Svar til øvelser i Algoritmer og Datastrukturer

Skrevet af Kim

Studieåret 2022-2023

Indhold

Kapitel 1

Uge 1: Bevisteknikker og Løkke-invarianter

1.1 Opgave 1 formulering

Bevis at $x^2 - y^2 = 1$ ikke har nogen løsninger for positive heltal x, y (de positive heltal er tallene $\{1, ...\}$).

1.1.1 Svar til opgave 1

Fremgangsmåde: Vi vil benytte modstridsbevis ved at antage at $x^2 - y^2 = 1$ har løsninger for positive heltal og derfra yderligere deducere os frem til et udtryk vi ved ikke er sandt, for så at slutte at $x^2 - y^2 = 1$ ikke kan have løsnigner for positive heltal.

Bevis: Antag for modstrid at $x^2 - y^2 = 1$ har løsninger af positive heltal for $x, y \in \mathbb{Z}$, da vi bemærker at de positive heltal blot er de naturlige tal.

Vi bemærker de positive heltalsløsninger skal være sandt for alle af de 3 tilfælde, $x < y$, $x > y$ og $x = y$, hvilket evalueres:

$x < y$: Vi bemærker at hvis $x < y$ da medfører det at $x^2 < y^2$, for alle $x, y \in \mathbb{N}$, derfor er $x^2 - y^2 = 1 \Leftrightarrow x^2 = y^2 + 1 \Leftrightarrow x = y + r$, hvor $0 < r \in \mathbb{R}$ hvilket medfører $x > y$, som er en modstrid for vores grundantagelse $x < y$ ↯

$x > y$: Dette gøres symmetrisk som tilfældet $[x < y]$.

Vi bemærker at hvis $x > y$ da medfører det at $x^2 > y^2$, for alle

$x, y \in \mathbb{N}$, derfor er $x^2 - y^2 = 1 \Leftrightarrow -y^2 = -x^2 + 1 \Leftrightarrow y^2 = x^2 - 1$ og siden at $x^2 - 1 \geq 0$, da har vi at $y^2 = x^2 - 1 \Leftrightarrow y = x - r$, hvor $0 < r \in \mathbb{R}$ hvilket medfører $x < y$, hvilket er en modstrid for vores grundantagelse at $x > y$ $\lightning$

$x = y$: Dette er trivielt, da $x^2 - y^2 = 1$ for $x = y$ hvilket giver
$x^2 - y^2 = x^2 - (x)^2 = x^2 - x^2 = 0 \neq 1$ $\lightning$

1.2 Opgave 2 formulering

Bevis at $x^2 - y^2 = 10$ ikke har nogen løsninger for positive heltal x, y.

1.2.1 Svar til opgave 2

Vi vil benytte modstridsbevis ved at antage at $x^2 - y^2 = 10$ har løsninger for positive heltal og derfra yderligere deducere os frem til et udtryk vi ved ikke er sandt, for så at slutte at $x^2 - y^2 = 10$ ikke kan have løsnigner for positive heltal.

Bevis: Antag for modstrid at $x^2 - y^2 = 10$ har løsninger af positive heltal for $x, y \in \mathbb{Z}$.

Vi bemærker de positive heltalsløsninger skal være sandt for alle af de 3 tilfælde, $x < y$, $x > y$ og $x = y$, hvilket evalueres:

$x < y$: Vi bemærker at hvis $x < y$ da medfører det at $x^2 < y^2$, for alle $x, y \in \digamma$, derfor er $x^2 - y^2 = 10 \Leftrightarrow x^2 = y^2 + 10 \Leftrightarrow x = y + r$, hvor $0 < r \in \mathbb{R}$ hvilket medfører $x > y$, som er en modstrid for vores grundantagelse $x < y$ $\lightning$

$x > y$ Vi vil starte med at faktorisere udtrykket, sådan at $x^2 - y^2 = 10 \Leftrightarrow (x + y)(x - y) = 10$, vi ser da at for $x > y$ kan dette kun lade sig gøre på 5 måder, nemlig:

1) $(x + y) = 1$ og $(x - y) = 10$.
For $x > y$ ser vi at $(x+y) = 1$ ikke kan lade sig gøre, da $\min\{y\} = \min\{x\} = 1$, så hvis $(x + \min\{y\})$ for $x > 1$, da er $1 < (x + \min\{y\}) \neq 1$ $\lightning$
2) $(x + y) = 10$ og $(x - y) = 1$.
Vi ser for $x > y$ at $(x - y) = 1$ kun kan være rigtigt for $x = y + 1$, men da ville $(x + y) \neq 10$ fordi $y_1 = 4ogy_2 = 5$, ville hhv. give $x_1 = 4 + 1 = 5$ og $x_2 = 5 + 1 = 6$ $\lightning$

3) $(x+y) = 2$ og $(x-y) = 5$.
Vi ser at for $x > y$ da vil $(x+y) = 2$ ikke kunne lade sig gøre, da $\min\{y\} = \min\{x\} = 1$, men vores grundantagelse $x > y$, så $y \neq x$ ↯

4) $(x+y) = 5$ og $(x-y) = 2$.
Vi ser for $x > y$ at $(x-y) = 2$ kun kan være rigtigt for $x = y+2$, men da ville $(x+y) \neq 5$ fordi $y_1 = 1 og y_2 = 2$, ville hhv. give $x_1 = 1+2 = 3$ og $x_2 = 2+2 = 4$ ↯

5) $(x+y) = (x-y) = \sqrt{10}$.
Trivielt for $(x+y) = (x-y) = \sqrt{10}$, da $x \neq y$, hvilket er i modstrid med vores grundantagelsen $x > y$ ↯

$x = y$: Dette er trivielt, da $x^2 - y^2 = 10$ for $x = y$ hvilket giver
$x^2 - y^2 = x^2 - (x)^2 = x^2 - x^2 = 0 \neq 10$ ↯

1.3 Opgave 3

Bevis at hvis a er et rationelt tal og b er et irrationelt tal, så er $a+b$ et irrationelt tal. (Et rationelt tal er et tal der kan skrives på formen $\frac{x}{y}$, hvor x og y er heltal og $y \neq 0$.)

