CS 70 Discrete Mathematics and Probability Theory
Spring 2015 Vazirani Midterm #1 Solution

PRINT your name: _____, _____
(last) (first)

SIGN your name: _____

PRINT your student ID: _____

CIRCLE your exam room: 306 Soda 141 McCone 1 Pimentel

Name of the person sitting to your left: _____

Name of the person sitting to your right: _____

Please write your name and student ID on every page.

Please write your answers in the spaces provided in the test. We will not grade anything on the back of an exam page or outside the space provided for a question unless we are clearly told on the front of the page in the space provided for the question to look there.

You may consult one single-sided sheet of handwritten notes. Apart from that, you may not look at books, notes, etc. Calculators and computers are not permitted.

You have 120 minutes. There are 6 questions worth 20 points each, for a total of 120. Use the number of points as a rough guide for the amount of time to allocate to that question. Note that the last question is harder than the rest, so you should attempt it only after you take a good shot at the rest. Note also that many of the points are for proofs and justifications for your answers. Please make sure you spend the time to write clear, correct and concise justifications. Good luck!

Do not turn this page until your instructor tells you to do so.

Q. 1	Q. 2	Q. 3	Q. 4	Q. 5	Q. 6	Total/120

Name: _____ SID: _____

1. **Propositions and Quantifiers**

   For each of these assertions say whether it is necessarily True or whether it could be False. Give a brief (couple of lines) justification for your answer, if true the main idea of the proof and if false then a counterexample. 4 points for each part — 1 for the correct answer and 3 for justification.

   (a) $P \Rightarrow (\neg P \Rightarrow Q)$

   True **False**

   Answer: True

   Justification: We can equivalently write the implication as

   $$P \Rightarrow (\neg P \Rightarrow Q) \equiv P \Rightarrow (P \vee Q) \equiv \neg P \vee (P \vee Q) \equiv (\neg P \vee P) \vee Q$$

   and $\neg P \vee P$ is always true.

   (b) $P \wedge Q$ and $\neg P \vee \neg Q$ cannot both be False.

   True **False**

   Answer: True

   Justification: Note that $\neg P \vee \neg Q \equiv \neg(P \wedge Q)$, and $(P \wedge Q)$ and $\neg(P \wedge Q)$ cannot both be False.

   (c) $(\forall x\, A(x) \wedge \forall x\, B(x)) \implies \forall x\, (A(x) \wedge B(x))$

   True **False**

   Answer: True

   Justification: Suppose the left hand side is true. This means for all x, $A(x)$ holds, and similarly, for all x, $B(x)$ holds. Therefore, for all x, $A(x) \wedge B(x)$ holds, showing the right hand side is also true.

   (d) $(\forall x\, A(x) \vee \forall x\, B(x)) \implies \forall x\, (A(x) \vee B(x))$

   True **False**

   Answer: True

   Justification: Assume towards contradiction that $(\forall x\, A(x) \vee \forall x\, B(x))$ is true but $\forall x\, (A(x) \vee B(x))$ is false. Then, $\exists x\, \neg(A(x) \vee B(x))$. However, this contradicts the assumption that $(\forall x\, A(x) \vee \forall x\, B(x))$, thus proving that whenever $(\forall x\, A(x) \vee \forall x\, B(x))$ is true, $\forall x\, (A(x) \vee B(x))$ must also be true.

   (e) $\forall x\, (A(x) \vee B(x)) \implies (\forall x\, A(x) \vee \forall x\, B(x))$

   True **False**

   Answer: False

   Justification: Consider $A(x)$ and $B(x)$ so that $A(x)$ is true if and only if x is even and $B(x)$ is true if and only if x is odd. Then, $\forall x\, (A(x) \vee B(x))$ is true (every number is even or odd) but $(\forall x\, A(x) \vee \forall x\, B(x))$ is false (it is not the case that every number is even or every number is odd).

2. **Induction Logic** For each of the following assertions below, circle the correct alternative: (A) it must always hold, or (N) it can never hold or (C) it can hold but need not always.[1] Give a short (couple of lines) justification for your answer. The domain of all quantifiers is the natural numbers. 2 points for each part — 1 point for correct answer and 1 point for justification.

