Massachusetts Institute of Technology
6.042J/18.062J: Mathematics for Computer Science
Professors David Karger and Nancy Lynch

Handout **??**
March 8 2000

Practice Quiz 1 Solutions

Problem 1 **(6 points)** Consider the following proposition.

$$\forall x \forall y \ (x < y \rightarrow \exists z \ (x < z \wedge z < y))$$

(a) **(2 points)** Prove or disprove this proposition, assuming that the universe is the set of integers.

The proposition is false. In the case where $x = 0$ and $y = 1$, there does not exist an integer z such that $x < z$ and $z < y$.

(b) **(4 points)** Write the negation of this proposition. Your solution may *not* use the $\neg$ symbol, but may use any of the relations $=$, $<$, $>$, $\leq$, or $\geq$.

$$\exists x \exists y \ (x < y \wedge \forall z \ (x \geq z \vee z \geq y))$$

Problem 2 **(10 points)** Define the matrix

$$M \ = \ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

Let M^n denote the product of n copies of the matrix M. Use ordinary induction to prove that for all $n \geq 1$, the upper right entry of M^n is F_n, the n-th Fibonacci number. State your inductive hypothesis clearly.

For reference, the standard formula for the product of 2×2 matrices is given below.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix} \ = \ \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix} 1g$$

Recall that $F_0 = 0$, $F_1 = 1$, $F_2 = 1$, $F_3 = 2$, $\ldots$.

Proof. The proof is by induction. Let $P(n)$ be the proposition that

$$M^n = \begin{bmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{bmatrix}.$$

In the base case, $P(1)$ is true because

$$\begin{aligned} M^1 &= \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} F_0 & F_1 \\ F_1 & F_2 \end{bmatrix}. \end{aligned}$$

■

In the inductive step, for $n \geq 1$ assume $P(n)$ to prove $P(n+1)$. The following equalities hold.

$$\begin{aligned} M^{n+1} &= M^n \cdot M \\ &= \begin{bmatrix} F_{n-1} & F_n \\ F_n & F_{n+1} \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} F_n & F_{n-1} + F_n \\ F_{n+1} & F_n + F_{n+1} \end{bmatrix} \\ &= \begin{bmatrix} F_n & F_{n+1} \\ F_{n+1} & F_{n+2} \end{bmatrix} \end{aligned}$$

The first step uses the definition of M^n. The second step uses the induction hypothesis and the definition of the matrix M. We carry out a matrix multiplication in the third step and use the Fibonacci recurrence in the last step. This shows that $P(n)$ implies $P(n+1)$, and the claim is proved by induction.

Problem 3 (12 points) The vertices of a directed graph can be partitioned into *strongly-connected components*. Two vertices u and v belong to the same strongly connected component if there is a path from u to v and a path from v to u. A strongly connected component may consist of a single vertex.

Math Moose has noticed a strange phenomenon. He starts with a directed graph and partitions the vertices into strongly-connected components. He then removes all edges that connect two vertices in the same strongly-connected component. The resulting graph always seems to be acyclic.

An example is worked out in the figure below. The original directed graph is shown on the left. The vertices a, b, and c form one strongly-connected component. The single vertex d

forms a second strongly-connected component. Math Moose removes the edges $a \to b$, $b \to c$, and $c \to a$, since these edges connect vertices in the same strongly-connected component. The resulting graph is shown on the right. Note that this graph is acyclic.

Use proof by contradiction to show that for every directed graph, application of Math Moose's procedure gives a directed acyclic graph.

Proof. Let $G = (V, E)$ be an arbitrary directed graph. Let $G' = (V, E')$ be the directed graph that remains after we remove all edges that connect two vertices in the same strongly-connected component.

The proof is by contradiction. Suppose that G' is not acyclic. Then let u and v be consecutive nodes in a directed cycle. There is a path from u to v consisting of a single edge. There also is a path from v to u, because the nodes are in a cycle. Since every edge in G' is also in G, these same two paths exist in the original graph G. Therefore, u and v are in the same strongly-connected component of G. But then the edge from u to v should have been removed during the construction of G'. Since the the edge was not removed, we have a contradiction. Therefore, G' is acyclic. ■

Problem 4 (12 points) One of the three monks working on the famed Towers of Hanoi project recently rubbed his pained back and burst out, "Yo! What are we doing? This is for chumps! Let's punt!" But before wandering off to start up fast food joints, they must evenly divide the monastery's collection of prayer beads.

