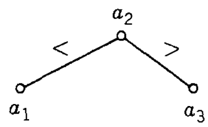
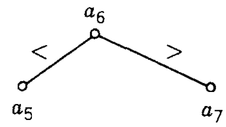
five vertices. There are five binary trees with four leaves: These are the binary trees with seven vertices.

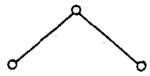
4.3 *(a)* Left subtree



Right subtree



(b) Left subtree



Right subtree

o

(C) Left subtree

o

Right subtree

o

(d) Left subtree

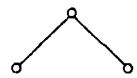
Right subtree

o

(e) Left subtree

o

Right subtree



4.4 The depth of the left subtree is $d - 1$ or less.

4.5 From Theorem 4.1, a full binary tree has 2^k vertices at depth k. In particular, a full binary tree with n or more leaves and depth k must have $n \leq 2^k$ or $\log(n) \leq \lceil \log(n) \rceil = k$. Thus a full binary tree of depth $\lceil \log(n) \rceil$ will have n or more leaves and by Corollary 4.2 will contain exactly $2^{\lceil \log(n) \rceil + 1} - 1$ vertices.

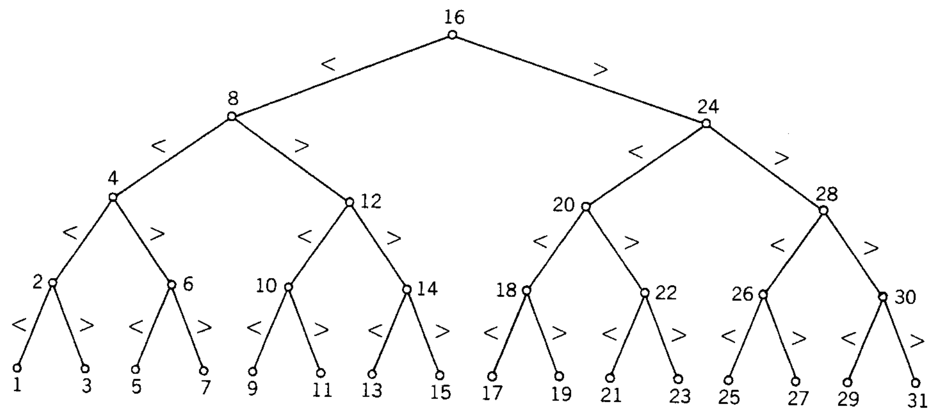
4.6

n	$k = \lfloor \log(n) \rfloor + 1$	$n' = 2^k - 1$
15	$\lfloor \log(15) \rfloor + 1 = 4$	$2^4 - 1 = 15$
26	$\lfloor \log(26) \rfloor + 1 = 5$	$2^5 - 1 = 31$
31	$\lfloor \log(31) \rfloor + 1 = 5$	$2^5 - 1 = 31$

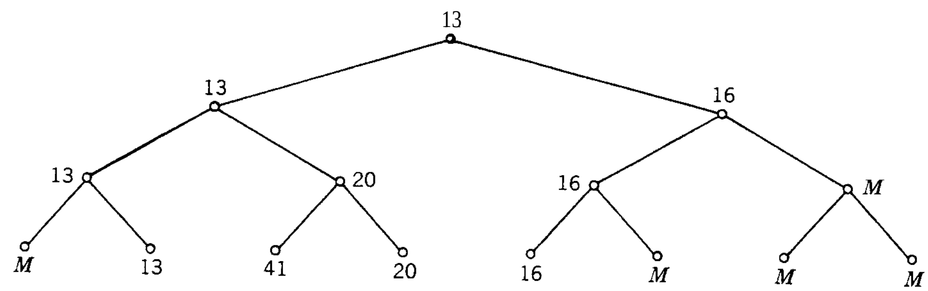
We must show in general that $n' \geq n$. We know that $n < 2^k$, or

$$n \leq 2^k - 1 = 2^{(\lfloor \log(n) \rfloor + 1)} - 1 = n'.$$

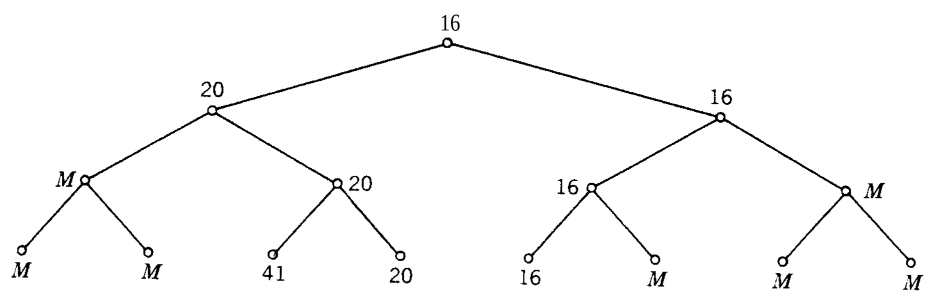
4.7 For convenience, we replace the label a_i with i. A binary search tree for a 23-element array follows. Note that the tree must have 31 vertices (see Question 4.6).



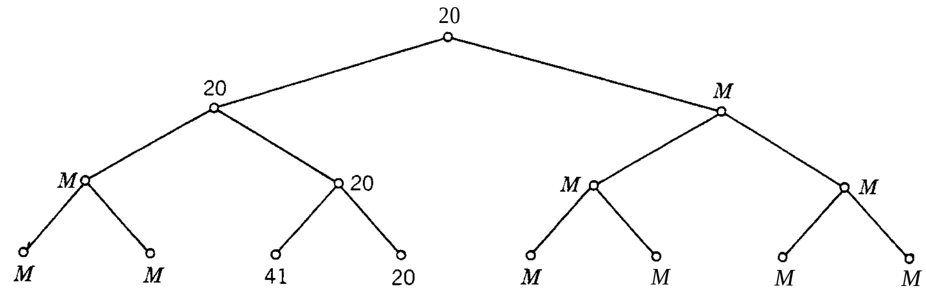
4.8 Labeled tree after the second execution of step 7:



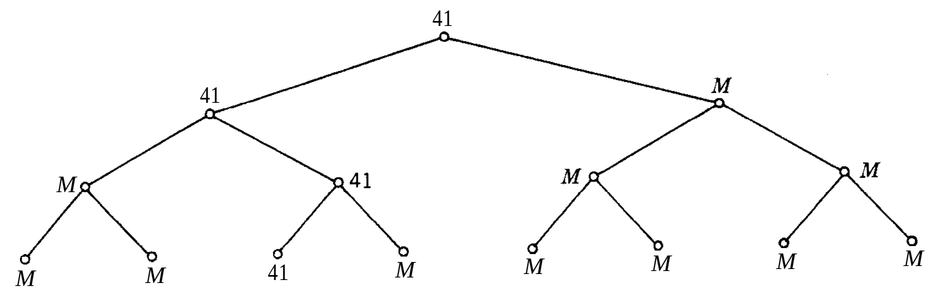
Labeled tree after the third execution of step 7:



Labeled tree after the fourth execution of step 7:



Labeled tree after the fifth execution of step 7:

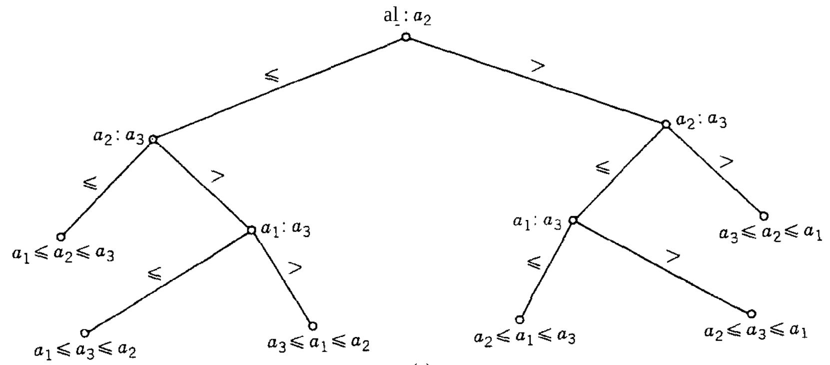


Result: $B = \langle b_1, b_2, b_3, b_4, b_5, b_6 \rangle = (7, 9, 13, 16, 20, 41)$.

SECTION 5

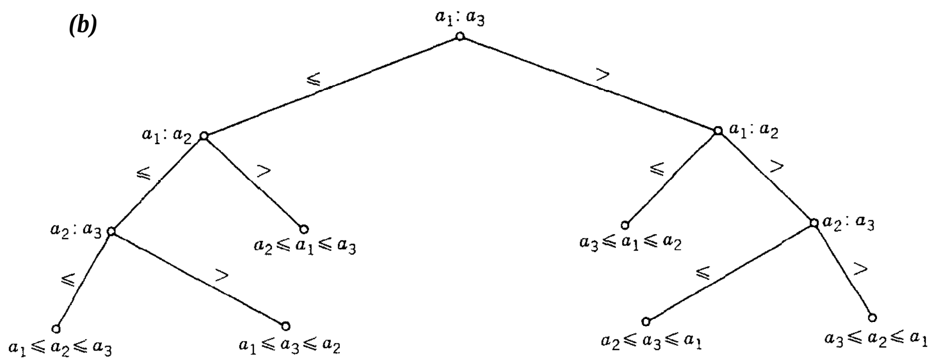
5.1 There are 3! = 6 orderings of the array $A = \langle a_1, a_2, a_3 \rangle$, one for each permutation on three elements, and $\binom{3}{2} = 3$ possible pairwise comparisons. For an array with four distinct elements, there are 4! = 24 possible orderings with $\binom{4}{2} = 6$ possible pairwise comparisons. Finally, an array with five distinct elements can be ordered in 5! = 120 different ways with $\binom{5}{2} = 10$ possible pairwise comparisons.

5.2 ***(a)***



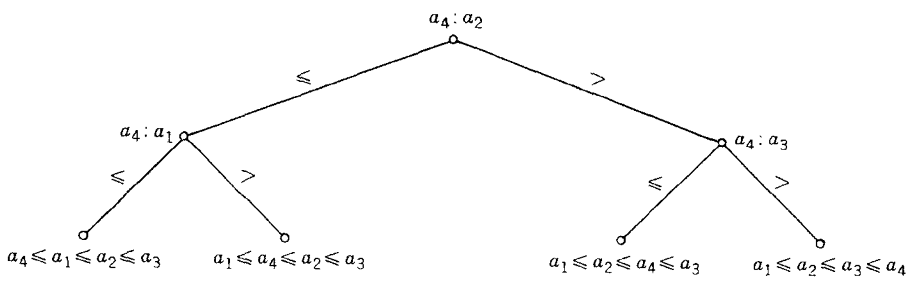
(a)

(b)



(b)

5.3 The graph shows a subtree of the binary search tree for a four-element array as specified by Example 5.3. In particular, the root of the subtree is labeled with $a_4 : a_2$ and we have assumed that $a_1 \leq a_2 \leq a_3$. If the whole binary search tree were drawn, it would look like the tree in Figure 6.6, only this subtree would be hanging in place of the leaf labeled "al $\leq a_2 \leq a_3$." And a similar subtree would be hanging under every other leaf. In the worst case five comparisons will be made.



5.4 A total of $5n \log(n)$ comparisons are made or roughly 27,000 comparisons when $n = 600$, roughly 61,000 comparisons with $n = 1200$, and roughly 97,000 comparisons with $n = 1800$.

SECTION 6

6.1 Trace of procedure MIN on the array <4,3,2,1, 5):

Step No.	n	k	a_k	a_{n+1}	
1	5				
2	4				
3	{Call MIN(⟨4, 3,2, 1⟩, 4, k)}				*(A)*
1	4				
2	3				
3	{Call MIN(⟨4, 3,2), 3, k)}				*(B)*
1	3				
2	2				
3	{Call MIN(⟨4, 3),2,/c)}				(c)
1	2				
2	1				
3	{Call MIN(⟨4⟩, 1, k)}				*(D)*
1	1	1			
5	{Return to *(D)*}				
	1	1	4	3	
4	1	2			
5	{Return to *(C)*}				
	2	2	3	2	
4	2	3			
5	{Return to *(B)*}				
	3	3	2	1	
4	3	4			
5	{Return to *(A)*}				
	4	4	1	5	
4	4	4			
5	{Return with k =4}				

6.2 Trace of procedure GCD run on b = 13 and c = 21:

Step No.	*b*	*c*	*r*	*9*	
1	13	21	?	?	
2	13	21	8		
3	{Call GCD(8, 13, g)}				*(A)*
2	8	13	5		
3	{Call GCD (5,8, g)}				*(B)*
2	5	8	3		
3	{Call GCD(3, 5, g)}				(c)
2	3	5	2		
3	{Call GCD(2, 3, g)}				*(D)*
2	2	3	1		
3	{Call GCD(1, 2, g)}				*(E)*
2	1	2	0		
3	{Call GCD(0, 1, g)}				*(F)*
1				1	
	{Return to (F)}			1	
	{Return to *(E)*}			1	
	{Return to *(D)*}			1	
	{Return to (C)}			1	
	{Return to *(B)*}			1	
	{Return to *(A)*}			1	

Result: gcd(13, 21) = 1. Note that the procedure GCD is called six times.

6.3 Trace of MIN on the array .4 = (– 1,0.333,5.2, – 10,6.001, 17)

(a) With start =2, finish = 3

Step No.	*start*	*finish*	a_{finish}	k	a_k	
1	2	3				
2	{Call **MIN** (A, 2,2, k)}					*(A)*
1	2	2		2		
4	{Return to (A)}					
	2	3	5.2	2	0.333	
3	2	3	5.2	2	0.333	
4	{Return with k = 2}					

Result: The index of the smallest entry in $\langle a_2, a_3 \rangle$ is *2*.

