ASSIGNMENT 1: COMPLEXITY ANALYSIS

due date: Sunday, 27 February 2022 (23.59 WITA)

Aturan pengerjaan tugas:

1. Tugas boleh diketik/ditulis tangan (pastikan bisa dibaca), boleh menggunakan Bahasa Indonesia/Inggris. Hindari menggunakan tinta merah.
2. Tugas dikumpulkan dalam format pdf. Jika menggunakan tulis tangan, harap discan (tidak difoto), kemudian dikompresi untuk memperkecil ukuran file.
3. Format penamaan tugas: **DAA01_Kelas_Nama Lengkap_NIM**.
   Contoh: **DAA01_6A_Gede Ganesha_1610101001**
4. Pengumpulan tugas melalui e-learning Undiksha.
5. Anda diizinkan untuk berdiskusi dengan rekan Anda. Namun Anda harus menuliskan/menjelaskan jawaban Anda sendiri, dan paham dengan baik apa yang Anda tulis. Anda siap bertanggung jawab terhadap hasil pekerjaan Anda. Hasil pekerjaan yang memiliki kemiripan yang tinggi dengan pekerjaan mahasiswa lain mempengaruhi poin penilain.
6. Tugas dinilai berdasarkan kejelasan serta kesesuaian jawaban/penjelasan dengan pertanyaan yang diajukan. Total nilai maksimum tugas ini adalah 115. Keterlambatan dalam pengumpulan tugas mengurangi poin penilaian.

Dengan ini, Anda menyatakan bahwa Anda siap menerima segala konsekuensi jika nantinya ditemukan adanya kecurangan dalam pengerjaan tugas ini.

Problems

1. (*Computing time complexity*, **15 point**)

    1. What is the time complexity of the "naive gcd algorithm" and the "Euclidean algorithm" explained in the lecture? Justify your answer!

Algorithm 1 Naive gcd algorithm of two integers

```
1: procedure GCD(m, n)
2:     r = 1
3:     x = min(m, n)
4:     for i = 1 to x do
5:         if a MOD i == 0 and b MOD i == 0 then r = i
6:         end if
7:     end for
8: end procedure
```

Algorithm 2 Euclidean algorithm

```
1: procedure EUCLIDGCD(m, n)
2:     while b ≠ 0 do
3:         r = a MOD b
4:         a = b
5:         b = r
6:     end while
7:     return a
8: end procedure
```

    2. Compute the best-case, worst-case, and average-case complexities of the following algorithm!

Algorithm 3 Sequential search

```
1: procedure SEQSEARCH(A[1..n], x)
2:     found ← False
3:     i ← 1
4:     while (not found) and (i ≤ N) do
5:         if (A[i] = x) then found ← True
6:         else i ← i + 1
7:         end if
8:     end while
9:     if (found) then index ← i
10:    else index ← 0
11:    end if
12: end procedure
```

Solution:

1. *Complexity of gcd algorithm:*
    - *The naive algorithm has time complexity of* $\mathcal{O}(\min(m, n))$.
    - EUCLIDGCD *algorithm has time complexity of* $\mathcal{O}(n)$ *???*
    - EUCLIDGC2D *algorithm has time complexity of* $\mathcal{O}(\log(\max(m, n)))$

2. *Complexity of sequential search:*
    - *Best case is when* $x = A[1]$, *i.e.* $T_{\min}(n) = 1$

- *Worst case is when $x = A[n]$ or x not found, i.e. $T_{\max}(n) = n$*
- *Average case can be computed as follows:*
  If $x = A[j]$, the time complexity is $T(j) = j$. So:

$$T_{avg}(n) = \frac{1}{n}\sum_{j=1}^{n} T(j) = \frac{1+2+\cdots+n}{n} = \frac{1/2 \cdot n(n+1)}{n} = \frac{n+1}{2}$$

2. (*Dominant terms in a function*, **20 point**)

   Select the dominant term(s) and specify the lowest $\mathcal{O}$-complexity of each algorithm! Give a short proof or an explanation for each function to justify your answer!

