Problem - MEXimize the Array

- **Question Analysis:**
  আমাকে একটা **array A** দেওয়া আছে।
  Operation :
  - যেকোনো **index i choose** করে, **Ai** এর মান **+1 or -1** করতে পারব।

  Minimum number of operations বের করতে হবে, যাতে **MEX -** এর **maximize** হয়।

- **Some Key Takeaways:**
  একটা **given array** এর **size** যদি হয়- **n,**
  তাহলে **possible maximum MEX = n [0-based]**

- **Observation:**
  1. আমি যেহেতু ইচ্ছামতো **increase or decrease** করতেই পারি, তাহলে **-** আমি **array** থেকে যত **MEX** পাওয়া **possible** সেটা **easily** বের করতে পারব।
  2. আচ্ছা , আমার কাছে একটা **array** মনে কর এমন আছে,
     0 1 3 5 5

     এখন, আমি এটা দুইভাবে ভাবতে পারি।
     - **already** যা আছে , তা **change** করব না। যেগুলো **0 - n-1** অবধি নাই, অথবা যেগুলো **more than 1 frequency** , সেগুলো **change** করতে পারি।

     তাহলে, এই **array** এর জন্য **cost = (5-2) + (5-4) = 3+1 = 4 [2,4** বানাচ্ছি**]**

     - **Already** থাকুক আর না থাকুক, আমি **ascending** করে **for each index check** করব, যা থাকার কথা, তা থেকে কত কম বা বেশি আছে, তাই তো আমরা **operation** এর মাধ্যমে **change** করব, তাই না**?**
       তাহলে **-**
       index - 0 1 2 3 4
       value - 0 1 3 5 5
       diff - 0 0 1 2 1 cost = 1+2+1 =4

  3. তাহলে দেখা গেলো, **cost same** । কারণ , আমাকে তো **ultimately change** করতেই হবে। অর্থাৎ **-**
     ধর, তোমার **2** আর **3** লাগবে। আছে **- 3 5**

এখন তুমি যদি **- 3 already** আছে , এতে হাত দিব না। **5 -** কে **2** বানাবা ভাব, তোমাকে কিন্তু **5** থেকে **3** কে **cross** করেই **2** বানাতে হচ্ছে, মানে **cost = 3**

এখন **same** জিনিস এভাবে কর, **3 -> 2 and 5 -> 2,** মানে **cross** না করে। এতে **cross** করে যে **extra cost (1)** আসতাছিল, সেই **cost** দিয়েই, **3 ->2.**

বিষয়টা **same-ই,** শুধু **difference** হলো **cross** না করে ভাবলে, আমার **implement** করতে সুবিধা হবে।

- **Implementation:**
  Sort করব। **index** এর মানটাই , আমার **desire value** গুলো । তারপর, **absolute difference** গুলোই আমার **minimum cost/operation.**

- **Code:**

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        vector<int> a(n);
        for(auto &i: a) cin >> i;
        sort(a.begin(), a.end());

        ll cnt=0;
        for(int i=0; i<n; i++) cnt+=abs(a[i]-i);

        cout << cnt << endl;
    }
    return 0;
}
/*   Author: Hridoy Barua (CS Instructor Phitron)   */
```

Problem - Anti Adjacent Swaps

Question Analysis:

আমাকে একটা **array A** দেওয়া আছে।

Operation :

- **adjacent** না এমন যেকোনো **2** টা **index** এর **element-**কে **swap** করতে পারব।

Target : array কে **sort** করা।

- ## Observation:

1. আমি যেহেতু যেকোনো **non-adjacent element -** কে **swap** করতে পারতাছি, তাই - আমি **easily array** কে **sort** করতে পারব।
2. কিন্তু কখন পারব না**?**
   n = 1 : one element , already sorted
   n = 2 : Two elements and adjacent, so already sorted না থাকলে, **operation** করে **sort** করা **possible** না।
   n = 3 : three elements, only 0th and 2 th can be swapped using operation. তারমানে, **1th** কে **already sorted** জায়গায় থাকতে হবে।

- ## Implementation:

Follow The observation part

- **Code:**

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        vector<int> a(n);
        for(auto &i: a) cin >> i;

        if(n>3 || n==1) cout << "YES\n";
        else if(n==2){
            if(a[0]<=a[1]) cout << "YES\n";
            else cout << "NO\n";
        }
        else{
            if(a[0]>a[2]) swap(a[0],a[2]);

            if(a[0]<=a[1] && a[1]<=a[2]) cout << "YES\n";
            else cout << "NO\n";
        }
    }
    return 0;
}
/*   Author: Hridoy Barua (CS Instructor Phitron)   */
```