Finding the majority, if it exists

Michiel Smid*

January 28, 2008

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 *same_color*. This operation takes two arbitrary elements, say x and y, of S, and returns the value

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

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 $mc(S) = \mathit{red}$. 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:

$N := S$; $I := \emptyset$; $D := \emptyset$;
while $N \neq \emptyset$
do if $I = \emptyset$
  then move one element from N to I
  else let x be an element of N;
    let y be an element of I;
    if $same\_color(x, y)$
    then move x from N to I
    else move y from I to D;
      move x from N to D
    endif
  endif
endwhile;
return an arbitrary element of I

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 $A[1 \dots n]$. 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. $D = A[1 \dots i]$,

3. $I = A[i + 1 \dots j - 1]$,

4. $N = A[j \ldots n]$.

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

```
i := 0; j := 1;
while j ≤ n
do if j ≤ i + 1
    then j := j + 1
    else if same_color(A[j], A[i + 1])
          then j := j + 1
          else i := i + 1;
                swap(A[j], A[i + 1]);
                i := i + 1;
                j := j + 1
          endif
    endif
endwhile;
return A[i + 1]
```

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

```
i := 0; j := 1;
while j ≤ n
do if j ≥ i + 2 and same_color(A[j], A[i + 1]) = false
    then i := i + 2;
         swap(A[j], A[i])
    endif;
    j := j + 1
endwhile;
return A[i + 1]
```

Observation 2 *In the pseudocode above, the condition*

$$j \geq i + 2 \text{ and } \mathit{same\_color}(A[j], A[i+1]) = \mathit{false}$$

is equivalent to the condition

$$\mathit{same\_color}(A[j], A[i+1]) = \mathit{false}.$$

Proof: Assume that $\mathit{same\_color}(A[j], A[i+1]) = \mathit{false}$. We have to show that $j \geq i+2$. We know from the invariant that $j \geq i+1$. If $j = i+1$, then $\mathit{same\_color}(A[j], A[i+1]) = \mathit{true}$. 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;
while j ≤ n
do if same_color(A[j], A[i + 1]) = false
    then i := i + 2;
         swap(A[j], A[i])
    endif;
    j := j + 1
endwhile;
return A[i + 1]
```