	Expression	**Dominant term(s)**	$\mathcal{O}(\cdot)$
1.	$5 + 0.001n^3 + 0.025n$		
2.	$500n + 100n^{1.5} + 50n \log_{10} n$		
3.	$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$		
4.	$n^2 \log_2 n + n(\log_2 n)^2$		
5.	$n \log_3 n + n \log_2 n$		
6.	$3 \log_8 n + \log_2 \log_2 \log_2 n$		
7.	$2n + n^{0.5} + 0.5n^{1.25}$		
8.	$0.01n \log_2 n + n(\log_2 n)^2$		
9.	$100n \log_3 n + n^3 + 100n$		
10.	$0.003 \log_4 n + \log_2 \log_2 n$		

Solution:

Expression	***Dominant term(s)***	$\mathcal{O}(\cdot)$
$5 + 0.001n^3 + 0.025n$	$0.001n^3$	$\mathcal{O}(n^3)$
$500n + 100n^{1.5} + 50n \log_{10} n$	$100n^{1.5}$	$\mathcal{O}(n^{1.5})$
$0.3n + 5n^{1.5} + 2.5 \cdot n^{1.75}$	$2.5 \cdot n^{1.75}$	$\mathcal{O}(n^{1.75})$
$n^2 \log_2 n + n(\log_2 n)^2$	$n^2 \log_2 n$	$\mathcal{O}(n^2 \log n)$
$n \log_3 n + n \log_2 n$	$n \log_3 n, n \log_2 n$	$\mathcal{O}(n \log n)$
$3 \log_8 n + \log_2 \log_2 \log_2 n$	$3 \log_8 n$	$\mathcal{O}(\log n)$
$2n + n^{0.5} + 0.5n^{1.25}$	$0.5n^{1.25}$	$\mathcal{O}(n^{1.25})$
$0.01n \log_2 n + n(\log_2 n)^2$	$n(\log_2 n)^2$	$\mathcal{O}(n(\log n)^2)$
$100n \log_3 n + n^3 + 100n$	n^3	$\mathcal{O}(n^3)$
$0.003 \log_4 n + \log_2 \log_2 n$	$0.003 \log_4 n$	$\mathcal{O}(\log n)$

3. (*Comparing time complexities of two algorithms*, **20 point**)

   (a) Suatu algoritma memiliki kompleksitas waktu $\mathcal{O}(f(n))$ dan *running time* $T(n) = cf(n)$. Jika algoritma tersebut membutuhkan waktu 10 detik untuk memproses 1000 data, berapakah waktu yang dibutuhkan untuk memproses 100,000 data jika $f(n) = n$? Bagaimana halnya jika $f(n) = n^3$?

   (b) Algoritma **A** dan **B** masing-masing memiliki kompleksitas waktu $T_A(n) = 5n \log 10n$ and $T_B(n) = 25n$ microdetik, jika diberikan input dengan ukuran n. Dalam aturan Big-O, algoritma manakah yang bekerja lebih baik? Berikan batasan nilai n agar algoritma **B** memiliki kompleksitas waktu yang lebih baik!

   (c) Suatu perangkat lunak **A** dan **B** masing-masing memiliki kompleksitas waktu $\mathcal{O}(n \log n)$ dan $\mathcal{O}(n^2)$. Keduanya memiliki running time $T_A(n) = c_A n \log_{10} n$ dan $T_B(n) = c_B n$ milidetik untuk memproses data dengan ukuran n. Selama proses pengujian, waktu rata-rata pemrosesan $n = 10^4$ data dengan perangkat **A** dan **B** masing-masing adalah 100 milidetik dan 500 detik.

Berikan batasan nilai n agar perangkat **A** bekerja lebh cepat dibandingkan perangkat **B**, dan berikan batasan nilai n untuk kondisi sebaliknya. Perangkat mana yang Anda rekomendasikan jika kita harus memproses data dengan ukuran tidak lebih dari $n = 10^9$?

Solution:

(a) Since $T(n) = cf(n)$, the constant factor: $c = \frac{T(n)}{f(n)} = \frac{T(1000)}{f(1000)} = \frac{10}{f(1000)}$. Dengan demikian: $T(n) = \frac{10}{f(1000)} \cdot f(n)$. Sehingga: $T(100,000) = \frac{10}{f(1000)} \cdot f(100,000)$. Untuk $f(n) = n$ diperoleh $T(100,000) = 1000$ ms dan untuk $f(n) = n^3$ diperoleh $T(100,000) = 10^7$.

