**BÀI TẬP**

Cho mảng A có N số nguyên, a[i] ∈ [1; N]. Bạn được phép thực hiện vô hạn các thao tác. Trong mỗi thao tác, chọn 1 phần tử a[i] bất kỳ tăng (giảm) 1 đơn vị, nhưng sau khi tăng (giảm) thì a[i] ∈ [1; N]

Tìm **số thao tác ít nhất** để biến tất cả các phần tử trong mảng thành các giá trị **đôi một khác nhau** (nghĩa là không có giá trị nào lặp lại trong mảng).

**Input**

Dòng đầu tiên là số nguyên dương N (1 N 2.105)

Dòng thứ hai là N số của mảng A, các số cách nhau bởi 1 dấu cách

**Output**

Ghi ra số thao tác tối thiểu để biến đổi N phần tử của mảng đôi một khác nhau. Nếu kết quả lớn hơn 109, ghi -1.

**Ví dụ**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  1 2 1 | 2 |
| 4  2 1 4 4 | 1 |
| 6  6 2 3 5 6 3 | 4 |

**Giải thích test:**

* Test 1:

+ Cách 1: Tăng a[2] lên 2 lần liên tiếp: [1, 2, 1] 🡪[1, 2, 2] 🡪[1, 2, 3]

+ Cách 2: Tăng a[0] lên 2 lần liên tiếp: [1, 2, 1] 🡪[2, 2, 1] 🡪[3, 2, 1]

+ Cách 3: [1, 2, 1] 🡪[1, 3, 1](a[1]++) 🡪[1, 3, 2] (a[2]++)

+ Cách 4: [1, 2, 1] 🡪[2, 2, 1](a[0]++) 🡪[2, 3, 1] (a[1]++)

* Test 2: Giảm a[2] hoặc a[3] đi 1 đơn vị ([2, 1, 4, 3] hoặc [2, 1, 3, 4])
* Test 3: [6, 2, 1, 5, 4, 3]

**Lời giải**

Việc đầu tiên là ta sẽ đếm tần suất xuất hiện của các phần tử trong mảng. Sau đó, tạo 1 vector để lưu các số bị dư ra, tạm gọi là X[]. Ví dụ, mảng gốc có 4 con 3 thì ta đưa vào X 3 con 3.

Tiếp đó, ta tạo 1 vector Y lưu các số từ 1 đến N mà chưa có trong mảng gốc.

Bài toán quy về việc tìm số bước biến đổi dãy số X sao cho xuất hiện tất cả các số trong Y với thao tác tăng giảm như trên đề bài.

Muốn cho X[i] và Y[i] gần nhau nhất, cách tiếp cận tối ưu chính là sắp xếp tăng cả 2 dãy X và Y, vì điều này sẽ khiến những số bé nhất cả 2 dãy cùng dạt về bên trái, khiến cho mỗi X[i] và Y[i] gần nhau hơn. Và tất nhiên, bao nhiêu số bị dư ra (X.size()) thì tương ứng sẽ có bấy nhiêu số thiếu. X[i] muốn trở thành Y[i] sẽ phải thêm (bớt) 1 lượng bằng |X[i]- Y[i]|

**Mã nguồn**

#include <bits/stdc++.h>

#define ll long long

using namespace std;

int main() {

ll n; cin >> n;

vector<ll> A(n);

for(int i = 0; i < n; i++) cin >> A[i];

ll freq[n + 1]={0};

//Đếm tần suất các phần tử

for(int i = 0; i < n; i++) freq[A[i]]++;

//vector lưu các số bị dư ra, các số chưa xuất hiện

vector<ll> trung, thieu;

for(ll x = 1; x <= n; x++) {

//Ví dụ mảng gốc có 4 con 3 thì ta đẩy vào vector 3 con 3

if(freq[x] > 1) {

for(ll i = 0; i < freq[x] - 1; i++) trung.push\_back(x);

} else if(freq[x] == 0) {

thieu.push\_back(x);

}

}

if(thieu.empty()) {

cout << 0 << "\n";

return 0;

}

sort(trung.begin(), trung.end());

sort(thieu.begin(), thieu.end());

ll ans = 0;

const ll LIMIT = 1000000000LL;

for(ll i = 0; i < trung.size(); i++) {

ll cost = llabs(trung[i] - thieu[i]);

ans += cost;

if(ans > LIMIT) {

cout << -1 << "\n";

return 0;

}

}

cout << ans << "\n";

return 0;

}