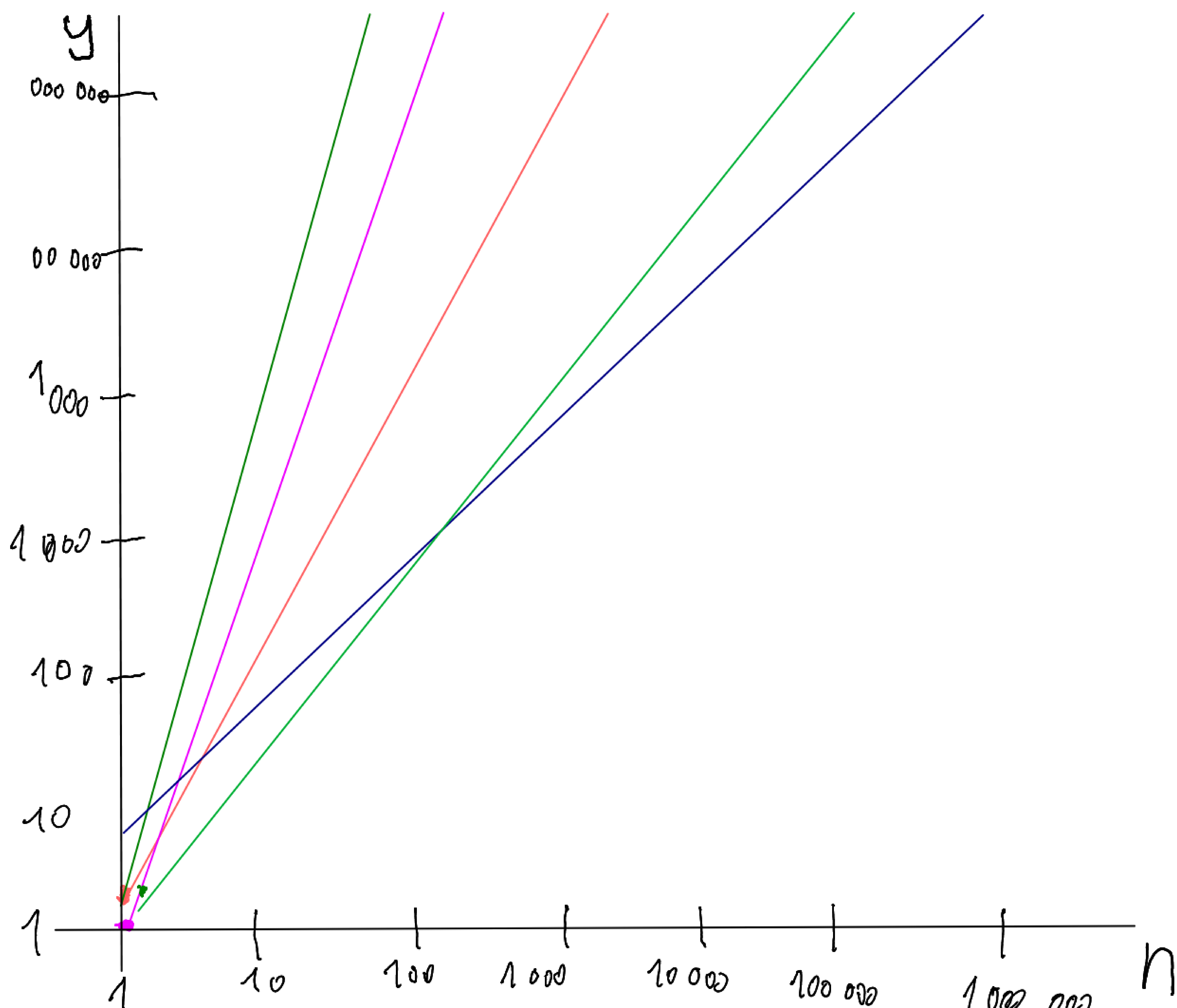
Chapter 3 answers

3.1



$y = 8n$

n	log n	y	log y
1	0	8	0.9
10	1	80	1.9
100	2	800	2.9
1000	3	8000	3.9
10 000	4	80 000	4.9
100 000	5	800 000	5.9
000 000	6	8 000 000	6.9

