Majority element using $O(1)$ memory

Anil Maheshwari

School of Computer Science
Carleton University
Canada
Majority Element
**Finding the Majority Element**

**Input:** A stream consisting of $n$ elements and it is given that it has a majority element, i.e. it occurs at least $1 + \lceil \frac{n}{2} \rceil$ times

**Output:** The majority element.

An Example: $n = 19$

Input Stream = [3 2 4 7 2 2 3 2 2 1 4 2 2 2 1 1 2 3 2]
Finding the Majority Element

Input: A stream consisting of \( n \) elements and it is given that it has a majority element.
Output: The majority element.

Solution 1: Store the stream in an array \( A \).
Sort and pick the middle element (if elements can be ordered).

Input: 3 2 4 7 2 2 3 2 2 1 4 2 2 2 1 1 2 3 2

Sorted: 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 4 4 7
Finding the Majority Element

Input: A stream consisting of \( n \) elements and it is given that it has a majority element.

Output: The majority element.

Solution 2: Count frequency of each element.

Input: 3 2 4 7 2 2 3 2 2 1 4 2 2 2 1 1 2 3 2

<table>
<thead>
<tr>
<th>Element</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Finding the Majority Element

**Input:** A stream consisting of \( n \) elements and it is given that it has a majority element.

**Output:** The majority element.

Memory required in Solutions 1 & 2 \( \geq \) Number of distinct elements in the stream.

**What if we can only use** \( O(1) \) **space?**
Majority Algorithm

**Input:** Array $A$ of size $n$ consisting a majority element

**Output:** The majority element

1. $c \leftarrow 0$
2. for $i = 1$ to $n$ do
   3. if $c = 0$ then
      4. $\text{current} \leftarrow A[i]; \ c \leftarrow c + 1$
   5. end
   6. else
      7. if $A[i] = \text{current}$ then
         8. $c \leftarrow c + 1$
      9. end
      10. else
          11. $c \leftarrow c - 1$
      12. end
   13. end
14. end
15. return $\text{current}$
for $i = 1$ to $n$ do
    if $c = 0$ then
        current $\leftarrow A[i]; c \leftarrow c + 1$
    end
    else
        if $A[i] = current$ then
            $c \leftarrow c + 1$
        end
        else
            $c \leftarrow c - 1$
        end
    end
end
return current
### Analysis of Majority Algorithm

#### Observations

1. Algorithm maintains only two variables: $c$ and current.
2. Correctness: Each non-majority element can ‘kill’ at most one majority element.

#### Claim

By performing a single pass, using only $O(1)$ additional space, we can report the majority element of $A$ (if it exists).
Generalization
2nd Problem

Generalization

For a data stream \( A \), using very little space, we are interested to report

1. All the elements that occur at least \( \frac{n}{4} \) times.
2. All the elements that occur at least \( \frac{n}{k} \) times for some constant \( k \), i.e. report all \textit{heavy-hitters}. 
**Input:** A stream consisting of \( n \) elements and an integer constant \( k < n \).
**Output:** All the elements that occur at least \( n/k \) times.

1. Initialize \( k \) bins, each with null element and a counter with 0.
2. **For** each element \( x \) in the stream **do**
   - if \( x \in \text{Bin } b \) then increment bin \( b \)'s counter
   - elseif find a bin whose counter is 0 and
     - Assign \( x \) to this bin
     - Assign 1 to its counter
   - else decrement the counter of every bin.
3. Output elements in the bins.
Analysis of Misra and Gries Algorithm

Running Time:
Initializing \( k \) bins: \( O(k) \) time
Processing each element requires looking at \( O(k) \) bins.
Total Run Time = \( O(nk) \)

Space: \( O(k) \)
Correctness: What can be the minimum value of the counter of a heavy hitter?

Claim

Let \( f_x^* \) = Frequency of \( x \) in the stream. Each heavy hitter \( x \) is in one of the bins with counter value \( \geq f_x^* - n/k \).

Proof.

Homework!