

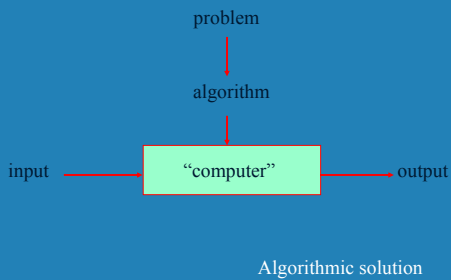
Algorithm

Q An *algorithm* is a sequence of unambiguous instructions for solving a problem, i.e., for obtaining a required output for any legitimate input in a finite amount of time.

Historical Perspective

- Q Euclid's algorithm for finding the greatest common divisor
- Q Muhammad ibn Musa al-Khwarizmi – 9th century mathematician
www.lib.virginia.edu/science/parshall/khwariz.html

Notion of algorithm



Example of computational problem: sorting

- Q Statement of problem:
 - Input: A sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$
 - Output: A reordering of the input sequence $\langle a'_1, a'_2, \dots, a'_n \rangle$ so that $a'_i \leq a'_j$ whenever $i < j$
- Q Instance: The sequence $\langle 5, 3, 2, 8, 3 \rangle$
- Q Algorithms:
 - Selection sort
 - Insertion sort
 - Merge sort
 - (many others)

Selection Sort

- Q Input: array $a[1], \dots, a[n]$
- Q Output: array a sorted in non-decreasing order
- Q Algorithm:

```
for  $i=1$  to  $n$ 
    swap  $a[i]$  with smallest of  $a[i], \dots, a[n]$ 
```

• see also pseudocode, section 3.1

Some Well-known Computational Problems

- Q Sorting
- Q Searching
- Q Shortest paths in a graph
- Q Minimum spanning tree
- Q Primality testing
- Q Traveling salesman problem
- Q Knapsack problem
- Q Chess
- Q Towers of Hanoi
- Q Program termination

Basic Issues Related to Algorithms

- Q How to design algorithms
- Q How to express algorithms
- Q Proving correctness
- Q Efficiency
 - Theoretical analysis
 - Empirical analysis
- Q Optimality

Algorithm design strategies

- Q Brute force
- Q Greedy approach
- Q Divide and conquer
- Q Dynamic programming
- Q Decrease and conquer
- Q Backtracking and Branch and bound
- Q Transform and conquer
- Q Space and time tradeoffs

Analysis of Algorithms

- Q How good is the algorithm?
 - Correctness
 - Time efficiency
 - Space efficiency
- Q Does there exist a better algorithm?
 - Lower bounds
 - Optimality

What is an algorithm?

- Q Recipe, process, method, technique, procedure, routine,... with following requirements:
 1. Finiteness
 - Q terminates after a finite number of steps
 2. Definiteness
 - Q rigorously and unambiguously specified
 3. Input
 - Q valid inputs are clearly specified
 4. Output
 - Q can be proved to produce the correct output given a valid input
 5. Effectiveness
 - Q steps are sufficiently simple and basic

Why study algorithms?

- Q Theoretical importance
 - the core of computer science
- Q Practical importance
 - A practitioner's toolkit of known algorithms
 - Framework for designing and analyzing algorithms for new problems