🧺 What is a Jagged Array?

📚 Formal Definition:

A **jagged array** in Java is a two-dimensional array where the **rows can have different lengths**. Unlike regular 2D arrays (which form a perfect rectangle), jagged arrays form an **irregular grid**. Technically, it's an array of arrays, where each sub-array can vary in size.

🇮🇳 Real-Life Analogy:

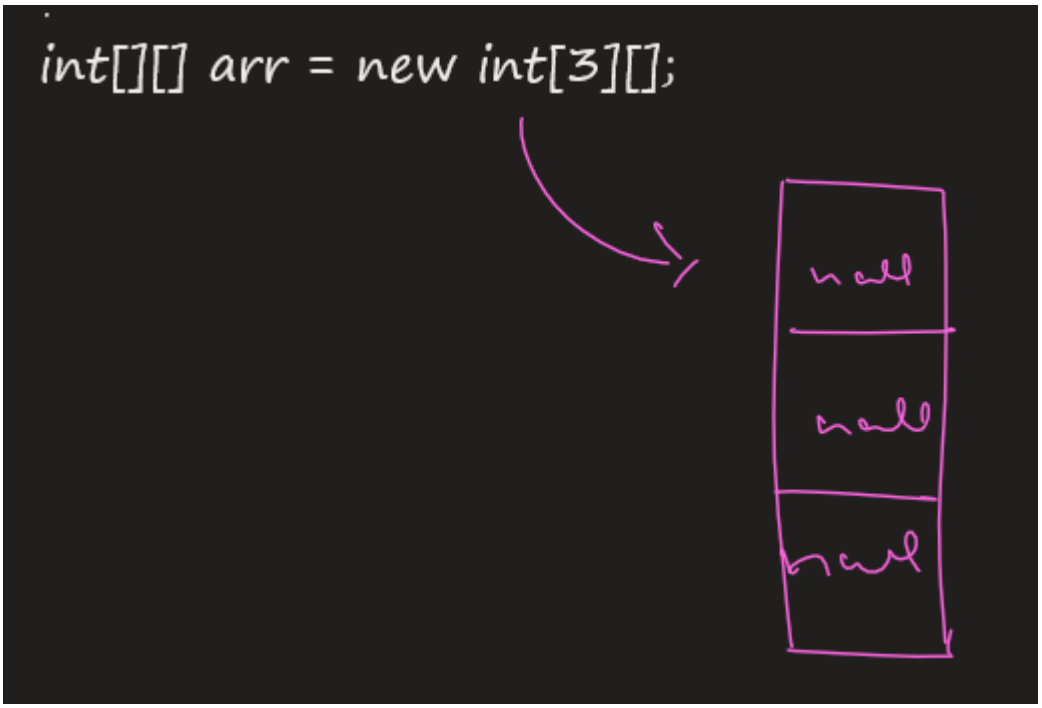
Imagine visiting a **sabzi mandi** where each vendor has a different number of baskets:

- Vendor 1 has 2 baskets
- Vendor 2 has 3 baskets
- Vendor 3 has just 1 basket

You want to record the prices in a chart—but each row (vendor) has a different number of columns (baskets). That's a jagged array!

```
int[][] jagged = new int[3][];
```

~~int[][] arr= new int[][3];~~




```java
public class Main {
      public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

            int n = sc.nextInt();
            int[][] arr = new int[n][];

            // System.out.println(arr[0]);
            // arr[0] = new int[2];
            // arr[1] = new int[3];
            // arr[2] = new int[1];

            for (int i = 0; i < arr.length; i++) {
```

```
                int col = sc.nextInt();
                arr[i] = new int[col];
                for (int j = 0; j < arr[i].length; j++) {
                        arr[i][j] = sc.nextInt();
                }
        }
        // display
        for (int i = 0; i < arr.length; i++) {
                for (int j = 0; j < arr[i].length; j++) {
                        System.out.print(arr[i][j] + " ");
                }
                System.out.println();
        }
    }

}
```

🎯 Majority Element (Moore's Voting Algorithm)

https://leetcode.com/problems/majority-element/

Basically, what happens in India is that people often support candidates who share their ideology. So, for example, if someone is contesting from the Congress party, others who share Congress's ideology will support that candidate otherwise will oppose the candidate. You might have noticed how coalitions are formed based on similar ideologies.

