#109. 2.5 For the follow-up question: The issue is that when the linked lists aren't the same length, the head of one linked list might represent the 1000's place while the other represents the 10's place. What if you made them the same length? Is there a way to modify the linked list to do that, without changing the value it represents?

#110. 1.6 Be careful that you aren't repeatedly concatenating strings together. This can be very inefficient.

#111. 2.7 If the two linked lists were the same length, you could traverse forward in each until you found an element in common. Now, how do you adjust this for lists of different lengths?

#112. 4.11 The reason that the earlier solution (picking a random number between 1 and 3) doesn't work is that the probabilities for the nodes won't be equal. For example, the root will be returned with probability $\frac{1}{3}$, even if there are 50+ nodes in the tree. Clearly, not all the nodes have probability $\frac{1}{3}$, so these nodes won't have equal probability. We can resolve this one issue by picking a random number between 1 and `size_of_tree` instead. This only resolves the issue for the root, though. What about the rest of the nodes?

#113. 4.5 Rather than validating the current node's value against `leftTree.max` and `rightTree.min`, can we flip around the logic? Validate the left tree's nodes to ensure that they are smaller than `current.value`.

#114. 3.4 We can remove the oldest item from a stack by repeatedly removing the newest item (inserting those into the temporary stack) until we get down to one element. Then, after we've retrieved the newest item, putting all the elements back. The issue with this is that doing several pops in a row will require `O(N)` work each time. Can we optimize for scenarios where we might do several pops in a row?

#115. 4.12 Once you've solidified the algorithm to find all contiguous subarrays in an array with a given sum, try to apply this to a tree. Remember that as you're traversing and modifying the hash table, you may need to "reverse the damage" to the hash table as you traverse back up.

#116. 4.2 Imagine we had a `createMinimalTree` method that returns a minimal tree for a given array (but for some strange reason doesn't operate on the root of the tree). Could you use this to operate on the root of the tree? Could you write the base case for the function? Great! Then that's basically the entire function.

#117. 1.1 Could a bit vector be useful?

#118. 1.3 You might find you need to know the number of spaces. Can you just count them?

#119. 4.11 The issue with the earlier solution is that there could be more nodes on one side of a node than the other. So, we need to weight the probability of going left and right based on the number of nodes on each side. How does this work, exactly? How can we know the number of nodes?

#120. 2.7 Try using the difference between the lengths of the two linked lists.

#121. 1.4 What characteristics would a string that is a permutation of a palindrome have?

#122. 1.2 Could a hash table be useful?

#123. 4.3 A hash table or array that maps from level number to nodes at that level might also be useful.

#124.	4.4	Actually, you can just have a single `checkHeight` function that does both the height computation and the balance check. An integer return value can be used to indicate both.
#125.	4.7	As a totally different approach: Consider doing a depth-first search starting from an arbitrary node. What is the relationship between this depth-first search and a valid build order?
#126.	2.2	Can you do it iteratively? Imagine if you had two pointers pointing to adjacent nodes and they were moving at the same speed through the linked list. When one hits the end of the linked list, where will the other be?
#127.	4.1	Two well-known algorithms can do this. What are the tradeoffs between them?
#128.	4.5	Think about the `checkBST` function as a recursive function that ensures each node is within an allowable `(min, max)` range. At first, this range is infinite. When we traverse to the left, the min is negative infinity and the max is `root.value`. Can you implement this recursive function and properly adjust these ranges as you traverse the tree?
#129.	2.7	If you move a pointer in the longer linked list forward by the difference in lengths, you can then apply a similar approach to the scenario when the linked lists are equal.
#130.	1.5	Can you do all three checks in a single pass?
#131.	1.2	Two strings that are permutations should have the same characters, but in different orders. Can you make the orders the same?
#132.	1.1	Can you solve it in `O(N log N)` time? What might a solution like that look like?
#133.	4.7	Pick an arbitrary node and do a depth-first search on it. Once we get to the end of a path, we know that this node can be the last one built, since no nodes depend on it. What does this mean about the nodes right before it?
#134.	1.4	Have you tried a hash table? You should be able to get this down to `O(N)` time.
#135.	4.3	You should be able to come up with an algorithm involving both depth-first search and breadth-first search.
#136.	1.4	Can you reduce the space usage by using a bit vector?

