Discrete Mathematics: Lecture 11

- Last time:
  - Chap 3.1: Algorithms
  - Chap 3.2: The Growth of functions
- Today:
  - Chap 3.3: Complexity of algorithms
  - Chap 4.1: Divisibility and modular arithmetic
- Next time:
  - Chap 4.2: Integer representations and algorithms
  - Chap 4.3: Primes and greatest common divisors

Review of last time

- Algorithms, pseudocode description of algorithms
- Linear search and binary search
- Insertion sort and bubble sort
- Greedy algorithms
- The halting problem is unsolvable
- Big-O/Ω/Θ notation and results

An Example

Give an algorithm for finding the maximum value in a finite sequence of integers.

procedure $max(a_1, a_2, \ldots, a_n$:integers)
$max := a_1$
for i := 2 to n
    if $max < a_i$ then $max := a_i$
$\{max$ is the largest element$\}$

Searching Algorithms

Problem: Given a pile of assignments, find your own assignment

Locating an element x in an ordered list of distinct elements $a_1, a_2, \ldots, a_n$, or determine that it is not in the list

The linear search algorithm

procedure linear search(x : integer, $a_1, a_2, \ldots, a_n$: distinct integers)
$i := 1$
while ($i \le n$ and $x \ne a_i$)
    $i := i + 1$
if $i \le n$ then $location := i$
else $location := 0$

Binary Search

Problem: If the pile of assignments is sorted according to increasing order of student number, can you find your assignments more efficiently?

Example: To search for 19 in the list
1 2 3 5 6 7 8 10 12 13 15 16 18 19 20 22

The binary search algorithm

procedure binary search(x : integer, $a_1, a_2, \ldots, a_n$: increasing integers)
$i := 1$
$j := n$
while $(i < j)$
  $m := \lfloor (i + j)/2 \rfloor$
  if $x > a_m$ then $i := m + 1$
  else $j := m$
if $x = a_i$ then $location := i$
else $location := 0$

Sorting Algorithms

Problem: Sort a pile of assignments according to increasing order of student number

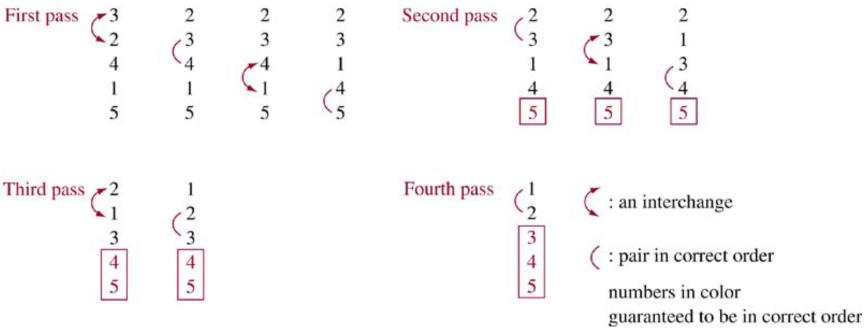
Insertion sort: consider elements one by one, when considering the jth element, insert it into the correct position of the previously sorted $j-1$ elements

Example: Sort 3,2,4,1,5

procedure insertion sort($a_1, a_2, \ldots, a_n$: real numbers with $n \geq 2$)
for $j := 2$ to n
    $i := 1$
    while $a_j > a_i$
        $i := i + 1$
    $m := a_j$
    for $k := i$ to $j - 1$
        $a_{k+1} := a_k$
    $a_i := m$

Bubble sort

compare adjacent elements, interchange them if they are in the wrong order



procedure bubble sort($a_1, a_2, \ldots, a_n$: real numbers with $n \geq 2$)
for $i := 1$ to $n - 1$
    for $j := 1$ to $n - i$
        if $a_j > a_{j+1}$ then interchange a_j and a_{j+1}

Time complexity (时间复杂性)

- expressed in terms of the number of operations used
- the operations can be comparison of integers, addition, multiplication, and division of integers
- not described in terms of actual computer time because of the difference in time needed for different computers to perform basic operations

Examples

- find the maximum value in a sequence
- linear search
- worst-case complexity (最坏情况复杂性): the largest number of operations needed
- binary search
- average-case complexity (平均情况复杂性): the average number of operations needed; usually more difficult to analyze than worst-case complexity
- average-case complexity of linear search
  - assumption: x is in the list and it is equally likely x is in any position

Examples

- from now on, we ignore the comparisons needed to determine if we have reached the end of a loop
- worst-case complexity of bubble sort
- worst-case complexity of insertion sort
- complexity of matrix multiplication
- how should the matrix chain $\mathbf{A}_1\mathbf{A}_2 \dots \mathbf{A}_n$ be computed?
  - *e.g.*, $\mathbf{A}_1$ is 30×20, $\mathbf{A}_2$ is 20×40, $\mathbf{A}_3$ is 40×10

TABLE 1 Commonly Used Terminology for the Complexity of Algorithms.

