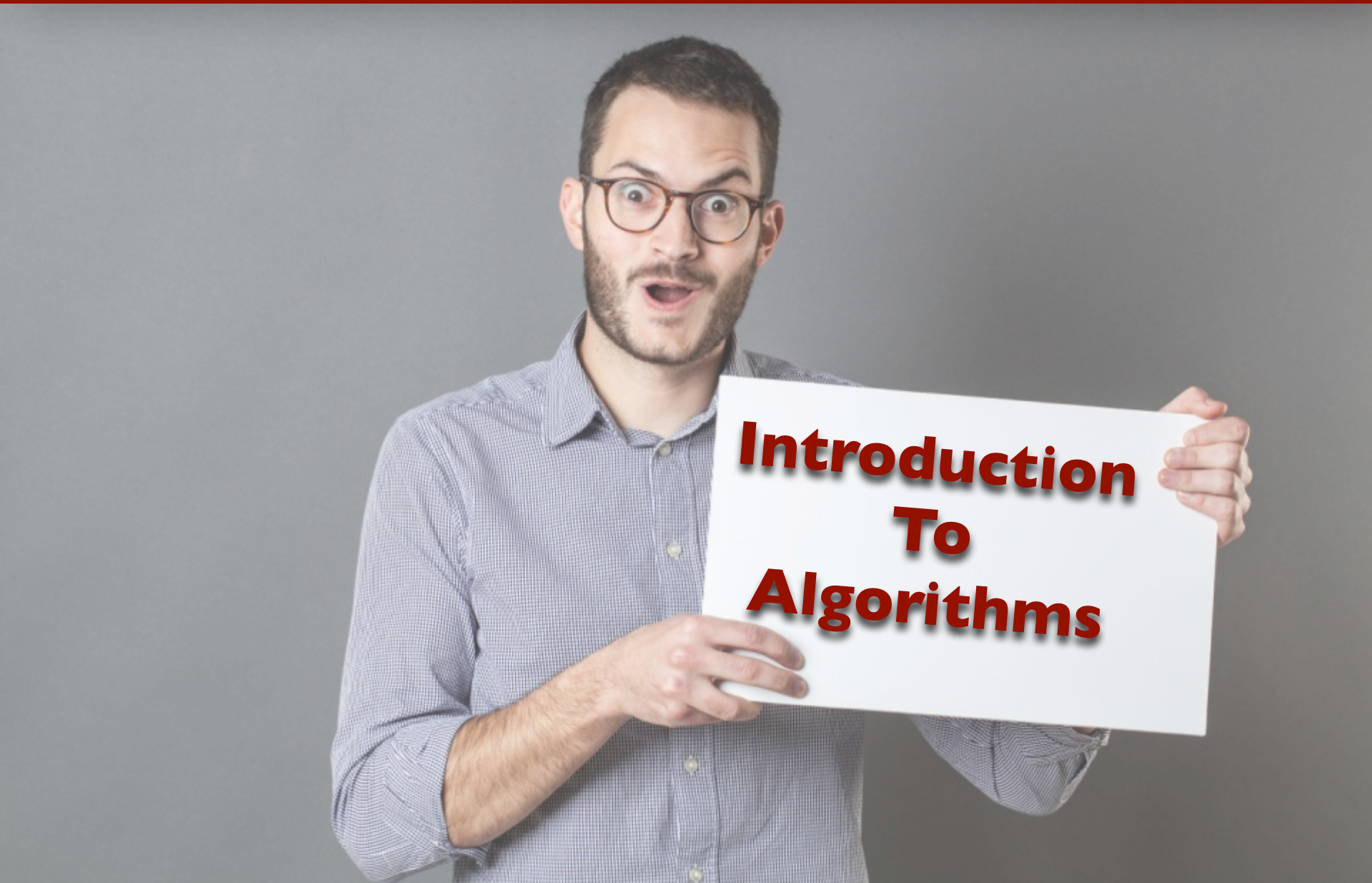

Introduction
To
Algorithms

How Do You Keep A Software Engineer Shampooing Forever?

Gift him a shampoo pack with instructions:

- Lather
- Rinse
- Repeat

Img Src: Wikipedia

Ambiguous Instructions Stump Everyone Not Just Software Engineers

Have you figured out which direction to go yet?

