**Algorithms Made Easy**

* **rubiks cube**
* **towers of Hanoi**
* **Game of Life**
* **Sorting algorithms**
* **API’s?**

**Serial and parallel algorithms**

**Programming languages**

**Background: What is an algorithm?**

At its most basic level, an algorithm is a roadmap for solving a problem. People create algorithms by thinking about ways to approach a problem, such as sorting objects in a row. Then, to allow computers to execute these algorithms, people create programs in various languages, such as Python, C++, and Java. Programs derived from algorithms are the backbone of nearly every computer process, from simple addition to communication via the Internet.

**What makes an algorithm useful?**

A useful algorithm solves a general, well-defined problem. For example, let us consider the problem of sorting a list of objects, with the assumption that there is some way of ordering them. There are many instances of this problem – for instance, consider the differences in sorting the lists {Albert, Calvin, Bob} versus sorting the list {4, 3, 1, 2}. However, the basic **idea**, sorting a list of objects by some method of ordering, remains constant.

As it turns out, there are many algorithms that solve this problem. One algorithm, which is the one humans use in their everyday lives, is scanning the list, finding the lowest value, and bringing it to the front. Thus, our sorting looks like this:

Step 0: {4, 3, 1, 2}

Step 1: {1, 4, 3, 2}

Step 2: {1, 2, 4, 3}

Step 3: {1, 2, 3, 4}

A more efficient algorithm is **Merge Sort**. We first group the list in sets of twos, then order those sets of twos as sets of fours, and so on. Our sorting looks like this:

Step 0: {4, 3 | 1, 2}

Step 1: {3, 4 | 1, 2}

Step 2: {1, 2, 3, 4}

Although there are some hidden calculations involved in both examples, one should note that the latter algorithm is good for efficiently sorting large lists, whereas the former can be very inefficient as list sizes grow. The efficiency of **Merge Sort** comes from the fact that it breaks a large problem into bite-sized pieces which are then combined into a final product.

Essentially, there are three things that makes an algorithm useful:

1. It solves a general, well-defined problem
2. It can be applied to different instances of a problem and still yield a valid solution
3. It can efficiently solve the problem, regardless of scale

TO LEARN ABOUT SCALE CLICK HEREEEEE

**Algorithms vs. Programs**

Algorithms are ideas, whereas a program is a series of commands written out for a computer to execute. Algorithms are what a program should do, if written correctly. Consider a simple algorithm for microwaving instant noodles:

Add boiling water

Place in microwave

Remove from microwave

Now consider a computer-style “program” to execute this algorithm:

Obtain (brand name) instant noodles

Obtain water

Boil water

Open lid of instant noodles

Add 3/4 cup boiled water to instant noodles

Open microwave

Close microwave

Set to 60 seconds

Wait 60 seconds

Open microwave

Remove instant noodles

Notice how this example illustrates a few properties of programs and algorithms:

1. An algorithm is an idea that is **implemented** by a program
2. A program is, by nature, more specific than an algorithm
3. A program is written for a computer, whereas an algorithm is written for humans to understand

By learning and understanding algorithms, humans can create approaches to complex problems in the real world. Then, these approaches can be written in a computer-friendly format in the form of a program, so that computers can carry out the ideas that humans produce.

Computers take instructions in steps. An **algorithm** is a series of such steps

**Algorithms vs Programs**

**Order of the steps**

**Conditionals (if/then/else?)**

**Errors**

Complexity: Understanding Complexity

* Multiple ways to do same thing

Outline:

* “Algorithm”
  + basic definition
  + etymology
  + basic examples
  + harder examples