(*b*) With start =3, finish =6, A = (– 1,0.333,5.2, – 10,6.001, 17)

Step No.	*start*	*finish*	a_{finish}	k	‘k	
1	3	6				
2	{Call **MIN** (A, 3, 5, k)}					*(A)*
1	3	5				
2	{Call **MIN** (A, *3,4, k)}*					*(B)*
1	3	*4*				
2	{Call **MIN** (A, 3, 3, k)}					(c)
1	3	3		3		
4	{Return to (C)}					
	3	*4*	– 10	3	5.2	
3	3	*4*	– 10	4	– 10	
4	{Return to (B)}					
	3	5	6.001	4	– to	
3	3	5	6.001	4	– 10	
4	{Return to (A)}					
	3	6	17	4	– 10	
3	3	6	17	4	–l o	
4	{Return with k = 4}					

Result: The index of the smallest entry in $\langle a_3, a_4, a_5, a_6 \rangle$ is 4.

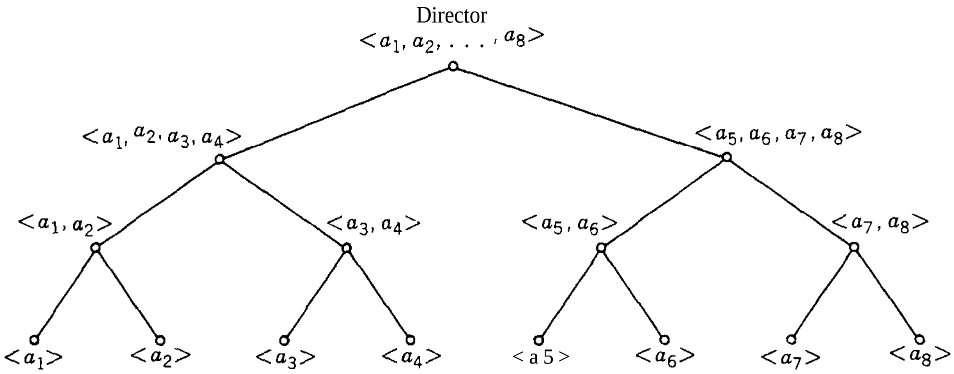
(c) With start = 1, finish = 6

Step No.	*start*	*finish*	a_{finish}	*k*	a_k	
1	1	6				
2	{Call MIN (A, 1,5, *k*)}					*(A)*
1	1	5				
2	{Call MIN (A, 1,4, *k*)}					(*B*)
1	1	4				
2	{Call MIN (*A*, 1,3, *k*)}					(c)
1	1	3				
2	{Call MIN (*A*, 1, 2, *k*)}					*(D)*
1	1	2				
2	{Call MIN (*A*, 1, 1, *k*)}					*(E)*
1	1	1		1		
4	{Return to (*E*)}					
	1	2	0.333	1	−1	
3	1	2	0.333	1	−1	
4	{Return to (*D*)}					
	1	3	5.2	1	−1	
3	1	3	5.2	1	−1	
4	{Return to (*C*)}					
	1	4	−10	1	−1	
3	1	4	− 10	*4*	− 10	
4	{Return to (*B*)}					
	1	5	6.001	*4*	− 10	
3	1	5	6.001	*4*	− 10	
4	{Return to *(A)*}					
	1	6	17	*4*	− 10	
3	1	6	17	*4*	− 10	
4	{Return with *k* = *4*}					

Result: The index of the smallest entry in the array *A* is 4.

SECTION 7

7.1 Each vertex of the following tree is labeled with the subarray of A to be sorted at that vertex.



The tree above has depth 3 and 15 vertices. Every vertex except the root corresponds with one assistant.

7.2 Trace of MERGE run on C = (0.1, 0.2,0.3,0,0.09,0.19, 0.29,0.39, 0.49) with start = 1, mid = 3, and finish = 9.

Step No.	*i*	*j*	*k*	*D*
1	1	4	1	
3	1	5	2	⟨0, . . .
	1	6	3	(o, 0.09,...
	2	6	4	<0,0.09,0.1,...
	2	7	5	<0,0.09,0.1,0.19,..,
	3	7	6	(0,0.09, 0.1,0.19,0.2,...
	3	8	7	⟨0, 0.09,0.1,0.19,0.2,0.29, .
	4	8	8	(O, 0.09,0.1,0.19,0.2,0.29, 0.3,...
10				(o, 0.09,0.1,0.19,0.2,0.29, 0.3,0.39, 0.49)
15	C = (0,0.09,0 .1, 0.19, 0.2, 0.29, 0 .3,0.39,0.49).			

7.3 (a) Trace of procedure MERGESORT run on the array C = (1, O) with start = 1 and finish = 2:

Step No.	c	*start*	*mid*	*finish*	
1,2	(1,0)	1	1	2	
3	{Call MERGESORT (C, 1, 1)}				(A)
1,2	(1)	1		1	
	{Return to (A)}	1	1	2	
4	{Call MERGESORT(C,2, 2)}				(B)
1,2	(o)	2		2	
	{Return to (B)}	1	1	2	
5	{Call MERGE (C, 1, 1,2)}	1	1	2	
	(o, 1)				
6	Return.				

(*b*) Trace of procedure MERGESORT run on the array *C* = (22,24, 23) with start = 1 and finish = 3:

Step No.	c	*start*	*mid*	*finish*	
1,2	<22,24, 23)	1	2	3	
3	{Call MERGESORT(C, 1, 2)				(*A*)
1,2	⟨22, 24)	1	1	2	
3	{Call MERGESORT(C, 1, 1)}				(*B*)
1,2	(22)	1		1	
	{Return to (*B*)}	1	1	2	
4	{Call MERGESORT (*C*, 2, 2)}				(*C*)
1,2	<24)	2		2	
	{Return to (C)}	1	1	2	
5	{Call MERGE (C, 1,1, 2)}	1	1	2	
	(22, 24⟩				
	{Return to (,4)}	1	2	3	
4	{Call MERGESORT (C, 3, 3)}	1	2	3	(D)
1,2	(23)	3		3	
	{Return to (D)}	1	2	3	
5	{Call MERGE (C, 1,2, 3)}	1	2	3	
	(22, 23, 24)				
6	Return.				