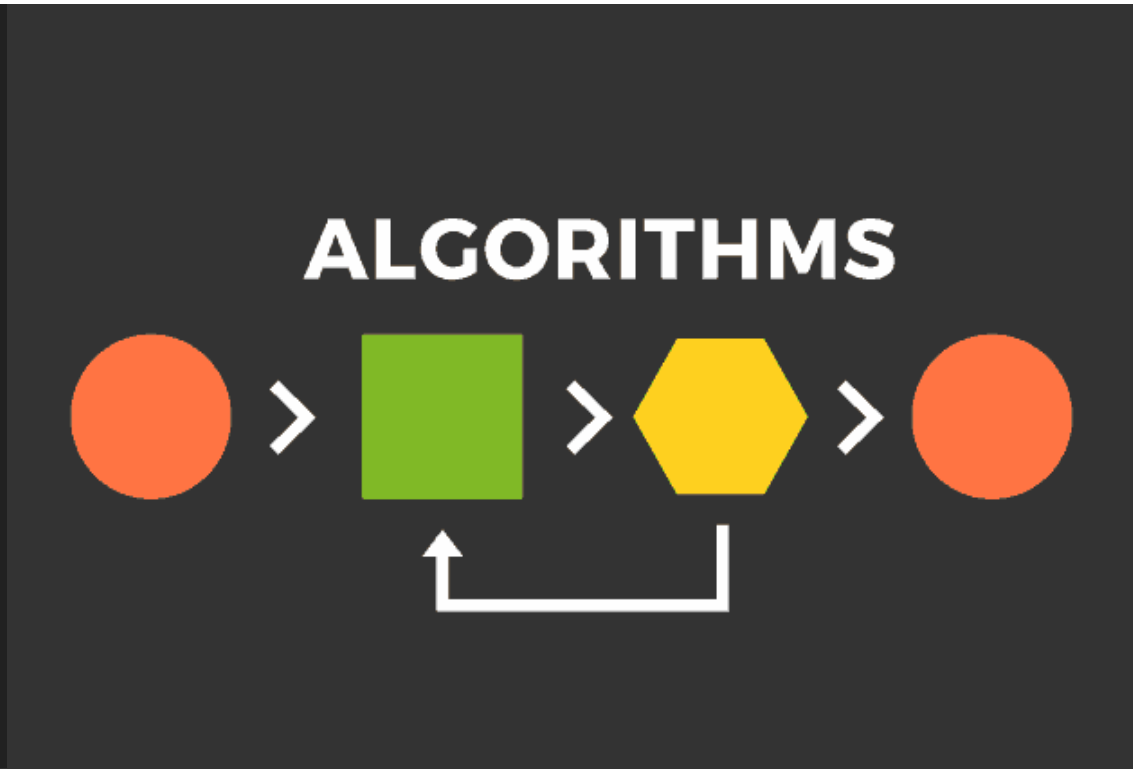
Img Src: pinterest

Machines Need Clear Instructions Too

A computer executes a program which is nothing but a sequence of codified instructions.

What Is An Algorithm?

The sequence of steps to solve a specific problem or attain a specific goal is called an algorithm.



