Notes on the Boyer-Moore Algorithm

In class we discussed the Boyer-Moore algorithm for finding the majority element in an array $A[1..n]$ (i.e., a value that appears in more than half of the array's entries) if such a majority element exists. Even though the algorithm and the problem it solves are quite simple, it takes a bit of work to see exactly why it is correct. Here is a more detailed version of the argument covered in the lecture.

We use the notation $matches(x, i)$ to be the number of occurrences of x in $A[1..i]$ and $nonmatches(x, i)$ to be the number of elements in $A[1..i]$ that are *not* equal to x.

Intuitively, after i iterations of the first loop, *cand* keeps track of the candidate for the majority element in $A[1..i-1]$ and *count* keeps track of its margin of victory. This invariant is stated more precisely on line 4, below.

```
1:  function Boyer-Moore(A[1..n])
2:      count ← 0
3:      for i ← 1..n do
4:          Invariant: if for some x, matches(x, i − 1) > (i−1)/2, then matches(cand, i − 1) > (i−1)/2 and
                        count = matches(cand, i − 1) − nonmatches(cand, i − 1)
5:          if count = 0 then cand ← A[i]
6:          end if
7:          if A[i] = cand then count + +
8:          else count − −
9:          end if
10:     end for
11:     ▷ Now check if cand really is a majority
12:     count ← 0
13:     for i ← 1..n do
14:         Invariant: count = matches(cand, i − 1)
15:         if A[i] = cand then count + +
16:         end if
17:     end for
18:     if count > n/2 then return cand is a majority
19:     else return there is no majority
20:     end if
21: end function
```

Let's check that the first loop's invariant is correct at the beginning of every iteration of that loop. We use $cand_i$ and $count_i$ to denote the values of the variables *cand* and *count* after i iterations of the loop.

Base Case: After 0 iterations of the loop, $i = 1$ and $matches(x, 0) = 0$ for all x, so there is no x with $matches(x, i-1) > \frac{i-1}{2}$.

Induction Step: Let $i \geq 1$. Assume the invariant holds at the start of iteration i. We wish to show that it holds at the start of the iteration $i + 1$.

Assume some element x is a majority element in $A[1..i]$, i.e.,

$$matches(x, i) > \frac{i}{2}. \tag{1}$$

We wish to prove that $matches(cand_i, i) > \frac{i}{2}$ and $count_i = matches(cand_i, i) - nonmatches(cand_i, i)$. We consider two cases.

Case 1: $count_{i-1} = 0$. If there were a majority element in $A[1..i-1]$, then by the induction hypothesis, $count_{i=1}$ would have to be its margin, which would have to be at least 1. Thus, no element is a majority in $A[1..i-1]$. In particular, we have

$$matches(x, i-1) \leq \frac{i-1}{2}. \tag{2}$$

If $A[i] \neq x$, then $matches(x, i) = matches(x, i-1) \leq \frac{i-1}{2} < \frac{i}{2}$, contradicting (1). So $A[i]$ must be x. Since $count_{i-1} = 0$, the ith iteration executes line 5, so $cand_i = A[i] = x$. Then, the iteration executes line 7, so $count_i = 1$. So, by (1), $matches(cand_i, i) > \frac{i}{2}$.

It remains to show that $count_i$ is correct.

$$\begin{aligned} matches(cand_i, i) &= matches(cand_i, i-1) + 1 && \text{since } A[i] = cand \\ &\leq \frac{i-1}{2} + 1 && \text{by (2)} \\ &= \frac{i+1}{2} \end{aligned}$$

Since $matches(cand_i, i) > \frac{i}{2}$, we must have $matches(cand_i, i) = \frac{i+1}{2}$. So,

$$\begin{aligned} matches(cand_i, i) - nonmatches(cand_i, i) &= matches(cand_i, i) - (i - matches(cand_i, i)) \\ &= \frac{i+1}{2} - (i - \frac{i+1}{2}) \\ &= 1 \\ &= count_i. \end{aligned}$$

Case 2: $count_{i-1} \geq 1$. This means that $cand_{i-1}$ is a majority in $A[1..i-1]$ and $count_{i-1} \geq 1$ is its margin of victory in $A[1..i-1]$. From the latter statement, we have:

$$\begin{aligned} matches(cand_{i-1}, i-1) - nonmatches(cand_{i-1}, i-1) &\geq 1 \\ matches(cand_{i-1}, i-1) - (i - 1 - matches(cand_{i-1}, i-1)) &\geq 1 \\ 2 \cdot matches(cand_{i-1}, i-1) &\geq i \\ matches(cand_{i-1}, i-1) &\geq \frac{i}{2} \end{aligned}$$

Thus, no value other than $cand_{i-1}$ can be a majority in $A[1..i]$, so $x = cand_{i-1}$.

In this case, the test on line 5 fails, so $cand_i = cand_{i-1} = x$ and $matches(cand_i, i) > \frac{i}{2}$ by (1). We just have to check that line 7 or 8 updates the margin appropriately.

If $A[i] = x$, then

$$count_i = count_{i-1} + 1 = matches(x, i-1) - nonmatches(x, i-1) + 1 = matches(x, i) - nonmatches(x, i).$$

If $A[i] \neq x$, then

$$count_i = count_{i-1} - 1 = matches(x, i-1) - nonmatches(x, i-1) - 1 = matches(x, i) - nonmatches(x, i).$$

This completes the proof that the first loop's invariant holds at every iteration. Thus, when the loop exits, *cand* will be the majority element in $A[1..n]$ if a majority exists. It is straightforward to see that the second loop just checks whether *cand* is indeed a majority in $A[1..n]$.