EL9343 Midterm Exam (2020 Fall)

Name: **ID:**

Oct 25, 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:** Tree height of a max heap is $\Theta(logn)$;
   (b) **T or F:** $2^n = \Theta(3^n)$;
   (c) **T or F:** By using uniform hashing functions, the number of elements hashed into a slot is independent from the number of elements in any other slot;
   (d) **T or F:** Bubble sort is stable;
   (e) **T or F:** After inserting n elements into an empty binary search tree, search complexity is $O(logn)$;
   (f) **T or F:** In QuickSort, if the array has n elements, the running time for the first partition procedure is $\Theta(n)$.

2. (**4 points**) Which of the following function pairs satisfy $f(n) = \omega(g(n))$?
   (a) $f(n) = n$ and $g(n) = 3n$;
   (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 $k > 1$;
   (e) None of the above

3. (**4 points**) For a given node v in a binary search tree, which of the following node can be its successor?
   (a) Left child of v's right child;
   (b) v's right child;

(c) Right child of v's left child;

(d) Parent of v;

(e) None of the above.

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

(a) Counting sort is stable;

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

(c) The most imbalanced partition generated by Hoare's scheme is caused by a already sorted input (without using randomization);

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

(e) None of the above.

5. (**4 points**) Which of the following statements regarding Divide-and-Conquer algorithms are true?

(a) In quicksort algorithm, partition takes more time than combine the solutions to the subproblems;

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

(c) The master theorem cannot be used to solve all recurrences;

(d) Insertion sort is a divide-and-conquer algorithm;

(e) None of the above.

6. (**4 points**) Which of the following statements about max-priority queue of size n are true?

(a) The queue can be stored in an array of size n;

(b) The max element is always at the root of the tree;

(c) Extract-max takes $\Theta(1)$;

(d) Find the minimum element takes $\Theta(\log n)$;

(e) None of the above.

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

(a) (**3 points**) $\sqrt{n} = o(n)$, where $o(\cdot)$ stands for little-o;

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

Answer: a). For any $C > 0$, $\sqrt{n} < Cn$ as long as $n \geq n_0(C) = \frac{1}{C^2} + 1$, so it satisfies the definition of $o(\cdot)$;

b). $h(n) = \Theta(g(n)) \Rightarrow c_1 g(n) \leq h(n) \leq c_2 g(n)$, $\forall n \geq n_0$;
$f(n) = \Omega(g(n)) \Rightarrow f(n) \geq c_3 g(n)$, $\forall n \geq n_1$;
let $n_2 = \max(n_0, n_1)$, then $f(n) \geq c_3 g(n) \geq \frac{c_3}{c_2} h(n)$, $\forall n \geq n_2$, so let $c_4 = \frac{c_3}{c_2}$, this satisfies the definition of $O(\cdot)$.

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

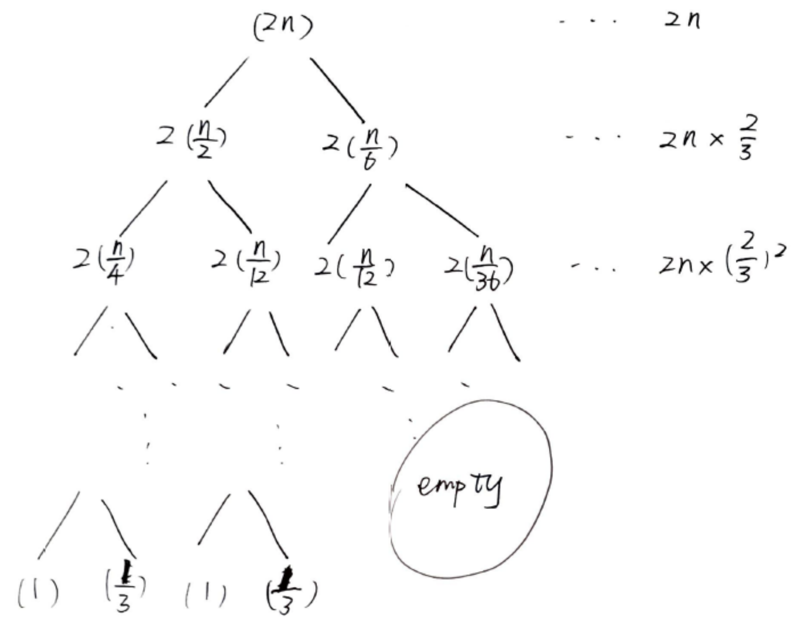
(a) (**4 points**) Use the iteration method to solve $T(n) = T(\frac{n}{6}) + T(\frac{n}{2}) + 2n$;

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

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

7.