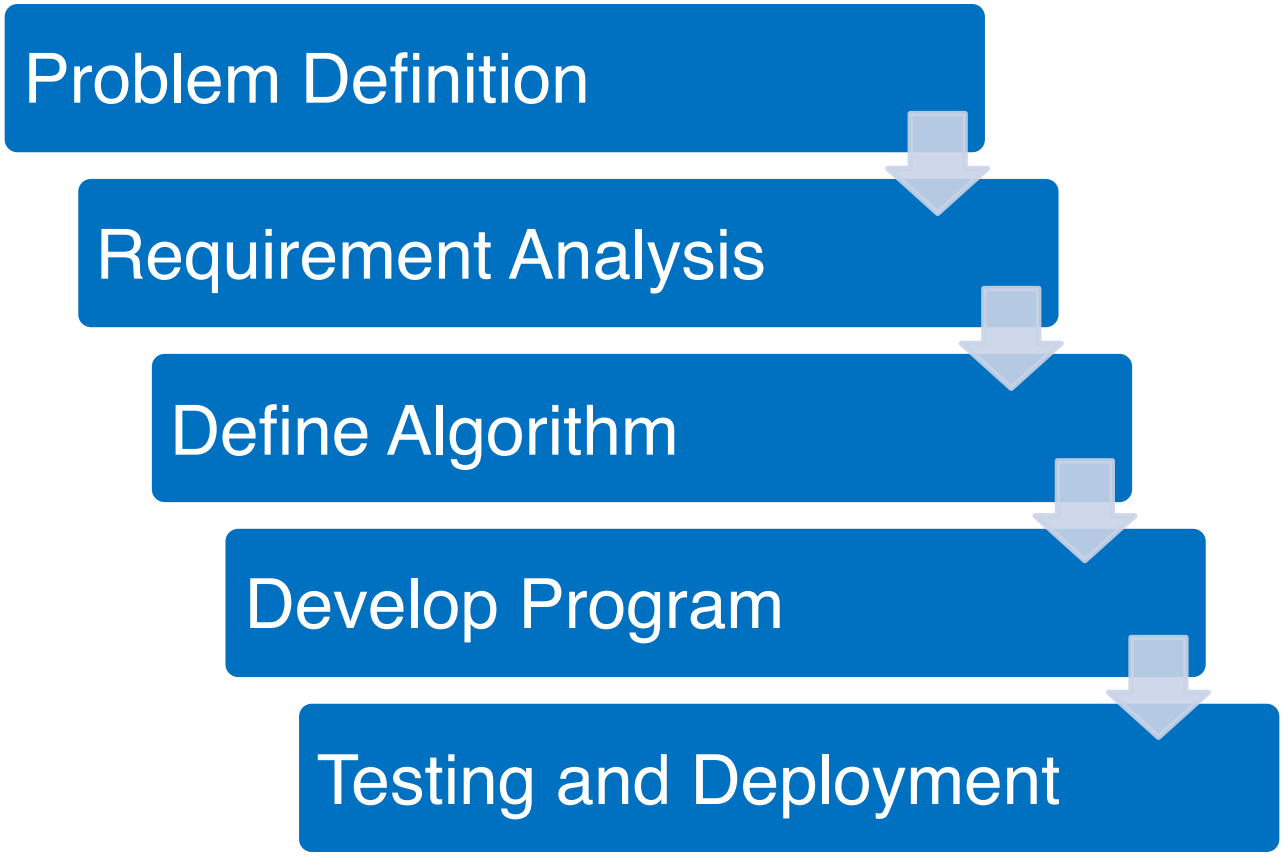
Img Src: pandora

Why Are Algorithms Important?

Machines need clear instructions to solve a problem optimally. You need to feed your machine the right algorithm to get the correct result within minimum resources (time, CPU units, memory).

Img Src: pchelp

The Software Development Life Cycle



What Software Engineers Do

As a software engineer, your key responsibility will be to understand business problems and find the best path to solve them. In short this is what your daily task will look like:

PROBLEM Algorithm **----------> SOLUTION**

Characteristics Of An Algorithm

Input and Output

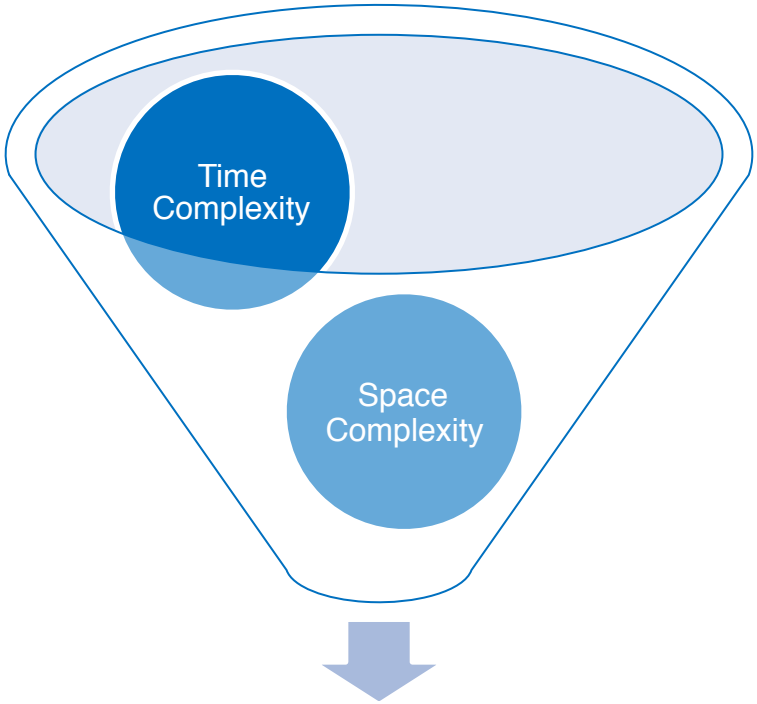
Finite

Correct

Definite

Effective

Efficiency Of An Algorithm



Efficiency Of The Algorithm

Time Complexity

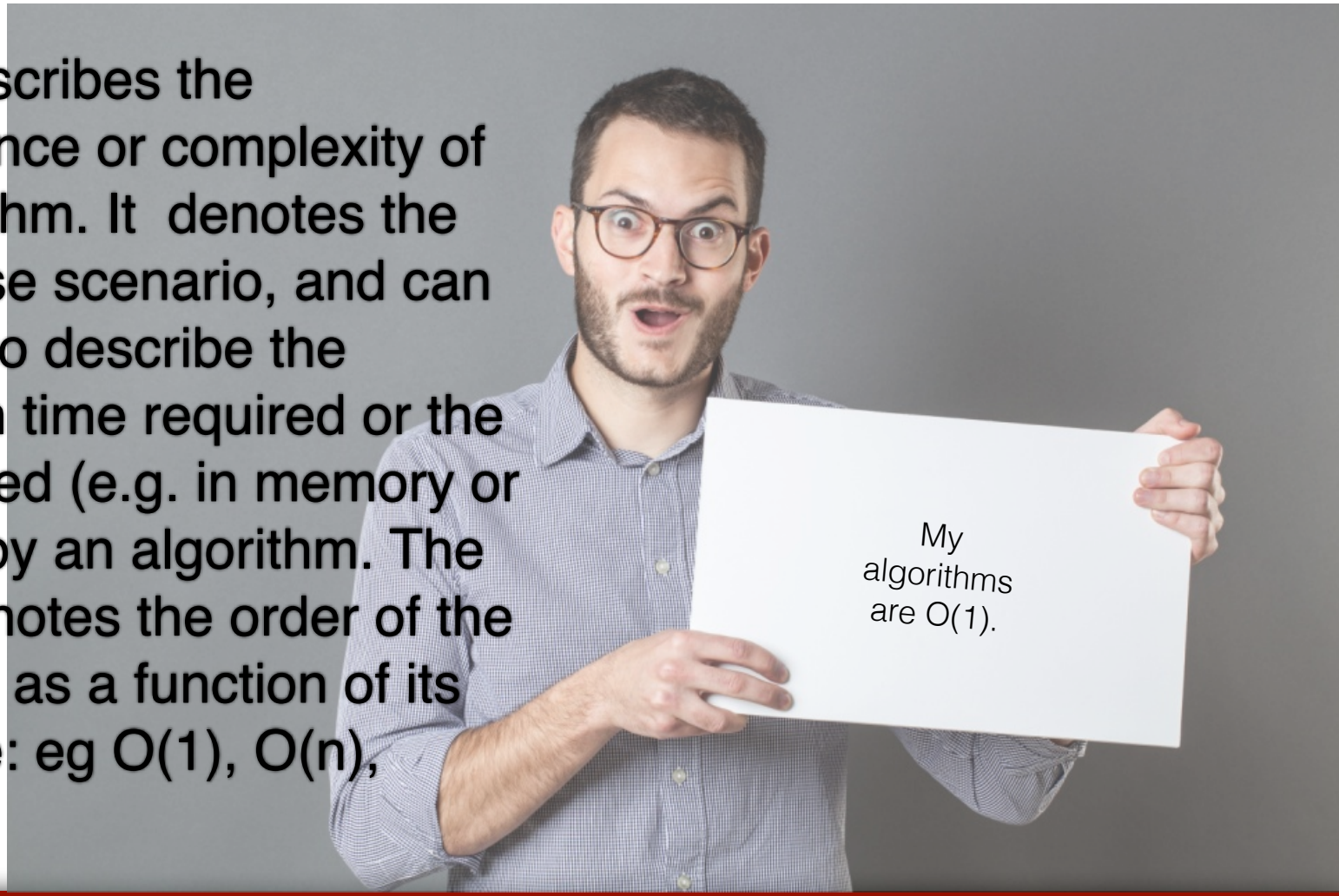
Time complexity of an **algorithm** denotes the amount of **time** taken by an **algorithm** to run, expressed as a function of the length of the input. Time complexity actually measures the number of steps the algorithm executes rather than the actual time taken which depends also on factors like processor speed, network load etc.

