CHAPTER

SEVEN

ANALYSIS OF ALGORITHMS

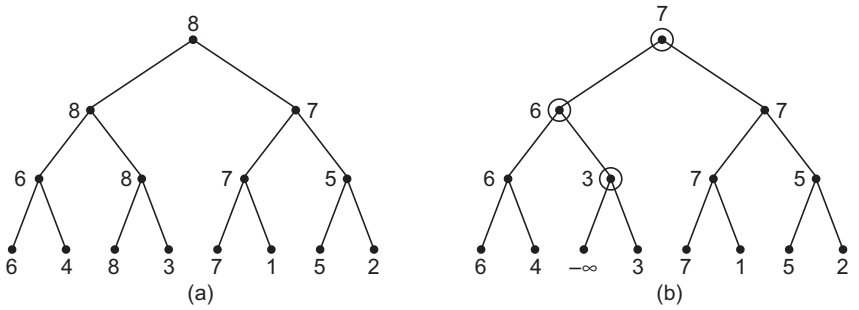
7.9 (a) For $i = 2, 3, \ldots, n$, compare ball 1 with ball i. If they are of different colors we can conclude that not all n balls are of the same color. The complexity of the algorithm is $n - 1$. The complexity of the problem is also $n - 1$, using essentially the same argument for the problem of finding the maximum of n numbers.

(b) Compare ball 1 with ball i for $i \geq 2$ until a different color ball is found. Let this be ball j. (If no ball j is found, the n balls are of the same color.) For $j < k \leq n$, compare ball k with both ball 1 and ball j. If the color of ball k is different from that of ball 1 and ball j, then the n balls are of three or more colors.

The complexity of the algorithm is $2(n - 2) + 1$. (The worst case happens when ball 1, 2, n are of three different colors.)

7.10 Use the bubble sort algorithm, the number of comparisons is $\dfrac{n(n-1)}{2}$.

7.11 Place the n numbers at the leaves of a binary tree. In $n - 1$ comparisons, we can determine the largest of n numbers as illustrated by the example in (a) of figure below. Replace the largest number found by $-\infty$ and repeat the



comparisons that involves the largest number. In $\lceil \lg n \rceil$ comparisons we can determine the second largest of the n numbers as illustrated by the example in (b) of the figure.

7.12 For $i = h - 1, h - 2, \ldots, 1$ do

For each internal node at height i do

Among all edges incident from it, pick the edge with the smallest weight, and add this weight to the edge incident into it.

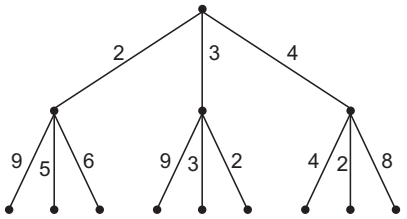
At the root, among all edges incident from it, pick the edge with the smallest weight.

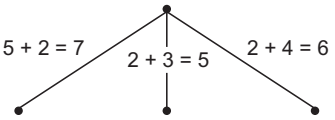
Total number of comparisons:

$$(m-1)\,(m^{h-1} + m^{h-2} + \ldots + m) + (m-1) = m^{h-1}$$

Total number of additions:

$$m^{h-1} + m^{h-2} + \ldots + m = \frac{m(m^{h-1}-1)}{m-1}$$





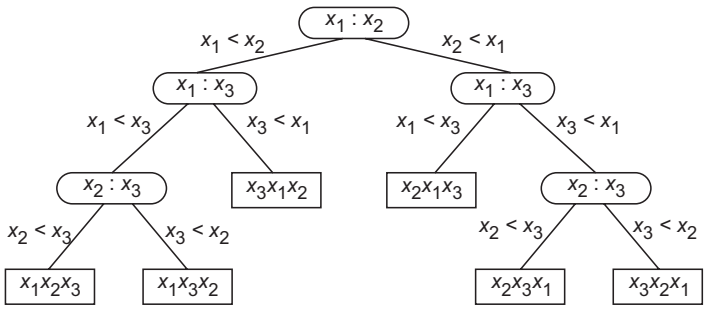
7.13 The first step takes $n - 2$ comparisons to pick the nearest of $n - 1$ neighbors.

The second step takes $n - 3$ comparisons to pick the nearest of $n - 2$ neighbors.

