Answers to Homework 1

p. 13 **1.2–3** *"What is the smallest value of n such that an algorithm whose running time is $100n^2$ runs faster than an algorithm whose running time is 2^n on the same machine?"*

Solution:

Find the smallest $n > 0$ such that $100n^2 < 2^n$.

Use a calculator and do a "binary search". I tried $n = 10, 20, 15, \ldots$.

n	$100n^2$	2^n
10	10^4	$\sim 10^3$
13	$1.69 \cdot 10^4$	$8.2 \cdot 10^3$
14	$1.96 \cdot 10^4$	$1.64 \cdot 10^4$
15	$2.25 \cdot 10^4$	$3.28 \cdot 10^4$
20	$4 \cdot 10^4$	$\sim 10^6$

At $n = 15$ you will find that 2^n exceeds $100n^2$.

p. 21 **2.1–3** *"Consider the **searching problem**:*

***Input:** A sequence of n numbers $A = \langle a_1, a_2, \ldots, a_n \rangle$ and a value v.*

***Output:** An index i such that $v = A[i]$ or the special value NIL if v does not appear in A.*

*Write pseudocode for **linear search**, which scans through the sequence, looking for v."*

Solution:

LINEAR-SEARCH(A, v)
1. $i \leftarrow 1$
2. **while** $i <$ *length*$[A]$ and $A[i] \neq v$
3. **do** $i \leftarrow i + 1$
4. **if** $A[i] \neq v$
5. **then** $i \leftarrow$NIL

We are not concerned with correctness in this course, but if you have thought of an invariant for LINEAR-SEARCH you may want to compare it against the following.

$$v \notin A[1 \ldots i-1] \text{ or } v = A[i]$$

p. 27 **2.2–1** *"Express the function $n^3/1000 - 100n^2 - 100n + 3$ in terms of Θ-notation."*

Solution:

$n^3/1000 - 100n^2 - 100n + 3 = \Theta(n^3)$

p. 27 **2.2–3** *"Consider linear search again (see Exercise 2.1–3). How many elements of the input sequence need to be checked on the average, assuming that the element being searched for is equally likely to be any element in the array? How about in the worst case? What are the*

average-case and worst-case running times of linear search in Θ-notation? Justify your answers."

Solution:

Let $n = \mathit{size}[A]$

Line	*Cost*	*Average # times*	*Worst-case # times*
1	c_1	1	1
2	c_2	$n/2$	n
3	c_3	$n/2$	n
4	c_4	1	1
5	c_5	1	1

Total "average time" $= c_1 + c_2 \cdot n/2 + c_3 \cdot n/2 + c_4 + c_5 = \Theta(n/2) = \Theta(n)$

Total "worst-case time" $= c_1 + c_2 \cdot n + c_3 \cdot n + c_4 + c_5 = \Theta(n)$

p. 27 **2.2–4** *"How can we modify almost any algorithm to have a good best-case running time?"*

Solution:

1. Guess an answer.
2. Verify that it is correct; in that case: stop.
3. Otherwise: run the original algorithm.

In the best case you will guess correctly in the first step thus enabling you to, for instance, sort in $\Theta(n)$ time.

p. 37 **2.3–3** *"Use mathematical induction to show that when n is an exact power of 2, the solution of the recurrence*

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

is $T(n) = n \lg n$."

Solution:

Assume that n is a power of two, $n = 2^m$ (for some m).

- **Base case:** $n = 2$
  $T(n) = n \lg n = 2 \lg 2 = 2$.
- **Inductive hypothesis:** Assume that $T(n) = n \lg n$ for all $n < 2^k$
- **Inductive step:** Now show that it also holds when $n = 2^k$

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &\quad \text{Use the inductive hypothesis on } T(n/2) \\ &= 2(n/2)\lg(n/2) + n \\ &= 2(n/2)(\lg n - \lg 2) + n \\ &= n \lg n - n + n = n \lg n \end{aligned}$$

p. 37 **2.3–7** *"Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x."*

Solution:

