Privacy, Security

This is CS50. Harvard University. Fall 2015.

Cheng Gong

Table of Contents

1. News

- This week, Oculus Rift, a virtual reality device, has started shipping. It works by using screens distorted by lenses to give the illusion of being in some virtual 3D space.
- Augmented reality is where the glasses are transparent but there is an overlay of objects or text over the environment actually around us.

2. Data Structures, Continued

- Last time we talked about arrays, continuous areas of memory, which have the advantage of allowing us to jump between parts easily and quickly, since we know the range of addresses they take up.
  - For example, if we have an array taking up locations 0 through 8 inclusive, we know that the middle element will be in location 4, with 4 elements to the left and 4 elements to the right.
- We also talked about linked lists, where elements are stored in different locations in memory but connected by saving each address to the next element.
- One tradeoff here is that we know exactly how big an array is, since we know how much memory was set aside, but as a linked list grows and shrinks we might have to follow the addresses one by one, if we don't keep track of how big it is each time we change it. We also can't jump to elements in a linked list without following the addresses each time.

- We could also copy the linked list to an array and be able to access it quickly, but now we need to take time to make that copy, as well as use extra memory.
- The idea running time for storing data would be #(1), where it takes a constant number of steps to access any element.
- Imagine we had a phone book in memory as an array with sections for each letter from A to Z. If we wanted to put names of people into our phone book, we'd write "Alice" in the section labeled A, "Bob" in the section labeled B, and "Charlie" in the section labeled C. But say we have someone else with the name "Bill", but "Bob" is already using the section in the array labeled B. And we can't increase the size of that section since it's a fixed size, so we'd want to combine the concept of an array and a linked list in a data structure called a hash table. We'd allocate some other area of memory to store the name "Bill", and link that to Bob's section in our array. And we can continue this process for any additional names. So each element in our array is actually the first item in a linked list, which we can follow to find other elements under that letter or label.
- The running time to find a name here would be #(*n*), since in the worst case everyone's names might start with the same letter, so the hash table would just be one long linked list. But if our elements were distributed more evenly, finding a name would take #(*n*/26), assuming we had 26 letters or linked lists. And though yesterday we said that we throw away constants when considering theoretical running time, in practice those constants do matter. For example, if we had many more sections, or buckets, in our hash table, then our linked lists would be much shorter, and we might get close to #(1) for access time if there were only one element in each list. And to spread out our names, we might look at not just the first letter of each name, but the first too. So perhaps "Aaron" would be in bucket 0, "Ab" would be in bucket 1, etc. And we can even use the fact that ASCII is a representation of numbers as letters, to use the numbers to create even larger values to spread out our names. This technique is generally called hashing, where we use a hash function that takes a value, like a name, as input and returns a number between some range, which we can use to store our inputs in some hash table. And later when we want to access our data, we use our hash function again to determine which bucket we should look in.
- We might be tempted to have a really large hash table and then have faster lookup times, but that comes at the cost of potentially wasted space if some buckets end up being completely empty. And an ideal hash function should distribute items evenly into buckets, so we minimize wasted space as well as maximum access time.

- There's another common data structure called a tree[1], where each of the circles are called a node, and circles below them are its children.
- We can add a further restriction to the tree, where we say that the child on the left side of a node has a lower value, and the child on the right has a higher value. So now we can easily do a binary search for a value by following the tree, even though a tree can be implemented with a linked list of flexible size, rather than an array. If we say that each node in the tree has two children (which means it is a binary search tree), then the height of the tree will be log(n), which means that finding a value would take at most log(n) steps. But the cost would be in inserting new values, since we might need to rebalance the tree.
- We take a look at this video[2] comparing sorting algorithms.

3. More News

- In the recent Apple vs. FBI case, FBI was asking Apple to help it unlock a phone it needed to search, but did not have the ability to, because it was encrypted.
- They could have tried every value for a 4- or 6-digit code, but because the phone increases the amount of time it takes to try a value, that method was not feasible.
- And there's also the feature for phones to have the ability to wipe themselves if enough incorrect codes are attempted.
- We watch this video[3] whereby a brute-force attack, trying every combination, might be implemented.
- So the FBI wanted Apple to remove the delay in checking codes, the ability to guess the code with software rather than manually, and the ability to load this operating system onto the phone.
- The phone probably has automatic updates enabled still, so if the update servers only targeted this phone, Apple might be able to install the modified software even if it's locked.
- In addition, a relatively new feature of Macs and iOS allows only "signed" software, or software that has been verified with a special key, to be installed by the current operating system, which is meant to increase security.

[1] https://en.wikipedia.org/wiki/Tree_(data_structure)
[2] https://www.youtube.com/watch?v=ZZuD6iUe3Pc
[3] https://www.youtube.com/watch?v=9R_D-zX3yP8

- And finally, this entire case has been viewed as a first attempt to get broader access to users' devices and data.
- An analogy for this problem might be that certain luggage locks are TSA-approved, since there is a master key that opens them, to allow for greater airport security. But this comes at a downside where copies of these master keys are sold on eBay or elsewhere, and allows for someone unauthorized to get into luggage that has been locked with those locks. So by weakening the security on their devices for the FBI, Apple also weakens the security for everyone else.
- More sophisticated adversaries, too, will continue to use other methods to encrypt their data, so we might not even gain much benefit from this.
- From a more technical standpoint, encrypted data means that it has been scrambled. For example, the letters "ifmmp" don’t seem to make much sense, but if we know that if this encrypted with a rotation of the alphabet forward by one letter, we can go backwards to figure out that the message says "hello". A really old version of this is called ROT13[4] where the message is rotated by 13 letters.
- These days, encryption algorithms use large random numbers to make our messages much more secure. There are generally two classes of these algorithms: secret-key cryptography and public-key cryptography. Secret-key is like ROT13 where there is a secret value (13 in this case) where we encrypt and decrypt our data with the same key. But two people sending messages have to both know the key in advance.
- To avoid this problem, public-key cryptography allows for encrypting messages with one key that everyone knows (a public key), but that message can only be decrypted by a matching private key, which only the recipient should know. Mathematically, these keys (numbers) appear random and are large enough to be difficult to guess, but have relationships or properties that allow for this process.
- When we go to a secure website, our browser sends the website we’re visiting our public key, and ask for their public key, and that allows for a secure connection to be established.
- And most protocols actually use public-key cryptography to set up a secret key for temporary use, since algorithms that encrypt data with just one secret key are generally faster.
- A private key for the RSA encryption algorithm might look like this:

[4] http://rot13.com/

  # And these characters shown as the key actually map to numbers, so we really just have a really large number.

- An 8-bit key allows for 256 options, so we would need at most 256 guesses to figure out the key. But a 256-bit key will allow for 2^256 options, which is a much, much larger number. And nowadays there are algorithms that allow for 2048-bit keys, which are even larger.
- So one service that offers file storage and syncing, Dropbox, encrypts our data for us. They could encrypt it with our password, but then we wouldn't be able to share our files and folders with someone else without sharing our password, or decrypting our files first. So Dropbox must have the key that actually encrypts the data, but that key might be again encrypted with our password and perhaps transferred as encrypted with our friend's password, so they can unlock it and see the shared file. But Dropbox still has access to the key. And this key is actually a master key that encrypts all the files, which helps them reduce the amount of data they use. For example, if many people had the same file, such as a movie file, if each copy was encrypted with a different key, they would have to store all the copies. But using one key, they only need to store one copy, and give people access to them if they are uploading the exact same file.
- If we were really concerned about our privacy or security, we could encrypt our files locally on our computers first, and then upload them to Dropbox, so Dropbox can't read those files.