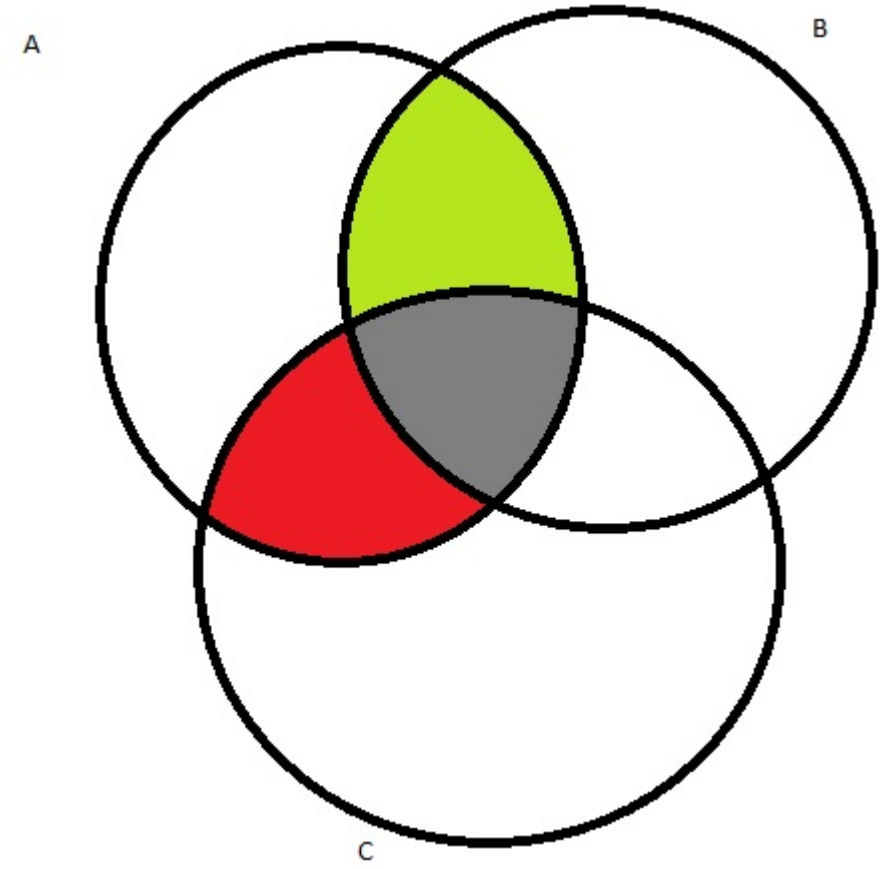
Appendix B

Michelle Bodnar, Andrew Lohr

August 31, 2015

Exercise B.1-1
FIrst we consider

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

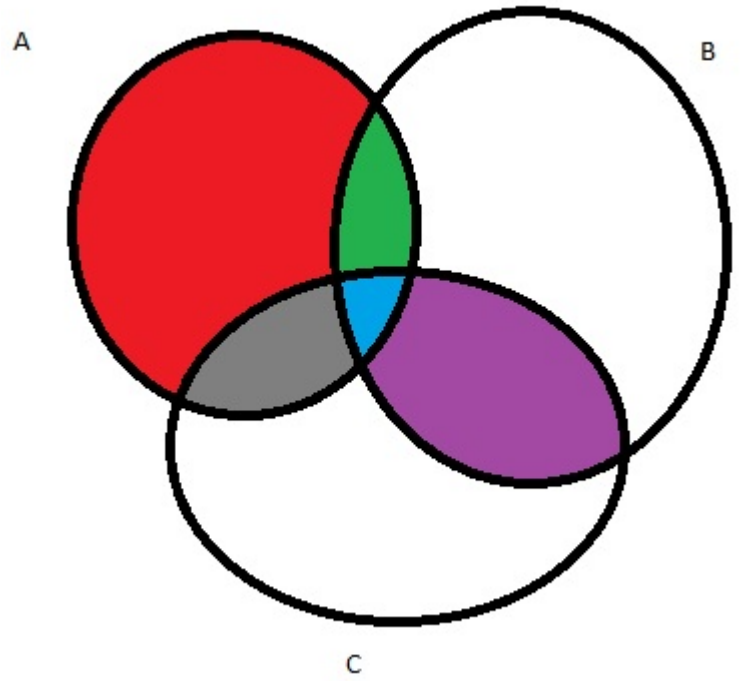


For the first picture, we can see that the shaded regions are all the regions that are in A and also in either B or C, and so are in the set described on the left hand side. Since the green and gray shaded regions are in both A and B,

they are in the right hand side, also, the red and gray regions are in A and C and so are also in the right hand side. There aren't any other regions that are either both in A and B or both in A and C, so the shaded regions are the right hand side of the equation as well.

Next we consider

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$



The ony regions in $B \cap C$ are blue and purple. All the other colored regions are in A. So, the shaded regions are exactly the left hand side. To see what is in the right hand side, we see what is both in $A \cup B$ and $A \cap C$. Individually both both contain all of the shaded regions plus one of the white regions. Since the white region contained differs, their intersection is all of the shaded regions.

Exercise B.1-3

Exercise B.1-5

For each of the elements of S, some subset of S can either contain that element or not contain that element. If the decision differs for any of the $|S|$ many elements, then you've just created a distinct set. Since we make a decision between two options $|S|$ many times, the total number of possible sets is $2^{|S|}$. It can be thought of as the number of leaves in a complete binary tree of depth $|S|$.

Exercise B.2-1
To see that it is a partial ordering, we need to show it is reflexive, antisymmetric, and transitive. To see that it is reflexive, we need to show $S \subseteq S$. That is, for every $x \in S$, $s \in S$, which is a tautology. To see that it is antisymmetric, we need that if $S_1 \neq S_2$ and $S_1 \subseteq S_2$ then $S_2 \not\subseteq S_1$. Since $S_1 \neq S_2$ there is some element that is in one of them but not in the other. Since $S_1 \neq S_2$, we know that that element must be in S_2 because if it were in S_1 it would be in S_2. Since we have an element in S_2 not in S_1, we have $S_2 \not\subseteq S_1$. Lastly, we show transitivity, suppose $S_1 \subseteq S_2 \subseteq S_3$. This means that any element that is in S_1 is in S_2. But since it is in S_2, it is in S_3. Therefore, every element in S_1 is in S_3, that is $S_2 \subseteq S_3$.