(a) Recursion tree is shown as below:



It is imbalanced, the recursions that divide the problem size by 6 reach the bottom the fastest at depth $\log_6 n$, whereas the recursion that divide the problem size by 2 reach the bottom at depth $\log_2 n$

$\therefore$ We have

$C_1 = 2n + 2n \cdot \frac{2}{3} + 2n \cdot (\frac{2}{3})^2 + \cdots 2n \cdot (\frac{2}{3})^{\log_6 n} \leq T(n) \leq 2n + 2n \cdot \frac{2}{3} + \cdots + 2n \cdot (\frac{2}{3})^{\log_2 n} = C_2$

$C_1 = 2n \dfrac{1 - (\frac{2}{3})^{\log_6 n + 1}}{1 - \frac{2}{3}} = 6n(1 - \frac{2}{3} n^{\log_6 \frac{2}{3}})$, which is $\Theta(n)$

$C_2 = 2n \dfrac{1 - (\frac{2}{3})^{\log_2 n + 1}}{1 - \frac{2}{3}} = 6n(1 - \frac{2}{3} n^{\log_2 \frac{2}{3}})$, which is $\Theta(n)$

So, $T(n)$ is $\Theta(n)$.

(b)

The upper bound:

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

$T(n) = T(\frac{n}{6}) + T(\frac{n}{2}) + 2n \le d\frac{n}{6} + d\frac{n}{2} + 2n$

To make it $\le dn$

$d\frac{n}{6} + d\frac{n}{2} + 2n \le dn \Rightarrow d \geq 6$

If we set $d \geq 6$, $T(n) \le dn$. $\therefore$ $T(n)$ is $O(n)$

The lower bound:

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

$T(n) = T(\frac{n}{6}) + T(\frac{n}{2}) + 2n \geq c\frac{n}{6} + c\frac{n}{2} + 2n$

To make it $\geq cn$.

$c\frac{n}{6} + c\frac{n}{2} + 2n \geq cn \Rightarrow c \le 6$

If we set $c \le 6$, $T(n) \le cn$. $\therefore$ $T(n)$ is $\Omega(n)$.

So, $T(n)$ is $\Theta(n)$.

(c) Let $m = \log n$

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

Let $S(m) = T(2^m)$, we have

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

use master method:

$S(m) = O(m^2)$

$\therefore$ $T(n) = T(2^m) = S(m) = O(m^2) = O(\log^2 n)$

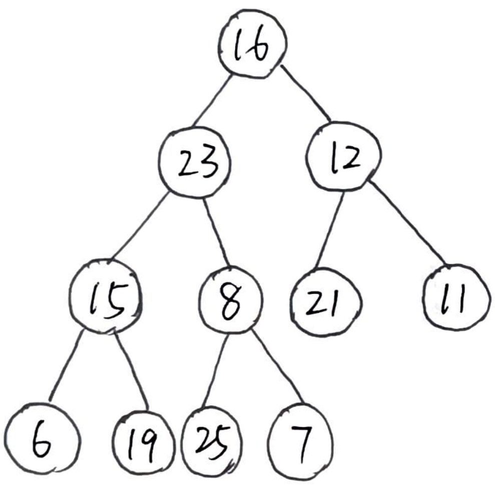
9. (**8 points**) If heap-sort is applied to an array of $[16, 23, 12, 15, 8, 21, 11, 6, 19, 25, 7]$,

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

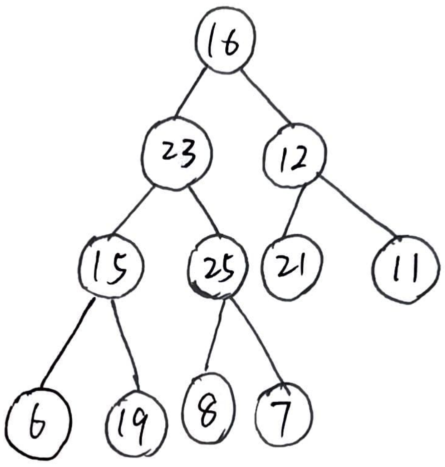
(b) (**4 points**) plot the max-heap after the third-maximum is extracted.

9.