II

Hints for Concepts and Algorithms

#137.	5.1	Break this into parts. Focus first on clearing the appropriate bits.
#138.	8.9	Try the Base Case and Build approach.
#139.	6.9	Given a specific door x, on which rounds will it be toggled (open or closed)?
#140.	11.5	What does the interviewer mean by a pen? There are a lot of different types of pens. Make a list of potential questions you would want to ask.
#141.	7.11	This is not as complicated as it sounds. Start by making a list of the key objects in the system, then think about how they interact.
#142.	9.6	First, start with making some assumptions. What do and don't you have to build?
#143.	5.2	To wrap your head around the problem, try thinking about how you'd do it for integers.
#144.	8.6	Try the Base Case and Build approach.
#145.	5.7	Swapping each pair means moving the even bits to the left and the odd bits to the right. Can you break this problem into parts?
#146.	6.10	Solution 1: Start with a simple approach. Can you just divide up the bottles into groups? Remember that you can't re-use a test strip once it is positive, but you can reuse it as long as it's negative.
#147.	5.4	Get Next: Start with a brute force solution for each.
#148.	8.14	Can we just try all possibilities? What would this look like?
#149.	6.5	Play around with the jugs of water, pouring water back and forth, and see if you can measure anything other than 3 quarts or 5 quarts. That's a start.
#150.	8.7	Approach 1: Suppose you had all permutations of abc. How can you use that to get all permutations of abcd?
#151.	5.5	Reverse engineer this, starting from the outermost layer to the innermost layer.
#152.	8.1	Approach this from the top down. What is the very last hop the child made?
#153.	7.1	Note that a "card deck" is very broad. You might want to think about a reasonable scope to the problem.
#154.	6.7	Observe that each family will have exactly one girl.
#155.	8.13	Will sorting the boxes help in any way?

#156.	6.8	This is really an algorithm problem, and you should approach it as such. Come up with a brute force, compute the worst-case number of drops, then try to optimize that.
#157.	6.4	In what cases will they not collide?
#158.	9.6	We've assumed that the rest of the eCommerce system is already handled, and we just need to deal with the analytics part of sales rank. We can get notified somehow when a purchase occurs.
#159.	5.3	Start with a brute force solution. Can you try all possibilities?
#160.	6.7	Think about writing each family as a sequence of Bs and Gs.
#161.	8.8	You could handle this by just checking to see if there are duplicates before printing them (or adding them to a list). You can do this with a hash table. In what case might this be okay? In what case might it not be a very good solution?
#162.	9.7	Will this application be write-heavy or read-heavy?
#163.	6.10	Solution 1: There is a relatively simple approach that works in 28 days, in the worst case. There are better approaches though.
#164.	11.5	Consider the scenario of a pen for children. What does this mean? What are the different use cases?
#165.	9.8	Scope the problem well. What will and won't you tackle as part of this system?
#166.	8.5	Think about multiplying 8 by 9 as counting the number of cells in a matrix with width 8 and height 9.
#167.	5.2	In a number like `.893` (in base 10), what does each digit signify? What then does each digit in `.10010` signify in base 2?
#168.	8.14	We can think about each possibility as each place where we can put parentheses. This means around each operator, such that the expression is split at the operator. What is the base case?
#169.	5.1	To clear the bits, create a "bit mask" that looks like a series of 1s, then 0s, then 1s.
#170.	8.3	Start with a brute force algorithm.
#171.	6.7	You can attempt this mathematically, although the math is pretty difficult. You might find it easier to estimate it up to families of, say, 6 children. This won't give you a good mathematical proof, but it might point you in the right direction of what the answer might be.
#172.	6.9	In which cases would a door be left open at the end of the process?
#173.	5.2	A number such as `.893` (in base 10) indicates $8 * 10^{-1} + 9 * 10^{-2} + 3 * 10^{-3}$. Translate this system into base 2.
#174.	8.9	Suppose we had all valid ways of writing two pairs of parentheses. How could we use this to get all valid ways of writing three pairs?
#175.	5.4	Get Next: Picture a binary number—something with a bunch of 1s and 0s spread out throughout the number. Suppose you flip a 1 to a 0 and a 0 to a 1. In what case will the number get bigger? In what case will it get smaller?