1.3.1 Svar til opgave 3

Antag for modstrid at $a+b$ er rationel, da ville $a+b = c \in \mathbb{Q}$, derfor er

$$a+b = c \Leftrightarrow b = c - a \Leftrightarrow b = \frac{x_1}{y_1} + \frac{x_2}{y_2} = \frac{x_1 y_2 + x_2 y_1}{y_1 y_2} = \frac{x}{y} \quad ↯$$

1.4 Opgave 10

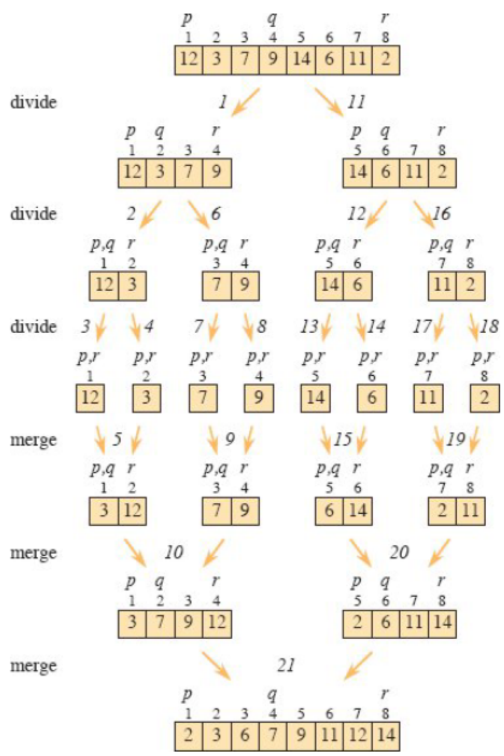
1.4.1 Svar til opgave 10

1.5 1 MergeSort Opgaveformulering

Antag at `MergeSort(A, p, r)`, implementeret som i CLRS sektion 2.3, bliver kaldt på 21 elementer (dvs. $r - p + 1 = 21$). Hvor mange kald bliver der totalt lavet til `Merge-Sort`? Hvad er antallet af kald generelt, når input har n elementer? Argumentér for dine svar.

1.5.1 Svar til 1 MergeSort

Vi ser fra CLRS figure 2.4, side 74(pdf).



Først deler vi array'et 1 gang, så deles array'et 2 gange, derefter 4 gange. Summen af disse delinger ville da være $1+2+4=7$, altså er formlen $2n-1$. Vi kan nu benytte denne formel til at finde hvor mange gange MergeSort bliver kaldt, for 21 elementer, hvilket så er $2 \cdot 21 - 1 = 42 - 1 = 41$.

1.6 Rekursionsligninger opgaveformulering (aflevering 1)

Hvilke af disse rekursionsligninger har løsningen $T(n) = \Theta(n2)$? Antag at $T(n) = 1$ for $n \geq 1$. Vælg ét eller flere korrekte svar og beskriv hvordan du kom frem til dem.

1. $T(n) = 4T(\lfloor n/2 \rfloor) + n \lg n$

2. $T(n) = 4T(\lfloor n/2 \rfloor) + n^2$

3. $T(n) = 2T(\lfloor n/4 \rfloor) + n^2$

4. $T(n) = 9T(\lfloor n/3 \rfloor) + n^3$

5. $T(n) = T(n-1) + n^2$

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

1.6.1 Svar til Rekursionsligninger (aflevering 1)

CLRS Theorem 4.1(Master Theorem) som løser rekurrencerelationer af formen: $T(n) = aT(n/b) + f(n^k \lg^p n)$, for $a \geq 1$, $b > 1$, $k \geq 0$ og p er et reel. Det deles op i 3 tilfælde.

1. Hvis $a > b^k$ så er $T(n) = f(n^{\lg_b a})$
2. Hvis $a = b^k$ og
   - Hvis $p < -1$ så er $T(n) = f(n^{\lg_b a})$
   - Hvis $p = -1$ så er $T(n) = f(n^{\lg_b a \cdot \lg^2 n})$
   - Hvis $p > -1$ så er $T(n) = f(n^{\lg_b a \cdot \lg^{p-1} n})$
3. Hvis $a < b^k$ og
   - Hvis $p < 0$ så er $T(n) = \mathcal{O}(n^k)$
   - Hvis $p \geq 0$ så er $T(n) = f(n^k \lg^p n)$

Kapitel 2

Uge 2: Asymptotisk køretid og Recurrenceligninger

2.1 Opgave 1 fra AU formulering Assymptotisk Køretid

I det følgende angiver lg n 2-tals-logaritmen af n. (Multiple-choice: Ja/Nej)

1) $(\lg\ n)^2$ er $\mathcal{O}(n^2)$
2) $n \lg\ n$ er $\mathcal{O}(n^2)$
3) $\sqrt{n}$ er $\mathcal{O}\left((\lg n)^3\right)$
4) $1 + \lg n^2$ er $mathcalO\left((\lg n)^2\right)$
5) $\lg(n) + \lg(n!)$ er $\mathcal{O}(n^2)$
6) n^3 er $\mathcal{O}(n)$
7) $\sqrt{n} \cdot \lg(n)$ er $\mathcal{O}(n)$
8) n^3 er $\mathcal{O}(\lg(n!))$
9) n^2 er $\mathcal{O}(n^{2/3})$
10) $7\lg(n) + \lg(n!)$ er $\Theta(n \cdot \lg(n))$
11) $2^{\lg(n)}$ er $\Omega(n^{0.01})$
12) $(\lg n)^3 + 3^n$ er $\Omega(2^n)$

2.1.1 svar til opgave 1 AU

2.2 Opgave 2 fra AU formulering Løkke-invarianter

Algoritme loop1(n)
  $s = 1$
  for $i = 1$ **to** $n * n$
    for $j = 1$ **to** n
      $s = s + 1$

Algoritme loop2(n)
  $i = n$
  while $i \leq n * n$
    $i = 2 * i$

Algoritme loop3(n)
  $i = 1$
  $j = n * n$
  while $i \leq j$
    $i = 2 * i$
    $j = j - 1$

Algoritme loop4(n)
  $i = 1$
  while $i \leq n$
    $j = i$
    while $j > 0$
      $j = \lfloor j/2 \rfloor$
    $i = i + i$

Angiv for hver af ovenstående algoritmer udførselstiden som funktion af n i Θ.