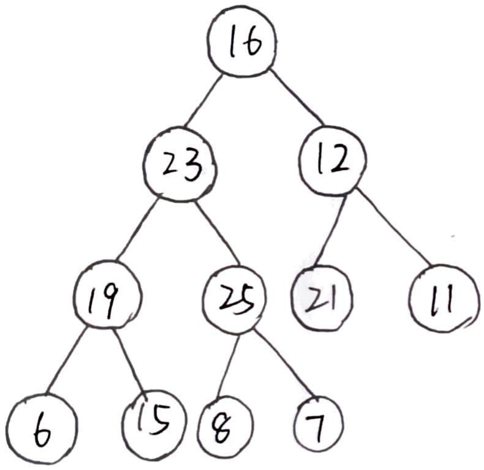
(a)



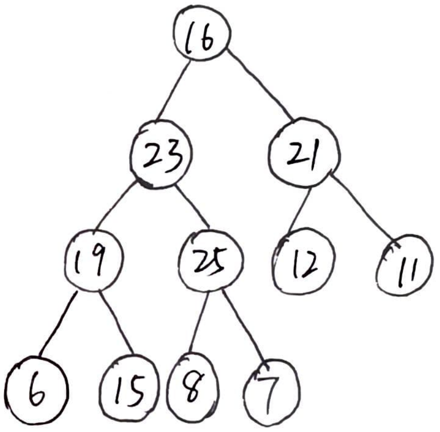
$\xrightarrow{\text{HEAPIFY}(A,5)}$



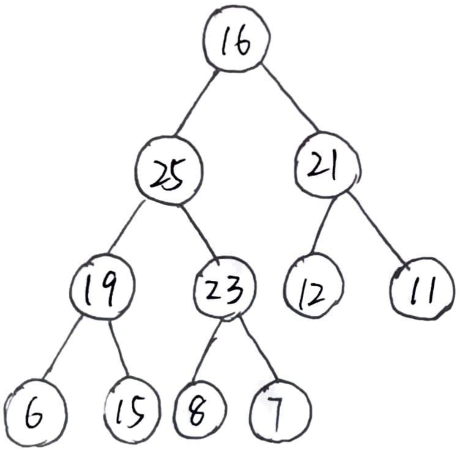
$\xrightarrow{\text{HEAPIFY}(A,4)}$



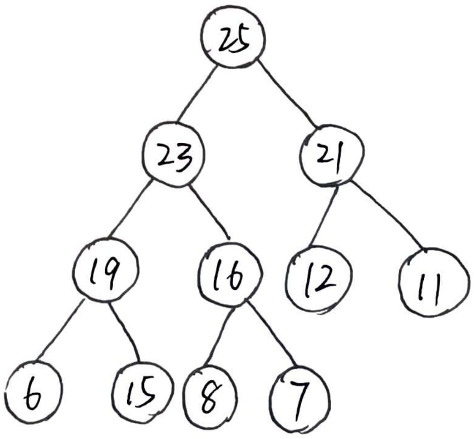
$\xrightarrow{\text{HEAPIFY}(A,3)}$



$\xrightarrow{\text{HEAPIFY}(A,2)}$



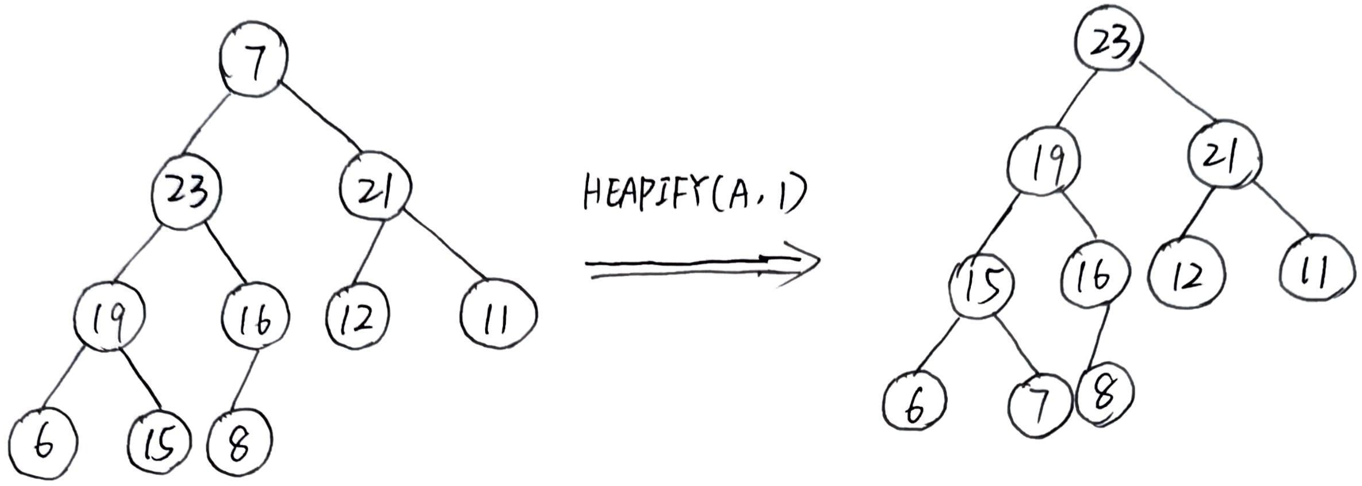
$\xrightarrow{\text{HEAPIFY}(A,1)}$