   For the following parts assume that $P(n)$, $Q(n)$, and $R(n)$ are predicates on the natural numbers, and suppose $\forall k \in \mathbb{N}\ P(k) \Rightarrow Q(k+1)$, $\forall k \in \mathbb{N}\ Q(k) \Rightarrow R(k+1)$, and $\forall k \in \mathbb{N}\ R(k) \Rightarrow P(k+1)$.

   (a) If $(P(0) \wedge Q(0) \wedge R(0))$ is true then $\forall n(P(n) \wedge Q(n) \wedge R(n))$ is true.

   A **N** **C**

   Answer: A

   Justification: Note that

   $$\begin{aligned} P(0) &\implies Q(1) \implies R(2) \implies P(3) \implies \cdots \\ Q(0) &\implies R(1) \implies P(2) \implies Q(3) \implies \cdots \\ R(0) &\implies P(1) \implies Q(2) \implies R(3) \implies \cdots \end{aligned}$$

   Therefore, $\forall n \in \mathbb{N}$, $P(n)$, $Q(n)$, and $R(n)$ each appear in one of the chains of implications, meaning that whenever $P(0)$, $Q(0)$, and $R(0)$ are all true, then $P(n)$, $Q(n)$, and $R(n)$ are all true for any n.

   (b) If $(P(0) \vee Q(0))$ is true then $\forall n(P(n) \vee Q(n))$ is true.

   A **N** **C**

   Answer: C

   Justification: In the case that $P(0)$, $Q(0)$, and $R(0)$ are all true, then the implication holds by the reasoning in part (a). However, in the case that $Q(0)$ is true but $P(0)$ and $R(0)$ are false, it is possible for $P(1)$ and $Q(1)$ to both be false. Therefore, this implication holds in some cases but not in all.

   (c) If $(P(0) \vee Q(0) \vee R(0))$ is true then $\forall n\ P(n)$ is true.

   A **N** **C**

   Answer: C

   Justification: If any one of $P(0)$ and $Q(0)$ and $R(0)$ are true, then at least one of the chains of implication are true. In that case, $P(n)$ is true for the values of n that appear in that true chain of implication, but we know nothing about the other chains of implication.

   (d) If $(P(101) \vee Q(101) \vee R(101))$ is true then $(\neg P(0))$ is true.

   A **N** **C**

   Answer: C

   Justification: The chain of implications do not go backwards. Therefore knowing $P(101) \vee Q(101) \vee R(101)$ does not tell us anything about $P(0)$. In particular here are two examples where the premise holds, but $P(0)$ has two different truth values in them.

   If we take $P(x), Q(x), R(x)$ to mean $x \geq 100$, then $P(101) \vee Q(101) \vee R(101)$ holds, the chains of implication hold, and $P(0)$ is false.

   If we take $P(x), Q(x), R(x)$ to mean $x \geq 0$, then again $P(101) \vee Q(101) \vee R(101)$ holds, the chains of implication hold, but this time $P(0)$ is true.

[1] This is just as in HW2. Each part is of the form if P is true then Q is true. Assuming P is true, you must choose (A) if Q is necessarily true, (N) if Q is necessarily false and (C) otherwise.

(e) If $(\neg P(100) \vee \neg P(101) \vee \neg P(102))$ is true then $(\neg P(0) \vee \neg Q(0) \vee \neg R(0))$ is true.

A **N** **C**

Answer: A

Justification The contrapositive of this statement is $(P(0) \wedge Q(0) \wedge R(0)) \Rightarrow (P(100) \wedge P(101) \wedge P(102))$. This is true because from part (a) we know $P(0) \wedge Q(0) \wedge R(0)$ implies that $P(n)$, $Q(n)$, and $R(n)$ are true for all n.

(f) If $(\neg P(100) \wedge \neg P(101) \wedge \neg R(101))$ is true then $(\neg P(0) \wedge \neg Q(0) \wedge \neg R(0))$ is true.

A **N** **C**

Answer: A

Justification The contrapositive of this statement is $(P(0) \vee Q(0) \vee R(0)) \Rightarrow (P(100) \vee P(101) \vee R(101))$. This is true because $P(0) \vee Q(0) \vee R(0)$ implies that one of the chains of implication is correct, and $P(100)$, $P(101)$, and $R(101)$ lie in different chains of implication – so one of them has to be in that true chain of implication.

