Solutions for Exercise Sheet 2

Handout: September 19th — Deadline: September 26th, 4pm

Question 2.1 (0.5 marks)

Express the following running times in Θ-notation. Justify your answer by referring to the definition of Θ (i. e. work out suitable c_1, c_2, n_0).

a) $3n^2 + 5n - 2 = \Theta(n^2)$.
To see this, we need to show that $c_1 n^2 \le 3n^2 + 5n - 2 \le c_2 n^2$ for all $n \ge n_0$ and suitable positive constants c_1, c_2 and a suitable n_0. Divide by n^2 and we get $c_1 \le 3 + \frac{5}{n} - \frac{2}{n^2} \le c_2$ which is true, for example, for $c_1 := 3, c_2 := 4$, and $n_0 := 5$.

b) $42 = \Theta(1)$.
To see this, we need to show that $c_1 \cdot 1 \le 42 \le c_2 \cdot 1$ for all $n \ge n_0$ and suitable positive constants c_1, c_2 and a suitable n_0. We can choose, for example, $c_1 := 42, c_2 := 42$, and $n_0 := 1$.

c) $4n^2 \cdot (1 + \log n) - 2n^2 = \Theta(n^2 \log n)$.
To see this, we need to show that $c_1 n^2 \log n \le 4n^2 \cdot (1 + \log n) - 2n^2 \le c_2 n^2 \log n$ for all $n \ge n_0$ and suitable positive constants c_1, c_2 and a suitable n_0. Divide by $n^2 \log n$ and we get $c_1 \le 4 \cdot \left(\frac{1}{\log n} + 1\right) - \frac{2}{\log n} \le c_2$, which we can rewrite as $c_1 \le 4 + \frac{2}{\log n} \le c_2$. This is true, for example, for $n_0 := 2$ (thus $\log n \ge 1$ for $n \ge n_0$) and $c_1 := 2$, $c_2 := 8$.

Question 2.2 (0.5 marks)

(a) Indicate for each pair of functions $f(n), g(n)$ in the following table whether $f(n)$ is O, o, Ω, ω, or Θ of $g(n)$ by writing "yes" or "no" in each box.

Solutions: **(a)** To fill in the table, it helps to remember that Θ ("equal asymptotic growth") holds if and only if both O ("grows at most as fast as") and Ω ("grows at least as fast as") hold. Also o ("grows slower than") implies O ("grows at most as fast as") and ω ("grows faster than") implies Ω ("grows at least as fast as"). So the answer in all cases below is either "o and O" or "ω and Ω" or "Θ and O and Ω".

The hint explains that $\log n$ grows slower than $\sqrt{n}$, hence $\log n = o(\sqrt{n})$ and, consequently, $\log n = O(\sqrt{n})$ (if it grows slower than $\sqrt{n}$, it also grows at most as fast as $\sqrt{n}$). The other symbols Θ, Ω, ω do not apply.

Obviously, n grows faster than $\sqrt{n}$, so we need to put $n = \omega(\sqrt{n})$ and, consequently, $n = \Omega(\sqrt{n})$ (if it grows faster than $\sqrt{n}$, it also grows at least as fast as $\sqrt{n}$). None of the other symbols Θ, O, o apply.

For the third line, n grows slower than $n \log n$, thus $n = o(n \log n)$ and $n = O(n \log n)$.

In row 4, the two expressions are asymptotically the same since by the hint, $(\log n)^3 = o(n^2)$ and hence $n^2 + (\log n)^3 = O(n^2)$. Alternatively, we can argue that $n^2 \le n^2 + (\log n)^3 \le c_2 n^2$

for some constant c_2, some n_0 and all $n \geq n_0$ (for example, $c_2 := 2$ and $n_0 := 100$). Hence $n^2 = \Theta(n^2 + (\log n)^3)$ and this implies that O and Ω also hold. (Remember that every time you tick Θ, you also need to tick O and Ω, but don't tick o and ω.)

For row 5, we can use the hint that tells us that every polynomial function grows slower than any exponential function. Hence, $2^n = \omega(n^3)$, thus $2^n = \Omega(n^3)$ and none of the other symbols apply.