	$\Theta(\sqrt{n})$	$\Theta(n^3)$	$\Theta(n)$	$\Theta(\sqrt[3]{n})$	$\Theta(n^2)$	$\Theta(\log n)$	$\Theta((\log n)^2)$	$\Theta(n \log n)$
loop1	☐	☐	☐	☐	☐	☐	☐	☐
loop2	☐	☐	☐	☐	☐	☐	☐	☐
loop3	☐	☐	☐	☐	☐	☐	☐	☐
loop4	☐	☐	☐	☐	☐	☐	☐	☐

2.2.1 svar til opgave 2 AU

2.3 Opgave 3-1 CLRS *Asymptotic behavior of polynomials*

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 $p(n) = \mathcal{O}(n^k)$

b) If $k \leq d$, then $p(n) = \Omega(n^k)$.

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

d) If $k > d$, then $p(n) = \boldsymbol{o}(n^k)$.

e) If $k < d$, then $p(n) = \omega(n^k)$.

2.3.1 Svar til opgave 3-1 CLRS

2.4 Opgave 3-2 CLRS *Relative Asymptotic Growth* (4th. ed.)

Indicate, for each pair of expressions (A, B) in the table below, whether A is $\mathcal{O}$, $\boldsymbol{o}$, Ω, ω, or Θ of B. Assume that $k \geq 1$, $\varepsilon > 0$, and $c > 1$ are constants. Your answer should be in the form of the table with "yes" or "no" written in each box.

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					
f.	$\lg(n!)$	$\lg(n^n)$					

Reasons:

a) Any polylogarithmic function is little-oh of any polynomial function with a positive exponent.

b) Any polynomial function is little-oh of any exponential function with a positive base.

c) The function $\sin(n)$ oscillates between 1 and 1. There is no value n_0 such that $\sin(n)$ is less than, greater than, or equal to $1/2$ for all $n \geq n_0$, and so there is no value n_0 such that $n^{\sin(n)}$ is less than, greater than, or equal to $cn^{1/2}$ for all $n \geq n_0$.

d) Take the limit of the quotient: $\lim_{n\to\infty} 2^n/2^{n/2} = \lim_{n\to\infty} 2^{n/2} = \infty$.

e) By equation (3.21), these quantities are equal.

Equation (3.21) for Logarithms states: For any constant $b > 1$, the function $\lg_b(n)$ is undefined if $n \leq 0$, strictly increasing if $n > 0$, negative if $0 < n < 1$, positive if $n > 1$, and 0 if $n = 1$. For all real $a > 0$, $b > 0$, $c > 0$, and n, we have

$$a^{\lg_b(c)} = c^{\lg_b(a)}$$

where, in the equation above, logarithm bases is not 1 (CLRS 4th. ed. Chapter. 3.3).

f) By equation (3.28), $\lg(n!) = \Theta(n \lg(n)$. Since $\lg(n^n) = n \lg(n)$, these functions are Θ, of each other.

Equation (2.28) for Factorials states: $\lg(n!) = \Theta(n \lg(n))$ (CLRS 4th. ed. Chapter. 3.3).

2.4.1 Svar til opgave CLRS 3-2

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

2.5 Opgave 3-4 CLRS *Asymptotic Notation Properties*

Let $f(n)$ and $g(n)$ be asymptotically positive functions. Prove or disprove each of the following conjectures:

a) $f(n) = \mathcal{O}\left(g(n)\right)$ implies $g(n) = \mathcal{O}\left(f(n)\right)$.

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

c) $f(n) = \mathcal{O}\left(g(n)\right)$ implies $\lg(f(n)) = \mathcal{O}\left(\lg(g(n))\right)$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficient large n.

d) $f(n) = \mathcal{O}(g(n))$ implies $2^{f(n)} = \mathcal{O}\left(2^{g(n)}\right)$.

e) $f(n) = \mathcal{O}\left((f(n))^2\right)$.

f) $f(n) = \mathcal{O}\left(g(n)\right)$ implies $g(n) = \Omega\left(f(n)\right)$.

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

h) $f(n) + \boldsymbol{o}\left(f(n)\right) = \Theta\left(f(n)\right)$

2.5.1 Svar til opgave 3-4 CLRS (3rd. ed.)

(a) The conjecture is <u>false</u>. For example, let $f(n) = n$ and $g(n) = n^2$. Then $f(n) = \mathcal{O}\left(g(n)\right)$, but $g(n)$ is not $\mathcal{O}\left(f(n)\right)$

(b) The conjecture is false. Again, let $f(n) = n$ and $g(n) = n^2$. Then the conjecture would be saying that $n + n^2 = \Theta\left(n\right)$, which is false.

(c) The conjecture is true. Since $f(n) = \mathcal{O}\left(g(n)\right)$ and $f(n) \geq 1$ for sufficiently large n, there are some positive constants c and n_0 such that $1 \leq f(n) \leq cg(n)$ for all n for all $n \geq n_0$, which implies $0 \leq \lg(f(n)) \leq \lg(c) = \lg(g(n))$. Without loss of generality, assume that $c > 1/2$, so that $\lg(c) > -1$. Define the constant $d = 1 + \lg(c) > 0$. Then, we have

$$\begin{aligned}
\lg(f(n)) &\leq \lg(c) + \lg(g(n)) \\
&= \left(1 + \frac{\lg(c)}{\lg(g(n))}\right)\lg(g(n)) \\
&\leq (1 + \lg(c))\lg(g(n)) && \text{(because } \lg(g(n)) \geq 1\text{)} \\
&= d\lg(g(n))
\end{aligned}$$

and so there exist positive constants d and n_0 such that $0 \leq \lg(f(n)) \leq d\lg(g(n))$ for all $n \geq n_0$. Thus, $\lg(f(n)) = \mathcal{O}\left(\lg(g(n))\right)$

(d) The conjecture is false. For example, let $f(n) = 2n$ and $g(n) = n$. Then $f(n) = \mathcal{O}(g(n))$, but $2^{f(n)} = 2^{2n}$ and $2^{g(n)} = 2^n$, so that $2^{f(n)}$ is not $\mathcal{O}\left(2^{g(n)}\right)$.

(e) The conjecture is false. For example, let $f(n) = 1/n$, so that $f(n)^2 = 1/n^2$. It is not the case that $1/n = \mathcal{O}\left(1/n^2\right)$.

(f) The conjecture is true, by transpose symmetry on page 62. That is, since $\mathcal{O}$, by definiton, is an assymptotic upper-bound and Ω, by definition, is an assymptotic lower-bound, then it is trivial that .