$y = 4n \log n$

n	log n	y	log y
1	0	0	undefined
10	1	40	1.6
100	2	800	2.9
1 000	3	12 000	4.08
10 000	4	16 000	5.2
100 000	5	2 000 000	6.3
1 000 000	6	24 000 000	7.4

$y = 2n^2$

n	log n	y	log y
1	0	2	0.3
10	1	200	2.3
100	2	20 000	4.3
1 000	3	2 000 000	6.3
10 000	4	200 000 000	8.3
100 000	5		
1 000 000	6		

$y = n^3$

n	log n	y	log y
1	0	1	0
10	1	1000	3
100	2	1 000 000	6
1 000	3	100 000 000	9
10 000	4		
100 000	5		

$y = 2^n$

n	log n	y	
1	0	2	0
10	1	1024	3
20	1.3	1 048 576	6.0
30	1.5	1 073 741 824	1.03

3.2

$8n\log_2 n = 2n^2$

$\log_2 n = n/4$

$n = 2^{n/4}$

$n = 16$

Therefore, $n_0 = 16$. This means that algorithm A, whose number of operations is $8n\log n$, starts outperforming algorithm B, whose number of operations is $2n^2$ at $n > 16$.

3.3

$40n^2 = 2n^3$

$0 = 2n^3 - 40n^2$

$0 = 2n^2(n-20)$

$n = 0, n = 20$

Therefore, $n_0 = 20$. This means that algorithm A, whose running time is $40n^2$ starts outperforming algorithm B, whose running time is $2n^3$ at $n > 20$

3.4

a constant function

3.5

$f(n) = n^c$

$y = f(n)$

$y = n^c$

$\log y = \log n^c$

$\log y = c\log n$

Because we can rewrite $\log n^c$ as $c\log n$, and the horizontal axis's is $\log n$, so it's the same as having $c \cdot x$ for a graph with a regular x axis.

3-6

Sum of all even numbers from 0 to $2n$

$n = 5$

$S_1 = 0 + 2 + 4 + 6 + 8 + 10$

$S_2 = 10 + 8 + 6 + 4 + 2 + 0$

$10 \quad 10 \quad 10 \quad 10 \quad 10 \quad 10 = 60$

$2S_n = 6 \times 10$

$$S_n = \frac{\underbrace{(n+1) \cdot 2n}}{2} = n(n+1)$$

3.7

If the running time is always $O(f(n))$, then $O(f(n))$ is an upper bound on the running time in all cases, including the worst case.

3.8

Fastest

2^{10}

$2^{\log n}$ ← same as $y = n$:

$y = 2^{\log_2 n}$

$\log_2 y = \log_2 2^{\log_2 n}$

$\log_2 y = \log_2 n \cdot \log_2 2$

$\log_2 y = \log_2 n$

$y = n$

$3n + 100 \log n$

$4n$

$n \log n$

$4n \log n + 2n$

$n^2 + 10n$

n^3

2^n

Slowest

3.9

if $d(n)$ is $O(f(n))$, then

$$d(n) \leq c\, f(n)$$

therefore $a\,d(n)$ is also $O(f(n))$, because we can multiply c by a to get a different constant

