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. Recall the definition of Big-O from the lecture: A function $f(n)$ is in $O(g(n))$ if there exist constants C and n_0 such that $f(n) \leq C \cdot g(n)$, for every $n \geq n_0$. To prove that $n^2 \in O(n^3)$, we thus need to find constants C and n_0 such that $n^2 \leq C \cdot n^3$, for every $n \geq n_0$. In this exercise it is obvious how to select these constants: We can for example set $C = n_0 = 1$ since $n^2 \leq 1 \cdot n^3$ clearly holds for every $n \geq 1 (= n_0)$, and we are done.
   In general, we transform the inequality at hand such that we obtain an equivalent inequality of the form $n \geq X$, where X is an expression that depends on C and does not depend on n. We can then select an approximate value for C and obtain a condition on n_0. Following this strategy for the current exercise, we start with the inequality $n^2 \leq Cn^3$ and transform it into one of the form $n \geq X$. In this case, we obtain $n \geq \frac{1}{C}$. We can then select a value for C, and then obtain a condition on n_0. If we select $C = 10$, then we obtain $n \geq \frac{1}{10}$, and we can select any $n_0 \geq \frac{1}{10}$. Observe that n_0 needs to be an integer, hence any integer $n_0 \geq 1$ works here.
   In order to fulfill the definition of Big-O, observe that we do not need to find the smallest possible values for C and n_0. We might as well for example select $C = 100$ and $n_0 = 20$, or any other values ≥ 1. Observe that the inequality $n^2 \leq Cn^3$ is equivalent to $1 \leq Cn$, and this inequality holds if $C \geq 1$ and $n \geq 1$.
   In a general situation, it may be not as straightforward to find good values for C and n_0, as illustrated in the next exercise. □

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

Proof. As before, 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 suggested strategy, i.e., transforming the inequality into one of the form $n \geq X$, where X does not depend on n, breaks down here, since we have both a $\log\log n$ term and a $\log n$ term. In such a situation the racetrack principle (see lecture) will be useful. But let us first transform the inequality into a simpler one:

$$\begin{aligned} \frac{2n^2}{\log n} &\leq C \cdot \frac{n^2}{\log\log n} \\ \frac{2}{\log n} &\leq C \cdot \frac{1}{\log\log n} \\ 2\log\log n &\leq C \log n \ . \end{aligned}$$

We thus need to find constants C and n_0 such that $2\log\log n \leq C \log n$, for every $n \geq n_0$. Recall that the racetrack principle states that if we find an integer n_0 such that $f(n_0) \geq g(n_0)$, for some functions f, g, and we can further show that $f'(n) \geq g'(n)$, for every $n \geq n_0$, then it holds that $f(n) \geq g(n)$, for every $n \geq n_0$. This is exactly what we need here.

We proceed as follows: First, we need to find a value n_0 such that $2\log\log n_0 \leq C \log n_0$. Let us pick a convenient value n_0 such that both $\log\log n_0$ and $\log n$ are integral. Let us pick $n_0 = 4$ (we could for example also pick 16).

Next, we need to show that $(2\log\log n)' \leq (C \log n)'$, for any $n \geq 4$ (recall we have picked $n_0 = 4$ already). This is a bit technical here (I apologize for using the $\log\log n$ function in this exercise!). Once this is shown, it follows from the racetrack principle that $2\log\log n \leq C \log n$, for every $n \geq n_0 = 4$ and we are done. We thus compute:

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

and we obtain:

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

which holds for $n \geq n_0 = 4$ if we select for example $C = 2$ (recall that $e \approx 2.71$). The racetrack principle thus proves that $2\log\log n \leq C \log n$ holds for $C = 2$ and for every $n \geq n_0 = 4$. This proves the result.

Comment: In order to avoid computing the derivative of $\log\log n$, we could transform the inequality $2\log\log n \leq C \log n$ above further. By exponentiation, we obtain: $2^{2\log\log n} \leq 2^{C\log n}$ which is equivalent to $(\log n)^2 \leq n^C$. We could then proceed from here as above by applying the racetrack principle. The two functions involved would be $(\log n)^2$ and n^C, and it is easier to compute their derivatives. □

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

Proof. Again, we need to find constants C, n_0 such that $2^{\sqrt{\log n}} \leq Cn$, for every $n \geq n_0$. Observe that:

$$2^{\sqrt{\log n}} \quad \leq \quad Cn = C2^{\log n} \ .$$

Let us pick $C = 1$. Next, we need to find a value for n_0 and show that $2^{\sqrt{\log n}} \leq 2^{\log n}$, for all $n \geq n_0$. This holds if $\sqrt{\log n} \leq \log n$. As in the previous exercise, we could now use the racetrack principle to find a value for n_0. A simpler argument is as follows: Observe that the right side is the square of the left side. We clearly have $x \leq x^2$, for every $x \geq 1$. Hence, if $\sqrt{\log n} \geq 1$, then $\sqrt{\log n} \leq \log n$ holds as well. We thus pick $n_0 = 2$, since $\sqrt{\log 2} = 1$. Hence, $C = 1$ and $n_0 = 2$ work here. Again, we do not need to find the smallest values for C and n_0. Any values $C \geq 1$ and $n_0 \geq 2$ work here. □

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:

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

   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. Similar as the last exercise, 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:

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

   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:

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

Here is a justification for this ordering (this is not a proof). We can use the following rough ordering suggested in the lecture:

"poly-log" , "polynomial" , "exponential"

Concerning the two poly-logs, we have $\log^2(20n) = \Theta(\log^2 n)$ and $4\log^2(n) = \Theta(\log^2 n)$. We could thus also exchange the places of these two functions in the previous ordering.
The function $2^{\sqrt{\log n}}$ is neither a polynomial nor a poly-log. To check that it grows faster than a poly-log, observe that $(\log n)^c = 2^{c\log\log n}$. Since $\sqrt{\log n}$ grows faster than $\log\log n$, we conclude that $2^{\sqrt{\log n}}$ grows faster than any poly-log. To see that $2^{\sqrt{\log n}}$ grows slower than every polynomial, observe that $n = 2^{\log n}$. Since $\log n$ grows faster than $\sqrt{\log n}$, the function $2^{\sqrt{\log n}}$ grows slower than the identity function n. We therefore place this function next. We have only one function that grows roughly as fast as a polynomial, which is $\frac{1}{2}n^2/\log(n)$ ($n^2/\log(n)$ is clearly between n and n^2, which are both polynomials). Concerning $n!$, observe that $n! = n\cdot(n-1)\cdot(n-2)\cdot\cdots\cdot 2\cdot 1 \geq (n/2)^{n/2} = 2^{\log(n/2)n/2}$. The exponent grows faster than $\sqrt{n}$ and hence $n!$ grows faster than $2^{\sqrt{n}}$.

□

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. Recall the definition of Θ: A function $f(n) \in \Theta(g(n))$ if there are constants c_1, c_2, n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$, for every $n \geq n_0$. Hence, 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 wrong. 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. In order to prove that two statements are equivalent, we assume first that the first statement holds and we deduce that the second statement then holds as well. Then we assume that the second statement holds and we deduce that the first statement then holds as well.
Let's first 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 choosing $c = c_2$ and $n_0' = n_0$ as above.
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 choosing $c = \frac{1}{c_1}$ 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.