CSCE 310J: Data Structures & Algorithms

Introduction

Dr. Steve Goddard
goddard@cse.unl.edu

http://www.cse.unl.edu/~goddard/Courses/CSCE310J

CSCE 310J: Data Structures & Algorithms

- **Giving credit where credit is due:**
  - **Most of the lecture notes are based on the slides from the Textbook's companion website**
    - **http://www.aw.com/cssuport/**
  - **I have modified them and added new slides**

What is a Computer Algorithm?

- **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.**

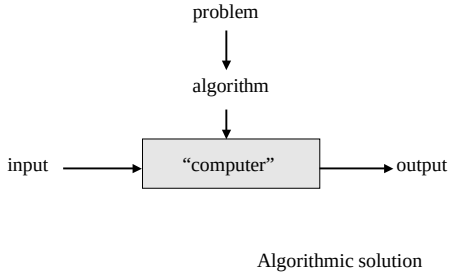
Features of Algorithm

- **"Besides merely being a finite set of rules that gives a sequence of operations for solving a specific type of problem, an algorithm has the five important features" [Knuth1]**
  - **finiteness (otherwise: computational method)**
    - » **termination**
  - **definiteness**
    - » **precise definition of each step**
  - **input (zero or more)**
  - **output (one or more)**
  - **effectiveness**
    - » **"Its operations must all be sufficiently basic that they can in principle be done exactly and in a finite length of time by someone using pencil and paper" [Knuth1]**

Computer Algorithm or Program?

- **A computer algorithm is**
  - **a detailed step-by-step method for solving a problem using a computer.**
- **A program is**
  - **an implementation of one or more algorithms.**

Notion of algorithm



Algorithmic solution

Example of computational problem: sorting

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

Selection Sort

- **Input: array a[1],…,a[n]**
- **Output: array a sorted in non-decreasing order**
- **Algorithm:**

for *i*=1 to *n*
swap a[*i*] with smallest of a[i],…a[*n*]

- see also pseudocode, section 3.1

Some Well-known Computational Problems

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

Basic Issues Related to Algorithms

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

Algorithm design strategies

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

Analysis of Algorithms

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

Correctness

- **Termination**
  - **Well-founded sets: find a quantity that is never negative and that always decreases as the algorithm is executed**
- **Partial Correctness**
  - **For recursive algorithms: induction**
  - **For iterative algorithms: axiomatic semantics, loop invariants**

Complexity

- **Space complexity**
- **Time complexity**
  - **For iterative algorithms: sums**
  - **For recursive algorithms: recurrence relations**

What is an algorithm?

- **Recipe, process, method, technique, procedure, routine,… with following requirements:**

1. **Finiteness**
   - **terminates after a finite number of steps**
2. **Definiteness**
   - **rigorously and unambiguously specified**
3. **Input**
   - **valid inputs are clearly specified**
4. **Output**
   - **can be proved to produce the correct output given a valid input**
5. **Effectiveness**
   - **steps are sufficiently simple and basic**

Why study algorithms?

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