For the remaining parts, assume we were trying to prove $P(n)$ is true for all $n \in \mathbb{N}$ by induction on n. Instead, we succeeded in proving $\forall k \in \mathbb{N}$ if $P(k)$ is true then $P(2k)$ is true.

(g) If $P(0)$ is true then $\forall n\ P(n)$ is true.

A **N** **C**

Answer: C

Justification: From our known implications, we only know that $P(0) \Rightarrow P(0)$; we do not have any information about $P(n)$ for any $n \neq 0$.

(h) If $P(1)$ is true then $\forall n\ P(2^n)$ is true.

A **N** **C**

Answer: A

Justification: We can show this by induction. In the base case $n = 0$, $P(2^0) = P(1)$ is true. Then if $P(2^k)$ is true, then $P(2^{k+1}) = P(2 \cdot 2^k)$ is true too. This shows that $P(2^n)$ is true for all n.

(i) If $P(0)$ and $P(1)$ are true then $\forall n\ P(n)$ is true.

A **N** **C**

Answer: C

Justification: $P(0)$ being true says nothing about $P(n)$ for any other values of n; $P(1)$ being true only forces $P(2^n)$ to be true for all n. Together, $P(0)$ and $P(1)$ being true implies nothing about $P(3)$.

(j) If $P(5)$ is true then $(\forall n > 1\ P(5 \times 2^n))$ is true.

A **N** **C**

Answer: A

Justification: We can show this by induction. In the base case $n = 0$, $P(5 \times 2^0) = P(5)$ is true. If $P(5 \times 2^k)$ is true, then $P(5 \times 2^{k+1}) = P(2 \times 5 \times 2^k)$ is true too. This shows that $P(5 \times 2^n)$ is true for all n.

3. **Hypercubes**

Prove that any cycle in an n-dimensional hypercube must have even length.

Recall that a cycle is a closed (simple) path and its length is the number of vertices (edges) in it. The n dimensional hypercube is a graph whose vertex set is the set of n-bit strings, with an edge between vertices u, v iff they differ in exactly one bit (Hamming distance $=$ 1).

Answer: Here are three ways to solve this problem: argue via bit flips, parity of Hamming distance, or induction on n. In each case we try to give credit to solutions according to how clearly they expressed the main idea. However, induction on n is more difficult and prone to build-up error.

Answer 1: Bit flips

Main idea: moving through an edge in a hypercube flips exactly one bit, and moreover each bit must be flipped an even number of times to end up at the starting vertex of the cycle.

Here are a sequence of four proofs roughly based on this idea, starting with the most convincing and ending with the least convincing. Also included is a critique saying what is missing in the later proofs.

Proof 1: Each edge of the hypercube flips exactly one bit position. Let E_i be the set of edges in the cycle that flip bit i. Then $|E_i|$ must be even. This is because bit i must be restored to its original value as we traverse the cycle, which means that bit i must be flipped an even number of times. Since each edge of the cycle must be in exactly one set E_j, the total number of edges in the cycle = $\sum_j |E_j|$ is a sum of even numbers and therefore even.

Proof 2: Let C be a cycle in an n-dimensional hypercube. As we go along the edges of C we must end up where we started. Because traversing an edge in a hypercube flips exactly one bit, this means every flipped bit must eventually be flipped back. This means that the number of edges in C must be even. □

Proof 3: Each edge of the cycle flips one bit. Let the starting point be x, and let the farthest the cycle goes from x be y which is at a Hamming distance of k. Then the cycle must flip all those k bits back to return to x. Therefore the total number of edges in the cycle is $k+k=2k$, an even number.

Proof 4: By induction on n, the dimension of the hypercube. For the induction step, we know that the $(n+1)$-dimensional hypercube is made up of two n-dimensional hypercubes, where every vertex in one n-dimensional hypercube has an edge connected to their 'twin' vertex in the other n-dimensional hypercube. Any cycle in the $(n+1)$-dimensional hypercube has to go back and forth from one n-dimensional hypercube to the other an even number of times, since otherwise it will start in one n-dimensional hypercube, and end in the other, and cannot be a cycle. So each edge in the cycle is either in n-dimensional hypercube or the other or goes between the two n-dimensional hypercubes. The number of edges of each of the first two types is even by the induction hypothesis and the last number is even as shown earlier. Therefore the total number of edges in the cycle is even. □

Critique of Proofs 2–4:

(a) Proof 2 is almost correct, and got full credit. However, if one were to be picky one might say that the proof does not make it clear that each bit can be flipped back and forth, and must indeed flip an even number of times (which might be greater than 2) to return to its original value. The proof is also not very explicit about stating that the total number of edges in the cycle is even because the number of edges in the cycle corresponding to each bit position is even.

