CLRS Solutions

Osbert Ngok

2013-03-20

Contents

Chapter 1

The Role of Algorithms in Computing

1.1 Algorithms

Exercise 1.1.1.

Answer 1. Calculating result of competition.

Exercise 1.1.2.

Answer 2. Productivity.

Exercise 1.1.3.

Answer 3. A set of 3NF tables of html, links and pdfs. It is stored in an Excel file and does great in terms of minimizing data redundancy, with the expense of difficulty to read the results without joining.

Exercise 1.1.4.

Answer 4. Both are looking for shorting path in a graph, but the known solutions are different in terms of order of growth.

Exercise 1.1.5.

Answer 5. An algorithm to determine how much change should be returned from buying a ticket with bank notes. Compose a piece of music using generic algorithms.

1.2 Algorithms as a technology

Exercise 1.2.1.

Answer 6. Keygen. To calculate a valid series code.

Exercise 1.2.2.

Answer 7.

$$\begin{aligned} 8n^2 &\leq 64n \lg n \\ n &\leq 8 \lg n \\ \frac{n}{\lg n} &\leq 8 \end{aligned}$$

Turning point is around 43.5. So when $n \leq 43$, insertion sort rules.

Exercise 1.2.3.

Answer 8.

$$\begin{aligned} 100n^2 &\leq 2^n \\ n &\geq 8 \lg n \\ \frac{n}{\lg n} &\geq 8 \end{aligned}$$

Turning point is around 14.325. So when $n \geq 15$, polynomial wins.

Problem 1.1.

Answer 9. This problem is hilarious. For the size of n that can handle $\lg n$ in 1 second $i.e. 1,000,000 microseconds$, the answer is obviously $2^1000000$, but people might just want to count how many digits that number could have. I typed the formula in python and python failed to respond because it tried to caculate the exact value.

	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\lg n$	$10^{3\times10^6}$	$10^{1.8\times10^7}$	$10^{1.08\times10^9}$	$10^{2.6\times10^{10}}$	$10^{7.8\times10^{11}}$	$10^{9.49\times10^{12}}$	$10^{9.5\times10^{15}}$
$\sqrt{n}$	10^{12}	$10^{15.6}$	$10^{19.1}$	$10^{21.9}$	$10^{24.8}$	10^{27}	10^{33}
n	10^6	$10^{7.8}$	$10^{9.5}$	$10^{10.9}$	$10^{12.4}$	$10^{13.5}$	$10^{16.5}$
$n \lg n$	$10^{4.8}$	$10^{6.45}$	$10^{8.1}$	$10^{9.44}$	$10^{10.86}$	$10^{11.9}$	$10^{14.8}$
n^2	10^3	$10^{3.9}$	$10^{4.7}$	$10^{5.9}$	$10^{6.2}$	$10^{6.7}$	$10^{8.2}$
n^3	10^2	$10^{2.6}$	$10^{3.2}$	$10^{3.6}$	$10^{4.1}$	$10^{4.5}$	$10^{5.5}$
2^n	19	25	31	36	41	44	54
$n!$	8	10	11	13	14	15	17

Chapter 2

Getting Started

2.1 Insertion sort

Exercise 2.1.1.

Answer 10.

31	41	59	26	41	58
31	41	59	26	41	58
31	41	59	26	41	58
26	31	41	59	41	58
26	31	41	41	59	58
26	31	41	41	58	59

Exercise 2.1.2.

Answer 11.

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] < key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

Exercise 2.1.3.

Answer 12.

Linear-Search(A, v)

```
o = NIL
for j = 1 to A.length
    if A[j] = v
        o = j
return o
```

Exercise 2.1.4.

Answer 13.

Input: Two sequences of n booleans $\langle a_1, a_2, \ldots a_n\rangle$ and $\langle b_1, b_2, \ldots b_n\rangle$
Output: One sequence of $n+1$ booleans $\langle c_1, c_2, \ldots c_{n+1}\rangle$ which represents the sum of two binary sequences combined.

