CSC236 Winter 2020
Assignment #2: recurrences & correctness
due March 12th, 3 p.m.

Assignments are to be completed individually and typeset in LaTeX. The .tex source file and rendered pdf should both be uploaded to MarkUs. For further details, see the course website: http://www.cs.toronto.edu/~colin/236/W20/assignments/.

1. In lecture, we used the following recurrence to represent the steps taken by an implementation of mergesort on a list of size n:

$$T_0(n) = \begin{cases} 1 & \text{if } n = 1 \\ n + 2T_0(n/2) & \text{if } n > 1 \end{cases}$$

(This recurrence assumes n is a power of 2, hence the absence of floor and ceiling. You may maintain this assumption throughout this question.)

In reality, some implementations of divide-and-conquer algorithms stop the recursion before the input size becomes trivial. For example, a programmer may find that their mergesort implementation ends up running a bit faster if they stop recursing when the list size is less than 10, sorting these small lists using selection sort.

Consider the following recurrence, which models this scenario:

$$T(n) = \begin{cases} c & \text{if } n \leq k \\ n + 2T(n/2) & \text{if } n > k \end{cases}$$

$k, c \in \mathbb{N}^+$ are fixed constants, where k represents the largest problem size which is solved non-recursively, and c represents the cost of solving these small problems.

(a) Use unwinding[1] to find a closed form for $T(n)$ when $n \geq k$. (You do

[1] **Logistical note**: If you wish to use tree diagrams for the unwinding portions of this question (parts (a) and (c)), you are welcome to include scanned hand-drawn images, or diagrams generated using other software. See this chapter of the LaTeX Wikibook for information on including images in LaTeXdocuments. You may also describe the solution tree without explicitly drawing it (a table may be helpful).

not need to prove that your closed form is correct, but it should be clear how you arrived at it.)

(b) What is the big-Θ complexity of $T(n)$? Does it depend on k? Briefly justify your answer (no proof required). You may not assume $n \geq k$ for this part. Do not use the master theorem.

(c) Rather than assigning a fixed cost to the $n \leq k$ case, it may be more realistic to use a function of n, since there are a range of input sizes which are handled non-recursively. Since selection sort is a $\Theta(n^2)$ algorithm, we'll define our new recurrence $T'(n)$ to be

$$T'(n) = \begin{cases} n^2 & \text{if } n \leq k \\ n + 2T'(n/2) & \text{if } n > k \end{cases}$$

Find a closed form for $T'(n)$ for $n \geq k$, and show how you got there. Rather than unwinding from scratch, you may find it simpler to build on your work from part (a).

(d) Is $T'(n) \in \Theta(T(n))$? Why or why not? Briefly justify your answer. As in part (b), you may not assume $n \geq k$. Do not use the master theorem.

2. Our boss has tasked us with writing a program to find the *unique* maximum of a non-empty list of positive integers. If there is no unique maximum, our program should signal this by returning a negative number. For example, on input `[5, 2, 1, 2]`, our algorithm should return 5. Given `[2, 1, 2]`, we may return -1, -2, -236, or any other negative number.

   Below is our first attempt to solve this problem.[2]

```python
def umax(A):
    if len(A) == 1:
        return A[0]
    head = A[0]
    tail = A[1:]
    tmax = umax(tail)
    if head == tmax:
        return -1
    elif head > tmax:
        return head
    else:
        return tmax
```

(a) Based on the informal specification above, write precise pre- and post-conditions for umax. Your postcondition should use symbolic

[2] You can download the code for this question from http://www.cs.toronto.edu/~colin/236/W20/assignments/umax.py

notation rather than restating the English description above ("find the unique maximum..."). The following postcondition was used in lecture for the function `max`, which found the (not necessarily unique) maximum of a list. It may be a useful starting point:

`max(A) = x` where $(\exists j \in \mathbb{N}, A[j] = x) \wedge (\forall i \in \mathbb{N}, i < \text{len}(A) \implies A[i] \leq x)$

You may find it helpful to formally define 'helper' functions or predicates, as is done in question 3.

(b) The given Python code above has a bug. Demonstrate the bug by finding a value of A which meets the precondition, where `umax` misbehaves. For the value of A that you find, you should state the expected behaviour (according to your postcondition) and how it differs from the function's actual behaviour on that input.

(c) Consider our second draft of the function `umax` below:

```
def umax(A):
    if len(A) == 1:
        return A[0]
    head = A[0]
    tail = A[1:]
    tmax = umax(tail)
    if head == tmax:
        return -1 * head
    elif head > abs(tmax):
        return head
    else:
        return tmax
```

Prove that this function is correct with respect to the specifications you devised in part (a).

