Engi 6892 Algorithms: Complexity and Correctness

T. S. Norvell (c) 2016

2016 Dec 8

Instructions: Answer all questions. If possible, write your answers in the space provided. Write on the back side of a sheet or request a yellow booklet if more space is required. *Answers in yellow booklets will **not** be marked **unless** there is a clear indication in the space provided on this exam paper that the answer is in the yellow booklet **and** your name is on the yellow booklet. Answers on the back sides of this booklet will **not** be marked **unless** there is a clear indication in the space provided on this exam paper of where the answer is located..* This is a **closed book** exam. Textbooks, notes, and electronic devices are not permitted; no, you can't use your cell phone as a clock. If your watch contains a computer, put it away. However, paper inter-language dictionaries are permitted, provided they contain no hand written notes within. *Set your cell phone to **silent** and put it away.*

Total points: 88 + bonus of 5

Name:

Student #:

Folders/Documents/courses/alg/2016/exam/Grace-Hopper-and-UNIVAC.jpg

Grace Hopper. Computer pioneer. Language designer. First compiler writer. Born Dec 9, 1906.

Student #: ____________________

Q0 [15] Suppose we can quickly compute the Frobnitz function $fr : \mathbb{N} \to \mathbb{N}$. As you may know $fr(0) = 0$ and $fr(2n+1) > n$, for all $n \in \mathbb{N}$. You need to design an algorithm that, for any natural number n, computes a natural number x such that $fr(x) \le n$ and $n < fr(x+1)$. You may call procedure fr to compute the Frobnitz function; your algorithm should, however, not call this procedure more than $O(\log n)$ times.

proc invFr($n : \mathbb{N}$) : $\mathbb{N}$
precondition: true
postcondition: $fr(result) \le n$ and $n < fr(result+1)$

(a)[5] What is the invariant?

(b)[3] What is the loop guard?

(c)[2] What is the variant?

(d)[5] Write a proof outline for this problem so that it takes $\Theta(\log n)$ time, presuming procedure fr takes $\Theta(1)$ (constant) time.

Student #: ____________________

Q1 **[15]** For each of the following proof outlines, either explain why the outline is correct or suggest a way to make it correct.

(a) **[5]**

$\{x > 5 \wedge y > 5\}\ x := y + 1\ \{x > 6\}$

(b) **[5]**

```
$\{a.\text{length} = n\}$
$i := 0$
$\{i = 0\}$
while $i < n \wedge a(i) \geq 0$ do
    $i := i + 1$
end while
$\{(i = n \vee i \in \{0, ..n\} \wedge a(i) < 0) \wedge \forall k \in \{0, ..i\} \cdot a(k) \geq 0\}$
```

(c) **[5]**

```
$\{A, B \in \mathbb{N}\}$
$a := A$
$b := B$
$\{b \geq 0 \wedge z \times a^b = A^B\}$
while $b > 0$ do
    $z := z \times a$
    $b := b - 1$
end while
$\{z = A^B\}$
```

Student #: ____________________

Q2 **[10]** Using the linking invariant

$$L : \forall i \in \{2, ..n\} \cdot ((i \in S) = a(i))$$

where $n = a$.length and a is a boolean array, data refine the following code to eliminate all need for variable S.

```
var $S := \{2, ..n\}$
var $k := 2$
while $k < n$ do
    if $k \in S$ then
        $m := k + k$
        while $m < n$ do
            $S := S - \{m\}$
            $m := m + k$
        end while
    end if
    $k := k + 1$
end while
```

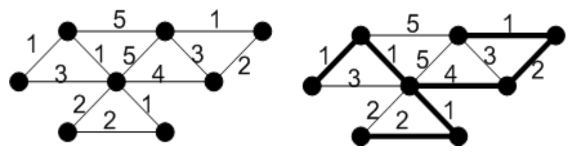
Q3 [15] A simple, undirected graph without loops[0] $G = (V, E)$ is said to have a cycle if there are two nodes that have two different simple paths[1] between them. Suppose that procedure hasCycle(V, E) indicates whether a graph has a cycle or not.

