SECTION 7.3 Divide-and-Conquer Algorithms and Recurrence Relations

2. The recurrence relation here is $f(n) = 2f(n/2) + 2$, where $f(1) = 0$, since no comparisons are needed for a set with 1 element. Iterating, we find that $f(2) = 2 \cdot 0 + 2 = 2$, $f(4) = 2 \cdot 2 + 2 = 6$, $f(8) = 2 \cdot 6 + 2 = 14$, $f(16) = 2 \cdot 14 + 2 = 30$, $f(32) = 2 \cdot 30 + 2 = 62$, $f(64) = 2 \cdot 62 + 2 = 126$, and $f(128) = 2 \cdot 126 + 2 = 254$.

4. In this algorithm we assume that $a = (a_{2n-1}a_{2n-2} \cdots a_1a_0)_2$ and $b = (b_{2n-1}b_{2n-2} \cdots b_1b_0)_2$.

$n = 2$, the matrix is $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, and its determinant is clearly $d_2 = 4 - 1 = 3$. For $n = 3$ the matrix is

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

and we get $d_3 = 4$ after a little arithmetic. For the general case, our matrix is

$$\mathbf{A}_n = \begin{bmatrix} 2 & 1 & 0 & 0 & \dots & 0 \\ 1 & 2 & 1 & 0 & \dots & 0 \\ 0 & 1 & 2 & 1 & \dots & 0 \\ 0 & 0 & 1 & 2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 2 \end{bmatrix}.$$

To compute the determinant, we expand along the top row. This gives us a value of 2 times the determinant of the matrix obtained by deleting the first row and first column minus the determinant of the matrix obtained by deleting the first row and second column. The first of these smaller matrices is just $\mathbf{A}_{n-1}$, with determinant d_{n-1}. The second of these smaller matrices has just one nonzero entry in its first column, so we expand its determinant along the first column and see that it equals d_{n-2}. Therefore our recurrence relation is $d_n = 2d_{n-1} - d_{n-2}$, with initial conditions as computed at the start of this solution. If we compute a few more terms we are led to the conjecture that $d_n = n + 1$. If we show that this satisfies the recurrence, then we have proved that it is indeed the solution. And sure enough, $n + 1 = 2n - (n - 1)$. (Of course, we could have also dragged out the machinery of this section to solve the recurrence relation and initial conditions.)

46. Let a_n represent the number of goats on the island at the start of the n^{th} year.

a) The initial condition is $a_1 = 2$; we are told that at the beginning of the first year there are two goats. During each subsequent year (year n, with $n \geq 2$), the goats who were on the island the year before (year $n - 1$) double in number, and an extra 100 goats are added in. So $a_n = 2a_{n-1} + 100$.

b) The associated homogeneous recurrence relation is $a_n = 2a_{n-1}$, whose solution is $a_n^{(h)} = \alpha 2^n$. The particular solution is a polynomial of degree 0, namely a constant, $a_n = c$. Plugging this into the recurrence relation gives $c = 2c + 100$, whence $c = -100$. So the particular solution is $a_n^{(p)} = -100$ and the general solution is $a_n = \alpha 2^n - 100$. Plugging in the initial condition and solving for α gives us $2 = 2\alpha - 100$, or $\alpha = 51$. Hence the desired formula is $a_n = 51 \cdot 2^n - 100$. There are $51 \cdot 2^n - 100$ goats on the island at the start of the n^{th} year.

c) We are told that $a_1 = 2$, but that is not the relevant initial condition. Instead, since the first two years are special (no goats are removed), the relevant initial condition is $a_2 = 4$. During each subsequent year (year n, with $n \geq 3$), the goats who were on the island the year before (year $n - 1$) double in number, and n goats are removed. So $a_n = 2a_{n-1} - n$. (We assume that the removal occurs after the doubling has occurred; if we assume that the removal takes place first, then we'd have to write $a_n = 2(a_{n-1} - n) = 2a_{n-1} - 2n$.)

d) The associated homogeneous recurrence relation is $a_n = 2a_{n-1}$, whose solution is $a_n^{(h)} = \alpha 2^n$. The particular solution is a polynomial of degree 1, say $a_n = cn + d$. Plugging this into the recurrence relation and grouping like terms gives $(-c + 1)n + (2c - d) = 0$, whence $c = 1$ and $d = 2$. So the particular solution is $a_n^{(p)} = n + 2$ and the general solution is $a_n = \alpha 2^n + n + 2$. Plugging in the initial condition $a_2 = 4$ and solving for α gives us $4 = 4\alpha + 4$, or $\alpha = 0$. Hence the desired formula is simply $a_n = n + 2$ for all $n \geq 2$ (and $a_1 = 2$). There are $n + 2$ goats on the island at the start of the n^{th} year, for all $n \geq 2$.