Binary-Addition(A, B, C)

```
for j = n to 1
    // Calculate next digit
    C[j] = (A[j] and B[j]) or (B[j] and C[j + 1]) or (C[j + 1] and A[j])
    // Calculate current digit
    C[j + 1] = A[j] xor B[j] xor C[j]
return C
```

2.2 Analyzing algorithms

Exercise 2.2.1.

Answer 14. $\Theta\left(n^3\right)$

Exercise 2.2.2.

Answer 15.

Selection-Sort(A)

```
for j = 1 to A.length
    smallest_index = j
    for k = j + 1 to A.length
        if A[k] < A[smallest_index]
            smallest_index = k
    // Swapping
    temp = A[j]
    A[j] = A[smallest_index]
    A[smallest_index] = A[j]
```

Loop invariant is $A[1\,..\,j]$, in ascending sorted order. When $j = n$, there is only one variable in the remaining list and it is obviously the smallest, so there is no need to sort. The best and worst time are both $\Theta\left(n^2\right)$.

Exercise 2.2.3.

Answer 16. **worse case**: n
average case: $\dfrac{1+\cdots+n}{n}=\dfrac{1+n}{2}$
best case: 1
average case Θ: $\Theta(n)$
worst case Θ: $\Theta(n)$

Exercise 2.2.4.

Answer 17. Pre-calculate an output of a possible input; judge if the input is the desired input first; if it is, output the prepared output; otherwise use the normal algorithm.

2.3 Designing algorithms

Exercise 2.3.1.

Answer 18.

3	9	26	38	41	49	52	59
3	26	41	52	9	38	49	59
3	41	26	52	38	59	9	49
3	41	52	26	38	59	9	49

Exercise 2.3.2.

Answer 19.

```
MERGE(A, p, q, r)
n1 = q − p + 1
n2 = r − q
let L[1 .. n1 + 1] and R[1 .. n2 + 2] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
i = 1
j = 1
k = p
while k ≤ r and i ≤ n1 and j ≤ n2
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
    k = k + 1
if k ≤ r
    if i ≤ n1
        for l = 1 to n1 − i + 1
            A[k + l − 1] = L[i + l − 1]
    else
        for l = 1 to n2 − j + 1
            A[k + l − 1] = R[j + l − 1]
```

Exercise 2.3.3.

Answer 20.

1. When $k = 1$, $n = 2^k = 2$, $T(n) = T(2) = 2 = 2\lg 2$

2. Suppose $T\left(2^k\right) = 2^t \lg\left(2^k\right)$ is true, we have
$$\begin{aligned} T\left(2^{k+1}\right) &= 2T\left(2^k\right) + 2^{k+1} \\ &= 2 \cdot 2^t \lg\left(2^k\right) + 2^{k+1} \\ &= 2^{k+1} \cdot k + 2^{k+1} \\ &= (k+1)\, 2^{k+1} \\ &= 2^{k+1} \lg\left(2^{k+1}\right) \end{aligned}$$
so $k + 1$ is true.

3. Therefore $\forall k \in \mathbb{Z}_{\geq 1}$, let $n = 2^k$, $T(n) = n \lg n$.

Exercise 2.3.4.

Answer 21.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c, \\ T(n-1) + C(n) & \text{otherwise.} \end{cases}$$

Exercise 2.3.5.

Answer 22.

```
BINARY-SEARCH(A, v, first, last)
// Boundary cases
if first > last return NIL
if first == last
    if A[first] == v
        return first
    else
        return NIL
// Find element in the middle of the array
mid = ⌊(first + last) / 2⌋
if A[mid] == v return mid
if A[mid] < v return BINARY-SEARCH(A, v, mid + 1, last)
if A[mid] > v return BINARY-SEARCH(A, v, first, mid − 1)
print "What?!"
```

$T(n) = T(n/2) + C$
$T(n) = \Theta(\lg n)$

Exercise 2.3.6.

Answer 23. line 6 that moves the data may still take $\Theta(n)$ in worst case even BINARY-SEARCH helps to find insertion point in $\Theta(\lg n)$, leading to the worst-case running time still $\Theta\left(n^2\right)$.

Exercise 2.3.7.

