EL9343 Midterm Exam (2020 Spring)

Name: **ID:**

Session(Circle one): Tuesday, Wednesday, Online

April 1st, 2020

Write all answers on your own blank answer sheets, scan and upload answer sheets to newclasses at the end of exam, keep your answer sheets until the grading is finished

Multiple choice questions may have multiple correct answers. You will get partial credits if you only select a subset of correct answers, and will get zero point if you select one or more wrong answers.

1. (**18 points**) **True or False (3 points each)**
   (a) **T or F:** Bubble-sort always takes $\Theta(n^2)$ time.
   (b) **T or F:** For any array with n elements, one can always find the i-th smallest element within $O(n)$ time.
   (c) **T or F:** Randomized quicksort can avoid the absolute worst case running time of $\Theta(n^2)$;
   (d) **T or F:** The n-th Fibonacci sequence number $Fib(n) = o(2^n)$;
   (e) **T or F:** Counting sort is NOT stable;
   (f) **T or F:** Radix sort can use comparison sort to sort each of the d-digits;

2. (**4 points**) If $f(n) = \Theta(g(n))$, and $g(n) = o(z(n))$, which of the following are true?
   (a) $2^{f(n)} = \Theta(2^{g(n)})$;
   (b) $f(n) = o(z(n))$;
   (c) $f(n) = \Theta(z(n))$;
   (d) $z(n) = \omega(g(n))$;
   (e) None of the above

3. (**4 points**) Which of the following statements regarding Divide-and-Conquer algorithms are true?
   (a) Divide-and-Conquer algorithms always run faster than iterative algorithms;
   (b) The master theorem cannot be used to solve all recurrences;

(c) Heapsort is a divide-and-conquer algorithm;

(d) In the maximum-subarray problem, combining solutions to the sub-problems is more complex than dividing the problem into sub-problems;

(e) None of the above.

4. (**4 points**) Which of the following statements about sorting algorithms are true?

(a) Quicksort runs in best-case time $\Theta(n)$;

(b) The pivot element in Lomuto's partition is not included in any of the two subarrays;

(c) Radix sort is a comparison sort algorithm;

(d) Mergesort runs in worst-case time $\Theta(nlogn)$;

(e) None of the above.

5. (**4 points**) Which of the following statements about a max-heap of size n are true?

(a) Max-heap can be stored in an array of size n;

(b) Build-max-heap procedure takes $\Theta(nlogn)$;

(c) Heapsort runs in worst-case time $O(nlogn)$;

(d) Outputs of heapsort using max-heap are in non-descending order;

(e) None of the above.

6. (**4 points**) When handling collisions using chaining, which of the following statements about a hash table of m slots and stored n elements are true?

(a) Worst-case insertion complexity is always $\Theta(1)$;

(b) Worst-case deletion complexity can be $\Theta(1)$;

(c) To have an average-case search time of $O(1)$, n must be $O(m)$;

(d) In universal hashing, a new hash function is picked randomly each time when inserting;

(e) None of the above

7. (**6 points**) Prove the following properties of asymptotic notation:

(a) (**3 points**) $n^3 = \omega(n^2)$, where $\omega(\cdot)$ stands for little-ω;

(b) (**3 points**) If $f(n) = \Theta(g(n))$, and $h(n) = O(g(n))$, then $f(n) = \Omega(h(n))$.

8. (**12 points**) Solve the following recurrences:

(a) (**4 points**) Use the iteration method to solve $T(n) = T(\frac{n}{4}) + T(\frac{n}{3}) + 2n$;

(b) (**4 points**) Use the substitution method to verify your solution for Question 8a.

(c) (**4 points**) Solve the recurrence $T(n) = 3T(\sqrt{n}) + (\log n)^2$.

Cb) $h(n) \le c_1 g(n)$ $n \ge n_0'$

$g(n) \le c_2 f(n)$ $n \ge n_0''$

$h(n) \le c_1 \cdot c_2 f(n)$

c

$n_0 = \max(n_0', n_0'')$

(a) Recursion tree is asymmetric and inbalanced: the recursions that divide the problem size by 3 reach the bottom the fastest at depth $\log_3 n$, whereas the recursions that divide the problem size by 2 reach the bottom at depth $\log_2 n$. Other recursions that keep dividing the problem size by combinations of 2 and 3 are in the middle. Note that the cost at each depth is reduced by a factor of $1/2 + 1/3 = 5/6$. In other words, the merging cost at depth k is $n(5/6)^k$. Then, we can bound $T(n)$ from above and below by

$$n \sum_{k=0}^{\log_3 n} (5/6)^k \leq T(n) \leq n \sum_{k=0}^{\log_2 n} (5/6)^k.$$

As $n \to \infty$, both the upper and lower bounds tend to $6n$, which implies $T(n) = \Theta(n)$.

