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. $n^2 \in O(n^3)$.
2. $\frac{2n^2}{\log n} \in O(\frac{n^2}{\log \log n})$. ($\log \log n$ is short for $\log(\log n)$)
3. $2^{\sqrt{\log n}} \in O(n)$.
4. Prove the following statements from the lecture:
   (a) $f \in O(h_1), g \in O(h_2)$ then $f + g \in O(h_1 + h_2)$
   (b) $f \in O(h_1), g \in O(h_2)$ then $f \cdot g \in O(h_1 \cdot h_2)$

   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).

2 Θ and Ω

1. Let $c > 1$ be a constant. Prove or disprove the following statements:
   (a) $\log_c n \in \Theta(\log n)$.
   (b) $\log(n^c) \in \Theta(\log n)$.
2. Let $c > 2$ be a constant. Prove or disprove the following statement:
$$2^n \in \Theta(c^n) .$$
3. Prove that the following two statements are equivalent:
   (a) $f \in \Theta(g)$.

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

4. Prove that the following two statements are equivalent:

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

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

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.