Initially, monk A has 5 beads, monk B has 3 beads, and monk C has 4 beads. The monastic order has strict rules regarding the exchange of prayer beads. Only the following transactions are allowed.

1. Monk B may give a bead to monk A at any time.
2. If C has an odd number of beads, then monk A may give a bead to monk B.
3. If C has an even number of beads or monk A has at least two more beads then monk B, then monk C may give or take a bead from either monk A or monk B.

(a) (3 points) Model the situation with a state machine. Define the set of states,the set of start states, and the set of transitions.

The set of states Q consists of all triples (a, b, c) such that $a, b, c \geq 0$ and $a + b + c = 12$. The set of starts states Q_0 consists of the single triple $(5, 3, 4)$. The set δ contains six types of transition:

$$
\begin{array}{lcll}
(a,b,c) & \to & (a+1,b-1,c) & (b>0) \\
(a,b,c) & \to & (a-1,b+1,c) & (a>0 \text{ and } c \text{ is odd}) \\
(a,b,c) & \to & (a-1,b,c+1) & (a>0 \text{ and either } c \text{ is even or } a \geq b+2) \\
(a,b,c) & \to & (a,b-1,c+1) & (b>0 \text{ and either } c \text{ is even or } a \geq b+2) \\
(a,b,c) & \to & (a+1,b,c-1) & (c>0 \text{ and either } c \text{ is even or } a \geq b+2) \\
(a,b,c) & \to & (a,b+1,c-1) & (c>0 \text{ and either } c \text{ is even or } a \geq b+2)
\end{array}
$$

(b) (3 points) Prove that the monks can reach the state where monk A has 0 beads, monk B has 9 beads, and monk C has 3 beads by describing a sequence of steps leading to this state.

Initially, monk C gives a bead to monk B using rule 3. Then monk A gives his 5 beads to monk B using rule 2 repeatedly.

(c) (6 points) Prove that the monks can not reach the state where every monk has 4 beads by using the Invariant Theorem.

Proof. The proof uses the Invariant Theorem. We prove that if monk C has an even number of beads, then monk A has more beads than monk B. This implies that the monks can never evenly divide up the prayer beads; an even division would require that C have 4 beads, but then the invariant would imply that A has more beads then B.

The invariant holds in the single start state Q_0, since in this case monk C has an even number of beads (4), and monk A has more beads than monk B (5 vs. 3). Furthermore, if the invariant holds before a transition, then it holds after a transition. There are six cases to check, corresponding to the six types of transition.

1. $(a,b,c) \to (a+1,b-1,c)$ where $b>0$. If c is odd, then the invariant holds after the transition because the hypothesis of the invariant if false. If c is even, then the invariant holds after the transition because $a+1>a>b>b-1$.

2. $(a,b,c) \to (a-1,b+1,c)$ where $a>0$ and c is odd. The invariant holds after the transition because the hypothesis of the invariant is false.

3. $(a,b,c) \to (a-1,b,c+1)$ where $a>0$ and either c is even or $a \geq b+2$. If c is even, then the invariant holds after the transition because the hypothesis is false. If c is odd, then the invariant holds after the transition because the conclusion is true; that is, monk A has more beads than monk B.

The argument in the remaining three cases is identical to the argument for case 3. Since the invariant hold in all start states and is maintained under all transitions, the claim is proved by the Invariant Theorem. ■

Problem 5 Use induction to prove our proposition, P(n),

$$1^3 + 2^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

whenever n is a natural number.

Base Case: $n = 0$

The left side is an empty sum, so $0 = \left(\frac{0 \cdot 1}{2}\right)^2 = 0$.

Inductive Step: Show that $P(n) \rightarrow P(n+1)$.

By the inductive hypothesis, we assume that

$$1^3 + 2^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$$\begin{aligned} 1^3 + 2^3 + \ldots + n^3 + (n+1)^3 & \\ &= \left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 \\ &= \frac{n^2(n+1)^2 + 4(n+1)(n+1)^2}{4} \\ &= \frac{(n^2+4n+4)(n+1)^2}{4} \\ &= \left(\frac{(n+1)(n+2)}{2}\right)^2 \end{aligned}$$

And our hypothesis is proved.