(b) If $T(n) = \Theta(n)$, we have $T(n) = O(n)$ and $T(n) = \Omega(n)$. Let's focus on proving $T(n) = O(n)$ first. Assume that this holds for some $c \geq 0$ and for some n_0 such that $n/2 > n/3 > n_0$. Then,

$$T(n) = T(\frac{n}{3}) + T(\frac{n}{2}) + n \leq cn/3 + cn/2 + n = n(1 + 5c/6).$$

If $1 + 5c/6 \leq c$, i.e., if $c \geq 6$, then $T(n) \leq cn$. This means if $T(n/3) \leq cn/3$ holds with some $c \geq 6$, $T(n) \leq cn$ must also hold. The argument is similar for $T(n) = \Omega(n)$, which results in $c \leq 6$.

(c) Let $\log n = m$. Then, $T(2^m) = 3T(2^{m/2}) + m^2$. Let $S(m) \triangleq T(2^m)$. Then, $S(m) = 3S(m/2) + m^2$. Invoke the master theorem with $a = 3$, $b = 2$ and $f(m) = m^2$. Since $f(m) = \Omega(m^{\log_2 3+\epsilon})$ for any $0 < \epsilon < 2 - \log_2 3$ and $af(m/b) = 3m^2/4 \leq cf(m)$ for $c = 3/4$ and all positive m, we have $S(m) = \Theta(m^2)$ and thus $T(n) = \Theta(\log^2 n)$.

9. (**10 points**) For an unsorted array of $[12, 25, 7, 16, 23, 15, 8, 6, 19, 21, 11]$,

(a) (**4 points**) If quick-sort with Hoare's partition is applied, show the result after the first partition call;

(b) (**3 points**) If heap-sort is applied, plot the initial max-heap built from the array;
Call the max-heapify algorithm. Initial max-heap is $[25, 21, 23, 19, 11, 16, 12, 15, 8, 7, 6]$.

(c) (**3 points**) following question (b), plot the max-heap after the third-maximum has been extracted.
Call extract-max 3 times. Resulting heap is $[19, 15, 16, 8, 11, 6, 12, 7]$.

10. (**10 points**) In Randomized Quicksort (with Lomuto's partition) for an array of n distinct numbers,

(a) (**4 points**) after the first call of randomized-partition, what is the maximum and minimum lengths of the left partition, what is the expected length of the left partition?

(b) (**3 points**) after the first call of randomized-partition, what is the probability that the i-th ranked number stays in the same partition as the $(i+k)$-th ranked number?

(c) (**3 points**) what is the probability that the i-th ranked number is never compared against the $(i+k)$-th ranked number throughout the sorting process?

Answer:

b) Randomized Quicksort uses Lomuto's partition, two numbers will be in the same partition if and only if the pivot's rank is either less than i or greater than $i+k$, the probability is $\frac{i-1}{n} + \frac{n-i-k}{n} = \frac{n-k-1}{n}$.

c) The probability that the two numbers will be compared is $\frac{2}{k+1}$, so the probability that they will not be compared is $\frac{k-1}{k+1}$.

a) Min = 0 Max = n-1 Avg = $\frac{n-1}{2}$

11. (**12 points**) We have a hash table with m slots with collisions resolved by chaining. Suppose n distinct keys are inserted into the table. Each key is equally likely to be hashed to each slot.

(a) (**4 points**) Let X_i be the random variable of the number of keys hashed into slot i, what is the probability mass function of X_i, i.e., calculate $P(X_i = k)$, $\forall k \geq 0$?

(b) (**4 points**) Let X be the maximum number of keys in any slot, i.e., $X = \max_{i=1}^{m} X_i$, please show that $P(X = k) \leq mP(X_i = k)$;

(c) (**4 points**) What is the expected number of elements examined when searching for a key randomly selected from the array?

Answer: a). X_i follows the binomial distribution with n-trial and $p = \frac{1}{m}$, so its p.m.f is

$$P(X_i = k) = \binom{n}{k} \frac{1}{m^k} (1 - \frac{1}{m})^{n-k};$$

b). The necessary condition for $\{X = \max_{i=1}^{m} X_i = k\}$ is to have least one $X_i = k$, therefore

$$P(\{\max_{i=1}^{m} X_i = k\}) \leq P(\cup_{i=1}^{m} \{X_i = k\}) \leq \sum_{i=1}^{m} P(X_i = k) = mP(X_i = k);$$

c). The average number of elements examined for a successful search is $1 + \frac{n-1}{2m}$. (see lecture notes).

12. (**12 points**) If there is an algorithm that can find the median of any array of n elements with worst-case performance of $\Theta(n)$, describe how to use this algorithm to develop another algorithm to find the i-th order statistic of any array of n elements with worst-case performance of $\Theta(n)$, justify the correctness and complexity of your algorithm.

Answer: Use the algorithm is find the median, partition the array, if i is the median index, done, if $i > \frac{n}{2}$, recursively work on the right subarray, if $i < \frac{n}{2}$, recursively work on the left subarray,

Recurrence:

$$T(n) = T(\frac{n}{2}) + \Theta(n) \rightarrow T(n) = \Theta(n)$$