(b) Proof 3 claims the total length of the cycle is twice the Hamming distance from the starting point x to the farthest point on the cycle. But this claim is false. Starting from x the cycle can move farther from and closer to x many times, and moreover if y is the farthest point in Hamming

distance from x, then the number of edges from x to y in the cycle does not need to be equal to the number of edges from y back to x.

(c) Proof 4 is even farther from being a proof. It correctly points out that the number of times the cycle crosses back and forth between the two n-dimensional cubes must be even (this is just saying that the number of times bit 1 is flipped is even). But then it appeals to the inductive hypothesis, and this is quite meaningless, since the intersection of the cycle with the n-dimensional hypercube will not in general look anything like a cycle.

Answer 2: Parity

Main idea: Parity of Hamming distance is equal to parity of number of edges traversed, so to get a cycle we need an even number of edges.

Sample proof: Let C be a cycle in an n-dimensional hypercube, and let x be any vertex in C. Argue (using induction on k or other techniques) that if we start from x and walk for k edges to another vertex y, then the Hamming distance $H(x,y)$ is even if and only if k is even. Each edge traversal brings about a change in vertex binary representation at one bit. Therefore, the Hamming distance changes by 1 and its parity flips when we traverse an edge.
But as we go along the edges of C we must eventually get back to x. At this point the Hamming distance is $H(x,x) = 0$, which is even. This shows that the number of edges in the cycle must be even as well. □

Answer 3: Induction on n

The most common answer made the mistake of a pretty serious build-up error. Here is an example of such a proof:

Proof by induction on n, the dimension of the hypercube.

Base case: For $n = 2$, there is only one cycle and it has length 4, which is even.
Induction Hypothesis: Any cycle in a n-dimensional hypercube has even length.
Induction Step: Let C be a cycle in the $(n+1)$-dimensional hypercube. The $(n+1)$-dimensional hypercube is made up of two n-dimension hypercubes. There are two cases:

(a) C lies in one of the two n-dimensional hypercubes. In this case we are done by the induction hypothesis.

(b) C crosses between the two n-dimensional hypercubes. In this case there is an even cycle in each of the two n-dimensional hypercubes. To connect them, we must remove an edge from each of the two cycles in the n-dimensional hypercubes and connect each of the endpoints to their twin vertex in the other n-dimensional hypercube. Now the number of edges in the cycle is odd + odd + 2 = even, where odd = even cycle − one edge.

□

There are many problems with this proof. First, it completely ignores the fact that the cycle can go back and forth between the two n-dimensional hypercubes a number of times. But even if we were to focus on the special case where it goes back and forth just once, the proof is still seriously wrong. This is because even in this case, the part of the cycle in each n-dimensional hypercube is just a path between two possibly distant vertices, i.e., it need not look anything like a cycle with one edge deleted. In particular this path could be of even or odd length.

There is nonetheless a way of writing down a correct proof by induction. This involves a couple of ideas, including strengthening the induction hypothesis to any tour in the n-dimensional hypercube:

Main idea: A tour in an n-dimensional hypercube can be decomposed into some components in the two $(n-1)$-dimensional subcubes plus an even number of crossing edges. The components in both subcubes can be superimposed to form a tour in the $(n-1)$-dimensional hypercube, allowing us to apply the inductive hypothesis.