Answer 24. $\Theta(n \lg n)$ gives a sorted array for free. We divide the array into two halves, and there are 3 possibilities: either the pair exist within one of the halves, or the pair reside into both halves each. The first two situations are no more than a recursive call, so let's take a look at the third (i.e. the **Combine** step).

Suppose we have array A with n_1 sorted elements, and B with n_2 sorted elements. To simplify the intimidating "whole sum is exactly x' statement, let's construct a new sorted array $A\prime$ from A, where element $a\prime \in A\prime$ if and only if $(x - a) \in A$. This will take $\Theta(n)$ which is OK for the **Combine** step of a $\Theta(n \lg n)$ divide-and-conquer algorithm. If there exists $a_p \in A$ and $b_q \in B$ such that $a_p + b_q = x$, obviously we have $b_q = x - a_p \in A\prime$. Now the question becomes:

"Given two sorted arrays $A\prime$ with n_1 elements and B with n_2 elements, describe a $\Theta(n_1 + n_2)$ algorithm that, return if there is any common element in these two arrays."

This proves to be easy, for we can have two pointers pointing to the smallest elements of both array and start the comparison. If they match, we got the job done; if they don't, we advance the smaller element pointer in its corresponding array and repeat the process until a match or an out-of-boundary exception, when we know there is no such a pair.

Problem 2.1.

Answer 25.

a. With worst-case time of INSERTION-SORT for a length k array is $\Theta\left(k^2\right)$, there is no doubt that n/k sublists will make it $\Theta\left(n/k \cdot k^2\right) = \Theta(nk)$.

b. It is easy to find a $\Theta\left(n^2/k\right)$ solution: for each iteration of putting the smallest element from n/k sorted lists, and there are n interations in total. But if we maintein a heap to store the smallest values (which initially takes $\Theta(n/k)$ time), its insertion and deletion will only take $\Theta(\lg(n/k))$ in worst-case time, making each iteration $\Theta(\lg(n/k))$. Thus to merge the sublists, it will take $\Theta(n \lg(n/k))$ worst-case time.

c. A good guess will be $k = \lg n$, where

$$\begin{aligned}\Theta(nk + n\lg(n/k)) &= \Theta(n\lg n + n\lg n - n\lg\lg n)\\ &= \Theta(n\lg n)\end{aligned}$$

d. I don't know, random pick from 2 to $\lg n$?

Problem 2.2.

Answer 26. a. We need to prove that for each $A\prime[i]$, there is a corresponding $A[j]$ in the original array, where i and j is one-to-one relationship.

b. **loop invariant:** for $A[1\,..\,i]$, we have $A[1] \leq A[2] \leq \cdots \leq A[i]$. It holds for **Initialization** when $i = 1$. line 3 ensures any value smaller than the newer value will reside on the left of the newer value, while any value bigger than the newer value will reside on the right of the new value. At **Termination**, all values are sorted.

c. Similar to b.

d. Both of them are $\Theta\left(n^2\right)$.

Problem 2.3.

Answer 27.

a. $\Theta(n)$

b. Code as follows:

EVALUATE-POLYNOMIAL$(a_0, a_1, \cdots, a_n, x)$

$sum = 0$
for $i = 0$ **to** n
    $y = a_i$
    for $j = 1$ **to** i
        $y = y \cdot x$
    $sum = sum + y$
return sum

Running time is $\Theta\left(n^2\right)$, significantly higher than **Horner's Rule**.

c. Can be proved by Mathematical Induction.

d. Omitted.

Problem 2.4.

Answer 28.

a. $\langle 2, 1\rangle, \langle 3, 1\rangle, \langle 8, 6\rangle, \langle 8, 1\rangle, \langle 6, 1\rangle$

b. When all elements are in reverse order, the number is $\frac{n(n-1)}{2}$.

c. In INSERTION-SORT line 6 - 7, every time these instructions are executed, it corresponds to the inversion of A[i] and A[j], so the running time is proportional.

d.

