**BSEQ.CPP | BSEQ.PY (100 điểm)**

Một dãy **N** phần tử gọi là dãy đẹp nếu dãy đó có dạng **x, x + K, ... x + (N - 1) \* K**. Trong toán học người ta có thể gọi dãy này là dãy cấp số cộng, với công sai là **K**.

Cho một dãy **A** gồm **N** phần tử **A1**, **A2**, ... **AN** và số **K**. Các bạn được phép thay đổi một phần tử bằng cách tăng hoặc giảm phần tử đấy đi 1 đơn vị.

***Yêu cầu:*** Hãy tìm số lần thay đổi ít nhất để biến dãy **A** thành một dãy đẹp.

***Dữ liệu:*** vào từ file **BSEQ.INP** gồm:

- Dòng đầu tiên gồm số nguyên dương **N** và số nguyên **K** (1 ≤ **N** ≤ 104)

- Dòng tiếp theo gồm **N** số nguyên **A1, A2, ..., AN**. (|**Ai**| ≤ 109).

***Kết quả*:** ghi ra file **BSEQ.OUT** là một số duy nhat là số lần thay đổi ít nhất tìm được.

***Ví dụ:***

|  |  |
| --- | --- |
| **BSEQ.INP** | **BSEQ.OUT** |
| 5 1  1 4 6 3 2 | 9 |

***Giới hạn:***

- Có 30% số điểm có **K** = 0.

- Có 70% số điểm có |**K|** ≤ 109.

### Ý tưởng thuật toán:

Bài toán yêu cầu biến đổi một dãy A thành một dãy đẹp có dạng cấp số cộng: x,x+K,x+2K,…,x+(N−1)K với số lần thay đổi nhỏ nhất (mỗi lần thay đổi 1 đơn vị).

### 1. Phân tích dãy đẹp:

Một dãy đẹp có dạng:

Bi​=x+(i−1)⋅K

Trong đó:

* x là giá trị đầu tiên của dãy đẹp.
* K là công sai giữa các phần tử.

Ta cần tìm cách biến dãy A thành một dãy đẹp bằng cách thay đổi các phần tử Ai​. Mục tiêu là tìm số lần thay đổi (tăng hoặc giảm các phần tử) ít nhất.

### 2. Chuẩn hóa dãy A:

Để dễ dàng tính toán, thay vì tính dãy đẹp B cho mỗi vị trí từ x, ta "chuẩn hóa" dãy A bằng cách chuyển đổi nó thành một dãy mà mỗi phần tử không còn phụ thuộc vào i, nghĩa là không còn phần tử nào thay đổi theo (i−1)⋅K.

Cụ thể:

* Tại vị trí thứ i, giá trị của dãy đẹp là Bi​=x+(i−1)⋅K.
* Vì vậy, ta trừ đi (i−1)⋅K khỏi Ai​ để loại bỏ sự phụ thuộc vào chỉ số i.
  + Bi​=Ai​−(i−1)⋅K.

Sau khi trừ, tất cả các phần tử trong dãy mới chỉ phụ thuộc vào giá trị x, và giá trị đẹp nhất cho các phần tử này sẽ là một số gần với trung vị của chúng.

### 3. Tại sao là trung vị?

Sau khi chuẩn hóa, ta thu được một dãy mới B, trong đó tất cả các phần tử cần phải gần với một giá trị nào đó để tổng số thay đổi là nhỏ nhất. Giá trị tốt nhất để chọn chính là **trung vị** của dãy B.

Lý do là:

* Trung vị là giá trị sao cho tổng khoảng cách tuyệt đối từ nó đến tất cả các phần tử khác là nhỏ nhất.
* Nếu bạn chọn trung vị làm giá trị chuẩn, tổng số lần thay đổi để đưa các phần tử khác đến trung vị sẽ là nhỏ nhất.

### 4. Tính tổng số lần thay đổi:

Khi đã có dãy chuẩn hóa B và chọn được trung vị, việc tính tổng số lần thay đổi là đơn giản:

* Với mỗi phần tử Bi​, ta tính khoảng cách tuyệt đối của nó với trung vị.
* Tổng số thay đổi là tổng tất cả các khoảng cách tuyệt đối này.

### 5. Tổng quan thuật toán:

1. **Chuẩn hóa dãy**: Tạo dãy B từ A bằng cách: Bi​=Ai​−(i−1)⋅K.
2. **Tìm trung vị**: Sắp xếp dãy B và chọn trung vị của dãy này làm giá trị tham chiếu.
3. **Tính tổng thay đổi**: Tính tổng khoảng cách tuyệt đối giữa mỗi phần tử trong B và trung vị.
4. **In kết quả**: In ra tổng số thay đổi ít nhất.

### 6. Ví dụ minh họa:

**Input:**

Copy code

5 1

1 4 6 3 2

**Bước 1 - Chuẩn hóa dãy**:

* K=1
* Dãy B sẽ được tính như sau:
  + B1​=A1​−0⋅1=1
  + B2​=A2​−1⋅1=4−1=3
  + B3​=A3​−2⋅1=6−2=4
  + B4​=A4​−3⋅1=3−3=0
  + B5​=A5​−4⋅1=2−4=−2

Dãy chuẩn hóa B=[1,3,4,0,−2].

**Bước 2 - Tìm trung vị**:

* Sắp xếp dãy B: [−2,0,1,3,4].
* Trung vị của dãy này là giá trị ở giữa, tức là 1.

**Bước 3 - Tính tổng số lần thay đổi**:

* Tính khoảng cách từ mỗi phần tử trong B đến trung vị 1:
  + ∣1−1∣=0
  + ∣3−1∣=2
  + ∣4−1∣=3
  + ∣0−1∣=1
  + ∣−2−1∣=3

Tổng số lần thay đổi = 0+2+3+1+3=9.

**Kết quả**: Số lần thay đổi ít nhất là 9.

### 7. Độ phức tạp:

* Bước sắp xếp có độ phức tạp O(NlogN).
* Tính tổng khoảng cách tuyệt đối có độ phức tạp O(N). => Tổng độ phức tạp của thuật toán là O(NlogN), đủ nhanh để xử lý với N≤104.

### Kết luận:

Thuật toán này sử dụng việc chuẩn hóa và tìm trung vị để tìm ra số lần thay đổi ít nhất, và nó có độ phức tạp tối ưu cho bài toán.

#include <iostream>

#include <vector>

#include <math.h>

#include <bitset>

#include <cctype>

#include <string>

#include <algorithm>

#include <unordered\_map>

#pragma GCC optimize("O3")

using namespace std;

#define fsize(array) (array).begin(), (array).end()

#define bsize(array) (array).rbegin(), (array).rend()

typedef long long ll;

typedef unsigned long long ull;

typedef vector <ll> vll;

typedef unordered\_map <ll, ll> table4l;

typedef unordered\_map <string, ll> tablesl;

const ll MAXN = 1e18;

const ll MINN = -1e18;

int main() {

cin.tie(nullptr), cout.tie(nullptr)

-> ios\_base::sync\_with\_stdio(false);

#ifndef ONLINE\_JUDGE

freopen ("main.inp", "r", stdin);

freopen ("main.out", "w", stdout);

#endif

ll N, K;

cin >> N >> K;

vll A(N), B(N);

for (int i = 0; i < N; ++i) {

cin >> A[i];

B[i] = A[i] - i \* K;

}

sort(fsize(B));

ll median = B[N / 2];

ll result = 0;

for (int i = 0; i < N; ++i) {

result += abs(B[i] - median);

}

cout << result << endl;

return 0;

}