*(b) In the Big-O sense, the algorithm **B** is better (since $n \log n$ grows faster than n).*
*The algorithm **B** outperforms algorithm **A** if $T_B(n) \leq T_A(n)$, that is, if:*

$$25n \leq 5n \log_{10} n \Leftrightarrow 5n \log_{10} n \geq 5 \Leftrightarrow \log_{10} n \geq 5 \Leftrightarrow n \geq 100,000$$

*(c) In the Big-O sense, the package **A** is better than the package **B** (since n^2 grows faster than $n \log n$). From the given processing time of the two algorithms, we can derive the constant factors:*

$$c_A = \frac{100}{10^4 \log_{10} 10^4} = \frac{1}{400}$$
$$c_B = \frac{500,000}{(10^4)^2} = \frac{1}{200}$$

*The package **A** begins to outperform **B** when $T_A(n) \leq T_B(n)$, that is, when*

$$\frac{n \log_{10} n}{400} \leq \frac{n^2}{200} \Leftrightarrow \log_{10} n \leq 2n$$

*The last inequality holds for $n \geq 1$. Thus, for processing up to 10^9 data, we better choose the package **A**.*

4. (*Formal proof*, **20 point**)

(a) Using the formal definition of Big-O notation, prove the following:

$$T(n) = a_0 + a_1 n + a_2 n^2 + a_3 n^3 \in \mathcal{O}(n^3)$$

Hint: temukan konstanta c dan batasan nilai n_0 sedemikian sehingga $cn^3 \geq T(n)$ untuk $n \geq n_0$.

(b) Let $T_1(n) \in \mathcal{O}(f(n))$ and $T_2(n) \in \mathcal{O}(g(n))$. Prove the following:
(a) $T_1(n) + T_2(n) \in \mathcal{O}(f(n)) + \mathcal{O}(g(n)) \in \mathcal{O}(\max(f(n), g(n)))$
(b) $T_1(n) \cdot T_2(n) \in \mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) \in \mathcal{O}(f(n) \cdot g(n))$
(c) $\mathcal{O}(cf(n)) \in \mathcal{O}(f(n))$, where c is a constant
(d) $f(n) \in \mathcal{O}(f(n))$

Solution:

(a) Note that $T(n) \leq |a_0| + |a_1|n + |a_2|n^2 + |a_3|n^3$. Thus if $n \geq 1$, then $T(n) \leq cn^3$ where $c = |a_0| + |a_1| + |a_2| + |a_3|$, so that $T(n)$ is in $\mathcal{O}(n^3)$.

(b) $T_1(n) \in \mathcal{O}(f(n))$ means that $\exists a > 0$ s.t. $T_1(n) \leq a \cdot f(n)$. Similarly, $T_2(n) \in \mathcal{O}(g(n))$ means that $\exists b > 0$ s.t. $T_2(n) \leq b \cdot g(n)$.

(a) $T_1(n) + T_2(n) \leq a \cdot f(n) + b \cdot g(n) \leq \max(a,b)\max(f(n), g(n) + \max(a,b)\max(f(n), g(n)) = 2\max(a,b)\max(f(n), g(n))$, which means that $T_1(n) + T_2(n) \in \mathcal{O}(\max(f(n), g(n)))$

(b) $T_1(n) \cdot T_2(n) \leq a \cdot f(n) \cdot b \cdot g(n) = ab \cdot (f(n) \cdot g(n))$. Clearly, $T_1(n) \cdot T_2(n) \in \mathcal{O}(f(n) \cdot g(n))$

(c) Let $g(n) \in \mathcal{O}(cf(n))$, we prove that $g(n) \in \mathcal{O}(f(n))$.
$g(n) \in \mathcal{O}(cf(n))$ means $\exists a > 0$ s.t. $g(n)]leqa \cdot cf(n) = ac \cdot f(n)$. Hence, $\mathcal{O}(cf(n)) \in \mathcal{O}(f(n))$.

(d) $f(n) \leq a \cdot f(n)$ for any $a \geq 1$. So, $f(n) \in \mathcal{O}(f(n))$.

