ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2019

Homework 4

Due date: 23:59, October 20, 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 above may result in zero score.

Problem 1: Binary Tree & Heap

Binary Tree and Heap have a ton of uses and fun properties. To get you warmed up with them, try working through the following problems.

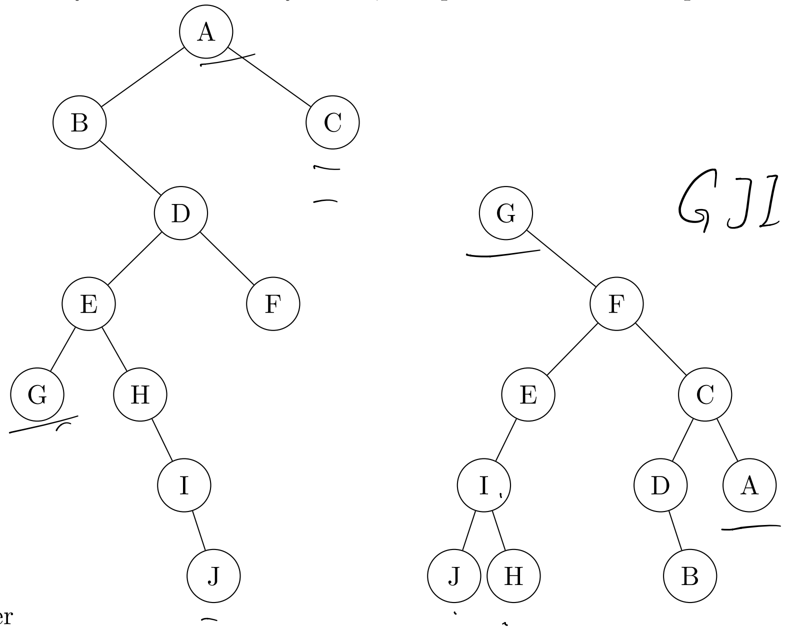
Multiple Choices: Each question has one or more correct answer(s). Select all the correct answer(s). For each question, you get 0 point if you select one or more wrong answers, but you get 0.5 point if you select a non-empty subset of the correct answers.

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

Q(1)	Q(2)	Q(3)	Q(4)	Q(5)
BCD	B	C	B	120 140 90 80 50 70 100 60 40

(1) Which of the following statements about the binary tree is true?
A. Every binary tree has at least one node.
B. Every non-empty tree has exactly one root node.
C. Every node has at most two children.
D. Every non-root node has exactly one parent.

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

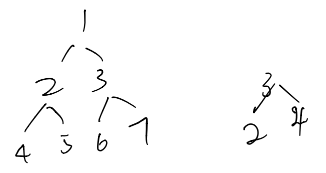


GJIHE

GJt

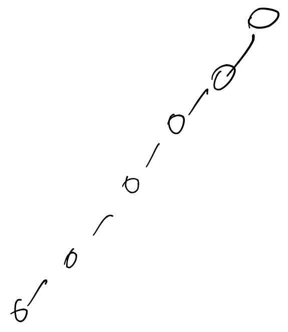
A. Postorder, Postorder
B. Postorder, Inorder
C. Inorder, Inorder
D. Preorder, Preorder

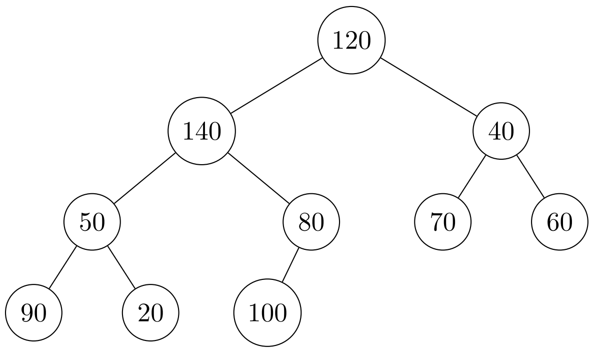
(3) Which of the following statements about the binary tree is **not** true?
A. A rooted binary tree has the property that $n_0 = n_2 + 1$, where n_i denotes the number of nodes with i degrees. √
B. Post-order traverse can give the same output sequence as a BFS. ✓
C. BFS and DFS on a binary tree always give different traversal sequences. ✗
D. None of the above.

