VOTING

Gas Optimization Highlights

1. **Use of `immutable` for `owner`:**
   - The `owner` address is set once during deployment and cannot be changed, saving gas when accessed later compared to a mutable state variable.
2. **Compact Data Types (`uint8`):**
   - Votes are stored as `uint8` instead of `uint256` because the maximum value required is small. This reduces storage costs.
3. **Efficient Candidate Tracking:**
   - Candidate votes are stored in a `mapping`, and their existence is verified with an auxiliary `array` (used sparingly).
4. **Bounded Loops:**
   - The loop in `isCandidate` is only used to verify the existence of a candidate and is kept small by limiting the number of candidates.
5. **Storage Minimization:**
   - Only the `votes` and `candidates` arrays are stored persistently, avoiding redundant or large state variables.

Gas Usage Breakdown

Operation	Optimization Applied	Gas Savings
`owner`	`immutable` keyword	Reduces gas costs for read-only access.
`votes`	Use of `uint8` instead of `uint256`	Saves storage space for each entry.
Candidate verification	Separate `mapping` and `array` for efficiency	Reduces storage duplication.
Candidate validation	Limits loop size in `isCandidate` function	Avoids unbounded gas costs.

Key Benefits

- **Lower Gas Costs:** Reducing storage and computation results in significant gas savings.
- **Efficient Access:** Mappings offer `O(1)` lookup times for votes.
- **Minimized Storage Size:** Compact data types (`uint8`) and selective storage of data reduce overall usage.