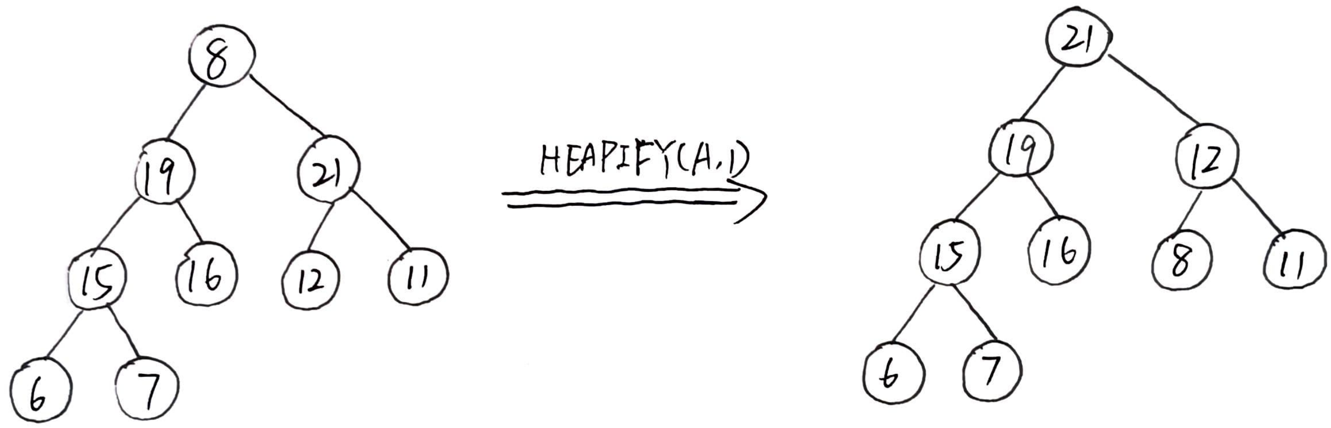
So, the max-heap is < 25, 23, 21, 19, 16, 12, 11, 6, 15, 8, 7 >

(b)

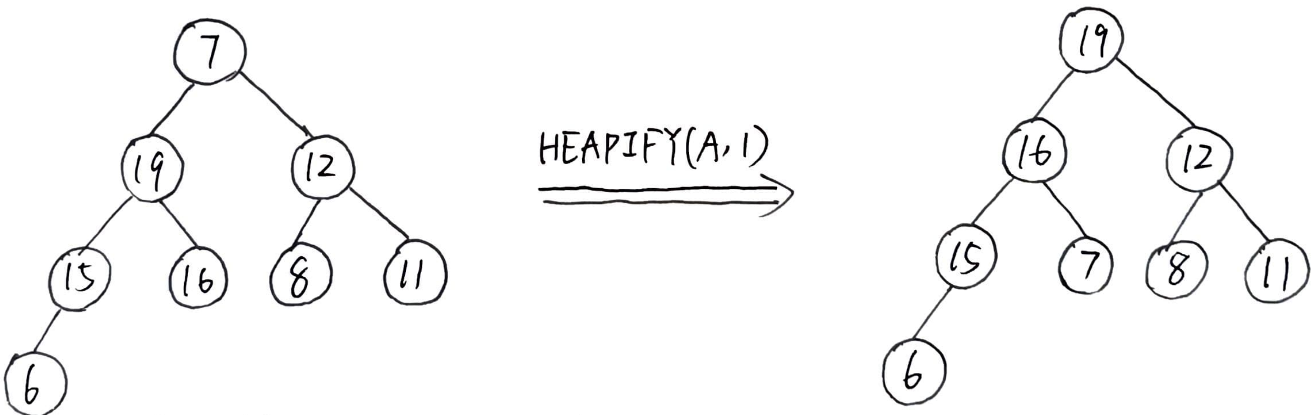
max1 = 25



max2 = 23



max3 = 21



So, the result heap is < 19, 16, 12, 15, 7, 8, 11, 6 >

10. (**12 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 probability that the left partition length is m, for any $0 \le m \le n$?