#176. 9.6 Think about what sort of expectations on freshness and accuracy of data is expected. Does the data always need to be 100% up to date? Is the accuracy of some products more important than others?

#177. 10.2 How do you check if two words are anagrams of each other? Think about what the definition of "anagram" is. Explain it in your own words.

#178. 8.1 If we knew the number of paths to each of the steps before step 100, could we compute the number of steps to 100?

#179. 7.8 Should white pieces and black pieces be the same class? What are the pros and cons of this?

#180. 9.7 Observe that there is a lot of data coming in, but people probably aren't reading the data very frequently.

#181. 6.2 Calculate the probability of winning the first game and winning the second game, then compare them.

#182. 10.2 Two words are anagrams if they contain the same characters but in different orders. How can you put characters in order?

#183. 6.10 Solution 2: Why do we have such a time lag between tests and results? There's a reason the question isn't phrased as just "minimize the number of rounds of testing." The time lag is there for a reason.

#184. 9.8 How evenly do you think traffic is distributed? Do all documents get roughly the same age of traffic? Or is it likely there are some very popular documents?

#185. 8.7 Approach 1: The permutations of `abc` represent all ways of ordering `abc`. Now, we want to create all orderings of `abcd`. Take a specific ordering of `abcd`, such as `bdca`. This `bdca` string represents an ordering of `abc`, too: Remove the `d` and you get `bca`. Given the string `bca`, can you create all the "related" orderings that include `d`, too?

#186. 6.1 You can only use the scale once. This means that all, or almost all, of the bottles must be used. They also must be handled in different ways or else you couldn't distinguish between them.

#187. 8.9 We could try generating the solution for three pairs by taking the list of two pairs of parentheses and adding a third pair. We'd have to add the third paren before, around, and after. That is: `()<SOLUTION>`, `(<SOLUTION>)`, `<SOLUTION>()`. Will this work?

#188. 6.7 Logic might be easier than math. Imagine we wrote every birth into a giant string of Bs and Gs. Note that the groupings of families are irrelevant for this problem. What is the probability of the next character added to the string being a B versus a G?

#189. 9.6 Purchases will occur very frequently. You probably want to limit database writes.

#190. 8.8 If you haven't solved 8.7 yet, do that one first.

#191. 6.10 Solution 2: Consider running multiple tests at once.

#192. 7.6 A common trick when solving a jigsaw puzzle is to separate edge and non-edge pieces. How will you represent this in an object-oriented manner?

#193. 10.9 Start with a naive solution. (But hopefully not too naive. You should be able to use the fact that the matrix is sorted.)

