Exercise Sheet 2
COMS10018 Algorithms

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

Example Question: Runtime Analysis

Question. What is the runtime of the following algorithm in big-O-notation:

Algorithm 1

```
Require: Integer n ≥ 1
1: x ← 0
2: for i = 1 ... n do
3:     for j = i ... n do
4:         x ← x + i · j
5:     end for
6: end for
7: return x
```

Solution. We need to sum up the runtimes of all the instructions of Algorithm 1. We account a runtime of $O(1)$ for each of the instructions in Lines 1,4,7, however, the two nested loops make Line 4 being executed multiple times. The runtime of the two nested loops, which dominates the overall runtime of the algorithm, can be computed as follows:

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=i}^{n} O(1) &= O\left(\sum_{i=1}^{n}\sum_{j=i}^{n} 1\right) = O\left(\sum_{i=1}^{n} n-i+1\right) = O\left(\sum_{i=1}^{n}(n+1) - \sum_{i=1}^{n} i\right) \\ &= O\left(n(n+1) - \frac{n(n+1)}{2}\right) = O\left(\frac{n(n+1)}{2}\right) = O(\frac{1}{2}n^2 + \frac{1}{2}n) = O(n^2)\ . \end{aligned}$$

The runtime of Algorithm 1 is therefore $O(n^2)$.

Remark: In the previous calculation, we used the simplification $\sum_{j=i}^{n} 1 = n - i + 1$. Observe that j takes on the values $\{i, i+1, \dots, n\}$, and, for each value, we have a contribution of 1 to the overall sum. Since $|\{i, i+1, \dots, n\}| = n - i + 1$, i.e., j takes on $n - i + 1$ different values, we obtain the result. We also used the identity $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$, which is an important identity that you should remember. In the last step, we used a lemma discussed in the lecture that states that a polynomial in n with constant maximum degree k is in $O(n^k)$. ✓

1 Θ and Ω

1. Prove that the following two statements are equivalent:

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

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

2. Prove that the following two statements are equivalent:

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

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

3. Let $c > 1$ be a constant. Prove or disprove the following statements:

   (a) $\log_c n \in \Theta(\log n)$.

   (b) $\log(n^c) \in \Theta(\log n)$.

4. Let $c > 2$ be a constant. Prove or disprove the following statement:

$$2^n \in \Theta(c^n) \ .$$

2 O-notation

1. Consider the following 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. Give functions f, g such that $f(n) \in O(g(n))$ and $2^{f(n)} \notin O(2^{g(n)})$.

3 Runtime Analysis

Algorithm 2

```
Require: Int n ≥ 1
x ← 0
for i = 1 . . . n do
    for j = 1 . . . n do
        for k = 1 . . . n do
            x ← x + i · j
        end for
    end for
end for
return x
```

Algorithm 3

```
Require: Int n ≥ 1
x ← 0
i ← 1
while i ≤ n do
    for j = 1 . . . n do
        x ← x + i · j
    end for
    i ← 2 · i
end while
return x
```

Algorithm 4

```
Require: Int n ≥ 1
x ← 0
i ← 1
while i ≤ n do
    for j = 1 . . . i do
        x ← x + i · j
    end for
    i ← 2 · i
end while
return x
```

Determine the runtimes of Algorithms 2, 3, and 4 using big-O-notation.

4 Optional and Difficult Questions

Exercises in this section are intentionally more difficult and are there to challenge yourself.

4.1 Peak Finding in 2D (hard!)

Let A be an n-by-n matrix of integers, for some integer n. We say that $A_{i,j}$ is a *peak* if the adjacent elements $A_{i-1,j}, A_{i+1,j}, A_{i,j-1}, A_{i,j+1}$ are not larger than $A_{i,j}$. The objective is to find a peak in A. Similar to the peak finding problem discussed in the lecture, reporting any peak is fine, in particular, it is not required that we find the maximum in A or that we report all the peaks in A.

Consider the following baseline algorithm: We scan the entire matrix and check whether every element $A_{i,j}$, for $i, j \in \{0, 1, 2, \ldots, n-1\}$, is a peak. This strategy requires a runtime of $O(n^2)$. Is there a faster algorithm?

Please send your ideas to `christian.konrad@bristol.ac.uk`. I am keen to hear if you found a solution!