Sample proof: We use induction on n.
Base case: For $n = 2$, it is easy to show that every tour has even length.
Inductive hypothesis: Any tour in the $(n-1)$-dimensional hypercube has even length.
Inductive step: Let C be a tour in the n-dimensional hypercube. Consider the decomposition of the n-dimensional hypercube into two $(n-1)$-dimensional subcubes. We decompose C into three parts: the edges that lie in the first subcube, the edges in the second subcube, and the edges crossing the subcubes. Argue that because C is a tour, the number of crossing edges must be even (but not necessarily 2). The edges of C in each subcube do not have to be a tour; in fact they can be collections of disjoint paths. The components in one subcube also don't have to be equal or symmetrical to the components in the other subcube. But argue that when you superimpose them (superimpose the vertices of one subcube with the corresponding vertices in the other subcube), you get a tour in the $(n-1)$-dimensional hypercube. Now apply the inductive hypothesis to conclude that the total number of edges of C in both subcubes must be even. To get the total number of edges in C we need to add the number of crossing edges, which is also even. This completes the inductive step. □

4. **Stable Marriage**

Consider the traditional propose and reject stable marriage algorithm. On day j, let $P_j(M)$ be the rank of the woman that M proposes to (where the first woman on his list has rank 1 and the last has rank n). Also, let $R_j(W)$ be the total number of men that woman W has rejected up through day $j-1$ (i.e. not including the proposals on day j).

(a) Prove that $\sum_M P_j(M) - \sum_W R_j(W)$ is independent of j. What is its value?

Answer: On day $j = 1$, each man proposes to the first woman on his list so $\sum_M P_1(M) = n$, and no woman rejected any man through day 0, and therefore $\sum_M P_1(M) - \sum_W R_1(W) = n$.
In general, each time a woman rejects a man on day $j-1$, it increases $\sum_W R_j(W)$ by exactly 1. It also increases $\sum_M P_j(M)$ by exactly 1, since the rejected man proposes to the next woman on his list on day j. Therefore $\sum_M P_j(M) - \sum_W R_j(W)$ stays constant and is independent of j.

More formally, we can prove this by induction on j, with $j = 1$ as base case.
Induction Hypothesis: Assume $\sum_M P_j(M) - \sum_W R_j(W) = n$.
Induction Step: The quantity $\sum_W R_{j+1}(W) - \sum_W R_j(W)$ is exactly the number of men rejected by women on day j. But each of the rejected men propose to the next woman on their list on day $j+1$, and so this quantity is also equal to $\sum_M P_{j+1}(M) - \sum_M P_j(M)$. Equating the two, we get

$$\sum_W R_{j+1}(W) - \sum_W R_j(W) = \sum_M P_{j+1}(M) - \sum_M P_j(M).$$

Therefore,

$$\sum_M P_{j+1}(M) - \sum_W R_{j+1}(W) = \sum_M P_j(M) - \sum_W R_j(W)$$

and the right hand side is equal to n by the induction hypothesis. □

(b) Conclude that one of the men or women must be matched to someone who is ranked in the top half of their preference list. You may assume that n is even.

Answer: Assume that no man is matched with a woman in the top half of his preference list. Each of them must have been rejected at least $\frac{n}{2}$ times, for a total of at least $\frac{n^2}{2}$ rejections. This implies that at least one woman must have rejected at least $\frac{n}{2}$ men (because if not, then the total number of rejections must be less than $\frac{n}{2} \times n$, contradiction). But now, by the improvement lemma, this woman must be matched with a man she likes more than the $\frac{n}{2}$ men she rejected, meaning that the man she is matched with is in the top half of her preference list. □

Alternative proof:

Assume towards contradiction that every man and every woman is matched to someone who is ranked in the bottom half of their preference list.
Observe that a man M is matched to someone in the top half of his preference list if and only if $P_m(M) \leq \frac{n}{2}$, where m is the last day of the algorithm. Therefore, if M is matched to someone in the bottom half of his preference list, then $P_m(M) > \frac{n}{2}$, i.e., $P_m(M) \geq \frac{n}{2} + 1$. Summing over the men gives us $\sum_M P_m(M) \geq \frac{n^2}{2} + n$. By part (a), it follows that $\sum_W R_m(W) = \sum_M P_m(M) - n \geq \frac{n^2}{2}$.
Observe also that if $R_m(W) \geq \frac{n}{2}$, then by the improvement lemma, W must be matched to someone in the top half of her preference list. Therefore, from our assumption that W is matched to someone in the bottom half of her preference list, we get $R_m(W) < \frac{n}{2}$. Summing over the women gives us $\sum_W R_m(W) < \frac{n^2}{2}$. But this contradicts our earlier result above! □

5. **Counterfeit coins**

A bank has a pile of 3^n coins, one of which is counterfeit. The counterfeit coin looks identical to a regular coin in every way except that it is lighter. The bank has a pan balance, but not much time. Give a recursive algorithm that uses just n weighings to identify the counterfeit coin. Ideally your algorithm will consist of 3-4 lines of pseudocode. Each weighing involves placing some coins on each pan of the balance, resulting in one of three outcomes: left side heavier, or right side heavier, or two sides are the same weight. Prove by induction that your algorithm is correct and terminates in the claimed number of weighings.