For row 6, we have $2^{n/2} = o(2^n)$ as the former is the square root of the latter. Another justification for o is that dividing $2^{n/2}$ by 2^n gives $2^{n/2}/2^n = 2^{-n/2}$, which goes to 0 as n grows.

In the final row, we use that two logarithms of n have the same order of growth as $\log_x(n) = \log_y(n)/\log_y(x)$ and $\log_y(x)$ is constant (for $x, y > 1$). Hence $\log_2 n$ and $\log_{10} n$ have the same order of growth. We can put Θ and this also implies that we need to tick O and Ω.

$f(n)$	$g(n)$	O	o	Ω	ω	Θ
$\log n$	$\sqrt{n}$	yes	yes	no	no	no
n	$\sqrt{n}$	no	no	yes	yes	no
n	$n \log n$	yes	yes	no	no	no
n^2	$n^2 + (\log n)^3$	yes	no	yes	no	yes
2^n	n^3	no	no	yes	yes	no
$2^{n/2}$	2^n	yes	yes	no	no	no
$\log_2 n$	$\log_{10} n$	yes	no	yes	no	yes

Hints: the book states that every polynomial of $\log n$ grows strictly slower than every polynomial n^ε, for constant $\varepsilon > 0$. For example, $(\log n)^{100} = o(n^{0.01})$. Likewise, every polynomial grows slower than every exponential function 2^{n^ε}, for example $n^{100} = o(2^{n^{0.01}})$.

To convert the base of a logarithm, use $\log_x(n) = \log_y(n)/\log_y(x)$.

Solution: Question 2.3 (0.5 marks)

State the number of "foo" operations for each of the following algorithms in Θ-notation. Pay attention to indentation and how long loops are run for. Justify your answer by stating constants $c_1, c_2, n_0 > 0$ from the definition of $\Theta(g(n))$ in your answer.

Example: Line 1 is executed once and line 3 is executed $n-4$ times. So the number of foos is $1 + n - 4 = n - 3 = \Theta(n)$ as $c_1 n \leq n - 3 \leq c_2 n$ for all $n \geq n_0$ when choosing, say, $n_0 = 6, c_1 = 1/2, c_2 = 1$.

```
EXAMPLE ALGORITHM
1: foo
2: for i = 1 to n − 4 do
3:     foo
```

```
ALGORITHM A
foo
for i = 1 to n do
    for j = 1 to n−2 do
        foo
        foo
        foo
```

```
ALGORITHM B
foo
for i = 1 to n do
    foo
for i = 1 to n/2 do
    foo
    foo
```

```
ALGORITHM C
foo
for i = 1 to n do
    for j = 1 to i do
        foo
    foo
foo
```

Solutions: For Algorithm A, line 1 is executed once and lines 4–6 are executed $n \cdot (n-2)$ times each. The total is $1 + 3n(n-2) = 3n^2 - 6n + 1 = \Theta(n^2)$ as $c_1 n^2 \leq 3n^2 - 6n + 1 \leq c_2 n^2$ for all $n \geq 3$ (thus $n_0 = 3$) when choosing, say, $c_1 = 1/2$ and $c_2 = 3$.

(If you have chosen other constants $0 < c_1 < 3$ and $c_2 \geq 3$ then your answer is correct so long as your n_0 is chosen large enough.)

For Algorithm B, line 1 is executed once, line 3 is executed n times, and lines 5 and 6 are executed $n/2$ times each. Hence the number of foos is $1 + n + 2n/2 = 2n + 1 = \Theta(n)$ as $c_1 n \leq 2n + 1 \leq c_2 n$ for all $n \geq 1$ (thus $n_0 = 1$) when choosing, say, $c_1 = 2$ and $c_2 = 3$.

(If you have chosen other constants $0 < c_1 \leq 2$ and $c_2 > 2$ then your answer is correct so long as n_0 is chosen large enough.)