#194.	8.13	We can sort the boxes by any dimension in descending order. This will give us a partial order for the boxes, in that boxes later in the array must appear before boxes earlier in the array.
#195.	6.4	The only way they won't collide is if all three are walking in the same direction. What's the probability of all three walking clockwise?
#196.	10.11	Imagine the array were sorted in ascending order. Is there any way you could "fix it" to be sorted into alternating peaks and valleys?
#197.	8.14	The base case is when we have a single value, 1 or 0.
#198.	7.3	Scope the problem first and make a list of your assumptions. It's often okay to make reasonable assumptions, but you need to make them explicit.
#199.	9.7	The system will be write-heavy: Lots of data being imported, but it's rarely being read.
#200.	8.7	Approach 1: Given a string such as `bca`, you can create all permutations of `abcd` that have `{a, b, c}` in the order `bca` by inserting `d` into each possible location: `dbca`, `bdca`, `bcda`, `bcad`. Given all permutations of `abc`, can you then create all permutations of `abcd`?
#201.	6.7	Observe that biology hasn't changed; only the conditions under which a family stops having kids has changed. Each pregnancy has a 50% odds of being a boy and a 50% odds of being a girl.
#202.	5.5	What does it mean if `A & B == 0`?
#203.	8.5	If you wanted to count the cells in an 8x9 matrix, you could count the cells in a 4x9 matrix and then double it.
#204.	8.3	Your brute force algorithm probably ran in `O(N)` time. If you're trying to beat that runtime, what runtime do you think you will get to? What sorts of algorithms have that runtime?
#205.	6.10	Solution 2: Think about trying to figure out the bottle, digit by digit. How can you detect the first digit in the poisoned bottle? What about the second digit? The third digit?
#206.	9.8	How will you handle generating URLs?
#207.	10.6	Think about merge sort versus quick sort. Would one of them work well for this purpose?
#208.	9.6	You also want to limit joins because they can be very expensive.
#209.	8.9	The problem with the solution suggested by the earlier hint is that it might have duplicate values. We could eliminate this by using a hash table.
#210.	11.6	Be careful about your assumptions. Who are the users? Where are they using this? It might seem obvious, but the real answer might be different.
#211.	10.9	We can do a binary search in each row. How long will this take? How can we do better?
#212.	9.7	Think about things like how you're going to get the bank data (will it be pulled or pushed?), what features the system will support, etc.
#213.	7.7	As always, scope the problem. Are "friendships" mutual? Do status messages exist? Do you support group chat?
#214.	8.13	Try to break it down into subproblems.

#215.	5.1	It's easy to create a bit mask of 0s at the beginning or end. But how do you create a bit mask with a bunch of zeroes in the middle? Do it the easy way: Create a bit mask for the left side and then another one for the right side. Then you can merge those.
#216.	7.11	What is the relationship between files and directories?
#217.	8.1	We can compute the number of steps to 100 by the number of steps to 99, 98, and 97. This corresponds to the child hopping 1, 2, or 3 steps at the end. Do we add those or multiply them? That is: Is it `f(100) = f(99) + f(98) + f(97)` *or* `f(100) = f(99) * f(98) * f(97)`?
#218.	6.6	This is a logic problem, not a clever word problem. Use logic/math/algorithms to solve it.
#219.	10.11	Try walking through a sorted array. Can you just swap elements until you have fixed the array?
#220.	11.5	Have you considered both intended uses (writing, etc.) and unintended use? What about safety? You would not want a pen for children to be dangerous.
#221.	6.10	Solution 2: Be very careful about edge cases. What if the third digit in the bottle number matches the first or second digit?
#222.	8.8	Try getting the count of each character. For example, ABCAAC has 3 As, 2 Cs, and 1 B.
#223.	9.6	Don't forget that a product can be listed under multiple categories.
#224.	8.6	You can easily move the smallest disk from one tower to another. It's also pretty easy to move the smallest two disks from one tower to another. Can you move the smallest three disks?
#225.	11.6	In a real interview, you would also want to discuss what sorts of test tools we have available.
#226.	5.3	Flipping a 0 to a 1 can merge two sequences of 1s—but only if the two sequences are separated by only one 0.
#227.	8.5	Think about how you might handle this for odd numbers.
#228.	7.8	What class should maintain the score?
#229.	10.9	If you're considering a particular column, is there a way to quickly eliminate it (in some cases at least)?
#230.	6.10	Solution 2: You can run an additional day of testing to check digit 3 in a different way. But again, be very careful about edge cases here.
#231.	10.11	Note that if you ensure the peaks are in place, the valleys will be, too. Therefore, your iteration to fix the array can skip over every other element.
#232.	9.8	If you generate URLs randomly, do you need to worry about collisions (two documents with the same URL)? If so, how can you handle this?
#233.	6.8	As a first approach, you might try something like binary search. Drop it from the 50th floor, then the 75th, then the 88th, and so on. The problem is that if the first egg drops at the 50th floor, then you'll need to start dropping the second egg starting from the 1st floor and going up. This could take, at worst, 50 drops (the 50th floor drop, the 1st floor drop, the 2nd floor drop, and up through the 49th floor drop). Can you beat this?