Common errors

The most common error was to assume that the hypothesis was true, and then to prove it by reducing both sides of the equation to something that is obviously equal. This is not a good proof technique.

$$\begin{aligned} 1^3 + 2^3 + \ldots + n^3 + (n+1)^3 &= \left(\frac{(n+1)(n+2)}{2}\right)^2 \\ \left(\frac{n(n+1)}{2}\right)^2 + (n+1)^3 &= \left(\frac{(n+1)(n+2)}{2}\right)^2 \\ \frac{n^2(n^2+2n+1)}{4} + n^3 + 3n^2 + 3n + 1 &= \frac{(n^2+3n+2)^2}{4} \\ \frac{(n^4+2n^3+n^2) + (4n^3+12n^2+12n+4)}{4} &= \frac{n^4+3n^3+2n^2+3n^3+9n^2+6n+2n^2+6n+4}{4} \\ \frac{n^4+6n^3+13n^2+12n+4}{4} &= \frac{n^4+6n^3+13n^2+12n+4}{4} \end{aligned}$$

A technique which is similar but which is not at all correct is to start with the equation you wanted to prove (ie, $f(n) = g(n)$) and then perform the same operations on both sides of the equation until they're equal.

$$\begin{aligned} n^2 + 3n + 1 &= n^2 + 3n + 1 \\ 3n + 1 &= 3n + 1 \\ 1 &= 1 \end{aligned}$$

This is OK for a quick check, but does not quite work as a proof, since it is possible to get equality at the end even if the starting sides of the equation were not equal. For example

$$\begin{aligned} n^2 + 3n + 1 &= 2n^2 + 5n + 8 \\ 0 \cdot (n^2 + 3n + 1) &= 0 \cdot (2n^2 + 5n + 8) \\ 0 &= 0 \end{aligned}$$

Remember that in an ideal direct proof, each step should follow from the previous one, until the final step which is what we want to prove.

Problem 6 Prove the set identity

$((A - B) \cup (B - A))^c = (A^c \cup B) \cap (A \cup B^c)$

Using the definition of set difference, both of De Morgan's Laws, and the commutativity of union (in that order):

$$\begin{aligned} ((A - B) \cup (B - A))^c &= ((A \cap B^c) \cup (B \cap A^c))^c \\ &= (A \cap B^c)^c \cap (B \cap A^c)^c \\ &= (A^c \cup B) \cap (B^c \cup A) \\ &= (A^c \cup B) \cap (A \cup B^c) \end{aligned}$$

Alternatively, one can use the definitions of the operations. We need to prove x is in the LHS if and only if it is in the RHS.

$x \in ((A - B) \cup (B - A))^c$ means that $x \notin (A - B) \cup (B - A)$. This means that $x \notin A - B$ and $x \notin B - A$.

So consider cases. If $x \in A$, then certainly $x \in A \cup B^c$. Furthermore, since by assumption $x \notin A - B$, we must have $x \in B$. Thus, $x \in B \cup A^c$. Putting these together shows $x \in (A \cup B^c) \cap (B \cup^c)$. Now consider the other case. If $x \notin A$, meaning $x \in A^c$, then certainly $x \in A^c \cup B$. Also, (since $x \notin B - A$), we must have $x \notin B$. That is, $x \in B^c$. Therefore, $x \in B^c \cup A$. Thus, $x \in (A^c \cup B) \cap (B^c \cup A)$.

This proves one direction; the other is similar.

Problem 7 A *full binary tree* is either

- A *leaf* node with no subtrees
- A *non-leaf* node with a left and a right subtree, each of which is a full binary tree.

(a) Draw all full binary trees with 5 or fewer nodes.

(b) Prove that for any full binary tree, the number of non-leaf nodes is exactly one less than the number of leaves.

Solution 1: Let,

$$P(n) = \text{“For a full binary tree with n non-leaf nodes } n = l - 1 \text{ where } l \text{ is the number of leaves”}$$

Proof by strong induction on the number of non-leaves, n:

Base Case: $n = 0$, tree is made up of exactly one node. $P(0)$ is true since a node with no subtrees is a leaf ($l = 1$) and thus $0 = 1 - 1$.

Inductive Hypothesis: For $n \leq k$ non-leaf nodes assume $P(n)$ is true, that is, “For a full binary tree with $n \leq k$, the non-leaf nodes $n = l - 1$”.

Inductive Step: Prove P(n=k+1): For a full binary tree with $k + 1$ non-leaf nodes, the number of non-leaf nodes in the tree by definition is made up of:

- one root node,
- the non-leaf nodes in the left-subtree of the root, n_{left}, and
- the non-leaf nodes in the right-subtree of the root, n_{right}.

