Midterm el9343 2023 Fall Solutions

Data Structure and Algorithm (New York University)

ECE GY 9343 Midterm Exam (Fall 2023)

Name: **NetID:**

Answer ALL questions. Exam is closed book. No electronic aids. However, you are permitted two cheat sheets, two sides each sheet. Any content on the cheat sheet is permitted.

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.

Requirements for in-person students ONLY

Please answer all questions on the question book. You should have enough space. Since we scan all the submissions in a batch, DON'T write on the back of the pages and don't tear off any page.

Make sure you write your NetID on the bottom of each page (not your N number).

If you need extra scrape papers, we can give to you. However, it will not be graded.

Requirements for remote students ONLY

Each student is required to open Zoom and turn on their video camera, making sure the camera captures your hands and your computer screen/keyboard. The whole exam will be recorded. To clearly capture the video of exam taking, it is recommended that: A student can use an external webcam connected to the computer, or use another device (smartphone/tablet/laptop with power plugged in). Adjust the position of the camera so that it clearly captures the keyboard, screen and both hands. During the exam, you should keep your video on all the time. If there is anything wrong with your Zoom connection, please reconnect ASAP. If you cannot, please email ASAP, or call your proctor.

During the exam, if you need to use the restroom, please send a message using the chat function in Zoom, so we know you left.

Please use a separate page for each question. Clearly write the question numbers on top of the pages. When you submit, please order your pages by the question numbers.

Once exam is finished, please submit a single PDF on Gradescope, under the assignment named "Midterm Exam". DEADLINE for submission is 15 minutes after the exam ended. Please remember to parse your uploaded PDF file. Before you leave the exam, it's your responsibility to make sure all your answers are uploaded. If you have technical difficulty and cannot upload 10 minutes after exam ended, email a copy to your proctor and the professor.

1. (**20 points**) **True or False**

   (a) **T or F:** The worst-case running time and average-case running time are equal asymptotically (same order) for any randomized algorithms;

   (b) **T or F:** One can find the maximum sub-array of any array with n elements within $O(n \log n)$ time;

   (c) **T or F:** Insertion-sort follows the divide-and-conquer design;

   (d) **T or F:** Bubble-sort always takes $\Theta(n^2)$ time;

   (e) **T or F:** Quicksort with Hoare's partition is stable;

   (f) **T or F:** The array of $[15, 14, 9, 7, 4, 3, 5, -1, 6, 2]$ is a max heap.

   (g) **T or F:** All comparison sorting algorithms have $\Omega(n \log n)$ complexity;

   (h) **T or F**: Pre-order walk of a binary search tree outputs a sorted sequence;

   (i) **T or F**: Merge-sort is stable;

   (j) **T or F:** Randomized Quicksort eliminates the worst-case running time of $\Theta(n^2)$;

2. (**4 points**) If $f(n) = \Omega(g(n))$, and $z(n) = \omega(g(n))$, which of the following can be true?

   (a) $f(n) = O(g(n))$;

   (b) $f(n) = \omega(z(n))$;

   (c) $f(n) = \Theta(z(n))$;

   (d) $g(n) = \Theta(z(n))$;

   (e) $f(n) = o(g(n))$;

3. (**4 points**) Which of the following statements regarding Divide-and-Conquer algorithms are true?

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

   (b) Divide-and-Conquer algorithms always run faster than iterative algorithms;

   (c) In quick-sort, dividing the problem into sub-problems is more complex than combining solutions to the sub-problems;

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

   (e) None of the above.

4. (**4 points**) If you were given a priority queue implemented by a max-heap, and the total elements stored is n, which of the following is true:

   (a) If the key of a single node is decreased, call max-heapify() at that node can restore max-heap property;

   (b) If all keys are distinct, the minimum element's index must be larger than $\lfloor n/2 \rfloor$;

   (c) Find the maximum element takes $O(\log n)$;

   (d) Extract the maximum element takes $O(\log n)$;

   (e) None of the above.

