Advanced information, computation, communication I
EPFL - Fall semester 2021-2022

Week 7 - Solution

November 5, 2021

1 Open Questions

Exercise 1. (**) Recall the exercise from last week where we used Bubble Sort, Selection Sort and Insertion Sort to sort the following sequence:

```
9, 12, -43, 20, -2, 3, 7, 28, 19
```

The pseudocode for all three is provided below

Algorithm 1 Bubble Sort

> **for** $i \leftarrow 1$ **to** $n-1$ **do**
>   **for** $j \leftarrow 1$ **to** $n-i$ **do**
>   **if** $a_j > a_{j+1}$ **then**
>   swap a_j and a_{j+1}

Algorithm 2 Selection Sort

> **for** $i \leftarrow 1$ **to** $n-1$ **do**
> $\min \leftarrow i+1$
>   **for** $j \leftarrow i+1$ **to** n **do**
>   **if** $a_{\min} > a_j$ **then**
> $\min \leftarrow j$
>   **if** $a_i > a_{\min}$ **then**
>   swap a_i and $a_{\min}$

Algorithm 3 Insertion Sort

> **for** $j \leftarrow 2$ **to** n **do**
> $i \leftarrow 1$
>   **while** $i < j$ and $a_j > a_i$ **do**
> $i \leftarrow i+1$
> $m \leftarrow a_j$
>   **for** $k \leftarrow 0$ **to** $j-i-1$ **do**
> $a_{j-k} \leftarrow a_{j-k-1}$
> $a_i \leftarrow m$

1. How many comparisons are done in each of the algorithms?
   Solution.
   (a) **Bubble sort**:
   The basic version of the algorithm performs $\frac{n(n-1)}{2} = 36$ comparisons. In the optimized version of the algorithm, number of comparisons can be reduced to 26.
   (b) **Selection sort**:
   The algorithm always performs $n-1$ comparisons at the first round and does one less per round, hence $\sum_{i=1}^{9-1} i = 36$ comparisons.
   (c) **Insertion sort**:
   The algorithm needs 20 comparisons for the given list of values.
2. How many swaps are done in each of the algorithms?
   Solution.
   (a) **Bubble sort**:
   The algorithm performs a swap each time an element is bigger than its neighbor, hence, in our case, $5+5+3 = 13$ swaps.
   (b) **Selection sort**:
   The algorithm performs a swap each time a lower element is found per round, hence, in our case, 7 swaps.
   (c) **Insertion sort**:
   The algorithm does not swap but inserts the lowest element at the end of the sorted sublist, which requires, in our case, $3+4+4+4+3 = 18$ insertions.

3. What is the approximate overall cost of the three algorithms for an input sequence of length $n+1$?
Solution.

Algorithm	Comp.	Swaps
Bubble sort	$O(n^2)$	$O(n^2)$
Selection sort	$O(n^2)$	$O(n)$
Insertion sort	$O(n^2)$	$O(n^2)$

Exercise 2. (*)

1. Show that $5x$ is $o(x^2)$.
Solution.
$$\lim_{x\to\infty} \frac{5x}{x^2} = \lim_{x\to\infty} \frac{5}{x} = 0$$

2. Show that $2x^2$ is not $o(x^2)$.
Solution.
$$\lim_{x\to\infty} \frac{2x^2}{x^2} = \lim_{x\to\infty} \frac{2}{1} = 2 \neq 0$$

3. Show that $1/x$ is $o(x)$.
Solution.
$$\lim_{x\to\infty} \frac{\frac{1}{x}}{x} = \lim_{x\to\infty} \frac{1}{x^2} = 0$$

Exercise 3. (**) Let f be arbitrary functions from $\mathbf{N}$ to $\mathbf{R}_{>0}$.

Let g_1, g_2 be two functions from $\mathbf{N}$ to $\mathbf{R}_{>0}$ such that g_1 and g_2 are both $\Theta(f)$.