Complexity	*Terminology*
$\Theta(1)$	Constant complexity
$\Theta(\log n)$	Logarithmic complexity
$\Theta(n)$	Linear complexity
$\Theta(n \log n)$	$n \log n$ complexity
$\Theta(n^b)$	Polynomial complexity
$\Theta(b^n)$, where $b > 1$	Exponential complexity
$\Theta(n!)$	Factorial complexity

TABLE 2 **The Computer Time Used by Algorithms.**

Problem Size	*Bit Operations Used*					
n	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-9} s	10^{-8} s	3×10^{-8} s	10^{-7} s	10^{-6} s	3×10^{-3} s
10^2	7×10^{-9} s	10^{-7} s	7×10^{-7} s	10^{-5} s	4×10^{13} yr	*
10^3	1.0×10^{-8} s	10^{-6} s	1×10^{-5} s	10^{-3} s	*	*
10^4	1.3×10^{-8} s	10^{-5} s	1×10^{-4} s	10^{-1} s	*	*
10^5	1.7×10^{-8} s	10^{-4} s	2×10^{-3} s	10 s	*	*
10^6	2×10^{-8} s	10^{-3} s	2×10^{-2} s	17 min	*	*

*: more than 10^{100} years

Tractable problems

- A problem that is solvable using an algorithm with polynomial-time (多项式时间) worst-case complexity is called tractable.
- The expectation is that the algorithm will produce the solution for reasonably sized input in a relatively short time.
- However, the expectation might not hold if the polynomial has high degree or the coefficients are extremely large
- Fortunately, in practice, the degree and coefficients of polynomials are often small.

Intractable problems (难解问题)

- A problem that is not tractable is called intractable.
- Usually, an extremely large amount of time is required to solve the problem for the worst cases of even small input values.
- However, in practice, an algorithm might be able to solve a problem much more quickly for most cases than for its worst cases.
- Another way that intractable problems are handled is to look for approximate solutions (近似解法).

NP and NP-complete problems (optional)

- Problems for which a solution can be checked in polynomial time are said to belong to the class **NP** (tractable problems are said to belong to class **P**).
- NP-complete problems (NP完全问题) have the property that if any of these problems can be solved by a polynomial worst-case time complexity algorithm, then so can all NP problems.
- So far we do not know if P=NP. There is a 1 million dollar prize for solving this problem.
- But it is generally accepted that P $\neq$ NP.
- The satisfiability problem: check if a compound proposition is satisfiable. It is an NP-complete problem.

Number theory (数论)

- The part of mathematics involving the integers and their properties belongs to number theory.
- This chapter develops the basic concepts of number theory used throughout computer science.

Division

- Definition: If a and b are integers with $a \neq 0$, we say that a divides b, denoted by $a \mid b$, if there is an integer c such that $b = ac$. When a divides b, we say that a is a factor of b and b is a multiple of a. We write $a \nmid b$ if a does not divide b.
- Example: Let n and d be positive integers. How many positive integers not exceeding n are divisible by d?
- Theorem: Let a, b, and c be integers. Then
  1. if $a \mid b$ and $a \mid c$, then $a \mid (b + c)$;
  2. if $a \mid b$, then $a \mid bc$ for all integers c;
  3. if $a \mid b$ and $b \mid c$, then $a \mid c$.
- Corollary: If a, b, and c are integers such that $a \mid b$ and $a \mid c$, then $a \mid (mb + nc)$ whenever m and n are integers.

Quotient and remainder

- Theorem: Let a be an integer and d a positive integer. Then there are unique integers q and r, with $0 \leq r < d$, such that $a = dq + r$. We write $q = a$ div d, and $r = a$ mod d. d – divisor, a – dividend, q – quotient (商), r – reminder (余数).
- Example: divide -11 by 3

Modular arithmetic (模算术)

- Definition: Let a and b be integers and m a positive integer. We say that a is congruent to b modulo m (模m同余), denoted by $a \equiv b (\text{mod } m)$, if m divides $a - b$.
- Theorem: Let a and b be integers and m a positive integer. Then $a \equiv b (\text{mod } m)$ iff $a \text{ mod } m = b \text{ mod } m$.
- Theorem: Let m be a positive integer. If $a \equiv b (\text{mod } m)$ and $c \equiv d (\text{mod } m)$, then $a + c \equiv b + d (\text{mod } m)$ and $ac \equiv bd (\text{mod } m)$.
- Corollary: Let m be a positive integer. Then $(a + b) \text{ mod } m = ((a \text{ mod } m) + (b \text{ mod } m)) \text{ mod } m$ and $ab \text{ mod } m = ((a \text{ mod } m)(b \text{ mod } m)) \text{ mod } m$.

Arithmetic modulo m

- $\mathbf{Z}_m = \{0, 1, \ldots, m-1\}$
- $a +_m b = a + b \bmod m$, $a \cdot_m b = a \cdot b \bmod m$
- Properties: closure, associativity, commutativity, identity elements, additive inverse, distributivity