5. (**4 points**) Which of the following sorting algorithms are in-place:

   (a) InsertionSort;

   (b) MergeSort;

   (c) QuickSort;

   (d) HeapSort;

   (e) None of the above

6. (**4 points**) Which of the following about binary search tree are true?

(a) The maximum node is always a leaf node;

(b) when deleting a node z with two children, we can replace z by its predecessor;

(c) when deleting a node z with two children, we can replace z by its successor;

(d) In an AVL tree, the left sub-tree and right sub-tree of any node should have the same height;

(e) the worst-case search time in a binary search tree with n nodes is $O(n)$;

7. (**4 points**) Prove that if $f(n) = o(g(n))$ and $g(n) = o(h(n))$ then $h(n) = \omega(f(n))$, where $o(\cdot)$ stands for little-o and $\omega(\cdot)$ stands for little-ω.

Answer:

$$f(n) = o(g(n)) \implies \forall c_1 > 0,\ \exists n_1 > 0,\ \text{such that } f(n) < c_1 g(n), \forall n \geq n_1$$
$$g(n) = o(h(n)) \implies \forall c_2 > 0,\ \exists n_2 > 0,\ \text{such that } g(n) < c_2 h(n), \forall n \geq n_2$$

To prove $h(n) = \omega(f(n))$, we need to show $\forall C > 0,\ \exists N > 0,\ \text{such that } h(n) > Cf(n), \forall n \geq N$

We choose $c_1 = c_2 = \frac{1}{\sqrt{C}} \implies \frac{1}{c_1 c_2} = C$, and since $f(n) = o(g(n))$ and $g(n) = o(h(n))$, we can find the corresponding n_1 and n_2 in the above statements, and let $N = \max(n_1, n_2)$, and

$$f(n) < c_1 g(n), \forall n \geq N \geq n_1$$
$$g(n) < c_2 h(n), \forall n \geq N \geq n_2$$
$$\therefore \forall n \geq N, h(n) > \frac{1}{c_2} g(n) > \frac{1}{c_1 c_2} f(n) = Cf(n)$$
$$\therefore h(n) = \omega(f(n))$$

8. (**10 points**) Solve the following recurrences:

(a) (**5 points**) Use the iteration method to solve $T(n) = T(\frac{n}{3}) + 2T(\frac{n}{4}) + n^2$;

(b) (**5 points**) Use the substitution method to verify your solution for Question 8a.

Answer:

(a)

$$\begin{aligned}
T(n) &= T(\frac{n}{3}) + 2T(\frac{n}{4}) + n^2 \\
&= T(\frac{n}{9}) + 4T(\frac{n}{12}) + 4T(\frac{n}{16}) + \frac{n^2}{9} + 2\frac{n^2}{16} + n^2 \\
&= T(\frac{n}{9}) + 4T(\frac{n}{12}) + 4T(\frac{n}{16}) + (\frac{17}{72} + 1)n^2 \\
&= T(\frac{n}{27}) + 6T(\frac{n}{36}) + 12T(\frac{n}{48}) + 8T(\frac{n}{64}) + ((\frac{17}{72})^2 + \frac{17}{72} + 1)n^2 \qquad \vdots \\
&= (1 + \frac{17}{72} + (\frac{17}{72})^2 + \dots)n^2 \qquad \text{(Assuming initial condition is negligible, i.e. } T(1) \to 0)
\end{aligned}$$

$$\therefore \quad T_{lower} \geq n^2 \sum_{i=0}^{\log_4 n} (\frac{17}{72})^i, \quad T_{upper} \leq n^2 \sum_{i=0}^{\log_3 n} (\frac{17}{72})^i$$

$$\lim_{n\to\infty} T_{lower} = \lim_{n\to\infty} T_{upper} = \frac{72}{55}n^2 = \Theta(n^2)$$

(b) The upper bound:

IH: $T(k) \leq dk^2,\ \forall k < n$, and we want to show the equation also holds when $k = n$, i.e. $T(n) \leq dn^2$.