5. (*Big-$\mathcal{O}$, Big-Ω, and Big-Θ*, **5 point / question, total 40 point**)

Let $f, g : \mathbb{N}^+ \to \mathbb{N}^+$ be two functions. Prove or invalidate each of the following:

1. If $f(n) \in \mathcal{O}(g(n))$ then $g(n) \in \mathcal{O}(f(n)$
2. $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$ (assuming that it exists)
3. If $f(n) \in \mathcal{O}(g(n)$, then $\log(f(n)) \in \mathcal{O}(\log(g(n))$
4. If $f(n) \in \mathcal{O}(g(n))$ then $2^{f(n)} \in \mathcal{O}(2^{g(n)})$
5. $f(n) \in \mathcal{O}(f^2(n))$
6. If $f(n) \in \mathcal{O}(g(n))$ then $g(n) \in \Omega(f(n))$
7. $f(n) \in \Theta(f(n/2))$
8. If $g(n) \in \mathcal{O}(f(n)$ then $f(n) + g(n) \in \Theta(f(n))$

Solution:

1. *False. Counter-example: $f(n) = n$ and $g(n) = n^2$. We will have $f(n) = O(n^2)$, since this is equivalent to $n \leq an^2$, or $1/n \leq a$, so suffice to choose $a = 1$. However, the converse would require $n^2 \leq an$, or $n \leq a$ to hold true, that is, a cannot be a constant.*
2. *False. Counter-example: $f(n) = 1$, $g(n) = n$. Then $\min(f(n), g(n)) = 1$ whatever the value of n. But clearly, $f(n) + g(n) = 1 + n \neq \mathcal{O}(1)$, so $f(n) + g(n) \neq \mathcal{O}(\min(f(n), g(n)))$ as well.*
3. *True provided $g(n) > 1$ for n large enough. Since we know that $f(n) = \mathcal{O}(g(n))$, there must exist constants $n_0 > 0$ and $a > 0$ such that:*

$$\forall n \geq n_0,\ f(n) \leq a \cdot g(n)$$
$$\log f(n) \leq \log a + \log(g(n))$$

On the other hand, $\log(f(n)) = \mathcal{O}(\log(g(n)))$ means there would exist $b > 0$ such that:

$$\log f(n) \leq b \cdot \log g(n)$$

for n large enough. Can we find a value of b that would keep both inequalities true? Indeed,

$$\log a + \log g(n) \leq b \cdot \log g(n)$$
$$\Leftrightarrow \frac{\log a}{\log(g(n))} + 1 \leq b$$

tells us that b should be chosen as $1 + \frac{\log a}{\inf_n(g(n))}$ to ensure this, and this is possible only if $g(n) > 1$ for n large enough. If g periodically hits 1 as n increases, then the claim is false.

4. *True. Again, $f(n) = \mathcal{O}(g(n))$ means there exists $a > 0$ such that $f(n) \leq ag(n)$ holds true for n large enough. On the other hand, $\log f(n) = \mathcal{O}(\log g(n))$ means we can find b such that:*

$$\log f(n) \leq b \log g(n)$$
$$f(n) \leq g(n)^b$$

holds true for n large enough. To conclude, suffice to see that:

$$ag(n) \leq g(n)^b$$
$$\Leftrightarrow a \leq g(n)^{b-1}$$
$$\Leftrightarrow \log a \leq (b-1)g(n)$$
$$\Leftrightarrow \frac{\log a}{g(n)} + 1 \leq b$$

So, $b > 1 + \log a$ is always a possible choice.

5. *True. Indeed, $f(n) \leq af^2(n)$ is equivalent to $a \geq \frac{1}{f(n)}$, so suffice to choose $a = 1$ to keep the inequality holds.*
6. *True. If $f(n) = \mathcal{O}(g(n))$, then there exist $a > 0$ and $n_0 > 0$ such that:*

$$n \geq n_0 \Rightarrow f(n) \leq ag(n)$$
$$\Leftrightarrow g(n) \geq \frac{1}{a} f(n)$$

which is the very definition of Ω with the rescaling constant chosen as $1/a$.

7. *False. Counter-example:* $f(n) = \exp(n)$. *Then* $\exp(n) \leq a \exp(n/2)$ *implies* $a \geq \exp(n/2)$, *so* $a \to \infty$ *as* $n \to \infty$, *a contradiction to the assumption it has to be constant.*

8. *True. Indeed,*

$$\begin{aligned} f(n) + \mathcal{O}(f(n)) &\leq f(n) + af(n) \\ &= (1+a)f(n) \end{aligned}$$

for some constant $a > 0$ *and* n *large enough. Then, very clearly*

$$(1+a)f(n) \leq bf(n) \Rightarrow b \geq 1 + a$$

shows the validity of the equality in $\mathcal{O}$, *and*

$$(1+a)f(n) \geq cf(n) \Rightarrow c \leq 1 + a$$

that in Ω, *and the result follows.*