Exercise Sheet 1
COMS10007 Algorithms 2018/2019

05.02.2019

Reminder: $\log n$ denotes the binary logarithm, i.e., $\log n = \log_2 n$.

1 O-notation

Give formal proofs of the following statements using the definition of Big-O from the lecture.

1. $n^2 \in O(n^3)$.

   Proof. To fulfill the definition of Big-O, we need to show that there are constants C, n_0 such that $n^2 \leq Cn^3$, for every $n \geq n_0$. The previous inequality is equivalent to $1 \leq Cn$. $C = 1$ and $n_0 = 1$ clearly fulfills this requirement. □

2. $\frac{2n^2}{\log n} \in O(\frac{n^2}{\log \log n})$. ($\log \log n$ is short for $\log(\log n)$)

   Proof. We need to find constants C, n_0 such that $\frac{2n^2}{\log n} \leq C \cdot \frac{n^2}{\log \log n}$. The previous inequality can be transformed into the following equivalent one:

   $$\begin{aligned} \frac{2n^2}{\log n} &\leq C \cdot \frac{n^2}{\log \log n} \\ 2 \log \log n &\leq C \log n \ . \end{aligned}$$

   Let us pick $n_0 = 4$. Observe that:

   $$\begin{aligned} (2 \log \log n)' &= \frac{2}{\ln(2) \ln(n) n} \\ (C \log n)' &= \frac{C}{\ln(2) n} \ . \end{aligned}$$

   The racetrack principle implies that $2^{2 \log \log n} \leq 2^{C \log n}$ holds for every $n \geq 4$ if $\frac{2}{\ln(2) \ln(n) n} \leq \frac{C}{\ln(2) n}$ holds, for every $n \geq 4$. We hence need to show that we can pick C such that $\frac{2}{\ln(2) \ln(n) n} \leq \frac{C}{\ln(2) n}$ holds.

   $$\begin{aligned} \frac{2}{\ln(2) \ln(n) n} &\leq \frac{C}{\ln(2) n} \\ e^{2/C} &\leq n \ , \end{aligned}$$

   which holds for $n \geq n_0 = 4$ if we select for example $C = 2$ (recall that $e \approx 2.71$). This proves the result. □

3. $2^{\sqrt{\log n}} \in O(n)$.

Proof. We need to find constants C, n_0 such that $2^{\sqrt{\log n}} \leq Cn$, for every $n \geq n_0$. We pick $C = 1$. Then it remains to find an n_0 such that $2^{\sqrt{\log n}} \leq n$, for every $n \geq n_0$. Next, observe that $n = 2^{\log n}$. Hence, we need to find an n_0 such that $\sqrt{\log n} \leq \log n$, for every $n \geq n_0$. Observe that the right side of the previous inequality is the square of the left side. Since it holds that $x^2 \geq x$, for every $x \geq 1$, we can pick $n_0 = 2$, since $\log n \geq 1$ for every $n \geq 2$. □

4. Prove the following statements from the lecture:

   (a) $f \in O(h_1), g \in O(h_2)$ then $f + g \in O(h_1 + h_2)$

   Proof. We need to find constants C, n_0 such that $f(n) + g(n) \leq C(h_1(n) + h_2(n))$, for every $n \geq n_0$. Since $f \in O(h_1)$, we know that there are constants c_1, n_1 such that:

   $$f(n) \leq c_1 \cdot h_1(n), \text{ for every } n \geq n_1 \ .$$

   Similar, since $g \in O(h_2)$, we know that there are constants c_2, n_2 such that:

   $$g(n) \leq c_2 \cdot h_2(n), \text{ for every } n \geq n_2 \ .$$

   Combining these two inequalities, we obtain:

   $$\begin{aligned} f(n) + g(n) \quad &\leq \quad c_1 \cdot h_1(n) + c_2 \cdot h_2(n) \leq \max\{c_1, c_2\} \left(h_1(n) + h_2(n)\right), \\ &\text{for every } n \geq \max\{n_1, n_2\} \ . \end{aligned}$$

   Hence, setting $C = \max\{c_1, c_2\}$ and $n_0 = \max\{n_1, n_2\}$ completes the proof. □