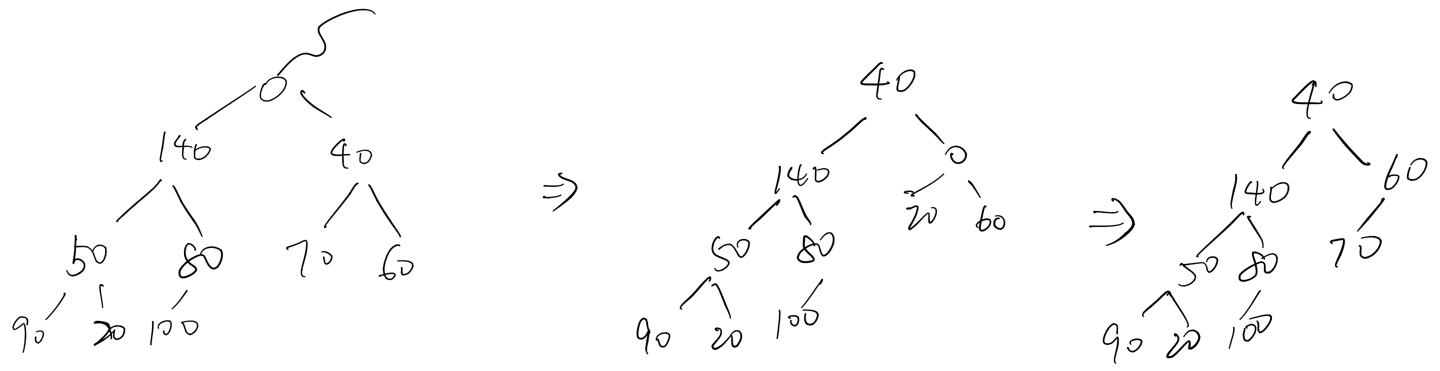


(4) Which of the following statements about the binary heap is **not** true?
A. There exists a heap with seven distinct elements so that the in-order traversal gives the element in sorted order.
B. If item A is an ancestor of item B in a heap (used as a Priority Queue) then it must be the case that the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

(5) 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.



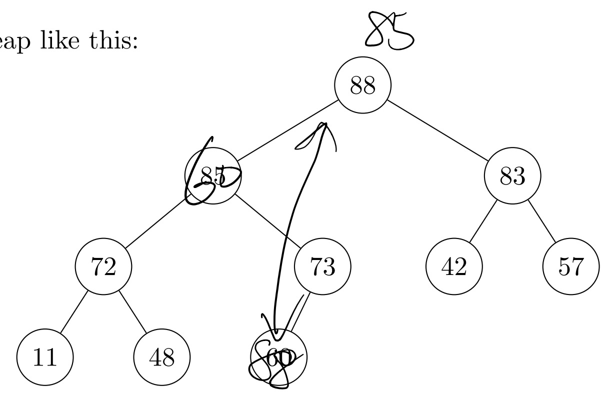




90 20 50 100 80 140 70 60 40

Problem 2: 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	72	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	11	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	48	42	73	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 3: Median Produce 101

Nowadays the hotest variety show *Produce 101* has a new rule to judge all the singers: for all 5 judgers, use the median score value among all the judgers to set her score. Previously, the programme group has a calculator to calculate the score for each singer. Accidentally, the calculator is broken one day. And the fans of famous star Yang Chaoyue are eagerly waiting for the score. So they want to help the programme group to get the correct score.

Recall that the median of a set is the value that separates the higher half of 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 $(78 + 76)/2 = 77$.

In Yang's fans group, the crazy fans have quarrelled for the following two opinions to get this median score:

- Use only one min heap.
- Use both min heap and max heap.

Now they are asking you for your help. Please help them solve this problem.

So first, let's try the case with only one min heap to get the median score.

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 in this whole problem that n can be odd or even. **Assume that we have inserted all the elements in set S to the minheap.**

(1) **Using natural language**, describe how to implement the algorithm that returns the median from set S. Analyze your time complexity. (Suppose the total number of element in S is given, which is n. And the minheap has been built.)

You will receive full credit only if your method runs in $O(n\log n)$ time.

