Smallest Rotation with Highest Score

Given an array `A`, we may rotate it by a non-negative integer `K` so that the array becomes `A[K], A[K+1], A{K+2], ... A[A.length - 1], A[0], A[1], ..., A[K-1]`. Afterward, any entries that are less than or equal to their index are worth 1 point.

For example, if we have `[2, 4, 1, 3, 0]`, and we rotate by `K = 2`, it becomes `[1, 3, 0, 2, 4]`. This is worth 3 points because 1 > 0 [no points], 3 > 1 [no points], 0 <= 2 [one point], 2 <= 3 [one point], 4 <= 4 [one point].

Over all possible rotations, return the rotation index K that corresponds to the highest score we could receive. If there are multiple answers, return the smallest such index K.

Example 1:
Input: [2, 3, 1, 4, 0]
Output: 3
Explanation:
Scores for each K are listed below:
K = 0, A = [2,3,1,4,0], score 1
K = 1, A = [3,1,4,0,2], score 3
K = 2, A = [1,4,0,2,3], score 3
K = 3, A = [4,0,2,3,1], score 4
K = 4, A = [0,2,3,1,4], score 3

So we should choose K = 3, which has the highest score.

Example 2:
Input: [1, 3, 0, 2, 4]
Output: 0
Explanation: A will always have 3 points no matter how it shifts.
So we will choose the smallest K, which is 0.

Note:

- `A` will have length at most `20000`.
- `A[i]` will be in the range `[0, A.length]`.

Solution 1

F*ck the examples.

written by xmeng525 original link here

Solution 2

Key point
Don't calculate the score for `K=0`, we don't need it at all.
(I see almost all other solutions did)
The key point is to find out how score changes when `K++`

Time complexity:
"A will have length at most 20000."
I think it means you should find a O(N) solution.

Explanation:

1. Search the index where score changes and record the changement to a list.
2. A simple for loop to calculate the score for every `K` value.
3. Find the index of best score.

What value of K changes score?
a) get point
Each time when we rotate, we make index `0` to index `N-1`, then we get one more point.
b) loss point
`(i - A[i] + N) % N` is the value of K making `A[i]'s` index just equal to `A[i]`.
For example, If `A[6] = 1`, then `K = (6 - A[6]) % 6 = 5` making `A[6]` to `index 1` of new array.
So when `K=5`, we get this point for `A[6]`
Then if `K` is bigger when `K = (i - A[i] + 1) % N`, we start to lose this point, making our score -= 1
All I have done is record the value of `K` for all `A[i]` where we will lose points.

c) `A[i]=0`
Rotation makes no change for it, becasue we alwars have `0 <= index`.
However, it is covered in `a)` and `b)`

C++:

```
    int bestRotation(vector<int>& A) {
        int N = A.size();
        int change[N] = {0};
        for (int i = 0; i < N; ++i) change[(i - A[i] + 1 + N) % N] -= 1;
        for (int i = 1; i < N; ++i) change[i] += change[i - 1] + 1;
        return distance(change, max_element(change, change + N));
    }
```

Java

```java
    public int bestRotation(int[] A) {
        int N = A.length;
        int change[] = new int[N];
        for (int i = 0; i < N; ++i) change[(i - A[i] + 1 + N) % N] -= 1;
        int max_i = 0;
        for (int i = 1; i < N; ++i) {
            change[i] += change[i - 1] + 1;
            max_i = change[i] > change[max_i] ? i : max_i;
        }
        return max_i;
    }
```

Python

```python
    def bestRotation(self, A):
        N = len(A)
        change = [1] * N
        for i in range(N): change[(i - A[i] + 1) % N] -= 1
        for i in range(1, N): change[i] += change[i - 1]
        return change.index(max(change))
```

written by lee215 original link here

Solution 3

```
class Solution {
    public int bestRotation(int[] A) {
        int len = A.length;

        // Store scores for each K value
        int[] kScores = new int[len];

        for (int i = 0; i < len; i ++) {
            int v = A[i];
            // Ideal K is the K that puts A[i] in location i.
            int idealK = (len - v + i) % len;
            // Increment kScores for all possible K's s.t. v in A rotated by K will
improve the score.
            for (int j = 0; j < len - v; j ++) {
                kScores[idealK] = kScores[idealK] + 1;
                idealK --;
                if (idealK < 0) idealK = len - 1;
            }
        }

        // Get the best K
        int bestK = 0;
        int bestScore = -1;
        for (int k = 0; k < len; k ++) {
            int score = kScores[k];
            if (score > bestScore) {
                bestK = k;
                bestScore = score;
            }
        }

        return bestK;
    }
}
```

written by JLepere2 original link here