48. **a)** This is just a matter of keeping track of what all the symbols mean. First note that $Q(n + 1) = Q(n)f(n)/g(n + 1)$. Now the left-hand side of the desired equation is $b_n = g(n + 1)Q(n + 1)a_n = Q(n)f(n)a_n$. The right-hand side is $b_{n-1} + Q(n)h(n) = g(n)Q(n)a_{n-1} + Q(n)h(n) = Q(n)(g(n)a_{n-1} + h(n))$. That the two sides are the same now follows from the original recurrence relation, $f(n)a_n = g(n)a_{n-1} + h(n)$. Note that

procedure *fast multiply*$(a, b$: nonnegative integers)
if $a \leq 1$ and $b \leq 1$ **then** *fast multiply*$(a, b) := ab$
else
begin
$A_1 := \lfloor a/2^n \rfloor$
$A_0 := a - 2^n A_1$
$B_1 := \lfloor b/2^n \rfloor$
$B_0 := b - 2^n B_1$
{we assume that these four numbers have length n; pad if necessary}
$x :=$ *fast multiply*(A_1, B_1)
answer := (x shifted left $2n$ places) + (x shifted left n places)
$x :=$ *fast multiply*(A_0, B_0)
answer := *answer* $+ x +$ (x shifted left n places)
if $A_1 \geq A_0$ **then** $A_2 := A_1 - A_0$ **else** $A_2 := A_0 - A_1$
if $B_0 \geq B_1$ **then** $B_2 := B_0 - B_1$ **else** $B_2 := B_1 - B_0$
$x :=$ *fast multiply*(A_2, B_2) shifted left n places
if $(A_1 \geq A_0 \wedge B_0 \geq B_1) \vee (A_1 < A_0 \wedge B_0 < B_1)$ **then**
answer := *answer* $+ x$
else *answer* := *answer* $- x$
fast multiply(a, b) := *answer*
end

6. The recurrence relation is $f(n) = 7f(n/2) + 15n^2/4$, with $f(1) = 1$. Thus we have, iterating, $f(2) = 7 \cdot 1 + 15 \cdot 2^2/4 = 22$, $f(4) = 7 \cdot 22 + 15 \cdot 4^2/4 = 214$, $f(8) = 7 \cdot 214 + 15 \cdot 8^2/4 = 1738$, $f(16) = 7 \cdot 1738 + 15 \cdot 16^2/4 = 13126$, and $f(32) = 7 \cdot 13126 + 15 \cdot 32^2/4 = 95{,}722$.

8. **a)** $f(2) = 2 \cdot 5 + 3 = 13$ **b)** $f(4) = 2 \cdot 13 + 3 = 29$, $f(8) = 2 \cdot 29 + 3 = 61$
c) $f(16) = 2 \cdot 61 + 3 = 125$, $f(32) = 2 \cdot 125 + 3 = 253$, $f(64) = 2 \cdot 253 + 3 = 509$
d) $f(128) = 2 \cdot 509 + 3 = 1021$, $f(256) = 2 \cdot 1021 + 3 = 2045$, $f(512) = 2 \cdot 2045 + 3 = 4093$, $f(1024) = 2 \cdot 4093 + 3 = 8189$

10. Since f increases one for each factor of 2 in n, it is clear that $f(2^k) = k + 1$.

12. An exact formula comes from the proof of Theorem 1, namely $f(n) = [f(1) + c/(a-1)]n^{\log_b a} - c/(a-1)$, where $a = 2$, $b = 3$, and $c = 4$ in this exercise. Therefore the answer is $f(n) = 5n^{\log_3 2} - 4$.

14. If there is only one team, then no rounds are needed, so the base case is $R(1) = 0$. Since it takes one round to cut the number of teams in half, we have $R(n) = 1 + R(n/2)$.