(b) (**4 points**) suppose the second partition call is for the left partition resulted from the first partition call, what is the probability that the i-th ranked number is in the same partition as the $(i+k)$-th ranked number after the first two partition calls?

(c) (**4 points**) is it true that the i-th ranked number will always be compared with the $(i+1)$-th ranked number exactly once throughout the sorting process? prove it if it is true, or disprove it if it is false.

Answer:

a) Let the rank of the randomly selected pivot be Y, the left partition length is $Y-1$, so

$$P\{\text{left partition length =m}\} = P\{Y = m+1\} = \begin{cases} \frac{1}{n}, & 0 \le m \le n-1 \\ 0, & m = n \end{cases}$$

b) after the first partition call, the i-th ranked number and $(i+k)$-th ranked number will both be in the left partition if the left partition length is larger than or equal to $i+k$; they will both end up in the right partition if the length of the left partition less than i;

Since the second partition call only works on the left partition, so the probability that the two numbers will be in the same partition after two calls can be calculated as the sum of the probability that both in left partition after the first call & stay in the same partition after the second call, and the probability that both in the right partition after the first call. If the left partition length is $j \ge i+k$, the probability that i and $i+k$ will be in the same partition after the second call is $\frac{j-k-1}{j}$. The the final probability is

$$P\{i \text{ and } i+k \text{ stay in the same partition after two calls}\} = \frac{1}{n}\sum_{j=i+k}^{n-1} \frac{j-k-1}{j} + \frac{i-1}{n},$$

where the last term can be replaced by $\frac{i}{n}$ if an empty left partition is also counted.

c) True. Based on the lecture, the probability that the two numbers with rank difference of k will be compared is $\frac{2}{k+1}$. Here the rank difference is $k=1$, so the probability that they will be compared is 1. Since each pair will be compared at most once, so the i-th ranked number will always be compared with the $(i+1)$-th ranked number exactly once.

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

(a) (**5 points**) When the i-th ($1 \le i \le n$) key is inserted to the hash table, let Y_i be the random variable of the length of the list the i-th key will be inserted to, what is the probability mass function of Y_i, i.e., calculate $P(Y_i = k)$, $\forall k \ge 0$?

(b) (**3 points**) What is the expected value of Y_i?

(c) (**3 points**) After all the n distinct keys have been inserted, what is the expected number of elements examined when searching for a new key not from the n inserted keys?

Answer: a). Y_i depends on how many keys inserted before the i-th key are hashed into the same slot as the i-th key, each of which happens with probability of $\frac{1}{m}$. Therefore, Y_i follows the binomial distribution with $(i-1)$-trial and $p = \frac{1}{m}$, so its p.m.f is

$$P(Y_i = k) = \binom{i-1}{k} \frac{1}{m^k} \left(1 - \frac{1}{m}\right)^{i-1-k} ; \forall 0 \le k \le i-1$$

b). The expected value is $E[Y_i] = \frac{i-1}{m}$;

c). The average number of elements examined for a unsuccessful search is simply the average length of the chains, which is $\frac{n}{m}$.

12. (**13 points**) Let X and Y be two arrays of n real numbers sorted into non-decreasing order. Design and analyze an algorithm that finds the median of the $2n$ combined numbers in worst-case $O(\log n)$ time.

(a) (**9 points**) write down the pseudo-code of your algorithm;

(b) (**4 points**) analyze its time complexity.

Answer:

a).

Compare $X[\frac{n}{2}]$ with $Y[\frac{n}{2}]$,
1) If $X[\frac{n}{2}] = Y[\frac{n}{2}]$, then the final median $M = X[\frac{n}{2}] = Y[\frac{n}{2}]$;
2) If $X[\frac{n}{2}] > Y[\frac{n}{2}]$, then the final median M should satisfy $X[\frac{n}{2}] > M > Y[\frac{n}{2}]$, therefore, we only need to find the median of two arrays $X[1 \cdots \frac{n}{2}]$ and $Y[\frac{n}{2} \cdots n]$;
3) If $X[\frac{n}{2}] < Y[\frac{n}{2}]$, then the final median M should satisfy $X[\frac{n}{2}] < M < Y[\frac{n}{2}]$, therefore, we only need to find the median of two arrays $Y[1 \cdots \frac{n}{2}]$ and $X[\frac{n}{2} \cdots n]$;
In the later two cases, the problem size reduces by half each time, so the worst-case running time is $O(\log n)$.

b).

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