CS101 Algorithms and Data Structures
Fall 2022
Homework 5

Due date: 23:59, October 23th, 2022

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 problems correctly.

6. No late submission will be accepted.

7. Violations to any of the above may result in zero points.

1. (?? points) Multiple Choices

Each question has **one or more** correct answer(s). Select all the correct answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 1 point if you select a non-empty subset of the correct answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)
BC	B	B	C

(a) (2') Which of the following is/are true about Trees?

    A. A full binary tree with n nodes has height of $O(\ln n)$.

    B. Only given the depth of all leaf nodes in a tree, we can infer the height of this tree.

    C. If T was transformed from a general tree T' through Knuth transform, which means T is a left-child right-sibling binary tree. Then the post-order traversal of T' identical to the in-order traversal of T.

    D. None of the above.

(b) (2') Let's look at some magic properties about trees, which of the following statements is/are true?

    A. A tree is a full binary tree if and only if every node has an odd number of descendants.

    B. A rooted binary tree has the property that the number of leaf nodes equals to the number of full nodes plus 1.

    C. There are 133 distinct shapes of binary trees with 6 nodes.

    D. None of the above.

(c) (2') Which of the following statements about the binary heap is/are true? Note that binary heaps mentioned in this problem are implemented by complete binary trees.

    A. If a binary tree is a min-heap, then the post-order traversal of this tree is a descending sequence.

    B. We have a binary heap of n elements and wish to add n more elements into it while maintaining the heap property. It can be done in $O(n)$.

    C. There exists a heap with seven distinct elements so that the in-order traversal gives the element in sorted order.

    D. If item A is an ancestor of item B in a heap, then it must be the case that the `Push` operation for item A occurred before the `Push` operation for item B.

    E. None of the above.

(d) (2') Which traversals of the left tree and right tree, will produce the same sequence node name?
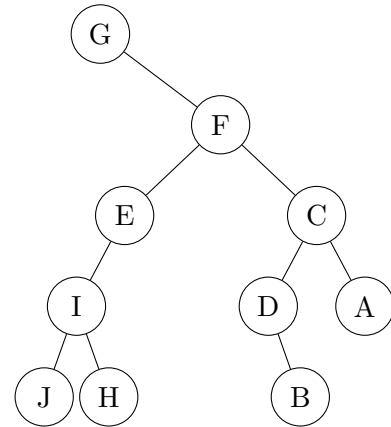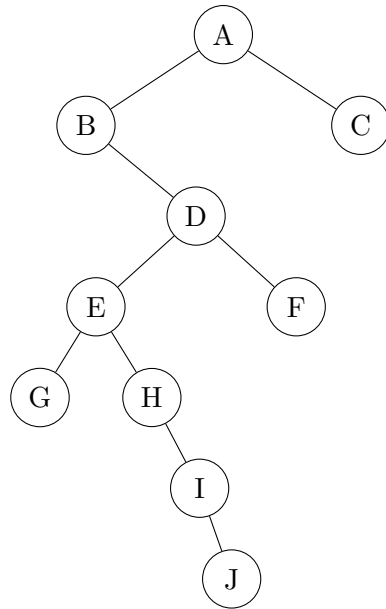
A. left: Post-order, right: Post-order

