1 The majority problem

These notes are based on the article *A cultural gap revisited* by A. Shen, which appeared in *The Mathematical Intelligencer, Volume 22, Number 2, 2000, pp. 16–17.*

We are given a set $S$ of $n$ objects ($n \geq 1$), each of which has a color. Furthermore, we are told that there is a *majority color* in $S$, i.e., a color that occurs strictly more than $n/2$ times. We denote this majority color by $mc(S)$. Our task is to find an element of $S$ whose color is equal to $mc(S)$.

We are only allowed to use the operation $\text{same\_color}$. This operation takes two arbitrary elements, say $x$ and $y$, of $S$, and returns the value

$$\text{same\_color}(x, y) = \begin{cases} 
\text{true} & \text{if } x \text{ and } y \text{ have the same color}, \\
\text{false} & \text{otherwise}.
\end{cases}$$

In particular, we cannot determine the color of any element of $S$.

2 The basic algorithm

Our algorithm will be based on the following observation.

**Observation 1** Let $x$ and $y$ be two elements of $S$ that have different colors. Then there is a majority color in the set $S \setminus \{x, y\}$, and

$$mc(S) = mc(S \setminus \{x, y\}).$$

**Proof:** Assume that $mc(S) = \text{red}$. Let $k$ be the number of red elements in $S$. Then we know that $k > n/2$. We have to show that the set $S \setminus \{x, y\}$ contains more than $(n - 2)/2$ red elements.

---

*School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. E-mail: michiel@scs.carleton.ca.*
Case 1: Neither $x$ nor $y$ is red. In this case, the number of red elements in $S \setminus \{x, y\}$ is equal to $k > n/2 > (n - 2)/2$.

Case 2: Exactly one $x$ and $y$ is red. In this case, the number of red elements in $S \setminus \{x, y\}$ is equal to $k - 1 > n/2 - 1 = (n - 2)/2$.

We maintain the following invariant:

1. $S$ is the disjoint union of three sets $N$, $I$, and $D$.
2. All elements of $I$ have the same color.
3. There is a majority color in the set $N \cup I$.
4. $mc(S) = mc(N \cup I)$.

(Remark: $N$ stands for “Not seen yet”; $I$ stands for “Identical colors”; $D$ stands for “Discarded”.)

Here is the basic version of our algorithm:

\[
\begin{align*}
N & := S; I := \emptyset; D := \emptyset; \\
\text{while } N \neq \emptyset & \\
\text{do if } I = \emptyset & \\
& \quad \text{then move one element from } N \text{ to } I \\
& \quad \text{else let } x \text{ be an element of } N; \\
& \quad \quad \text{let } y \text{ be an element of } I; \\
& \quad \quad \text{if } \text{same\_color}(x, y) \\
& \quad \quad \quad \text{then move } x \text{ from } N \text{ to } I \\
& \quad \quad \quad \text{else move } y \text{ from } I \text{ to } D; \\
& \quad \quad \quad \quad \text{move } x \text{ from } N \text{ to } D \\
& \quad \quad \text{endif} \\
& \quad \text{endif} \\
\text{endwhile;} \\
\text{return an arbitrary element of } I \\
\end{align*}
\]

3 A simple representation of the algorithm

Until now, we did not specify how the sets $N$, $I$, and $D$ are represented. There turns out to be a very simple way to do this: Let the elements of $S$ be stored in an array $A[1 \ldots n]$. We will use two indices $i$ and $j$ to represent the sets $N$, $I$, and $D$:

1. $0 \leq i \leq j - 1 \leq n$,
2. $D = A[1 \ldots i]$,
3. $I = A[i + 1 \ldots j - 1]$,
4. \( N = A[j \ldots n] \).

If we “translate” our basic algorithm, then we get the following algorithm:

\[
i := 0; \quad j := 1;
\]
\[
\text{while } j \leq n \text{ do if } j \leq i + 1 \text{ then } j := j + 1 \text{ else if same_color}(A[j], A[i + 1]) \text{ then } j := j + 1 \text{ else } i := i + 1; \quad \text{swap}(A[j], A[i + 1]); \quad i := i + 1; \quad j := j + 1 \text{ endif} \text{ endif} \quad \text{endwhile; return } A[i + 1]
\]

If we change the order of the operations, then we get the following algorithm:

\[
i := 0; \quad j := 1;
\]
\[
\text{while } j \leq n \text{ do if } j \geq i + 2 \text{ and same_color}(A[j], A[i + 1]) = \text{false} \text{ then } i := i + 2; \quad \text{swap}(A[j], A[i]) \text{ endif; } \quad j := j + 1 \text{ endwhile; return } A[i + 1]
\]

**Observation 2** In the pseudocode above, the condition

\[ j \geq i + 2 \text{ and same_color}(A[j], A[i + 1]) = \text{false} \]

is equivalent to the condition

\[ \text{same_color}(A[j], A[i + 1]) = \text{false}. \]

**Proof:** Assume that \( \text{same_color}(A[j], A[i + 1]) = \text{false} \). We have to show that \( j \geq i + 2 \). We know from the invariant that \( j \geq i + 1 \). If \( j = i + 1 \), then \( \text{same_color}(A[j], A[i + 1]) = \text{true} \). Therefore, \( j \neq i + 1 \). It follows that \( j \geq i + 2 \). □

Using this observation, we can further simplify the algorithm, and obtain the final algorithm:
\( i := 0; j := 1; \)
\( \textbf{while } j \leq n \textbf{ do if } \text{same}_\text{color}(A[j], A[i + 1]) = false \textbf{ then } i := i + 2; \)
\( \quad \text{swap}(A[j], A[i]) \)
\( \textbf{endif; } \)
\( j := j + 1 \)
\( \textbf{endwhile; } \)
\( \text{return } A[i + 1] \)