(c) Trace of procedure MERGESORT run on the array C = (1.1, 3.3,2.2, 4.4) with start = 1 and finish= 4:

Step No.	c	start	*mid*	*finish*	
1,2	(1.1,3.3,2.2,4.4)	1	2	4	
3	{Call MERGESORT(C, 1,2)}				*(A)*
1,2	<1.1,3.3)	1	1	2	
3	{Call MERGESORT (C, 1, 1)}				*(B)*
1,2	<1.1)	1		1	
	{Return to (B)}	1	1	2	
4	{Call MERGESORT(C,2,2)}	1	1	2	(c)
1,2	(3.3)	2		2	
	{Return to (C)}	1	1	2	
5	{Call MERGE(C, 1, 1,2)}	1	1	2	
	(1.1,3.3)				
	{Return to (A)}	1	2	*4*	
4	{Call MERGESORT(C, 3,4)}				*(D)*
1,2	(2.2, 4.4)	3	3	*4*	
3	{Call MERGESORT(C, 3, 3)}				*(E)*
1,2	⟨2.2⟩	3		3	
	{Return to *(E)}*	3	3	*4*	
4	{Call MERGESORT (C, 4, 4)}				*(F)*
1,2	(4.4)	4		*4*	
	{Return to (F)}	3	3	*4*	
5	{Call MERGE (C, 3, 3,4)}	3	3	*4*	
	(2.2, 4.4)				
	{Return to *(D)}*	1	2	4	
5	{Call MERGE (C, 1,2, 4)}	1	2	4	
	<1.1,2.2,3.3,4.4)				
6	Return.				

7.4 ***(a)*** Since Mergesort was called three times and Merge once on an array of size 2, $3 + 3 \cdot 2 = 9$ comparisons were performed. $9 = 3 \cdot 2 \cdot \log(2) + 2 \cdot 2 - 1$.
(c) Since Mergesort was called seven times and Merge three times on arrays of sizes 2, 2, and 4, $7 + 3(2 + 2 + 4) = 31$ comparisons were performed.
$31 = 3 \cdot 4 \cdot \log(4) + 2 \cdot 4 - 1$.

7.5 As in Question 7.3 (b) Mergesort is called five times and Merge is called twice on arrays of sizes 2 and 3. Thus $5 + 3(2 + 3) = 20$ comparisons are performed.
$3 \cdot 3 \cdot \log(3) + 2 \cdot 3 - 1 < 20 < 6 \cdot 3 \cdot \log(3) + 10 \cdot 3 - 1$.

SOLUTIONS TO QUESTIONS IN CHAPTER 7

SECTION 1

1.1 ***(a)*** 64, ***(b)*** 21, (c) 13, ***(d)*** 217, (e) 17.

1.2 ***(a)*** $S_4 : f_n = n!$, ***(b)*** $S_5 : g_n = \binom{n}{2}$ ***(c)*** $S_6 : h_n = 2^n - 1$,

(d) $$S_7 : j_n \begin{cases} n + 1 & \text{if } 1 \leq n \leq 2 \\ 2n - 1 & \text{if } 3 \leq n \leq 4 \\ 2n + 1 & \text{if } 5 \leq n \leq 6 \\ 2n + 3 & \text{if } n = 7 \end{cases}$$

is one of many "creative" solutions to the problem of finding a function that generates the first seven prime numbers. (e) $S_8 : k_n = n^2$,
(f) $S_9 : m_n = (-1)^{n-1} 3^{n-1}$

1.3 The functions $f_n = 2^{n+1} + 1$ and $g_n = 2n^2 - 2n + 5$ are two of many that produce the values $f_1 = g_1 = 5$, $f_2 = g_2 = 9$ and $f_3 = g_3 = 17$.

1.4 $S_1 : a_n = a_{n-1} + 1$ with $a_1 = 1$
$S_2 : a_n = 2a_{n-1}$ with $a_1 = 2$

1.5

$n =$	1	2	3	4	5	6	7	8
$a_n =$	0	*1*	3	6	10	15		
$M_n =$	1	2	2	3	3	3	3	4

1.6

$n=1$	2	3	4	5
$H'_n = 1$	$\frac{3}{2}$	$\frac{11}{6}$	$\frac{25}{12}$	$\frac{137}{60}$
$H''_n = 1$	$\frac{3}{2}$	$\frac{11}{6}$	$\frac{25}{12}$	$\frac{137}{60}$

We see that for $n = 1, 2, \ldots, 5$, $H'_n = H''_n$.

1.7

$n =$ 1	2	3	4	5
$C_n = 1$ 1	2	5	14	

SECTION 2

2.1 S_6 satisfies $a_n = 2a_{n-1} + 1$ as does 9, 19, 39, 79, 159, ... S_9 satisfies $a_n = (-3)a_{n-1}$ as does –6, 18, –54, 162, –486, ...

2.2 If one of a_1 and a_3 is unspecified, then a_5 and all subsequent "odd entries of the sequence will be undefined. Similarly, if one of a_2 and a_4 is unspecified, then a_6 and all subsequent "even" entries will be undefined.

$n = 1$	2	3	4	5	6	7	8	9	10
$a_n = 1$	1	1	1	2	2	3	3	5	5

The sequence listed above can be obtained from the Fibonacci sequence by listing each term twice. Since, by Theorem 4.3.1,

$$F_n = \frac{\phi^n - \phi'^n}{\sqrt{5}}$$

where

$$\phi = \frac{1+\sqrt{5}}{2} \text{ and } \phi' = \frac{1-\sqrt{5}}{2},$$

$$a_n = F_{\lceil n/2 \rceil} = \frac{\phi^{\lceil n/2 \rceil} - \phi'^{\lceil n/2 \rceil}}{\sqrt{5}}$$

2.3 (i) 2, (ii) 3, and (iii) 1.

2.4 (i) $a_n = na_{n-1} = n(n-1)a_{n-2} = n(n-1)(n-2)a_{n-3} = \cdots$
$= n(n-1)(n-2)\cdots(n-(n-2))a_1 = n(n-1)(n-2)\cdots 2\cdot 1 = n!$.

(ii) $b_n = b_{n-1} + 2 = b_{n-2} + 2 + 2 = b_{n-3} + 2 + 2 + 2 = \cdots$
$= b_{n-(n-1)} + 2 + 2 + \cdots + 2 \qquad \{n-1\ 2\text{s}\}$
$= b_1 + (n-1)2 = 1 + 2n - 2 = 2n - 1.$

2.5 (i) For the base case we have $a_1 = 1! = 1$. Then

$$\begin{aligned} a_{k+1} &= (k+1)a_k \\ &= (k+1)k! && \text{by the inductive hypothesis} \\ &= (k+1)!, && \text{as desired.} \end{aligned}$$

(ii) For the base case we have $b_1 = 2.1 - 1 = 1$. Then

$$\begin{aligned} b_{k+1} &= b_k + 2 \\ &= (2k-1) + 2 && \text{by the inductive hypothesis} \\ &= 2k + 1 = 2(k+1) - 1, && \text{as desired.} \end{aligned}$$

2.6 The relations $a_n = na_{n-2}, a_n = a_{n-1} + a_{n-3}, a_n = 2a_{\lfloor n/2 \rfloor}$, and $a_n = na_{n-1}$ are all homogeneous, since each is satisfied by the sequence that is identically 0. The relation $b_n = b_{n-1} + 2$ is inhomogeneous, since when we replace each b_j with 0, the result, $0 = 0 + 2$, is not valid.