1. Show that the function $g_1 + g_2$ is $\Theta(f)$ or provide a counterexample.
Solution. $g_1 + g_2$ is $\Theta(f)$
The functions g_1, g_2 are $\Theta(f)$, i.e. there exist $c_1, d_1, c_2, d_2 > 0, k_1, k_2 > 0$, s.t.
$$\forall x > k_1 \quad c_1 f(x) \leq g_1(x) \leq d_1 f(x)$$
$$\forall x > k_2 \quad c_2 f(x) \leq g_2(x) \leq d_2 f(x)$$
Note that we dropped all absolute value signs because the co-domain is $\mathbf{R}_{>0}$.
Let $k = \max\{k_1, k_2\}$. Then, for all $x > k$,
$$(c_1 + c_2) f(x) \leq g_1(x) + g_2(x) \leq (d_1 + d_2) f(x)$$
which implies that $g_1 + g_2$ is $\Theta(f)$.

2. Show that the function $g_1 g_2$ is $\Theta(f^2)$ or provide a counterexample.
Solution. $g_1 g_2$ is $\Theta(f^2)$
The same notation used as the previous question. For all $x > k$,
$$c_1 c_2 \cdot f^2(x) \leq g_1(x) g_2(x) \leq d_1 d_2 \cdot f^2(x)$$
which implies that $g_1 g_2$ is $\Theta(f^2)$.

Let g_3, g_4 be two functions from $\mathbf{N}$ to $\mathbf{R}$ such that g_3 and g_4 are both $\Theta(f)$.

3. Show that the function $g_3 + g_4$ is $\Theta(f)$ or provide a counterexample.
Solution. $g_3 + g_4$ is **not** $\Theta(f)$
This time we can no longer omit the absolute value symbols, since the co-domain is $\mathbf{R}$.
Simply take $g_3 = -g_4$. Then $g_3 + g_4 \equiv 0$. Shouldn't be too hard to find f such that 0 is **not** $\Omega(f)$.

4. Show that the function $g_3 g_4$ is $\Theta(f^2)$ or provide a counterexample.
Solution. $g_3 g_4$ is $\Theta(f^2)$
Because $|f^2| = |f| \cdot |f|$, $|g_3 g_4| = |g_3| \cdot |g_4|$, the reasoning in Exercise 2.2 will still hold, in spite of the absolute value.

Let g be a function from $\mathbf{N}$ to $\mathbf{R}_{>0}$ such that g is $O(f)$.

5. Show that 2^g is $O(2^f)$, or provide a counterexample.
Solution. 2^g is **not** $O(2^f)$
Take $g(x) = 2x$ and $f(x) = x$. Then $2^f = 2^x, 2^g = 2^{2x} = (2^f)^2$
2^g is not $O(2^f)$ for the same reason why x^2 is not $O(x)$

Exercise 4. (*) What is the largest n for which one can solve within a minute using an algorithm that requires $f(n)$ bit operations, where each bit operation is carried out in 10^{-12} seconds, with these functions $f(n)$?

a. $\log n$
Solution. Each bit operation is carried out in 10^{-12} seconds: $T = 10^{-12}$ seconds.
The algorithm can take at most 1 minute which contains 60 seconds, while there are $\frac{t}{T} = \frac{60}{10^{-12}} = 60 \times 10^{12}$ possible bit operations in 60 seconds.
Algorithm requires $f(n) = log\ n$ bit operations:

$$\log n = 60 \times 10^{12}$$

Note: The logarithm has base 2, because bits only have 2 possible values.

$$\log_2 n = 60 \times 10^{12}$$

Let us take the exponential with base 2 of each side of the previous equation:

$$n = 2^{60 \times 10^{12}}$$

b. $1,000,000n$
Solution. Each bit operation is carried out in 10^{-12} seconds: $T = 10^{-12}$ seconds.
The algorithm can take at most 1 minute which contains 60 seconds, while there are $\frac{t}{T} = \frac{60}{10^{-12}} = 60 \times 10^{12}$ possible bit operations in 60 seconds.
Algorithm requires $f(n) = 1,000,000n$ bit operations:

$$1,000,000n = 60 \times 10^{12}$$
$$n = 60 \times 10^6 = 60,000,000$$

c. n^2
Solution. Each bit operation is carried out in 10^{-12} seconds: $T = 10^{-12}$ seconds.
The algorithm can take at most 1 minute which contains 60 seconds, while there are $\frac{t}{T} = \frac{60}{10^{-12}} = 60 \times 10^{12}$ possible bit operations in 60 seconds.
Algorithm requires $f(n) = n^2$ bit operations:

$$n^2 = 60 \times 10^{12}$$

Take the square root of each side of the previous equation:

$$n = \sqrt{60 \times 10^{12}} \approx 7.745967 \times 10^6 = 7{,}745{,}967$$

2 Exam Questions

Exercise 5. (***) How many comparisons among list elements does insertion sort perform when sorting the following list of length $2n$, $n \geq 1$, in ascending order:

$$2n-1, 2n-3, \ldots, 3, 1, 2n, 2n-2, \ldots, 4, 2$$

- ○ $\frac{1}{2}(n^2+3n-2)$
- ✓ $\frac{1}{2}(n^2+5n-4)$
- ○ $\frac{1}{2}(n^2+7n-6)$
- ○ $\frac{1}{2}(n^2+n)$

Solution. $2n-1$ causes zero comparison. $2n-3$ causes 1 comparision(with $2n-1$); $2n-5$ causes 1 comparision(with $2n-3$); $2n-7$ causes 1 comparision(with $2n-5$); ...; 3 causes 1 comparision(with 5); 1 causes 1 comparision(with 3). After $n-1$ comparisions, the list becomes

$$1, 3, \ldots, 2n-3, 2n-1, 2n, 2n-2, \ldots, 4, 2$$

$2n$ causes n comparisions, with $[1, 3, \ldots, 2n-1]$. Note that $2n$ does not compare with itself(would not be a reasonable algorithm design anyway). $2n-2$ causes n comparisions, with $[1, 3, \ldots, 2n-3, 2n-1]$; $2n-4$ causes $n-1$ comparisions, with $[1, 3, \ldots, 2n-3]$; $2n-6$ causes $n-2$ comparisions, with $[1, 3, \ldots, 2n-5]$; ...; 4 causes 3 comparisions, with $[1, 3, 5]$; 2 causes 2 comparisions, with $[1, 3]$.

If we sum up the number of comparisions, it would be

$$(n-1) + n + [n + (n-1) + \cdots + 2] = 2n - 1 + \frac{1}{2}(n-1)(n+2) = \frac{1}{2}(n^2+5n-4)$$

Exercise 6. (***) Which of the following functions has the fastest growth when n goes to infinity?

- ○ $2^{(\log_2(\log_2 n))^2}$
- ✓ $(\log_2 n)^{2(\log_2 n)^2}$
- ○ $(\log_2(n^2))^{\log_2(n^2)}$
- ○ $n^{\log_2(\log_2 n)}$

Solution. We can simply the expressions by taking log. For simplicity, we omit the base 2 in logarithm. Denote the four expressions as A, B, C, D

$$\begin{aligned}
\log A &= (\log\log n)^2 = \log\log n \cdot \log\log n \\
\log B &= 2(\log n)^2 \cdot \log\log n \\
\log C &= (2\log n) \cdot (1 + \log\log n) = 2\log n + 2\log n \cdot \log\log n \\
\log D &= \log n \cdot \log\log n
\end{aligned}$$

Divide both sides by $\log\log n$,

$$\begin{aligned}
\log A/\log\log n &= \log\log n \\
\log B/\log\log n &= 2(\log n)^2 \\
\log C/\log\log n &= \frac{2\log n}{\log\log n} + 2\log n \\
\log D/\log\log n &= \log n
\end{aligned}$$

After this, it shouldn't be hard to see that B grows the fastest.

Exercise 7. (**) Which function below grows fastest when n goes to infinity?

- ✓ $(\log_3(33))^{n-3}$
- ◯ 3^n
- ◯ $n^{3\log_3(n)}$
- ◯ $n^3\log_3(n)$

Solution. Between $(\log_3(33))^{n-3}$ and 3^n

$$3 = \log_3(3^3) = \log_3(27) < \log_3(33) \implies (\log_3(33))^{n-3} \text{ grows faster than } 3^n$$

Between 3^n and $n^{3\log_3(n)}$

$$n^{3\log_3 n} = (3^{\log_3 n})^{3\log_3 n} = 3^{3(\log_3 n)^2}$$

$$n \text{ grows faster than } 3(\log_3 n)^2 \implies 3^n \text{ grows faster than } 3^{3(\log_3 n)^2}$$

Lastly, $n^3\log n$ is obviously slower than 3^n.

