Product of Three Largest Distinct Elements Using Priority Queue

Mr. Akash Yadav
Assistant Professor
Artificial Intelligence & Data Science

Lesson Plan

Subject/Course	**Competitive Coding**
Lesson Title	**Product of Three Largest Distinct Elements Using Priority Queue**

Lesson Objectives
Identify and extract the three largest unique elements from an array using a max-heap (priority queue).
Implement insertion, extraction, and duplicate handling for O(n log n) efficiency.
Compute products of top elements while ensuring distinctness to avoid errors in real-world data.
Apply priority queues to top-k problems, boosting speed in interviews and contests.

Problem Statement:

Write a Program for finding the Product of the **three largest Distinct** Elements. Use a Priority Queue to efficiently find and remove the largest elements.

Key Concepts & Prerequisites

- A Priority Queue is a data structure where each element has a priority — the highest-priority element is served first.
- It can be implemented as a Heap:
  - Min-Heap → smallest element at top
  - Max-Heap → largest element at top

Why use it here?

- To find the **3 largest distinct elements**, sorting takes O(n log n).
- A **Priority Queue** can do it faster: O(n log k) for k = 3 → practically O(n).

Distinct Elements

- We must ensure all elements are unique before finding the top 3.
  → Use a **set** to remove duplicates.

Key Concepts & Prerequisites

Prerequisites:

- Arrays from basics;
- Queues from Q4;
- Understand heap property (parent > children).

Algorithm/Logic

High-Level Approach

1. Read array and n.
2. Push all elements to max-heap priority_queue.
3. Pop largest, add to result list; use set to skip if duplicate.
4. Repeat for 3 unique; if <3, return error.
5. Compute product of the three.

Algorithm/Logic

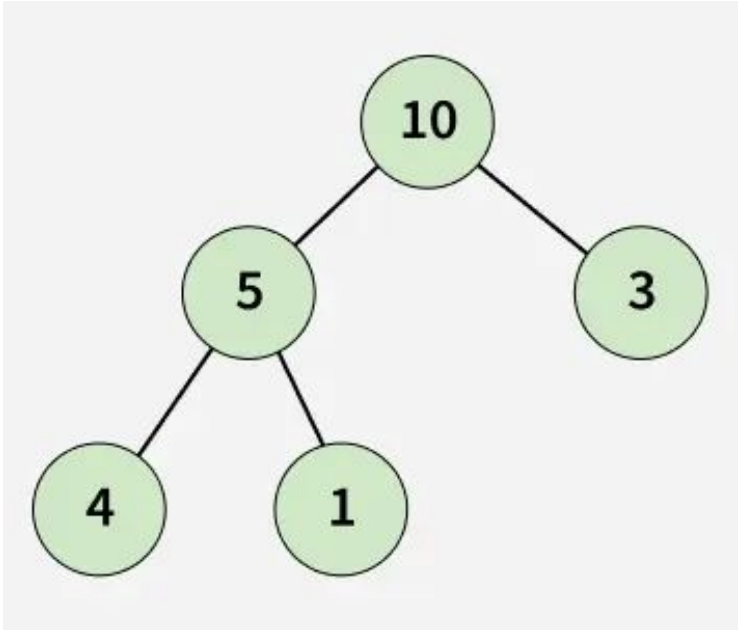
Detailed Algorithm

1. Input: An array arr of integers.
2. Remove duplicates using a set → ensures only distinct elements.
3. Create a Max-Heap:
   - Since Python's *heapq* is a min-heap, insert negative values to simulate a max-heap.
4. Extract top 3 elements by popping the heap 3 times.Compute product of those 3 numbers.
5. Return / print the result.

Visualization

Example Input:
arr = [4, 7, 2, 7, 5, 9, 9]

Step 1: Remove duplicates
→ {2, 4, 5, 7, 9}



Step 3: Extract Top 3 → 9, 7, 5
Step 4: Product = 9 × 7 × 5 = 315

Step	Heap (max-heap view)	Extracted	Product
1	[9, 7, 5, 4, 2]	9	9
2	[7, 4, 5, 2]	7	63
3	[5, 4, 2]	5	315

Code Implementation

```java
import java.util.*;

public class Practical6_ProductThreeLargest {

    public static int product(int[] arr) {
        Set<Integer> set = new HashSet<>();
        for (int x : arr) set.add(x); // remove duplicates

        PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
        pq.addAll(set);

        if (pq.size() < 3) return -1; // not enough distinct elements
        int a = pq.poll();
        int b = pq.poll();
        int c = pq.poll();
        return a * b * c;
    }
```

```
19      public static void main(String[] args) {
20          int[] arr = {10, 3, 5, 6, 20};
21          System.out.println("Array: " + Arrays.toString(arr));
22          System.out.println("Product of 3 largest distinct elements: " + product(arr));
23      }
24  }
```

Output :

```
Array: [10, 3, 5, 6, 20]
Product of 3 largest distinct elements: 1200
```

Complexity Analysis

Aspect	Complexity	Why?
Time	O(n log n)	n inserts/pops at O(log n); Set ops O(log k) for k=3.
Space	O(n)	Heap stores all n; Set O(1) since k=3 fixed.

Variations & Competitive Tips

- **Easier** Variation: No distinct check (just pop 3).
- **Harder**: K largest (generalize to input k); Streaming input (no full array).
- **Competitive Angle**: LeetCode #414: Third Largest; Codeforces "Top K" problems.
- **Optimization**: For huge n, use partial_sort (O(n log k)) instead of full heap.

Practice Exercises

1️⃣ **Third Maximum Number — LeetCode #414**

🔗 https://leetcode.com/problems/third-maximum-number/

Concept: Finding the 3rd largest distinct element in an array.

Why Practice: Directly mirrors the logic of selecting top 3 unique numbers.

Practice Exercises

2️⃣ **Kth Largest Element in an Array — LeetCode #215**

🔗 https://leetcode.com/problems/kth-largest-element-in-an-array/

Concept: Using a priority queue (heap) to find kth largest value efficiently.

Why Practice: Strengthens heap operations and selection of maximum elements.

Practice Exercises

3️⃣ Maximum Product of Three Numbers — LeetCode #628

🔗

https://leetcode.com/problems/maximum-product-of-three-numbers/

Concept: Compute product of three largest (or smallest) numbers.

Why Practice: Exactly applies the product-of-largest-elements logic efficiently.

Thanks