For Algorithm C, lines 1 and 6 are executed once. The inner for loop goes from 1 to i, hence line 4 is executed $\sum_{i=1}^{n} i = n(n+1)/2$ times. Line 5 is executed n times. The total is $2+n(n+1)/2+n = n^2/2 + 3n/2 + 2 = \Theta(n^2)$ as $c_1 n^2 \leq n^2/2 + 3n/2 + 2 \leq c_2 n^2$ for $n \geq 1$ when choosing, say, $c_1 = 1/2$ and $c_2 = 4$.

(If you have chosen other constants $0 < c_1 \leq 1/2$ and $c_2 > 1/2$ then your answer is correct so long as n_0 is chosen large enough.)

Question 2.4 (0.5 marks)

Recall from Lecture 2 that a statement like $2n^2 + \Theta(n) = \Theta(n^2)$ is true if *no matter how the anonymous functions are chosen on the left of the equal sign, there is a way to choose the anonymous functions on the right of the equal sign to make the equation valid.* You might want to think of the $\Theta(n)$ on the left-hand side being a placeholder for some (anonymous) function that grows as fast as n.

For each of the following statements, state whether it is true or false. Justify your answers.

1. $O(\sqrt{n}) = O(n)$
   Solution: true.

   This equation can be read as "some (anonymous) function that grows at most as fast as $\sqrt{n}$ grows at most as fast as n". Whatever this anonymous function is, if it grows at most as fast as $\sqrt{n}$, it also grows at most as fast as n. Hence the statement is true. We can also express this in set notation as $O(\sqrt{n}) \subseteq O(n)$.

2. $n + o(n^2) = \omega(n)$
   Solution: false.

   The statement can be read as "the sum of n plus some (anonymous) function that grows slower than n^2 grows faster than n". The statement needs to hold for *all* anonymous functions $o(n^2)$, that is, all functions that grow slower than n^2. This includes, say, n, which obviously grows slower than n^2 and hence is included in the set $o(n^2)$. Then the left-hand side would be $n + n = 2n$, which is not in $\omega(n)$. So the statement is false as it does not hold for all functions in $o(n^2)$. In set notation, $n + o(n^2) \not\subseteq \omega(n)$.

3. $3n \log n + O(n) = \Theta(n \log n)$
   Solution: true.

   The left-hand side can be read as "$3n \log n$ plus some (anonymous) function that grows at most as fast as n" and then the statement asserts that this sum grows as fast as $n \log n$. The term $3n \log n$ dominates the left-hand side as every function in $O(n)$ grows more slowly than $3n \log n$, hence $O(n)$ is just a small-order term compared to $3n \log n$. The left-hand side grows asymptotically like $n \log n$, hence it is in $\Theta(n \log n)$. In set notation, we have $3n \log n + O(n) \subseteq \Theta(n \log n)$.

Also, explain why the statement "The running time of Algorithm A is at least $O(n^2)$" is meaningless.

Solution: "$O(n^2)$" is read as "at most cn^2" for a constant $c > 0$ and all $n \geq n_0$. So the statement spells out as "The running time of Algorithm A is at least at most cn^2", which is obviously pointless.

Question 2.5 (0.5 marks)

The following algorithm computes the product C of two $n \times n$ matrices A and B, where $A[i, j]$ corresponds to the element in the i-th row and the j-th column.

```
MATRIX-MULTIPLY(A, B)
1: for i = 1 to n do
2:     for j = 1 to n do
3:         C[i, j] := 0
4:         for k = 1 to n do
5:             C[i, j] := C[i, j] + A[i, k] · B[k, j]
6: return C
```

Give the running time of the algorithm (number of operations in a RAM machine) in Θ-notation. Justify your answer. Feel free to use the rules on calculating with Θ-notation from the lecture.

Solution: The first line is executed $\Theta(n)$ times, lines 2 to 3 are each executed $\Theta(n^2)$ times and cost $\Theta(1)$. The inner for loop is executed $\Theta(n^3)$ times, and the time for one execution of lines 4 to 5 is $\Theta(1)$. The time for the return statement is $\Theta(1)$. So the total time is

$$\Theta(n) + \Theta(n^2) \cdot \Theta(1) + \Theta(n^3) \cdot \Theta(1) + \Theta(1) = \Theta(n^3).$$

Programming Question 2.6 (0.25 marks)

Implement MATRIX-MULTIPLY(A,B).