(g) The conjecture is false. Let $f(n) = 2^n$. It is not the case that 2^n is $\Theta\left(2^{n/2}\right)$.

(h) The conjecture is true. Let $g(n)$ be any function in $o(f(n))$. Then there exists a constant $n_0 > 0$ such that for any positive constant $c > 0$ and all $n \geq n_0$, we have $0 \leq g(n) < cf(n)$. Since $f(n) + g(n) \geq f(n)$, we have $f(n) + g(n) = \Omega(f(n))$. For the upper bound, choose the n_0 used for $g(n)$ and choose any constant $c > 0$. Then, we have

$$\begin{aligned} 0 &\leq f(n) + g(n) \\ &< f(n) + cf(n) \\ &= (1+c)f(n) \\ &\leq c' f(n) \end{aligned}$$

for cosntant $c' = 1 + c$. Therefore, $f(n) + g(n) = \mathcal{O}(f(n))$, so that $f(n) + g(n) = \Theta(f(n))$

2.6 Opgave 4.3-1 CLRS *The substition method for solving reccurences*

Use the substitution method to show that each of the following recurrences defined on the reals has the asymptotic solution specified:

a) $T(n) = T(n-1) + n$ has solution $T(n) = \mathcal{O}\left(n^2\right)$.

b) $T(n) = T(n/2) + \Theta(1)$ has solution $T(n) = \mathcal{O}(\lg(n))$.

c) $T(n) = 2T(n/2) + n$ has solution $T(n) = \Theta(n \lg(n))$.

d) $T(n) = 2T(n/2 + 17) + n$ has solution $T(n) = \mathcal{O}(n \lg(n))$.

e) $T(n) = 2T(n/3) + \Theta(n)$ has solution $T(n) = \Theta(n)$.

f) $T(n) = 4T(n/2) + \Theta(n)$ has solution $T(n) = \Theta\left(n^2\right)$

2.6.1 Svar til opgave 4.3-1 CLRS

(a) We guess that $T(n) \le cn^2$ for some constant $c > 0$. We have

$$\begin{aligned} T(n) &= T(n-1) + n \\ &\le c(n-1)^2 + n \\ &= cn^2 - 2cn + c + n \\ &= cn^2 + c(1-2n) + n \end{aligned}$$

This last quantity is less than or equal to cn^2 if $c(1-2n) + n \le 0$ or, equivalently, $c \ge n/(2n-1)$. This last condition holds for all $n \ge 1$ and $c \ge 1$.

For the boundary condition, we set $T(1) = 1$, and so $T(1) = 1 \le c \cdot 1^2$. Thus, we can choose $n_0 = 1$ and $c = 1$.

(b) We guess that $T(n) = c\ \lg(n)$, where c is the constant in the $\Theta(1)$ term. We have

$$\begin{aligned} T(n) &= T(n/2) + c \\ &= c\ \lg(n/2) + c \\ &= c\ \lg(n) - c + c \\ &= c\ \lg(n) \end{aligned}$$

For the boundary condition, choose $T(2) = c$.

(c) We guess that $T(n) = n\ \lg(n)$. We have

$$\begin{aligned} T(n) &= 2T(n/2) + n \\ &= 2((n/2)\lg(n/2)) + n \\ &= n\ \lg(n/2) + n \\ &= n\ \lg(n) - n + n \\ &= n\ \lg(n) \end{aligned}$$

For the boundary condition, choose $T(2) = 2$.

(d) We will show that $T(n) \le cn\ \lg(n)$ for $c = 20$ and $n \ge 917$. (Different combinations of c and n_0 work. We just happen to choose this combination.) First, observe that $n/2 + 17 \le 3n/4 < n$ for all $n \ge 68$. We have

$$
\begin{aligned}
T(n) &= 2T(n/2+17)+n \\
&= 2(c(n/2+17)\lg(n/2+17))+n && \text{(substitute)} \\
&= cn\ \lg(n/2+17)+34c\lg(n/2+17)+n \\
& && \text{(reducing paranthesis)} \\
&< cn\ \lg(3n/4)+34c\lg(n)+n && \text{(because } n \geq 68) \\
&= cn\ \lg(n)-cn\ \lg(4/3)+34c\ \lg(n)+n \\
&= cn\ \lg(n)+(34c\lg(n)-n(c\ \lg(4/3)-1)) \\
& cn\ \lg(n)
\end{aligned}
$$

if $34c\ \lg(n) \leq n(c\ \lg(4/3)-1)$. If we choose $c = 20$, then this inequality holds for all $n \geq 917$. (Notice that for there to be an n_0 such that the inequality holds for all $n \geq n_0$, we must choose c such that $c\ \lg(4/3) - 1 > 0$, or $c > 1/\lg(4/3) \approx 3.476$.)

(e) Let c be the constant in the $\Theta(n)$ term. We need to show only the upper bound of $\mathcal{O}(n)$, since the lower bound of $\Omega(n)$ follows immediately from the $\Theta(n)$ term in the recurrence. We guess that $T(n) \leq dn$, where d is a constant that we will choose. We have

$$
\begin{aligned}
T(n) &= 2T(n/3)+cn \\
&\leq 2dn/3+cn \\
&= n(2d/3+c) \\
&\leq dn
\end{aligned}
$$

if $2d/3+c \leq d$ or, equivalently, $d \geq 3c$.

(f) Let c be the constant in the $\Omega()n$ term. We guess that $T(n) = dn^2 - d'n$ for constants d and d' that we will choose. We will show the upper ($\mathcal{O}$) and lower (Ω) bounds separately.