3. The function `maj` takes as input a list of natural numbers and, if the list has a majority element (one that appears more often than all other elements combined), it returns that element.[3] For example `maj([1, 3, 3, 2, 3])` returns 3.[4]

```
def R(A):
    B = []
    i = 0
    while i < len(A):
        a = A[i]
        b = A[(i+1) % len(A)]
```

[3]Note that the function's behaviour is undefined if the input list does not have a majority element. i.e. the presence of a majority element is a precondition of `maj`.

[4]You can download the code for this question from http://www.cs.toronto.edu/~colin/236/W20/assignments/maj.py

```
        if a == b:
            B.append(a)
        i += 1
    return B

def maj(A):
    prev = A
    curr = R(A)
    while len(curr) != len(prev):
        prev = curr
        curr = R(curr)
    return curr[0]
```

Prove that maj is correct.

To aid you on your quest, the appendix below contains some definitions that you may find useful in your proof, as well as a proof of Theorem 1.1, which, roughly speaking, says that every element x in $\mathrm{R}(A)$ corresponds to an $[x, x]$ pair in A (with the twist that pairs can "wrap around" from the end of the list to the front).

The A2 starter `.tex` source also includes the same definitions and a restatement of Theorem 1.1, which you may use in your proof. For brevity, the proofs of 1.1 and the associated helper lemmas are omitted. (You're unlikely to need to reuse these lemmas - they're provided here only as optional reading.)

You can download the `.tex` source for the appendix, if you'd like to emulate the commands used to generate numbered theorems/lemmas, and refer back to them.

Warning: This proof is both conceptually challenging and long (and the allocation of marks will reflect this). Count on it taking significantly longer than the other two questions. I recommend initially focusing on breaking the problem down into smaller pieces. We will allocate a significant number of marks for identifying an appropriate set of facts (invariants, postconditions, etc.) which link together to form a proof of the correctness of `maj`. If you can't prove a particular fact in the chain, simply state it, and assume it in the remainder of your proof.[5]

[5] The number of marks you will earn in this scenario will depend on the magnitude of the assumption. For example, if you simply assume the partial correctness of `maj`, your proof will earn few marks. On the other hand, if you have a nearly complete proof of the partial correctness of `maj` which uses one very specific unproven assumption, you may earn most of the marks.

1 Appendix: Q3 starting point

1.1 Definitions

We will begin by introducing some formally defined functions, predicates, and other notation (along with informal English descriptions) which capture some useful concepts. (You are welcome to introduce additional such definitions in your own proof.)

We will use $\mathbb{N}^*$ to denote the set of lists of natural numbers. Each of the functions below takes as its first argument a list $A \in \mathbb{N}^*$.

$\mathrm{COUNT}(A, x)$: $|\{i \in \mathbb{N} \mid A[i] = x\}|$

> *i.e. the number of occurrences of x in list A*

$$\mathrm{SUCC}(A, i) : \begin{cases} 0 & \text{if } i = \mathrm{len}(A) - 1 \\ i + 1 & \text{if } i < \mathrm{len}(A) - 1 \end{cases}$$

> *i.e. the 'next' index in list A after index i, treating A as a circular list (with the last index 'wrapping around' back to index 0). Not defined if $i \geq \mathrm{len}(A)$.*

$\mathrm{PAIRS}(A, x)$: $|\{i \in \mathbb{N} \mid A[i] = A[\mathrm{SUCC}(A, i)] = x\}|$

> *i.e. the number of $[x, x]$ pairs in list A, again treating A as circular*

$\mathrm{ADVANTAGE}(A, x)$: $\mathrm{COUNT}(A, x) - len(A)/2$

$\mathrm{MAJORITY}(A, x)$: $\mathrm{ADVANTAGE}(A, x) > 0$

> *i.e. x is the majority element of A*

1.2 R counts are pair counts

This section proves a theorem about a postcondition of the function R. You may use this theorem in your proof. For the purposes of answering question 3, **you do not need to read or understand the proof of Theorem 1.1 that follows**. However, you may find it useful as an example to model your proof on, in terms of content, organization (notice how we take a major result and break it down into several digestible sub-proofs), and level of rigour.

Theorem 1.1 (R "correctness"). *Given any $A \in \mathbb{N}^*$, $\mathrm{R}(A)$ terminates and returns a list of natural numbers B such that $\forall x \in \mathbb{N}, \mathrm{COUNT}(B, x) = \mathrm{PAIRS}(A, x)$*

We will prove this theorem in pieces, first proving a loop invariant for R, then proving that R terminates, and finally proving that termination implies the postcondition above.

