ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2020

Homework 6

Due date: 23:59, September 26, 2019

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL Name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
5. When submitting, match your solutions to the according problem numbers correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero grade.

Problem 1: Multiple Choices

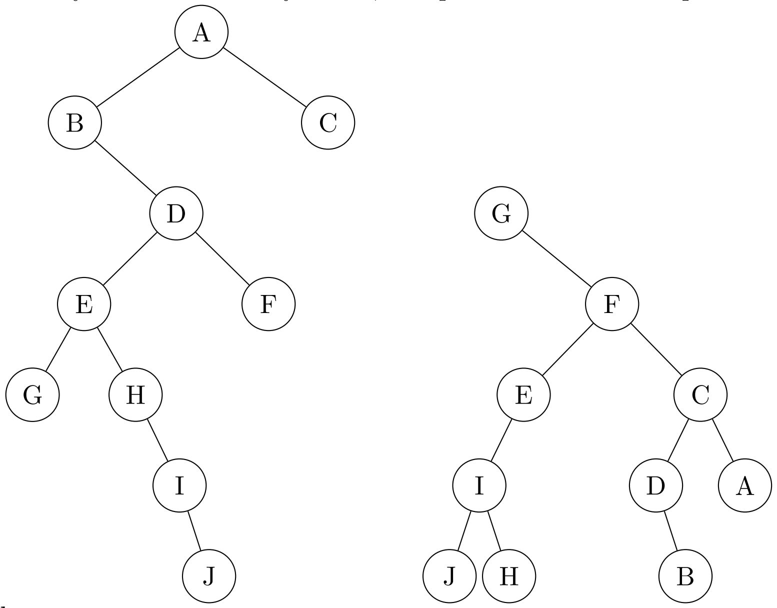
Multiple Choices: Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get half points if you select a non-empty subset of the correct answers.

Note that you should write you answers of Problem 1 in the table below.

Q(1)	Q(2)	Q(3)
AC	B	AB

(1) Which of the following statements about the binary tree and the heap is true?
A. If the in-order traversal and post-order traversal of two binary trees are equal respectively, then the two binary trees are exactly the same.
B. If the pre-order traversal and post-order traversal of two binary trees are equal respectively, then the two binary trees are exactly the same.
C. If the post-order traversal of a complete binary tree is descending, then the binary tree is a min-heap.
D. If a binary tree is a min-heap, then the post-order traversal of this tree is descending.

(2) Which traversals of binary tree 1 and binary tree 2, will produce the same sequence node name?



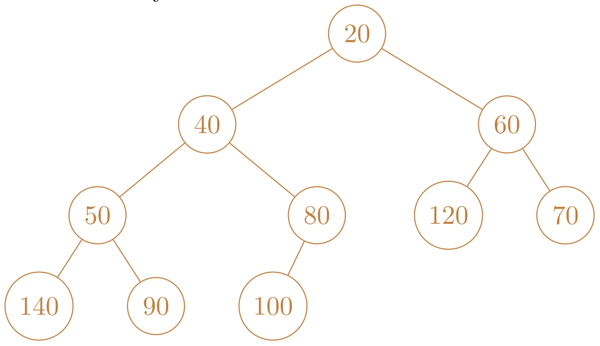
A. Post-order, Post-order
B. Post-order, In-order
C. In-order, In-order
D. Pre-order, Pre-order

(3) Which of the following statements about the binary heap is **not** true?
A. There exists a heap with seven distinct elements so that the in-order traversal gives the element in sorted order.
B. If item A is an ancestor of item B in a heap (used as a Priority Queue) then it must be the case that the

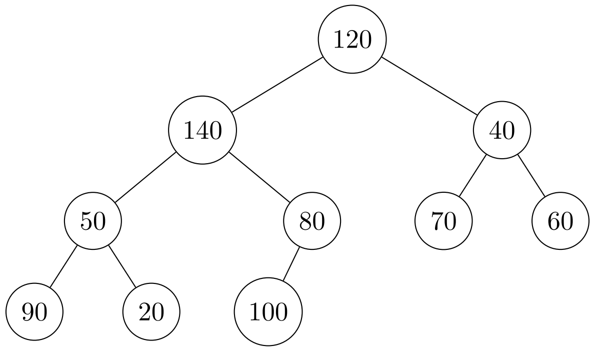
Insert operation for item A occurred before the **Insert** operation for item B.
C. If array A is sorted from smallest to largest then A (excluding A[0]) corresponds to a min-heap.
D. None of the above.

Problem 2: Construct a Heap

(1) Suppose we construct a min-heap from the following array: $120, 140, 40, 50, 80, 70, 60, 90, 20, 100$, by inserting elements into a empty heap in sequence. After the construction is completed, what should this heap look like? Please draw it as a binary tree.