For the upper bound, we have

$$
\begin{aligned}
T(n) &\leq 4T(n/2)+cn \\
&= 4(d(n/2)^2-d'n/2)+cn \\
&= dn^2-2d'n+cn \\
&= dn^2-d'n
\end{aligned}
$$

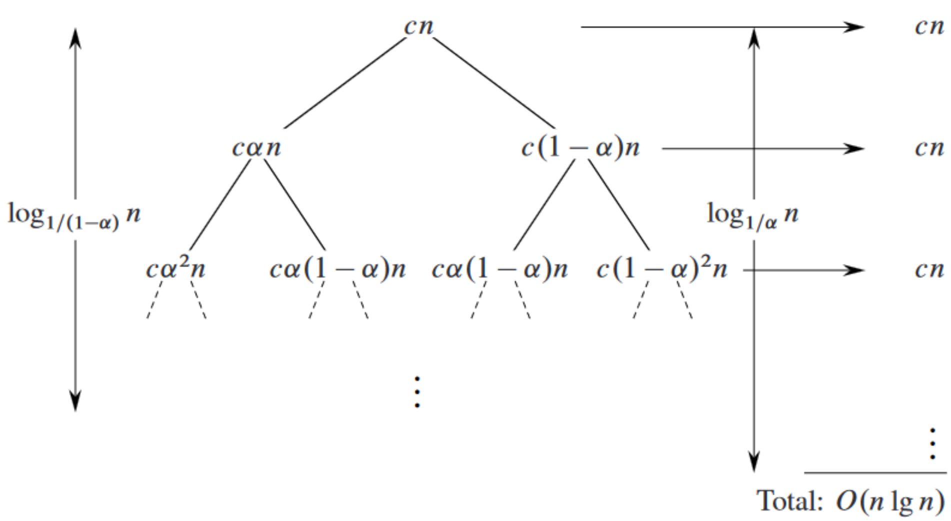
if $-2d'n + cn \leq -d'n$ or, equivalently, $d'n \geq c$. For the lower bound, we just need $d' \leq c$. Thus, setting $d' = c$ works for both the upper and lower bounds.

2.7 Opgave 4.4-4 CLRS *The substition method for solving reccurences*

Use a recursion tree to justify a good guess for the solution to the recurrence $T(n) = T(\alpha n) + T((\alpha - 1)n) + \Theta(n)$, where α is a constant in the range $0 < \alpha < 1$.

2.7.1 Svar til opgave 4.4-4 CLRS

$T(n) = T(\alpha n) + T((\alpha - 1)n) + cn$
We saw the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$ in the text. This recurrence can be similarly solved. Without loss of generality, let $\alpha \geq 1 - \alpha$, so that $0 < 1 \leq 1/2$ and $1/2 \leq \alpha < 1$.



The recursion tree is full for $\lg_{1/(1-\alpha)}(n)$ levels, each contributing cn, so we guess $\Omega(n \lg_{1/(1-\alpha)}(n)) = \Omega(n \lg(n))$. It has $\lg_{1/\alpha)}(n)$ levels, each contributing $\leq cn$, so we guess $\mathcal{O}(n \lg_{1/\alpha}(n)) = \mathcal{O}(n \lg(n))$.
Now we show that $T(n) = \Theta(n \lg(n))$ by substitution. To prove the upper

bound, we need to show that $T(n) \leq dn\lg(n)$ for suitable constant $d > 0$:

$$\begin{aligned}
T(n) &= T(\alpha n) + T((\alpha - 1)n) + cn \\
&\leq d\alpha n \lg(\alpha n) + d\alpha n \lg(n) + d(1-\alpha)n\lg(1-\alpha) + d(1-\alpha)n\lg(n) + cn \\
&= dn\lg(n) + dn(\alpha\lg(\alpha) + (1-\alpha)\lg(1-\alpha)) + cn \\
&\leq dn\lg(n),
\end{aligned}$$

if $dn(\alpha\lg(\alpha) + (1-\alpha)\lg(1-\alpha)) + cn \leq 0$. This condition is equivalent to $dn(\alpha\lg(\alpha) + (1-\alpha)\lg(1-\alpha)) \leq -c$.

Since $1/2 \leq \alpha < 1$ and $0 < 1-\alpha \leq 1/2$, we have that $\lg(\alpha) < 0$ and $\lg(1-\alpha) < 0$. Thus, $\alpha\lg(\alpha) + (1-\alpha)\lg(1-\alpha) < 0$, so that when we multiply both sides of the inequality by this factor, we need to reserve the inequality:

$$d \geq \frac{-c}{\alpha\lg(\alpha) + (1-\alpha)\lg(1-\alpha)} \qquad \text{or} \qquad d \geq \frac{c}{-\alpha\lg(\alpha) - (1-\alpha)\lg(1-\alpha)}$$

The fraction on the right-hand side is a positive constant, and so it suffices to pick any value of d that is greater than or equal to this fraction.

To prove the lower bound, we need to show that $T(n) \geq dn\lg(n)$ for a suitable constant $d > 0$. We can use the same proof as for the upper bound, substituting $\geq$ for $\leq$, and we get the requirement that

$$0 < d \geq \frac{c}{-\alpha\lg(\alpha) - (1-\alpha)\lg(1-\alpha)}.$$

Therefore, $T(n) = \Theta(n\lg(n))$.

2.8 Opgave 4.5-1 CLRS *The master method for solving reccurences*

Use the master method to give tight asymptotic bounds for the following recurrences.

a) $T(n) = 2T(n/4) + 1$.

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

c) $T(n) = 2T(n/4) + \sqrt{n}\lg^2(n)$.

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

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

2.8.1 Svar 4.5-1 CLRS

In all parts of this problem, we have $a = 2$ and $b = 4$, and thus $n^{\lg_b(a)} = n^{\lg_4(2)} = n^{1/2} = \sqrt{n}$

(a) $T(n) = \Theta(\sqrt{n})$. Here, $f(n) = \mathcal{O}(n^{1/2-\varepsilon})$ for $\varepsilon = 1/2$. Case 1 applies, and $T(n) = \Theta(n^{1/2}) = \Theta(\sqrt{n})$.

(b) $T(n) = \Theta(\sqrt{n}\lg(n))$. Now $f(n) = \sqrt{n} = \Theta(n^{\lg_b(a)})$. Case 2 applies, with $k = 0$

(c) $T(n) = \Theta(\sqrt{n}\lg^2(n))$. Now $f(n) = \sqrt{n}\lg^2(n) = \Theta(n^{\lg_b(a)}\lg^2(n))$. Case 2 applies, with $k = 2$

(d) $T(n) = \Theta(n)$. This time, $f(n) = n^1$, and so $f(n) = \Omega(n^{\lg_b(a)+\varepsilon})$ for $\varepsilon = 1/2$. In order for case 3 to apply, we have to check the regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$. Here $af(n/b) = n/2$, and so the regularity condition holds for $c = 1/2$. Therefore, case 3 applies.