B. left: Pre-order, right: Pre-order

C. left: Post-order, right: In-order

D. left: In-order, right: In-order

2. (?? points) Draw a binary tree
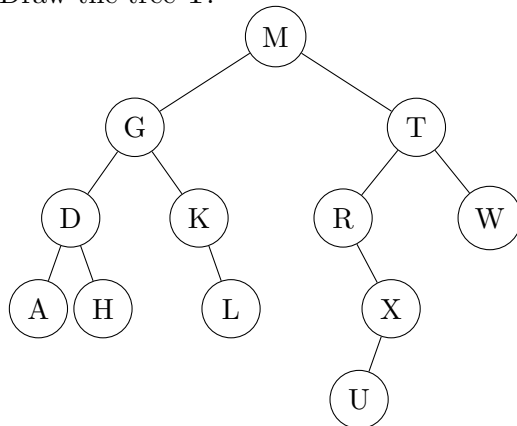
    (a) (3') Given the in-order and pre-order traversal of a binary tree T are IGDHBAECF and ABDGIHCEF respectively.
Draw the tree T.
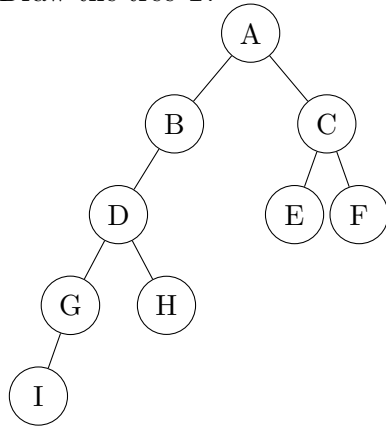
    (b) (3') Given the in-order and post-order traversal of a binary tree T are ADHGKLM-RUXTW and AHDLKGUXRWTM respectively.
Draw the tree T.

(c) (2') Given the pre-order and post-order traversal of a binary tree T, can you decide the tree T? If yes, please describe an algorithm to construct T; if no, please provide a counterexample.

No.
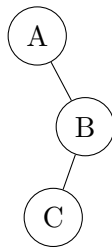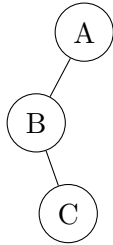
these two trees both have the
pre-orfer traversal   ABC
and the post-orfer traversal   CBA
so we can not get a tree with it pre-oder and post-order traversal.

3. (?? points) Heap

We are given the following array representing a min-heap where each letter represents a unique number. Assume the root of the min-heap is at index zero, i.e. A is the root. Note that there is no significance of the alphabetical ordering, i.e. just because B precedes C in the alphabet, we do not know if B is less than or greater than C.

Array: [A, B, C, D, E, F, G]

Six unknown operations are then executed on the min-heap. An operation is either a pop or a push.The resulting state of the min-heap is shown below

Array: [A, G, B, X, E, F, C]

(a) (8') The first and the final operations have already been filled in for you. Fill in the space below using the following options.

Note: You are free to use each of the following options 0 time, once or more than once.

(A) pop()
(B) push(B)
(C) push(C)
(D) push(X)

Hint: The number of elements in the heap doesn't change after 6 operations; In the final heap, X's position is in the left sub-heap.

1. pop()
2. _____D_____
3. _____A_____
4. _____A_____
5. _____B_____
6. push(A)

(b) (4') Fill in the following comparisons with either $>$, $<$, or ? if unknown. We recommend considering which elements were compared to reach the final array.

1. X $>$ B
2. X $>$ C
3. G $?$ F
4. G $>$ D

4. (?? points) Heap Sort

(a) (10') You are given such a max heap like this:
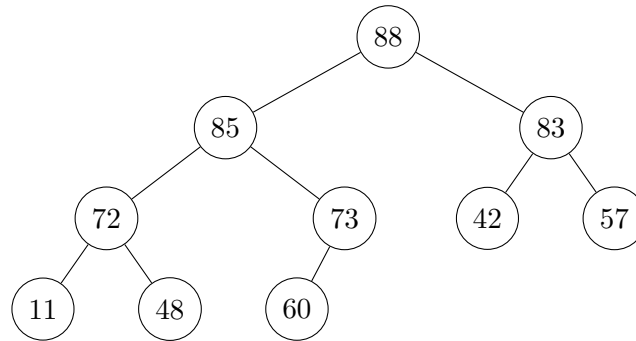
Then you need to use array method to show each step of heap sort in increasing order. Fill in the value in the table below. Notice that the value we have put is the step of each value sorted successfully. For each step, you should always make you heap satisfies the requirement of max heap property.

index	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	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	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	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.

7

index	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	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	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	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	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	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.

5. (?? points) k-ary Heap

In class, our professor has mentioned the method of the array storage of a binary heap. In order to have a better view of heap, we decide to extend the idea of a binary heap to a k-ary heap. In other words, each node in the heap now has at most k children instead of just two, which is stored in a complete binary tree.