INVERSION-NUMBER(A, p, r)

if $p \geq r$
    return 0
else
    $q = \lfloor (p + r) / 2 \rfloor$
    return INVERSION-NUMBER(A, p, q)
+INVERSION-NUMBER$(A, q + 1, r)$
+INVERSION-NUMBER-MERGE(A, p, q, r)

```
Inversion-Number-Merge(A, p, q, r)
count = 0
n1 = q − p + 1
n2 = r − q
let L[1 . . n1 + 1] and R[1 . . n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i − 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else
        // Inverse exists
        count = count + n1 − i + 1
        A[k] = R[j]
        j = j + 1
return count
```

Chapter 3

Growth of Functions

3.1 Asymptotic notation

Exercise 3.1.1.

Answer 29.
Because both $f(n)$ and $g(n)$ are **asymptotically nonnegative**, there exist n_1 and n_2 such that for $n \geq n_1$, $f(n) \geq 0$ and $n \geq n_2$, $g(n) \geq 0$. Take $n_0 = \max(n_1, n_2)$, and when $n \geq n_0$, $f(n) \geq 0$ and $g(n) \geq 0$.

Under the same condition we will have: $\max(f(n), g(n)) \geq \frac{1}{2}(f(n) + g(n))$, so $\max(f(n), g(n)) = \Omega(f(n) + g(n))$.

On the other hand, $\max(f(n), g(n)) \leq f(n) + g(n)$, so $\max(f(n), g(n)) = O(f(n) + g(n))$

Therefore, $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

Exercise 3.1.2.

Answer 30.
$(n+a)^b = \sum\limits_{k=0}^{\infty} \binom{b}{k} n^{b-k} a^k = n^b + o\left(n^b\right) = \Theta\left(n^b\right)$

Exercise 3.1.3.

Answer 31. $O\left(n^2\right)$ provides an upper bound; the "at least" waives the upper bound, so it is meaningless.

Exercise 3.1.4.

Answer 32.
$2^{n+1} = 2 \cdot 2^n$, so $2^{n+1} = \Theta(2^n) = O(2^n)$. Suppose there exist c_0, n_0 when $n \geq n_0$, $2^{2n} \leq c_0 2^n$, which implies $2^n \leq c_0$ and won't hold true when n is sufficiently large.

Exercise 3.1.5.

Answer 33. Yes it is easy to prove.

Exercise 3.1.6.

Answer 34. Omitted.

Exercise 3.1.7.

Answer 35. Suppose there exists function $f(n)$ such that $f(n) = o(g(n))$ and $f(n) = \omega(g(n))$, let $c = 2$, we obtain n_1 and n_2. Let $n_0 = \max(n_1, n_2)$, we have:

For $n \geq n_0$, $f(n) < 2g(n)$ and $f(n) > 2g(n)$, contradiction.

Exercise 3.1.8.

Answer 36.

$$
\begin{aligned}
\Omega(g(n,m)) = \{f(n,m) &: \text{there exist positive constants } c, n_0, \text{ and } m_0 \\
&\text{such that } 0 \leq cg(n,m) \leq f(n,m) \text{ for all } n \geq n_0 \text{ or } m \geq m_0\} \\
\Theta(g(n,m)) = \{f(n,m) &: \text{there exist positive constants } c_1, c_2, n_0, \text{ and } m_0 \\
&\text{such that } 0 \leq c_1 g(n,m) \leq f(n,m) \leq c_2 g(n,m) \text{ for all } n \geq n_0 \text{ or } m \geq m_0\}
\end{aligned}
$$

3.2 Standard notations and common functions

Exercise 3.2.1.

Answer 37. Proved by contradiction.

Exercise 3.2.2.

Answer 38.

$$a^{\log_b c} = a^{\frac{\log_a c}{\log_a b}} = a^{log_a c \cdot \log_b a} = c^{\log_b a}$$

Exercise 3.2.3.