First, build the min-heap that all the father node should be less than the child node, applying the floyd algorithm, that is
1.Call the buildMaxHeap() function on the list. Also referred to as heapify(), this builds a heap from a list in O(n) operations.
2.Swap the first element of the list with the final element. Decrease the considered range of the list by one.
3.Call the siftDown() function on the list to sift the new first element to its appropriate index in the heap.
4.Go to step (2) unless the considered range of the list is one element.
And the process of setting up the min-heap is O(n).
Then apply the heap sort algorithm, That is:
1.Swap the root with the last element in the heap
2.Discard this last node by dicreasing the heap size.
3.Call MinHeapify() until only one node remains.
the heap sort algorithm takes O(n log n) time.
We can finally get the median value by simple get index n\2 's value.

And now, let's try the case with both max heap and min heap to get the median score.

Now, another fancy fan Wang Xiaoming finds a data structure for storing a set S of numbers, supporting the following operations:

- `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 that he can use a maxheap A and a minheap B to get the median score easily. These two heaps need to always satisfy the following two properties:

- Every element in A is smaller than every element in B.
- The size of A equals the size of B, or is one less.

To return a median of S, he proposes to return the minimum element of B. **Assume that we have inserted all the elements in set S to the two heaps.**

(2) Using two properties above, argue that this is correct, partially correct or wrong (i.e., that a median is returned). If it is not correct, can we find a strategy that calculates the median in $O(1)$ time complexity? Explain the reason and strategy (if we need) briefly. (Suppose the total number of element in S is given, which is n. And heap A, B have been built.)

Partially correct, he only take the even condition in to consideration, If it is the odd condition, the output should be the average of the min in the Maxheap and max in the Minheap.
We can get ir in O(1) if both heaps have been built.

(3) **Using natural language**, explain how to implement `INSERT(x)` operation. You should notice that these two properties should always be held. Analyze the most efficient running time of `INSERT` algorithm in terms of n. (Suppose the total number of element in S is given, which is n.)

To build the A and B is O(log n)
first define two int variables, medianlow and medianlarge, should be maintained at the same time. When the number of elements is odd, the median is stored in medianlow. When the number of elements is even, the median is the average of medianlow and medianlarge.

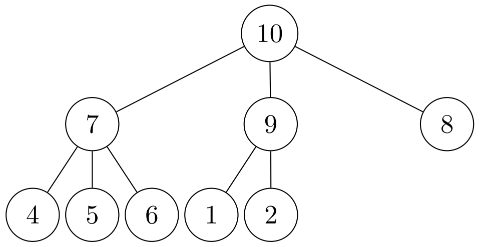
When inserting, if there are odd elements before inserting, discuss by situation:
If the inserted element is larger than the median before insertion, that is, meidanlow, the element is inserted into the minimum heap, and then the heap top element is taken out of the minimum heap and stored in medianlarge.
If the inserted element is less than the median before insertion, that is, meidanlow, insert the element into the maximum heap, then take out the heap top element in the maximum heap, store medianlow in medianlarge, and then store the taken heap top element in medianlow.
If there are even elements before the insertion, we will discuss them in the same way:

If the inserted element is larger than medianlarge, insert the element into the minimum heap, and then insert medianlow into the maximum heap. At this time, medianlarge is the median of the inserted array.
If the inserted element is less than medianlow, insert the element into the maximum heap, and then insert medianlarge into the minimum heap. At this time, meidanlaw is the median of the inserted array.
If the inserted element size is between medianlow and medianlarge, insert medianlow and medianlarge into the maximum heap and the minimum heap respectively. At this time, the element to be inserted is the median.
To build the heap one by one is O (log n), and the efficient time is constant numbers of O(log n), so the final time complexity is O(log n).

Problem 4: k-ary Heap

In class, Prof. Zhang 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-ary max-heap.



(1) 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, j and i. Please use flooring or ceiling to ensure that your final answer is as tight as possible if you need, i.e. $\lfloor x \rfloor$, $\lceil x \rceil$.

We can represent a k-ary heap in an array.

The root resides in $A[1]$, its k childdren reside in $A[2]$ through $A[k+1]$

according to the products Law, the children resides in $A[j+2]$ though $A[j^2+j+1]$ for $j \in [1,k]$

for any $j \in [1,k]$

k-ary-parent(i) = $\lfloor (i-2)/k+1 \rfloor$ or $\lceil (i-1)/k \rceil$

k-ary-child(i,j) = $k(i+1)+j+1$ or $k(i-1)+j+1$

and k-ary-parent(k-ary-child(i,j)) = k-ary-parent$(k(i+1)+j+1)$ = $\left\lfloor \frac{ki+k+i+j-1}{k+1} \right\rfloor = i$

or k-ary-parent$(k(i-1)+j+1)$ = $\left\lceil \frac{k(i-1)+j}{k} \right\rceil = i$.

specially $k=2$ is correct.