Space Complexity

Space Complexity of an **algorithm** is total **space** taken by the **algorithm** as a function of the input size.

The Big O

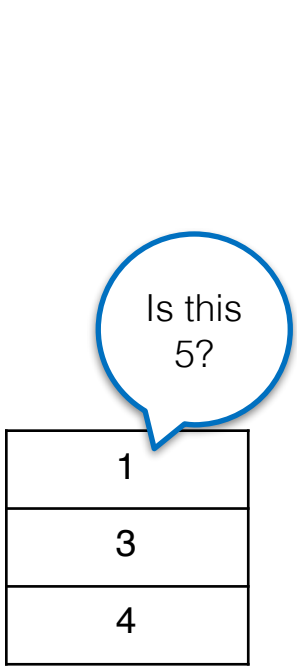
Big O describes the performance or complexity of an algorithm. It denotes the worst-case scenario, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm. The Big O denotes the order of the algorithm as a function of its input size: eg O(1), O(n), O(logn).

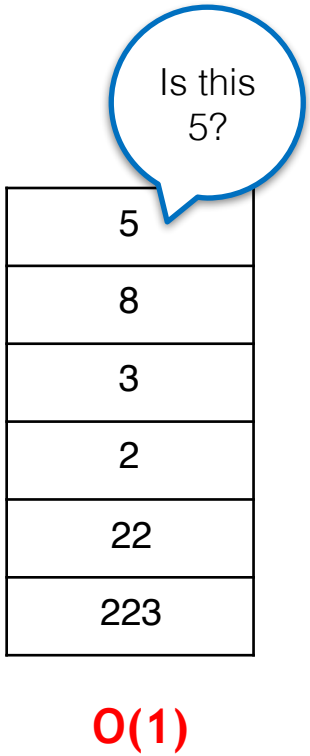


O(1) Algorithms

O(1) algorithms always take a constant amount of time irrespective of the size of the input.

For example, the algorithm to check if the first number in a list is 5 will always take constant time irrespective of the size of the list.

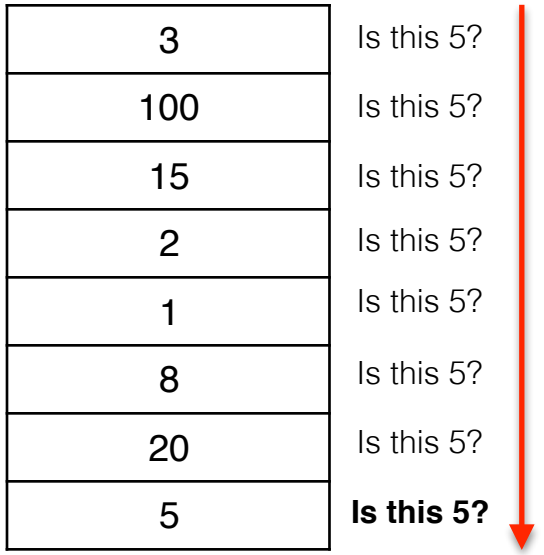




O(n) Algorithms

Say you have a jumbled up list of 8 numbers. You have to find if number 5 is in the list. What will you do?

Pick each number
If number = 5, terminate.
If number is not equal to 5, keep number aside. Search the rest of the numbers. In the worst case you have to do this n = 8 times. The algorithm has complexity = O(n).