To see that it is not a total order, consider the elements $\{1, 2\}$ and $\{2, 3\}$. Neither is contained in the other, so there is no proscribed ordering between the two based off of inclusion. If it were a total ordering, we should be able to compare any two elements, which we just showed we couldn't.

Exercise B.2-3

a. Consider the set of vertices in some non-complete, non-empty graph where we make two vertices related if they are adjacent or the same vertex.

b. Consider the vertices in a digraph where we have aRb if there is some possibly empty path from a to b. For a concrete example, suppose we have only the set $\{0, 1\}$ and the relations $0R0$, $0R1$, and $1R1$.

c. Consider the relation that makes no two elements related. This satisfies both the symmetric and transitive properties, since both of those require that certain elements are related to conclude that some particular other elements are related.

Exercise B.2-5
Professor Narcissus is full of himself! The relation that makes no two elements related to each other is both symmetric and transitive, but is not reflexive.

Exercise B.3-1

a. Since f is injective, for every element in it's range, there is at most one element that maps to it. So, we proceed by induction on the number of elements in the domain of the function. If there is only a single element in the domain, since the function has to map to something, we have that $|B| \geq 1 = |A|$. Suppose that all functions that are on a domain of size n satisfy this relation. Then, look at where that $n + 1$st element gets mapped. This point will never be mapped to by any of the other elements of the domain. So, we can think of restricting the function to just the first n

elements, and use the inductive assumption to get the desired relation of the two sets.

b. As a base case, assume that $|B| = 1$. Then, since the function is surjective, some element must map to that, so, $|A| \geq 1 = |B|$. Not, suppose it is true for all function with a range of size at most n. Then, just look at all the elements that map to the $n+1$st element, there is at least one by surjectivity. This gets us that $|A| \geq 1 + |A'| \geq 1 + |B'| = |B|$.