Consider the following problem.

Input: a simple, undirected graph without loops $G = (V, E)$ and a function w that maps edges to real numbers.

Output: a set of edges $F \subseteq E$ such that: (a) the graph (V, F) has no cycles, (b) adding any more edges would create a cycle, (c), out of all subsets of E satisfying (a) and (b), F minimizes the function $f(D) = \sum_{e \in D} w(e)$.

Design a greedy algorithm to solve this problem. There is no need to prove your algorithm. But please show the steps that you take to arrive at the algorithm. State the complexity of the algorithm.

Folders/Documents/courses/alg/2016/exam/MST.png 

On the left is a graph. On the right, the set of bolded edges satisfies (a), (b), and (c).

[0]A *simple* graph has at most one edge between any pair of nodes. A *loop* is an edge that connects a node to itself.
[1]A *simple* path has no repeated nodes, except that the first and last nodes may be the same.

Student #: ____________________

Q4 [6] In the LISP language, an S-expression is one of three things: an atom, an empty list, or a non-empty list. An atom is written just as a terminal from a finite set $X = \{x_0, x_1, \ldots, x_{n-1}\}$; the set X does not include either '(' or ')'. An empty list is written as the terminals '(' and ')' in that order. A nonempty list is written as a sequence of 1 or more S-expressions preceded by a '(' and followed by a ')'. Design a context-free grammar over the set $\{`(', `)'\} \cup X$ that describes S-expressions by answering the following questions

- What is the set of terminals?
- What is the set of nonterminals?
- What is the start nonterminal?
- What are the productions?

Q5 [12]

For all parts below *clearly explain your answer.*

(a)[3] Suppose the Dafny verifier reports that a postcondition of a method may be violated. Does this mean:

0. That, on every execution of the method for which the preconditions are true, the postcondition will be false.
1. That, there is at least one execution in which the preconditions are true and the postcondition is false.
2. Neither of the above.

(b)[3] Suppose the Dafny verifier does not report that a postcondition of a method may be violated. Does this mean:

0. That, on every execution of the method for which the precondition is true, the postcondition will be true.
1. That, there is at least one execution in which the preconditions are all true and the postcondition is true.
2. Neither of the above.

(c)[3] True or false: In Dafny it is necessary for the programmer to provide a sufficiently strong loop invariant for a loop to be shown correct.

(d)[3] Will the following Dafny code verify

```
method between( a : int, c : int ) returns (b : int)
requires a + 1 < c
ensures a < b < c {
    b := a + (c-a)/2 ; }
method check() {
    var y : int := between( 10, 20 ) ;
    assert y == 15 ; }
```

Q6 [15] Joe's Discount Programming House has 2 software engineering teams. Joe has a set of jobs presented as a list j of length m. Each job $j(i)$ takes time $j(i)$.time, a positive integer. Given a sequence of m jobs $[j(0), j(1), .., j(m-1)]$, we need to find a way of dividing up the jobs between the 2 teams so that each team does exactly the same amount of work. The jobs can be done in any order. Suppose T is the sum of the times over all the jobs. We'll assume T is even.

(a) [5] Design a (possibly very inefficient) recursive algorithm to determine whether there is a subset of the jobs that can be done in time $T/2$. Be sure to state the pre- and postcondition of all procedures.

(b) [5] Design a top-down dynamic programming algorithm for the problem. What is the time complexity of the algorithm?

(c) [5] Design a bottom-up dynamic programming algorithm for solving this problem. What is the time complexity of the algorithm?

Student #: ____________________

Bonus [**5**] Either extend or modify your algorithm from Q6(b) or Q6(c) to efficiently compute a way (assuming there is one) to assign jobs to team 0 and team 1 so that team 0 does work that adds up to $T/2$ and team 1 also does work that sums to $T/2$.