Algorithms and Data Structures

Exercise Session 4

https://n.ethz.ch/~ahmala/and

Common Mistake

$n^3 \leq n^3 + n$

$\Rightarrow \log(n^3) \leq \log(n^3 + n)$

$n^3 \leq n^3 + n$

$\Rightarrow \log(n^3) \leq \log(n^3 + n)$ | log is a monotone increasing function

Prove or disprove: $\Omega(n^2) \cap \mathcal{O}(n^3) = \Theta(n^2) \cup \Theta(n^3)$

Prove or disprove: $\Omega(n^2) \cap \mathcal{O}(n^3) = \Theta(n^2) \cup \Theta(n^3)$

Solution: We disprove with the following counterexample. Let $f = n^2 \log n$. Then $f \in \Omega(n^2)$, $f \in \mathcal{O}(n^3)$, but $f \notin \Theta(n^2)$, and $f \notin \Theta(n^3)$.

Prove or disprove: Let $f, g \in \Theta(h)$, then $|f - g| \in \mathcal{O}(1)$.

Prove or disprove: Let $f, g \in \Theta(h)$, then $|f - g| \in \mathcal{O}(1)$.

Solution: We disprove with the following counterexample: $f(n) = n^2 + n$, $g(n) = n^2$, $h(n) = n^2$.

Is there a function $f : \mathbb{N} \to \mathbb{R}^+$ that is neither in $\mathcal{O}(n^2)$ nor in $\Omega(n^2)$? If no, prove, if yes, give an example.

Is there a function $f : \mathbb{N} \to \mathbb{R}^+$ that is neither in $\mathcal{O}(n^2)$ nor in $\Omega(n^2)$? If no, prove, if yes, give an example.

$$f(n) = \begin{cases} n & \text{if } n \text{ is even,} \\ n^3 & \text{if } n \text{ is odd.} \end{cases}$$

Exercise Sheet 4

Exercise 4.1 *Applying the master theorem.*

For this exercise, assume that n is a power of two (that is, $n = 2^k$, where $k \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}$).

(a) Let $T(1) = 1, T(n) = 4T(n/2) + 100n$ for $n > 1$. Using the master theorem, show that

$$T(n) \leq O(n^2).$$

(b) Let $T(1) = 5, T(n) = T(n/2) + \frac{3}{2}n$ for $n > 1$. Using the master theorem, show that

$$T(n) \leq O(n).$$

[1] For this asymptotic bound we assume $n \geq 2$ so that $n^{\log_2 a} \cdot \log n > 0$.

(c) Let $T(1) = 4, T(n) = 4T(n/2) + \frac{7}{2}n^2$ for $n > 1$. Using the master theorem, show that

$$T(n) \leq O(n^2 \log n).$$

Exercise 4.2 *Asymptotic notations.*

(a) **(This subtask is from January 2019 exam).** For each of the following claims, state whether it is true or false. You don't need to justify your answers.

claim	**true**	**false**
$\frac{n}{\log n} \leq O(\sqrt{n})$	$\square$	$\square$
$\log(n!) \geq \Omega(n^2)$	$\square$	$\square$
$n^k \geq \Omega(k^n)$, if $1 < k \leq O(1)$	$\square$	$\square$
$\log_3 n^4 = \Theta(\log_7 n^8)$	$\square$	$\square$

(b) **(This subtask is from August 2019 exam).** For each of the following claims, state whether it is true or false. You don't need to justify your answers.

claim	**true**	**false**
$\frac{n}{\log n} \geq \Omega(n^{1/2})$	$\square$	$\square$
$\log_7(n^8) = \Theta(\log_3(n^{\sqrt{n}}))$	$\square$	$\square$
$3n^4 + n^2 + n \geq \Omega(n^2)$	$\square$	$\square$
$(*) \quad n! \leq O(n^{n/2})$	$\square$	$\square$

Note that the last claim is challenge. It was one of the hardest tasks of the exam. If you want a 6 grade, you should be able to solve such exercises.

$$(*) \quad n! \leq O(n^{n/2})$$

Exercise 4.3 *Formal proof of correctness for Bubble Sort* **(1 point)**.

Recall the bubble sort algorithm that was introduced in the lecture.

Algorithm 1 Bubble Sort (input: array $A[1 \ldots n]$).

> **for** $j = 1, \ldots, n$ **do**
>
>   **for** $i = 1, \ldots, n-1$ **do**
>
>   **if** $A[i] > A[i+1]$ **then**
>
>   Swap $A[i]$ and $A[i+1]$

Prove correctness of this algorithm by mathematical induction.

Hint: *Use the invariant $I(j)$ that was introduced in the lecture: "After j iterations the j largest elements are at the correct place."*

Exercise 4.4 *Exponential search* **(1 point)**.

Suppose we are given a positive integer $N \in \mathbb{N}$, and a *monotonously increasing* function $f : \mathbb{N} \to \mathbb{N}$, meaning that $f(i) \geq f(j)$ for all $i, j \in \mathbb{N}$ with $i \geq j$. Assume that $\lim_{n\to\infty} f(n) = \infty$. We are tasked to determine the *smallest* integer $T \in \mathbb{N}$ such that $f(T) \geq N$.

