University of Limerick
Department of Mathematics and Statistics
Dr. S. Franz

MA4016 - Engineering Mathematics 6

Solution Sheet 2: Algorithms (February 12, 2010)

1. $\mathcal{O}(x^2)$: **a**), **b**), **c**), **d**), **g**), **h**
$\Omega(x^2)$: **c**), **e**), **f**), **g**)
$\Theta(x^2)$: **c**), **g**)

2.

$$\begin{aligned}\textbf{a})\ f(n) &= \mathcal{O}(n^3 \log n),\\ \textbf{b})\ f(n) &= \mathcal{O}(6^n),\\ \textbf{c})\ f(n) &= \mathcal{O}(n^{2n})\end{aligned}$$

3. $\mathcal{O}(1)$: There exists $C \geq 0$, $k \geq 0$ such that $|f(x)| \leq C$ for $x \geq k$. Thus the absolute values of f are bounded for large arguments. Example: $f(x) = \sin(x)$.
$\Theta(1)$: There exists $C_1 > 0$, $C_2 \geq 0$, $k \geq 0$ such that $0 < C_1 \leq |f(x)| \leq C_2$. Thus the absolute value is bounded from above and bounded away from zero from below. Examples: $f(x) = \arctan(x)$, $2 + \sin(x)$, $(-1)^{\lfloor x \rfloor}$.

4. Construction like binary search (see lecture) but with two intermediate values and two if-clauses per inner loop. Thus the complexity is

$$3 \log_3 n = \mathcal{O}(\log n)$$

This algorithm is slightly better then binary search because

$$3 \log_3 n = 3 \log_3(2) \log_2 n \approx 1.893 \log_2 n < 2 \log_2 n.$$

Note that with further splitting into 4 sublists and nested if-clauses one can create an algorithm with complexity

$$3 \log_4 n = 3 \log_4(2) \log_2 n = 1.5 \log_2 n.$$

6. Introduce logical variable to monitor whether changes are made and check it in while loop too.
Best-case complexity: $\mathcal{O}(n)$ if list is already sorted.
Worst-case complexity: $\mathcal{O}(n^2)$ if list is wrongly sorted.
Average-case complexity: $\mathcal{O}(n^2)$

7. Use the formulas given in the lecture and replace the M's by their definition.