## Chapter Notes

Every decent algorithm book reflects the design philosophy of its author. For students seeking alternative presentations and viewpoints, we particularly recommend the books of Corman, et. al [CLRS01], Kleinberg/Tardos [KT06], and Manber [Man89].

Formal proofs of algorithm correctness are important, and deserve a fuller discussion than we are able to provide in this chapter. See Gries [Gri89] for a thorough introduction to the techniques of program verification.

The movie scheduling problem represents a very special case of the general *independent set* problem, which is discussed in Section 16.2 (page 528). The restriction limits the allowable input instances to *interval* graphs, where the vertices of the graph $G$ can be represented by intervals on the line and $(i, j)$ is an edge of $G$ iff the intervals overlap. Golumbic [Gol04] provides a full treatment of this interesting and important class of graphs.

Jon Bentley's *Programming Pearls* columns are probably the best known collection of algorithmic "war stories." Originally published in the *Communications of the ACM*, they have been collected in two books [Ben90, Ben99]. Brooks's *The Mythical Man Month* [Bro95] is another wonderful collection of war stories, focused more on software engineering than algorithm design, but they remain a source of considerable wisdom. Every programmer should read all these books, for pleasure as well as insight.

Our solution to the lotto ticket set covering problem is presented in more detail in [YS96].

## 1.7 Exercises

### Finding Counterexamples

1-1. *[3]* Show that $a + b$ can be less than $\min(a, b)$.

1-2. *[3]* Show that $a \times b$ can be less than $\min(a, b)$.

1-3. *[5]* Design/draw a road network with two points $a$ and $b$ such that the fastest route between $a$ and $b$ is not the shortest route.

1-4. *[5]* Design/draw a road network with two points $a$ and $b$ such that the shortest route between $a$ and $b$ is not the route with the fewest turns.

1-5. *[4]* The *knapsack problem* is as follows: given a set of integers $S = \{s_1, s_2, \ldots, s_n\}$, and a target number $T$, find a subset of $S$ which adds up exactly to $T$. For example, there exists a subset within $S = \{1, 2, 5, 9, 10\}$ that adds up to $T = 22$ but not $T = 23$.

Find counterexamples to each of the following algorithms for the knapsack problem. That is, giving an $S$ and $T$ such that the subset is selected using the algorithm does not leave the knapsack completely full, even though such a solution exists.

(a) Put the elements of $S$ in the knapsack in left to right order if they fit, i.e. the first-fit algorithm.

(b) Put the elements of $S$ in the knapsack from smallest to largest, i.e. the best-fit algorithm.

(c) Put the elements of $S$ in the knapsack from largest to smallest.

1-6. *[5]* The *set cover problem* is as follows: given a set of subsets $S_1, ..., S_m$ of the universal set $U = \{1, ..., n\}$, find the smallest subset of subsets $T \subset S$ such that $\cup_{t_i \in T} t_i = U$. For example, there are the following subsets, $S_1 = \{1, 3, 5\}$, $S_2 = \{2, 4\}$, $S_3 = \{1, 4\}$, and $S_4 = \{2, 5\}$ The set cover would then be $S_1$ and $S_2$.

Find a counterexample for the following algorithm: Select the largest subset for the cover, and then delete all its elements from the universal set. Repeat by adding the subset containing the largest number of uncovered elements until all are covered.

## Proofs of Correctness

1-7. *[3]* Prove the correctness of the following recursive algorithm to multiply two natural numbers, for all integer constants $c \geq 2$.

```
function multiply($y, z$)
    comment Return the product $yz$.
1.      if $z = 0$ then return(0) else
2.      return(multiply($cy, \lfloor z/c \rfloor$) + $y \cdot (z \bmod c)$)
```

1-8. *[3]* Prove the correctness of the following algorithm for evaluating a polynomial. P(x) $= a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

```
function horner($A, x$)
    $p = A_n$
    for $i$ from $n - 1$ to 0
        $p = p * x + A_i$
    return $p$
```

1-9. *[3]* Prove the correctness of the following sorting algorithm.

```
function bubblesort ($A$ : list[$1 \ldots n$])
    var int $i, j$
    for $i$ from $n$ to 1
        for $j$ from 1 to $i - 1$
            if ($A[j] > A[j + 1]$)
                swap the values of $A[j]$ and $A[j + 1]$
```

## Induction

1-10. *[3]* Prove that $\sum_{i=1}^{n} i = n(n + 1)/2$ for $n \geq 0$, by induction.

