Solution to ARC 183 B - Near Assignment

Aug 25, 2024

Problem: https://atcoder.jp/contests/arc183/tasks/arc183_b
Difficulty: 2118 (Kenkoooo).
Prerequisites: Why bubble sort works; why A must contain every element in B; solving $k = 1$.

This post is basically https://atcoder.jp/contests/arc183/editorial/10797, other than a few counterexamples for false claims, a longer explanation, and shorter code.

The case $k = 1$ will not be covered here. We'll start with some tempting but false claims for $k \geq 2$:

> ⚠ **Bogus Claim – At least one duplicate element.**
>
> If $k \geq 2$, A contains every element in B, and B contains a duplicate element, then the answer is `Yes`.
>
> **Counterexample:** $k = 2$, $A = [1, 2, 3, 4]$, $B = [1, 2, 3, 1]$. This meets the conditions, but the answer is `No`.

> ⚠ **Bogus Claim – At least two duplicate elements.**
>
> If $k \geq 2$, A contains every element in B, and B contains at least two duplicate elements, then the answer is `Yes`.
>
> **Counterexample:** $k = 2$, $A = [1, 2, 3, 4, 4]$, $B = [1, 2, 3, 1, 2]$. This meets the conditions, but the answer is `No`.

For both examples, we can prove that the answer is `No` by looking at a hypothetical **last operation**. The idea is that after we apply an operation on i and j, $A_i = A_j$ holds.

So, if the answer were `Yes`, then after the last operation, we would be able to find two distinct indices i and j such that $|i - j| \leq k$ and $A_i = A_j$. But no such values for i and j exist in B, so this is impossible.

This leads to the following claim:

> **Claim – Last operation must be possible.**
>
> If $A \neq B$ and the answer is `Yes`, then there exist distinct indices i and j such that $|i - j| \leq k$ and $B_i = B_j$.

It's important to note why $A \neq B$ is required: if $A = B$, then no operations are needed, so there isn't a "last operation" that forces $B_i = B_j$.

On the other hand, if a last operation is possible, we can continue by considering all possible previous states that could lead to B. This leads to the following key result:

> **Claim – Last operation implies `Yes` for all reasonable A.**
>
> If there exist distinct indices i and j such that $|i - j| \leq k$ and $B_i = B_j$, then the answer is `Yes` for all arrays A that contain every element of B.

Proof: Say that an array A is *reachable* from B if there is a sequence of operations from A that lead to B.

We will work backwards starting from B.

Suppose a last operation is possible on indices i and j. Then, we have $B_i = B_j = x$ for some x, so the array looks like this:

$$B = [\ldots x \ldots x \ldots]$$

This operation would have overwritten the previous value of B_j. Here are some possible previous arrays:

$$\begin{aligned} A_1 &= [\ldots x \ldots 1 \ldots] \\ A_2 &= [\ldots x \ldots 2 \ldots] \\ A_3 &= [\ldots x \ldots 3 \ldots] \\ A_4 &= [\ldots x \ldots 4 \ldots] \\ A_5 &= [\ldots x \ldots 5 \ldots] \end{aligned}$$

We'll abbreviate the unknown value with $*$, so we can say $A = [\ldots x \ldots * \ldots]$ is a set of possible previous arrays. Then this set A is reachable because all arrays in A are reachable.

Subclaim 1: $*$ is swappable.

If $A = [\dots\ x\ *\ \dots]$ is reachable, then $A' = [\dots\ *\ x\ \dots]$ is reachable. The reverse direction is also true.

Proof of subclaim:

1. $A = [\dots\ x\ *\ \dots]$ contains the array $A_x = [\dots\ x\, x\ \dots]$, so A_x is reachable.
2. A previous operation from A_x is to replace the left value with x, so $A' = [\dots\ *\ x\ \dots]$ is reachable.

The proof for swapping in the other direction is identical.

$\square$

Subclaim 2: $*$ can swap adjacent elements.

If $A = [\dots\ x\, y\, * \dots]$ is reachable, then $A' = [\dots\ y\, x\, * \dots]$ is reachable.
The $*$ can also be to the left of x and y.

Proof of subclaim:

1. $A = [\dots\ x\, y\, *\ \dots]$ contains $A_x = [\dots\ x\, y\, x \dots]$, so A_x is reachable.
2. A previous operation from A_x shows $[\dots\ *\, y\, x \dots]$ is reachable.
3. By Subclaim 1, we can swap $*$ twice, so $A' = [\dots y\, x\, * \dots]$ is reachable.

The proof where the $*$ is on the left is identical.

$\square$

Proving the main claim:
These two subclaims allow us to rearrange an array however we want (via bubble sort).

This is enough to show that any array A containing every element in B is reachable. One way to achieve this is with the following procedure:

1. Start with B, and obtain $*$ anywhere in the array.
2. Sort the array using the $*$.
3. **Replace any duplicates** with $*$.
4. **Rearrange the unique elements** so that each one matches an element in A.
5. **Replace all $*$'s** with the corresponding element of A.

Example: $k = 2$, $A = [1, 3, 2, 1, 4, 2, 1]$, $B = [3, 1, 3, 1, 1, 2, 1]$.

$$\begin{aligned}
B &= [3, 1, 3, 1, 1, 2, 1] && \text{Start} \\
B_1 &= [*, 1, 3, 1, 1, 2, 1] && \text{Obtain a } * \\
B_2 &= [1, 1, 1, 1, 2, 3, *] && \text{Sort} \\
B_3 &= [1, *, *, *, 2, 3, *] && \text{Replace duplicates} \\
B_4 &= [1, 3, 2, *, *, *, *] && \text{Rearrange to match } A \\
B_5 &= [1, 3, 2, 1, 4, 2, 1] && \text{Replace } *\text{'s}
\end{aligned}$$

$B_5 = A$, so A is reachable.

☐

Final algorithm.

Suppose $k \geq 2$. There are three cases:

1. If $A = B$, output `Yes`.
2. If B contains an element not in A, output `No`.
3. Otherwise, determine if there exist distinct indices i and j such that $|i - j| \leq k$ and $B_i = B_j$. If they exist, output `Yes`. Otherwise, output `No`.

These checks can be implemented in $O(n)$ or $O(n \log n)$ time, which is fast enough.

Submission: https://atcoder.jp/contests/arc183/submissions/57139382
Code (C++20):

```
#include "bits/stdc++.h"

using namespace std;

bool solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n), b(n);
    for (int i = 0; i < n; i++) cin >> a[i];
    for (int i = 0; i < n; i++) cin >> b[i];

    if (k == 1) {
        for (int i = 0, j = 0; j < n; j++) {
            while (i < n && a[i] != b[j]) i++;
            if (i >= n) return false;
        }
        return true;
    }
```

```cpp
    // k >= 2

    // 1.
    if (a == b) return true;

    // 2.
    set<int> sa(begin(a), end(a));
    for (int x : b) {
        if (!sa.contains(x)) return false;
    }

    // 3.
    map<int, int> m; // m[x] = index of rightmost occurrence of x so far
    for (int i = 0; i < n; i++) {
        if (m.contains(b[i]) && i - m[b[i]] <= k) return true;
        m[b[i]] = i;
    }
    return false;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        cout << (solve() ? "Yes" : "No") << '\n';
    }
    return 0;
}
```