Note that a majority of the points for this question are for proving the correctness of your algorithm and the bound on the number of weighings. However, you must give a recursive algorithm and failure to do so will result in 0 points for the entire question.

Answer: This question has three parts, the algorithm, a proof of correctness and a proof of runtime.

Algorithm:

```
function FindCounterfeit(C)                          ▷ Takes in a pile of coins
    if Size(C)= 1 then                               ▷ Base case
        return Only coin in pile                     ▷ The only coin must be counterfeit
    end if
    Split coins into 3 equal piles, A, B, C          ▷ Recursive case
    R ← Weigh(A, B)
    if R = "Same weight" then
        return Find Counterfeit(C)                   ▷ C must be lighter
    else
        return Find Counterfeit(Lighter of A and B)
    end if
end function
```

Proof of correctness:

We prove that FindCounterfeit correctly finds the counterfeit coin in the pile by induction on n.

Base case When $n = 0$, there is only one coin, and since there is exactly one counterfeit coin, this coin must be the counterfeit.

Inductive Hypothesis Suppose that when $n = k$, FindCounterfeit can find the counterfeit coin in a pile with 3^k coins containing exactly one counterfeit coin which is lighter than the rest.

Inductive Step Suppose that we have a pile of 3^{k+1} coins. As in the algorithm, we divide it into three sub-piles of 3^k coins, and weigh two of them against each other, and then isolate one pile to recursively call FindCounterfeit on. We need to prove that this pile contains the counterfeit coin. There are two cases:

(a) One pile is lighter than the other. Since the piles have equal number of coins, and there is only one counterfeit coin, it follows that the lighter pile must have the counterfeit coin. Recursion on that lighter sub-pile of 3^k coins would then allow us to correctly find the counterfeit coin, by the induction hypothesis.

(b) The two weighed piles have exactly the same weight. Again, since there is exactly one counterfeit coin in the pile of 3^{k+1} coins, the counterfeit coin must be in the remaining pile. We recurse on the remaining pile, and thus by the induction hypothesis will find the counterfeit coin.

Proof of runtime:

We prove with induction on n that FINDCOUNTERFEIT, when run on a pile of 3^n coins, will call WEIGH exactly n times.

Base case When $n = 0$, the only coin is returned, so WEIGH is not called.

Inductive Hypothesis Suppose that when $n = k$, calling FINDCOUNTERFEIT on a pile of 3^k coins will call WEIGH k times.

Inductive Step When we call FINDCOUNTERFEIT on a pile of 3^{k+1} coins, it only calls WEIGH once before a recursive call of FINDCOUNTERFEIT on a pile of 3^k coins. Because by the induction hypothesis this will result in k additional weighings. Therefore calling FINDCOUNTERFEIT on a pile of 3^{k+1} coins will result in $k+1$ weighings.

Grading rubric:

Most students got the majority of points if the general idea is correct, i.e., divide the pile into three equal piles and weigh two of them once. Some lost points because they lacked proofs of runtime/correctness, or there are some errors in their proofs, or the proofs weren't explicit enough in explaining the runtime or showing why the algorithm always can find the pile with the counterfeit in it. Some students also lost points for minor errors in the algorithm, such as missing a base case or confusing lighter with heavier.

There are two other common incorrect answers:

Incorrect answer 1:

Divide the piles into two, leaving one coin out if necessary. Then weigh each of the two piles and recurse on them. This algorithm will not run in exactly n time as required by this problem, although it will run in approximately $\log_2 3^n = n \log_2 3$ time, which is in $O(n)$ but not exactly n. Students with this solution will get at most 9 points if they have a proof of correctness or a proof of runtime, or less if these are missing/incorrect.

