Algorithm Analysis and Design

Exercises, Ch 2

(Mathematical Analysis)

Solutions are based on book's solution manual

1. For each of the following algorithms, indicate (i) a natural size metric for its inputs, (ii) its basic operation, and (iii) whether the basic operation count can be different for inputs of the same size:
   a. computing the sum of n numbers
      Answer:
      (i) n ; (ii) addition of two numbers; (iii) no
   b. computing $n!$
      Answer:
   (i) the magnitude of n, *i.e.*, the number of bits in its binary representation [=floor($\log_2 n$)+1]; (ii) multiplication of two numbers; (iii) no
   c. finding the largest element in a list of n numbers
      Answer:
      (i) n ; (ii) comparison of two numbers;(iii) no

   d. Euclid's algorithm
      Answer:
      (i) either the magnitude of the larger of two input numbers, or the magnitude of the smaller of two input numbers, or the sum of the magnitudes of two input numbers;; (ii) modulo division; (iii) yes

   e. pen-and-pencil algorithm for multiplying two n-digit decimal integers
      Answer:
      (i) n; (ii) multiplication of two digits; (iii) no

2.
   a. Consider the definition-based algorithm for adding two $n \times n$ matrices. What is its basic operation? How many times is it performed as a function of the matrix order n? As a function of the total number of elements in the input matrices?
      Answer:
      Addition of two numbers.

It's performed n^2 times (once for each of n^2 elements in the matrix being computed) .

Since the total number of elements in two given matrices is $N=2n^2$ the total number of additions can also be expressed as $n^2 = N/2$

b. Answer the same questions for the definition-based algorithm for matrix multiplication.

Answer:

The total number of multiplications is $n \cdot n^2 = n^3 = (N/2)^{3/2}$

3. For each of the following functions, indicate how much the function's value will change if its argument is increased fourfold.

a. $\log_2 n$ b. $\sqrt{n}$ c. n d. n^2 e. n^3 f. 2^n

a. $\log_2 4n - \log_2 n = (\log_2 4 + \log_2 n) - \log_2 n = 2.$
b. $\sqrt{4n}/\sqrt{n}=2\sqrt{n}/\sqrt{n}=2$
c. $4n/n=4$
d. $(4n)^2/ n^2=16$
e. $(4n)^3/ n^3=64$
f. $2^{4n}/ 2^n= (2^n)^3$

4. For each of the following pairs of functions, indicate whether the first function of each of the following pairs has a lower, same, or higher order of growth (to within a constant multiple) than the second function.
   a. $n(n + 1)$ and $2000n^2$
   b. $100n^2$ and $0.01n^3$
   c. $log_2^2 n$ and $\log_2 n^2$
   d. 2^{n-1} and 2^n
   e. $(n-1)!$ and $n!$
   a. $n(n + 1) \approx n^2$ has the same order of growth (quadratic) as $2000 n^2$ to within a constant multiple
   b. $100n^2$ (quadrtatic) has a lower order of growth than $0.01n^3$ (cubic)
   c. $log_2^2 n =$ $\log_2 n \log_2 n$, $\log_2 n^2 = 2 \log_2 n$, so $log_2^2 n$ has a higher order of growth than $\log_2 n^2$
   d. $2^{n-1} = ½ . 2^n$ so 2^{n-1} has the same order of growth as 2^{n-1}
   e. $n!=n(n-1)!$. So $(n-1)!$ has a lower order of growth than $(n)!$

5. Use the informal definitions of O, Θ and Ω to determine whether the following assertions are true or false.
   a. $n(n + 1)/2 \in O(n^3)$ true
   b. $n(n + 1)/2 \in O(n^2)$ true
   c. $n(n + 1)/2 \in \Theta(n^3)$ false

d. $n(n + 1)/2 \in \Omega(n)$ true

6. For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. (Use the simplest $g(n)$ possible in your answers.) Prove your assertions.
   a. $(n^2 + 1)^{10}$
   b. $\sqrt{(10 n^2+7n +3)}$
   c. $2n \log(n + 2)^2 + (n + 2)^2 \log n^2$
   d. $2^{n+1} + 3^{n-1}$
   e. floor $(\log_2 n)$

   Answer
   a. $\in \Theta(n^{20})$
   b. $\in \Theta(n)$
   c. $\in \Theta(n^2 \log n)$
   d. $\in \Theta(3^n)$
   e. $\in \Theta(\log_2 n)$

7. Prove that the following are listed in increasing order of their order of growth.

   $\log n$, n, $n \log n$, n^2 , n^3 , 2^n

   Answer

   Hint see *lim* of 1st on the 2nd (*i.e.*, is $lim(\log n/n)) = 0$,) the prove that $\lim(n/n \log n) = 0)$, ….