1-11. *[3]* Prove that $\sum_{i=1}^{n} i^2 = n(n + 1)(2n + 1)/6$ for $n \geq 0$, by induction.

1-12. *[3]* Prove that $\sum_{i=1}^{n} i^3 = n^2(n + 1)^2/4$ for $n \geq 0$, by induction.

1-13. *[3]* Prove that

$$\sum_{i=1}^{n} i(i + 1)(i + 2) = n(n + 1)(n + 2)(n + 3)/4$$

1-14. *[5]* Prove by induction on $n \geq 1$ that for every $a \neq 1$,

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1} - 1}{a - 1}$$

1-15. *[3]* Prove by induction that for $n \geq 1$,

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$$

1-16. *[3]* Prove by induction that $n^3 + 2n$ is divisible by 3 for all $n \geq 0$.

1-17. *[3]* Prove by induction that a tree with $n$ vertices has exactly $n - 1$ edges.

1-18. *[3]* Prove by mathematical induction that the sum of the cubes of the first $n$ positive integers is equal to the square of the sum of these integers, i.e.

$$\sum_{i=1}^{n} i^3 = (\sum_{i=1}^{n} i)^2$$

## Estimation

1-19. *[3]* Do all the books you own total at least one million pages? How many total pages are stored in your school library?

1-20. *[3]* How many words are there in this textbook?

1-21. *[3]* How many hours are one million seconds? How many days? Answer these questions by doing all arithmetic in your head.

1-22. *[3]* Estimate how many cities and towns there are in the United States.

1-23. *[3]* Estimate how many cubic miles of water flow out of the mouth of the Mississippi River each day. Do not look up any supplemental facts. Describe all assumptions you made in arriving at your answer.

1-24. *[3]* Is disk drive access time normally measured in milliseconds (thousandths of a second) or microseconds (millionths of a second)? Does your RAM memory access a word in more or less than a microsecond? How many instructions can your CPU execute in one year if the machine is left running all the time?

1-25. *[4]* A sorting algorithm takes 1 second to sort 1,000 items on your local machine. How long will it take to sort 10,000 items…

(a) if you believe that the algorithm takes time proportional to $n^2$, and

(b) if you believe that the algorithm takes time roughly proportional to $n \log n$?

## Implementation Projects

1-26. *[5]* Implement the two TSP heuristics of Section 1.1 (page 5). Which of them gives better-quality solutions in practice? Can you devise a heuristic that works better than both of them?

1-27. *[5]* Describe how to test whether a given set of tickets establishes sufficient coverage in the Lotto problem of Section 1.6 (page 23). Write a program to find good ticket sets.

### Interview Problems

1-28. *[5]* Write a function to perform integer division without using either the `/` or `*` operators. Find a fast way to do it.

1-29. *[5]* There are 25 horses. At most, 5 horses can race together at a time. You must determine the fastest, second fastest, and third fastest horses. Find the minimum number of races in which this can be done.

1-30. *[3]* How many piano tuners are there in the entire world?

1-31. *[3]* How many gas stations are there in the United States?

1-32. *[3]* How much does the ice in a hockey rink weigh?

1-33. *[3]* How many miles of road are there in the United States?

1-34. *[3]* On average, how many times would you have to flip open the Manhattan phone book at random in order to find a specific name?

## Programming Challenges

These programming challenge problems with robot judging are available at *http://www.programming-challenges.com* or *http://online-judge.uva.es.*

1-1. "The $3n + 1$ Problem" – Programming Challenges 110101, UVA Judge 100.

1-2. "The Trip" – Programming Challenges 110103, UVA Judge 10137.

1-3. "Australian Voting" – Programming Challenges 110108, UVA Judge 10142.