3.10

if $d(n)$ is $O(f(n))$, then:

$$d(n) \leq c \cdot f(n)$$

and $e(n)$ is $O(g(n))$, so:

$$e(n) \leq k \cdot g(n)$$

therefore

$$d(n) \cdot e(n) \leq c \cdot k \cdot f(n) \cdot g(n)$$

therefore

$$d(n) \cdot e(n) \text{ is } O(f(n) \cdot g(n))$$

3.11

if $d(n)$ is $O(f(n))$, then:

$$d(n) \leq c \cdot f(n)$$

and $e(n)$ is $O(g(n))$, so:

$$e(n) \leq k \cdot g(n)$$

therefore $d(n) + e(n) \leq c \cdot f(n) + k \cdot g(n)$

therefore $d(n) + e(n)$ is $O(f(n) + g(n))$

3.12

let's consider the following

$d(n) = 2n$
$e(n) = n$

Both are $O(n)$, so $f(n)=n$ and $g(n)=n$

$d(n)-e(n) = n$, which is $O(n)$

But $O(f(n)-g(n))=O(n-n)=O(1)$

3.13

$d(n) \leq c_1 \cdot f(n)$

$f(n) \leq c_2 \cdot g(n)$

$\therefore d(n) \leq c_1 \cdot c_2 \cdot g(n)$

$\therefore d(n)$ is $O(g(n))$

3.14

$O(\max\{f(n), g(n)\})$ is $O(f(n)+g(n))$

because $\max\{f(n), g(n)\} < f(n)+g(n)$, assuming $f(n)>0, g(n)>0$

3.15

Show that $f(n)$ is $O(g(n))$ if and only if $g(n)$ is $\Omega(f(n))$

From big O definition:

if $f(n)$ is $O(g(n))$, then

$f(n) \leq c_1 \cdot g(n)$

From big omega definition:

if $g(n)$ is $\Omega(f(n))$, then:

$g(n) \geq c_2 \cdot f(n)$

3.16

$p(n) = a_1 n + a_2 n^2 + a_3 n^3 + a_n n^d$

$p(n) \leq (a_1 + a_2 + a_3 + a_n) n^d$

$\log(p(n)) \leq \log\left((a_1 + a_2 + a_3 + a_n) n^d\right)$

$\log(p(n)) \leq d \cdot \log\left((a_1 + a_2 + a_3 + a_n) n\right)$

$\therefore \log(p(n))$ is $O(\log(n))$

3.17

$(n+1)^5 = n^5 + 5n^4 + 10n^3 + 10n^2 + 5n + 1$

$(n+1)^5 \leq (1+5+10+10+5+1) n^5$

$(n+1)^5$ is $O(n^5)$

3.18

$2^{n+1} = 2 \cdot 2^n$, so we have

$2^{n+1} \leq c \cdot 2^n$ for $n \geq n_0$

by choosing $c = 2$ and $n_0 = 1$

3.19

$n \leq c \cdot n \cdot \log(n) \quad / \div n$

$1 \leq c \cdot \log(n)$

holds for $n = 10$, $c = 1$

3.20

$n^2 \geq c \cdot n \cdot \log(n) \quad / \div n$

$n \geq c \cdot \log(n)$

holds for all $n > 0$ if we choose $c = 1$

3.21

$n \log n \geq cn \quad / \div n$

$\log n \geq c$

holds for $n \geq 10$, $c = 1$

3.22

$f(n) \leq c_1 f(n)$

$\lceil f(n) \rceil \leq f(n) + 1$

$\therefore$ we can choose c_2 such that

$f(n) + 1 \leq c_2 f(n)$

assuming $f(n)$ is a positive non-decreasing function greater than 1

3.23

loop with n iterations, so $O(n)$

3.24

loop with $\frac{n}{2}$ iterations, so $O(n)$

3.25

loop with $1 + 2 + 3 + \ldots + n$ iterations, which is $\frac{n(n+1)}{2}$, so $O(n^2)$

3.26

loop with n iterations, so $O(n)$

3.27 $O(n^3)$

3.28

	1 second	1 hour	1 month
$\log n$	$\approx 10^{3\times10^5}$	$\approx 10^{10^9}$	$10^{7.2\times10^{11}}$
n	10^6	3.6×10^9	2.4×10^{12}
$n \log n$	63K	$\approx 10^8$	6.6×10^{10}
n^2	1000	60K	1.5×10^6
2^n	20	32	41

3.29

$O(n \log n)$

3.30

$O(n \log n)$

3.31

best case: $n \log n$

worst case: n^2

3.32

$O(n^2)$

3.33

The $O(n \log n)$ algorithm could have a greater constant K, so it only starts beating the $O(n^2)$ algorithm at larger inputs

3.34

Assuming uniform distribution, the probability that the j^{th} meal they experience is the best meal they have experienced is $\frac{1}{j}$.

