Weekly Assignment 1:
Big-O

September 2023

Some facts (no exercises)

Some tips and tricks (most of these are not needed for this particular exercise sheet, but might come in handy at some point during the course). Summations:

$$1 + 2 + \cdots + n = \sum_{i=0}^{n} i = \frac{n(n+1)}{2} \qquad 1 + r + \cdots + r^n = \sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}$$

If these are not familiar to you, try to prove them yourself (for example with induction), practicing such proofs is always useful ;-). About logarithms and exponential:

$$2^0 = 1 \qquad 2^1 = 2 \qquad 2^a 2^b = 2^{a+b} \qquad 2^{ab} = (2^a)^b = (2^b)^a$$

$$\log_2(1) = 0 \qquad \log_2(2) = 1 \qquad \log_2(ab) = \log_2(a) + \log_2(b)$$

If the logarithm rules are unfamiliar to you, try proving them from the exponentiation rules and the identity:

$$\log_2(2^x) = x \qquad \text{for all } x \in \mathbb{R}$$

1. Are the following statements true or not? No proof is required.
   (a) $n + 1 \in \Theta(n)$
   (b) $2n \in \Omega(n^2)$
   (c) $n + \log n \in \mathcal{O}(n)$
   (d) $2^{n+1} \in \Theta(2^n)$
   (e) $n\sqrt{n} \in \mathcal{O}(n \log n)$
   (f) $n! \in \mathcal{O}(2^n)$

2. Recall that in order to prove $f \in \mathcal{O}(g)$ one has to choose $c > 0$ and n_0 and then prove that $f(n) \leq cg(n)$ for all $n \geq n_0$.
   (a) Prove $n + 37 \in \mathcal{O}(n)$.
   (b) Prove that $\frac{1}{2}n(n-1) \in \mathcal{O}(n^2)$.
   (c) Prove that $n \in O(2^n)$. (Hint: use $c = 1$ and $n_0 = 1$, then prove the inequality with induction.)

3. Consider the following algorithm.

```
search(int v[], int n) {
    int i = 0
    bool foo = false
```

```
    while (i < n && !foo) {
        if (v[i] > 42) {
            foo = true
        }
        i++
    }
    return foo
}
```

(a) What is the worst case scenario? How many operations does it take in this case (your answer should use $\mathcal{O}$ notation and depend on n)?

(b) What is the best case scenario? How many operations does it take in this case (your answer should use $\mathcal{O}$ notation and depend on n)?

4. Consider the following algorithm, which takes as input an integer array `v` of length `n`. To analyse this algorithm, we will only count *array access* operations on line 4.

Algorithm 1 Reordering even and odd

Require: integer array v of length $n \geq 2$
1: $i \leftarrow 0$
2: $j \leftarrow n - 1$
3: **while** $i < j$ **do**
4:     **if** $v[i]$ is even **then**
5:         $i \leftarrow i + 1$
6:     **else if** $v[j]$ is even **then**
7:         swap($v[i]$, $v[j]$)
8:         $i \leftarrow i + 1$
9:         $j \leftarrow j - 1$
10:     **else**
11:         $j \leftarrow j - 1$
12:     **end if**
13: **end while**

(a) Show that line 4 is a good measure of the complexity of the algorithm, by showing that the number of times each other line is performed on a given input is in $\mathcal{O}(\#4)$, where #4 is the number of times line 4 is executed.

(b) How many times is Line 4 executed in the worst case (your solution should depend on n)? Describe some inputs for which this worst case is achieved.

(c) How many times is Line 4 executed in the best case (your solution should depend on n)? Describe some inputs for which this best case is achieved.

(d) Compare the best and worst case from your previous two solutions asymptotically, that is, using the $\mathcal{O}, \Theta, \Omega$ notation.

(e) Suppose we make a mistake and forget the assignment `j--` on line 9. Does the algorithm still work? How does this affect the best case? How does this compare asymptotically to the original best case?

5. The n-th harmonic number is the sum of the reciprocals of the first n natural numbers such that

$$H_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

Write a function returning H_n with $n \in \mathbb{N}$ and comment on its asymptotic time complexity.