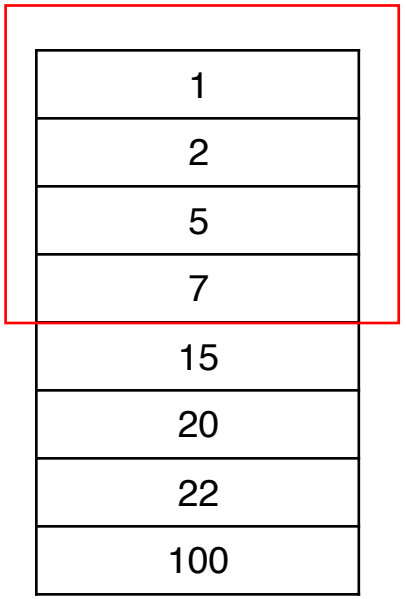
O(logn) Algorithms

Say the list in the last example is now sorted. How much time will you take to find 5 in the list?

1
2
5
7
15
20
22
100

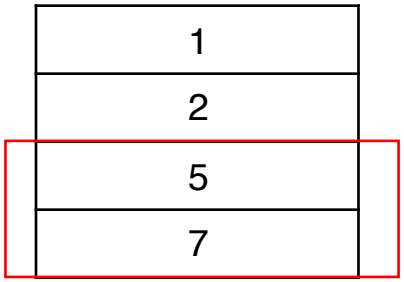
O(logn) Algorithms

Pass 1: Divide the list into 2. Compare 5 to the largest (bottom-most number) in upper partition. Since 5 < 7, 5 will be in the upper partition. Discard lower partition.



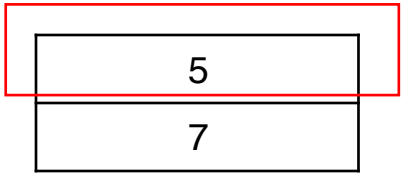
O(logn) Algorithms

Pass 2: Divide the list from pass1 into 2 halves. Compare 5 with the largest number in the upper partition. Since 5 > 2, 5 must be in the lower partition. Discard the upper partition



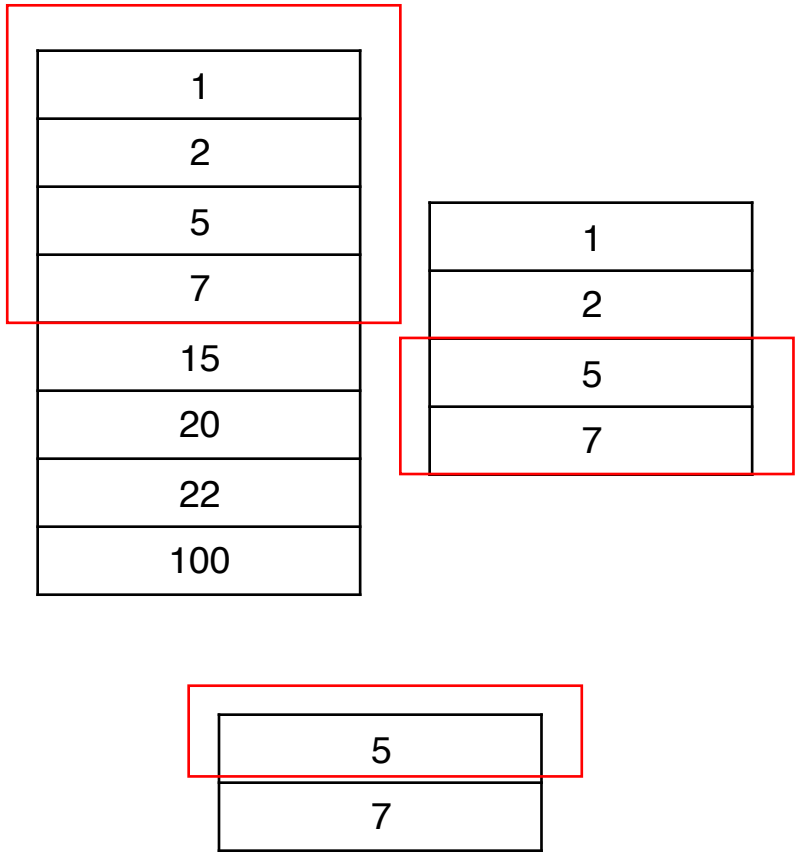
O(logn) Algorithms

Pass 3: Divide the list from pass 2 into 2 halves. Compare 5 with the largest number (the lower-most number) in the upper partition. You have found your 5!



