ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2018

Homework 4

Due date: 23:00, October 28, 2018

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 above may result in zero score.

Problem 1: Binary Tree Warmup!

Suppose the Binary Tree has nine nodes. The value of each node is: $1, 2, 3, 4, 5, 6, 7, 8, 9$ respectively.

(1) What's the Binary Tree you can make from those numbers with the maximum height? Draw one of the trees, write down the height and explain your answer.

(2) What's the Binary Tree you can make from those numbers with the minimum height? Draw one of the trees, write down the height and explain your answer.

(3) For a Binary tree with n distinct node, what is the maximum height of such a tree? Represent your answer in terms of n and explain your answer.

(4) For a Binary tree with n distinct node, what is the minimum height of such a tree? Represent your answer in terms of n and explain your answer.

Problem 2: Reconstructing Binary Trees via Traversals

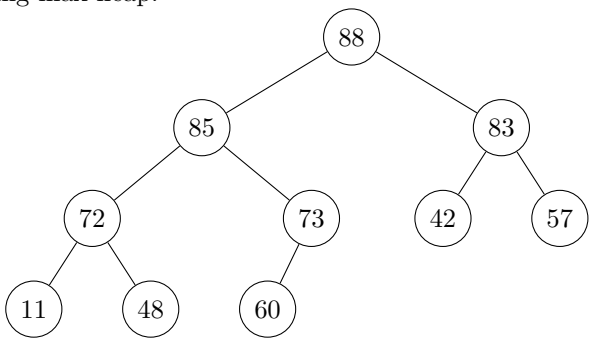
Recall the binary tree data structure; recall three algorithms for traversing the tree: the *inorder* traversal, the *preorder* traversal, and the *postorder* traversal.

(1) Suppose you are given the preorder traversal and the inorder traversal of a binary tree. Can you reconstruct the **unique binary tree**? If so, give an algorithm for doing so and prove that the unique tree can be built using the preorder and inorder sequences. If not, give a counter example using a binary tree of 5 nodes: 1,2,3,4,5. Note that a binary tree is not necessarily a binary search tree.

(2) Suppose you are given the preorder and postorder traversals of a binary tree. Can you reconstruct the **unique binary tree**? If so, give an algorithm for doing so and prove its correctness. If not, give a counter example using a binary tree of 5 nodes: 1,2,3,4,5.

Problem 3: Heap Sort

You are given the following max heap:



You need to represent the heap in an array (start from the index 1), then conduct in-place heap sort.

In each step of the sort, we pop the max value from the heap and place it in the end of the heap.

Fill in the blanks of the arrays in each step.

index	0	1	2	3	4	5	6	7	8	9	10
value											

Table 1: The original array to represent max heap.

index	0	1	2	3	4	5	6	7	8	9	10
value											88

Table 2: First value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value										85	88

Table 3: Second value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value									83	85	88

Table 4: Third value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value								73	83	85	88

Table 5: Fourth value is successfully sorted.

index	0	1	2	3	4	5	6	7	8	9	10
value							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						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					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				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: Heap Analysis

For parts(1), (2) of this problem, you may assume that the *input* heaps are **complete binary trees**; for parts (1), (2) your *output* heaps need **not be necessarily complete**. Note that the input heaps are not given to you in the form of arrays and your output heaps need not be in the form of arrays.

(1) Given an algorithm to merge two heaps of size m and n (assume $n \geq m$), into a single heap of size $(m + n)$ in time $O(\log n)$. Analyze your algorithm and prove the time complexity.

(2) Suppose you are given a heap H of size m, and another n unordered elements, where $n \geq m$. What is the runtime to insert these n elements into the heap H? What is the total runtime required to build a second heap $H^{'}$ of size n and then merge H and $H^{'}$ into a single heap using your algorithm in part (1)?

Problem 5: k-ary Heap

Consider a new heap structure called k-ary heap, where each node in the heap now has k **children** instead of just two.

(1) If the heap is represented by a one-dimensional array A, describe how to find the parent and the (at most) k children of element $A[i]$. That is, given a node of index i, what is the index of its parent and its jth child?

(2) What is the height of a k-ary heap of n elements in terms of n and k?

(3) The worst-case number of comparisons $T(n,k)$ performed by HEAPSORT is $\Theta(nhk) = cnhk$, where $h = \lfloor(\log_k n)\rfloor$ is the height of the heap, and constant c is a fixed constant known to you. In this problem, **we suppose the constant $\mathbf{c = 1}$**.

Solve for k so that $T(n,k)$ is minimized. In other words, we need to make the number of comparisons as small as possible. Note that k is an integer actually.

Problem 6: Median Produce 101

Now the hotest TV show *Produce 101* has a new rule to judge all the singers: for all judgers, use the median score among all the judgers to set her score. Previously, the programme gropus have a calculator to calculate the score for each singer. Accidentally, the calculator is broken one day. And the fans of singers ChaoYue are eagerly waiting for the score. So they want to help the programme 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)

(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.