(2) 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 k and n. Please use flooring or ceiling to ensure that your final answer is as tight as possible if you need, i.e. $\lfloor x \rfloor$, $\lceil x \rceil$.

Since each node has k children, the hight of a k-ary heap with n nodes is $\Theta(\log_k n)$

We can rewrite it in ceiling mode $\lceil \log_k n \rceil$

(3) Now we want to study which value of k can minimize the comparison complexity of heapsort. For heapsort, given a built heap, the worst-case number of comparisons is $\Theta(nhk)$, where $h = \Theta(\log_k n)$ is the height of the heap. Suppose the worst-case number of comparisons is $T(n,k)$. You need to do:

- Explain why $T(n,k) = \Theta(nhk)$.
- Suppose n is fixed, solve for k so that $T(n,k)$ is minimized.

Notice: k is an integer actually. In this problem, we only consider the complexity of comparision, not the accurate number of comparision.

We need to find if the root element given to k-heapify is larger than all its children. If not, we swap it with its largest child and unstoppably call k-heapify until the largest is i

As for the k-heapify function.

```
k-heapify(A, i, k).
  largest = i.
  for j = 1 to k.
      j = k-child(i,j)
      if j ≤ size A and A[j] > A[largest]
          largest = j
  if largest ≠ i.
      swap(A[i], A[largest])
      k-heapify(A, largest, k)
```

```
k-build-heap(A, k)
  size[A] = length[A]
  for i = ⌈(length[A]-1)/k⌉ to 1
      k-heapify(A, i, d)
```

The complexity of the algorithm above is $O(k)$ comparisons which should operates n times(heapsort) in each all k-heapify that is $O(\log_k n)$ times call of heapify, and heapify takes constant time.

Mathmetically let $\log_k n = h$

$$T(n,k) \le \left(\sum_{h=0}^{\log_k(n(k-1))} kh \frac{n(k-1)}{k^{h+1}}\right) \cdot k \cdot h < n(k-1)\sum_{h=0}^{\infty} \frac{h}{k^h} \cdot k \cdot h$$

$$\le n(k-1)\left(\frac{1/k}{1-1/k}\right)^2 \cdot k \cdot h = n\frac{k}{k-1} \cdot hk$$

$\because \forall k \ge 2 \quad 1 \le \frac{k}{k-1} \le 2$ So $T(n,k) = \Theta(nhk)$

n is fixed $\quad f(k) = n^{1/k} = n \dfrac{\ln(n)}{\ln(k)} k$

$$f'(k) = n \ln n \frac{\ln k - 1}{\ln k^2}$$

$\because \dfrac{2}{\ln 2} > \dfrac{3}{\ln 3}$

$\therefore$ when $k=3$, $T(n,k)$ is minimized

(4) TA Yuan is motivated by professor, and he has a new idea. He wants to use k-ary heap to implement the heapsort algorithm. Because he wants to loaf on the job, he chooses $k = 1$. And he argues that we only need to do the `BUILD-HEAP` operation in the heapsort algorithm if $k = 1$. Since we know from the lecture that `BUILD-HEAP` takes $O(n)$ time, he thinks it can actually sort in $O(n)$ time!

From the above, we can conclude his two stataments:

- When $k = 1$, the only operation required in heapsort algorithm is `BUILD-HEAP`.
- When $k = 1$, the `BUILD-HEAP` operation will run in $O(n)$ time.

Now you are the student of TA Yuan, and you need to judge his two statements are true or false **respectively**.

- If his statement is true, please explain the reason and prove its correctness.
- If his statement is false, please help him find the fallacy in his argument with your own reason. In the heapsort he proposes, what sorting algorithm is actually performed? What is the worst running time of such a sorting algorithm?

stmt 1 is true

when k=1, after the heap built, the father is greater than its child if take max heap, also, the heap can be seen as a linked list for its 1ary property.

stmt2 is false

its equivalent to a linked list, so the time complexity is $O(n^2)$