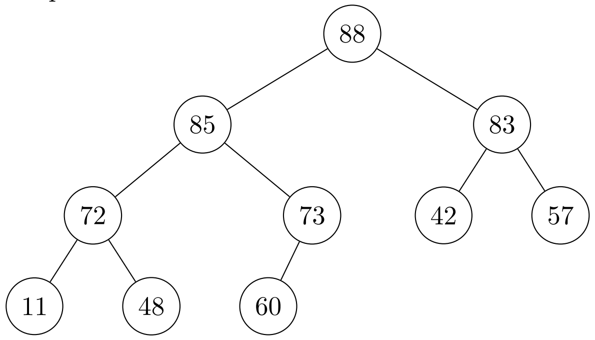
(2) Suppose we construct a min-heap from the following initial heap by Floyd's method. After the construction is completed, we delete the root from the heap. What will be the post-order traversal of the heap? Write down your answer in the table above directly.
Your answer: 120, 140, 90, 80, 50, 70, 100, 60, 40.



(3) If we use the Floyd's method to build a heap of size N, the worst-case time complexity is $O(n)$.
(4) If we use the "inserting elements into a empty heap" method to build a heap of size N, the worst-case time complexity is $O(n \log n)$.

Problem 3: Heap Sort

You are given such a max heap like this:



Then you need to use array method to show each step of heap sort in increasing order. Fill in the value in the table below. Notice that the value we have put is the step of each value sorted successfully. For each step, you should always make you heap satisfies the requirement of max heap property.

index	0	1	2	3	4	5	6	7	8	9	10
value		88	85	83	72	73	42	57	11	48	60

Table 1: The original array to represent max heap.

index	0	1	2	3	4	5	6	7	8	9	10
value		85	73	83	72	60	42	57	11	48	88

Table 2: First value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		83	73	57	72	60	42	48	11	85	88

Table 3: Second value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		73	72	57	11	60	42	48	83	85	88

Table 4: Third value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		72	60	57	11	48	42	73	83	85	88

Table 5: Fourth value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		60	48	57	11	42	72	73	83	85	88

Table 6: Fifth value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		57	48	42	11	60	72	73	83	85	88

Table 7: Sixth value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		48	11	42	57	60	72	73	83	85	88

Table 8: Seventh value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		42	11	48	57	60	72	73	83	85	88

Table 9: Eighth value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value		11	42	48	57	60	72	73	83	85	88

Table 10: Last 2 values are successfully sorted.

Problem 4: Median Produce 101

The well known TV show *Produce 101* has a new rule to judge all the singers: for each singer, use the median score among all the judges to set her score. Previously, the program groups have a calculator to calculate the score for each singer. Accidentally, the calculator is broken one day. And the fans of singer ChaoYue are eagerly waiting for the score. So they want to help the program group to calculate the correct median score.
Recall that the median value of a set is the value that separates the higher half of set's values from the set's values from the set's lower values.
For example, given the set with **odd** numbers of elements:

$$\{78, 94, 17, 87, 65\}$$

The median score is 78.

For another example, given the set with **even** numbers of elements:

$$\{78, 94, 17, 87, 65, 76\}$$

The median score is $(78 + 76)/2 = 77$.

Consider a set S of arbitrary and distinct integer scores (not necessatily the set shown above). Let n denote the size of set S, and assume throughtout this problem that n can be odd or even.
Now, fancy fans finds a data structure "Dynamic Medians" for maintaining a set S of numbers, supporting the following operations:
CREATE(): Creat an empty set S
INSERT(x): Add a new given number x to S
MEDIAN(): Return a median of S.
Assume no duplicates are added to S. He proposes to implement this "dynamic median" data structure DM using a maxheap A and a minheap B, such that the following two properties always hold:
1. Every element in A is less than every element in B, and
2. the size of A equals the size of B, or is one less.
To return a median of S, she proposes to return the minimum element of B.

(1) Argue that this is correct (i.e., that a median is returned)

1. For odd situation: Assume that A has n elements, then B has n or n+1 elements. The median element should be the (n+1)-th element, which is the minimum of B.
2. Since the difference between the even case and the odd case lies only in the definition of median, So that no matter how you discuss the even case, we think it is correct.

(2) Explain how to implement **INSERT(x)**, while maintaining the relevant properties. Analyze the most efficient running time of **INSERT** algorithm in terms of n, the number of elements in S.
A is max-heap and B is min-heap.

1. If A's size and B's size are all zero, then **PUSH x** into B.
2. Else if A's size is smaller than B's size, and **x** is less than B's top, then **PUSH x** into A.
3. Else if A's size is equal to B's size, and **x** is less than A's top, then **POP** A's top and **PUSH** it into B, after that, **PUSH x** into A.
4. Else if A's size is smaller than B's size, and **x** is greater than B's top, then **POP** B's top and **PUSH** it into A, after that, **PUSH x** into B.
5. Else (if A's size is equal to B's size, and **x** is greater than A's top), **PUSH x** into B

Since we only use at most 3 times of heap **POP** or heap **PUSH**, so that the time complexity should be $O(\log n)$