SECTION 3

3.1 (i) Not: inhomogeneous; (ii) not: not linear and (iii) not: inhomogeneous.

3.2 (i) The characteristic equation is $x - 2 = 0$ and the characteristic root is $q_1 = 2$. (ii) The characteristic equation is $x^2 - x - 6 = 0$ and the characteristic roots are $q_1 = 3$ and $q_2 = -2$. (iii) The characteristic equation is $x^2 - 2x + 1 = 0$ and the characteristic root is $q_1 = 1$.

3.3 The base cases are $a_1 = 2^1 - 1 = 1$ and $a_2 = 2^2 - 1 = 3$. The inductive hypothesis is that $a_k = 2^k - 1$. We substitute this in the given recurrence relation

$$\begin{aligned} a_{k+1} &= 3a_k - 2a_{k-1} \\ &= 3(2^k - 1) - 2(2^{k-1} - 1) \\ &= 3 \cdot 2^k - 3 - 2^k + 2 \\ &= 2^k(3-1) - 1 \\ &= 2^{k+1} - 1, && \text{as desired.} \end{aligned}$$

3.4 (i) $a_0 = 0$, (ii) $a_0 = 1$, and (iii) $a_0 = 0$.

3.5 (i) From Question 3.2 the characteristic equation for *an* = *an*-~ + $6a_{n-2}$ is X^2— $x - 6 = O$ with characteristic roots $q_1 = 3$ and $q_2 = -2$. Thus the general formula that solves this recurrence relation is given by an = $c3^n + d(-2)$" for some constants c and d. We determine the constants c and d from the initial conditions:

$$2 = a_0 = c3^0 + d(-2)^0 = c + d$$
$$1 = a_1 = c3^1 + d(-2)^1 = 3c - 2d.$$

Adding twice the first equation to the second, we obtain $5c = 5$. Thus c = 1 and then $d = 1$ by substitution. Thus a_n = 3* + (– 2)".

(ii) We have accomplished most of the work for this part of the problem above; the only difference is in the initial conditions. Hence we have

$$1 = a_0 = c3^0 + d(-2)^0 = c + d$$
$$3 = a_1 = c3^1 + d(-2)^1 = 3c - 2d.$$

Again adding twice the first equation to the second we obtain 5C = 5. Thus $c = 1$ and $d = 0$. Thus $a_n = 3^n$.

(iii) From Question 3.2, the characteristic equation for $a_n = 2a_{n-1} - a_{n-2}$ is $x^2 - 2x + 1 = O$, which has a root of multiplicity 2.

SECTION 4

4.1 For the base cases we have $b_1 = 1$ and $b_2 = 2$. Then

$$\begin{aligned} b_{k+1} &= 2b_k - b_{k-1} && \text{the given recurrence} \\ &= 2k - (k-1) && \text{by the inductive hypothesis} \\ &= k + 1, && \text{as desired.} \end{aligned}$$

4.2 For p(x) = X^2– 2x + 1, we construct $D(x)$ as follows:

$$\begin{aligned} D(x) &= \frac{x^2 - 2x + 1 - (q^2 - 2q + 1)}{x - q} \\ &= \frac{(x^2 - q^2) - 2(x - q) + (1 - 1)}{x - q} && \text{by regrouping} \\ &= \frac{(x - q)(x + q) - 2(x - q)}{x - q} && \text{by algebra} \\ &= x + q - 2 && \text{by division.} \end{aligned}$$

4.3 The characteristic equation of $b_n = 4b_{n-1} - 4b_{n-2}$ *is* $x^2 - 4x + 4 = O$ and its characteristic root is $q_1 = 2$. If $b_k = 2^k$, then

$$4b_{n-1} - 4b_{n-2} = 4 \cdot 2^{n-1} - 4 \cdot 2^{n-2}$$
$$.\ \ 2^{n+1} - 2^n = 2^n(2-1) = 2^n = b_n.$$

If $b_k = k2^k$, then

$$4b_{n-1} - 4b_{n-2} = 4(n-1)2^{n-1} - 4(n-2)2^{n-2}$$
$$= 2^n[2(n-1) - (n-2)] = n2^n = b_n.$$

4.4 The characteristic equation of $c_n = -3c_{n-1} - 3c_{n-2} - c_{n-3}$ is

$$x^3 + 3x^2 + 3x + 1 = (x+1)^3 = 0$$

and the characteristic root is $q_1 = -1$ which has multiplicity y 3.
If $c_k = (-1)^k$, then

$$\begin{aligned} -3c_{n-1} - 3c_{n-2} - c_{n-3} &= -3(-1)^{n-1} - 3(-1)^{n-2} - (-1)^{n-3} \\ &= (-1)^{n-3}[-3(-1)^2 - 3(-1) - 1] \\ &= (-1)^{n-3}[-3 + 3 - 1] \\ &= (-1)^{n-3}(-1) = (-1)^{n-2} = (-1)^n = c_n. \end{aligned}$$

If $c_k = k(-1)^k$, then

$$\begin{aligned} & -3c_{n-1} - 3c_{n-2} - c_{n-3} \\ &= -3(n-1)(-1)^{n-1} - 3(n-2)(-1)^{n-2} - (n-3)(-1)^{n-3} \\ &= (-1)^{n-3}[-3(n-1)(-1)^2 - 3(n-2)(-1) - (n-3)] \\ &= (-1)^{n-3}[-3n + 3 + 3n - 6 - n + 3] \\ &= (-1)^{n-3}[-n] = n(-1)^{n-2} = n(-1)^n = c_n. \end{aligned}$$

Finally, if $c_k = k^2(-1)^k$, then

$$\begin{aligned} & -3c_{n-1} - 3c_{n-2} - c_{n-3} \\ &= -3(n-1)^2(-1)^{n-1} - 3(n-2)^2(-1)^{n-2} - (n-3)^2(-1)^{n-3} \\ &= (-1)^{n-3}[-3(n^2 - 2n + 1) + 3(n^2 - 4n + 4) - (n^2 - 6n + 9)] \\ &= (-1)^{n-3}[-3n^2 + 6n - 3 + 3n^2 - 12n + 12 - n^2 + 6n - 9] \\ &= (-1)^{n-3}[-n^2] = (-1)^{n-2}n^2 = n^2(-1)^n = c_n. \end{aligned}$$

4.5 From Theorem 4.2, a solution of the recurrence relation given in Question 4.3 is of the form

$$a_n = c_1(-1)^n + c_2 n(-1)^n + c_3 n^2(-1)^n.$$

With the initial conditions $a_0 = 1$, $a_1 = -2$ and $a_2 = 1$, we can solve for the constants c_1, C_2, and C_3:

$$\begin{aligned} 1 &= a_0 = c_1 + 0c_2 + 0c_3 \\ -2 &= a_1 = -c_1 - c_2 - c_3 \\ 1 &= a_2 = c_1 + 2c_2 + 4c_3 \end{aligned}$$

The solution to this system of equations is $c_1 = 1$, $c_2 = 2$, and $C_3 = -1$. Thus a solution to the recurrence relation with the given initial conditions is $a_n = (-1)^n + 2n(-1)^n - n^2(-1)^n$.

