1.2 Algorithms as a technology

Kevin Woo

Desperate college kid who keeps on trying.

Introduction

Computer resources are limited. There are sufficient reasons to make things faster/more efficient.

Efficiency

Efficiency is measured with big O notation mostly. For example, we will see in Chapter 2 **insertion sort** which has a complexity as follows along with **merge sort.**

$$Insertion\ Sort = c_1 n^2$$

$$Merge\ Sort = c_2 nlg(n),\ where\ lg = log_2$$

Example:

Given a **faster** computer "A" running **insertion sort** and a **slower** computer "B" running **merge sort,** assume they have to sort an array of 10 million numbers.

Supposing that "A" executes 10 billion instructions per second and "B" executes 10 million instructions per second, (basically "A" is 1000x faster than "B"). We will also assume that the programmer of each algorithm made the insertion sort to have a constant of 2 and the merge sort to have a constant of 50. **Who will sort faster?**

Computer A

$$\frac{2 * (10^7)^2 instructions}{10^{10} instructions/second} = 20,000\ seconds\ (more\ than\ 5.5\ hours)$$

Computer B

$$\frac{50 * 10^7 lg 10^7 instructions}{10^7} instructions/second = 1163\ seconds\ (less\ than\ 20\ minutes)$$

Through this we can see that despite having a worse constant and computer, because of the algorithm, we managed to reduce the execution time.

Algorithms and other technologies

TLDR; Algorithms in real life is an important part despite the use of different programming paradigm such as OOP, GUIS, UI/UX, arichitectures, and networking/infrastructure.

Exercises

1.2-1 Give an example of an application that requires algorithmic content at the application level, and discuss the function of the algorithms involved An example of an application that requires algorithmice content at the application level could be any front end component-based framework such as Angular or React where component tree has to be made. This component tree could be traversed, prioritized and rendered depending on the underlying algorithm.

1.2-2 Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in 8n2 steps, while merge sort runs in 64n lg n steps. For which values of n does insertion sort beat merge sort?

$$8n^2 = 64nlg(n)$$

$$n^2 = 8nlg(n)$$

$$\frac{n^2}{nlg(n)} = 8$$

$$\frac{n}{lg(n)} = 8$$

$$n = 43.5593...$$

1.2-3 What is the smallest value of n such that an algorithm whose running time is 100n2 runs faster than an algorithm whose running time is 2n on the same machine?

$$100n^2 = 2^n = 14.3$$

We would then need 15 elements as n.