For example, the following heap is a 3-heap.
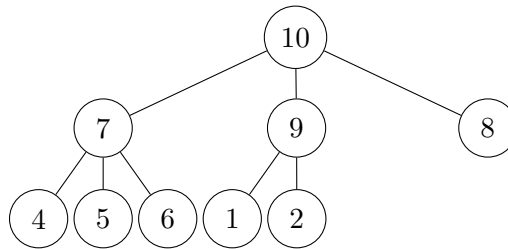
(a) (4') If you are given the node with index i, what is the index of its parent and its jth child $(1 \leq j \leq k)$?

Notice: We assume the root is kept in $A[1]$. For your final answer, please represent it in terms of k, i and j. Please use flooring or ceiling to ensure that your final answer is as tight as possible if you need, i.e. $\lfloor m \rfloor$, $\lceil m \rceil$.

> **Solution:**
> parent: $\lfloor \frac{i+k-2}{k} \rfloor$
> j_th child: $k \cdot (i-1) + j + 1$

(b) (4') What is the height of a k-ary heap of n elements? Please show your steps.

Notice: For your final answer, please represent it in terms of i and n. Please use flooring or ceiling to ensure that your final answer is as tight as possible if you need, i.e. $\lfloor m \rfloor$, $\lceil m \rceil$.

> **Solution:**
> suppose that the k-ary heap with n elements has the height h
> it is meaningless when $k \le 1$, so we just let $k \ge 2$
> and the heap is a complete k-ary tree, so the number of its node is more than a perfect k-ary tree with height $h - 1$, and is less than or equal to a perfect k-ary tree with height h
> and since a perfect k-ary tree with height h has the number of node:
> $1 + k + k^2 + \cdots + k^h = \frac{k^{h+1} - 1}{k - 1}$
> so $\frac{k^{(h-1)+1} - 1}{k - 1} < n \le \frac{k^{h+1} - 1}{k - 1}$
> since $k \ge 2$, so $k - 1 > 0$, so simplify it we have
> $\log_k(n(k - 1) + 1) - 1 \le h < \log_k(n(k - 1) + 1)$
> so we can take $h = \lceil \log_k(n(k - 1) + 1) \rceil$
> above all, the height of heap is $\lceil \log_k(n(k - 1) + 1) - 1 \rceil$

(c) (4') Now we want to study the complexity of operations on k-ary heap. Suppose the $\mathtt{Push}$, $\mathtt{Pop}$ and $\mathtt{Heap-sort}$ operations execute the same as what we mentioned in our lectures. Denote the height of the heap as h, please analyse the worst-case time complexity of $\mathtt{Push}$ and $\mathtt{Pop}$ operations on k-ary heap in terms of n, h, k with tight bound. For heap-sort, given a built heap, explain why the worst-case number of comparisons is $\Theta(nhk)$.

> **Solution:**
> let the heap be the min heap.
> $\mathtt{push}$ operation: after putting a new node into the heap, the new node need to compare with its father, and if its value is less than its father's, then swap the two nodes. for the worst case, the new node become the root of the heap, with needs to compare n times,
> so the complexity of $\mathtt{push}$ is $\Theta(h)$
>
> $\mathtt{pop}$ operation: after removing the root of the heap, we put the last node to the root, and it need to compare with all its k children, if there exist some children with less value, then we need to swap the child with minimum value with it. since the node have k children, so we need to compare k times each time. And for the worst case, the node become the leaf node, so we need to repeat h times above.
> so the complexity of $\mathtt{pop}$ operation is $\Theta(k \cdot h)$
>
> for $\mathtt{heap-sort}$: for the worst case, each time when we want to pop, the pop operation takes the worst time, and the heap-sort of n elements requires total n pop operations, so it takes time of $n\cdot$ pop()'s complexity $= n \cdot (k \cdot h) = nhk$
> so above all, the worst case of heap-sort's complexity is $\Theta(nhk)$.