This is a typical example of how sorting can be used to simplify or improve an algorithm. The naive approach of looking for pairs $(a, b) \in S$ such that $x = a + b$ takes $\Theta(n^2)$ time in the worst case. There are obviously many ways to solve the problem in $\Theta(n \lg n)$; here is one.

```
1.  for i ← 1 to n                          ▷ Θ(n)
2.      do R[i] ← x − S[i]
3.  S ←MERGE-SORT(S)                        ▷ Θ(n lg n)
4.  for i ← 1 to n
5.      do                                  ▷ Try to find R[i] in S using binary search
6.          j ←BINARYSEARCH(S, 1, n, R[i])          ▷ Θ(lg n)
7.          if i ≠ j and j > 0
8.              then write R[i], S[j]
```

If you don't understand the algorithmm, try to construct a small example and see what happens. (If you still don't understand, one of us have made a mistake.)

The dominating terms are line 3 and the **for**-loop in line 4 which runs $n \cdot \Theta(\lg n)$ steps. Thus, the overall time for this algorithm is $\Theta(n \lg n)$.

p. 50 **3.1–1** *"Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of Θ-notation, prove that $\max(f(n), g(n)) = \Theta(f(n) + g(n))$."*

Solution:

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

This is true iff $\exists c_1, c_2, n_0 > 0$ such that

$$0 \leq c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n))$$

for all $n \geq n_0$.

- Choose for instance $c_1 = \frac{1}{2} \Rightarrow$ two situations:
  1. $f(n) > g(n) \Rightarrow \frac{1}{2}(f(n) + g(n)) < \frac{1}{2}(f(n) + f(n)) = f(n) = \max(f(n), g(n))$
  2. $f(n) < g(n) \Rightarrow \frac{1}{2}(f(n) + g(n)) < \frac{1}{2}(g(n) + g(n)) = g(n) = \max(f(n), g(n))$
- Choose for instance $c_2 = 1 \Rightarrow \max(f(n), g(n)) \leq f(n) + g(n)$.

So, choose $c_1 = \frac{1}{2}, c_2 = 1$ and the statement holds.

(Actually, any $c_1 \leq \frac{1}{2}$ and $c_2 \geq \frac{1}{2}$ works.)

p. 50 **3.1–3** *"Explain why the statement 'The running time of algorithm A is at least $O(n^2)$,' is meaningless."*

Solution:

"The running time of algorithm A is at least $O(n^2)$" is like saying that $T(n)$ is asymptotically greater than, or equal to, all those functions that are $O(n^2)$.

This is a bit like saying

x is at least n, where $n < 2$
$\Leftrightarrow$ $x \geq n$ och $n < 2$
$\Leftrightarrow$ x can be anything.

p. 50 **3.1–4** *"Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?"*

Solution:

Is $2^{n+1} = O(2^n)$, i.e., is $2^{n+1} \leq c \cdot 2^n$ for any c and all $n > n_0$?

Yes, choose $c \geq 2$ and the statement is true.

Is $2^{2n} = O(2^n)$, i.e., is $2^{2n} \leq c \cdot 2^n$ for any c and all $n > n_0$?

This is not the case since the ratio

$$\frac{2^{2n}}{c \cdot 2^n} = \frac{(2^n)^2}{c \cdot 2^n} = \frac{2^n}{c}$$

grows unbounded with n. Thus, $c \cdot 2^n$ can never be an upper bound to 2^{2n}, regardless how big we choose c.

p. 57 **3–1** *"Let $p(n) = \sum_{i=0}^{d} a_i n^i$, where $a_d > 0$, be a degree-d polynomial in n, and let k be a constant. Use the definitions of the asymptotic notations to prove the following properties.*

a. *If $k \geq d$, then $p(n) = O(n^k)$.*

c. *If $k = d$, then $p(n) = \Theta(n^k)$."*

Solution:

$p(n) = \sum_{i=0}^{d} a_i n^i$ where $a_d > 0$.

a) Show that if $k \geq d$, then $p(n) = O(n^k)$.

This is true iff $0 \leq \sum_{i=0}^{d} a_i n^i \leq c \cdot n^k$ for all $n > n_0$ and some c.

Divide with n^k and obtain $0 \leq \sum_{i=0}^{d} a_i n^{i-k} \leq c$.