Using the recurrence, we have:

$$T(n) = T(\frac{n}{3}) + 2T(\frac{n}{4}) + n^2 \leq d(\frac{n}{3})^2 + 2d(\frac{n}{4})^2 + n^2$$

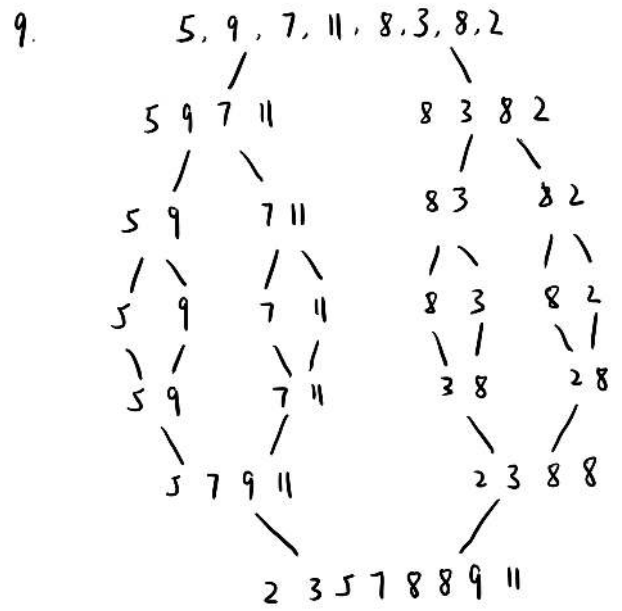
If we find d such that $d(\frac{n}{3})^2 + 2d(\frac{n}{4})^2 + n^2 \leq dn^2$, we can say $T(n) \leq dn^2$ holds.

Solving for the value of d, we can see $d \geq \frac{72}{55}$, and by induction, we know $T(n) = O(n^2)$.

Verifying for the lower bound is similar. The upper bound and lower bound together make the tight bound.

9. (**5 points**) Show the steps of merge-sort on the following array $[5, 9, 7, 11, 8, 3, 8, 2]$

Answer:

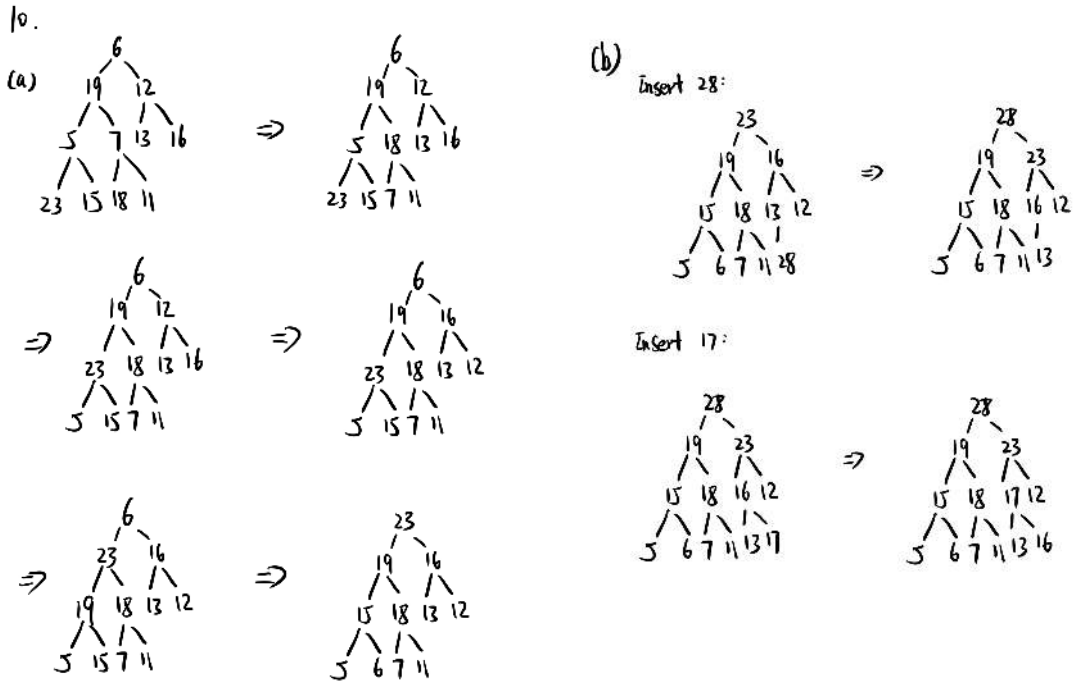


10. (**7 points**) If heap-sort is applied to an array of $[6, 19, 12, 5, 7, 13, 16, 23, 15, 18, 11]$,

(a) (**4 points**) plot the initial max-heap built from the array;

(b) (**3 points**) plot the max-heap after keys 28 and 17 are inserted to the heap sequentially.

Answer:



11. (**7 points**) For the AVL Tree in Figure 1,

(a) (**4 points**) when a key with value of 7 is inserted, how many rotations are needed to restore the AVL tree property? what is the restored AVL tree with the inserted key?

(b) (**3 points**) after step (a), a key with value of 13 is inserted, how many rotations are needed to restore the AVL tree property? what is the restored AVL tree with the inserted key?

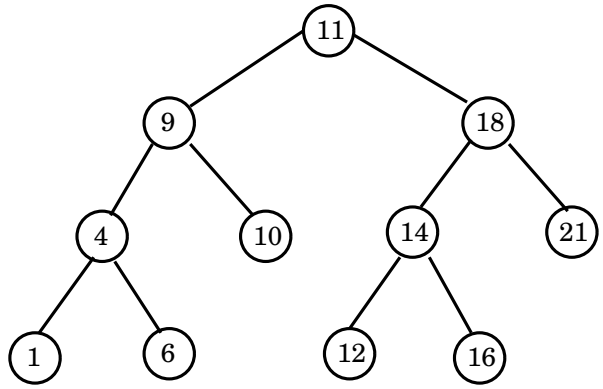
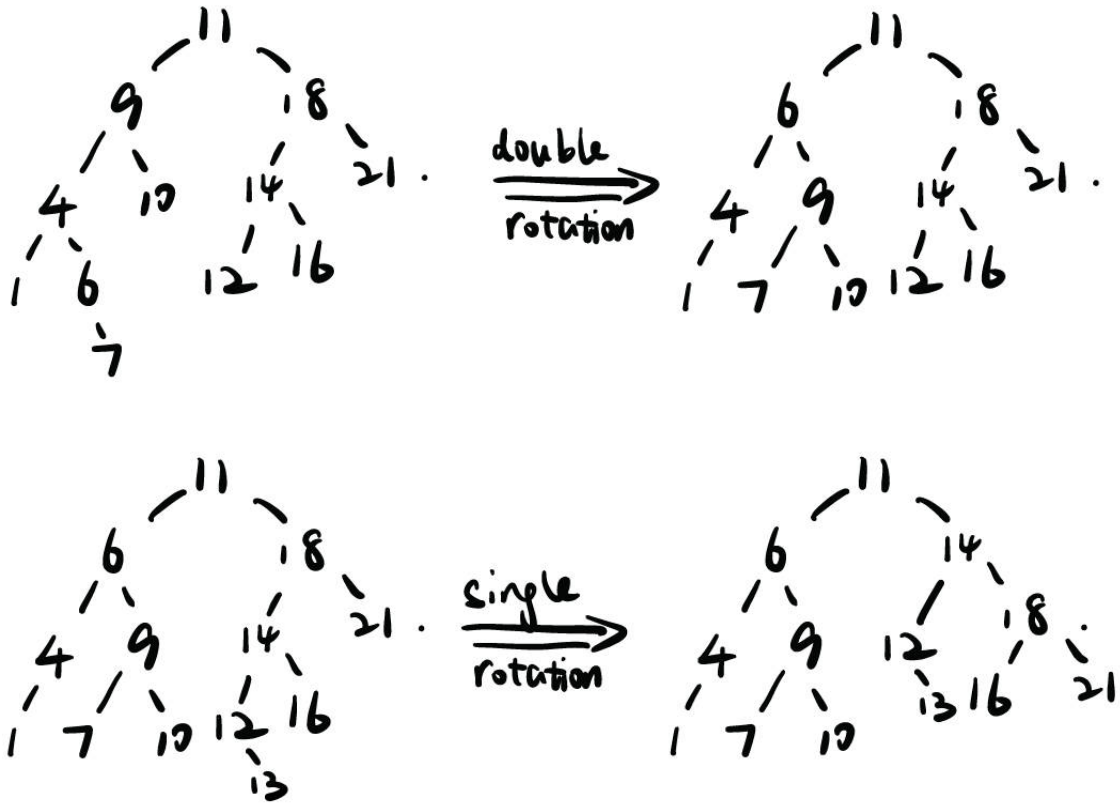