Exercise 8. (**) Consider the two statements below, where k and ℓ are constants with $k > \ell \geq 2$ and $m \to \infty$:

$$\log_m(k) \text{ is } \Theta(\log_m(\ell)) \qquad k^{\log_\ell(m)} \text{ is } O(\ell^{\log_k(m)}).$$

- ◯ They are both false.
- ✓ Only the first is true.
- ◯ Only the second is true.
- ◯ They are both true.

Solution. Because $\log_m(x) = \frac{\log_2(x)}{\log_2(m)}$ both $\log_m(k)$ and $\log_m(\ell)$ are of order $\frac{1}{\log_2(m)}$; in particular $\log_m(k)$ is $\Theta(\log_m(\ell))$.

Writing $k = \ell^{\log_\ell(k)}$ and $\log_k(m) = \frac{\log_\ell(m)}{\log_\ell(k)}$ the comparison for the second problem is between $k^{\log_\ell(m)} = \ell^{\log_\ell(k)\log_\ell(m)}$ and $\ell^{\log_k(m)} = \ell^{\log_\ell(m)/\log_\ell(k)}$. Because $k > \ell \geq 2$ we find that $k^{\log_\ell(m)} = \ell^{c\log_\ell(m)} = m^c$ for the constant $c = \log_\ell(k) > 1$ and that $\ell^{\log_k(m)} = \ell^{(\log_\ell(m))/c} = m^{1/c}$. It follows that $k^{\log_\ell(m)}$ is not $O(\ell^{\log_k(m)})$.

Exercise 9. (*) Consider the following two statements:

$$f \text{ is } o(f) \qquad\qquad f \text{ is } o(g) \text{ implies } f \text{ is } O(g).$$

- ✓ Only the second is true.
- ◯ They are both false.
- ◯ Only the first is true.
- ◯ They are both true.

Solution. The statement "f if $o(f)$" would imply that for any function f it is the case that $\lim_{x\to\infty}\frac{|f(x)|}{|f(x)|} = 0$. That is cleary incorrect: for instance for the function $f(x) = 1$ it is the case that $\lim_{x\to\infty}\frac{|f(x)|}{|f(x)|} = \frac{1}{1} = 1$. Thus the first statement is not correct.

The statement "f is $o(g)$" implies that $\lim_{x\to\infty}\frac{|f(x)|}{|g(x)|} = 0$, and thus that for any $\epsilon > 0$ there exists an x_0 such that $\frac{|f(x)|}{|g(x)|} < \epsilon$ for all $x > x_0$, implying that $|f(x)| < \epsilon|g(x)|$ for all $x > x_0$, which in turn implies that f is $O(g)$. Thus the second statement is correct.

Exercise 10. (**) Given the two statements below, where $d > 0$ is an integer constant and a_i are strictly positive integers for all $i \in \mathbf{Z}$, with $\max_{i\in\mathbf{Z}} a_i = D$, $\min_{i\in\mathbf{Z}} a_i = M$ for $D > 0, M > 0$,

$$\sum_{i=0}^{n} a_i i^d \text{ is } \Theta(n^{d+1}) \qquad\qquad \sum_{i=0}^{d} a_i n^i \text{ is } \Theta(n^d)$$

✓ They are both true.

○ Only the first is true.

○ Only the second is true.

○ They are both false.

Solution. For the first, we know

$$\sum_{i=0}^{n} a_i i^d = a_0 0^d + a_1 1^d + ... + a_{n-1}(n-1)^d + a_n n^d \leq D \cdot n \cdot n^d = Dn^{d+1}$$

Let's assume for simplicity that n is even. For the more general case, simply change $\frac{n}{2}$ into $\lfloor \frac{n}{2} \rfloor$.

$$\begin{aligned}\sum_{i=0}^{n} a_i i^d &= a_0 0^d + a_1 1^d + ... + a_{n-1}(n-1)^d + a_n n^d \\ &\geq a_{n/2}(n/2)^d + a_{n/2+1}(n/2+1)^d + ... + +a_{n-1}(n-1)^d + a_n n^d \\ &\geq M \cdot \left[(n/2)^d + (n/2+1)^d + ... + (n-1)^d + n^d\right] \\ &\geq M \cdot (\frac{n}{2}) \cdot (\frac{n}{2})^d = \frac{M}{2^{d+1}} \cdot n^{d+1}\end{aligned}$$

The second is just a polynomial of degree d and thus behaves like its highest order term.