Since $k \geq d$, all $i - k \leq 0$, i.e., all n^{i-k} will be less than 1.

So, choose $c = \sum_{i=0}^{d} a_i$ and the statement is true.

c) Show that if $k = d$, then $p(n) = \Theta(n^k)$.

This is true iff $0 \leq c_1 \cdot n^k \leq \sum_{i=0}^{d} a_i n_i \leq c_2 \cdot n^k$

Divide with n^k and obtain $0 \leq c_1 \leq \sum_{i=0}^{d} a_i n^{i-k} \leq c_2$

Since $k = d$, this is the same as

$$0 \leq c_1 \leq a_0 n^{-k} + a_1 n^{1-k} + \cdots + a_k n^0 \leq c_2$$

Here you can choose $c_1 = a_d$ and $c_2 = \sum_{i=0}^{d} a_i$ and the statement is true.

p. 58 **3–3** *"Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, \ldots, g_{30}$ of the functions satisfying $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.*

$\lg(\lg^* n)$	$2^{\lg^* n}$	$(\sqrt{2})^{\lg n}$	n^2	$n!$	$(\lg n)!$
$(\frac{3}{2}^n)$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg^* n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	1
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$4^{\lg n}$	$(n+1)!$	$\sqrt{\lg n}$
$\lg^*(\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$ „

Solution:

The ranking is based on the following rules of thumb:

- Exponential functions grows faster than polynomial functions, which in turn grows faster than polylogarithmic functions.
- The base of a logarithm is asymptotically irrelevant, but the base in an exponential function and the degree of a polynomial does matter.

In addition, some of the formulas in the course notes are useful, as well as Stirling's formula for the factorial function.

Here is the order then, where functions in the same class have been placed on the same line:

2^{2^n}

$(n+1)!$

$n!$

$n \cdot 2^n$

2^n

$(3/2)^n$

$(\lg n)^{\lg n} = n^{\lg \lg n}$

$(\lg n)!$

n^3

$n^2 = 4^{\lg n}$

$n \lg n$ och $\lg(n!)$

$n = 2^{\lg n}$

$(\sqrt{2})^{\lg n}$

$2^{\sqrt{2 \lg n}}$

$\lg^2 n$

$\ln n$

$\sqrt{\lg n}$

$\lg \lg n$

$\lg^* n$

$n^{1/\lg n}$ och 1

p. 59 **3–4** *"Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures."*

a. $f(n) = O(g(n))$ *implies* $g(n) = O(f(n))$.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

c. $f(n) = O(g(n))$ *implies* $\lg(f(n)) = O(\lg(g(n)))$, *where* $\lg(g(n)) \geq 1$ *and* $f(n) \geq 1$ *for all sufficiently large* n.

d. $f(n) = O(g(n))$ *implies* $2^{f(n)} = O(2^{g(n)})$.

e. $f(n) = O((f(n))^2)$.

f. $f(n) = O(g(n))$ *implies* $g(n) = \Omega(f(n))$.

g. $f(n) = \Theta(f(n/2))$.

h. $f(n) + o(f(n)) = \Theta(f(n))$."

Solution:

a) Counter proof: $n = O(n^2)$ but $n^2 \neq O(n)$.

b) Counter proof: $n^{17} + n \neq \Theta(n)$.

c)

$$\begin{aligned} f(n) = O(g(n)) &\Leftrightarrow 0 \leq f(n) \leq c \cdot g(n) \text{ for some } c > 0 \text{ and all } n > n_0 \\ &\Rightarrow \lg(f(n)) \leq \lg(c \cdot g(n)) \\ &\Rightarrow \lg(f(n)) \leq \lg c + \lg(g(n)) \\ &\quad \leq c_2 \cdot \lg(g(n)) \text{ for some } c_2 > 1 \\ &\Rightarrow \lg(f(n)) = O(\lg(g(n))) \end{aligned}$$

This only holds if $\lg(g(n))$ is not approaching 0 as n grows.