Answer 39.

$$
\begin{aligned}
\lg(n!) &= \lg\left(\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \Theta\left(\frac{1}{n}\right)\right)\right) \\
&= \lg\left(\sqrt{2\pi n}\right) + \lg\left(\left(\frac{n}{e}\right)^n\right) + \lg\left(1 + \Theta\left(\frac{1}{n}\right)\right) \\
&= \Theta(n) + \Theta(n \lg n) + \Theta(1) \\
&= \Theta(n \lg n)
\end{aligned}
$$

$\forall c > 0$, $\exists n_0 = \max(4, \lceil 4c \rceil)$ such that $n_0! \geq 1 \cdot 2^{n_0 - 2} \cdot (4 \cdot c) \geq c2^{n_0}$.

$\forall c > 0$, $\exists n_0 = \lceil c \rceil$ such that $n^n = n \cdot n^{n-1} \geq cn!$.

Exercise 3.2.4.

Answer 40. $\lceil \lg n \rceil ! = O\left(n^a\right)$ is equal to $n! = O\left(2^{an}\right)$, and can be proved similarly by 3.2.3 that it is not true.

$\lceil \lg \lg n \rceil ! = O\left(n^a\right)$ is equal to $\lceil \lg n \rceil ! = O\left(2^{an}\right)$

$(\lg n)! = O\left(\lg n^{\lg n}\right) = O\left(\lg n^{an}\right) = O\left(\lg n^{\lg \lg n \cdot an}\right) = O\left(2^{an}\right)$

Exercise 3.2.5.

Answer 41. $\lg^*\left(\lg n\right) = \lg^*\left(n\right) - 1$, asymtotically larger than $\lg\left(\lg^* n\right)$.

Exercise 3.2.6.

Answer 42. Through Vieta's Formula, $\phi + \hat{\phi} = 1$, $\phi \cdot \hat{\phi} = -1$, satisfy the equation.

Exercise 3.2.7.

Answer 43. F_1, F_2 are trivial. When $n \le n_0 (n_0 \ge 2)$ the equality holds,

$$\begin{aligned}
F_{n_0+1} &= F_{n_0-1} + F_{n_0} \\
&= \frac{\phi^{n_0-1} - \hat{\phi}^{n_0-1}}{\sqrt{5}} + \frac{\phi^{n_0} - \hat{\phi}^{n_0}}{\sqrt{5}} \\
&= \frac{\phi^{n_0+1}\hat{\phi}^2 - \hat{\phi}^{n_0+1}\phi^2 - \phi^{n_0+1}\hat{\phi} + \hat{\phi}^{n_0+1}\phi}{\sqrt{5}} \\
&= \frac{\phi^{n_0+1}\left(\hat{\phi}^2 - \hat{\phi}\right) - \hat{\phi}^{n_0+1}\left(\phi^2 - \phi\right)}{\sqrt{5}} \\
&= \frac{\phi^{n_0+1} - \hat{\phi}^{n_0+1}}{\sqrt{5}}
\end{aligned}$$

Exercise 3.2.8.

Answer 44.

By definition, there exists c_1, c_2, n_0 such that

$$0 \le c_1 g\left(n\right) \le f\left(n\right) \le c_2 g\left(n\right)$$

for $n \ge n_0$

WLOG let's assume $c_1 > \ln c_2$.

$$\begin{aligned}
0 \le c_1 n \le k \ln k \le c_2 n \\
\frac{k \ln k}{c_2} \le n \le \frac{k \ln k}{c_1}
\end{aligned}$$

Because $\left(\frac{n}{\ln n}\right)' = \left(\frac{1}{\ln n}\right)\left(1 - \frac{1}{\ln n}\right) > 0$ when n is sufficiently large,

$$\begin{aligned}\frac{k\ln k}{c_2\left(\ln\left(k\ln k\right)-\ln c_2\right)} \le \frac{n}{\ln n} &\le \frac{k\ln k}{c_1\left(\ln\left(k\ln k\right)-\ln c_1\right)}\\ \frac{1}{c_2\left(1+\frac{\ln\ln k-\ln c_2}{\ln k}\right)}k \le \frac{n}{\ln n} &\le \frac{1}{c_1\left(1+\frac{\ln\ln k-\ln c_1}{\ln k}\right)}k\end{aligned}$$