8. Prove that every polynomial of degree k, $p(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_0$ with $a_k > 0$, belongs to $\Theta(n^k)$.

   Answer

   Hint: compute $\lim(p(n)/ n^k)$

9. Compute the following sums.

   a. $1 + 3 + 5 + 7 + \cdots + 999$

   b. $2 + 4 + 8 + 16 + \cdots + 1024$

   c. $\sum_{i=3}^{n+1} 1$ **d.** $\sum_{i=3}^{n+1} i$ **e.** $\sum_{i=0}^{n-1} i(i+1)$

   Answer:

   a. $1+3+5+7+...+999 = \sum_{i=1}^{500}(2i\text{-}1) = \sum_{i=1}^{500} 2i\text{-}\sum_{i=1}^{500} 1 = 2\frac{500*501}{2}\text{-}500 = 250,000.$

   b. $2+4+8+16+...+1,024 = \sum_{i=1}^{10} 2^i = \sum_{i=0}^{10} 2^i - 1 = (2^{11} - 1) - 1 = 2,046.$

c. $\sum_{i=3}^{n+1} 1 = (n+1) - 3 + 1 = n - 1.$

d. $\sum_{i=3}^{n+1} i = \sum_{i=0}^{n+1} i - \sum_{i=0}^{2} i = \frac{(n+1)(n+2)}{2} - 3 = \frac{n^2+3n-4}{2}.$

e. $\sum_{i=0}^{n-1} i(i+1) = \sum_{i=0}^{n-1} (i^2+i) = \sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} i = \frac{(n-1)n(2n-1)}{6} + \frac{(n-1)n}{2}$
$= \frac{(n^2-1)n}{3}.$

10. Find the order of growth of the following sums. Use the Θ(g(n)) notation with the simplest function g(n) possible.

a. $\sum_{i=0}^{n-1}(i^2+1)^2$ **b.** $\sum_{i=2}^{n-1} \lg i^2$

Answers:

a. $\sum_{i=0}^{n-1} (i^2+1)^2 = \sum_{i=0}^{n-1} (i^4+2i^2+1) = \sum_{i=0}^{n-1} i^4 + 2\sum_{i=0}^{n-1} i^2 + \sum_{i=0}^{n-1} 1$
$\in \Theta(n^5) + \Theta(n^3) + \Theta(n) = \Theta(n^5)$ (or just $\sum_{i=0}^{n-1} (i^2+1)^2 \approx \sum_{i=0}^{n-1} i^4 \in \Theta(n^5)$).

b. $\sum_{i=2}^{n-1} \log_2 i^2 = \sum_{i=2}^{n-1} 2\log_2 i = 2\sum_{i=2}^{n-1} \log_2 i = 2\sum_{i=1}^{n} \log_2 i - 2\log_2 n$
$\in 2\Theta(n\log n) - \Theta(\log n) = \Theta(n \log n).$

11. Consider the following algorithm.

```
ALGORITHM Mystery(n)
    //Input: A nonnegative integer n
    S ← 0
    for i ← 1 to n do
        S ← S + i * i
    return S
```

a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the basic operation executed?

d. What is the efficiency class of this algorithm?
e. Suggest an improvement, or a better algorithm altogether, and indicate its efficiency class.

Answers:

a. Computes $S(n) = \sum_{i=1}^{n} i^2$.

b. Multiplication (or, if multiplication and addition are assumed to take the same amount of time, either of the two).

c. $C(n) = \sum_{i=1}^{n} 1 = n$.

d. $C(n) = n \in \Theta(n)$. Since the number of bits $b = \lfloor \log_2 n \rfloor + 1 \approx \log_2 n$ and hence $n \approx 2^b$, $C(n) \approx 2^b \in \Theta(2^b)$.

e. Use the formula $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$ to compute the sum in $\Theta(1)$ time (which assumes that the time of arithmetic operations stay constant irrespective of the size of the operations' operands).

Note that for e. when carry out the left-hand side, the number of basic operation is as above *i.e.*, $\Theta(n)$, however, we carry out operation in the right-hand side so it is $\Theta(1)$ which means the number of basic operation is constant regardless of *n*: it is always 3 multiplication.