d) Counter proof: $2n = O(n)$ but $2^{2n} = 4^n \neq O(2^n)$

e) Counter proof: $f(n) = 1/n \neq O(1/n^2)$

f)

$$\begin{aligned} f(n) = O(g(n)) &\Leftrightarrow 0 \leq f(n) \leq c_1 \cdot g(n) \text{ for some } c_1 > 0 \text{ and all } n > n_0 \\ &\Leftrightarrow 0 \leq (1/c_1)f(n) \leq g(n) \\ &\Leftrightarrow g(n) = \Omega(f(n)) \end{aligned}$$

g) Counter proof: $4^n \neq \Theta(4^{n/2} = 2^n)$

h) Show that $f(n) + o(f(n)) = \Theta(f(n))$.

$$o(f(n)) = \{g(n) : 0 \leq g(n) < c \cdot f(n), \forall c > 0, \forall n > n_0\}$$

At most, $f(n) + o(f(n))$ can be $f(n) + c \cdot f(n) = (1 + c)f(n)$.

At least, $f(n) + o(f(n))$ can be $f(n) + 0 = f(n)$

Thus, $0 < f(n) < f(n) + o(f(n)) < (1 + c)f(n)$, which is the same as saying $f(n) + o(f(n)) = \Theta(f(n))$

p. 1062 **A.1–1** *"Find a simple formula for* $\sum_{k=1}^{n}(2k-1)$*".*

Solution:

$\sum_{k=1}^{n}(2k-1) = 2\sum_{k=1}^{n} k - \sum_{k=1}^{n} 1 = n(n+1) - n = n^2$

Verify by induction:

- **Base case:** $n = 1 \Rightarrow \sum_{k=1}^{1}(2k-1) = 1$.
- **Inductive hypothesis:** Assume that $\sum_{k=1}^{n}(2k-1) = n^2$
- **Inductive step:** Show for $n + 1$:

$$\sum_{k=1}^{n+1}(2k-1) = \sum_{k=1}^{n}(2k-1) + 2(n+1) - 1 = n^2 + 2n + 1 = (n+1)^2$$

p. 1062 **A.1–7** *"Evaluate the product* $\prod_{n}^{k=1} 2 \cdot 4^k$*."*

Solution:

$\prod_{k=1}^{n} 2 \cdot 4^k = 2^{n^2+2n}$ (Hint: take the logarithm of the product.)

p. 1067 **A.2–1** *"Show that* $\sum_{k=1}^{n} 1/k^2 < c$ *is bounded above by a constant."*

Solution:

Since $1/k^2$ is a decreasing function we can bound the sum from above with an integral.

$$\sum_{k=1}^{n} 1/k^2 = 1 + \sum_{k=2}^{n} 1/k^2 \leq 1 + \int_1^n 1/x^2 dx = 2 - 1/n < 2$$

You have to handle the first term separately to avoid dividing by 0 once the integral has been solved.

p. 67 **4.1–1** *"Show that the solution of* $T(n) = T(\lceil n/2 \rceil) + 1$ *is* $O(\lg n)$*."*

Solution:

It is enough to show that $T(n) < c \cdot \lg n$. Suppose first (inductive hypothesis) that $T(k) < c \lg k$ for all $k < n$. Then show that the inductive hypothesis is also true for $k = n$:

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + 1 \\ &< c \lg(n/2) + 1 \qquad \text{(using the IH)} \\ &= c(\lg n - \lg 2) + 1 \\ &= c \lg n - (c - 1) \\ &< c \lg n \text{ om } c > 1 \end{aligned}$$

sid 67 **4.1–2** *"We saw that the solution of* $T(n) = 2T(\lceil n/2 \rceil) + n$ *is* $O(n \lg n)$*. Show that the solution of this recurrence is also* $\Omega(n \lg n)$*. Conclude that the solution is* $\Theta(n \lg n)$*."*

Solution:

To show that $T(n) = \Omega(n \lg n)$ it is enough to show that $T(n) > cn \lg n$. First assume (inductive hypothesis) that $T(k) > ck \lg k$ for all $k < n$. Then show that the inductive hypothesis is true also for $k = n$:

$$\begin{aligned} T(n) &= 2T(\lfloor n/2 \rfloor) + n \\ &> 2c(n/2) \lg(n/2) + n \qquad \text{(using the IH)} \\ &= cn(\lg n - 1) + n \\ &= cn \lg n - cn + n \\ &= cn \lg n + n(1 - c) \\ &> cn \lg n \text{ if } c < 1 \end{aligned}$$