SECTION 5

5.1 Reread Section 6.2.

$n =$	2	3	4	5
$B_n =$	7	7	10	10

We note for $n = 2, 3, 4$, and 5 that $B_n = 3\lfloor \log(n) \rfloor + 4$.

5.2 Reread Exercise 7 in Chapter 6, Section 7.

$n =$	2	4	8
$M_n =$	9	31	87
$3n\log(n) + 2n - 1 =$	9	31	87

5.3 (*a*) $k = 1$, $d = 2$, $c = 0$ and $e = 3$. (*b*) $k = 1$, $d = 2$, $c = 0$ and $e = 2$. (*c*) $k = 2$, $d = 2$, $c = 3$ and $e = 1$.

5.4 1 initial condition. If $a_0 = 1$, then

$n =$	1	2	3	4	5	6	7
$a_n =$	2	2	3	3	3	3	3

5.5 The proof is by induction on i, where $n = 2^i$. The base case is $i = 0$: $a_{2^0} = a_1 + \log(1)c = a_1 + 0 = a_1$. Assuming the result for $i = k$, let $i = k + 1$.

$$\begin{aligned}
a_n &= a_{2^{k+1}} && \text{since } n = 2^i = 2^{k+1} \\
&= a_{\lfloor n/2 \rfloor} + c && \text{by the recurrence relation} \\
&= a_{2^k} + c && \text{since } n/2 = 2^k \\
&= a_1 + \log(2^k)c + c && \text{by the inductive hypothesis} \\
&= a_1 + kc + c = a_1 + (k+1)c \\
&= a_1 + \log(2^{k+1})c && \text{by properties of log} \\
&= a_1 + \log(n)c.
\end{aligned}$$

5.6 With c = 2, Theorem 5.1 implies that $C_n \le 2\lfloor \log(n) \rfloor + 4$. This is the same result as Theorem 3.1 from Chapter 6.

5.7
$$\begin{aligned}
M_{2^k} &\le 2M_{(2^k)/2} + 4 \cdot 2^k \quad \text{by (C)} \\
&= 2M_{2^{k-1}} + 2^{k+2}. \\
&\le 2(2M_{2^{k-2}} + 2^{k+1}) + 2^{k+2} \\
&= 2^2 M_{2^{k-2}} + 2^{k+2} + 2^{k+2} \\
&= 2^2 M_{2^{k-2}} + 2 \cdot 2^{k+2} \\
&\ \ \cdots \\
&\le 2^i M_{2^{k-i}} + i2^{k+2} \\
&\ \ \cdots \\
&\le 2^k M_{2^{k-k}} + k2^{k+2} \\
&= 2^k M_1 + 4(2^k)k
\end{aligned}$$

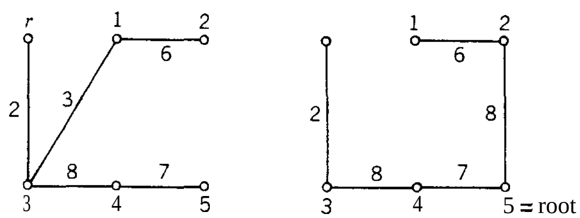
Since $\log(2^k) = k$ and $n = 2^k$, the previous expression can be rewritten as $nM_1 + 4n\log(n)$. Next we verify this formula by induction on k, where $n = 2^k$. That is, if $n = 2^k$ and M_n satisfies (C'), then we must show that $M_n \le 4n\log(n) + M_1 n$. For the base case $k = 0$ and $n = 2^0 = 1$. Then $M_n = M_1 \le 4 \cdot 1 \cdot 0 + M_1 \cdot 1 = M_1$. We assume the result for $n = 2^k$ and check $n = 2^{k+1}$:

$$\begin{aligned}
M_n &= M_{2^{k+1}} \\
&\le 2M_{2^k} + 4 \cdot 2^{k+1} && \text{by } (C') \\
&\le 2(4 \cdot 2^k k + M_1 2^k) + 4 \cdot 2^{k+1} && \text{by the inductive hypothesis} \\
&= 4 \cdot 2^{k+1} k + M_1 2^{k+1} + 4 \cdot 2^{k+1} \\
&= 4 \cdot 2^{k+1}(k+1) + M_1 2^{k+1} \\
&= 4n\log(n) + M_1 n.
\end{aligned}$$

SOLUTIONS TO QUESTIONS IN CHAPTER 8

SECTION 1

1.1 The union of shortest paths forms a minimum-distance spanning tree, shown in *(a)*. With the root specified to be 5, we obtain the different tree shown in (b).



(a) *(b)*

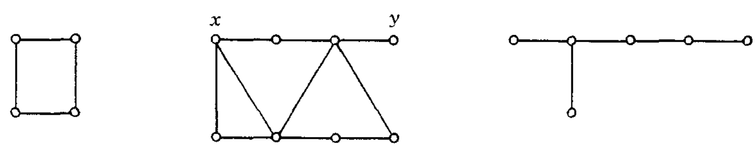
1.2 Here is a trace of DIJKSTRA on the weighted graph from Figure 8.4.

Step	*No.*	*j*	*z*	*V(T)*	*E(T)*
2		?	?	$\{r\}$	$\varnothing$
4		1	t		
5		1	t	$\{r, t\}$	$\{(r, t)\}$
4		2	b		
5		2	b	$\{r, t, b\}$	$\{(r, t), (r, b)\}$
4		3	s		
5		3	s	$\{r, t, b, s\}$	$\{(r, t), (r, b), (t, s)\}$
4		4	C		
5		4	c	$\{r, t, b, s, c\}$	$\{(r, t), (r, b), (t, s), (r, c)\}$
4		5	a		
5		5	a	$\{r, t, b, s, c, a\}$	$\{(r, t), (r, b), (t, s), (r, c), (b, a)\}$
4		6	q		
5		6	q	$\{r, t, b, s, c, a, q\}$	$\{(r, t), (r, b), (t, s), (r, c), (b, a), (a, q)\}$
4		7	p		
5		7	p	$\{r, t, b, s, c, a, q, p\}$	$\{(r, t), (r, b), (t, s), (r, c), (b, a), (a, q), (q, p)\}$

1.3 If G is not connected, then step 4 cannot be executed $V - 1$ times as required. If some edge weights were negative, then the minimum distance from the root to the first attached vertex x might be less than the weight of the first edge e = (r, x). Thus the tree T in the base case might not have a shortest path in it. Later in the proof, when we add the edge (u, x), we claim that x is closer to the root than U. This would not be true if the weight of (u, x) were negative. If DIJKSTRA is run on the graph in Figure 8.5, then the distance from the root to any of the other vertices is not well defined, since every time you traverse a cycle around the triangle you add a total of – 1 to your path length.

