CMPSC 465 Data Structures & Algorithms
Fall 2025 Ke Chen and Yana Safonova

Quiz 1

Lecture Section: **Monday, Sep 08, 2025**

Student Name: **PSU Email ID:**

1. (2 pts.) While performing InsertionSort on the array $\{8, 2, 6, 2, 3, 1\}$, which of the following is NOT a transition state of the array:

(a) 2 6 8 2 3 1
(b) 2 2 3 6 8 1
(c) 2 2 8 6 3 1
(d) 2 8 6 2 3 1
(e) None of the above

Answer: (c) For an input array $A[1..n]$, InsertionSort runs from index $k = 1$ to $k = n$ and ensures that after the k-th iteration the first k elements are sorted.

2. (2 pts.) When analyzing an algorithm's running time, we use big-O to denote its worst-case performance and use big-Omega to denote its best-case performance.

(a) True
(b) False

Answer: False Big-O and big-Omega are used to describe upper and lower bounds, respectively, on the growth rate of functions. They are not inherently tied to worst-case or best-case performance. (Though in practice, algorithm analysis typically focuses on worst-case or average-case performance, and best-case performance is rarely considered.)

3. (2 pts.) Let $f(n) = 10n^2$ and $g(n) = n^3 + 5n$. Which of the following is correct?

I. $f(n) = O(g(n))$
II. $g(n) = O(f(n))$

(a) I is correct
(b) II is correct
(c) Both I and II are correct

Answer: (a) The (growth rate of the) cubic function g is not upper bounded by the quadratic function f.

4. (2 pts.) In order to show $2(n+1)^2 = \Omega(n^2)$ by definition, which of the following choices of c and n_0 is NOT valid?

(a) $c = 1.11$ and $n_0 = 111$
(b) $c = 2.42$ and $n_0 = 10$
(c) $c = 0.5$ and $n_0 = 16$
(d) $c = 1$ and $n_0 = 1$
(e) $c = 2$ and $n_0 = 3$

Answer: (b) By definition of Ω, we need to show that there exist constants $c, n_0 > 0$ such that $2(n+1)^2 \geq c \cdot n^2$ for all $n \geq n_0$. But $2(n+1)^2 \geq 2.42n^2$ only holds for small n (≤ 10).

5. (2 pts.) Which of the following statement regarding MergeSort is correct?

(a) MergeSort always divides the input array into two equal halves
(b) MergeSort does not allow duplicate elements in the input
(c) MergeSort always runs in $O(n \log n)$ time even if the input array is split arbitrarily at each step
(d) MergeSort is a divide-and-conquer algorithm where all the sorting happens at the merge step
(e) None of the above

Answer: (d) (a) the two halves cannot be equal if the input has an odd length. (b) duplicates are allowed. (c) it becomes a quadratic time algorithm if we only split out a single element at each step.