On page 64 they show that $T(n)$ is also $O(n \lg n)$. Taken together, this gives us that $T(n)$ is $\Theta(n \lg n)$.

p. 67 **4.1–6** *"Solve the recurrence* $T(n) = 2T(\sqrt{n}) + 1$ *by making a change of variables. Your solution should be asymptotically tight. Do not worry about whether values are integral."*

Solution:

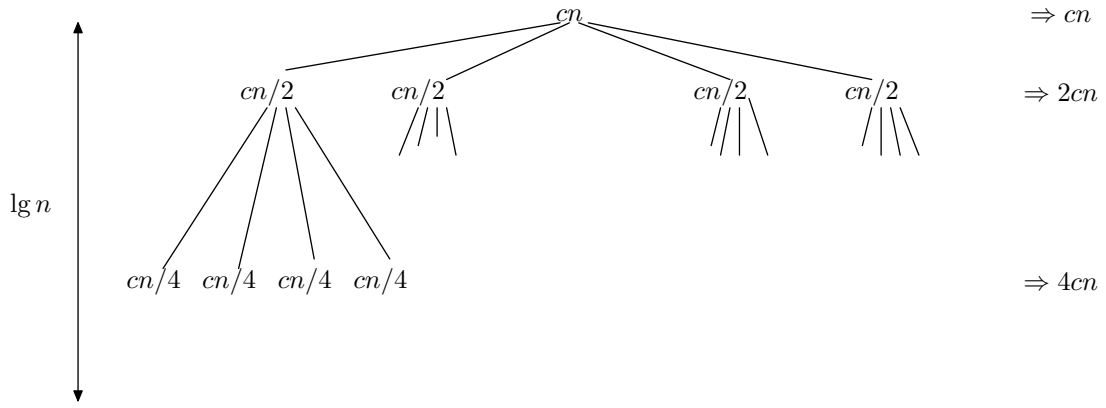
Change variables so that $m = \lg n \Rightarrow n = 2^m$ and the recurrence becomes $T(2^m) = 2T(2^{m/2}) + 1$.

Now let $S(m) = T(2^m)$ and write the recurrence as $S(m) = 2S(m/2) + 1$, the solution being $S(m) = O(\lg m)$.

Going back, this gives us $T(n) = T(2^m) = S(m) = O(\lg m) = O(\lg \lg n)$.

p. 72 **4.2–3** *"Draw the recursion tree for* $T(n) = 4T(\lfloor n/2 \rfloor) + cn$*, where* c *is a constant, and provide a tight asymptotic bound on its solution. Verify your bound by the substitution method."*

Solution:



The total sum is

$$T(n) = cn \sum_{i=0}^{\lg n} 2^i = cn(2^{1+\lg n} - 1) = cn(2n - 1) = 2cn^2 - cn = O(n^2)$$

Now you must verify that $T(n) < dn^2$. Assume that $T(k) < dk^2$ for all $k < n$ and then show it for $k = n$:

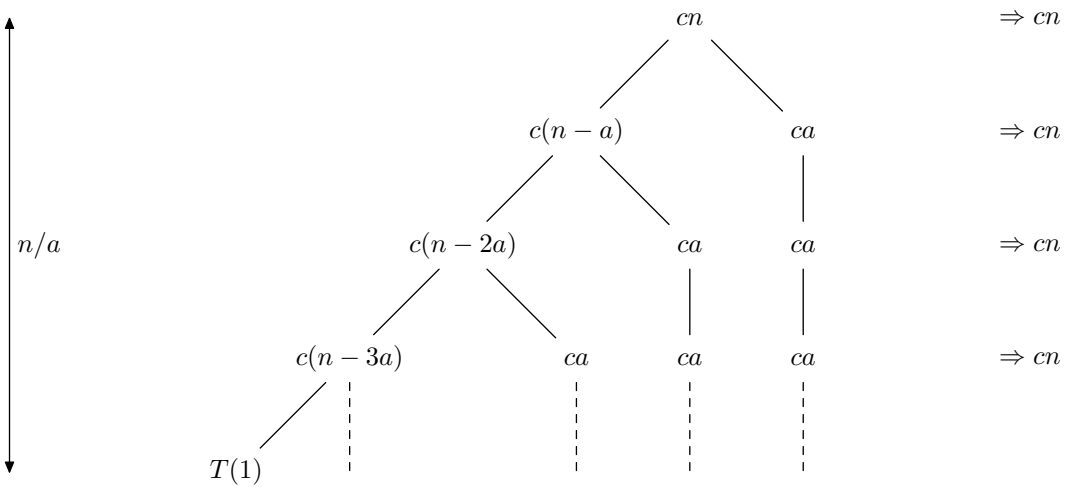
$$\begin{aligned} T(n) &= 4T(\lfloor n/2 \rfloor) + cn \\ &< 4d(n/2)^2 + cn \\ &= dn^2 + cn \end{aligned}$$