By **supporting** the candidate, I will **increase his vote count**; by **opposing him**, I will **decrease** it. If his **vote count reaches zero**, a **new person becomes the leader** with one vote—his own.

The same kind of thing happens in the **Moore's Voting Algorithm**.

Let me explain:

- The first element comes in, and we assume it's the majority element.
- The next element arrives — if it's the same as the current candidate, we increase the vote count.
- If it's different, we decrease the vote count.
- If the vote count drops to zero, we discard the current candidate and treat the new element as the new majority candidate with one vote.

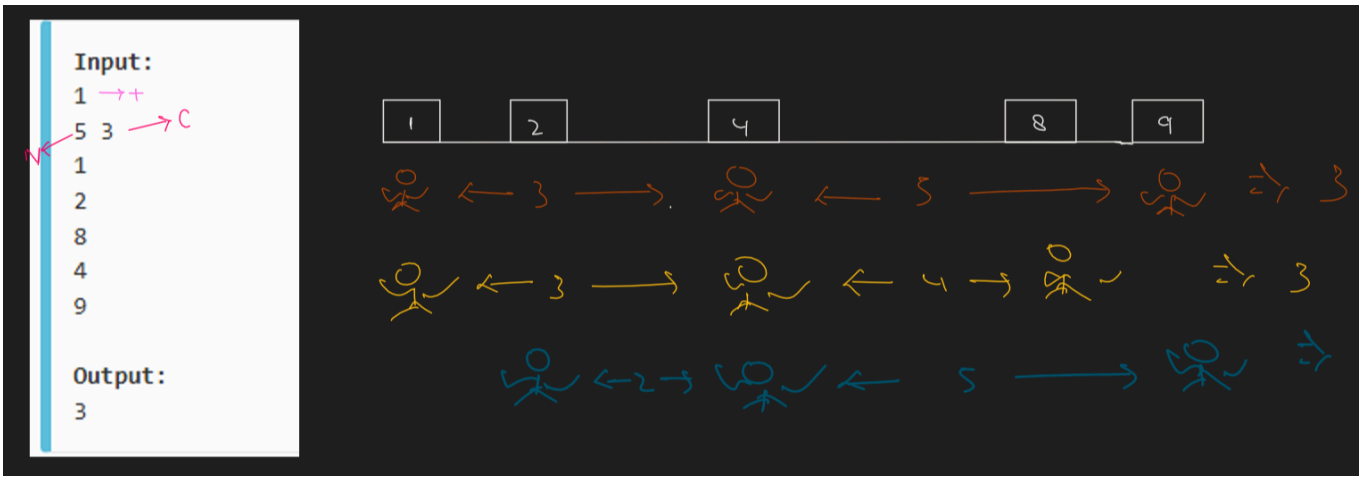
We continue this process through the entire array. By the end, the element that is left as the candidate is considered the majority element — the one with the most support, like a political leader forming the government.

