

The Role of Algorithms in Computing

What will we study?

- Look at some **classical algorithms** on different kinds of problems
- How to **design** an algorithm
- How to show that an algorithm works **correctly**
- How to **analyze** the performance of an algorithm

1.1 Algorithms

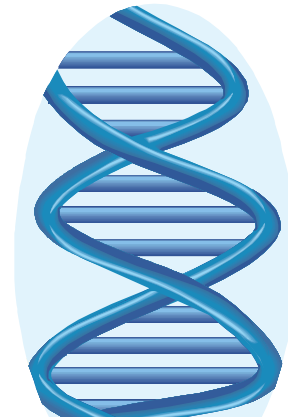
- Algorithm: Any **well-defined** computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.
- *Or; Algorithm*: A method of solving a problem, using a sequence of **well-defined** steps
- Defined a Sorting problem
- Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$
- Output: A permutation of the input sequence such that $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Instances of a problem

- An **instance of a problem** consists of the input (satisfying constraint 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
- A **correct** algorithm **solves** the given computational problem. An **incorrect** algorithm might not halt at all on some input instance, or it might halt with other than the desired answer

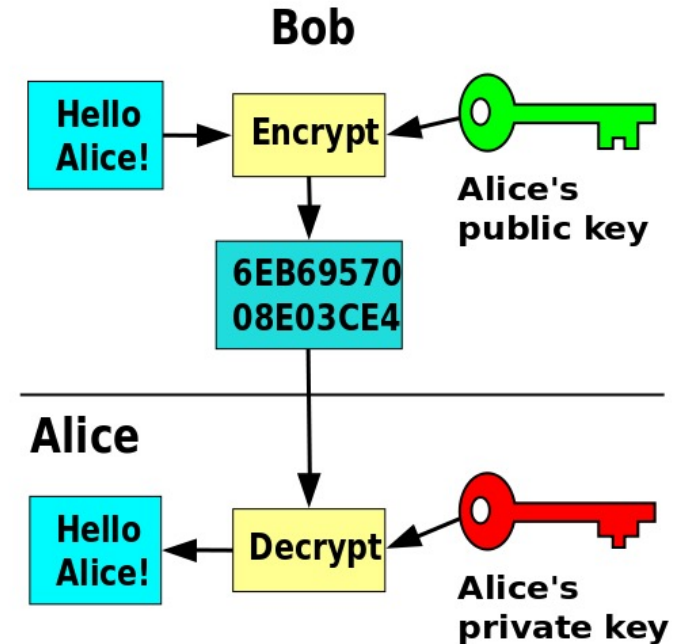
What kinds of problems are solved by algorithms? (1/2)

- The Human Genome Project
 - Determine the sequences of the 3 billion chemical base pairs of DNA
 - Identify all the 100,000 genes in human DNA
- The Internet applications
 - Quickly access and retrieve large amounts of information such as Google
 - Large content delivery networks (CDNs) and major Internet service providers (ISPs) may handle tens of terabytes or even petabytes of data per second during peak times.



What kinds of problems are solved by algorithms? (2/2)

- Electronic commerce with public-key cryptography and digital signatures
- Manufacturing and other commercial enterprises need to allocate scarce resources most beneficially.



1.2 Algorithms as a technology

- **Efficiency:**
 - Different algorithms that solve the same problem often differ noticeably in their efficiency
 - These differences can be much more significant than differences due to hardware and software
- For example, in Chapter 2 we will see that *insertion sort* takes time roughly equal to $c_1 n^2$ (c_1 is constant) to sort n items. But, merge sort takes time approximately equal to $c_2 n \lg n$ (c_2 is constant)

1.2 Algorithms as a technology

- For example, assume a faster computer A (10^{10} instructions/sec) running insertion sort against a slower computer B (10^7 instructions/sec) running merge sort.
- Suppose that $c_1=2$, $c_2=50$ and $n = 10^7$.
 - the execution time of computer A is $2(10^7)^2 / 10^{10}$ instructions/sec = **20,000** seconds
 - the execution time of computer B is $50 \times 10^7 \times \lg 10^7 / 10^7$ instructions/sec = **1,163** seconds

Exercises

- Exercises: 1.2-2, 1.2-3
- Problem 1.1