Obviously, we won't succeed here. In these cases, you can try to strengthen the inductive hypothesis by subtracting a lower-order term: Assume that $T(k) < dk^2 - ek$ and then try to show $T(n) < dn^2 - en$.

$$\begin{aligned} T(n) &= 4T(\lfloor n/2 \rfloor) + cn \\ &< 4(d(n/2)^2 - e(n/2)) + cn \\ &= dn^2 - 2en + cn \\ &= dn^2 - en - (en - cn) \\ &= dn^2 - en - n(e - c) \\ &< dn^2 - en \text{ if } e > c \end{aligned}$$

p. 72 **4.2–4** *"Use a recursion tree to give an asymptotically tight solution to the recurrence* $T(n) = T(n - a) + T(a) + cn$ *where* $a \geq 1$ *and* $c > 0$ *are constants."*

Solution:



The tree ends at $n = 1$ which happens when $n - ka = 1$, i.e., $k = (n-1)/a \approx n/a$ levels. The total sum is $T(n) = (n/a)cn = O(n^2)$.

Now verify that $T(n) = O(n^2)$, i.e., that $T(n) < dn^2$.

Assume that $T(k) < dk^2$ for all $k < n$ (inductive hypothesis), then show it for $k = n$:

$$\begin{aligned} T(n) &= T(n-a) + T(a) + cn \\ &< d(n-a)^2 + da^2 + cn \\ &= dn^2 - 2adn + 2da^2 + cn \\ &= dn^2 - (2adn - 2da^2 - cn) \end{aligned}$$

In order for the residual $(2adn - 2da^2 - cn)$ to be greater than 0, we can choose, e.g., $d = (c+1)/2a$ in which case we get $T(n) < dn^2 - (n - ac - a)$ which will be less than dn^2 for large values of n, i.e., $T(n) = O(n^2)$.

p. 75 **4.3–1** *"Use the master method to give tight asymptotic bounds for the following recurrences.*

a. $T(n) = 4T(n/2) + n$.

b. $T(n) = 4T(n/2) + n^2$.

c. $T(n) = 4T(n/2) + n^3$."

Solution:

a) $T(n) = 4T(n/2) + n$. Using the Master method, $a = 4$, $b = 2$ and $f(n) = n$ which gives $n^{\log_b a} = n^2$. Is $f(n) = O(n^{2-\epsilon})$? Yes, if $\epsilon = 1$. This means that $T(n) = \Theta(n^2)$.

b) $T(n) = 4T(n/2) + n^2$. Here we have $a = 4$, $b = 2$ and $f(n) = n^2$ which gives $n^{\log_b a} = n^2$. There is only one choice, $f(n) = \Theta(n^2)$ which means that $T(n) = \Theta(n^2 \lg n)$.

c) $T(n) = 4T(n/2) + n^3$. Here, $a = 4$, $b = 2$ and $f(n) = n^3$ which gives $n^{\log_b a} = n^2$. This time $f(n) = \Omega(n^2 + \epsilon)$ if $\epsilon = 1$, if $4f(n/2) \le cf(n)$, i.e., if $4(n/2)^3 \le cn^3$, which is true for, e.g., $c = 1/2$. Thus, $T(n) = \Theta(n^3)$.