16. The solution of this recurrence relation for $n = 2^k$ is $R(2^k) = k$, for the same reason as in Exercise 10.

18. **a)** Our recursive algorithm will take a sequence of $2n$ names (two different names provided by each of n voters) and determine whether the two top vote-getters occur on our list more than $n/2$ times each, and if so, who they are. We assume that our list has the votes of each voter adjacent (the first voter's choices are in positions 1 and 2, the second voter's choices are in positions 3 and 4, and so on). Note that it is possible for more than two candidates to receive more than $n/2$ votes; for example, three voters could have choices AB, AC, and BC, and then all three would qualify. However, there cannot be more than three candidates qualifying, since the sum of four numbers each larger than $n/2$ is larger than $2n$, the total number of votes cast. If $n = 1$, then the two people on the list are both winners. For the recursive step, divide the list into two parts of even size—the first half and the second half—as equally as possible. As is pointed out in the hint

in Exercise 17, no one could have gotten a majority (here that means more than $n/2$ votes) on the whole list without having a majority in one half or the other, since if a candidate got approval from less than or equal to half of the voters in each half, then he got approval from less than or equal to half of the voters in all (this is essentially just the distributive law). Apply the algorithm recursively to each half to come up with at most six names (three from each half). Then run through the entire list to count the number of occurrences of each of those names to decide which, if any, are the winners. This requires at most $12n$ additional comparisons for a list of length $2n$. At the outermost stage of this recursion (i.e., when dealing with the entire list), we have to compare the actual numbers of votes each of the candidates in the running got, since only the top two can be declared winners (subject to the anomaly of three people tied, as illustrated above).

b) We apply the Master Theorem with $a = 2$, $b = 2$, $c = 12$, and $d = 1$. Since $a = b^d$, we know that the number of comparisons is $O(n^d \log n) = O(n \log n)$.

20. a) We compute $a^n \bmod m$, when n is even, by first computing $y := a^{n/2} \bmod m$ recursively and then doing one modular multiplication, namely $y \cdot y$. When n is odd, we first compute $y := a^{(n-1)/2}$ recursively and then do two multiplications, namely $y \cdot y \cdot a$. So if $f(n)$ is the number of multiplications required, assuming the worst, then we have essentially $f(n) = f(n/2) + 2$.

b) By the Master Theorem, with $a = 1$, $b = 2$, $c = 2$, and $d = 0$, we see that $f(n)$ is $O(n^0 \log n) = O(\log n)$.

22. a) $f(16) = 2f(4) + 4 = 2(2f(2) + 2) + 4 = 2(2 \cdot 1 + 2) + 4 = 12$

b) Let $m = \log n$, so that $n = 2^m$. Also, let $g(m) = f(2^m)$. Then our recurrence becomes $f(2^m) = 2f(2^{m/2}) + m$, since $\sqrt{2^m} = (2^m)^{1/2} = 2^{m/2}$. Rewriting this in terms of g we have $g(m) = 2g(m/2) + m$. Theorem 2 (with $a = 2$, $b = 2$, $c = 1$, and $d = 1$ now tells us that $g(m)$ is $O(m \log m)$. Since $m = \log n$, this says that our function is $O(\log n \cdot \log \log n)$.

24. To carry this down to its base level would require applying the algorithm three times, so we will show only the outermost step. The points are already sorted for us, and so we divide them into two groups, using x coordinate. The left side will have the first four points listed in it (they all have x coordinates less than 2.5), and the right side will have the rest, all of which have x coordinates greater than 2.5. Thus our vertical line will be taken to be $x = 2.5$. Now assume that we have already applied the algorithm recursively to find the minimum distance between two points on the left, and the minimum distance on the right. It turns out that $d_L = \sqrt{2}$ and $d_R = \sqrt{5}$, so $d = \sqrt{2}$. This is achieved by the points $(1, 3)$ and $(2, 4)$. Thus we want to concentrate on the strip from $x = 2.5 - \sqrt{2} \approx 1.1$ to $x = 2.5 + \sqrt{2} \approx 3.9$ of width $2d$. The only points in this strip are $(2, 4)$, $(2, 9)$, $(3, 1)$, and $(3, 5)$, Working from the bottom up, we compute distances from these points to points as much as $d = \sqrt{2} \approx 1.4$ vertical units above them. According to the discussion in the text, there can never be more than seven such computations for each point in the strip. In this case there is in fact only one, namely $\overline{(2,4)(3,5)}$. This distance is again $\sqrt{2}$, and it ties the minimum distance already obtained. So the minimum distance is $\sqrt{2}$.

