EL9343 Midterm Exam (2021 Fall)

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, 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.

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 n-th Fibonacci sequence number $Fib(n) = O(2^n)$;

   (b) **T or F:** To prove $T(n) = o(f(n))$, we need to find $c > 0$ and $n_0 > 0$, such that $T(n) < cf(n)$ when $n > n_0$;

   (c) **T or F:** Randomized quicksort can avoid the absolute worst case running time of $\Theta(n^2)$;

   (d) **T or F:** The complexity of insertion in a binary search tree with n nodes is $O(\log n)$;

   (e) **T or F:** Radix sort can use comparison sort to sort each of the d-digits;

   (f) **T or F:** It is possible to use counting sort on an array of real numbers;

   (g) **T or F:** Tree height of a max heap with n numbers is always $\Theta(\log n)$;

   (h) **T or F:** A pure random number generator can be used as the hash function;

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

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

2. (**4 points**) Which of the following function pairs satisfy $f(n) = \Omega(g(n))$?

   (a) $f(n) = n$ and $g(n) = \sqrt{n}$;

   (b) $f(n) = n \log n$ and $g(n) = 3n + \log n$;

   (c) $f(n) = 2n$ and $g(n) = \log n$;

   (d) $f(n) = n^k$ and $g(n) = k^n$, where $k > 1$ is a constant;

   (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**) For a given node v in a binary search tree, which of the following node can be its successor?

   (a) right child of v's left child;

   (b) left child of v's right child;

   (c) v's right child;

   (d) parent of v's parent node;

   (e) None of the above

5. (**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) The linked list at each slot is sorted;

   (c) Worst-case time to search is $\Theta(n)$;

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

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

6. (**4 points**) Which of the following statements about sorting algorithms are true?

   (a) QuickSort runs in best-case time $O(n)$;

   (b) HeapSort runs in worst-case time $O(nlogn)$;

   (c) Radix sort is a comparison sort algorithm;

   (d) MergeSort runs in worst-case time $O(nlogn)$;

   (e) None of the above

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

   (a) (**3 points**) $n = \omega(\sqrt{n})$;

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

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

   (a) (**4 points**) Use the iteration method to solve $T(n) = T(\frac{n}{8}) + T(\frac{n}{3}) + 3n$;

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

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

9. (**6 points**) Illustrate the operation of merge-sort on the array [71, 25, 40, 7, 60, 13, 20, 80]

10. (**6 points**) Given an array of $[4, 11, 6, 12, 16, 3, 9, 5, 8, 15, 7]$,

    (a) (**3 points**) plot the initial max-heap built from the array;

    (b) (**3 points**) plot the max-heap after the maximum is extracted, then a new number 10 is inserted.

7

(a)

$n = \omega(\sqrt{n})$

For any constant $c > 0$, There exist $n_0 = c^2 + 1 \geq 0$, such that $n > c\sqrt{n}$ for all $n \geq n_0$

Therefore, $n = \omega(\sqrt{n})$

(b)

If $f(n) = \Omega(g(n))$, and $h(n) = \Theta(g(n))$, then $f(n) = \Omega(h(n))$.

$f(n) = \Omega(g(n)) \Leftrightarrow$ there exist constants $c_1 > 0$ and $n_1 \geq 0$ such that $f(n) \geq c_1 g(n)$ for all $n \geq n_1$

$h(n) = \Theta(g(n)) \Rightarrow$ there exist constants $c_2 > 0, n_2 > 0$ such that $h(n) \leq c_2 g(n) \Leftrightarrow g(n) \geq \frac{1}{c_2} h(n)$ for all $n \geq n_2$

Therefore, there exist $c_3 = \frac{c_1}{c_2} > 0$, $n_3 = max(n_1, n_2) > 0$, such that for all $n \geq n_3, f(n) \geq c_1 g(n) \geq c_1(\frac{1}{c_2} h(n)) = c_3 h(n)$

Therefore, $f(n) = \Omega(h(n))$.