SECTION 2

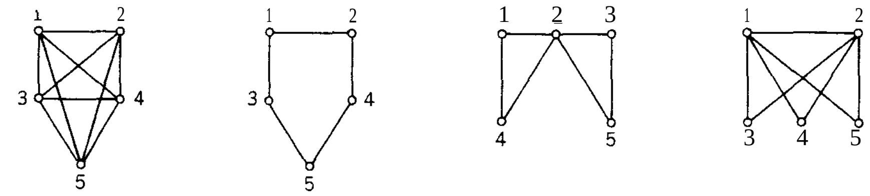
2.1 The graph shown in *(a) is* the only Eulerian graph with four vertices. The graph in (*b*) has an Eulerian path from x to y but is not Eulerian. The simplest such graph would just be a path with eight vertices.



(a) (b) (c)

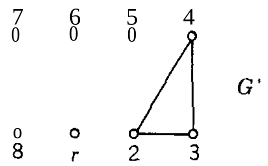
2.2 The first and last graphs are Eulerian (See Theorem 2.1). The second and fourth contain Eulerian paths but not Eulerian cycles.

2.3 A graph with four vertices all of whose degrees are even must have every degree either O or 2. To be connected there cannot be any vertices of degree O. There is only one graph, the 4-cycle, which is Eulerian. Similarly, a graph with five vertices must have every degree either 2 or 4. We list them together with one Eulerian cycle.

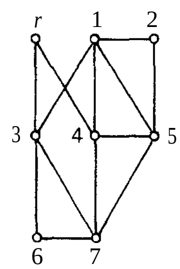


<1,2, 4, 5, 3, 1, 4,3,2,5, 1>   <1,2,4,3,5, 1>   <1,2, 3, 5, 2,4, 1>   <1,3,2,4, 1, 5,2, 1>

2.4

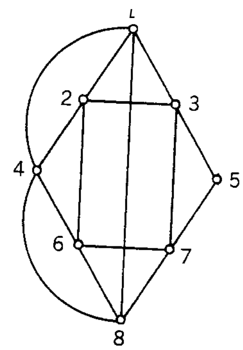


2.5 The vertices 3 and 4 have odd degree. To construct G' *we* create vertex *r*. adjacent to 3 and 4 as shown in the following figure.



(4, r, 3,6,7,5,2,1,5,4,7,3, 1, 4) is an Eulerian cycle in G'. Removing *r* and its incident edges produces an Eulerian path from 3 to 4.

2.6 For convenience we label the graph.



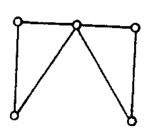
Step No.	*x*	*C*	*D*
1	1	⟨1⟩	
3	1		
4	1		
6		⟨1, 2, 3, 1,4,6,7,8, 1⟩	⟨1, 2, 3, 1, 4, 6, 7, 8, 1⟩
3	*2*		
4	*2*		
6		*(1,2,4,8,6,2,3 >1,4,6,7>8>1)*	⟨2, *4,8, 6,2)*
3	*3*		
4			
6		(1,2,4,8> 6,2,3,5,7,3,1,4,6,7,8,1)	(3, 5,7, 3)

2.7 (1, *2,4, 3,1)* is an Eulerian cycle in the first graph. The second graph does not contain an Eulerian path or cycle. (2,3, 1,2,4,3) is an Eulerian path in the third graph as is (2,4,3, 1,2, 3). The fourth graph contains lots of Eulerian cycles, for example, (1, 2,1,3,4,2,4,3,1 *)*.

SECTION 3

3.1 The first and third do; (1, 4, 2, *3,1)* and (1,2,3,4,5,1), respectively.

3.2



3.3

Step No.	z	P	c
2		$\emptyset$	
4	x	$\langle x, v, t, w, y\rangle$	
5			$\langle x, v, t, w, y, x\rangle$
4	u	$\langle u, v, t, w, y, x\rangle$	
5			$\langle u, t, w, y, x, v, u\rangle$

3.4 First graph

Step No.	J	K	T	$E(T)$
1	1		$\{1\}$	$\emptyset$
3	1	2		
4–6	2	2	$\{1,2\}$	$\{(1,2)\}$
3	2	3		
4-6	3	3	$\{1,2,3\}$	$\{(1,2), (2, 3)\}$
3	3	4		
4-6	4	4	$\{1,2,3,4\}$	$\{(1,2), (2, 3), (3,4)\}$
3	4	{no K}		
7	3			
3	3	{no K}		
7	2			
3	2	5		
4-6	5	5	$\{1,2,3,4,5\}$	$\{(1, 2), (2, 3), (3, 4), (2, 5)\}$
9 STOP				

Second graph

Step No.	J	K	T	$E(T)$
1	1		$\{1\}$	$\emptyset$
3	1	2		
4-6	2	2	$\{1,2\}$	$\{(1,2)\}$
3	2	6		
4-6	6	6	$\{1, 2, 6\}$	$\{(1, 2), (2, 6)\}$
3	6	{no K}		
7	2			
3	2	{no K}		
7	1			
3	1	{no K}		
8 STOP			$\{1.2,6\}$	$\{(1,2), (2,6)\}$

3.5 First graph

Step No.	*J*	*K*	*FOR WARD*	*PATH*
Main 1	2		TRUE	**(1,0,0,0,0)**
Build 1	2	2		
3-5	3	2	TRUE	⟨1, 2, 0, 0, 0⟩
Build 1	3	3		
3-5	*4*	3	TRUE	(1,2,3,0,0)
Build 1,2	4	{no *K*}	FALSE	
Main 4, 5	3			(1,2,3,0,0)
Build 1	3	4		
3–5	4	4	TRUE	⟨1, 2, 4, 0, 0⟩
Build 1, 2	4	{no *K*}	FALSE	
Main 4, 5	3			⟨1, 2, 4, 0, 0⟩
Build 1, 2	3	{no *K*}	FALSE	
Main 4, 5	2			⟨1, 2, 0, 0, 0⟩
Build 1	2	3		
3–5	3	3	TRUE	(1,3,0,0,0)
Build 1	3	2		
3-5	4	2	TRUE	<1,3,2,0,0)
Build 1	4	4		
3-5	5	4	TRUE	(1,3,2,4,0)
Main 7, 8				(1,3,2,4,1)
Main 9 STOP				

