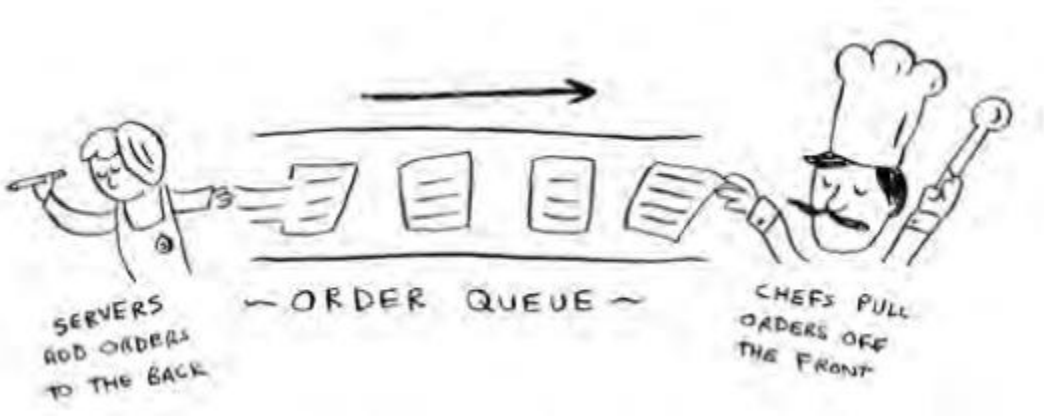
2.1 Suppose you're building an app to keep track of your finances. Every day, you write down everything you spent money on. At the end of the month, you review your expenses and sum up how much you spent. So, you have lots of inserts and a few reads. Should you use an array or a list?

Answer: In this case, you're adding expenses to the list every day and reading all the expenses once a month. Arrays have fast reads and slow inserts. Linked lists have slow reads and fast inserts. Because you'll be inserting more often than reading, it makes sense to use a linked list. Also, linked lists have slow reads only if you're accessing random elements in the list. Because you're reading every element in the list, linked lists will do well on reads too. So a linked list is a good solution to this problem.

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?)

Answer: A linked list. Lots of inserts are happening (servers adding orders), which linked lists excel at. You don't need search or random access (what arrays excel at), because the chefs always take the first order off the queue.

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

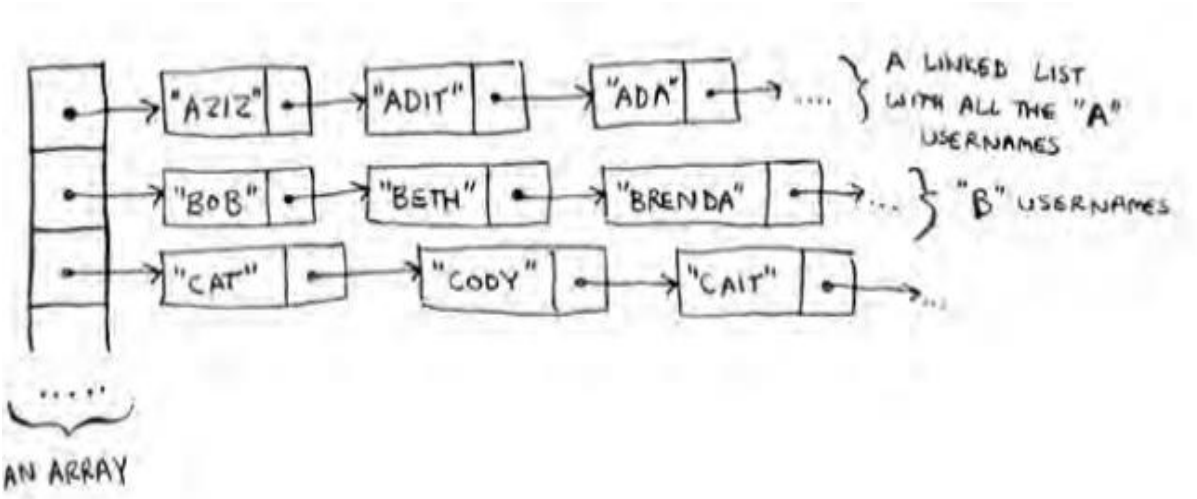
name is in the list of usernames, they can log in. People log in to Facebook pretty often, so there are a lot of searches through this list of usernames. Suppose Facebook uses binary search to search the list. Binary search needs random access—you need to be able to get to the middle of the list of usernames instantly. Knowing this, would you implement the list as an array or a linked list?

Answer: A sorted array. Arrays give you random access—you can get an element from the middle of the array instantly. You can't do that with linked lists. To get to the middle element in a linked list, you'd have to start at the first element and follow all the links down to the middle element.

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?

Answer: Inserting into arrays is slow. Also, if you're using 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.

Answer: Searching—slower than arrays, faster than linked lists. Inserting—faster than arrays, same amount of time as linked lists. So it's slower for searching than an array, but faster or the same as linked lists for everything. We'll talk about another hybrid data structure called a hash table later in the book. This should give you an idea of how you can build up more complex data structures from simple ones. So what does Facebook really use? It probably uses a dozen different databases, with different data structures behind them: hash tables, B-trees, and others. Arrays and linked lists

are the building blocks for these more complex data structures.