Midterm Solution for Q8

Zhen Wang

Fall-2021

1 Problem 8(12 points)

(a) (4 points) Recursion tree is asymmetric and inbalanced: the recursion that

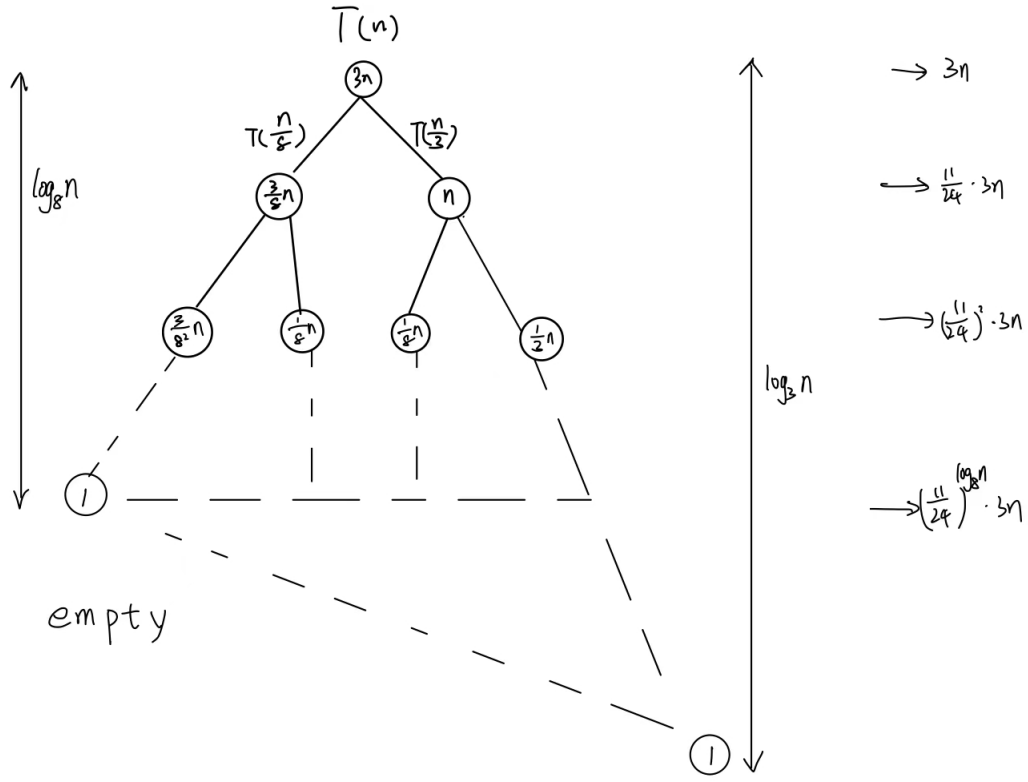


Figure 1: asymmetric and inbalanced recursion tree

divide the problem size by 8 reach the bottom the fastest at depth $log_8 n$, whereas the recursions that divide the problem size by 3 reach the bottom at depth $log_3 n$.

Other recursions that keep dividing the problem size by combinations of 8 and 3 are in the middle. Note that the cost at each depth is reduced by a factor of $\frac{1}{8} + \frac{1}{3} = \frac{11}{24}$. In other words, the merging cost at depth k is $3n * \frac{11}{24}^k$. Then we can bound $T(n)$ from above and below by

$$upper = 3n(1 + \frac{11}{24} + \frac{11}{24}^2 + \frac{11}{24}^3 + \cdots + \frac{11}{24}^{log_3 n}) \tag{1}$$

$$lower = 3n(1 + \frac{11}{24} + \frac{11}{24}^2 + \frac{11}{24}^3 + \cdots + \frac{11}{24}^{log_8 n}) \tag{2}$$

$$lower \leq T(n) \leq upper \tag{3}$$

As $n \to \infty$, both the upper and lower bounds tend to $\frac{72}{13}n$, which implies $T(n) = \Theta(n)$

(b) (4 points)

The upper bound:

IH: $T(k) \leq dk$ for all $k < n$

$$T(n) = T(\frac{n}{8}) + T(\frac{n}{3}) + 3n \leq d\frac{n}{8} + d\frac{n}{3} + 3n \tag{4}$$

$$d\frac{n}{8} + d\frac{n}{3} + 3n \leq dn \tag{5}$$

$$d \geq \frac{72}{13} \tag{6}$$

If we set $d \geq \frac{72}{13}, T(n) \leq dn \Rightarrow T(n) = O(n)$

The lower bound:

IH: $T(k) \geq ck$ for all $k < n$

$$T(n) = T(\frac{n}{8}) + T(\frac{n}{3}) + 3n \geq c\frac{n}{8} + c\frac{n}{3} + 3n \tag{7}$$

$$c\frac{n}{8} + c\frac{n}{3} + 3n \geq cn \tag{8}$$

$$c \leq \frac{72}{13} \tag{9}$$

If we set $c \leq \frac{72}{13}, T(n) \geq cn \Rightarrow T(n) = \Omega(n)$

Bound by the upper and the lower, we can get $T(n) = \Theta(n)$

(c) (4 points)

$$T(n) = 3T(\sqrt{n}) + logn^2 \tag{10}$$

set $m = logn$, we have,

$$T(2^m) = 3T(2^{\frac{m}{2}}) + m^2 \tag{11}$$

set $S(m) = T(2^m)$

$$S(m) = 3S(\frac{m}{2}) + m^2 \tag{12}$$

Use master method: $a = 3, b = 2, f(m) = m^2$
In case 3: $af(m/b) \leq cf(m)$ for some $c \leq 1$ and all sufficiently large n

$$3(\frac{m}{2})^2 \leq cm^2 \tag{13}$$

$$\frac{3}{4}m^2 \leq cm^2 \tag{14}$$

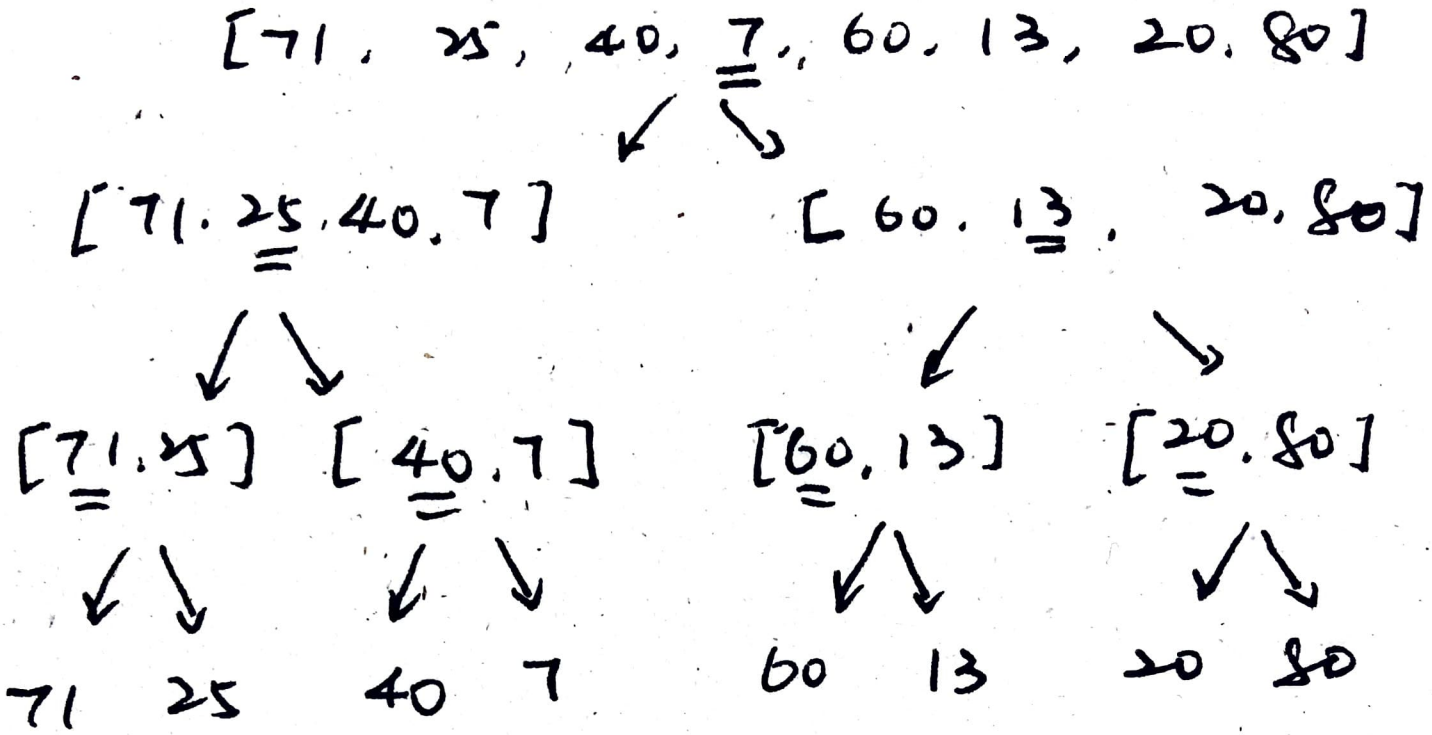
Then we can use the master method case 3 in this recurrence:

$$S(m) = \Theta(m^2) \tag{15}$$

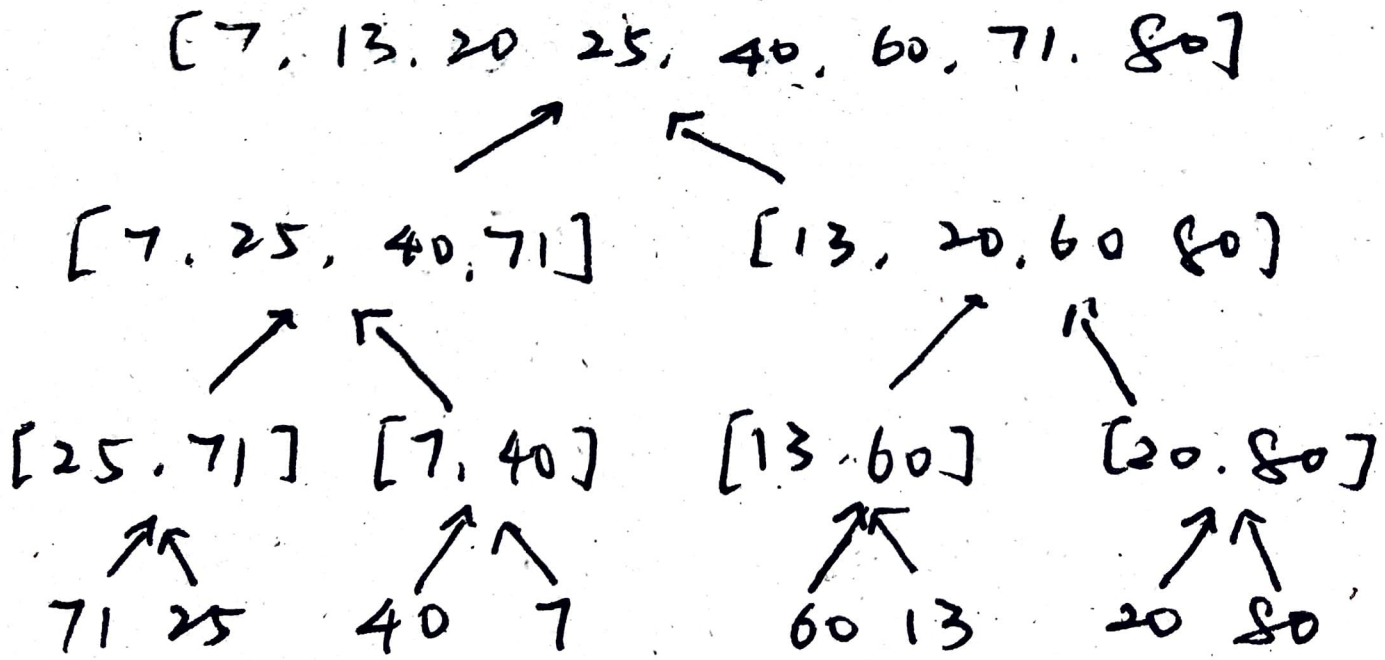
$$T(n) = \Theta((logn)^2) \tag{16}$$

9.