#234.	8.5	If there's duplicated work across different recursive calls, can you cache it?
#235.	10.7	Would a bit vector help?
#236.	9.6	Where would it be appropriate to cache data or queue up tasks?
#237.	8.1	We multiply the values when it's "we do this then this." We add them when it's "we do this or this."
#238.	7.6	Think about how you might record the position of a piece when you find it. Should it be stored by row and location?
#239.	6.2	To calculate the probability of winning the second game, start with calculating the probability of making the first hoop, the second hoop, and not the third hoop.
#240.	8.3	Can you solve the problem in O(log N)?
#241.	6.10	Solution 3: Think about each test strip as being a binary indicator for poisoned vs. non-poisoned.
#242.	5.4	Get Next: If you flip a 1 to a 0 and a 0 to a 1, it will get bigger if the 0->1 bit is more significant than the 1->0 bit. How can you use this to create the next biggest number (with the same number of 1s)?
#243.	8.9	Alternatively, we could think about doing this by moving through the string and adding left and right parens at each step. Will this eliminate duplicates? How do we know if we can add a left or right paren?
#244.	9.6	Depending on what assumptions you made, you might even be able to do without a database at all. What would this mean? Would it be a good idea?
#245.	7.7	This is a good problem to think about the major system components or technologies that would be useful.
#246.	8.5	If you're doing `9*7` (both odd numbers), then you could do 4*7 and 5*7.
#247.	9.7	Try to reduce unnecessary database queries. If you don't need to permanently store the data in the database, you might not need it in the database at all.
#248.	5.7	Can you create a number that represents just the even bits? Then can you shift the even bits over by one?
#249.	6.10	Solution 3: If each test strip is a binary indicator, can we map , integer keys to a set of 10 binary indicators such that each key has a unique configuration (mapping)?
#250.	8.6	Think about moving the smallest disk from tower `X=0` to tower `Y=2` using tower `Z=1` as a temporary holding spot as having a solution for `f(1, X=0, Y=2, Z=1)`. Moving the smallest two disks is `f(2, X=0, Y=2, Z=1)`. Given that you have a solution for `f(1, X=0, Y=2, Z=1)` and `f(2, X=0, Y=2, Z=1)`, can you solve `f(3, X=0, Y=2, Z=1)`?
#251.	10.9	Since each column is sorted, you know that the value can't be in this column if it's smaller than the min value in this column. What else does this tell you?
#252.	6.1	What happens if you put one pill from each bottle on the scale? What if you put two pills from each bottle on the scale?
#253.	10.11	Do you necessarily need the arrays to be sorted? Can you do it with an unsorted array?

