CSC B36 Additional Notes

majority element

⋆ Introduction

Here is an example of an iterative program for which the loop invariant is not trivial. Indeed it is difficult to see how the program works at all without a proof of correctness. So to all those who say

I will never need to prove correctness for any code

what if you were the inventor of this funky algorithm?! How would you convince anyone, even yourself, that it works?

◦ Definition of a majority element

Given a list A and an element x, we say that x is a *majority element* of A iff

x occurs in A more than $\text{len}(A)/2$ times.

E.g., 3 is a majority element of $[1, 2, 3, 3, 4, 3, 3]$, but not of $[2, 3, 4, 3]$.
Note: A list cannot have more than one majority element. It can possibly have none.

◦ Our problem specification

Suppose we want a correct program with respect to the following specification.

Precondition:	A is a list (possibly empty).
Postcondition:	Return the majority element of A if it exists. Return NONE if A has no majority element.

⋆ Naive algorithm

Here is a simple program to solve our problem. We use **for** loops here because we are mainly interested in partial correctness, not termination.

```
    NaiveMajority(A)
1       e = None
2       for i in range(len(A)):
3           count = 0:
4           for j in range(len(A)):
5               if A[j] == A[i]:
6                   count = count + 1
7           if count > len(A)/2:
8               e = A[i]
9       return e
```

Exercise: Prove that NAIVEMAJORITY is correct with respect to our specification. Each loop needs its own loop invariant, but they should not be hard to find.

The trouble with NAIVEMAJORITY is that it runs in quadratic time. I.e., its worst case running time is $O(n^2)$, where $n = \text{len}(A)$. Even if we make certain improvement, such as making the i-loop go from 0 to $\text{len}(A)/2$, it would still be $O(n^2)$.

⋆ Linear time algorithm

Here is a program that solves our problem in $O(n)$ time, where $n = \text{len}(A)$. As before, we use **for** loops.

```
    CleverMajority(A)
    ▷ For more information, see Boyer-Moore majority vote algorithm.
1       m = 0;    e = None
2       for i in range(len(A)):
3           if m == 0:
4               m = 1;    e = A[i]
5           elif A[i] == e:
6                   m = m + 1
7           else:
8               m = m − 1
9       count = 0
10      for j in range(len(A)):
11          if A[j] == e:
12              count = count + 1
13      if count > len(A)/2:
14          return e
15      else:
16          return None
```

◦ How to prove CleverMajority correct?

It is clear that lines 9-16 count the number of occurrences of e in A, then return e if it occurs more than $\text{len}(A)/2$ times, and otherwise return None.

Exercise: Give an appropriate precondition/postcondition pair for lines 9-16, then prove those lines correct with respect to your specification.

Question: What do lines 1-8 do?

Answer: In order for CleverMajority to be correct, lines 1-8 should have the following specification.

> **Precondition:** A is a list.
> **Postcondition:** If A has a majority element, then it is e.
>
> Then checking whether e is indeed the majority element, as done in lines 9-16, will lead to the correct returned value.

CleverMajority is indeed correct with respect to our specification!

Question: How do we prove it?

Answer: Use a loop invariant, of course!

Here is one appropriate LI for lines 1-8. It has four parts.

(a) $0 \le i \le \text{len}(A)$.

(b) $m \ge 0$.

(c) If $m = 0$, then $A[0 : i]$ has no majority element.

(d) If $m > 0$, then $A[0 : i]$ contains at least m copies of e and $A[0 : i]$ with m copies of e removed has no majority element.

Exercise: Complete the proof of correctness for CleverMajority.