Select n_1 such that $\ln\left(k\left(n\right)\right) > c_2 > c_1$ when $n \ge n_1$,

$$\begin{aligned}\frac{1}{c_2\left(1+\frac{\ln\ln k}{\ln k}\right)}k \le \frac{n}{\ln n} &\le \frac{1}{c_1\left(1+\frac{\ln c_1-\ln c_1}{\ln k}\right)}k\\ \frac{1}{c_2\left(1+\frac{\ln k}{\ln k}\right)}k \le \frac{n}{\ln n} &\le \frac{1}{c_1}k\\ \frac{1}{2c_2}k \le \frac{n}{\ln n} &\le \frac{1}{c_1}k\end{aligned}$$

Problem 3.1.

Answer 45. Only need to prove $p\left(n\right) = \Theta\left(n^d\right)$. Let $a' = \max\left(a_i\right)$, construct $p'\left(n\right) = \sum\limits_{i=0}^{d} a'\left(n^i\right)$ as an upperbound of p to compare.

$$p'\left(n\right) = a'\frac{n^{d+1}-1}{n-1} < a'\frac{n^{d+1}-1}{n-\frac{1}{10}n} = a'\frac{10}{9}\left(n^d - \frac{1}{n}\right) = \Theta\left(n^d\right)$$

So $p\left(n\right) = O\left(n^d\right)$.

On the other hand, let $a'' = \left|\min\left(0, \min\ a_i\right)\right|$ let $n_0 = \max\left(10, \frac{10}{8a_d}a''\right)$.

$$\begin{aligned}p''\left(n\right) &= a_d n^d - \sum_{i=0}^{d-1} a'' n^i\\ &= a_d n^d - a''\frac{n^d-1}{n-1}\\ &> a_d n^d - a''\frac{n^d-1}{n-\frac{1}{10}n}\\ &> \left(a_d - \frac{\frac{10}{9}a''}{n}\right)n^d\\ &> \left(a_d - \frac{\frac{10}{9}a''}{\frac{10}{8a_d}a''}\right)n^d\\ &= \frac{1}{9}a_d n^d = \Theta\left(n^d\right)\end{aligned}$$

So $p\left(n\right) = \Omega\left(n^d\right)$. So $p\left(n\right) = \Theta\left(n^d\right)$.

Problem 3.2.

Answer 46.

- $\lg(\lg^* n)$

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ϵ	yes	yes	no	no	no
n^k	c^n	yes	yes	no	no	no
$\sqrt{n}$	$n^{\sin n}$	no	no	no	no	no
2^n	$2^{n/2}$	no	no	yes	yes	no
$n^{\lg c}$	$c^{\lg n}$	yes	no	yes	no	yes
$\lg(n!)$	$\lg(n^n)$	yes	no	yes	no	yes

Problem 3.3.

Answer 47.

- $n^{1/\lg n} = \Theta(1)$
- $\lg(\lg^* n)$
- $\lg *(\lg n) = \lg^* n$
- $2^{\lg^* n}$
- $\ln \ln n$
- $\sqrt{\lg n}$
- $\ln n$
- $\lg^2 n$
- $2^{\sqrt{2 \lg n}}$
- $\left(\sqrt{2}\right)^{\lg n} = \Theta\left(\sqrt{n}\right)$
- $2^{\lg n} = \Theta(n)$
- $\lg(n!) = n \lg n$
- $4^{\lg n} = n^2$
- n^3
- $(\lg n)!$
- $n^{\lg \lg n} = (\lg n)^{\lg n}$
- $\left(\frac{3}{2}\right)^n$
- 2^n

- e^n
- $n \cdot 2^n$
- $n!$
- $(n+1)!$
- 2^{2^n}
- $2^{2^{n+1}}$

Let $g_i'(n) = g_i(n)^{2\sin n}$.

Problem 3.4.

Answer 48.

1. Wrong. Let $f(n) = n$ and $g(n) = n^2$.
2. Wrong. Let $f(n) = n$ and $g(n) = n^2$.
3. Correct.
4. Correct.
5. Correct.
6. Correct.
7. Wrong. Let $f(n) = 4^n$.
8. Correct.