EL9343 Midterm Exam (Fall, 2016)

Name: **ID:** **Session: Online or In-Person**

March 4, 2020

2.5 hours exam; Write all answers in the white space provided under each question, write on the back of the page if you need more space

1. (**8 points**) **True or False**

(a) **(2 points) T or F** $f(n) + g(n) = O(min\{f(n), g(n)\})$;

(b) **(2 points) T or F** $\log n! = \Omega(\log n^n)$;

(c) **(2 points) T or F** Randomized quick-sort has lower worst-case running time than quick-sort;

(d) **(2 points) T or F** For any array with n elements, one can always find the i-th smallest element within $O(n)$ time.

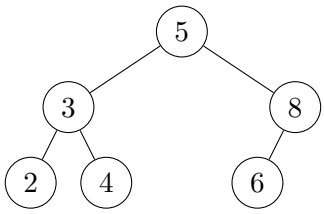
2. (**6 points**) Consider the following sorting algorithms:
A: InsertionSort; B: MergeSort; C: QuickSort; D: HeapSort.

(a) (**2 points**) Which of the above sorting algorithms run in worst-case time $O(n \log n)$?
Circle <u>all</u> that apply: A B C D None

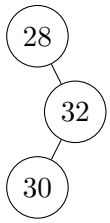
(b) (**2 points**) Which of the above sorting algorithms can run even better than $O(n \log n)$ in the best case?
Circle <u>all</u> that apply: A B C D None

(c) (**2 points**) Which of the above sorting algorithms are stable?
Circle <u>all</u> that apply: A B C D None

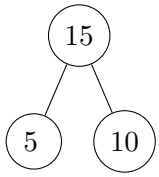
3. (**6 points**) Consider the following trees:



Tree **A**



Tree **B**



Tree **C**

(a) (**2 points**) Which are *binary search trees*?
Circle <u>all</u> that apply: A B C None

(b) (**2 points**) Which are the first three items in a post-order traversal of tree A?
Circle one: 2,3,4 5,3,8 2,3,5 5,3,2 2,4,3 2,4,6 None

(c) (**2 points**) Which are *max heaps*?
Circle <u>all</u> that apply: A B C None

4. (**12 points**) Solve the following recurrences:

(a) (**4 points**) Use the iteration method to solve $T(n) = \frac{1}{2}\left(T(\alpha n) + T(\beta n)\right) + n$, $0.5 < \alpha, \beta < 1$;

(b) (**4 points**) Use the substitution method to verify your solution for Question 4a.

(c) (**4 points**) Solve the recurrence $T(n) = 4T(n/2) + n^{2.5}$ using master method.

5. (**11 points**) For an unsorted array of $[19, 13, 8, 26, 12, 7, 6, 25, 16, 30, 11]$,

(a) (**5 points**) If heap-sort is applied, plot the initial max-heap built from the array

(b) (**3 points**) If quick-sort with Hoare's partition is applied, show the result of the first partition call

(c) (**3 points**) If quick-sort with Lomuto's partition is applied, show the result of the first partition call

6. (**13 points**) Design an iterative algorithm to find the k-th smallest element in an array of n distinct elements (k is a given constant, does not grow with n) with worst-case running time of $O(n)$, you are not allowed to use partition and recursive call,

(a) (**8 points**) write down the pseudo-code of your algorithm

(b) (**5 points**) what is the maximum number of comparisons your algorithm will make (exact number, not just in the asymptotic sense), under what situation it will happen?

7. (**13 points**) For Radix sort:

(a) (**8 points**) prove that it is correct;

(b) (**5 points**) prove or disprove that it is stable.

8. (**11 points**) An array of n distinct keys were inserted into a hash table of size m sequentially over n time slots, suppose chaining was used to resolve collisions, let X_k be the random variable of the number of elements examined when searching for the k-th inserted key (the key inserted in the k-th time slot, $1 \leq k \leq n$),

(a) (**5 points**) what is the probability mass function of X_k, i.e., calculate $P(X_k = i)$, $i \geq 0$?

(b) (**3 points**) what is the expected value of X_k?

(c) (**3 points**) what is the expected number of elements examined when searching for a key randomly selected from the array?

9. (**20 points**) A k-way merge operation. Suppose you have k sorted arrays, each with n elements, and you want to combine them into a single sorted array of kn elements.

(a) (**5 points**) Here's one strategy: Using the merge procedure, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of k and n?

(b) (**8 points**) Design and analyze a more efficient algorithm, using divide-and-conquer, that solves this problem in $O(knlogk)$ time.

(c) (**7 points**) Design and analyze another efficient algorithm, using min-heap, that solves this problem in $O(knlogk)$ time.