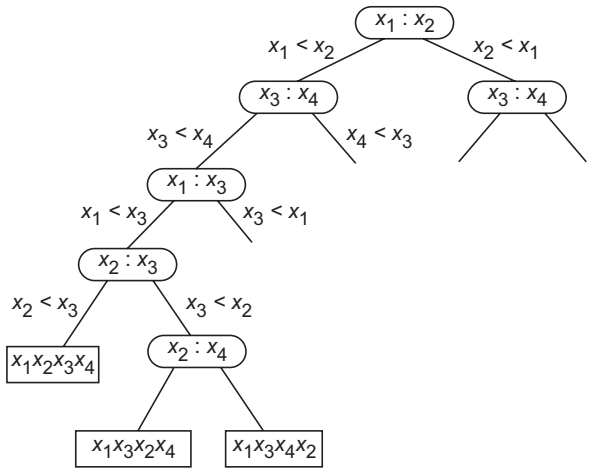
The $(n-2)^{\text{th}}$ step takes 1 comparison to pick the nearest of 2 neighbors.

Total number of comparisons $= (n-2) + (n-3) + \ldots + 1 = \dfrac{(n-1)\,(n-2)}{2}$.

7.14 (a) 3



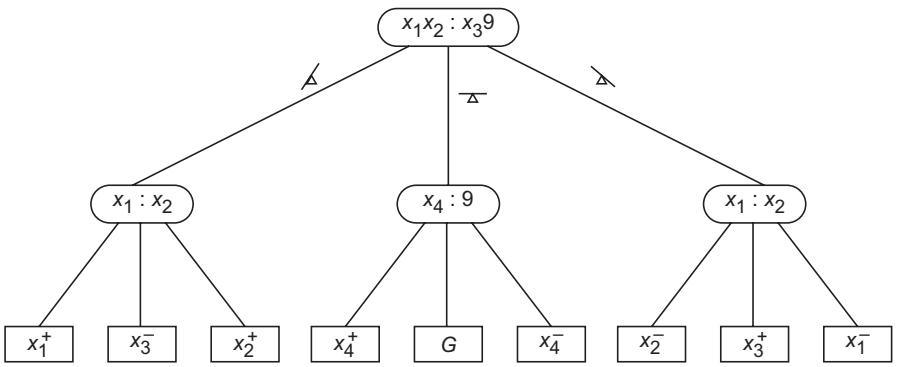
(b)



(c) No matter what algorithm is used, the corresponding binary tree must have $n!$ or more leaves. Thus the height of the tree, which is also the number of comparisons needed, is larger than or equal to $\lceil \lg n! \rceil$.

7.15 (a) Put the given coin on the left pan and a good coin on the right. Clearly, the given coin is good, light or heavy depending on whether the two pans balance, or the left pan is lighter or heavier, respectively. Also, the complexity of the problem is also 1, since we need at least 1 weighing.

(b) The following scheme for $n = 4$ requires at most 2 weighings.



The complexity of the problem is also 2 since 1 weighing can distinguish between at most 3 possible outcomes and there are 9 possible outcomes for $n = 4$.

(c) For a binary tree with k levels, there are 3^k leaves. There are $2n + 1$ possible outcomes for n coins, thus

$$2n + 1 \leq 3^k$$

$$n \leq \frac{3^k - 1}{2}$$

(d) We present the solution for $n = 13$. The 3×13 matrix

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

contains 13 column vectors with elements from the set $\{0, 1, 2\}$ such that no column is a multiple of another. (Addition and multiplication on $\{0, 1, 2\}$ are carried out modulo 3.) Each of the 13 vectors above can be paired off with a vector that is 2 times itself. These 13 pairs together with the vector of all 0s complete the set of all vectors from the set $\{0, 1, 2\}$. Each column in the matrix is identified with a coin in the collection. We shall refer to them as the 1^{st}, 2^{nd},…, i^{th},…, 13^{th} coins. Each row of the matrix specifies a way to use the balance:

(1) Put all coins corresponding to 1's in the row on the left pan.
(2) Put all coins corresponding to 2's in the row on the right pan.
(3) Use additional good coins if necessary so that the total number of coins on both pans are equal.

If the two pans are balanced, write down a 0. If the left pan is lighter than the right pan, write down a 1. If the left pan is heavier than the right pan, write down a 2.

After using the balance three times, we obtain a 3-digit column vector. If the column vector contains three 0s, all coins are good. If the column vector is identical to the i^{th} column in the matrix, then the i^{th} coin is a bad coin that is lighter than a good coin. If the column vector is identical to 2 times the i^{th} column in the matrix, then the i^{th} coin is a bad coin that is heavier than a good coin.