12. Solve the following recurrence relations.
   a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
   b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
   c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
   d. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)
   Answers

a. $x(n) = x(n-1) + 5 \quad \text{for } n > 1, \quad x(1) = 0$

$$\begin{aligned}
x(n) &= x(n-1) + 5 \\
&= [x(n-2) + 5] + 5 = x(n-2) + 5 \cdot 2 \\
&= [x(n-3) + 5] + 5 \cdot 2 = x(n-3) + 5 \cdot 3 \\
&= \ldots \\
&= x(n-i) + 5 \cdot i \\
&= \ldots \\
&= x(1) + 5 \cdot (n-1) = 5(n-1).
\end{aligned}$$

b. $x(n) = 3x(n-1) \quad \text{for } n > 1, \quad x(1) = 4$

$$\begin{aligned}
x(n) &= 3x(n-1) \\
&= 3[3x(n-2)] = 3^2 x(n-2) \\
&= 3^2[3x(n-3)] = 3^3 x(n-3) \\
&= \ldots \\
&= 3^i x(n-i) \\
&= \ldots \\
&= 3^{n-1} x(1) = 4 \cdot 3^{n-1}.
\end{aligned}$$

c. $x(n) = x(n-1) + n \quad \text{for } n > 0, \quad x(0) = 0$

$$\begin{aligned}
x(n) &= x(n-1) + n \\
&= [x(n-2) + (n-1)] + n = x(n-2) + (n-1) + n \\
&= [x(n-3) + (n-2)] + (n-1) + n = x(n-3) + (n-2) + (n-1) + n \\
&= \ldots \\
&= x(n-i) + (n-i+1) + (n-i+2) + \ldots + n \\
&= \ldots \\
&= x(0) + 1 + 2 + \ldots + n = \frac{n(n+1)}{2}.
\end{aligned}$$

d. $x(n) = x(n/2) + n \quad \text{for } n > 1, \quad x(1) = 1 \quad (\text{solve for } n = 2^k)$

$$\begin{aligned}
x(2^k) &= x(2^{k-1}) + 2^k \\
&= [x(2^{k-2}) + 2^{k-1}] + 2^k = x(2^{k-2}) + 2^{k-1} + 2^k \\
&= [x(2^{k-3}) + 2^{k-2}] + 2^{k-1} + 2^k = x(2^{k-3}) + 2^{k-2} + 2^{k-1} + 2^k \\
&= \ldots \\
&= x(2^{k-i}) + 2^{k-i+1} + 2^{k-i+2} + \ldots + 2^k \\
&= \ldots \\
&= x(2^{k-k}) + 2^1 + 2^2 + \ldots + 2^k = 1 + 2^1 + 2^2 + \ldots + 2^k \\
&= 2^{k+1} - 1 = 2 \cdot 2^k - 1 = 2n - 1.
\end{aligned}$$

13. Consider the following recursive algorithm for computing the sum of the first *n* cubes: $S(n) = 1^3 + 2^3 + \ldots + n^3$.

```
ALGORITHM S(n)
    //Input: A positive integer n
    //Output: The sum of the first n cubes
        if n = 1 return 1
        else return S(n − 1) + n ∗ n ∗ n
```

a. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.

b. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this sum?

Answer:

a. Let $M(n)$ be the number of **multiplications** made by the algorithm. We have the following recurrence relation for it:
$M(n) = M(n-1) + 2$, $M(1) = 0$.

$$\begin{aligned}
M(n) &= M(n-1) + 2 \\
&= [M(n-2) + 2] + 2 = M(n-2) + 2 + 2 \\
&= [M(n-3) + 2]+2+2 = M(n-3)+2+2 + 2 \\
&= \ldots \\
&= M(n-i) + 2i \\
&= \ldots \\
&= M(1) + 2(n-1) = 2(n-1).
\end{aligned}$$

b. Algorithm NonrecS(n)

```
//Computes the sum of the first n cubes nonrecursively
//Input: A positive integer n
//Output: The sum of the first n cubes.
S ← 1
for i ← 2 to n do
    S ← S + i ∗ i ∗ i
return S
```

The number of multiplications made by this algorithm will be

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

This is exactly the same number as in the recursive version, but the nonrecursive version doesn't carry the time and space overhead associated with the recursion's stack.