Incorrect answer 2:

Take 2 coins at a time and weigh them until all the coins are weighed. Although this algorithm can be written recursively, it would take $O(3^n)$ time, which is way more than the time we requested. Such solutions and similar incorrect algorithms will get at most 3 points.

6. **Trees** (This question is harder than the rest. Attempt it only after you take a good shot at the rest.)

   Show that the edges of a complete graph on n vertices for n even can be partitioned into $\frac{n}{2}$ edge disjoint spanning trees.

   Recall that a complete graph is an undirected graph with an edge between every pair of vertices. The complete graph has $\frac{n(n-1)}{2}$ edges. A spanning tree is a tree on all n vertices — so it has $n-1$ edges. So the complete graph has enough edges (for n even) to create exactly $\frac{n}{2}$ edge disjoint spanning trees (i.e. each edge participates in exactly one spanning tree). You have to show that this is always possible.

   Answer: We proceed by induction.

   Base Case: Consider a complete graph on 2 vertices; this is simply •—•. This can clearly be partitioned into $\frac{2}{2} = 1$ edge disjoint spanning tree, because the graph is already a tree.

   Inductive Hypothesis: Assume that the edges of a complete graph on k vertices (for k even) can be partitioned into $\frac{k}{2}$ edge disjoint spanning trees.

   Inductive Step: We need to partition the edges of a complete graph G_{k+2} on $k+2$ vertices into $\frac{k}{2}+1$ edge disjoint spanning trees.

   To do this, label the vertices of G_{k+2} as $v_1, v_2, \ldots, v_{k+2}$. Remove the vertices v_{k+1} and v_{k+2} (and associated edges) to form a complete graph G_k with k vertices $v_1, \ldots, v_k$. By the inductive hypothesis, G_k has $\frac{k}{2}$ edge disjoint spanning trees; call these trees $T_1, \ldots, T_{k/2}$.

   Add the vertices v_{k+1} and v_{k+2} back into G_k to once again form the graph G_{k+2}. These vertices come with $2k+1$ extra edges, connecting (v_i, v_{k+1}) and (v_i, v_{k+2}) for each $i = 1, 2, \ldots, k$, and also (v_{k+1}, v_{k+2}). These edges must be included into spanning trees.

   We wish to extend the trees $T_1, \ldots, T_{k/2}$ to include the new vertices v_{k+1} and v_{k+2}. To do this, for each tree T_i, attach two new edges (v_i, v_{k+1}) and $(v_{i+k/2}, v_{k+2})$. This extends each tree T_i to be a spanning tree.

   The remaining edges form one additional spanning tree. These edges are $(v_{i+k/2}, v_{k+1})$ and (v_i, v_{k+2}) for $i = 1$ to $k/2$, along with the connecting edge (v_{k+1}, v_{k+2}). These edges connect each of the vertices v_{k+1} and v_{k+2} to half the remaining vertices, and together with the edge between v_{k+1} and v_{k+2} this gives the desired spanning tree.

   Therefore, we have covered the graph in $\frac{k}{2}+1$ edge disjoint spanning trees. This completes the induction.

   Remark: The key idea here is the following:

   Take a graph with k vertices that is partitioned into $\frac{k}{2}$ spanning trees. In the inductive step, we want to add two vertices (with associated edges). To maintain a partitioning into spanning trees, we must expand the preexisting $\frac{k}{2}$ trees to the new vertices, but this is a bit subtle!

   We need to add the two new vertices to each of the preexisting $\frac{k}{2}$ trees, which takes $2 \cdot \frac{k}{2} = k$ edges connecting the preexisting k vertices to the two new vertices. *It's really important that we use only one edge out of each of the original vertices!* This is because otherwise, we would use up both new edges out of one of the vertices v_j, but then our final new spanning tree wouldn't be able to reach v_j, so the remaining $k+1$ edges wouldn't be able to form a spanning tree!

   So to do this, we need to split the original k vertices into two equal subsets of $\frac{k}{2}$ vertices each, and connect each half to one of the two new vertices. Once we do that, we can then justify forming a new spanning tree from the remaining edges, which allows us to complete the argument.

Rubric: This was a hard problem, and we graded this problem as an extra credit problem. In order to receive credit, you needed to have the key idea above – that was really the heart of the problem!