Abstract

Given an array with n integers, your task is to find the number of subarrays having at most k distinct values.

Tutorial

There is one brute force solution with time complexity $O(n^3)$. The idea is to use 3 loops to traverse all subrrays, one loop to iterate the start index, one for the end and other for calculate the number of distinct values of the subarray. The pseudo code as follow:

```
cnt = 0
for i in [1:n] # start index
    for j in [i:n] # end index
        distinct_value = calc_distinct_values(a[i:j])
        if distinct_value <= k:
            cnt += 1
return cnt
```

But can we reduce the complexity to $O(n^2)$?

Observation 1

Let's take a deep look into the function `calc_distinct_values`. The idea of this function is to use a marker array (1 for existed value, else 0). It can be implement as below:

```
mark = []
distinct_cnt = 0
for p in a[i:j]:
    if mark[p] == 0:
        distinct_cnt += 1
    mark[p] = 1
return distinct_cnt
```

We can literally remove this function and calculating the number of distinct values parallel with iterating the end indexs (j in the code).

```
    cnt = 0
    for i in [1:n] # start index
        mark = []
        distinct_cnt = 0
        for j in [i:n] # end index
            p = a[j]
            if mark[p] == 0:
                distinct_cnt += 1
            mark[p] = 1

            if distinct_value <= k:
                cnt += 1
    return cnt
```

How about $O(n)$?

Observation 2

For each i, iterate j until the number of distinct values exceeds k. Then all subarrays have whose end index $g \geq j$ also has number of distinct values exceeds k.

Observation 3

If we know the number of distinct values $a[i, j]$, we also can calculate the answer of $a[i, x - 1] + a[x + 1, j]$. Let's modify the marker array above to be a counting array (call it `cnt_value[]`).

```
    # remove value a[x] from subarray and update the number of distinct values
    p = a[x]
    cnt_value[p] -= 1
    if cnt_value[p] == 0:
        distinct_cnt += 1
```

Apply this idea to remove $a[i]$ from $a[i, j]$.

Using 2 observations above to optimize the algorithm.

```
    cnt_value = []

    cnt = 0
    distinct_cnt = 0
    j = 1
    for i in [1:n] # start index
        while j <= n and distinct_cnt <= k:
            p = a[j]

            # add j to [i, j-1]
            if cnt_value[p] == 0:
                distinct_cnt += 1
            cnt_value[p] += 1

            j += 1

        cnt += (j - i)

        p = a[i]

        # remove i from [i, j]
        cnt_value[p] -= 1
        if cnt_value[p] == 0:
            distinct_cnt += 1

    return cnt
```