Bài này là mở rộng của bài Dãy nghịch thế.

Gọi F[k][i] là số lượng dãy nghịch thế độ dài k, kết thúc tại A[i].

Ban đầu F[1][i]=1 với $1 \le i \le N$, ta có $F[k][i]=\sum F[k-1][j]$ với j thỏa $j<i$ và $A[j]>A[i]$.

Kết quả bài toán là $\sum F[K][i]$.

Để tính được F[k] từ F[k−1] trong O(NlogN), ta sử dụng cấu trúc BIT như đã sử dụng trong bài NKINV.

Do đề bài không giới hạn giá trị của số nên ta cần nén số lại để các phần tử trong mảng A đều có giá trị trong khoảng 1 đến N.

Có thể xem về cách nén số ở đây.

Đoạn code ở dưới hoán đổi thứ tự lớn/bé của mảng đầu vào để thuận tiện cho việc xử lí.

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
using namespace std;

#define M 1000000000

struct Fenwick {
    int n;
    vector<int> f;
    Fenwick(int n): n(n), f(n+1, 0) {}
    void up(int i, int x) {
        for (; i<=n; i += i&-i) (f[i] += x) %= M;
    }
    int get(int i) {
        long long res = 0;
        for (; i>=1; i -= i&-i) res += f[i];
        return res % M;
```

```
    }
    void reset() {
        f.assign(n+1, 0);
    }
};

void compress(vector<int> &a) {
    set<int> s(a.begin(), a.end());
    vector<int> b(s.begin(), s.end());
    for (int &x: a) {
        x = b.end() - lower_bound(b.begin(), b.end(), x);
    }
}

int main() {
    ios::sync_with_stdio(false); cin.tie(0);
    int n, k; cin >> n >> k;
    vector<int> a(n);
    for (int &x: a) cin >> x;
    compress(a);

    vector<int> f(n, 1), g(n);
    Fenwick bit(n);
    while (--k) {
        g.assign(n, 0);
        bit.reset();
        for (int i=0; i<n; i++) {
            g[i] = bit.get(a[i]-1);
            bit.up(a[i], f[i]);
        }
```

```
        swap(f, g);
    }

    long long res = 0;
    for (int x: f) res += x;
    cout << res % M;
    return 0;
}
```