Second graph

Step No.	*J*	*K*	*FOR WARD*	*PATH*
Main 1	2		TRUE	**(1.0,0,0,0)**
Build 1	2	2		
3-5	3	2	TRUE	⟨1, 2, 0, 0. 0⟩
Build 1	3	3		
3–5	4	3	TRUE	(1,2,3,0,0)
Build 1,2	4	{no *K*}	FALSE	
Main 4, 5	3			(1,2,3,0.0)
Build 1	3	4		
3-5	4	4	TRUE	<1,2,4,0,0)
Build 1, 2	4	{no *K*}	FALSE	
Main 4, 5	3			⟨1, 2, 4. 0, 0⟩
Build 1,2	3	{no *K*}	FALSE	
Main 4, 5	2			(1,2,0,0,0)

continued

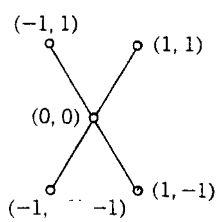
Second graph (continued)

Step No.	*J*	*K*	*FOR WARD*	*PATH*
Build 1	2	4		
3-5	3	4	TRUE	<1,4,0,0,0)
Build 1	3	2		
3-5	4	2	TRUE	(1,4,2,0,0)
Build 1	4	3		
3-4	5	3	TRUE	<1,4,2,3,0)
Main 7, 8			FALSE	
Main 4, 5	4			(1,4,2,3,0)
Build 1,2	4	{no *K*}	FALSE	
Main 4, 5	3			(1,4,2,0,0)
Build 1,2	3	{no *K*}	FALSE	
Main 4, 5	2			<1,4,0,0,0)
Build 1, 2	2	{no *K*}	FALSE	
Main 4, 5	1			(1,0,0,0,0)
Main 6 NO HAM CYCLE, STOP				

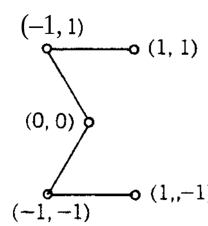
SECTION 4

4.1 Denote the locations by O = (0, O), *P* = (1, O), Q = (O, 1), and *R* = (1, 1). Since the drill must start and end at O, there are 3! = 6 possible drilling sequences: *OPRQ* and *OQRP* have total distance 4, *OPQR* and *ORQP* have total distance $2 + 2\sqrt{2}$ as do *ORPQ* and *OQPR*. Note that the second sequence of each of the preceding pairs is the reverse of the first.

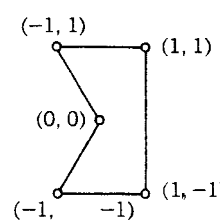
4.2



T(G)
$4\sqrt{2}$



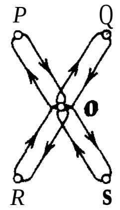
P (G)
$4 + 2\sqrt{2}$



H(G)
$6 + 2\sqrt{2}$

4.3 D might consist of $\langle O, P, O, Q, O, S, O, R, O\rangle$.

$D(T) =$

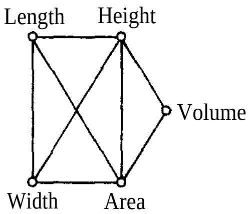


4.4 The Hamiltonian cycle C produced from D is $\langle O, P, Q, S, R, O\rangle$. $W(C) = 6 + 2\sqrt{2} = w(H(G))$.

SECTION 5

5.1 See Figure 8.26: Each interfering pair of variables is represented by an edge of this graph. Sets of four mutually interfering variables: {L, W, Wt, A}, {Wt, Vol, A, Cl}, {Ht, A, Cl, Vol}, {Wt, Vol, Cl, C2}. There is no set of five mutually interfering variables.

5.2



5.3 Adding edges may increase the chromatic number: If x and y are not adjacent in H, but are adjacent in G, then a coloring of H is a coloring of G unless x and y are assigned the same color. In that case, an additional color may be needed to color G. If the compiler assigns $k = \chi(G)$ memory locations to a program based on a coloring of G, then no two variables, joined by an edge in G, receive the same color and so are not assigned to the same memory location. Edges join every pair of "truly" interfering variables and maybe more.

5.4 (i) $\chi(G)$ = cl (G) =2; (ii) $\chi(G)$= cl (G) = 3; (iii) $\chi(G)$= cl (G) =3; (iv) $\chi(G)$= 3, cl (G) = 2; and (v) $\chi(G)$ = cl (G) = 4.

5.5 A graph is 1-colorable if and only if it does not contain an edge. An algorithm to 1-color could check that the graph contains no edges and then assign the color 1 to every vertex.

5.6 *(a)*

Step	*No.*	*I*	c_I	*J*	L_J
5	1	1			
7	1			2	⟨2⟩
7	1			6	(2, 3,4, 5,6)
7	1			7	<2,3,4,5,6, 7)
5	2	2			
7	2			3	(1,3)
7	2			7	<3,4,5,6, 7⟩
5	3	1			
7	3			4	(2,3,4)
7	3			7	⟨3, 4, 5,6, 7⟩
5	4	2			
7	4			5	(1,3,4,5)
7	4			7	(3,4,5,6,7)
5	5	1			
7	5			6	⟨2, 3,4, 5,6)
7	5			7	(3,4> 5,6,7)
5	6	2			
7	6			7	(3,4,5,6,7)
5	7	3			

(b)

Step	No.	I	c_I	*J*	L_J
5	1	1			
7	1			2	⟨2⟩
7	1			5	(2, 3,4, 5)
7	1			6	<2,3,4, 5,6)
5	2	2			
7	2			3	⟨1, 3⟩
7	2			4	<1,3,4)
7	2			5	(3,4,5)
7	2			6	(3,4, 5,6)
7	2			7	<1,3,4,5,6,7)
5	3	1			
7	3			4	(3, 4)
7	3			7	(3,4,5,6,7)
5	4	3			
7	4			5	<4, 5)
5	5	4			
5	6	3			
7	6			7	<4,5,6, 7)
5	7	4			

5.7 Here is a trace of BACKTRACKCOLOR applied to K_4 with N = 3. Let the vertices be x_1, x_2, x_3, and x_4.

Step No.	*J*	*K*	*FOR WARD*	C[l, 2, 3,4]
Main 1	2		TRUE	[1,0,0,0]
Color 1	2	2		
3–5	3	2	TRUE	[1,2,0,0]
Color 1	3	3		
3-5	4	3	TRUE	[1,2,3,0]
Color 1	4	4		
6	4	4	FALSE	
Main 4,5	3			[1, 2, 3,0]
Color 1	3	4		
6	3	4	FALSE	
Main 4,5	2			[1,2,0,0]
Color 1	2	3		
3-5	3	3	TRUE	[1,3,0,0]
Color 1	3	2		
3-5	4	2	TRUE	[1,3,2,0]
Color 1	4	4		
6	4	4	FALSE	
Main 4,5	3			[1,3,2,0]
Color 1	3	4		
6	3	4	FALSE	
Main 4,5	2			[1,3,0,0]
Color 1	2	4		
6	2	4	FALSE	
Main 4,5	1			[1,0,0,0]
6 THERE IS NO 3-COLORING OF G, STOP				

INDEX

ALGORITHMS AND PROCEDURES

NOTATIONS