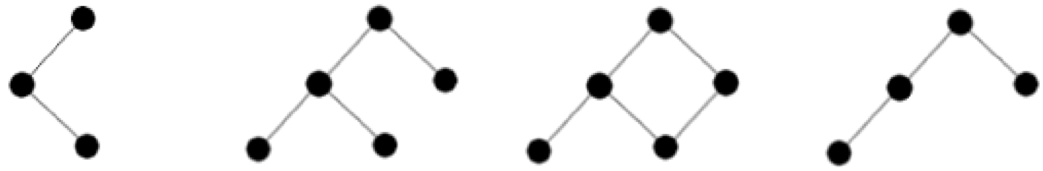
DMC Problem 8.11(a)

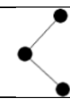
Consider these graphs:



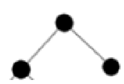
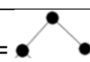
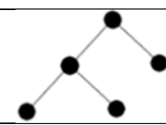
(a) Which are RBTs? Explain your answers.

I'll assign each tree a number (one to four) starting from the left to right.

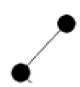
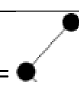
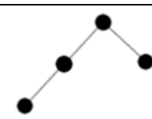
Tree one is a RBT because it can be derived like so:

<table>
<tr><td rowspan="2">E</td><td>T1 = E</td><td rowspan="2">•</td><td>T1 = •</td><td rowspan="2"></td><td>T1 = E</td><td rowspan="2"></td></tr>
<tr><td>T2 = E</td><td>T2 = E</td><td>T2 = •</td></tr>
</table>

Tree two is a RBT because it can be derived like so

<table>
<tr><td rowspan="2">E</td><td>T1 = E</td><td rowspan="2">•</td><td>T1 = •</td><td rowspan="2"></td><td>T1 = </td><td rowspan="2"></td></tr>
<tr><td>T2 = E</td><td>T2 = •</td><td>T2 = •</td></tr>
</table>

Tree four is a RBT because it can be derived like so:

<table>
<tr><td rowspan="2">E</td><td>T1 = E</td><td rowspan="2">•</td><td>T1 = •</td><td rowspan="2"></td><td>T1 = </td><td rowspan="2"></td></tr>
<tr><td>T2 = E</td><td>T2 = E</td><td>T2 = •</td></tr>
</table>

Tree three is not a RBT. I'll prove this by showing (through structural induction) that every RBT node can only have at most 1 parent.

I'll start with the statement P(Sn): Every node in a RBT, n, has at most one parent.

Base Case: The empty tree doesn't contain a node so it doesn't have any parents. The statement is vacuously true.

Induction Step: I'll show $P(S_1)$ and $P(S_2)$ and $P(S_3)$… $P(S_n) \rightarrow P(S_{n+1})$ using a direct proof.

I'll assume that everything ($P(S_1)$ and $P(S_2)$ and $P(S_3)$… $P(S_n)$) up until the next element $P(S_{n+1})$ is true.

The constructor for a RBT takes two RBTs (for which the property is assumed to be true) and connects their roots to a new root so that each of their other roots has a single parent. The constructor preserves the property.

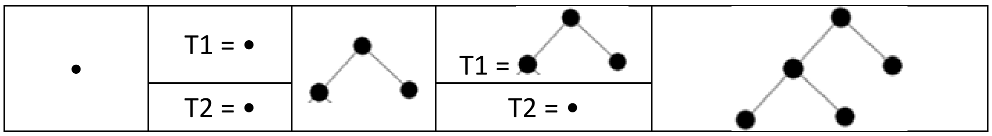
By induction, $P(S_n)$ is true for all RBT.

Therefore, tree three cannot be a RBT because it violates this property – one of the nodes has two parents.

DMC Problem 8.11(b)

(b) Which are RFBTs? Explain your answers.

Only tree two is a RFBT. Here is the derivation:



By structural induction I will prove that every node in a RFBT has exactly 0 or two children – therefore excluding trees 1, 3, and 4.

I'll start with the statement $P(S_n)$: In a RFBT, n, every node has exactly zero or two children.

Base Case: $P(S_1)$ is obviously true because a single node has zero children.

