Ch.1

1.1Suppose you have a sorted list of 128 names, and you're searching through it using binary search. What's the maximum number of steps it would take?

Log2(128)=7

1.2Suppose you double the size of the list. What's the maximum number of steps now? Give the run time for each of these scenarios in terms of Big O

Log2(128*2)=8

1.3 You have a name, and you want to find the person's phone number in the phone book.

O(log(n))

1.4 You have a phone number, and you want to find the person's name in the phone book. (Hint: You'll have to search through the whole book!)

O(n)

1.5 You want to read the numbers of every person in the phone book.

O(n)

1.6 You want to read the numbers of just the As. (This is a tricky one! It involves concepts that are covered more in chapter 4. Read the answer—you may be surprised!)

O(n)

Ch.2

2.1 Suppose you're building an app to keep track of your finances. Every day, you write down everything you spent money on. At the end of the month, you review your expenses and sum up how much you spent. So, you have lots of inserts and a few reads. Should you use an array or a list?

Linked lists (because we have lots of inserts and a few reads.)

2.2 Suppose you're building an app for restaurants to take customer orders. Your app needs to store a list of orders. Servers keep adding orders to this list, and chefs take orders off the list and make them. It's an order queue: servers add orders to the back of the queue, and the chef takes the first order off the queue and cooks it. Would you use an array or a linked list to implement this queue? (Hint: linked lists are good for inserts/deletes, and arrays are good for random access. Which one are you going to be doing here?)

Linked lists (because we have lots of inserts and a few reads.)

2.3 Let's run a thought experiment. Suppose Facebook keeps a list of usernames. When someone tries to log in to Facebook, a search is done for their username. If their name is in the list of usernames, they can log in. People log in to Facebook pretty often, so there are a lot of searches through this list of usernames. Suppose Facebook uses binary search to search the list. Binary search needs random access—you need to be able to get to the middle of the list of usernames instantly. Knowing this, would you implement the list as an array or a linked list?

Arrays (random access)

2.4 People sign up for Facebook pretty often, too. Suppose you decided to use an array to store the list of users. What are the downsides of an array for inserts? In particular, suppose you're using binary search to search for logins. What happens when you add new users to an array?

Inserting into arrays is slow.

if we use binary search to search for usernames, the array needs to be sorted. Suppose someone named Adit B signs up for Facebook. Their name will be inserted at the end of the array. So you need to sort the array every time a name is inserted!

2.5 In reality, Facebook uses neither an array nor a linked list to store user information. Let's consider a hybrid data structure: an array of linked lists. You have an array with 26 slots. Each slot points to a linked list. For example, the first slot in the array points to a linked list containing all the usernames starting with a. The second slot points to a linked list containing all the usernames starting with b, and so on. Suppose Adit B signs up for Facebook and you want to add them to the list. You go to slot 1 in the array, go to the linked list for slot 1, and add Adit B at the end. Now, suppose you want to search for Zakhir H.

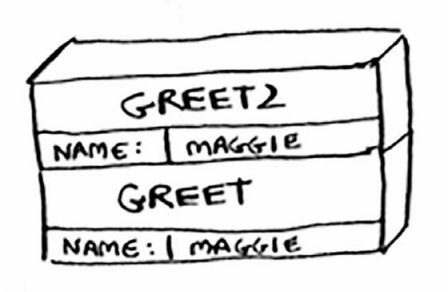
You go to slot 26, which points to a linked list of all the Z names. Then you search through that list to find Zakhir H. Compare this hybrid data structure to arrays and linked lists. Is it slower or faster than each for searching and inserting? You don't have to give Big O run times, just whether the new data structure would be faster or slower.

Searching—slower than arrays, faster than linked lists.

Inserting—faster than arrays, slower than linked lists.

Ch.3

3.1 Suppose I show you a call stack like this.



What information can you give me, just based on this call stack?

Now let's see the call stack in action with a recursive function.

1)The greet function is called first, with name = maggie.

2) Then the greet function calls the greet2 function, with name = maggie.

3)At this point, the greet function is in an incomplete, suspended state.

4)The current function call is the greet2 function.

5)After this function call completes, the greet function will resume.

3.2 Suppose you accidentally write a recursive function that runs forever. As you saw, your computer allocates memory on the stack for each function call. What happens to the stack when your recursive function runs forever?

stack overflow error.