Nikolas Mavrogeneiadis
gravitorious
June 29, 2024

Bazoka and Mocha's Array

Πρόβλημα: *Link*

Λήμμα 1. *Έστω ο πίνακας a με n στοιχεία a_i και $1 \leqslant i \leqslant n$. Προφανώς $|x| + |y| = n$. Έστω ότι επιλέγουμε $|y| = 1$ (μέγεθος του δεύτερου subarray), και εκτελούμε την πράξη $y + x$. Αν επαναλάβουμε αυτή την διαδικασία k φορές, το αποτελέσμα θα είναι το ίδιο με το να επιλέγαμε από την αρχή $|y| = k$.*

Απόδειξη. Για $|y| = 1$ προφανώς ισχύει. Έστω ότι ισχύει για $|y| = k$. Θα δείξουμε ότι ισχύει για $|y| = k + 1$. Αν επιλέξουμε $|y| = k$ ($|x| = n - k$), τότε το αποτέλεσμα είναι $A = \{a_{n-k+1}, ..., a_n, a_1, a_2, ..., a_{n-k}\}$. Από επαγωγική υπόθεση αυτό είναι το ίδιο με το να εκτελούσαμε k φορές την διαδικασία για $|y| = 1$. Αν τώρα επιλέξουμε $|y| = 1$ από αυτό το σημείο και εκτελέσουμε την διαδικασία στο A, το αποτέλεσμα είναι $A = \{a_{n-k}, a_{n-k+1}, ..., a_n, a_1, a_2, ..., a_{n-k-1}\}$, το οποίο είναι το ίδιο με το να επιλέγαμε από την αρχή $|y| = k + 1$. □

Πόρισμα 1. *Αν επαναλάβουμε την διαδικασία για $|y| = 1$ παραπάνω από $n-1$ φορές, τότε κάθε n επαναλήψεις θα παίρνουμε ως αποτέλεσμα τον αρχικό πίνακα.*

Πόρισμα 2. *Αν γίνεται ο πίνακας να ταξινομηθεί σε αύξουσα σειρά, τότε αυτό μπορεί να γίνει σε μόλις ένα βήμα (με τα κατάλληλα x και y).*

Θεώρημα 1. *Ο πίνακας μπορεί να ταξινομηθεί σε αύξουσα σειρά με την συγκεκριμένη διαδικασία, αν και μόνο αν υπάρχει στον αρχικό πίνακα το πολύ ένα ζευγάρι a_i, a_{i+1} με $1 \leqslant i \leqslant n-1$ για το οποίο ισχύει $a_i > a_{i+1}$ και τότε στον αρχικό πίνακα ισχύει $a[n] \leqslant a[0]$.*

Απόδειξη. Έστω ότι ο πίνακας μπορεί να ταξινομηθεί σε αύξουσα σειρά. Αν είναι ήδη σε αύξουσα σειρά, το συμπεράσμα ισχύει. Αν δεν είναι σε αύξουσα σειρά, τότε αναγκαστικά υπάρχει τουλάχιστον ένα ζευγάρι a_i, a_{i+1} για το οποίο $a_i > a_{i+1}$. Έστω ότι υπάρχει ακριβώς ένα. Τότε, αν $|x| = i$ και $|y| = n - i$, το $y + x$ είναι σε αύξουσα σειρά (γιατί· - Ο πίνακας περιέχει μια αύξουσα σειρά από την θέση 1 μέχρι την θέση i και άλλη μια αύξουσα σειρά από το $i + 1$ μέχρι το n και αφού ο πίνακας μπορεί να ταξινομηθεί, τότε αν ενώσουμε αυτές τις δύο αύξουσες ακολουθίες βάζοντας την 2η πρώτα έχουμε το αποτέλεσμα - οποιαδήποτε άλλη επιλογή θα οδηγούσε στην διατήρηση του ζευγαριού $a_i > a_{i+1}$ και επομένως σε άτοπο) και άρα ισχύει ότι $a[n] \leqslant a[0]$. Έστω ότι έχουμε παραπάνω από ένα ζευγάρι, για παράδειγμα δύο τέτοια ζευγάρια. Τότε οποιαδήποτε επιλογή και αν κάνουμε, τουλάχιστον ένα από τα δύο ζευγάρια θα διατηρηθεί και άρα ο πίνακας δεν θα είναι σε αύξουσα σειρά σε καμία περίπτωση, κάτι το οποίο είναι άτοπο. Η αντίστροφη περίπτωση είναι τετριμμένη. □

```cpp
//created by gravitorious
#include <bits/stdc++.h>
using namespace std;

#ifdef XOX
#include "/home/nikolas/gravitonlib/mydebug/debug.h"
#else
#define debug(...) 21198
#endif

using ll = long long;
using ld = long double;
using uint = unsigned int;
using ull = unsigned long long;

#define getunique(v) {sort(v.begin(), v.end()); v.erase(unique(v.begin(), v
    .end()), v.end());}

double int_part(double x, double* intpart){
    //returns the real part
    return modf(x, intpart);
}

ll udiv(ll a, ll b) {
    return a / b + ((a ^ b) > 0 && a % b);
}  // divide a by b rounded up
ll ddiv(ll a, ll b) {
    return a / b - ((a ^ b) < 0 && a % b);
}  // divide a by b rounded down

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

long long myRand(long long B) {
    //0 to B-1
    return (unsigned long long)rng() % B;
}

int myUniRand(int a, int b){
    //a to b
    uniform_int_distribution<int> distribution(a,b);
    return distribution(rng);
}

void solve() {
    int n;
    cin >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    bool flag = false, flag2 = false;
    int first;
    for(int i = 0; i < n - 1; i++){
        if(v[i] > v[i + 1] && flag){
            flag2 = true;
            break;
        }
```

```
        if(v[i] > v[i + 1]){
            flag = true;
        }
    }
    if(flag && flag2) cout << "NO\n";
    else if(!flag && !flag2) cout << "YES\n";
    else if(flag){
        if(v[n - 1] <= v[0]) cout << "YES\n";
        else cout << "NO\n";
    }
    else cout << "YES\n";
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```