CSE 431/531: Algorithm Analysis and Design	Spring 2022

Homework 1

Instructor: Shi Li **Deadline: 2/23/2022**

Your Name: ______________________ Your Student ID: ________________

Problems	1	2	3	Total
Max. Score	20	25	35	80
Your Score				

Problem 1. For each pair of functions f and g in the following table, indicate whether $f = O(g)$, $f = \Omega(g)$ and $f = \Theta(g)$ respectively.

$f(n)$	$g(n)$	O	Ω	Θ
$\log_2 n$	$5\log_2(n^3) + 3$	yes	yes	yes
$10n^2 - n$	$n^2 \log n$	yes	no	no
$n^3 - 4n^2 + 10$	n^2	no	yes	no

Prove $\lceil 10n\sqrt{n} \rceil + \lceil n \log n \rceil = O(n\sqrt{n})$.

We use the following fact: For every $n \geq 25$, we have $\log n \leq \sqrt{n}$.
For every $n \geq 25$, we have

$$\begin{aligned}\lceil 10n\sqrt{n} \rceil + \lceil n \log n \rceil &\leq 10n\sqrt{n} + 1 + n\log n + 1 \\ &\leq 10n\sqrt{n} + \frac{n\sqrt{n}}{2} + n\sqrt{n} + \frac{n\sqrt{n}}{2} \\ &= 12n\sqrt{n}.\end{aligned}$$

So, $\lceil 10n\sqrt{n} \rceil + \lceil n \log n \rceil = O(n\sqrt{n})$.

Problem 2. Consider the following algorithm for sorting an array A of n numbers.

Algorithm 1 Sorting the integer array A, which is of size n

```
for i ← 1 to n − 1 do
    for j ← i + 1 to n do
        if A[i] > A[j] then t ← A[i], A[i] ← A[j], A[j] ← t
```

(2a) What does the pseudo-code "$t \leftarrow A[i], A[i] \leftarrow A[j], A[j] \leftarrow t$" do?

It swaps $A[i]$ and $A[j]$.

(2b) What is the running time of the algorithm? Briefly explain why. Your bound should be tight (that is, "the running time is $O(n^{10})$" is not considered as a correct answer).

The running time of the algorithm is $O(n^2)$. Step 3 will run for $\binom{n}{2} = \frac{n(n-1)}{2} = O(n^2)$ times, and the running time for each time is $O(1)$. So over all the running time of the algorithm is $O(n^2)$.

(2c) Why is the algorithm correct? To answer the question, you just need to describe the property that the array A satisfies after each iteration i of the outer loop.

At the end of iteration i of the outer loop of the algorithm, the first i numbers of A are the i smallest numbers in A, sorted in non-decreasing order. That is, for every $i' \le i$, $A[i']$ is the i'-th smallest number in A.

Problem 3. We are given a directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, using the linked-list representation. You need to design an $O(n + m)$-time algorithm to decide between the following three cases:

(i) there is no topological-ordering for G, in which case your algorithm should output "none",

(ii) there is a unique topological-ordering for G, in which case your algorithm should output "unique", and

(iii) there are at least two different topological orderings for G, in which case your algorithm should output "multiple".

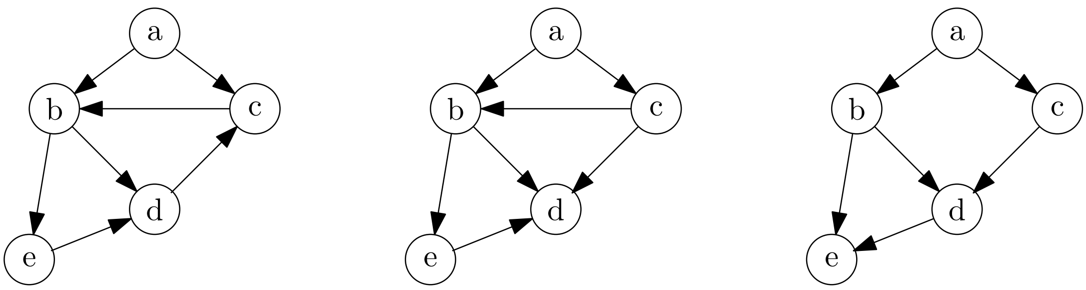

Figure 1: Example input graphs for Problem 3.

For example, consider the three graphs in Figure 1. The outputs for the left-side, middle and right- side graphs are respectively "none", "unique" and "multiple": There is no topological ordering for the left-side graph, there is a unique topological ordering (a, c, b, e, d) for the middle graph, and there are two different topological orderings (a, b, c, d, e) and (a, c, b, d, e) for the right-side graph.

Giving a pseudo-code for your algorithm is sufficient, if the correctness and running time can be easily seen.

Algorithm 2 Algorithm for Problem 3

```
let d_v ← 0 for every v ∈ V
for every edge (u, v) ∈ E do d_v ← d_v + 1
head ← 1, tail ← 0, unqiue ← true
for every v ∈ V do: if d_v = 0 then tail ← tail + 1, queue[tail] ← v
while head ≤ tail do
    if tail > head then unique ← false
    head ← head + 1, v ← queue[head]
    for every out-going edge (v, u) of v do
        d_u ← d_u − 1
        if d_u = 0 then tail ← tail + 1, queue[tail] ← u
if tail < n then return "none"
if unique then return "unique" else return "multiple"
```