Question 2

First, we need to iterate through the whole array A to find all the beauties corresponding to the index and store them in array B.

To do this we need two hash tables, hash table A is used to record the current number of occurrences of each value, hash table B is used to record the current number of occurrences of each value inside hash table A, that is, the number of occurrences of the number of occurrences of each value of array A.

We need to initialize the first value of the hash table B to m. We also need to store the current beauty with an integer, initialized to 0.

While traversing the entire array A and updating the two hash tables, if the value of index corresponding to the current beauty value in hash table B is 0, then the current beauty value +1. The last step of each iteration is to append the current beauty value to the array B.

The second step is to traverse the array B. If B[i] > B[i-1] then i is the fulfilling index. After this step is done all fulfilling indices that satisfy A[1..i] has strictly greater beauty than A[1..i-1] are found.

Given that we do not have nested loops to process the array A and all operations are done in linear time, the time complexity of this algorithm is $\boldsymbol{O(n)}$.

Supplement Python code

```python
def q2_solution(a,m):
    answer = []
    occurrence = [0] * (m+1) # hash table A
    times = [0] * (len(a)+1) # hash table B
    times[0] = m
    curr_beauty = 0

    for i in a:
        times[occurrence[i]] -= 1
        occurrence[i] += 1
        times[occurrence[i]] += 1
        if times[curr_beauty] is 0:
            curr_beauty += 1
        answer.append(curr_beauty)

    print(answer)
    print("All fulfilling indices:")
    for i in range(len(answer)):
        if(answer[i] > answer[i-1]):
            print(i, end=' ')

q2_solution([1,3,1,2,3,3,2,3],3)

# Output
# [0, 0, 0, 1, 1, 1, 2, 2]
# All fulfilling indices:
# 3 6
```