Using the above matrix as an illustration, for row one of the matrix, we put coins 4, 5, 7, 10 on the left pan and coins 1, 9, 11, 13 on the right pan. For row two of the matrix, we put coins 4, 6, 8, 11 on the left pan and coins 2, 7, 12, 13 on the right pan. For row three of the matrix, we put coins 5, 6, 9, 12 on the left pan and coins 3, 8, 10, 13 on the right pan.

A matrix for $n = 4$, is

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

A similar construction will give a procedure for general k.

7.16 Write $p(x)$ as

$$((\ldots(a_n x + a_{n-1})\, x + a_{n-2})\, x + \ldots + a_1)\, x + a_0$$

It takes n multiplications and n addition operations.

7.17 (a) k

(b) Simply replace the operation $x_i + x_j$ in the addition algorithm by max (x_i, x_j). Thus, the time complexity of the algorithm is also k.

(c) Do in parallel, $i = 1, 2, \ldots, 2^k$

$$x_i = \begin{cases} 1 & \text{if } x_i \text{ is positive} \\ 0 & \text{otherwise} \end{cases}$$

Add all the x_i using the addition algorithm above.
Time complexity is $1 + k$.

7.18 (a) Do $i = \lfloor n/2 \rfloor, \lfloor n/2 \rfloor - 1, \ldots, 2, 1$

1. $k = i$,
2. If x_k is a leaf then done
3. Otherwise let x_j be the maximum of the two sons of x_k.
4. If $x_k < x_j$ then exchange x_k and x_j and set $k = j$.
5. Repeat steps 2, 3, 4 until done

Each time through the loop, we scan a subtree rooted at i. The height of this tree is $\lfloor \log(n/i) \rfloor$. Thus the complexity of this step in terms of the number of exchanges (the number of comparisons is roughly 2 times the number of exchanges) is

$$\sum_{i=1}^{\lfloor n/2 \rfloor} \lfloor \log(n/i) \rfloor \leq \sum_{i=1}^{\lfloor n/2 \rfloor} \log(n/i) = \lfloor n/2 \rfloor \log n - \log \lfloor n/2 \rfloor ! = O(n)$$

(b) Use the algorithm in given part (a) with $i = 1$ and apply the algorithm on the tree with the last element removed. Complexity of this part is $\log ((n - 1)/1)$.

(c) (i) First form a heap on n elements using the algorithm in (a).

(ii) Do $f = n, n - 1, \ldots, 3, 2.$
Exchange x_1 and x_f
Restore the tree using algorithm in (b)

Complexity of part (ii) is

$$\sum_{i=2}^{n} \lfloor \log (i-1) \rfloor = n \log n + O(n)$$

Thus the complexity of the entire algorithm is

$$n \log n + O(n).$$

7.19 (a) $x \cdot x = x^2$, $x^2 \cdot x^2 = x^4$, $x^4 \cdot x^4 = x^8$, $x^8 \cdot x^8 = x^{16}$.

(b) 6. $x \cdot x = x^2$, $x^2 \cdot x = x^3$, $x^3 \cdot x^3 = x^6$, $x^6 \cdot x = x^7$, $x^7 \cdot x^7 = x^{14}$, $x^{14} \cdot x = x^{15}$.

7. $x \cdot x = x^2$, $x^2 \cdot x^2 = x^4$, $x^4 \cdot x = x^5$, $x^5 \cdot x^5 = x^{10}$, $x^{10} \cdot x = x^{11}$, $x^{11} \cdot x^{11} = x^{22}$, $x^{22} \cdot x = x^{23}$

(c) *square* $=> x^2$
multiply $=> x^3$
square $=> x^6$
multiply $=> x^7$
square $=> x^{14}$
square $=> x^{28}$
multiply $=> x^{29}$

(d) Binary representation of 59 is 111011.
This will give the sequence *SXSXSSXSX*.

(e) The algorithm corresponds to the evaluation of the binary representation of n, using Horner's method.

(f) An addition chain gives a procedure for computing x^n. Corresponding to $a_i = a_j + a_k$, we perform the multiplication $x^{a_i} = x^{a_j} \cdot x^{a_k}$.

(g)

For	19, 1 2 4 8 9 18 19.	7 multiplications.
For	33, 1 2 4 8 16 32 33.	7 multiplications.
For	46, 1 2 4 5 10 11 22 23 46.	9 multiplications.
For	79, 1 2 4 8 9 18 19 38 39 78 79.	11 multiplications.
For	87, 1 2 4 5 10 20 21 42 43 86 87.	11 multiplications.