The number of leaves, l_{tree} in the tree by definition is made up of:

- the number of leaves in the left-subtree of the root, l_{left}, and
- the number of leaves in the right-subtree of the root, l_{right}.

$$\begin{aligned}
k+1 &= 1 + n_{left} + n_{right} && \text{by the definition of full binary tree} \\
&= 1 + (l_{left} - 1) + (l_{right} - 1) && \text{by the inductive hypothesis} \\
&= 1 + (l_{left} + l_{right}) - 1 - 1 && \\
&= l_{tree} - 1 && \text{by definition of full binary tree}
\end{aligned}$$

Solution 2:

We use induction on the given recursive definition.

There is only one base case, which is a leaf node with no subtrees. So there is one leaf node and no non-leaf nodes, satisfying the statement.

There is only one combination case, which is a non-leaf node (call it v) with a left and a right subtree, each of which is a full binary tree. By the IH, each of these subtrees has exactly one fewer non-leaf nodes than leaf nodes. Taken together, there are then two fewer non-leaf nodes than leaf nodes. Since this accounts for all of the nodes in the tree except for v (which is a non-leaf node), the tree has exactly one fewer non-leaf nodes than leaf nodes.

Errors: The most common error in this problem was to use facts about full binary trees that were not at all obvious from the above definition. For example, some people said during the inductive step

- "find a node with two children that are leaves; take away the leaves; this gives a smaller full binary tree to which an inductive hypothesis applies."
- "find a leaf turn it into a non-leaf by adding two children; the new tree has $k+1$ non-leaf nodes, while the original tree has k non-leaf nodes to which the inductive hypothesis can be applied"

Unfortunately, it is not at all obvious from the above definition of a full binary tree that that taking away or adding two leaf nodes results in a full binary tree. While this may be true, it is something that would have to be proven.

Main point is, that when you do induction on a recursive definition, your inductive step needs to be based on the combination rules of the inductive definition, and not on other things that are "obvious" from drawing pictures.

Problem 8 Use the following tables to describe the properties of the various relations and functions. Write yes or no in each box. Don't give proofs (it'd be too hard to squeeze them into the boxes). **Pay careful attention** to the specified domains and ranges.

(a) The following relations:

- $R = \{(a,b) \in (\mathbb{N} - \{0\})^2 \mid a \text{ divides } b\}$
- $S = \{(X,Y) \in (2^{\mathbb{N}} - \{\emptyset\})^2 \mid X \cap Y \neq \emptyset\}$
- $T = \{(x,y) \in \mathbb{R} \times \mathbb{R} \mid x + y = 0\}$

(Recall that antisymmetric means that $a \sim b$ and $b \sim a$ imply $a = b$).

relation	reflexive	symmetric	antisymmetric	transitive	function
$R: \ a \mid b$					
$S: \ X \cap Y \neq \emptyset$					
$T: \ x + y = 0$					

relation	reflexive	symmetric	antisymmetric	transitive	function
$R: \ a \mid b$	yes	no	yes	yes	no
$S: \ X \cap Y \neq \emptyset$	yes	yes	no	no	no
$T: \ x + y = 0$	no	yes	no	no	yes

(b) The following functions:

- $f : \mathbb{N} \to \mathbb{N}, \ f(x) = 5x$.
- $g : \mathbb{R}^+ \to \mathbb{N}, \ g(x) = \lfloor x \rfloor$.
- $h : \mathbb{R} \to \mathbb{R}^+, \ h(x) = e^x$.

Give an inverse if one exists (under the given domain and range).

function	injective	surjective	inverse
$f(x) = 5x$			
$g(x) = \lfloor x \rfloor$			
$h(x) = e^x$			

function	injective	surjective	inverse
$f(x) = 5x$	yes	no	none
$g(x) = \lfloor x \rfloor$	no	yes	none
$h(x) = e^x$	yes	yes	$\ln(x)$

Errors: A common error here was not to pay attention to the domain and range. Although $f(x) = 5x$ is a bijection from the reals to the reals, it is not a bijection from the naturals to the naturals because there is no *natural* number that satisfies $5x = 1$.

Problem 9 Consider the following algorithm, which takes natural numbers a and b, with $b > 0$, and returns an ordered pair of natural numbers.