Now, induction step: I'll assume $P(S_n)$ is true for every RFBT from S_1 to S_n and implies $P(S_{n+1})$ is also true.

The constructor rule says that to make a RFBT you must take two disjointed trees and connect them to a single root. This preserves the property because the new root will have the previous two trees roots as children. Since we've assumed that the two previous trees have the property, this means the new tree also has the property.

We also don't have to worry about an empty tree making the constructor create a node where there is only one child because the empty set is not considered to be a RFBT.

DMC Problem 9.2(g)

(g) Compute the sum for the following: $\sum_{i=0}^{n} \sum_{j=0}^{i} 2^i$

Start with the inner sum. This one is fairly simple to reason out. Since i is the upper bound, the inner sum is just adding 2^i (i+1) times. It's (i+1) because j starts at zero. This leaves us with:

$$\sum_{i=0}^{n} 2^i(i+1)$$

We can distribute the 2^i to each term in the parenthesis and apply one of the rules concerning the addition of terms within a summation to get this:

$$\sum_{i=0}^{n} 2^i(i+1) = \sum_{i=0}^{n} i * 2^i + \sum_{i=0}^{n} 2^i$$

We have rules for both these sums from the book.

The first can be written as $(n-1)2^{n+1} + 2$
The second can be written as $2^{n+1} - 1$

Adding these formulas gets

$$(n-1)2^{n+1} + 2 + 2^{n+1} - 1 = n2^{n+1} - 2^{n+1} + 2 + 2^{n+1} - 1 = n2^{n+1} + 1$$

Thus, for our final answer is the formula $f(n) = n2^{n+1} + 1$

DMC Problem 6.19(a)

You have a stack of *n* boxes. You may split a stack into two. If you split a stack of *k* boxes into two stacks of k_1, k_2 ($k_1 + k_2 = k$), you earn $\$k_1k_2$ (the product). You must reduce the stack of *n* boxes to *n* stacks of one box and earn as much money as possible.

(a) Tinker with different ways of unstacking 4 and 5 boxes.

Four boxes:

<table>
<tr><td colspan="4">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="4">4</td><td></td></tr>
<tr><td>1</td><td colspan="3">3</td><td>3</td></tr>
<tr><td></td><td>1</td><td colspan="2">2</td><td>2</td></tr>
<tr><td></td><td></td><td>1</td><td>1</td><td>1</td></tr>
</table>

Number of turns: 3
Money earned: 6

<table>
<tr><td colspan="4">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="4">4</td><td></td></tr>
<tr><td colspan="2">2</td><td colspan="2">2</td><td>4</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1+1</td></tr>
</table>

Number of turns: 3
Money earned: 6

Five boxes:

<table>
<tr><td colspan="5">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="5">5</td><td></td></tr>
<tr><td>1</td><td colspan="4">4</td><td>4</td></tr>
<tr><td>1</td><td colspan="2">2</td><td colspan="2">2</td><td>4</td></tr>
<tr><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>1+1</td></tr>
</table>

Number of turns: 4
Money earned: 10

<table>
<tr><td colspan="5">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="5">5</td><td></td></tr>
<tr><td colspan="2">2</td><td colspan="3">3</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td colspan="2">2</td><td>1+2</td></tr>
<tr><td colspan="2"></td><td></td><td>1</td><td>1</td><td>1</td></tr>
</table>

Number of turns: 4
Money earned: 10

<table>
<tr><td colspan="5">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="5">5</td><td></td></tr>
<tr><td colspan="4">4</td><td>1</td><td>4</td></tr>
<tr><td colspan="3">3</td><td>1</td><td></td><td>3</td></tr>
<tr><td colspan="2">2</td><td>1</td><td></td><td></td><td>2</td></tr>
<tr><td>1</td><td>1</td><td></td><td></td><td></td><td>1</td></tr>
</table>

Number of turns: 4
Money earned: 10

Six boxes:

<table>
<tr><td colspan="6">Box Breakdown</td><td>$$$ gained from breakdown</td></tr>
<tr><td colspan="6">6</td><td></td></tr>
<tr><td>1</td><td colspan="5">5</td><td>5</td></tr>
<tr><td></td><td>1</td><td colspan="4">4</td><td>4</td></tr>
<tr><td></td><td></td><td>1</td><td colspan="3">3</td><td>3</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td colspan="2">2</td><td>2</td></tr>
<tr><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td></tr>
</table>

Number of turns: 5
Money earned: 15

There are clear patterns emerging…

Table of values:

N	**Number of turns**	**Money earned**
4	3	6
5	4	10
6	5	15

DMC Problem 6.19(b)

(b) Make a conjecture for the number of turns you need. Prove it by strong induction.

