SOLUTIONS FOR HOMEWORK 1

JIECAO CHEN

DEFINITIONS FROM THE TEXTBOOK

We suggest to use $O, o, \Omega, \Theta$ as defined in our textbook *Introduction to Algorithms, 3Ed*.

**Definition 0.1 ($\Theta$).** $\Theta(g(n)) = \{f(n)| \exists \text{ positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0\}.$

**Definition 0.2 ($O$).** $O(g(n)) = \{f(n)| \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$

**Definition 0.3 ($\Omega$).** $\Omega(g(n)) = \{f(n)| \exists \text{ positive constants } c \text{ and } n_0 \text{ such that } 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0\}.$

**Definition 0.4 ($o$).** $o(g(n)) = \{f(n)| \text{ for any } c > 0, \exists \text{ a constant } n_0 > 0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0\}.$

The textbook also assumes that above notations are defined in terms of functions whose domains are the set of natural numbers $\mathbb{N}$ because we are dealing with the running time function $T(n)$ and $n$ represents the size of the input. However, above definitions are actually applicable to functions whose domains are the set of real numbers $\mathbb{R}$. **Unless we make explicit requirement, you can choose whichever domain ($\mathbb{R}$ or $\mathbb{N}$) you like.**

One more thing worth to mention, $O(g(n))$ (other notations similar) is a set of functions based on our definition. When we write $f(n) = O(g(n))$, it is just a convention made in the community of computer science, what we really mean here is $f(n) \in O(g(n))$. 
Reminders:
- Solutions provide one possible solution process. In many cases, there are multiple correct processes that will result in the correct final answer.
- Solutions are references that may also contain errors.

**Question 1**

We have neither \( f(n) = O(g(n)) \) nor \( g(n) = O(f(n)) \). Let us show \( f(n) \neq O(g(n)) \) here, the other direction can be proved similarly.

**Claim 0.5.** \( f(n) \neq O(g(n)) \)

*Proof.* Prove by contradiction. Assume \( f(n) = O(g(n)) \), by the definition, there exist constants \( c, n_0 > 0 \) such that \( 0 \leq f(n) \leq cg(n) \) or \( 0 \leq n \leq cn^{1 + \sin n} \) for all \( n \geq n_0 \). It implies \( \theta \)

\[
0 \leq 1 \leq cn^{\sin n} \text{ for all } n \geq n_0.
\]

Can it be true? To show that the answer is No, it suffices to show:

For any \( n_0 > 0 \), we can always pick an \( n \geq n_0 \) such that \( cn^{\sin n} < 1 \).

**Easy Version:** use domain \( \mathbb{R} \). For any given \( n_0 > 0 \), let \( n = 2k\pi - \frac{\pi}{2} \) where \( k \) is a large enough integer to make \( n > \max\{c,n_0\} \). Then \( cn^{\sin n} = \frac{c}{n} < 1 \).

**Hard Version:** use domain \( \mathbb{N} \). First let’s consider the set \( S = \{x \in \mathbb{R}^+ | \sin y \leq -0.5 \text{ for all } y \in (x - \frac{\pi}{3}, x + \frac{\pi}{3})\} \), it is trivial from the graph of \( \sin x \) to see that there will be infinitely many elements in \( S \).

Since each \( (x - \frac{\pi}{3}, x + \frac{\pi}{3}) \) has length \( \frac{2\pi}{3} > 1 \), it must contain some integer, it tells us that there will be infinitely many \( n \in \mathbb{N} \) such that \( \sin n \leq -0.5 \).

Now given \( c,n_0 > 0 \), we can always pick an \( n \) that \( n > c^2, n > n_0 \) and \( \sin n \leq -0.5 \), hence \( cn^{\sin n} \leq cn^{-0.5} < \frac{c}{\sqrt{n}} \leq 1 \) which conflicts Inequality \((0.6)\).

□

**Question 2**

**Claim 0.7.** \( f(n) = \frac{1}{n} = o(1) \).

*Proof.* For any constant \( c > 0 \), choose an \( n_0 > \frac{1}{c} \), for any \( n > n_0 > \frac{1}{c} \), we have \( 0 \leq \frac{1}{n} \leq \frac{1}{1/c} = c \cdot 1 \).

□

**Question 3**

**Claim 0.8.** \( f(n) = \Theta(g(n)) \) does not necessarily imply \( 2f(n) = \Theta(2g(n)) \).

*Proof.* It suffices to prove the claim by showing a counterexample: \( f(n) = \log n, g(n) = 2\log n \), then \( 2f(n) = n \) but \( 2g(n) = n^2 \). Clearly \( n \neq \Theta(n^2) \).

□

**Claim 0.9.** \( f(n) = \Theta(g(n)) \) implies \( f^2(n) = \Theta(g^2(n)) \).

*Proof.* \( f(n) = \Theta(g(n)) \) implies that there exist \( c_1, c_2, n_0 > 0 \) such that \( 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \) for all \( n > n_0 \), which further implies that \( 0 \leq c_1^2 g^2(n) \leq f^2(n) \leq c_2^2 g^2(n) \) for all \( n > n_0 \). The claim follows because \( c_1^2 \) and \( c_2^2 \) are also positive constants.

□
Question 4

Claim 0.10. If \( f(n) = O(g(n)) \) then \( f(n) + g(n) = O(g(n)) \)

Proof. \( f(n) = O(g(n)) \Rightarrow \)

there exist \( c,n_0 > 0 \) such that \( 0 \leq f(n) \leq cg(n) \) for all \( n \geq n_0 \).

Hence \( 0 \leq f(n) + g(n) \leq (c+1)g(n) \) for all \( n \geq n_0 \).

Therefore \( f(n) + g(n) = O(g(n)) \). \( \square \)

Claim 0.11. If \( f(n) = \Omega(g(n)) \), \( f(n) - g(n) \neq \Omega(g(n)) \).

Proof. Here is a counterexample: \( f(n) = g(n) = n \), clearly \( f(n) = \Omega(g(n)) \) but \( f(n) - g(n) = 0 \neq \Omega(n) \). \( \square \)

Question 5

Let \( T(n) \) be the running time of this algorithm and let a function \( f(n) = O(n^2) \).

The statement says that \( T(n) \) is at least \( O(n^2) \). That is \( f(n) = O(T(n)) \), but it does not tell us nothing about the growth rate of \( T(n) \), because by the definition of \( O \)-notation, there exist \( c_1, c_2, n_1, n_2 > 0 \) such that

(0.12) \( 0 \leq f(n) \leq c_1 n^2 \) for all \( n \geq n_1 \).

(0.13) \( 0 \leq f(n) \leq c_2 T(n) \) for all \( n \geq n_2 \).

(0.12) allows us to take \( f(n) = 0 \), substitute 0 for \( f(n) \) in (0.13), \( 0 \leq T(n) \) tells us nothing about the growth rate of \( T(n) \).

If we want to give a lower bound, we can say that “the running time of this algorithm is \( \Omega(n^2) \)”. When we need an upper bound, we can say “the running time of this algorithm is \( O(n^2) \)”. 