p. 75 **4.3–2** *"The recurrence* $T(n) = 7T(n/2) + n^2$ *describes the running time of an algorithm* A*. A competing algorithm* A' *has a running time of* $T'(n) = aT'(n/4) + n^2$*. What is the largest integer value for* a *such that* A' *is asymptotically faster than* A*?"*

Solution:

$T(n) = 7T(n/2) + n^2$. The master method gives us $a = 7$, $b = 2$ and $f(n) = n^2$. Since $n^{\log_b a} = n^{\lg 7} \approx n^{2.8}$ it is the first case of the master method that we investigate: Is $f(n) = n^2 = O(n^{\lg 7 - \epsilon})$? Yes, with $\epsilon \approx 0.8$, which gives us $T(n) = \Theta(n^{\lg 7})$.

The other recurrence $T'(n) = aT'(n/4) + n^2$ is a bit more difficult to analyze since it's not so easy to say which of the three cases applies in the master method when a is unknown.

However, $f(n)$ are the same in both algorithms which leads us to try the first case. There, $\log_4 a$ must be $\lg 7$, which happens when $a = 48$. In other words, A' is asymptotically faster than A as long as $a < 48$.

The other cases in the master method do not apply for $a > 48$.

Hence, A' is asymptotically faster than A up to $a = 48$.

p. 85 **4–1** *"Give asymptotic upper and lower bounds for* $T(n)$ *in each of the following recurrences. Assume that* $T(n)$ *is constant for* $n \leq 2$*. Make your bounds as tight as possible, and justify your answers.*

a. $T(n) = 2T(n/2) + n^3$.

b. $T(n) = T(9n/10) + n$.

c. $T(n) = 16T(n/4) + n^2$.

d. $T(n) = 7T(n/3) + n^2$.

e. $T(n) = 7T(n/2) + n^2$.

f. $T(n) = 2T(n/4) + \sqrt{n}$.

g. $T(n) = T(n-1) + n$.

h. $T(n) = T(\sqrt{n}) + 1$."

Solution:

a) $T(n) = 2T(n/2) + n^3$. With $a = 2$, $b = 2$ and $f(n) = n^3$ we get $n^{\log_b a} = n$ and $f(n) = \Omega(n^{1+\epsilon})$ for $\epsilon = 2$, because $2f(n/2) = cf(n)$ if $c = 1/4$. Thus, $T(n) = \Theta(n^3)$.

b) $T(n) = T(9n/10) + n$. With $a = 1$, $b = 10/9$ and $f(n) = n$ we get $n^{\log_b a} = n^0$ and $f(n) = \Omega(n^{0+\epsilon})$ with $\epsilon = 1$, because $f(9n/10) \leq cf(n)$ for large values of n if $\frac{9}{10} < c < 1$. Thus, $T(n) = \Theta(n)$.

c) $T(n) = 16T(n/4) + n^2$. With $a = 16$, $b = 4$ and $f(n) = n^2$ we get $n^{\log_b a} = n^2$ and $f(n) = \Theta(n^2)$, thus $T(n) = \Theta(n^2 \lg n)$.

d) $T(n) = 7T(n/3) + n^2$. With $a = 7$, $b = 3$ and $f(n) = n^2$ we get $n^{\log_b a} = n^{\log_3 7 + \epsilon})$. Since $1 < \log_3 7 < 2$ we can choose $\epsilon = 2 - \log_3 7$ so that $f(n) = \Omega(n^{\log_3 7 + \epsilon})$ and obtain $T(n) = \Theta(n^2)$.

e) See problem 4.3-2: $T(n) = \Theta(n^3)$.

f) $T(n) = 2T(n/4) + \sqrt{n}$. With $a = 2$, $b = 4$ we get $n^{\log_b a} = \sqrt{n}$ which directly leads us to conclude that $T(n) = \Theta(\sqrt{n} \lg n)$.

g) $T(n) = T(n-1) + n$. In this case the master method does not apply, but you can draw a recursion tree and obtain $T(n) = \Theta(n^2)$.

h) $T(n) = T(\sqrt{n}) + 1$. The master method doesn't apply here either, but by drawing a recursion tree you can see that $T(n) = \Theta(\lg \lg n)$.

p. 1091 **B.5–3** *"Show by induction that the number of degree-2 nodes in any nonempty binary tree is 1 less than the number of leaves."*

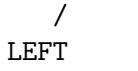
Solution:

We are required to show by induction that a binary tree fulfills the following condition:

The number of nodes with degree 2 = the number of leaves -1.