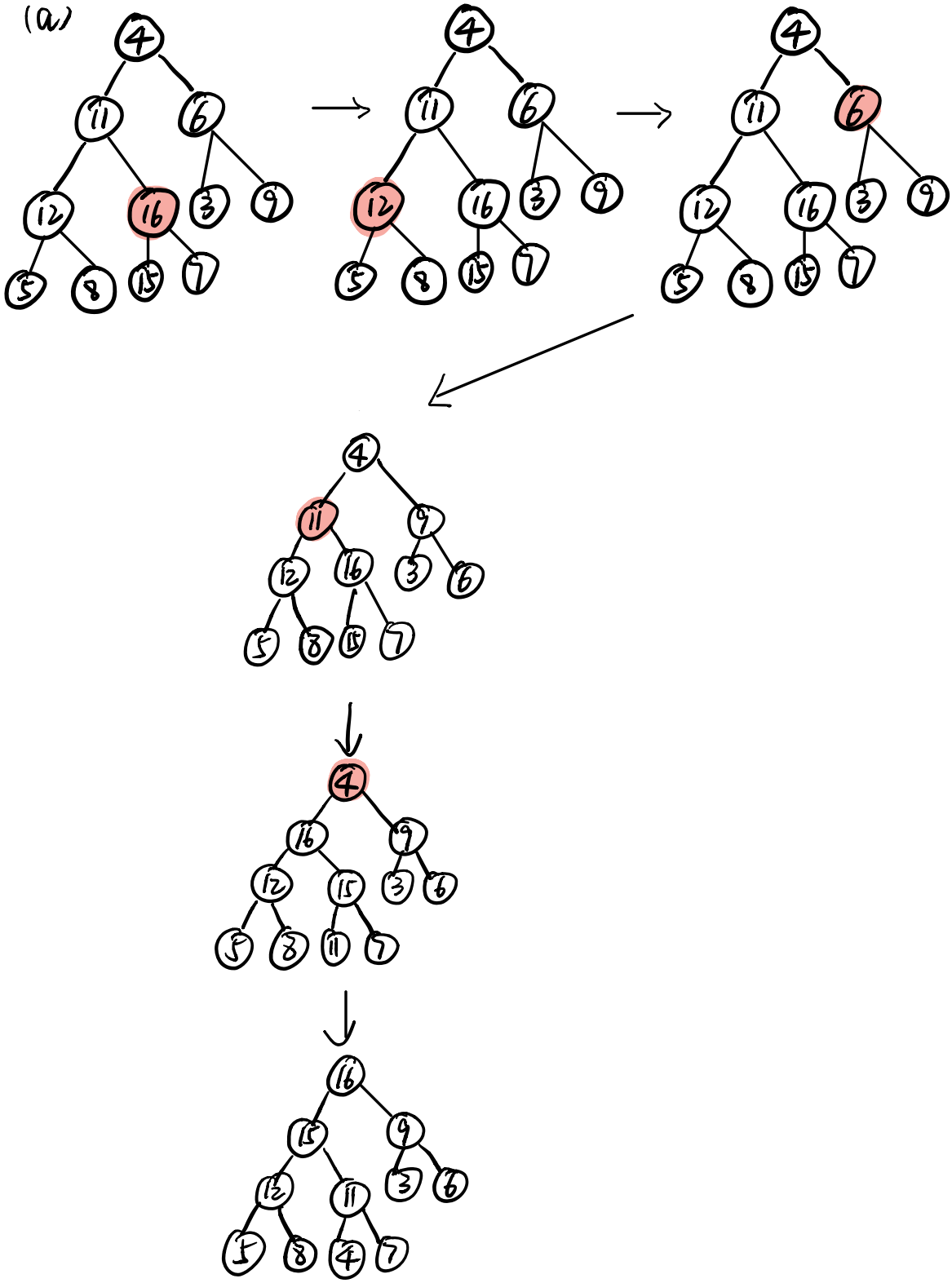
① Divide:



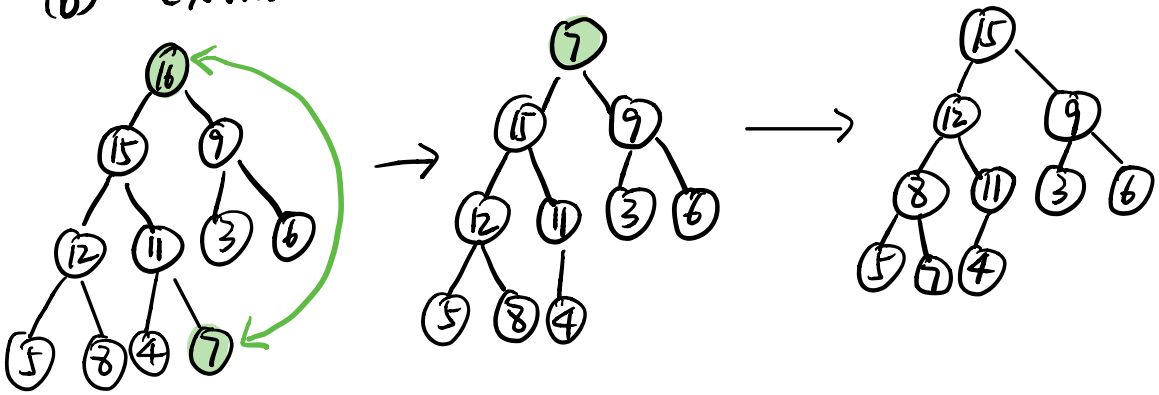
② Conquer and merge:



10. (a)

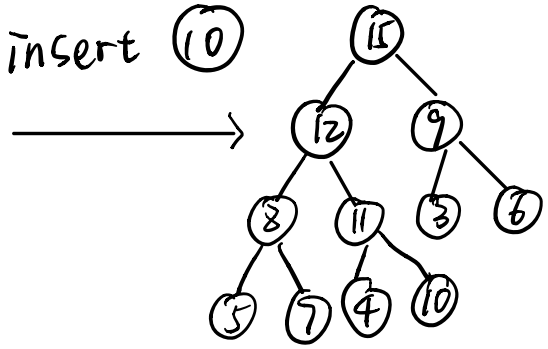

4
11
6
12
16
3
9
5
8
15
7
4
11
6
12
16
3
9
5
8
15
7
4
11
6
12
16
3
9
5
8
15
7
4
11
9
12
16
3
6
5
8
15
7
4
16
9
12
15
3
6
5
8
11
7
16
15
9
12
11
3
6
5
8
4
7

(b) extract - max



get maximum : 16 .

insert 10



11. (**6 points**) Design a $\Theta(n)$ algorithm to find the minimum of an unordered array.

(a) (**2 points**) write down the pseudocode;

(b) (**4 points**) prove the correctness of your algorithm using Loop-Invariant.

Answer: a)

```
Alg.: Find-Minimum(A)
 n <-- length[A]
 smallest=A[1]
 for i <-- 2 to n
      if A[i] < smallest
         then smallest=A[i]
 return smallest
```

b):

(a) *Loop Invariant:* before iteration i, `smallest` is the minimum of the first $i-1$ elements in the original array.

(b) *Initialization:* when $i = 2$, $i - 1 = 1$, trivially true.

(c) *Maintenance:* In the i-th iteration, if `A[i]` is less than `smallest`, which is the minimum of the first $i-1$ elments, then `A[i]` is the minimum of the first i elements, and `smallest` is updated. Therefore, at the end of the i-th iteration, `smallest` becomes the minimum of the first i elements. The Loop Invariant holds.

(d) *Termination:* The loop finishes when $i = n$, then the minimum of the whole array is returned.

12. (**12 points**) Answer the following questions about Selection:

(a) (**4 points**) To find the k-th order statistic of a list of n distinct numbers using randomized selection, what is the probability that the k-th order statistic is found right after the first call of randomized-partition (i.e., no need to do second partition)? what is the probability that the k-th order statistic is found right after the second call of randomized-partition?

(b) (**8 points**) If there is a selection algorithm that can find any k-th order statistic of any array with n or less numbers with worst-case time complexity of $O(n)$, describe how to use this algorithm to find the k-th order statistic of any array of $2n$ numbers with worst-case complexity of $O(n)$, where $1 \leq k \leq \frac{n}{2}$ or $\frac{3n}{2} \leq k \leq 2n$. (*NOTE: the given selection algorithm cannot work with any array with more than n numbers, and you are not allowed to use any other $O(n)$ Selection algorithm.*)

Answer: a) The k-th order statistic is found right after the first call of randomized-partition if and only if the pivot chosen is the k-th order statistic itself, which happens with probability of $\frac{1}{n}$.