26. In our algorithm d contains the shortest distance and is the value returned by the algorithm. We assume a function *dist* that computes Euclidean distance given two points (a, b) and (c, d), namely $\sqrt{(a-c)^2 + (b-d)^2}$. We also assume that some global preprocessing has been done to sort the points in nondecreasing order of x coordinates before calling this program, and to produce a separate list P of the points in nondecreasing order of y coordinates, but having an identification as to which points in the original list they are.

procedure *closest*$((x_1, y_1), \ldots, (x_n, y_n)$: points in the plane)
if $n = 2$ **then** $d := dist((x_1, y_1), (x_2, y_2))$
else
begin
$m := (x_{\lfloor n/2 \rfloor} + x_{\lceil n/2 \rceil})/2$
$d_L := closest((x_1, y_1), \ldots, (x_{\lfloor n/2 \rfloor}, y_{\lfloor n/2 \rfloor}))$
$d_R := closest((x_{\lceil n/2 \rceil}, y_{\lceil n/2 \rceil}), \ldots, (x_n, y_n))$
$d := \min(d_L, d_R)$
form the sublist P' of P consisting of those points whose x-coordinates are within d of m
for each point (x, y) in P'
for each point (x', y') in P' after (x, y) such that $y' - y < d$
if $dist((x, y), (x', y')) < d$ then $d := dist((x, y), (x', y'))$
end
{ d is the minimum distance between points in the list }

28. a) We follow the discussion given here. At each stage, we ask the question twice, "Is x in this part of the set?" if the two answers agree, then we know that they are truthful, and we proceed recursively on the half we then know contains the number. If the two answers disagree, then we ask the question a third time to determine the truth (the first person cannot lie twice, so the third answer is truthful). After we have detected the lie, we no longer need to ask each question twice, since all answers have to be truthful. If the lie occurs on our last query, however, then we have used a full $2 \log n + 1$ questions (the last 1 being the third question when the lie was detected).

b) Divide the set into four (nearly) equal-sized parts, A, B, C, and D. To determine which of the four subsets contains the first person's number, ask these questions: "Is your number in $A \cup B$?" and "Is your number in $A \cup C$?" If the answers are both "yes," then we can eliminate D, since we know that at least one of these answers was truthful and therefore the secret number is in $A \cup B \cup C$. By similar reasoning, if both answers are "no," then we can eliminate A; if the answers are first "yes" and then "no," then we can eliminate C; and if the answers are first "no" and then "yes," then we can eliminate B. Therefore after two questions we have a problem of size about $3n/4$ (exactly this when $4 \mid n$).

c) Since we reduce the problem to one problem of size $3n/4$ at each stage, the number $f(n)$ of questions satisfies $f(n) = f(3n/4) + 2$ when n is divisible by 4.

d) Using iteration, we solve the recurrence relation in part (c). We have $f(n) = 2 + f((3/4)n) = 2 + 2 + f((3/4)^2 n) = 2 + 2 + 2 + f((3/4)^3 n) = \cdots = 2 + 2 + \cdots + 2$, where there are about $\log_{4/3} n$ 2's in the sum. Noting that $\log_{4/3} n = \log n / \log 4/3 \approx 2.4 \log n$, we have that $f(n) \approx 4.8 \log n$.

e) The naive way is better, with fewer than half the number of questions. Another way to see this is to observe that after four questions in the second method, the size of our set is down to 9/16 of its original size, but after only two questions in the first method, the size of the set is even smaller (1/2).

30. The second term obviously dominates the first. Also, $\log_b n$ is just a constant times $\log n$. The statement now follows from the fact that f is increasing.

32. If $a < b^d$, then $\log_b a < d$, so the first term dominates. The statement now follows from the fact that f is increasing.

34. From Exercise 31 (note that here $a = 5$, $b = 4$, $c = 6$, and $d = 1$) we have $f(n) = -24n + 25n^{\log_4 5}$.

36. From Exercise 31 (note that here $a = 8$, $b = 2$, $c = 1$, and $d = 2$) we have $f(n) = -n^2 + 2n^{\log 8} = -n^2 + 2n^3$.