Exercise Sheet 1
COMS10017 Algorithms 2020/2021

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

1 O-notation: Part I

Give formal proofs of the following statements using the definition of Big-O from the lecture (i.e., identify positive constants c, n_0 for which the definition holds):

1. $5\sqrt{n} \in O(n)$.

   Solution. We need to show that there are positive constants c, n_0 such that $5\sqrt{n} \leq c \cdot n$, for every $n \geq n_0$. This is equivalent to showing that $(\frac{5}{c})^2 \leq n$. We choose $c = 5$, which implies $1 \leq n$. We can thus select $n_0 = 1$, since then $1 \leq n$ holds for every $n \geq n_0$. This prove that $5\sqrt{n} \in O(n)$.

   Remark: Observe that there are many other combinations of values for c and n_0 that satisfy the inequality we need to prove. For example, if we pick $c = 1$ then we obtain $25 \leq n$ (which follows from $(\frac{5}{c})^2 \leq n$). In this case, we would have to choose a value for n_0 that is ≥ 25, in particular, $n_0 = 25$ would do. ✓

2. $n^2 + 10n + 8 \in O(\frac{1}{2}n^2)$.

   Solution. We need to show that there are positive constants c, n_0 such that $n^2+10n+8 \leq c \cdot \frac{1}{2}n^2$, for every $n \geq n_0$. To make our life easier, we use the following estimate:

   $$n^2 + 10n + 8 \leq n^2 + 10n^2 + 8n^2 = 19n^2 \ ,$$

   which holds for every $n \geq 1$. If we can prove that there are constants c, n_0 such that $19n^2 \leq c \cdot \frac{1}{2}n^2$ holds for every $n \geq n_0$, then these constants also work for showing that $n^2 + 10n + 8 \leq c \cdot \frac{1}{2}n^2$ for every $n \geq n_0$.

   This, however, is easy: We can pick $c = 38$ and $n_0 = 1$, which completes the proof. ✓

3. $n^3 + n^2 + n = O(n^3)$.

   Solution. We need to show that there are constants c, n_0 such that $n^3 + n^2 + n \leq c \cdot n^3$ holds for every $n \geq n_0$. Using the idea from the previous exercise, we use the inequality $n^3 + n^2 + n \leq 3n^3$, which holds for every $n \geq 1$, and prove instead that there are constants c, n_0 such that $3n^3 \leq cn^3$ holds for every $n \geq n_0$. Again, this is easy to do: We pick $c = 3$ and $n_0 = 1$. ✓

4. $10 \in O(1)$.

Solution. We need to show that there are positive constants c, n_0 such that $10 \leq c \cdot 1$, for every $n \geq n_0$. Observe that this expression does not depend on n at all. Therefore any positive value for n_0 would work, e.g., $n_0 = 1$ (or $n_0 = 23$ or any other value). We chose $c = 10$ which implies that $10 \leq c \cdot 1$ is satisfied. This proves that $10 \in O(1)$. ✓

5. $\sum_{i=1}^{n} i \in O(4n^2)$.

   Solution. First, observe that $\sum_{i=1}^{n} i = n(n+1)/2 = \frac{n^2}{2} + \frac{n}{2}$. We need to find positive constants c, n_0 such that $\frac{n^2}{2} + \frac{n}{2} \leq c \cdot 4n^2$, for every $n \geq n_0$. We pick $n_0 = 1$. Since $n \leq n^2$, for every $n \geq n_0 = 1$, we will satisfy the inequality $\frac{n^2}{2} + \frac{n^2}{2} \leq c \cdot 4n^2$, which is equivalent to $1 \leq 4c$. We can hence pick $c = 1$ and we are done. ✓

2 Racetrack Principle

Use the racetrack principle to prove the following statement:

$$n \leq e^n \text{ holds for every } n \geq 1 \ .$$

Solution. First, we verify that $n \leq e^n$ holds for $n = n_0 = 1$. This is true, since $1 \leq e$ holds. Next, we verify that $(n)' \leq (e^n)'$ holds for every $n \geq n_0$. We have $(n)' = 1$ and $(e^n)' = e^n$. We thus need to show that $1 \leq e^n$ holds for every $n \geq 1$. Taking the natural logarithm on both sides, we obtain $0 \leq n$, which is true for every $n \geq n_0 = 1$. Hence, $n \leq e^n$ holds for every $n \geq 1$. ✓

3 O-notation: Part II

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

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

   Solution. Similar as in the previous exercise, we know that there are constants c_1, c_2, n_1, n_2 such that $f(n) \leq c_1 \cdot h_1(n)$, for every $n \geq n_1$, and $g(n) \leq c_2 \cdot h_2(n)$, for every $n \geq n_2$. Then:

   $$f(n) \cdot g(n) \leq c_1 \cdot h_1(n) \cdot c_2 \cdot h_2(n) = c_1 c_2 \cdot h_1(n) h_2(n)$$

   for every $n \geq \max\{n_1, n_2\}$. We thus select $C = c_1 \cdot c_2$ and $N = \max\{n_1, n_2\}$ and obtain $f(n)g(n) \leq C(h_1(n)h_2(n))$, for every $n \geq N$. ✓

2. $2^n \in O(n!)$.