If the k-th order statistic is not found after the first call, let the rank of the chosen pivot to be $i, (i \neq k)$, the Selection algorithm will work on the partition containing the k-th order statistic. If $i < k$, the active partition length is $n - i$; if $i > k$, then the active partition length is $i - 1$. The second randomized-partition call will chose a pivot from the active partition randomly, so the probability that k-th order statistic is chosen as the second pivot is:

$$\frac{1}{n}\left(\sum_{i=1}^{k-1}\frac{1}{n-i} + \sum_{i=k+1}^{n}\frac{1}{i-1}\right),$$

which is the probability that the k-th order statistic is found right after the second call of randomized-partition.

b)

(a) divide the array A equally into two arrays A_1 and A_2, each with n elements;

(b) if $k \leq \frac{n}{2}$,
using the given algorithm to find the k-th order statistic of A_1 and A_2;
partition A_1 and A_2 using its k-th order statistic; (assuming Lomuto's partition)
combine the two left partitions and the two pivot values into one new array B of $2k(\leq n)$ elements;
using the given algorithm to find the k-th order statistic of B, return it as the k-th order statistic of A.

(c) if $k > \frac{3n}{2}$,
using the given algorithm find the $(k-n)$-th order statistic of A_1 and A_2;
partition A_1 and A_2 using its $(k-n)$-th order statistic; (assuming Lomuto's partition)
combine the two right partitions and the two pivot values into one new array B of $2(2n-k+1)(\leq n)$ elements;
using the given algorithm to find the $(2n-k+2)$-th order statistic of B, return it as the k-th order statistic of A.

Optional: if $k = \frac{3n}{2}$,
find and remove the maximum and the second maximum from the original array A and get a new array A' of $2(n-1)$ elements;
find the $k = \frac{3n}{2}$-th order statistic of A' using a process similar to case (c), return it as the $k = \frac{3n}{2}$-th order statistic of A.

Correctness: For case b), any number in the right partition of A_1 and A_2 already have more than k elements less than them, so they are larger than the k-th order statistic of A; Then B contains all the elements less than or equal to the k-th order statistic, and the k-th order statistic of B is the same as k-th order statistic of A.

For case c), any number in the left partition of A_1 and A_2 already have more than $n - (k-n) = 2n - k$ elements larger than them, so they are smaller than the k-th order statistic of A; Then B contains all the

elements larger than or equal to the k-th order statistic of A, the k-th order statistic of A is the same as $2(2n-k+1)-(2n-k) = 2n-k+2$-th order statistic of B.

13. (**12 points**) Answer the following questions about hash table with collision resolved by chaining.

(a) (**3 points**) If the hash table has m slots numbered between 1 and m, 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. Let X_i be the position of the i-th ($1 \le i \le n$) inserted key in its chain, what is the probability mass function of X_i, i.e., calculate $P(X_i = k)$, $\forall k \ge 0$?

(b) (**4 points**) If the hash table has $2m$ slots numbered between 1 and $2m$, 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 ($1 \le j \le 2m$)?

(c) (**5 points**) For the hash table in (b), what is the expected time complexity for unsuccessful search? what is the expected time complexity for successful search?

Answer a) There were $n-i$ keys inserted after the i-th key, each of them has a probability of $1/m$, to be inserted into the same chain as the i-th key. The position of the i-th key X_i is just the number of keys positioned before the i-th key in its chain (assuming the head position is 0), it is a Bernoulli trial of $n-i$ experiments, and the success probability of each trial is $1/m$, so X_i follows a binomial distribution of $(n-i, \frac{1}{m})$, the p.m.f. of X_i is

$$P(X_i = k) = \binom{n-i}{k} \frac{1}{m^k} \left(1 - \frac{1}{m}\right)^{n-i-k}, \quad n-i \ge k \ge 0$$

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

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

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

(a) $P_1 = 0$;

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

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

C) For unsuccessful search of a key not in the table, it will be hashed to slot j with probability P_j as obtained in (b), the average length of the list at slot j is nP_j, so the average number of elements visited by the unsuccessful search is $n\sum_{j=1}^{2m} P_j^2$, the average time complexity is $1 + n\sum_{j=1}^{2m} P_j^2$;

For successful search of a key, if it is the i-th inserted key, it will be in the chain at slot j with probability P_j, and its expected position in the chain is $(n-i)P_j$, so the expected number of elements examined by a successful search of the i-th inserted key is $1 + (n-i)\sum_{j=1}^{2m} P_j^2$.

Finally, the expected number of elements examined by a successful search of any already inserted key is:

$$1 + \frac{1}{n}\sum_{i=1}^{n}(n-i)\sum_{j=1}^{2m} P_j^2 = 1 + \frac{n-1}{2}\sum_{j=1}^{2m} P_j^2$$

This page intentionally is left blank. You can write answers here if you used up space in the front.
Don't tear off this page.