Algorithm Div(a, b)

```
if (a < b)
    return (0, a)
else
    (q, r) ←Div(a − b, b)
    return (q + 1, r)
```

(a) Prove Div terminates in at most $a + 1$ steps.

We use strong induction on a. The inductive hypothesis is "if Div(a', b) terminates for all $a' < a$, then Div(a, b) terminates".

Base Case: $a < b$. Then Div returns immediately (that is, in one step), which is certainly no more than $a + 1$ steps, since $a \geq 0$.

Inductive step: $a \geq b$. Then $0 \leq a - b < a$, since $b > 0$. In this case, there is a recursive call $\text{Div}(a - b, b)$. By the IH, this recursive call returns in at most $a - b + 1$ steps. Since $a - b < a$, this is at most $a + 1$ steps.

Common errors

Please note that many people received credit for a base case of $a = 0$, which is actually wrong. This case is only reached when a is a multiple of b, so the induction doesn't stand, since for cases where a does not divide b, no base case has been shown.

We insisted that you use induction for this because any sort of argument that goes something like "And then it keeps on subtracting X until..." is not rigorous. Such an argument is begging for induction.

(b) Prove that Div returns the quotient and remainder of a on division by b.

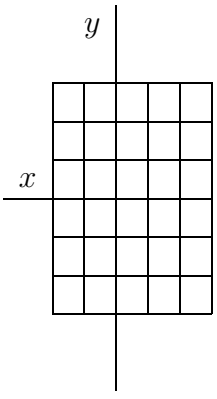
Recall that the quotient and remainder of a on division by b are defined to be the unique integers q and r such that $a = qb + r$ and $0 \leq r < b$. These integers must exist by the Division Algorithm.

We use strong induction on a.

Base case: $a < b$. Then Div immediately returns $(0, a)$. Since $a = 0 \cdot b + a$ and $0 \leq a < b$, these are the quotient and remainder of a on division by b.

Inductive step: $a \geq b$. Then $0 \leq a - b < a$. In this case, there is a recursive call $\text{Div}(a - b, b)$. By the IH, this recursive call returns the quotient and remainder of $a - b$ on division by b. So $a - b = qb + r$ and $0 \leq r < b$. Then $a = (q + 1)b + r$ and $0 \leq r < b$. So $q + 1$ and r are the quotient and remainder of a on division by b, and these are returned by Div.

Problem 10 David is located at position $(1, 0)$ on the two-dimensional $(\mathbb{Z} \times \mathbb{Z})$ x-y plane. His house is located at the origin. David wants to go home. But it is windy, and each time David takes a one-unit step east or west, he also gets blown a one-unit step north or south (he can pick which).



(a) On the grid above, mark with X's all the points that David can reach in one step. Mark with O's the points that David can reach in two steps. (The axes labeled x and y intersect at David's house.)

(b) Give an informal explanation for why David can never get home.

David starts off with the sum of his coordinates ($1 + 0 = 1$) odd. Every time he moves, the parity of each coordinate changes, so the sum of his coordinates remains odd. Since the sum of the coordinates of his home ($0 + 0 = 0$) is even, he can never reach it.

(c) Describe a state machine that represents David's motion, and list its transitions.

We introduce state variables $x, y \in \mathbb{Z}$. We designate exactly one start state: the one for which $x = 1$ and $y = 0$. The transitions are as follows:

move($go, blow$)
  Precondition: $go, blow \in \{-1, 1\}$
  Effect: $x \leftarrow x + go$
  $y \leftarrow y + blow$

Getting the notation for transitions wrong was a common error. In particular, some people wrote program code that returned values. The fragments we use for transitions reflect changes to state variables, which are best represented by assignment operations. The role of the Effect clause is to *describe* what the value of the state variables will be after the transition. We have adopted an assignment style of notation for the Effect clause to accentuate the fact that state variables are changed by the transition.

(d) Prove that David can never get home.

We use the following invariant: $x+y$ is odd for all reachable states. This is true for the start state ($0 + 1 = 1$). For each transition, go and $blow$ are each odd, so their sum is even. Thus each transition increases $x + y$ by an even amount ($go + blow$), preserving the invariant.

Since $x + y$ is even for the state corresponding to home, it is not a reachable state of our state machine. Thus David can never get home.

Errors: There were two main types of errors in coming up with invariants.

