Introduction

An algorithm is a well-defined computational procedure that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.

An algorithm is a sequence of computational steps that transform the input into the output.

An algorithm is a tool for solving a well-specified computational problem.

The sorting Problem: The input is sequence of numbers $\langle a_1, a_2, ..., a_n \rangle$. The output is a permutation (reordering) $\langle a_1', a_2', ..., a_n' \rangle$ of the input sequence such that $a_1' \leq a_2' \leq ... \leq a_n'$.

Example: If the input is $\langle 31,41,59,26,41,58 \rangle$ a sorting algorithm should return the sequence $\langle 26,31,41,41,58,59 \rangle$ as output.

The above input sequence is called an **instance** of the sorting problem. In general, an **instance of a problem** consists of the input (satisfying whatever constraints are imposed in the problem statement) needed to compute a solution to the problem.

An algorithm is said to be **correct** if, for every input instance, it halts with the correct output. We say that a correct algorithm **solves** the given computational problem. An incorrect algorithm might not halt at all on some input instances, or it might halt with an answer other than the desired one.

Analyzing an algorithm: Predicting the resources required such as memory, communication bandwidth, or computational time. In most cases, the computational time is really what we want to measure.

Efficiency: Algorithms devised to solve the same problem often differ dramatically in their efficiency. These differences can be much more significant than differences due to hardware and software.

Example: Suppose we have two computers, computer A, a fast computer running insertion sort. Computer B, a slow computer running merge sort. Each must sort an array of ten million numbers. Suppose that computer A executes ten billion instructions per second and computer B executes only ten million instructions per second (Notice that computer A is 1000 times faster than computer B). Suppose Computer A requires $2n^2$ instructions to sort n numbers and computer B requires $50n\lg n$ ($\lg n = \log_2 n$) instructions to sort the n numbers. Then
Computer A takes

$$\frac{2.(10^7)^2\, instructions}{10^{10}\, instructions/\sec ond} = 20{,}000 \sec onds$$ (more than 5.5 hours)

As for computer B, it takes

$$\frac{50.10^7.\lg 10^7\, instructions}{10^7\, instructions/\sec ond} \approx 1163 \sec onds$$ (less than 20 minutes)

This means that computer B solves the problem at hand 20 times faster than computer A!