   (b) $f \in O(h_1), g \in O(h_2)$ then $f \cdot g \in O(h_1 \cdot h_2)$

   Proof. We need to find constants C, n_0 such that $f(n) \cdot g(n) \leq C(h_1(n) \cdot h_2(n))$, for every $n \geq n_0$. Since $f \in O(h_1)$, we know that there are constants c_1, n_1 such that:

   $$f(n) \leq c_1 \cdot h_1(n), \text{ for every } n \geq n_1 \ .$$

   Similar, since $g \in O(h_2)$, we know that there are constants c_2, n_2 such that:

   $$g(n) \leq c_2 \cdot h_2(n), \text{ for every } n \geq n_2 \ .$$

   Combining these two inequalities, we obtain:

   $$\begin{aligned} f(n) \cdot g(n) \quad &\leq \quad c_1 \cdot h_1(n) \cdot c_2 \cdot h_2(n) = (c_1 \cdot c_2) \left(h_1(n) \cdot h_2(n)\right), \\ &\text{for every } n \geq \max\{n_1, n_2\} \ . \end{aligned}$$

   Hence, setting $C = c_1 \cdot c_2$ and $n_0 = \max\{n_1, n_2\}$ completes the proof. □

   Remind yourself why these statements could be useful for the analysis of algorithms.

5. Given are the functions:

$$f_1 = 2^{\sqrt{n}}, f_2 = \log^2(20n), f_3 = n!, f_4 = \frac{1}{2}n^2/\log(n), f_5 = 4\log^2(n), f_6 = 2^{\sqrt{\log n}} \ .$$

   Relabel the functions such that $f_i \in O(f_{i+1})$ (no need to give any proofs here).

Proof. The ordering is:

$$\log^2(20n), 4\log^2(n), 2^{\sqrt{\log n}}, \frac{1}{2}n^2/\log(n), 2^{\sqrt{n}}, n!$$

The positions of the two expressions $\log^2(20n)$, $4\log^2(n)$ can be exchanged. □

2 Θ and Ω

1. Let $c > 1$ be a constant. Prove or disprove the following statements:

   (a) $\log_c n \in \Theta(\log n)$.

   Proof. We need to find constants c_1, c_2, n_0 such that

   $$c_1 \log n \leq \log_c n \leq c_2 \log n \ ,$$

   for every $n \geq n_0$. Observe that $\log_c n = \frac{\log n}{\log c}$. We can hence chose $c_1 = c_2 = \frac{1}{\log c}$ and $n_0 = 1$, since $c_1 \cdot \log n = c_2 \cdot \log n = \log_c n$. This clearly holds for every $n \geq 1$. □

   (b) $\log(n^c) \in \Theta(\log n)$.

   Proof. Again, we need to find constants c_1, c_2, n_0 such that

   $$c_1 \log n \leq \log(n^c) \leq c_2 \log n \ ,$$

   for every $n \geq n_0$. Observe that $\log(n^c) = c \log n$. We can hence chose $c_1 = c_2 = c$ and $n_0 = 1$. □

2. Let $c > 2$ be a constant. Prove or disprove the following statement:

   $$2^n \in \Theta(c^n) \ .$$

   Proof. This statement is false. We will show that $c^n \notin O(2^n)$. This disproves this statement since if $f \in \Theta(g)$ then $g \in O(f)$ as well.
   For the sake of a contradiction, suppose that $c^n \in O(2^n)$. Then there are constants d, n_0 such that

   $$c^n \leq d \cdot 2^n \ ,$$

   for every $n \geq n_0$. Taking logarithms on both sides, we obtain the equivalent inequality:

   $$\begin{aligned} n\log(c) &\leq \log(d2^n) = \log(d) + n \\ n &\leq \frac{\log(d)}{\log(c) - 1} \ . \end{aligned}$$

   Observe that we only obtain the last inequality since $c > 2$ (since $c > 2$ we also have $\log c > 1$ and $\log(c) - 1 > 0$). This inequality hence does not hold for every $n > \frac{\log(d)}{\log(c)-1}$. This is a contradiction to the assumption that it holds for every $n \geq n_0$. □

3. Prove that the following two statements are equivalent:

   (a) $f \in \Theta(g)$.