Exercise B.3-3
Define R^{-1} by $aR^{-1}b$ if and only if bRa. This clearly swaps the domain and range for the relation, and so, if R was bijective, it is the inverse function. If R was not bijective, then R^{-1} might not even be a function.

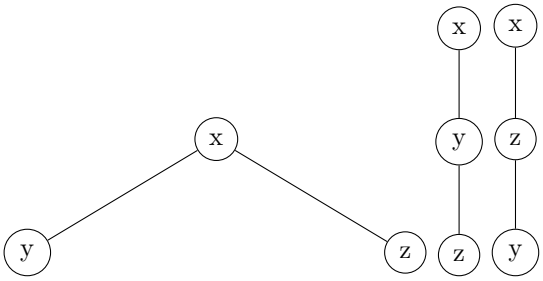
Exercise B.4-1
Let $f(u, v)$ be equal to 1 if u shook hands with v. Then, $\sum_{v \in V} degree(v) = \sum_{v \in V} \sum_{u \in V} f(v, u)$ then, since handshaking is symmetric, we are counting both when u shakes v hand and when v shakes u's hand. So, this sum is $\sum_{e \in E} 2 = 2|E|$.

Exercise B.4-3
We proceed by induction on the number of vertices. If there is a single vertex, then the inequality trivially holds since the left hand side is the size of a set and the right hand side is zero. For the inductive case, pick one vertex in particular. We know that that vertex must have at least one edge to the rest of the vertices because the graph is connected. So, when we take the induced subgraph on the rest of the vertices, we are decreasing the number of edges by at least one. So, $|E| \geq 1 + |E'| \geq 1 + |V'| - 1 = |V| - 1$.

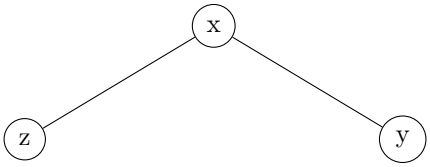
Exercise B.4-5
The undirected version of the graph in B.2(a) is on the same vertex set, but has $E = \{(1, 2), (2, 4), (2, 5), (4, 1), (4, 5), (6, 3)\}$. That is, we threw out the antisymmetric edge, and the self edge. There is also a difference in that now the edges should be viewed as unordered unlike before.

The directed version of B.2(b) looks the same, except it has arrows drawn in. One from 2 to 1, one from 1 to 5, one from 2 to 5, and one from 3 to 6.

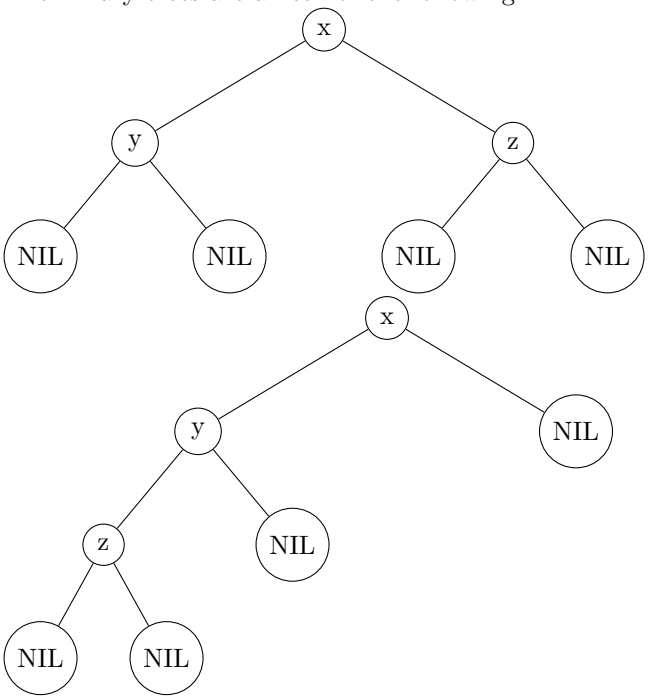
Exercise B.5-1
There are three such rooted trees:

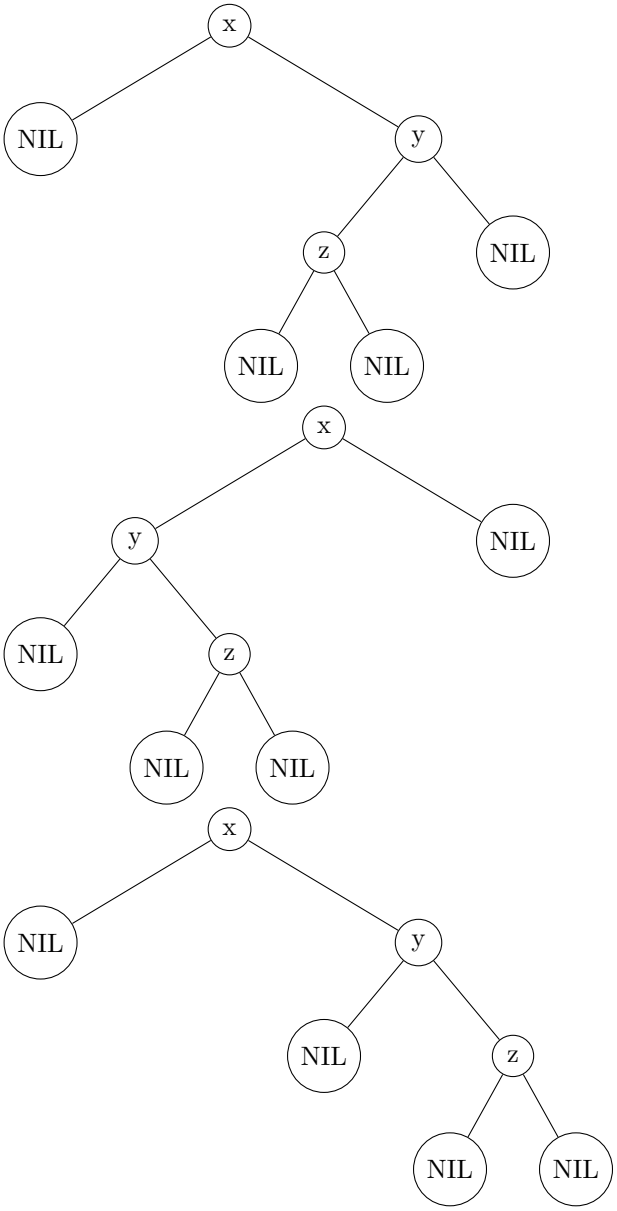


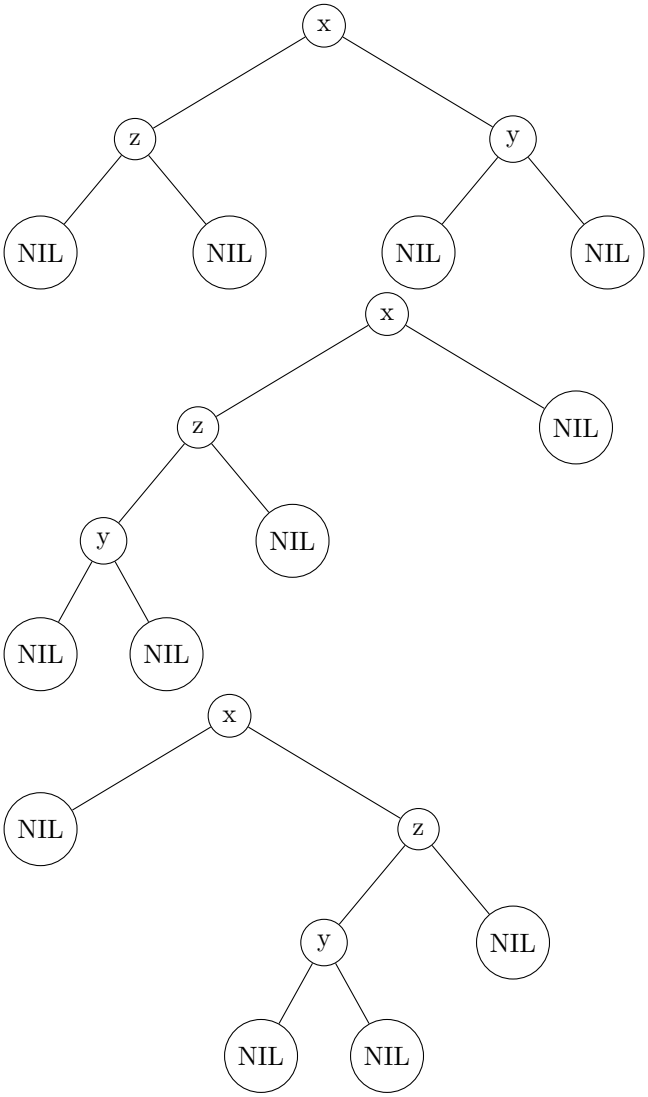
The ordered trees are those listed above, in addition to the tree

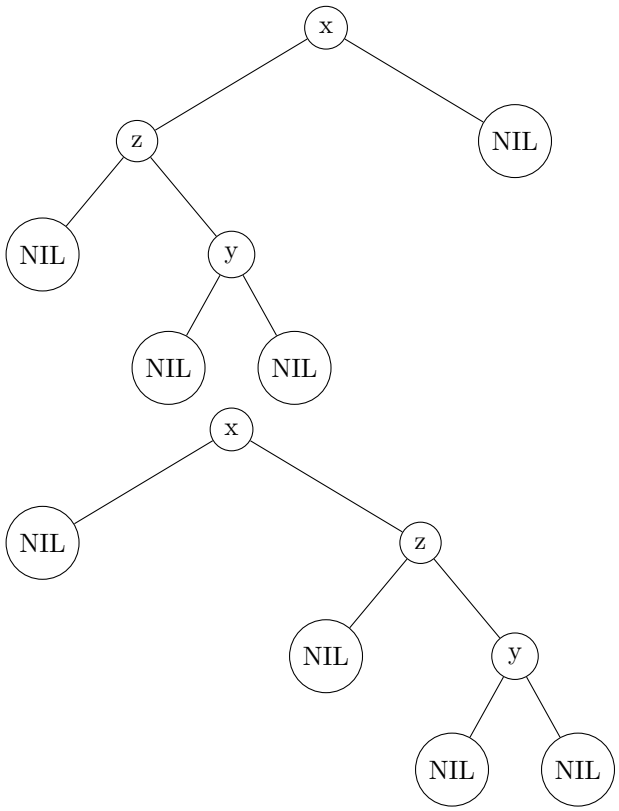


The Binary trees are all ten of the following:




x
NIL
y
z
NIL
NIL
NIL
x
y
NIL
NIL
z
NIL
NIL
x
NIL
y
NIL
z
NIL
NIL


x
z
y
NIL
NIL
NIL
NIL
x
z
NIL
y
NIL
NIL
NIL
x
NIL
z
y
NIL
NIL
NIL



Exercise B.5-3

As a base case, consider the binary tree that consists of a single node. This tree has no degree two nodes and one leaf, so, the number of degree two nodes is one less than the number of leaves. As an inductive step, suppose it is true for all binary trees with at most n degree two nodes. Then, let G be a binary tree with $n+1$ internal nodes. Let v be the first node, possibly the leaf itself, that is the child of a degree two node, and can be obtained by taking the parent pointer from the leaf. Then, consider the tree obtained by removing v and all its children. Doing so removes only one leaf, and it makes the parent of v drop to being degree 2. None of the children of v were degree 2 because otherwise we would of stopped earlier when we were selecting v. Since we have decreased the number of degree two nodes and leaves both by 1, we have completed the inductive case, because if $|T'|$ is the modified tree, the number of leaves in T is one more than that in T' which means it is one more than the number of degree 2 nodes in T', which is the number of degree two nodes in T.