(e) $T(n) = \Theta(n^2)$. Now, $f(n) = n^2$, and so $f(n) = \Omega(n^{\lg_b(a)+\varepsilon})$ for $\varepsilon = 3/2$. In order for case 3 to apply, we again have to check the regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$. Here, $af(n/b) = n^2/8$, and so the regularity condition holds for $c = 1/8$. Therefore, case 3 applies.

2.9 Opgave 16 formulering AU

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$
$T(n) = 4 \cdot T(n/2) + n^2$	☐	☐	☐	☐	☐	☐	☐
$T(n) = 2 \cdot T(n/5) + n$	☐	☐	☐	☐	☐	☐	☐
$T(n) = T(n-1) + \log n$	☐	☐	☐	☐	☐	☐	☐
$T(n) = T(n/4) + 5$	☐	☐	☐	☐	☐	☐	☐
$T(n) = 4 \cdot T(n/2) + 1$	☐	☐	☐	☐	☐	☐	☐

2.9.1 Svar til opgave 16 AU

	$\Theta(\log n)$	$\Theta(\sqrt{n})$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n^2 \log n)$	$\Theta(n^3)$
$T(n) = 4 \cdot T(n/2) + n^2$	☐	☐	☐	☐	☒	☐	☐
$T(n) = 2 \cdot T(n/5) + n$	☐	☐	☒	☐	☐	☐	☐
$T(n) = T(n-1) + \log n$	☒	☐	☐	☐	☐	☐	☐
$T(n) = T(n/4) + 5$	☐	☒	☐	☐	☐	☐	☐
$T(n) = 4 \cdot T(n/2) + 1$	☐	☐	☐	☐	☐	☐	☒

2.10 Opgaveopsamling task 1-6 ("gamle"aflevering 1)

2.10.1 Task 1

For the following pricing schemes, use the master method

1) $p(n) = 8p(n/2) + n^2$.

2) $p(n) = 8p(n/4) + n^3$

3) $p(n) = 10p(n/9) + n \lg_2(n)$

Svar til task 1

(1)

(2)

(3)

2.10.2 Task 2

For the following pricing schemes, use the substitution method. You may ignore the induction start and only show the induction step. When showing the induction step, you may assume that $n/2$ and $n/3$ are integers 1 in the first part and that $\sqrt{n}$ is an integer in the second part. Be careful to avoid the pitfalls of the substitution method (see CLRS section 4.3).

Additionally, you have to draw a recursion tree down to at least four levels for both of the recurrences. Your guess for the substitution method needs to be derived from the recursion tree.

1) $p(n) = p(n/2) + p(n/3) + n$

2) $p(n) = \sqrt{n}p(\sqrt{n} + \sqrt{n})$ Hint: This one might be tricky. Take a close look at the section on subtracting lower order terms in CLRS.

Svar til task 2

(1)

(2)

2.10.3 Task 3

Write pseudo-code for introsort. You may use functions from the book such as `HeapSort`, `InsertionSort`, and `Randomized-Partition`. You may find inspiration in the pseudocode for `quicksort` from CLRS.

2.10.4 Task 4

Show that the running time of introsort is worst-case $\mathcal{O}(n \lg(n))$. You may use results from the course book without proving them.

2.10.5 Task 5

Discuss why we use heap sort rather than another $\mathcal{O}(n \lg(n))$ sorting algorithm such as merge sort. (Hint: What are the properties of the different sorting functions?)

2.10.6 Task 6

In the description above, we use insertion sort to sort the small arrays of size $j - i < c$. An alternative is to simply return the recursive call without sorting this small array and instead call insertion sort with the entire (nearly sorted) array as input. Why is it a good idea to run insertion sort on the nearly sorted data, when we know from CLRS that its worst-case running time is $\Theta(n^2)$?

Kapitel 3

Uge 3: Træer, fibonnacihobe og dynamisk programmering

******Mulighed for Teori******

3.1 Opgave 16.1-1 CLRS

If the set of stack operations includes a `MULTIPUSH` operation, which pushes k items onto the stack, does the $\mathcal{O}(1)$ bound on the amortized cost of stack operations continue to hold?

3.1.1 Svar til 16.1-1

With a `MULTIPUSH` operation, the amortized cost of stack operations would no longer be $\mathcal{O}(1)$ The cost of a single MULTIPUSH that pushes k items onto the stack is $\Theta(k)$

3.2 Opgave 16.1-3 CLRS

Use aggregate analysis to determine the amortized cost per operation for a sequence of n operations on a data structure in which the ith operation costs i if i is an exact power of 2, and 1 otherwise.

3.2.1 Svar til opgave 16.1-3 CLRS

Let c_i = cost of ith operation.

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise.} \end{cases}$$

Operation	Cost
1	1
2	2
3	1
4	4
5	1
6	1
7	1
8	8
9	1
10	1
$\vdots$	$\vdots$

n operations cost

$$\sum_{i=1}^{n} c_i \leq n + \sum_{i=1}^{\lg(n)} 2^j = n + (2n-1) < 3n.$$

(Note: Ignoring floor in upper bound of $\sum 2^j$.)

$$\text{Average cost of operation} \ = \ \frac{\text{Total cost}}{\#\text{ operations}} < 3.$$

By aggregate analysis, the amortized cost per operation = $\mathcal{O}(1)$.

3.3 Opgave 16.2-2 CLRS

Redo Exercise 16.1-3 using an accounting method of analysis.

3.3.1 Svar til opgave 16.2-2 CLRS

Let c_i = cost of ith operation.

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise.} \end{cases}$$

Charge each operation \$3 (amortized cost $\hat{c}_i$)

- If i is not an exact power of 2, pay \$1, and store \$2 as credit.
- If i is an exact power of 2, pay \$i, using stored credit.

Operation	Amortized cost	Actual cost	Credit remaining
1	3	1	2
2	3	2	3
3	3	1	5
4	3	4	4
5	3	1	6
6	3	1	8
7	3	1	10
8	3	8	5
9	3	1	7
10	3	1	9
⋮	⋮	⋮	⋮

Since the amortized cost is \$3 per operation, $\sum_{i=1}^{n} \hat{c}_i < 3n$.

We know from Exercise 16.1-3 that $\sum_{i=1}^{n} c_i < 3n$.

Then we have $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i \Rightarrow$ credit $=$ amortized cost $-$ actual cost ≥ 0.

