1.3 SUM OF SQUARES OF DISTINCT COUNTS

Question:

You are given a 0-indexed integer array nums. The distinct count of a subarray of nums is defined as: Let nums[i..j] be a subarray of nums consisting of all the indices from i to j such that 0 <= i <= j < nums.length. Then the number of distinct values in nums[i..j] is called the distinct count of nums[i..j]. Return the sum of the squares of distinct counts of all subarrays of nums. A subarray is a contiguous non-empty sequence of elements within an array.

AIM:

To compute the sum of the squares of distinct counts of all possible subarrays of a given integer array.

ALGORITHM:

1. Initialize total sum = 0.
2. Generate all possible subarrays:
    - Outer loop for starting index i.
    - Inner loop for ending index j.
3. For each subarray nums[i..j]:
    - Use a set to find the distinct elements.
    - Let count = size of set.
    - Add count * count (square of distinct count) to total sum.
4. Return total sum after processing all subarrays.

PROGRAM:

```
def sum_of_squares(nums):
    total = 0
    for i in range(len(nums)):
        seen = set()
        for j in range(i, len(nums)):
            seen.add(nums[j])
            total += len(seen) ** 2
    return total

nums = list(map(int, input("Enter nums: ").split()))
print("Sum of squares of distinct counts:", sum_of_squares(nums))
```

Input:

nums = [1,2,1]

Output:

```
Enter nums: 1 2 1
Sum of squares of distinct counts: 15
>>>
```

RESULT:

Thus the program is successfully executed, and the output is verified.

PERFORMANCE ANALYSIS:

Time Complexity:

- Generating all subarrays = O(n^2)
- Counting distinct using set (up to O(n) each)
- Worst case = O(n^3)
- For small n this is fine.

Space Complexity:

- Temporary set of size up to O(n)
- So overall = O(n)