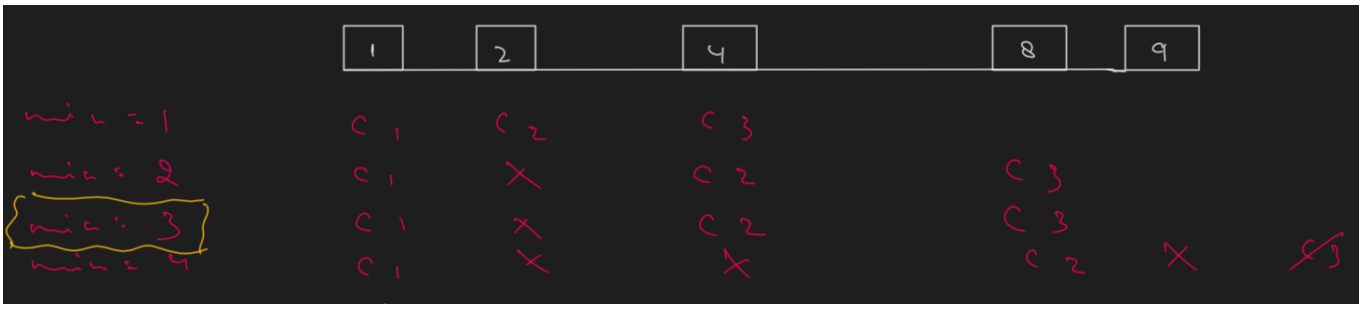
```
class Solution {

    public int majorityElement(int[] nums) {
        int vote = 1;
        int majority = nums[0];
        for(int i=1; i<nums.length; i++){
            if(nums[i] == majority){
                vote++;
            }
            else{
                vote--;
```

```
                if(vote == 0){
                    majority = nums[i];
                    vote = 1;
                }
            }
        }
        return majority;
    }

}
```

🎯 AGGRCOW - Aggressive cows

https://www.spoj.com/problems/AGGRCOW/





Implementation is similar to **“Find the Floor of the K-th Root”** solved earlier.

Binary Search

Searching for ⇒ 1 to maxDist
⇒ 1 to 8 ⇒ Sorted → ?

low : 1
high : 8 → mid : 4 ⇒ canPlaceCows → ? → ✗
⇓
left side → try smaller values
⇓
high : mid - 1 ;

low : 1
high = 4 - 1 = 3 ⇒ mid : 2 ⇒ canPlaceCows → ? → ✓
⇓
one of the answer ⇒ try larger values
⇓
low : mid + 1 ⇐ right side

low : 2 + 1 = 3
high : 3 ⇒ 3 ⇒ canPlaceCows → ? → ✓
⇓
one of the answer ⇒ try larger values
⇓
low : mid + 1 ⇐ right side

low : 3 + 1 = 4 ⇒ while (low < high) → ✗
high : 3

```java
import java.util.*;

public class Main {

    // Checks if cows can be placed with at least 'minDist' distance apart O(N)
    public static boolean canPlaceCows(int[] stalls, int cows, int minDist) {
        int count = 1;
        int lastPos = stalls[0];

        for (int i = 1; i < stalls.length; i++) {
            if (stalls[i] - lastPos >= minDist) {
                count++;
                lastPos = stalls[i];
                if (count == cows) {
                    return true;
                }
```

```java
            }
        }
        return false;
    }

    // Brute force implementation (O(maxDist * N))
    public static int bruteForceAggCows(int[] stalls, int cows) {
        Arrays.sort(stalls);
        int maxDist = stalls[stalls.length - 1] - stalls[0];
        int best = 0;

        for (int d = 1; d <= maxDist; d++) {
            if (canPlaceCows(stalls, cows, d)) {
                best = d;
            } else {
                break;
            }
        }
        return best;
    }

    // Binary Search implementation (O(N log(maxDist)))
    public static int binarySearchAggCows(int[] stalls, int cows) {
        Arrays.sort(stalls);
        int low = 1;
        int high = stalls[stalls.length - 1] - stalls[0];
        int best = 0;

        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (canPlaceCows(stalls, cows, mid)) {
                best = mid;       // Try for larger distance
                low = mid + 1;
            } else {
                high = mid - 1;   // Try for smaller distance
            }
        }
        return best;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();

        while (t-- > 0) {
            int n = sc.nextInt();
            int c = sc.nextInt();
            int[] stalls = new int[n];
            for (int i = 0; i < n; i++) {
                stalls[i] = sc.nextInt();
            }

            // int result = bruteForceAggCows(stalls, c);
```

```
            int result = binarySearchAggCows(stalls, c);

            System.out.println(result);
        }
    }

}
```

🎯 Book Allocation Problem

https://codeskiller.codingblocks.com/problems/1365
https://leetcode.com/problems/split-array-largest-sum/