Joker

Calvin Fung

July 13, 2025

1 Problem

Consider a deck of cards. The positions in the deck are numbered from 1 to n from top to bottom. A joker is located at position m .

operations are applied sequentially to the deck. During the i-th operation, you need to take the card at position a_i and move it either to the beginning or to the end of the deck. For example, if the deck is [2,1,3,5,4] , and $a_i = 2$, then after the operation the deck will be either [1,2,3,5,4] (the card from the second position moved to the beginning) or [2,3,5,4,1] (the card from the second position moved to the end).

Your task is to calculate the number of distinct positions where the joker can be after each operation.

2 Solution

We present a linear time ie. $O(q)$ solution.

2.1 analysis

Given a set of possible positions of the joker $S \subseteq \{1, ..., n\}$, let $f_k(S)$ be the set of possible positions of the joker after applying operation k. It's easy to see that given any z sets we have $\cup_{j=1}^{z} f_k(S_j) = f_k(\cup_{j=1}^{z} S_j)$.

Given 3 indexed intervals I_1, I_2, I_3 of $\{1, ..., n\}$, we say they satisfy property $(*)$ if satisfy all of below :
i. I_1 always starts at 1 if it's non-empty and I_3 always ends at n if it's non-empty.
ii. I_1 is non-empty iff I_3 is non-empty.

Let I_1, I_2, I_3 be intervals (may not be disjoint) that satisfy property $(*)$. Suppose $\cup_{j=1}^{3} I_j$ represents all possible positions where the joker can be after operations $1, ..., k$. We first show how to compute $f_{k+1}(I_j), j = 1, 2, 3$. Suppose before apply operation $k+1$, the joker is at a position in I_j, there're several cases :

(notice $I_j.l, I_j.r$ denote the start and end of the interval respectively)

$a_{k+1} > I_j.r$: Then when the card at position a_{k+1} is moved to the back, the joker's position will not change, thus all positions in I_j remains possible after operation $k+1$. When the card at position a_{k+1} is moved to the front, the joker's position will increase by 1, thus a new position $I_j.r + 1$ becomes possible after operation $k+1$. Therefore, $f_{k+1}(I_j) = [I_j.l, I_j.r + 1]$.

$a_{k+1} < I_j.l$: Similar reasoning as above, $f_{k+1}(I_j) = [I_j.l - 1, I_j.r]$.

$a_{k+1} \in I_j$: Clearly if the joker was at position a_{k+1}, it'll be at either 1 or n after operation $k+1$, so we consider other cases below.

(when the card at position a_{k+1} is moved to the front) If the joker was at a position in $[a_{k+1} + 1, I_j.r]$, its position will not change after the operation, thus $[a_{k+1} + 1, I_j.r]$ remains possible. Otherwise the joker was at a position in $[I_j.l, a_{k+1} - 1]$, after the operation its position will increase by 1 so $[I_j.l + 1, a_{k+1}]$ is possible.

(when the card at position a_{k+1} is moved to the back) If the joker was at a position in $[a_{k+1}+1, I_j.r]$, its position will decrease by 1 after the operation, thus $[a_{k+1}, I_j.r-1]$ is possible. Otherwise the joker was at a position in $[I_j.l, a_{k+1}-1]$, after the operation its position will not change so $[I_j.l, a_{k+1}-1]$ remains possible.

Overall

$$f_{k+1}(I_j) = [a_{k+1}+1, I_j.r] \cup [I_j.l+1, a_{k+1}] \cup [a_{k+1}, I_j.r-1] \cup [I_j.l, a_{k+1}-1] \cup \{1, n\}$$

This is equal to $I_j \cup \{1, n\}$ when $|I_j| > 1$ and equal $\{1, n\}$ otherwise.

We now show how to produce intervals I'_1, I'_2, I'_3 that satisfy property $(*)$ and that $\cup_{j=1}^{3} I'_j$ represents all possible positions where the joker can be after operations $1, ..., k+1$. For $j = 1, 2, 3$:

$a_{k+1} > I_j.r$ or $a_{k+1} < I_j.l$: let $I'_j = f_{k+1}(I_j)$

$a_{k+1} \in I_j$: If $j \in \{1, 3\}$, then $I'_j = I_j$.

Otherwise, if $|I_j| > 1$, set $I'_j = I_j$, else $I'_j = \emptyset$. Moreover, if I_1 is empty, set $I'_1 = [1, 1]$ and $I'_3 = [n, n]$.

It's not hard to see they satisfy property $(*)$, to see why $\cup_{j=1}^{3} I'_j$ represents all possible positions where the joker can be after operations $1, ..., k+1$, notice $\cup_{j=1}^{3} I'_j = \cup_{j=1}^{3} f_{k+1}(I_j) = f_{k+1}(\cup_{j=1}^{3} I_j)$.

2.2 algorithm

The algorithm runs in iterations as described above, along with base cases. It runs in linear time.

```
f(n,m,q,A){
        I_1, I_2, I_3 = empty intervals
        //base case
        if(A[1] == m){
                I_1 = [1,1]
                I_3 = [n,n]
        }else if(A[1] > m){
                I_2 = [m,m+1]
        }else{
                I_2 = [m-1,m]
        }

        for(i = 2;i <= q; ++i){
                for each interval I_j ≠ ∅, j = 1,2,3 {
                        if(I_j.l <= A[i] <= I_j.r){
                                /*
                                notice |I_2| is either 0 or >1 after the base case
                                so in this case we can always set I'_2 = I_2
                                */
                                if(I_1 == ∅){
                                                I_1 = [1,1]
                                                I_3 = [n,n]
                                }
                        }else if (A[i] > I_j.r){
                                I_j = [l,r+1]
                        }else{
                                I_j = [l-1,r]
                        }
                }

                //make them disjoint, clearly this doesn't affect correctness
                if(I_1.r == I_2.l){
                        I_1 = empty
                        I_2 = [1,I_2.r]
                }
                if(I_3.l == I_2.r){
                        I_3 = empty
```

```
                    I_2 = [I_2.l,n]
            }
    }

    return |I_1| + |I_2| + |I_3|
}
```