time complexity - What does O(log n) mean exactly? - Stack Overflow

> I cannot understand how to identify a function with a log time.

The most common attributes of logarithmic running-time function are that:

- the choice of the next element on which to perform some action is one of several possibilities, and
- only one will need to be chosen.

or

- the elements on which the action is performed are digits of n

This is why, for example, looking up people in a phone book is O(log n). You don't need to check *every* person in the phone book to find the right one; instead, you can simply divide-and-conquer by looking based on where there name is alphabetically, and in every section you only need to explore a subset of the each section before you eventually find someone's phone number.

Of course, a bigger phone book will still take you a longer time, but it won't grow as quickly as the proportional increase in the additional size.

We can expand the phone book example to compare other kinds of operations and *their* running time. We will assume our phone book has *businesses* (the "Yellow Pages") which have unique names and *people* (the "White Pages") which may not have unique names. A phone number is assigned to at most one person or business. We will also assume that it takes constant time to flip to a specific page.

Here are the running times of some operations we might perform on the phone book, from best to worst:

- **O(1) (worst 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.
- **$O(n^2)$:** 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.

- **$O(n \cdot n!)$:** 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**.)

- **$O(n^n)$:** 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.