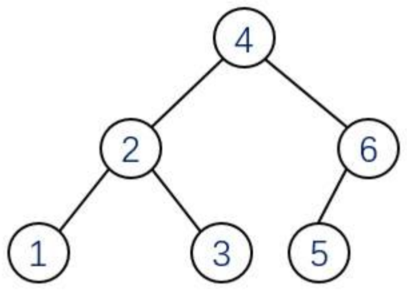
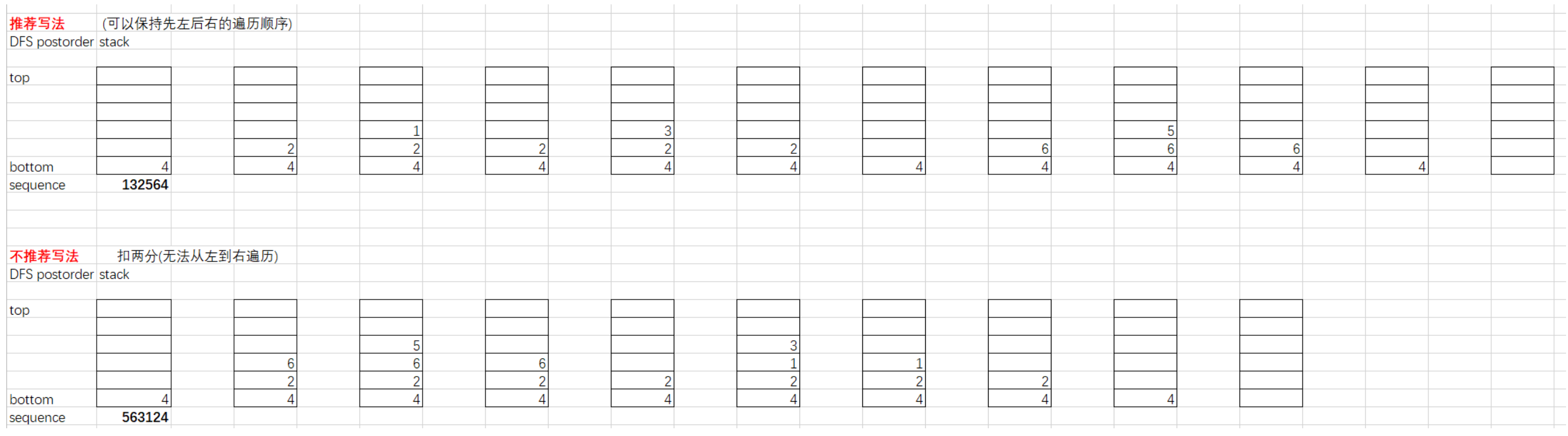
CS 101 Fall 2020 - Quiz 5-B

10/22/2020 - 20 Minutes **Name:** **ID number:**

Problem 1(4pts): Use a stack to implement **postorder** DFS, and show what are inside the stack step by step. Note that you should push the items onto stack according to **least first** when handling the cases where there are 2 children of some node. Then write the sequence of your **postorder** DFS.
(Don't worry if you can't write the right answer at one chance. You are provided with three stacks below and you can use them one by one until you think your answer is correct.)





推荐写法 (可以保持先左后右的遍历顺序)

DFS postorder stack

top												
			1		3				5			
		2	2	2	2	2		6	6	6		
bottom	4	4	4	4	4	4	4	4	4	4	4	
sequence	**132564**											

不推荐写法 扣两分(无法从左到右遍历)

DFS postorder stack

top										
			5			3				
		6	6	6		1	1			
		2	2	2	2	2	2	2		
bottom	4	4	4	4	4	4	4	4	4	
sequence	**563124**									

Problem 2 Single Choice(2×2pts): The following questions are single choice questions, each question has **only one** correct answer. Select the correct answer.

Note: You should write those answers in the box below.

Question 1	Question 2
A	B

2.1: What is time complexity of running DFS which contains n nodes using a stack?

(a) $O(n)$

(b) $O(nlogn)$

(c) $O(logn)$

(d) $O(n^2)$

2.2: Given the recurrence $f(n) = pf(n/q) + 1$, how many sub-problems will a divide-and-conquer algorithm divide the original problem into, and what will be the size of those sub-problems?

(a) p sub-problems, each of size q

(b) p sub-problems, each of size n/q

(c) q sub-problems, each of size p

(d) q sub-problems, each of size n/p

Problem 3(2pts): Let $T(n)$ be the function defined by $T(1) = 1$, $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + \sqrt{n}$ for $n \geq 2$. Which of the following statements is true? Write your answer directly.

Your answer: O(n)

Hint: you can get the correct answer quickly using Master Theorem.

Problem 3(5pts): There are n students and each student i has 2 scores x_i, y_i. Students i, j are friends if and only if $x_i < x_j$ and $y_i > y_j$. How many pairs of friends are there? Design an efficient algorithm. For comparison, our algorithm runs in $O(n \log n)$ time. (Hint: extend merge sort.)

Solution:

Sort these students in the ascending order of their x scores, so $x_i < x_j$ if $i < j$. Denote the array containing their corresponding y scores as Y. Two students i, j are friends if their y scores form an inversion, which means $y_i > y_j$ but $i < j$. So we transform this problem into finding number of inversions in array Y.

We can count number of inversions in the process of mergesort:

(1) Divide array into two halves.

(2) Recursively sort each half and count nuber of inversions each half.

(3) Merge: when comparing two head numbers of each half, if head of right half is added into new array, then number of inversions += number of remaining elements in the left half.

After finished, number of pairs of friends = number of inversions.