#254.	10.7	To do it with less memory, can you try multiple passes?
#255.	8.8	To get all permutations with 3 As, 2 Cs, and 1 B, you need to first pick a starting character: A, B, or C. If it's an A, then you need all permutations with 2 As, 2 Cs, and 1 B.
#256.	10.5	Try modifying binary search to handle this.
#257.	11.1	There are two mistakes in this code.
#258.	7.4	Does the parking lot have multiple levels? What "features" does it support? Is it paid? What types of vehicles?
#259.	9.5	You may need to make some assumptions (in part because you don't have an interviewer here). That's okay. Make those assumptions explicit.
#260.	8.13	Think about the first decision you have to make. The first decision is which box will be at the bottom.
#261.	5.5	If `A & B == 0`, then it means that A and B never have a 1 at the same spot. Apply this to the equation in the problem.
#262.	8.1	What is the runtime of this method? Think carefully. Can you optimize it?
#263.	10.2	Can you leverage a standard sorting algorithm?
#264.	6.9	Note: If an integer `x` is divisible by `a`, and `b = x / a`, then `x` is also divisible by `b`. Does this mean that all numbers have an even number of factors?
#265.	8.9	Adding a left or right paren at each step will eliminate duplicates. Each substring will be unique at each step. Therefore, the total string will be unique.
#266.	10.9	If the value `x` is smaller than the start of the column, then it also can't be in any columns to the right.
#267.	8.7	Approach 1: You can create all permutations of `abcd` by computing all permutations of `abc` and then inserting `d` into each possible location within those.
#268.	11.6	What are the different features and uses we would want to test?
#269.	5.2	How would you get the first digit in `.893`? If you multiplied by 10, you'd shift the values over to get `8.93`. What happens if you multiply by 2?
#270.	9.2	To find the connection between two nodes, would it be better to do a breadth-first search or depth-first search? Why?
#271.	7.7	How will you know if a user signs offline?
#272.	8.6	Observe that it doesn't really matter which tower is the source, destination, or buffer. You can do `f(3, X=0, Y=2, Z=1)` by first doing `f(2, X=0, Y=1, Z=2)` (moving two disks from tower 0 to tower 1, using tower 2 as a buffer), then moving disk 3 from tower 0 to tower 2, then doing `f(2, X=1, Y=2, Z=0)` (moving two disks from tower 1 to tower 2, using tower 0 as a buffer). How does this process repeat?
#273.	8.4	How can you build all subsets of `{a, b, c}` from the subsets of `{a, b}`?
#274.	9.5	Think about how you could design this for a single machine. Would you want a hash table? How would that work?
#275.	7.1	How, if at all, will you handle aces?

#276.	9.7	As much work as possible should be done asynchronously.
#277.	10.11	Suppose you had a sequence of three elements (`{0, 1, 2}`, in any order. Write out all possible sequences for those elements and how you can fix them to make `1` the peak.
#278.	8.7	Approach 2: If you had all permutations of two-character substrings, could you generate all permutations of three-character substrings?
#279.	10.9	Think about the previous hint in the context of rows.
#280.	8.5	Alternatively, if you're doing 9 * 7, you could do 4*7, double that, and then add 7.
#281.	10.7	Try using one pass to get it down to a range of values, and then a second pass to find a specific value.
#282.	6.6	Suppose there were exactly one blue-eyed person. What would that person see? When would they leave?
#283.	7.6	Which will be the easiest pieces to match first? Can you start with those? Which will be the next easiest, once you've nailed those down?
#284.	6.2	If two events are mutually exclusive (they can never occur simultaneously), you can add their probabilities together. Can you find a set of mutually exclusive events that represent making two out of three hoops?
#285.	9.2	A breadth-first search is probably better. A depth-first search can wind up going on a long path, even though the shortest path is actually very short. Is there a modification to a breadth-first search that might be even faster?
#286.	8.3	Binary search has a runtime of `O(log N)`. Can you apply a form of binary search to the problem?
#287.	7.12	In order to handle collisions, the hash table should be an array of linked lists.
#288.	10.9	What would happen if we tried to keep track of this using an array? What are the pros and cons of this?
#289.	10.8	Can you use a bit vector?
#290.	8.4	Anything that is a subset of `{a, b}` is also a subset of `{a, b, c}`. Which sets are subsets of `{a, b, c}` but not `{a, b}`?
#291.	10.9	Can we use the previous hints to move up, down, left, and right around the rows and columns?
#292.	10.11	Revisit the set of sequences for `{0, 1, 2}` that you just wrote out. Imagine there are elements before the leftmost element. Are you sure that the way you swap the elements won't invalidate the previous part of the array?
#293.	9.5	Can you combine a hash table and a linked list to get the best of both worlds?
#294.	6.8	It's actually better for the first drop to be a bit lower. For example, you could drop at the 10th floor, then the 20th floor, then the 30th floor, and so on. The worst case here will be 19 drops (10, 20, ..., 100, 91, 92, ..., 99). Can you beat that? Try not randomly guessing at different solutions. Rather, think deeper. How is the worst case defined? How does the number of drops of each egg factor into that?