Since the amortized cost of each operation is $\mathcal{O}(1)$ and the amount of credit never goes negative, the total cost of n operations is $\mathcal{O}(n)$.

3.4 Opgave 16.3-2 CLRS

Redo Exercise 16.1-3 using a potential method of analysis

3.4.1 Svar til opgave 16.3-2 CLRS

Define the potential of D_i by

$$\Phi(D_i) = \begin{cases} 0 & \text{if } i = 0\,, \\ 2i - 2^{\lfloor \lg i \rfloor + 1} & \text{if } i \geq 1\,. \end{cases}$$

Since $2^{\lfloor \lg i \rfloor} \leq i$ for $i \geq 1$, the value of $\Phi(D_i)$ is nonnegative for all i.
If i is not a power of 2, then the amortized cost of the ith operation is

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
&= 1 + (2i - 2^{\lfloor \lg i \rfloor + 1}) - (2(i-1) - 2^{\lfloor \lg(i-1) \rfloor + 1}) \\
&= 1 + (2i - 2^{\lfloor \lg i \rfloor + 1}) - (2(i-1) - 2^{\lfloor \lg i \rfloor + 1}) \\
&= 3\,.
\end{aligned}$$

If $i = 2^k$ for some nonnegative integer k, then the amortized cost of the ith operation is

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}) \\
&= i + (2i - 2^{\lfloor \lg i \rfloor + 1}) - (2(i-1) - 2^{\lfloor \lg(i-1) \rfloor + 1}) \\
&= 2^k + (2 \cdot 2^k - 2^{\lfloor \lg 2^k \rfloor + 1}) - (2(2^k - 1) - 2^{\lfloor \lg(2^k - 1) \rfloor + 1}) \\
&= 2^k + (2^{k+1} - 2^{k+1}) - (2^{k+1} - 2 - 2^{(k-1)+1}) \\
&= 2^k - (2^k - 2) \\
&= 2\,.
\end{aligned}$$

3.5 Opgave 16.3-3 CLRS

Consider an ordinary binary min-heap data structure supporting the instructions `INSERT` and `EXTRACT-MIN` that, when there are n items in the heap, implements each operation in $\mathcal{O}(\lg(n))$ worst-case time. Give a potential function Φ such that the amortized cost of `INSERT` is $\mathcal{O}(\lg(n))$ and the amortized cost of `EXTRACT-MIN` is $\mathcal{O}(1)$, and show that your potential function yields these amortized time bounds. Note that in the analysis, n is the number of items currently in the heap, and you do not know a bound on the maximum number of items that can ever be stored in the heap.

3.5.1 Svar til opgave 16.3-3 CLRS

Let D_i be the heap after the ith operation, and let D_i consist of n_i elements. Also, let k be a constant such that each `INSERT` or `EXTRACT-MIN` operation takes at most $k \ln(n)$ time, where $n = \max(n_{i-1}, n_i)$. (We don't want to worry about taking the log of 0, and at least one of n_{i-1} and n_i is at least 1. We'll see later why we use the natural log.)

Define

$$\Phi(D_i) = \begin{cases} 0 & \text{if } n_i = 0 \\ kn_i \ln(n_i) & \text{if } n_i > 0 \end{cases}$$

This function exhibits the characteristics we like in a potential function: if we start with an empty heap, then $\Phi(D_0) = 0$, and we always maintain that $\Phi(D_i) \geq 0$. Before proving that we achieve the desired amortized times, we show that if $n \geq 2$, then $n \ln\left(\frac{n}{n-1}\right) \leq 2$. We have

$$\begin{aligned} n \ln\left(\frac{n}{n-1}\right) &= n \ln\left(1 + fracnn - 1\right) \\ &= \ln\left(1 + fracnn - 1\right)^2 \\ &\leq \ln\left(e^{\frac{1}{n-1}}\right) && \text{(since } 1 + x \leq e^x \text{ for all real } x\text{)} \\ &= \ln\left(e^{\frac{n}{n-1}}\right) \\ &= \frac{n}{n-1} \\ &\leq 2, \end{aligned}$$

assuming that $n \geq 2$. (The equation $\ln\left(e^{\frac{n}{n-1}}\right) = \frac{n}{n-1}$ is why we use the natural log.)
If the ith operation is an `INSERT`, then $n_i = n_{i-1} + 1$. If the ith operation inserts into an empty heap, then n_{i-1}, $n_{i1D}0$, and the amortized cost is

$$\begin{aligned} \hat{c}_i =& c_i + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) \\ \leq& k \lg(1) + k \cdot 1 \lg(1) - 1 \\ =& 0 \end{aligned}$$

If the ith operation inserts into a nonempty heap, then $n_i = n_{i-1} \geq 2$, and the amortized cost is

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) \\
&\leq k\lg(n_i) + kn_i\lg(n_i) - kn_{i-1}\lg(n_{i_1}) \\
&= k\lg(n_i) + kn_i\lg(n_i) - k\lg(n_i - 1)\lg(n_i - 1) \\
&= k\lg(n_i) + kn_i\lg(n_i) - kn_i\lg(n_i - 1) + k\lg(n_i - 1) \\
&< 2k\ln(n_i) + kn_i\lg\left(\frac{n_i}{n_i - 1}\right) \\
&\leq 2k\lg(n_i) + 2k && \text{(since } n_i \geq 2\text{)} \\
&= \mathcal{O}(\lg(n_i)).
\end{aligned}$$

If the ith operation is an `EXTRACT-MIN`, then $n_i = n_{i-1} - 1$. If the ith operation extracts the one and only heap item, then $n_i = 0$, $n_{i-1} = 1$, and the amortized cost is

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) \\
&\leq k\lg(1) + 0 - k \cdot 1\lg(1) \\
&= 0.
\end{aligned}$$

If the ith operation extracts from a heap with more than one item, then $n_i = n_{i-1} - 1$ and $n_{i-1} \geq 2$, and the amortized cost is

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi\left(D_i\right) - \Phi\left(D_{i-1}\right) \\
&\leq k\lg(n_i) + kn_i\lg(n_i) - kn_{i-1}\lg(n_{i_1}) \\
&= k\lg(n_{i-1}) + k(n_{i-1} - 1)\lg(n_{i-1} - 1) - kn_{i-1}\lg(n_{i-1}) \\
&= k\lg(n_{i-1}) + k(n_{i-1} - 1)\lg(n_{i-1}) - k\lg(n_{i-1}) - kn_{i-1}\lg(n_{i-1}) \\
&= k\lg\left(\frac{n_{i-1}}{n_{i-1} - 1}\right) + kn_{i-1}\lg\left(\frac{n_{i-1} - 1}{n_{i-1}}\right) \\
&< k\lg\left(\frac{n_{i-1}}{n_{i-1} - 1}\right) + kn_{i-1}\lg(1) \\
&= k\lg\left(\frac{n_{i-1}}{n_{i-1} - 1}\right) \\
&\leq k\lg(2) && \text{(since } n_{i-1} \geq 2\text{)} \\
&= \mathcal{O}(\lg(1)).
\end{aligned}$$

A slightly different potential function—which may be easier to work with–is as follows. For each node x in the heap, let $d_i(x)$ be the depth of x in D_i.