Since a full binary tree has each internal node with degree two, this result gets us that the number of internal nodes in a full binary tree is one more than the number of leaves.

Exercise B.5-5

We will perform structural induction. For the empty tree that has $n = 0$, the equation is obviously satisfied. Now, let r be the root of the tree T, and let T_L and T_R be the left and right subtrees respectively. By the inductive hypothesis, we may assume that $e_L = i_L + 2n_L$ and $e_R = i_R + 2n_R$. Then, since being placed as a child of the root adds one to the depth of each of the nodes in both T_L and T_R, we have that $i = i_L + i_R + n_L + n_R$ and

$$\begin{aligned} e =& e_L + |leaves(T_L)| + e_R + |leaves(T_R)| \\ =& e_L + e_R + |leaves(T)| \\ =& i_L + 2n_L + i_R + 2n_R + |leaves(T)| \\ =& i + n_L + n_R + |leaves(T)| \\ =& i + n - 1 + |leaves(T)| \end{aligned}$$

By problem B.5-3, since the tree is full, we know that $|leaves(T)|$ is one more than the number of internal nodes of T. So, $e = i+2n$, completing the induction.

Exercise B.5-7
Suppose to a contradiction that there was some binary tree with more than 2 leaves that had no subtree with a number of leaves in the desired range. Since $L > 1$, the root is not a leaf. Now, define the sequence $x_0 = root$, x_{i+1} is the larger(more leaves) of the children of x_i, until we have reached a root. Now, we consider the number of leaves in the subtree rooted at each x_i. For x_0 it is L, which is too large, and at x_h, it is 1 which is not too large. Now, we keep incrementing from x_0 to x_1 to x_2, and so on until we have some number of leaves that falls in the desired range. If it happens we are done and have contradicted the assumption that this binary tree didn't have such a subtree. Since it doesn't that means there is some step where the number of leaves jumps from more than $2L/3$ to less than $L/3$, Since both of the children of the next x_i were less than $L/3$, their sum cannot be more than $2L/3$, a contradiction.
Problem B-1

a. If the tree is unrooted, pick an arbitrary node to be the root. Assign color 0 to all nodes that are at an even height, and assign color 1 to all nodes that are at an odd height. Since the child of any node has height one greater, there will never be two adjacent nodes that have received the same color.

b. We show the following implications in order to get equivalence of all three

$1 \Rightarrow 2$ Since the graph is bipartite, we can partition the vertices into two sets L and R so that there are no edges going between vertices in L and no edges going between vertices in R. This means that we can assign color 0 to all vertices in L and color 1 to all vertices in R without having any edges going between two vertices of the same color.

$2 \Rightarrow 3$ Suppose to a contradiction that there was a cycle of odd length, but we did have a valid two coloring. As we go along the cycle, each time we go to the next vertex, the color must change because no two adjacent vertices can have the same color. If we go around the cycle like this though, we have just flipped the color an odd number of times, and have returned back to the original color, a contradiction.

$3 \Rightarrow 2$ If G has no cycles of an odd length, then we can just perform the greedy operation to two color. That is, we pick a vertex, color it arbitrarily, then, for any vertex adjacent to a colored vertex, we color it the opposite color. If this process doesn't end up coloring everything, i.e. the graph is disconnected, we repeat it. Since the only way this process could fail is if there is an odd length cycle, it provides a two coloring, proving that the graph is two colorable.

$2 \Rightarrow 1$ Partition the vertices based on what color they received. Since there are no edges going between the vertices of the same color, there won't be any edges going between vertices that are in the same part in the partition.

c. Consider the process where we pick an arbitrary uncolored vertex and color it an arbitrary color that is not the color of any of its neighbors. Since a vertex can only have at most d neighbors, and there are $d+1$ colors to choose from, this procedure can always be carried out. Also, since we are maintaining at each step that there are no two adjacent vertices that have been colored the same, we have that the end result of all the coloring is also valid.