P(n): For stacks of sizes from 1 to n, you need n-1 turns to separate the boxes into stacks of 1.

Base Case: P(1) – Takes zero turns because it is already a stack of 1. This fits P(n).

Induction Step: Assume P(1) and P(2) and P(3) up to P(n) is true and → P(n+1). I'll prove this via a direct proof.

A box of size n+1 can be broken into 2 stacks of n and 1. We know n satisfies P(n) because of the induction hypothesis. A stack of size N can be further broken down into a stack of n-1 and 1 boxes. Each of these stacks also fulfill P(n) because of the induction hypothesis. These breakdowns can be repeated until only stacks of size 1 are left and all stacks will satisfy P(n) because of the original strong induction.

Essentially, increasing the size of an initial stack by 1 just means that you have to spend 1 extra turn doing the breakdown which is a phenomena encapsulated by P(n).

DMC Problem 6.19(c)

(c) Make a conjecture for the maximum $ you can earn. Prove it by strong induction.

First I plugged in values found in part a of the question then filled out the rest of the table below.

N	Amount of money
1	0
2	1
3	3
4	6
5	10
6	15

I noticed the values fit within this summation formula: T(n) = $\sum_{i=1}^{n} i - 1$

Using the rules of summation, that can be rewritten as .5n(n+1) – x which simplifies down to .5n(n-1).

Thus T(n) = the max $ you can earn with a starting stack of n boxes = .5n(n-1)

Statement: P(n): T(n) is a correct amount for the total money to be earned for all values of 1 up to n.

I'll prove P(n) for all n from 1 to infinity.

Base case: T(0) = .5(1)(1-1) = 0. This checks out.

Induction step: Assume P(1) and P(2) and P(3) … P(n) are true and imply → P(n+1) – prove this via a direct proof.

T(n+1) can be written as .5(n+1)(n+1-1) which can be written as .5(n)(n+1). That can be written as the sum of T(n) and T(n-1) -- .5n(n-1) and .5(n-1)(n-2) respectively which we assume to be accurate equations because of the induction hypothesis.

Thus we have used true formulas to derive something that must therefore also be true.

By strong induction, P(n) is true for all n greater than 1.

DMC Problem 8.9(c)

In a tree (RBT or RFBT): a node is a leaf if both its children are empty; a node is half-full if one child is empty and the other is non-empty. A node is full if both children are non-empty. Let L be the number of leaves, H the number of half-full nodes, and F the number of full nodes. Let n = L + H + F be the total number of nodes.

(c) Prove by structural induction that in any RFBT, n = 2F + 1.

P(n): For any RFBT with n nodes, the number of nodes can also be written as 2F + 1.

Base Case: P(1) is true because a RFBT with a single node with no children satisfies the equation (2 * 0 + 1) = 1.

Induction Step: Assume P(1) and P(2) and P(3) … P(n) are true and imply P(n+1) is also true – I'll prove this with a direct proof.

The constructor for a RFBT takes two other RFBTs that are assumed to have the property and connects them to a new root – adding one node in the process. This new node will always be "full" because it's two children have to be the roots of other RFBTs.

The minimum difference in number of nodes between the new constructed tree and a "child tree" is 2 (the new root plus the smallest possible RFBT of size 1), which is why the equation is *2F* + 1.

Thus, the constructor preserves the property stated which means for any RFBT n = 2F + 1 by structural induction.

DMC Problem 9.15

Order these functions so that each function is in big-Oh of the next.

$n^{n^2}, (1.5)^n, \sqrt{n}, n^{100}, n!, 2^n, (\ln(n))^n, n^{\ln(n)}, \ln^3 n,\ n^2, nln(n), n^3, n^2 2^n, n^{2^n}, F^2_{[H_n]}, H_{F_n}$