For our loop invariant, we introduce the following function, PPAIRS, which counts the pairs in a prefix of a list:

PPAIRS(A, x, k): $|\{i \in \mathbb{N} \mid i < k \wedge A[i] = A[\text{SUCC}(A, i)] = x\}|$

> *i.e. the number of x pairs in a length k prefix of A. (Note that we use A's successor function, rather than one specific to the prefix $A[: k]$. This means we are allowed to 'peek' ahead to the $k+1$th element of A, if it exists, rather than wrapping around to the beginning of the prefix.*

Lemma 1.2. $\text{PAIRS}(A, x) = \text{PPAIRS}(A, x, \text{len}(A))$.

Proof. When $k = \text{len}(A)$, the $i < k$ condition in the definition of PPAIRS does not exclude any valid indices of A, so the definitions of PAIRS and PPAIRS become equivalent. □

Lemma 1.3 (R loop invariant). *The following are all true at the end of the jth iteration of R's while loop (if it occurs), when A is a list of natural numbers:*

(a) $i_j \leq \text{len}(A)$

(b) $\forall x \in \mathbb{N}, \text{COUNT}(B_j, x) = \text{PPAIRS}(A, x, i_j)$

(c) B_j contains only natural numbers

Proof. Base Case: Before the first iteration, $i = 0$, so (a) holds. $B_0 = [\,]$, so (c) trivially holds. (b) holds because, by definition, $\text{COUNT}([\,], x)$ and $\text{PPAIRS}(A, x, 0)$ are 0 for any x and A.

Inductive Step: Assume that the invariant holds at the end of the jth iteration, and assume that the code runs for a $j+1$th iteration.

By the while loop condition (line 4), $i_j < \text{len}(A)$, so it follows that $i_j + 1 \leq \text{len}(A)$ ((a)).

Either $B_{j+1} = B_j$, or line 8 is reached and B_{j+1} differs from B_j by the addition of element $a_{j+1} = A[i_j] \in \mathbb{N}$. In either case, (c) is satisfied.

It remains to show (b). Let $x \in \mathbb{N}$, and consider

$$\begin{aligned}
\Delta_c &= \text{COUNT}(B_{j+1}, x) - \text{COUNT}(B_j, x) \\
\Delta_p &= \text{PPAIRS}(A, x, i_{j+1}) - \text{PPAIRS}(A, x, i_j) \\
&= \text{PPAIRS}(A, x, i_j + 1) - \text{PPAIRS}(A, x, i_j)
\end{aligned}$$

By the code (lines 7-8), we can see Δ_c is 1 if $a_{j+1} = b_{j+1} = x$, and 0 otherwise.

By the definition of PPAIRS, we can see that Δ_p is 1 if $A[i_j] = A[\text{SUCC}(A, i_j)] = x$, and 0 otherwise. By line 5, $a_{j+1} = A[i_j]$. By line 6 and the definition of SUCC, $b_{j+1} = A[\text{SUCC}(A, i_j)]$. Therefore $\Delta_c = \Delta_p$. Since (b) holds at the beginning of the iteration, and for arbitrary x, the change in the LHS and the RHS (relative to the values at the beginning of the iteration) is identical, it follows that (b) holds at the end of the iteration. □

Lemma 1.4 (R termination). *R terminates on any $A \in \mathbb{N}^*$*

Proof. Let $q_j = \text{len}(A) - i_j$ be a quantity associated with each loop iteration j. By Lemma 1.3 (a), $q_j \in \mathbb{N}$. By line 9, $q_{j+1} = q_j - 1$. Thus $q_0, q_1, q_2, \ldots$ is a decreasing sequence of natural numbers, and therefore finite. Therefore, R terminates. □

We are now ready to prove the postcondition of R given in Theorem 1.1.

Proof of 1.1. Let A be an arbitrary list. By Lemma 1.4 the loop on line 4 terminates at the end of some iteration, call it j. By the loop condition, $i_j \geq \text{len}(A)$. By Lemma 1.3 (a) $i_j \leq \text{len}(A)$. Therefore $i_j = \text{len}(A)$.

By line 10, we return B_j. By 1.3, we have that $\forall x \in \mathbb{N}$

$$\begin{aligned} \text{COUNT}(B_j, x) &= \text{PPAIRS}(A, x, i_j) \\ &= \text{PPAIRS}(A, x, \text{len}(A)) \\ &= \text{PAIRS}(A, x) \qquad \# \text{ by Lemma 1.2} \end{aligned}$$

Also, by 1.3, we know that B_j contains only natural numbers. □