O(logn) Algorithms

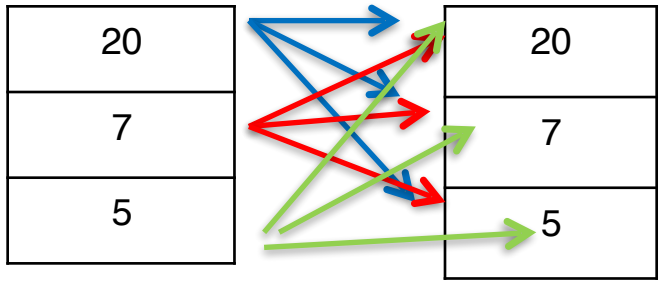
In this algorithm you are halving the input set till you have partitions of 1. How many times do you have to repeat the operation of halving the set?
Let k be the number of times the set is halved
Then $n/2^k = 1$ or $k = \log_2 n$



O(n²) Algorithms

The Problem: Find the smallest number in the list

The easiest solution is to take each number in the list and compare to all other numbers in the list to see if it is the smallest number. For a list of size 3, in the worst case this will be done 3 x 3 times or 9 times. The algorithm has complexity O(n²)

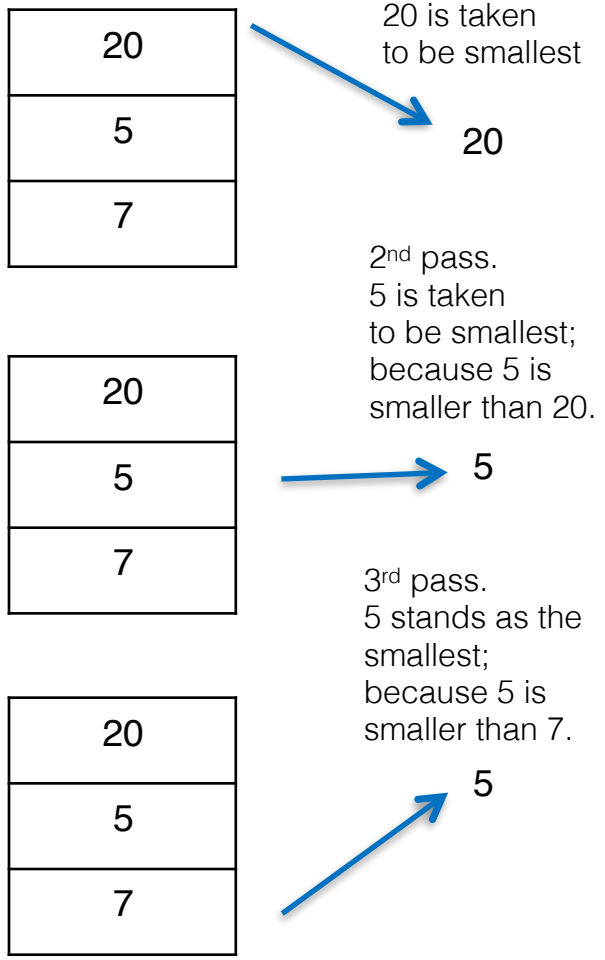


O(n²) Comparisons

Reducing Algorithmic Complexity

Is there a more efficient way of finding the smallest number in a list?

Lets designate the first number in the list, in this case 20 as the smallest number. Compare each number in the list with the smallest number. If the number is smaller than the smallest number, set smallest number = number. You are repeating this n=3 times. The complexity is O(n).

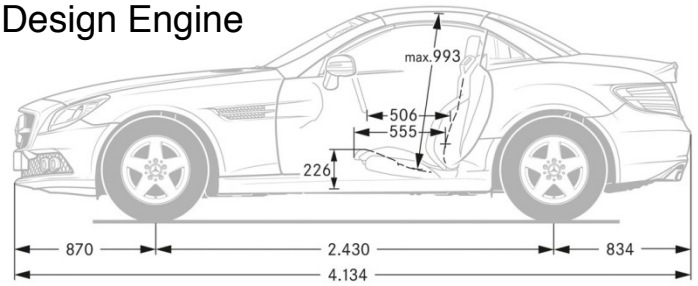


Modular Approach To System Design

While designing complex systems, large systems are broken down into modules. This process is called modularization.

Design Storage

Design Engine



Design Body

Img Src: Wikipedia

Problem Solution Approach