Hacking a Google Interview Practice Questions – Person B

Question: Binary Search Tree Validity

Write a function to determine whether a given binary tree of distinct integers is a valid binary search tree. Assume that each node contains a pointer to its left child, a pointer to its right child, and an integer, but not a pointer to its parent. You may use any language you like.

Good Answer: Note that it's not enough to write a recursive function that just checks if the left and right nodes of each node are less than and greater than the current node (and calls that recursively). You need to make sure that all the nodes of the subtree starting at your current node are within the valid range of values allowed by the current node's ancestors. Therefore you can solve this recursively by writing a helper function that accepts a current node, the smallest allowed value, and the largest allowed value for that subtree. An example of this is the following (in Java):

```
boolean isValid(Node root) {
    return isValidHelper(root, Integer.MIN_VALUE,
                         Integer.MAX_VALUE);
}

boolean isValidHelper(Node curr, int min, int max) {
    if (curr.left != null) {
        if (curr.left.value < min ||
                !isValidHelper(curr.left, min, curr.value))
            return false;
    }
    if (curr.right != null) {
        if (curr.right.value > max ||
                !isValidHelper(curr.right, curr.value, max))
            return false;
    }
    return true;
}
```

The running time of this algorithm is O(n).

Question: Odd Man Out

You're given an unsorted array of integers where every integer appears exactly twice, except for one integer which appears only once. Write an algorithm (in a language of your choice) that finds the integer that appears only once.

Good Answer: Set up a hash set that we will put the integers from the array into. Have a second variable that will keep a sum. Start going through the array and for each integer, check to see if it's already in the hash set. If it is not, add that integer to the sum and store that integer in the hash set. If it is in the hash set, subtract that integer from the sum. When the algorithm finishes going through the array, the sum variable should be equal to the integer we were looking for, since it is the only number we never subtracted from the sum. This takes O(n) time and O(n) space.

```
int oddManOut(int[] array) {
    HashSet<Integer> s = new HashSet<Integer>();
    int sum = 0;
    for (int i = 0; i < array.length; i++) {
        if (s.contains(array[i])) {
            sum = sum - array[i];
        } else {
            s.add(array[i]);
            sum = sum + array[i];
        }
    }
    return sum;
}
```

Really Awesome Answer: XOR all the values of the array together! Since XOR is commutative and is its own inverse, each integer in the array that appears twice will cancel itself out, and we'll be left with the integer we're looking for. This takes O(n) time and O(1) space. We told you bitwise stuff was handy!

```
int oddManOut(int[] array) {
    int val = 0;
    for (int i = 0; i < array.length; i++) {
        val ^= array[i];
    }
    return val;
}
```

Question: Design a Poker Game

(Don't ask all these questions at the same time; ask one after another, since they build upon each other.) Without writing any actual code, describe as much as possible how you would design a poker game program. What classes would you have? What relationships would they have with each other? What would be the basic flow of the program and how would those classes play a part? If you then wanted to add a new type of poker game (such as Texas Hold 'em), how would that fit into your design?

Answer: There are so many possible answers to this problem that it would be difficult to say that one answer is the best. Look to make sure that they make classes to simulate the basic parts of a poker game (perhaps a hand, the pot, a game type or rules, a round, the deck, etc.). Using inheritance (subclassing in object-oriented programming) where it makes sense is also good for reusability and extendibility. Using design patters (such as Model-View-Controller, Listener/Observer, or the Singleton pattern) is also a good thing. The main point is for them to get used to thinking about how they would design a system. Most importantly, they need to think about simplicity, reusability, and extendibility in their design.

Question: Leader Election

Describe a technique to identify a "leader" among a group of 10 identical servers that are all connected to every other server. There are no prior distinguishing characteristics of any of them and the same program to identify the leader starts running on all of them at the same time. After an answer is given, ask how much network traffic it requires and, if "ties" are possible, ask how you can break ties.

Good Answer: Have each server wait a random amount of time and then say "I'm it." The "I'm it" announcement is time-stamped, and the computer that time-stamped its announcement first is elected the leader. This approach requires sending out 9 messages. If there is a tie, the computers can repeat the procedure.

Note that other answers are possible, but every correct answer will use randomness in some way.

Question: Queue Using Stacks

Describe a queue data structure that is implemented using one or more stacks. Don't worry about running time. Write the enqueue and dequeue operations for the queue. You may use any language you like.

Good answer: You can use two stacks: an "incoming" stack and an "outgoing" stack. The enqueue and dequeue operations would look like this (in Java):

```
Stack in;
Stack out;

void enqueue(int value) {
    while (!out.isEmpty())
        in.push(out.pop());
    in.push(value);
}
```

```
int dequeue() {
    while (!in.isEmpty())
        out.push(in.pop());
    return out.pop();
}
```

Question: Instant Messaging

Describe a design for an instant messaging program where there are several servers, clients are connected to each server, and the servers communicate with each other. Describe the classes, interfaces, and so on that you would use and how you would organize them.

Answer: As in the previous design questions, there is no best answer. Good topics to discuss are how each client communicates with a server, how the servers maintain state with the other servers, how state information is communicated between servers and clients, and the speed/reliability of their design.

Question: Maximal Subarray

Given an array, describe an algorithm to identify the subarray with the maximum sum. For example, if the input is [1, -3, 5, -2, 9, -8, -6, 4], the output would be [5, -2, 9].

Good Answer: Observe that the sum of a subarray from element i to element j is equal to the sum of the subarray from element 1 to element j minus the subarray from element 1 to element i - 1. Our algorithm will iterate through the array. The algorithm keeps track of the sum x of the elements no later than the element. It will also keep track of the minimum sum y of the subarray from the first element to an element no later than the current element. Finally, It will also keep track of the subarray z with the maximum sum so far. At each step, we update x by adding the current element to it. We update y by checking whether x < y; if so, we set y to be x. We update z by checking whether y - x is greater than z; if so, we set z to be y - x.

For example, with the sample input, our algorithm would do the following:

Current element	x	y	z
1	1	0	1
-3	-2	-2	0
5	3	-2	5
-2	1	-2	5
9	10	-2	12
-8	2	-2	12
-6	-4	-4	12
4	0	-4	12

Surprisingly, this problem is equivalent to the stock market problem described in handout 3. Given an array a1, you can "convert" it to an array a2 for the stock market problem by setting each element a2[i] to be a1[0] + a1[1] + ... + a1[i].

Question: Obstacle Avoidance

Given an n x n grid with a person and obstacles, how would you find a path for the person to a particular destination? The person is permitted to move left, right, up, and down.

Good Answer: Use the A* algorithm or another fast path-finding algorithm. (It is described on Wikipedia.)