(a) Describe an algorithm that finds an *upper bound* $T_{\text{ub}} \in \mathbb{N}$ on T, such that $f(T_{\text{ub}}) \geq N$ and $T_{\text{ub}} \leq 2T$, making $O(\log T)$ function calls to f.[2] Prove that your algorithm is correct, and uses at most the desired number of function calls.

(b) Describe an algorithm that determines the *smallest* integer $T \in \mathbb{N}$ such that $f(T) \geq N$, making $O(\log T)$ function calls to f. Prove that your algorithm is correct, and uses at most the desired number of function calls.

Hint: *Consider using a two-step approach. In the first step, apply the algorithm of part (a). For the second step, modify the binary search algorithm and apply it to the array* $\{1, 2, \ldots, T_{\text{ub}}\}$. *Use helper variables* $i_{\text{low}}, i_{\text{high}} \in \mathbb{N}$, *that satisfy* $i_{\text{low}} \leq T \leq i_{\text{high}}$ *at all times during the algorithm. In each iteration, update* i_{low} *and/or* i_{high} *so that the number of remaining options for* T *is halved.*

Exercise 4.5 *Counting function calls in loops (cont'd)* **(1 point).**

For each of the following code snippets, compute the number of calls to f as a function of $n \in \mathbb{N}$. We denote this number by $T(n)$, i.e. $T(n)$ is the number of calls the algorithm makes to f depending on the input n. Then T is a function from $\mathbb{N}$ to $\mathbb{R}^+$. For part (a), provide **both** the exact number of calls and a maximally simplified asymptotic bound in Θ notation. For part (b), it is enough to give a maximally simplified asymptotic bound in Θ notation. For the asymptotic bounds, you may assume that $n \geq 10$.

Algorithm 2

(a)

```
i ← 1
while i ≤ n do
    j ← i
    while 2^j ≤ n do
        f()
        j ← j + 1
    i ← i + 1
```

[2]For the asymptotic bounds here and also in the following we assume $T \geq 2$ such that $\log(T) > 0$.

Algorithm 3

(b)

```
function A(n)
    i ← 0
    while i < n^2 do
        j ← n
        while j > 0 do
            f()
            f()
            j ← j − 1
        i ← i + 1
    k ← ⌊n/2⌋
    for l = 0 ... 3 do
        if k > 0 then
            A(k)
            A(k)
```

You may assume that the function $T : \mathbb{N} \to \mathbb{R}^+$ denoting the number of calls of the algorithm to f is increasing.

Hint: *To deal with the recursion in the algorithm, you can use the master theorem.*

(c)* Prove that the function $T : \mathbb{N} \to \mathbb{R}^+$ from the code snippet in part (b) is indeed increasing.

Hint: *You can show the following statement by mathematical induction: "For all* $n' \in \mathbb{N}$ *with* $n' \leq n$ *we have* $T(n'+1) \geq T(n')$*".*

For the following code fragments count how many times the function f is called. Report the number of calls as nested sum, and then simplify your expression in Θ-notation and prove your result.

Hint: *Note that in order to justify your Θ-notation you are required to show two parts: an upper bound on your nested sum as well as a lower bound.*

a) Consider the snippet:

Algorithm 1

```
for j = 1, ..., n do
    k ← 1
    while k ≤ j do
        m ← 1
        while m ≤ j do
            f()
            m ← 2 · m
        k ← 2 · k
```

$$\sum_{j=1}^{n} \sum_{l=0}^{\lfloor \log_2 j \rfloor} \sum_{i=0}^{\lfloor \log_2 j \rfloor} 1 = \sum_{j=1}^{n} (\lfloor \log_2 j \rfloor + 1)^2 \leq \sum_{j=1}^{n} (\lfloor \log_2 n \rfloor + 1)^2 \leq O(n \log^2 n)$$

times. Notice that, when $n \geq 4$, then $\log_2(n/2) = \log_2 n - 1 \geq (\log_2 n)/2$. Therefore, for all $n \geq 4$ we have

$$\begin{aligned}
\sum_{j=1}^{n} \sum_{l=0}^{\lfloor \log_2 j \rfloor} \sum_{i=0}^{\lfloor \log_2 j \rfloor} 1 = \sum_{j=1}^{n} (\lfloor \log_2 j \rfloor + 1)^2 &\geq \sum_{j=\lceil n/2 \rceil}^{n} (\lfloor \log_2 (n/2) \rfloor + 1)^2 \\
&\geq \sum_{j=\lceil n/2 \rceil}^{n} \log_2 (n/2)^2 \geq n/2 \cdot ((\log_2 n)/2)^2 \geq \Omega(n \log^2 n),
\end{aligned}$$

How many times will f() be called?

```
int i,j;

for(i = 0; i < N; i++)
    f()

for(i = 0; i < M; i++)
    f()

```

Kahoot!

Peer Grading

Exercise 4.4

While sending to me please include the group you received their work in cc.

https://docs.google.com/spreadsheets/d/1owPsJsd9THBWInwFcVjKCc0f_r6n4pGwwDKMDdwaCjM/edit?usp=sharing