One was to provide a correct statement that was not amenable to an inductive proof. For example, the fact that $x \neq y$ is true of all reachable states, but is not preserved by all transitions (you can get from (2,0) to (1,1), for instance). This means that it is not really possible to prove it by induction because there are certain states (such as $(2, 0)$) where the inductive step doesn't work. Note that writing an inductive proof does not strictly require that the property be preserved for all transitions (only those starting from reachable states). As a practical matter, however, it is hard to make the inductive step take advantage of the fact that the first step was reachable. (This is because reachability is not a property of the state in isolation but depends on the *entire* state machine definition). Thus one has little

choice but to show in the inductive step that *any* state (reachable or not) that satisfies the invariant has outgoing transitions leading only to states that satisfy the invariant.

Another mistake was to prove an invariant that is true but does not imply what we want. Some proved that the total distance travelled was always a multiple of $\sqrt{2}$. While this invariant is indeed true, it does not prove what we want because *straight-line* distance from the starting point does not always have to be a multiple of $\sqrt{2}$, so the above fact does not amount to a proof.

Problem 11 Let F be a function from A to B and let $G = F^{-1}$ be its inverse relation (G is a relation from B to A, but not necessarily a function). Prove that $G \circ F$ is an equivalence relation on A.

By definition, x is related to y by $G \circ F$ iff there is a $z \in B$ such that x is related to z by F and z is related to y by G. That is, $z = F(x)$ and $z = F(y)$. So x is related to y by $G \circ F$ iff $F(x) = F(y)$.

Reflexitivity: Let $x \in A$. Then $F(x) = F(x)$, so $(x, x) \in G \circ F$.

Symmetry: Let $x, y \in A$ such that $(x, y) \in G \circ F$. Then $F(x) = F(y)$ (and $F(y) = F(x)$). So $(y, x) \in G \circ F$.

Transitivity: Let $x, y, z \in A$ such that $(x, y), (y, z) \in G \circ F$. Then $F(x) = F(y)$ and $F(y) = F(z)$. So $F(x) = F(z)$ and $(x, z) \in G \circ F$.

One common mistake for this problem was to assume that $G \circ F$ is the identity function. In general, G is not a function, and neither is $G \circ F$.

Problem 12 Consider the following algorithm that takes an array (list) $A[1 \ldots n]$ of numbers such that $A[1] < A[2] < \cdots < A[n]$, and a number q, and is supposed to find q if it is present in the list.

Find(A, n, q)

$i \leftarrow 1$
$j \leftarrow n + 1$
While $(j - i > 1)$
    choose some k with $i < k < j$
    if $A[k] \leq q$
        $i \leftarrow k$
    else
        $j \leftarrow k$
Return i

(a) prove this function terminates

We first prove (by induction on l) that after the lth pass through the while loop, $j - i \leq n - l$.

Base Case: $l = 0$. When we first reach the while loop, $j - i = n$ because we just set i to 1 and j to $n + 1$.

Inductive Step: By the IH, $j - i \leq n - l$ after the lth pass through the while loop. During the $(l + 1)$st pass, either i or j is moved to a value (that of k) strictly between their values from the end of the lth pass. Thus the $(l + 1)$st pass decreases $j - i$ by at least 1. So after the $(l + 1)$st pass through the while loop, $j - i < n - l - 1 = n - (l + 1)$.

The most common error on this part was hand-waving. Informal descriptions of the long-term behavior of the algorithm are not considered proofs by 6.042 standards. Since the reader can safely be assumed to accept induction as a valid proof technique, the above proof *at very least* gives the reader much less room to object to the argument.

The second most common error was to try to prove the desired statement directly by induction. This just doesn't work. If it did, we would not have introduced the more complicated proof technique that we said should be used in this case.

(b) Now suppose that $q = A[r]$ for some r. State an invariant of the while loop relating i, j, and r. The invariant should be "obviously provable by induction" but you need not prove it.

$i \leq r < j$.

(c) Assuming the invariant, prove that Find returns r.

When the loop terminates, $j - i \leq 1$ $(j \leq i + 1)$. Also, the invariant still holds. So $i \leq r < j \leq i + 1$. Since i and r are integers, $i = r$. Since i is then returned, Find returns r.

Surprisingly enough, a number of people didn't even try to make use of the invariant. Use of induction was also common. The invariant deals with exactly the issues that would otherwise require induction. This is why the correctness of the invariant would be proven by induction. Some people apparently didn't read the question and tried to prove the invariant correct instead of proving that Find returns r.