   Solution. To prove this statement, we will show that $2^n \leq C \cdot n!$ holds for $C = 2$ and every $n \geq 2$. To this end, observe that $2^n \leq 2n!$ is equivalent to $2^{n-1} \leq n!$. Observe that

   $$2^{n-1} = \underbrace{2 \cdot 2 \cdot \dots \cdot 2}_{(n-1) \text{ times}} \ ,$$

   and

   $$n! = \underbrace{2 \cdot 3 \cdot \dots \cdot n}_{(n-1) \text{ factors, each larger equal to } 2} \ .$$

Trading off the factors of the two expressions, we see that $2^{n-1} \leq n!$, which proves the result. ✓

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

Solution. We need to show that there are constants c, n_0 such that $2^{\sqrt{\log n}} \leq c{\cdot}n$ holds for every $n \geq n_0$. Observe that the previous inequality is equivalent to $2^{\sqrt{\log n}} \leq 2^{\log(n)+\log(c)}$, which holds if $\sqrt{\log n} \leq \log(n) + \log(c)$. Observe that $\sqrt{x} \leq x$ for every $x \geq 1$. Hence, $\sqrt{\log n} \leq \log(n)$ holds for every $\log(n) \geq 1$, or every $n \geq 2$. We can thus pick $n_0 = 2$ and $c = 1$ (observe that $\log(x) < 0$ for $x < 1$, we therefore couldn't choose $c < 1$). ✓

4 Fast Peak Finding

Consider the following variant of FAST-PEAK-FINDING where the "$\geq$" sign in the condition in instruction 4 is replaced by a "$<$" sign:

1. **if** A is of length 1 **then return** 0
2. **if** A is of length 2 **then** compare $A[0]$ and $A[1]$ and **return** position of larger element
3. **if** $A[\lfloor n/2 \rfloor]$ is a peak **then return** $\lfloor n/2 \rfloor$
4. Otherwise, **if** $A[\lfloor n/2 \rfloor - 1]$ **<** $A[\lfloor n/2 \rfloor]$ **then**
   return FAST-PEAK-FINDING$(A[0, \lfloor n/2 \rfloor - 1])$
5. **else**
   return $\lfloor n/2 \rfloor + 1+$ FAST-PEAK-FINDING$(A[\lfloor n/2 \rfloor + 1, n - 1])$

Give an input array of length 8 on which this algorithm fails.

Solution. Consider the instance $A[i] = i$, for every $0 \leq i \leq 7$. Then the algorithm recurses on the subarray $A[0 \dots 2]$ in line 4. Observe however that none of the elements in $A[0 \dots 2]$ constitute a peak in array A. ✓

5 Optional and Difficult

5.1 Advanced Racetrack Principle

Use the racetrack principle and determine a value n_0 such that

$$\frac{2}{\log n} \leq \frac{1}{\log \log n} \text{ holds for every } n \geq n_0 .$$

Hint: Transform the inequality and eliminate the log-function from one side of the inequality before applying the racetrack principle. If needed, apply the racetrack principle twice!
Recall that $(\log n)' = \frac{1}{n \ln(2)}$. The inequality $\ln(2) \geq 1/2$ may also be useful.

Solution. We use the provided "Hint" and transform the given inequality as follows:

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

We now pick $n_0 = 16$. Then, $(\log 16)^2 \leq 16$ holds. Next, observe that $((\log n)^2)' = \frac{2\ln(n)}{(\ln(2))^2 n}$ and $(n)' = 1$. Using the racetrack principle, it is enough to show that $\frac{2\ln(n)}{(\ln(2))^2 n} \leq 1$, for every $n \geq n_0 = 16$. This is equivalent to showing that $2\ln(n) \leq \ln(2)^2 n$ (for every $n \geq 16$). We now apply the racetrack principle again: To this end, we first verify that $2\ln(n) \leq \ln(2)^2 n$ holds for $n = n_0 = 16$: We indeed have $2\ln(16) = 2\ln(2^4) = 8\ln(2) \leq \ln(2)^2 \cdot 16$ (which holds since $\ln(2) \geq 1/2$). Next, observe that $(2\ln(n))' = \frac{2}{n}$ and $(\ln(2)^2 n)' = \ln(2)^2$. It thus remains to argue that $\frac{2}{n} \leq \ln(2)^2$ for every $n \geq 16$. The previous inequality is equivalent to $\frac{2}{\ln(2)^2} \leq \frac{2}{(\frac{1}{2})^2} \leq 8 \leq n$, which holds for every $n \geq 16$.

Hence, $\frac{2}{\log n} \leq \frac{1}{\log\log n}$ holds for every $n \geq 16$.

✓

5.2 Funding Two Peaks

We are given an integer array A of length n that has exactly two peaks. The goal is to find both peaks. We could do this as follows: Simply go through the array with a loop and check every array element. This strategy has a runtime of $O(n)$ (requires $c \cdot n$ array accesses, for some constant c). Is there a faster algorithm for this problem (e.g. similar to FAST-PEAK-FINDING)? If yes, give such an algorithm. If no, justify why there is no such algorithm.