Fruits into Basket.

05 August 2025 14:58

→ We will keep a map to keep track of no of unique elements and window to get longest subarray.

In this question, We are getting asked find the longest substring with atmost 2 different fruits

[1,2,3,2,2]

l, r

map:

nums[r] = 1

map.getOrDefault(nums[r], map.getOrDefault(nums[r],0)+1); 0+1 = 1

if(map.size() > 2) → false

max = [max, r-l+1] = 1

1,1

l=0 ; r=1

nums[r] = 2

map.getOrDefault(nums[r], map.getOrDefault(nums[r],0)+1); 0+1 = 1

if(map.size() > 2) → false

max = [max, r-l+1] = 2

2,1
1,1

l=0 ; r=2

nums[r] = 3

map.getOrDefault(nums[r], map.getOrDefault(nums[r],0)+1); 0+1 = 1

while(map.size() > 2) true

↳ map.put(nums[l], map.get(nums[l])-1) 1 − 1 = 0

if(map.get(nums[l]) == 0) map.remove(nums[l]);

l++ = 1

map.size = 2 ⇒ got out of loop.

math.max(max, r-l+1); 2

r++

3,1
2,1
1,1

l=1 r=3

nums[r] = 2

map.getOrDefault(nums[r], map.getOrDefault(nums[r],0)+1); 1+1 = 2

if(map.size() > 2) → false

max = [max, r-l+1] = 3

r++

3,1
2,2

l=1 r=4

nums[r] = 3

map.getOrDefault(nums[r], map.getOrDefault(nums[r],0)+1);

while(map.size() > 2) false

max = [max, r-l+1] ⇒ 4

3,2
2,2

□ Step-by-Step Logic:

1. Initialize:
   - A `HashMap` (or array if fruit types are small) to track fruit counts.
   - Two pointers: `l` (left), `r` (right).
   - Variable `maxLen` to store the answer.
2. Expand the window (move `r`):
   - Add `fruits[r]` to the map → increase its count.
3. Check window validity:
   - If map size > 2, the window is invalid.
   - While invalid, shrink from left (`l++`), and reduce frequency of `fruits[l]`.
   - Remove fruit type from map if its count becomes 0.
4. Update max length:
   - After each valid window, do `maxLen = max(maxLen, r - l + 1)`.
5. Repeat until end of array.

nums[i] = 3

map.getOrDefault(nums[i], map.getOrDefault(nums[i], 0) + 1);

while (map.size() > 2)

max = [3 max, 4 i - i + 1] ⇒ 4

3, 2
2, 2