ShanghaiTech University

CS101 Algorithms and Data Structures

Fall 2019

Homework 3

Due date: 23:59, October 6, 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.

1: (6') Insertion Sort, Bubble Sort and Trees

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 section 1 in the table below.

Question 1	Question 2	Question 3	Question 4	Question 5	Question 6
AC	AD	C	ABCD	B	ABC

Question 1. *If all the elements in an array are equal, for example* $\{1, 1, 1, 1, 1, 1\}$*, what would be the time complexity of the insertion sort?*

(A) $O(2n)$

(B) $O(n^2)$

(C) $O(n)$

(D) None of the above

Question 2. *Which of the following statements is/are true?*

(A) BFS uses a queue to keep track of vertices.

(B) DFS uses a queue to keep track of vertices.

(C) BFS uses a stack to keep track of vertices.

(D) DFS uses a stack to keep track of vertices.

Question 3. *Which of the following statements about basic bubble sort without flag is/are true?*

(A) The maximum number of comparisons for an n-element array is $(1/2)n(n+1)$*.*

(B) The time complexity is $O(n)$*.*

(C) There are two for loops in the structure, one nested in the other.

(D) None of the above.

Question 4. *Which of the following statements is/are false?*

(A) There exist no tree with negative height.

(B) Each node in a tree has exactly one parent.

(C) Nodes connecting to the same node are siblings.

(D) The root node is not an descendant of any node.

Question 5. *Consider a 4-degree tree with 20 4-degree nodes, 10 3-degree nodes, 1 2-degree node and 10 1-degree nodes, then how many leaf nodes does it have?*

(A) 41

(B) 82

(C) 113

(D) 122

Question 6. *The worst case asymptotic upper bound on the running time for both insertion-sort and bubble-sort will be the same if you know additionally that*

(A) the input is already sorted

(B) the input is reversely sorted

(C) the input is a list containing n copies of the same number

(D) None of the above

2: (4'+2') Time Complexity

Question 7. *Arrange the following functions in ascending order of growth. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.*
$f_1(n) = 2019$ $f_2(n) = 2^{\log_{\sqrt{2}} n}$ $f_3(n) = 2^{3^n}$ $f_4(n) = 3^{2^n}$
$f_5(n) = n^n$ $f_6(n) = \log \sqrt{n}$ $f_7(n) = \log(2^n n^2)$

f_1 f_6 f_7 f_2 f_5 f_4 f_3

Question 8. *Judge whether the following statement is true or false and explain why. Give a counter-example if it is false.*

(a) For any two functions $f(n)$ and $g(n)$, it must be the case that either $f(n) = O(g(n))$ or $f(n) = \Omega(g(n))$ holds.

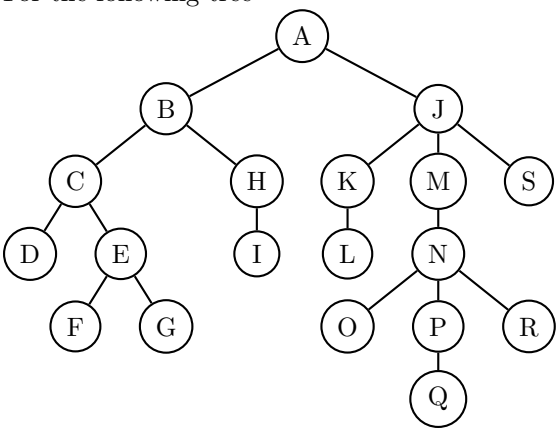
(b) If $f(n) = 2^n$ and $g(n) = 2.1^{n+\sin n}$, then $f(n) = O(g(n))$ always hold.

(a) F. Counter-example: n and $n^{1+\sin n}$.

(b) T. The limitation goes to 0.

3: (8*0.5'+3*1) Tree Structure

For the following tree



(a) What are the leaves? D,F,G,I,L,O,Q,R,S

(b) What are the children of B? C,H

(c) What is the depth of G? 4

(d) What is the degree of J? 3

(e) What are the ancestors of G? G,E,C,B,A

(f) What are the descendants of G? G

(g) What is the height of the tree? 5

(h) What is the degree of the tree? 3

(i) Enumerate the nodes in breadth-first order. ABJCHKMSDEILNFGOPRQ

(j) Enumerate the nodes in depth-first preorder. ABCDEFGHIJKLMNOPQRS

(k) Enumerate the nodes in depth-first posterorder. DFGECIHBLKOQPRNMSJA

4: (3') Binary tree property

In a rooted tree, the degree of a node is defined as the number of children of the node. For a binary tree, the degree is at least 0 and at most 2. We denote n_0 to be the number of nodes with degree 0, n_1 to be the number of nodes with degree 1 and n_2 to be the number of nodes with degree 2. Justify whether $n_0 = n_2 + 1$.

Let the number of all the edges in the binary tree be m. As we know the number of edges equals to the number of nodes minus one, we get $m = n_0 + n_1 + n_2 - 1$. Then consider that each node with degree 0 has 1 edge connecting it except for the root node, each node with degree 1 has 2 edges connecting it except for the root node, and each node with degree 2 has 3 edges connecting it except for the root node. Therefore, we get $2m = n_0 + 2n_1 + 3n_2 = 1$, where the -1 means exclude the missing edge connection of root. Combine the two equations, we can get that $n_0 = n_2 + 1$.

5: (3') Binary Tree Traversal

In the lecture, the pre-ordering, post-ordering of tree has been introduced. In fact, for a binary tree, the in-ordering of a tree is defined as:

(1) Check if the current node is empty or null.

(2) Traverse the left subtree by recursively calling the in-order function.

(3) Output the data part of the root (or current node).

(4) Traverse the right subtree by recursively calling the in-order function.

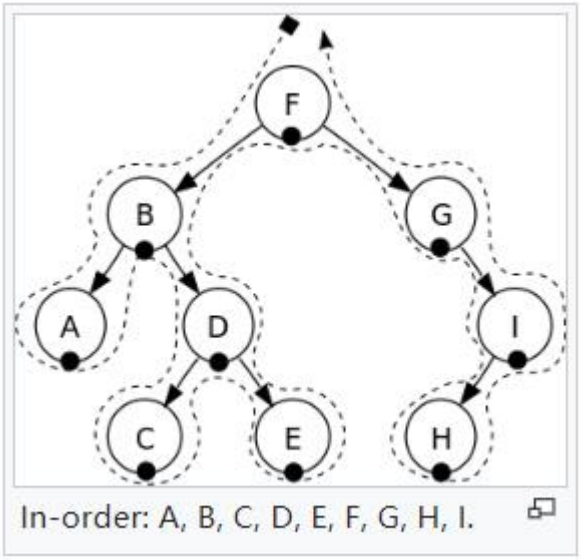


Figure 1: An illustration of an in-ordering of a binary tree

Given the pre-ordering and in-ordering of a binary tree as follows, is it possible to reconstruct the original tree? If it is, write down that tree. If it does not, write NO and enjoy your national holiday.

(1) Pre-ordering: abdeijfcgh

(2) In-ordering: dijefbghca