Define

$$\Phi(D_i) = \sum_{x \in D_i} k(d_i(x) + 1)$$
$$= k\left(n_i + \sum_{x \in D_i} d_i(x)\right),$$

where k is defined as before.

Initially, the heap has no items, which means that the sum is over an empty set, and so $\Phi(D_0) = 0$. We always have $\Phi(D_i) \geq 0$, as required.

Observe that after an INSERT, the sum changes only by an amount equal to the depth of the new last node of the heap, which is $\lfloor \lg(n_i) \rfloor$. Thus, the change in potential due to an `INSERT` is $k(1 + \lfloor \lg(n_i) \rfloor)$, and so the amortized cost is $\mathcal{O}(\lg(n_i) + \mathcal{O}(\lg(n_i)) = \mathcal{O}(\lg(n_i)) = \lg(\backslash)$.

After an `EXTRACT-MIN`, the sum changes by the negative of the depth of the old last node in the heap, and so the potential decreases by $k(1 + \lfloor \lg(n_i) \rfloor)$. The amortized cost is at most $k \lg(n_{i-1}) - k(1 + \lfloor \lg(n_{i-1}) \rfloor) = \mathcal{O}(1)$.

3.6 Opgave 16.3-4 CLRS

What is the total cost of executing n of the stack operations `PUSH`, `POP`, and `MULTIPOP`, assuming that the stack begins with s_0 objects and finishes with s_n objects?

3.6.1 Svar til opgave 16.3-4 CLRS

Starting with

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1}),$$

subtracting $\Phi(D_i) - \Phi(D_{i-1})$ from both sides gives

$$\begin{aligned}
\sum_{i=1}^{n} c_i &= \sum_{i=1}^{n} (\hat{c}_i + \Phi(D_{i-1}) - \Phi(D_i)) \\
&= \sum_{i=0}^{n} \hat{c}_i + \Phi(0) - \Phi(D_n) && \text{(telescoping sum)} \\
&= \sum_{i=0}^{n} \hat{c}_i + s_0 - s_n && (\Phi(D_i) \text{ equals number of objects in the stack}) \\
&\leq 2n + s_0 - s_n && (\hat{c}_i \leq 2)
\end{aligned}$$

3.7 Opgave 16.4-4 CLRS

Suppose that instead of contracting a table by halving its size when its load factor drops below 1/4, you contract the table by multiplying its size by 2/3 when its load factor drops below 1/3. Using the potential function

$$\Phi(T) = |2(T.num - T.size/2)|$$

show that the amortized cost of a `TABLE-DELETE` that uses this strategy is bounded above by a constant.

3.8 Opgave 19-1 CLRS

3.9 Opgave 19.2-1 CLRS

3.10 Opgave 19-3(a) CLRS

3.11 OPgave 19.4-1 CLRS

3.12 Opgave 2 spørgsmål 1 og 2. fra opgavesamling (gamle afleveringer)

Kapitel 4

Uge 4: Dynamisk Programmering

4.1 Opgave 21 og 22 fra AU

4.2 Opgave 14.1-2 CLRS

4.3 Opgave 14.1-3 CLRS

4.4 Opgave 14.3-5 CLRS

4.5 Opgave 14.4-1 CLRS

4.6 Opgave 14-2 CLRS

4.7 Opgave 14.4-5 CLRS

4.8 Opgave 14.4-6 CLRS

Kapitel 5

Uge 5: Greedy Algoritmer, Binær søge træ'er og Rød-sorte træ'er

5.1 Opgave 15.2-1 CLRS

5.2 Opgave 15.2-5 CLRS

5.3 Opgave 15-1(a) CLRS (penny = 1, nickel = 5, dime = 10, quarter = 25)

5.4 Opgave 15-1(c) CLRS

5.5 Opgave 15-1(b) CLRS

5.6 Opgave 15.3-3 CLRS

5.7 Opgave 15.1-4 CLRS

5.8 Opgave 7, 8 og 9 fra AU

5.9 Opgave 12.1-5 CLRS

5.10 Opgave 12.2-3 CLRS

5.11 Opgave 12.3-5 CLRS

5.12 Opgave 13.1-2 CLRS

5.13 Opgave 13.2-3 CLRS

5.14 Opgave 13.3-1 CLRS

Kapitel 6

Uge 6: Disjunkte mængder og Minimum Spanning Tree

6.1 Opgave 11 fra AU

6.2 Opgave CLRS 19.2-2 CLRS

6.3 Opgave 19.2-4 CLRS

6.4 Opgave 19.3-1 CLRS

6.5 Opgave 19.3-3 CLRS

6.6 Opgave 19-1 CLRS

6.7 Opgave 33.1-3 CLRS (digital chap.)

6.8 Opgave 33.1-4 CLRS (digital chap.)

6.9 Opgave 33.2-3 CLRS (digital chap.)

6.10 Opgave 33.2-4 CLRS (digital chap.)

6.11 Opgave 33.2-5 CLRS (digital chap.)

6.12 Opgave 15 fra 15 AU

6.13 Opgave 21.1-1 CLRS

6.14 Opgave 21.1-3 CLRS

6.15 Opgave 21.1-5 CLRS

6.16 Opgave 21.2-1 CLRS

Kapitel 7

Uge 6: Convex Hulls and Closets pair

7.1 Opgave 33.3-4 CLRS

7.2 Opgave 33.3-5 CLRS

7.3 Opgave 33.4-1 CLRS

7.4 Opgave 33-1 CLRS