Algorithms and Data Structures 2016/17  
Week 3 tutorial sheet (30 Jan. - 3 Feb.)

Below are a list of suggested exercises, some from the back of either lecture 1, 2 or 3). Ask your tutor to cover the questions most important to you. You should see the tutorial as a resource to get answers to any questions, so don’t feel compelled to stick to the sheet.

1. Work out (don’t bother proving) for each pair of expressions below, whether $A$ is $O(B)$, $\Omega(B)$, $\Theta(B)$ (it could be none of these). Assume $k \geq 1, \epsilon > 0, c > 1$ are all constants.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$O$</th>
<th>$\Omega$</th>
<th>$\Theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lg^k n$</td>
<td>$n^\epsilon$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n^k$</td>
<td>$c^n$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sqrt{n}$</td>
<td>$n^{\ln n}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$2^n$</td>
<td>$2^{n/2}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n^{\ln n}$</td>
<td>$m^{\ln n}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\lg(n!)$</td>
<td>$\lg(n^n)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Consider the recursive algorithm REC-MAX (below) for finding the maximum element in an array of integers.

Give a recurrence for the worst-case running time of REC-MAX, justify its constants with reference to the algorithm, and solve the recurrence using the Master Theorem.

**Algorithm REC-MAX($A$, $i$, $j$)**

(a) if $i < j$ then

(b) $m \leftarrow \lfloor \frac{i+j}{2} \rfloor$

(c) $\ell \leftarrow$ REC-MAX($A$, $i$, $m$)

(d) $r \leftarrow$ REC-MAX($A$, $m+1$, $j$)

(e) if $\ell \geq r$ then

(f) return $\ell$

(g) else

(h) return $r$

(i) else

(j) return $A[i]$

3. Consider the following recurrence:

$$T(n) = \begin{cases} 
1 & \text{if } n = 1, \\
4T(\lfloor n/2 \rfloor) + n^2 & \text{if } n > 1.
\end{cases}$$

Prove $T(n) \in \Omega(n^2 \cdot \lg(n))$ by first principles (not using Master theorem) in 3 steps:
(a) Prove by induction $T(\hat{n}) = \hat{n}^2(1 + \lg(\hat{n}))$ if $\hat{n} = 2^p$ for $p \in \mathbb{N}^0$ (power-of-2 case).

(b) Prove by induction that $T(j) \leq T(k)$ for all $j < k$.

Similar to “Step 2” for MERGE-SORT in Lecture slides 2,3.

(c) For the lower bound, consider the largest power-of-2 $2^q$ satisfying $2^q \leq n$. Show that we have $2^q > n/2$. Hence use (b) and (a) to show $T(n) \in \Omega(\hat{n}^2 \cdot \lg(n))$.

This is similar to “Step 3” for MERGE-SORT in Lecture slides 2,3. However, because we want to prove $\Omega$, we choose the closest power-of-2 lower than $n$.

4. Define

$$p(n) = \sum_{i=0}^{d} a_i n^i,$$

where $a_d > 0$. So $p(n)$ is a degree-$d$ polynomial in $n$. Let $k$ be some constant.

Prove the following by first principles, from the definitions of $O(\cdot)$, $\Omega(\cdot)$ and $\Theta(\cdot)$.

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

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

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

5. Use Strassen’s algorithm to compute the matrix product

$$\begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \begin{pmatrix} 8 & 4 \\ 6 & 2 \end{pmatrix}.$$

This is Exercise 28.2-1 of [CLRS].

6. Describe an algorithm for efficiently multiplying a $(p \times q)$ matrix with a $(q \times r)$ matrix, where $p, q, r$ are arbitrary positive integers. The running time should be $\Theta(n^{\log_2(7)})$, where $n = \max\{p, q, r\}$. 