I. PIVOT

PROBLEM DESCRIPTION

Given a “pivot’d” array (the one used in quicksort). Determine how many elements in the array could have been the chosen pivot.

SOLUTION TECHINQUES

Dynamic Programming

SOLUTION SKETCHES

An element a[i] could have been the pivot,
if all the elements on the left side ≤ a[i] and a[i] < all the elements on the right side.

To determine whether this is true, we only need to know whether a[i] is greater than or equal to the greatest element of the left side and whether a[i] is less than the smallest element on the right side. This can be efficiently with Dynamic Programming using a prefix-sum-like approach.

Let f[i] be $\max\limits_{0 \le j \le i} a_j$ and b[i] be $\min\limits_{i \le j \le n-1} a_j$

$$f[i] = \begin{cases} a_0, & i = 0 \\ \max(f[i-1], a_i, & i > 0 \end{cases}$$

And likewise for b[i].

An element a[i] could have been the pivot if $f[i-1] \le a[i] < b[i+1]$.

TIME COMPLEXITY

O(n)

SOLUTION PROGRAM FOR REFERENCE

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;

const int N = 1e5 + 5;

int a[N];
int f[N], b[N];

int main()
{
    int i, n, ans = 0;
    scanf("%d", &n);
    for (i = 0; i < n; i++) scanf("%d", &a[i]);
    f[0] = a[0];
    for (i = 1; i < n; i++)
        f[i] = max(f[i - 1], a[i]);
    b[n - 1] = a[n - 1];
    for (i = n - 2; i >= 0; i--)
        b[i] = min(b[i + 1], a[i]);
    if (a[0] < b[1]) ans++;
    if (f[n - 2] <= a[n - 1]) ans++;
    for (i = 1; i <= n - 2; i++)
        if (f[i - 1] <= a[i] && a[i] < b[i + 1])
            ans++;
    printf("%d\n", ans);
    return 0;
}
```