So the expected number of best meals is $\sum_j^n \frac{1}{j}$, which is the n^{th} Harmonic number, which is $O(\log n)$.

3.35

1. Put all numbers into single sequence - $O(n)$
2. Sort new joined sequence $O(n \log n)$
3. Iterate over sequence, checking numbers which are next to each other. If there are no same numbers next to each other, then the sets are disjoint.

3.36

Iterate over entire sequence. Keep a second list of numbers for finding 10 highest. Keep that list ordered. For each number in sequence, check if it is greater than smallest number in the list. If it is, remove smallest number from list and insert current number at right position. $O(n)$

3-37

$n^2(1+\sin(n))$

3.38

$$\sum_{i=1}^{n} i^2 < \int_0^{n+1} x^2 \, dx$$

because the graph of x^2 from 0 to $n+1$ has width $n+1$ and height x^2.

$$\int_0^{n+1} x^2 \, dx = \frac{(n+1)^3}{3}$$

$\therefore$ it is $O(n^3)$, since we can find Kn^3 which binds $\frac{(n+1)^3}{3}$

3.39

$$S = \sum_{i=1}^{n} \frac{i}{2^i}$$

$$= \frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \frac{5}{32} + \frac{6}{64} + \dots + \frac{n}{2^n}$$

$$= \sum_{i=1}^{n} \frac{1}{2^i} + \sum_{i=2}^{n} \frac{i-1}{2^i}$$

$$= \left\{\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{2^n}\right\} + \left\{\frac{1}{4} + \frac{2}{8} + \frac{3}{16} + \dots + \frac{n-1}{2^n}\right\}$$

$$= \sum_{i=1}^{n} \frac{1}{2^i} + \sum_{i=1}^{n-1} \frac{i}{2^{i+1}}$$

$$\underbrace{\sum_{i=1}^{n} \frac{1}{2^i} + \sum_{i=1}^{n-1} \frac{i}{2^{i+1}}}_{S} < 1 + \frac{S}{2}$$

since $\sum_{i=1}^{n} \frac{1}{2^i} < 1$

and $\sum_{i=1}^{n-1} \frac{i}{2^{i+1}} < \frac{S}{2}$, because it is the same as $S/2$ but only up to $n-1$

$$S < 1 + \frac{S}{2}$$

$$S < 2$$

3.40

$$\log_b f(n) = \frac{\log f(n)}{\log b} = \frac{1}{\log b} \cdot \log f(n) = c \cdot \log f(n)$$

3.41

- pair up the numbers in the sequence
- compare numbers in each pair, taking $n/2$ comparisons. All the greater numbers now form a group of maximum candidates, and the smaller numbers form a group of minimum candidates
- we can now find the min and the max in $\frac{n}{2} - 1$ comparisons each

total: $\frac{n}{2} + \left(\frac{n}{2} - 1\right) + \left(\frac{n}{2} - 1\right) = \frac{3n}{2} - 2$

3.42

total visits if each friend visited one less time than allowed:

$$\underbrace{0 + 1 + 2 + \ldots\ldots + n-1}_{n \text{ of these}}$$

$= \frac{n(n-1)}{2}$ so if there is one more visit than that, then one friend must have visited the maximum allowed number of times $\therefore \frac{n(n-1)}{2} + 1$

3.44

loop runs $\approx 2^{100}$ times in the worst case, performing 2^{100} divisions. Each division takes 10^{-6} seconds, so the function would run $\approx 10^{24}$ seconds $\approx 1.5 \times 10^{19}$ days $\approx 4 \times 10^{16}$ years

3.45