DATASTRUCTURES WITH JAVA EXAMPLES

Big – O notations:

<https://stackoverflow.com/questions/2307283/what-does-olog-n-mean-exactly>

<https://www.interviewcake.com/article/java/big-o-notation-time-and-space-complexity>

* **O(1) (best case):** Given the page that a business's name is on and the business name, find the phone number.
* **O(1) (average case):** Given the page that a person's name is on and their name, find the phone number.
* **O(log n):** Given a person's name, find the phone number by picking a random point about halfway through the part of the book you haven't searched yet, then checking to see whether the person's name is at that point. Then repeat the process about halfway through the part of the book where the person's name lies. (This is a binary search for a person's name.)
* **O(n):** Find all people whose phone numbers contain the digit "5".
* **O(n):** Given a phone number, find the person or business with that number.
* **O(n log n):** There was a mix-up at the printer's office, and our phone book had all its pages inserted in a random order. Fix the ordering so that it's correct by looking at the first name on each page and then putting that page in the appropriate spot in a new, empty phone book.

For the below examples, we're now at the printer's office. Phone books are waiting to be mailed to each resident or business, and there's a sticker on each phone book identifying where it should be mailed to. Every person or business gets one phone book.

* **O(n log n):** We want to personalize the phone book, so we're going to find each person or business's name in their designated copy, then circle their name in the book and write a short thank-you note for their patronage.
* **O(n2):** A mistake occurred at the office, and every entry in each of the phone books has an extra "0" at the end of the phone number. Take some white-out and remove each zero.
* **O(n · n!):** We're ready to load the phonebooks onto the shipping dock. Unfortunately, the robot that was supposed to load the books has gone haywire: it's putting the books onto the truck in a random order! Even worse, it loads all the books onto the truck, then checks to see if they're in the right order, and if not, it unloads them and starts over. (This is the dreaded [**bogo sort**](http://en.wikipedia.org/wiki/Bogosort).)
* **O(nn):** You fix the robot so that it's loading things correctly. The next day, one of your co-workers plays a prank on you and wires the loading dock robot to the automated printing systems. Every time the robot goes to load an original book, the factory printer makes a duplicate run of all the phonebooks! Fortunately, the robot's bug-detection systems are sophisticated enough that the robot doesn't try printing even more copies when it encounters a duplicate book for loading, but it still has to load every original and duplicate book that's been printed.

Cheat sheet with all summary

<https://gist.github.com/TSiege>

<https://algs4.cs.princeton.edu/cheatsheet/>

Binary tree: best video

<https://www.youtube.com/watch?v=oSWTXtMglKE>