Figure 1: AVL Tree for Question 11

Answer:

Part (a) requires **two** rotations (or **one double rotation**), part (b) requires **one** (single) rotation.



12. (**8 points**) Using the Loop-invariant method to prove the correctness of Hoare's partition.

(a) (**2 points**) write down the Loop Invariant Statement

(b) (**6 points**) Complete the proof by Initialization, Maintenance, and Termination.

```
Alg.: Hoare's PARTITION (A, p, r)
1. x <-- A[p]
2. i <-- p-1
3. j <-- r +1
4. while TRUE
5.      do repeat j <--  j-1
6.            until A[j] <= x
7.      do repeat i <--  i+1
8.            until A[i] >= x
9.       if i < j
10.          then exchange A[i] with  A[j]
11.     else return j
```

Answer:

($A[p, i]$ represents the numbers in sub-array starting at index p and ending at index i)

Loop-invariant (LI): before each loop, $A[p, i] \leq x$ and $A[j, r] \geq x$, where x is the selected pivot.

Initialization: Before the first while loop, $i = p - 1, j = r + 1$, both $A[p, i]$ and $A[j, r]$ are empty, the LI holds.

Maintenance: Assume LI holds before an iteration, during which the algorithm is not returned. We have $A[p, i_{\text{old}}] \leq x$ and $A[j_{\text{old}}, r] \geq x$. During the iteration, j decreases to j_{new} and i increases to i_{new}, and we can see $A[p, i_{\text{new}} - 1] \leq x, A[i_{\text{new}}] \geq x, A[j_{\text{new}} + 1, r] \geq x, A[j_{\text{new}}] \leq x$. After the exchange, $A[i_{\text{new}}] \leq x, A[j_{\text{new}}] \geq x$, and $A[p, i_{\text{new}}] \leq x, A[j_{\text{new}}, r] \geq x$. So the LI still holds after the iteration.

Termination: The loop terminates when $i \geq j$, meaning that the two indices meet. We have $A[p, i - 1] \leq x, A[j + 1, r] \geq x, A[j] \leq x$, and $i \geq j \implies A[p, j - 1] \leq x$, so $A[p, j] \leq x \leq A[j + 1, r]$. Around index j, the array is partitioned into two sub-arrays. All values in the first sub-array is smaller than or equal to the pivot, while all values in the second sub-array is larger than or equal to the pivot. The Hoare's partition is correct.

13. (**7 points**) Answer the following questions about hash table with collision resolved by chaining.

(a) (**3 points**) If the hash table has m slots, and n keys are sequentially inserted into the hash table, the slot of each key is determined by a simple universal hash function $h(\cdot)$ with value range of 1 to m. What is the probability that the number of keys inserted into a given slot is larger than K, $0 < K < n$?