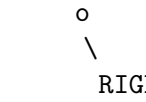
- **Base case:** A tree consisting of a single leaf node doesn't have any nodes of degree 2 and since the number of leaves is 1, the condition is trivially true.
- **Inductive hypothesis:** Assume that the condition holds for all subtrees `LEFT` and `RIGHT`.
- **Inductive step:** The following cases must be handled:

a)



This tree has the same number of leaf nodes as `LEFT` and the same number of degree-2 nodes, so the condition is true.

b)



This tree has the same number of leaf nodes as `RIGHT` and the same number of degree-2 nodes, so the condition is true.

c)

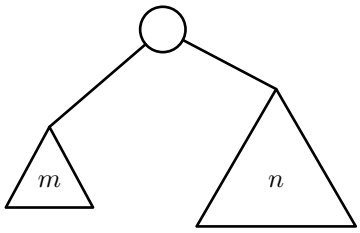
Assume that `LEFT` and `RIGHT` have m and n number of leaf nodes, respectively. From the inductive hypothesis we can conclude that `LEFT` and `RIGHT` have $m-1$ and $n-1$ degree-2 nodes, respectively. Taken together, the tree shown above has $m+n$ leaves and $1+m+n-2$ degree-2 nodes, so the condition is still true.

p. 1091 **B.5–4** *"Use induction to show that a nonempty binary tree with n nodes has height at least $\lfloor \lg n \rfloor$."*

Solution:

- **Base case:** A tree consisting of only one leaf node has the height $0 \geq \lfloor \lg 1 \rfloor$, so the condition is true.

- **Inductive hypothesis:** Assume the condition holds for two binary trees with m and n nodes, respectively.
- **Inductive step:**



Height of the left subtree with m nodes, $h_m \geq \lfloor \lg m \rfloor$.
Height of the right subtree with n nodes, $h_n \geq \lfloor \lg n \rfloor$.
Let h be the height of the whole tree with $1 + m + n$ nodes.
According to the picture, the following must hold:

$$h = \max(h_m, h_n) + 1 \geq \max(\lfloor \lg n \rfloor, \lfloor \lg m \rfloor) + 1$$

We can assume that $m \leq n$ since the following argument can also be carried out if $m > n$. Let us first simplify:

$$h = h_n + 1 \geq \lfloor \lg n \rfloor + 1$$

We shall now perform the inductive step and show that

$$h \geq \lfloor \lg(1 + m + n) \rfloor$$

If $\lfloor \lg n \rfloor + 1 \geq \lfloor \lg(1 + m + n) \rfloor$, then the condition is true. The "worst" situation we can face is when $m = n$ so we need to show that

$$\lfloor \lg n \rfloor + 1 \geq \lfloor \lg(1 + 2n) \rfloor$$

Since $2n$ is an even number, the right part of the equality is $\lfloor \lg 2n \rfloor$, due to the truncation. It then follows that

$$\lfloor \lg 2n \rfloor = \lfloor \lg 2 + \lg n \rfloor = \lfloor 1 + \lg n \rfloor$$

which reduces the inequality to

$$\lfloor \lg n \rfloor + 1 \geq \lfloor 1 + \lg n \rfloor$$

which is always true.

p. 1092 **B–2**

a) The statement "in any group of $n \geq 2$ people, there are two people with the same number of friends in the group" can be rephrased in terms of a graph such that "in a graph with more than two nodes, there must be two nodes with the same degree".

Proof: Assume we have n nodes. If the graph is connected, the maximum degree for a node is $n - 1$. (If the graph is not connected but rather has an isolated node of degree 0, then the maximum degree is $n - 2$, etc.)

In that case there are simply not enough nodes to distribute all the $n - 1$ different degrees uniquely. In other words, at least two nodes must have the same degree.