

Puntos de Monitoreo (Combinando Cobertura y Distancia Máxima)

Descripción del Problema

Tienes N puntos de interés en una línea recta.

Debes seleccionar M de estos puntos de interés para instalar sensores de monitoreo.

Los M sensores deben ubicarse de tal manera que la distancia máxima de cualquier punto de interés al sensor más cercano sea lo más pequeña posible.

En otras palabras: encuentra M ubicaciones de sensores s_1, s_2, \dots, s_m tales que el radio de cobertura R sea mínimo, donde

$R = \text{máximo, para cada punto de interés } p_i, \text{ de la distancia mínima entre } p_i \text{ y cualquier sensor.}$

Detalles y Restricciones

- Entrada:
 - La primera línea contiene N (puntos de interés) y M (sensores a colocar) con $1 \leq M \leq N \leq 100000$.
 - La segunda línea contiene N enteros p_1, p_2, \dots, p_n (entre -1000000000 y 1000000000), que son las coordenadas de los puntos. Se dan en orden no decreciente.
- Salida:
 - Imprime el valor mínimo del radio de cobertura R que puede lograrse seleccionando óptimamente los M sensores.

Ejemplo

Input	Output
5 2 1 2 5 6 10	2

Explicación del Ejemplo:

Puntos: 1, 2, 5, 6, 10

Se deben colocar 2 sensores.

Si se ponen sensores en 2 y 6:

- 1 está a distancia 1
 - 2 a distancia 0
 - 5 a distancia 1
 - 6 a distancia 0
 - 10 a distancia 4
- Radio máximo = 4

Si se ponen sensores en 2 y 8 (ubicación no necesariamente dentro de la lista):

- 1 y 2 quedan a distancia como máximo 1

- 5 y 6 quedan a distancia 3 y 2 respecto de 8, o 3 y 4 respecto de 2 → mínimos 2
- 10 queda a distancia 2 respecto de 8
Radio máximo = 2, que es el mínimo posible.

CÓDIGO

```
#include <iostream>

#include <algorithm>

#include <cmath>

using namespace std;

int main() {

    int N, M;

    cin >> N >> M;

    long long* p = new long long[N];

    for (int i = 0; i < N; i++)
        cin >> *(p + i);

    sort(p, p + N);

    long long R = 9223372036854775807LL;

    // Intentar colocar sensores en puntos existentes
    for (int i = 0; i < N - M + 1; i++) {

        long long inicio = *(p + i);
        long long fin = *(p + i + M - 1);

        long long r = (fin - inicio) / 2;
```

```
        if (r < R) R = r;
    }

    cout << R;

    delete[] p;
    return 0;
}
```