0

2.2 Ciclo único

Se realiza un solo recorrido del arreglo, comparando cada par consecutivo:

for i desde 0 hasta n - 2

Se compara:

arreglo[i] < 2 * arreglo[i + 1]

Si la condición se cumple, se incrementa valido.

Si no se cumple, se reinicia a 0, ya que la cadena de desigualdades consecutivas se rompe.

2.3 Detección de un subarreglo válido

Cada vez que:

valido >= k

esto significa que se han encontrado k comparaciones válidas consecutivas, lo que implica la existencia de un subarreglo de longitud k+1 que cumple la propiedad:

respuesta++

No hay que reiniciar valido porque las comparaciones pueden seguir acumulándose para futuros subarreglos solapados.

2.4 Complejidad

Este enfoque realiza un único recorrido:

O(n)

Cumple las restricciones del problema sin riesgo de tiempos excesivos.
Conclusión:
Para resolver el problema “2 elevado a la sorted”, el método cándido funciona conceptualmente, pero es inviable en términos de eficiencia, ya que repite comparaciones y utiliza un doble ciclo que puede llegar a ser cuadrático.
Por el contrario, el método óptimo:

- Aprovecha la continuidad entre subarreglos
- Evita recalcular comparaciones
- Funciona en tiempo lineal
- Y cumple todos los límites de la competencia

Por ello, es la implementación recomendada y la única capaz de resolver el problema dentro de los tiempos permitidos.