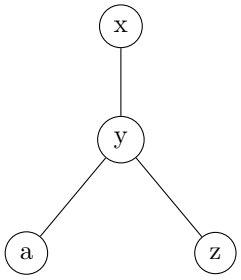
d. Let V' be the set of vertices whose degree is at least $\sqrt{|E|}$. The total number of edges incident to at least one of these vertices is at least $\frac{|V'|\sqrt{|E|}}{2}$ by exercise B.4-1. Since this also has to be bounded by the total number of edges in the graph, we have that $\frac{|V'|\sqrt{|E|}}{2} \leq |E|$, which gets us that $|V'| \leq 2\sqrt{|E|}$. So, we'll assign all of the vertices in V' their own distinct color. Then, so long as we color the rest of the edges with colors different from those used to color V', the edges between V' and the rest of the vertices won't be important for affecting the validity of the coloring. So, we look at the graph induced on the rest of the vertices. This graph is degree at most $\sqrt{|E|}$ because we already removed all the vertices of high degree. This means that by the previous part, we can color it using at most $\sqrt{|E|}+1$ colors. Putting it together, the total number of colors used to obtain a valid coloring is $2\sqrt{|E|}+\sqrt{|E}+1 \in O(\sqrt{|E|}) = O(\sqrt{|V|})$.

Problem B-3

a. Suppose that $n \geq 4$, as all smaller binary trees can be easily enumerated and this fact checked for. Start at the root, let that be x_0, then, let x_{i+1} be the larger child of x_i, or it's only child if there is just one. Eventually this process will stop once it has reached a root. Let $s(v)$ be the size of the

subtree rooted at v. Since we always pick the larger of the two subtrees, we have that $s(x_i) \leq 2s(x_{i+1}) + 1$. So, if we have that $s(x_{i+1}) \leq n/4$, then, we have that $s(x_i) \leq 2n/4 + 1 \leq 3n/4$. Since eventually this sequence goes to 1, which is below the range, and it starts at n, which is above, we must have that at some point it dips below, the parent of this node is the one that we need to snip off of the original tree to get the desired size subtree.

b. Take a binary tree on four vertices as follows:



Where it is unimportant whether y is the left or right child of x. Then, any cut that is made of the three edges in the graph results in there being one set of vertices of size 3 that are connected and one vertex that is all by its lonesome self. This means that the larger of the two sets the vertices is partitioned into has size $3 = (3/4) \cdot n$.

c. We will make a single cut to the original tree to take off a piece that is less than or equal to $\lfloor \frac{n}{2} \rfloor$. As in part a, we let x_0 be the root, and let x_{i+1} be the larger child of x_i. We show that the size of the subtree rooted at x_i, say $s(x_i)$ can old drop by at most a factor of 3. To do this, we use the fact that $s(x_i) \leq 2s(x_{i+1}) + 1$. So, if $s(x_{i+1}) \leq \frac{s(x_i)}{3}$, then,

$$\begin{aligned} s(x_i) &\leq 2s(x_{i+1}) + 1 \leq \frac{2s(x_i)}{3} + 1 \\ \frac{s(x_i)}{3} &\leq 1 \\ s(x_i) &\leq 3 \end{aligned}$$

However, for there to even be a x_{i+1}, it must have size at least one, so, even then, we have that has dropped by at most a factor of 3.

This means that we can always select a tree that is less than a fact of three below $\lfloor \frac{n}{2} \rfloor$. So, what we do is cut off that subtree, decrease the amount we are trying to cut off by that amount, and then continue. Each cut doing this procedure must decrease the most significant digit of the ternary representation of the amount need to be cut off by 1. This means that it needs at most $2 \log_3(n)$ many cuts, but this is $O(\lg(n))$, so, we are done.