(b) $f \in O(g)$ and $g \in O(f)$.

Proof. Assume that $f \in \Theta(g)$. This means that there are constants c_1, c_2, n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$, for every $n \geq n_0$.
To show that $f \in O(g)$, we need to show that there are constants c, n_0' such that $f(n) \leq cg(n)$, for every $n \geq n_0'$. This follows immediately by chosing $c = c_2$ and $n_0' = n_0$.
To show that $g \in O(f)$, we need to show that there are constants c, n_0' such that $g(n) \leq cf(n)$, for every $n \geq n_0'$. This follows immediately by chosing $c = \frac{1}{c_1}$ and $n \geq n_0'$.
Next, we assume that $f \in O(g)$ and $g \in O(f)$. This implies that there are constants c_1, n_1 such that $f(n) \leq c_1 g(n)$, for every $n \geq n_1$, and constants c_2, n_2 such that $g(n) \leq c_2 f(n)$, for every $n \geq n_2$. We need to show that there are constants d_1, d_2, n_0 such that $d_1 g(n) \leq f(n) \leq d_2 g(n)$, for every $n \geq n_0$. We can chose $d_2 = c_1$, $d_1 = \frac{1}{c_2}$, and $n_0 \geq \max\{n_1, n_2\}$. □

4. Prove that the following two statements are equivalent:

   (a) $f \in \Omega(g)$.

   (b) $g \in O(f)$.

Proof. Let's first assume that $f \in \Omega(g)$. This means that there are constants c_1, n_1 such that $c_1 g(n) \leq f(n)$, for every $n \geq n_1$. We need to show that there are constants c_2, n_2 such that $g(n) \leq c_2 f(n)$, for every $n \geq n_2$. We can pick $c_2 = \frac{1}{c_1}$ and $n_2 = n_1$.
The reverse direction, i.e., assuming that $g \in O(f)$ and deducing that $f \in \Omega(g)$ is very similar. □

3 Peak Finding in 2D

In the lecture we discussed a recursive algorithm for PeakFinding. Below is an algorithm that finds a peak in two dimensions. Your task is to analyze this algorithm, by bounding its runtime and proving its correctness. As in the lecture, the runtime of the algorithm is defined as the number of accesses to the input matrix.

Let A be an n-by-n matrix of integers. A *peak* in A is a position (i, j) such that $A_{i,j}$ is at least as large as its (at most) 4 neighbors (above, below, left, and right). The algorithm is defined for non-square matrices. It is recursive and proceeds as follows:

Require: n-by-m matrix A of integers
Suppose that the number of columns is larger than the number of rows, i.e., $n \geq m$. If this is not the case then consider A^T (i.e., rotate the matrix by 90°) instead of A. Observe that a peak in A^T is also necessarily a peak in A.
if $n \leq 10$ **then**
Compute the maximum of A and **return** its position
end if
Find the position of a maximum $(i_{\max}, j_{\max})$ among the elements in the boundary (top row, bottom row, first column, last column) and the most central column (column $\lceil n/2 \rceil$).
if $(i_{\max}, j_{\max})$ is a peak in A **then**
return $(i_{\max}, j_{\max})$
else
Let (i', j') be an adjacent element (either above, below, left, or right) of $(i_{\max}, j_{\max})$ such that $A_{i',j'} > A_{i_{\max},j_{\max}}$.
$A_{i',j'}$ is necessarily contained in either the submatrix A_1 consisting of the first $\lceil n/2 \rceil - 1$ columns or the submatrix A_2 consisting of columns $\lceil n/2 \rceil + 1, \lceil n/2 \rceil + 2 \ldots n$. Let A_s be this submatrix (i.e., $s \in \{1, 2\}$).
return Find a peak in A_s recursively using this algorithm
end if

It is not required that you give formal proofs in this exercise. However, try to find a clear argumentation.

1. Explain the algorithm in plain English.
2. Argue why the algorithm is correct, i.e., why is a peak found by the algorithm in the submatrix A_s necessarily also a peak in A?
3. Bound the runtime of this algorithm using O-notation when executed on an n-by-n matrix.