

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

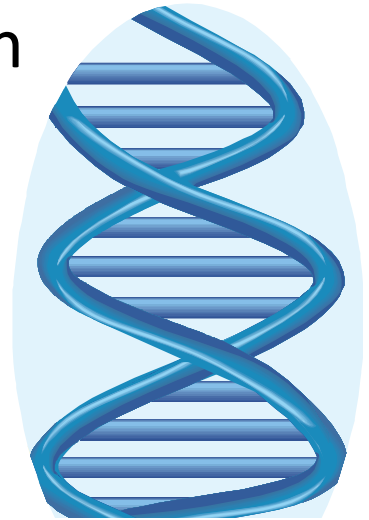
- **Algorithm:** Any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.
- *Or;* **Algorithm:** A method of solving a problem, using a sequence of well-defined steps
- **Example:** Sorting problem
- Input: A sequence 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 algorithm is said to be correct if for every input instance, it halts with the correct output
- An instance of a problem consists of all inputs needed to compute a solution to the problem
- 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

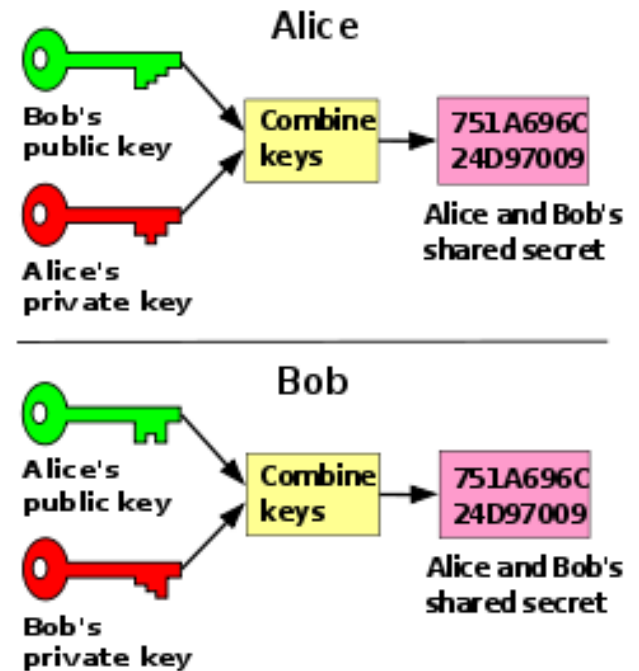
What kind of problem are solved by algorithms? (1/2)

- The Human Genome Project
 - Identify all the 100,000 genes in human DNA
 - Determine the sequences of the 3 billion chemical base pairs of DNA
- The Internet applications
 - Quickly access and retrieve large amount of information such as Google



What kind of problem 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 in the most beneficial way.



1.2 Algorithms as a technology

- **Efficiency:**
 - Different algorithms solve the same problem often differ noticeably in their efficiency
 - These differences can be much more significant than difference 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 roughly 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 $50 \cdot 10^7 \lg 10^7 / 10^7$ instructions/sec = **1,163** seconds