(b) (**4 points**) If the hash table has $2m$ slots, and n keys are sequentially inserted into the hash table, the slot of each key is determined by the sum of two simple universal hash functions $h_1(\cdot) + h_2(\cdot)$, each with value range of 1 to m. What is the probability that a key is insert into the chain stored at hash table slot j $(1 \leq j \leq 2m)$?

Answer: a). Let X be the random variable of the number of keys inserted into any given slot. X follows the binomial distribution with n-trial and $p = \frac{1}{m}$, its p.m.f is

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

So the probability of $X > K$ is

$$P(X > K) = \sum_{k=K+1}^{n} \binom{n}{k} \frac{1}{m^k} (1 - \frac{1}{m})^{n-k};$$

b) Let h_1 and h_2 be the two hash values of a key, the key is placed into the chain at slot j iff $h_1 + h_2 = j$, it happens with probability of

$$P_j = \frac{1}{m^2} \sum_{h_1=1}^{m} I(1 \leq j - h_1 \leq m), \quad 1 \leq j \leq 2m,$$

where $I(\cdot)$ is the indicator function. More specifically,

(a) $P_1 = 0$;

(b) $P_j = \frac{j-1}{m^2}$, for $2 \leq j \leq m + 1$;

(c) $P_j = \frac{2m+1-j}{m^2}$, for $m + 1 < j \leq 2m$

14. (**12 points**) Suppose there are n min-heaps, each min-heap contains m numbers ($1 \leq m \leq n$). Develop an algorithm that selects the i-th ($1 \leq i \leq n$) order statistic from the combined nm numbers from all min-heaps with the complexity of $\Theta(n + i(\log n + \log m))$.

(a) (**4 points**) describe the main steps of your algorithm (no need to write down the detailed pseudo code);

(b) (**4 points**) analyze the complexity of your algorithm;

(c) (**4 points**) we showed that there is a worst-case linear time SELECT algorithm that selects the i-th order statistic from n numbers with $T(n) = \Theta(n)$ comparisons. Briefly describe how you can utilize this algorithm to update your algorithm in (a) to reduce its complexity to $\Theta(n + i(\log i + \log m))$.

Answer a) The root of each min-heap is the minimum of the m numbers in that min-heap. Form another min-heap H with the roots of the n min-heaps, extract the minimum z from H, also extract z from its original min-heap Z, insert the new root y of Z into H, repeat the previous steps for i times to get the i-th smallest of the nm numbers.

b). Get the root of each min-heap takes $\Theta(1)$, a total of $\Theta(n)$. Build the new min-heap H takes $\Theta(n)$ time. The complexity of each of the following i steps: extract min of H takes $\Theta(\log n)$, extract z and y from Z takes $\Theta(\log m)$, finally insert y into H takes $\Theta(\log n)$, so the total complexity is $\Theta(n + i(\log n + \log m))$.

c) Apply the linear time SELECT algorithm on all the roots of the min-heap to find the i-th smallest, let it be x. Then select the i min-heaps whose roots are smaller than or equal to x, the i-th minimum of the combined nm numbers must be in the selected i min-heaps. (*Any element in the non-selected heaps will be larger than the i selected roots, so it cannot be the i-th smallest among all elements.*)

Form another min-heap H with the roots of the i min-heaps, extract the minimum z from H, also extract z from its original min-heap Z, insert the new root y of Z into H, repeat the previous steps for i times to get the i-th smallest of the nm numbers.

Updated Complexity: initial selection to find x: $\Theta(n)$, the complexity of each of the following i steps: extract min of H takes $\Theta(\log i)$, extract z and y from Z takes $\Theta(\log m)$, finally insert y into H takes $\Theta(\log i)$, so the total complexity is $\Theta(n + i(\log i + \log m))$.

If you use up the space under any particular problem, you can write your answer here. On the page of the problem, tell us part of your work is here and write down the page number of this page and. Don't tear off any pages.

If you use up the space under any particular problem, you can write your answer here. On the page of the problem, tell us part of your work is here and write down the page number of this page and. Don't tear off any pages.