473 Levitin problems and hints HW 08

Problem 1: (15) 5.3.8 [4.4.7]

8. a. Draw a binary tree with 10 nodes labeled 0, 1, ..., 9 in such a way that the inorder and postorder traversals of the tree yield the following lists: 9, 3, 1, 0, 4, 2, 7, 6, 8, 5 (inorder) and 9, 1, 4, 0, 3, 6, 7, 5, 8, 2 (postorder).

   b. Give an example of two permutations of the same n labels $0, 1, .., n-1$ that cannot be inorder and postorder traversal lists of the same binary tree.

   c. Design an algorithm that constructs a binary tree for which two given lists of n labels $0, 1, .., n-1$ are generated by the inorder and postorder traversals of the tree. Your algorithm should also identify inputs for which the problem has no solution.

Author's Hints:

8. Find the root's label of the binary tree first, and then identify the labels of the nodes in its left and right subtrees.

Details that I gave on the assignment page:

(a) 3 points. Instead of drawing the tree, you can simply list the order of its preorder traversal.
(b) 2 points. Come up with as small an example as you can.
(c) 10 points. I am changing the input and output specifications from what is given in the problem. The elements in the tree will be characters, not numbers. The same character cannot appear in two different nodes of the tree. An input to the algorithm will be one (even-length) string. The first half of the string is the inorder traversal of a binary tree, and the second half of the string is the postorder traversal of the same tree. Output should be the preorder traversal of the tree.

For definiteness, I will show you my top-level Python code and its output. You can use my code, adapt it to another language, start from scratch in any language, or simply write very clear pseudocode. You must present your algorithm in a way that makes it easy for the grader to determine whether it is correct. Include in your submission at least your code or pseudocode for `buildTree`, and the preorder traversal of the tree built from the string '`itfwGLOAIRsHMTySehtfiGOLIRAwMHyheSTs`'.

Note: At first this may appear to be very complicated, but it does not have to be so. The body of my buildTree function is two lines of code; one of them is a call to the recursive function that actually builds the tree. My recursive procedure's body is 6 lines; preOrder's body is two lines.

Top-level code:

```
def processTraversalString(s):
    try:
        print (preOrder(buildTree(s)))
    except ValueError:
        print ('Inconsistent traversal strings')

processTraversalString('OCDCOD')
processTraversalString('PODYEPOYED')
processTraversalString('ASBLUFHSALUFHB')
processTraversalString('URPYMGUYPMRG')
processTraversalString('bctdcbda')
processTraversalString('BCADECEABD')
processTraversalString('XYTMPAUCLFJKYMAUPTXJFLKC')
```

Problem 2: **(5)** 5.3.11 [4.4.10]

11. *Chocolate bar puzzle* Given an n-by-m chocolate bar, you need to break it into nm 1-by-1 pieces. You can break a bar only in a straight line, and only one bar can be broken at a time. Design an algorithm that solves the problem with the minimum number of bar breaks. What is this minimum number? Justify your answer by using properties of a binary tree.

Author's Hints:

11. Breaking the chocolate bar can be represented by a binary tree.

Problem 3: **(5)** 5.4.9 [4.5.9]

9. V. Pan [Pan78] has discovered a divide-and-conquer matrix multiplication algorithm that is based on multiplying two 70-by-70 matrices using 143,640 multiplications. Find the asymptotic efficiency of Pan's algorithm (you can ignore additions) and compare it with that of Strassen's algorithm.

Author's Hints:

9. The recurrence for the number of multiplications in Pan's algorithm is similar to that for Strassen's algorithm. Use the Master Theorem to find the order of growth of its solution.

Problem 4: **(10)** 5.5.3 [4.6.2]

3. Consider the version of the divide-and-conquer two-dimensional closest-pair algorithm in which, instead of presorting input set P, we simply sort each of the two sets P_l and P_r in nondecreasing order of their y coordinates on each recursive call. Assuming that sorting is done by mergesort, set up a recurrence relation for the running time in the worst case and solve it for $n = 2^k$.

Author's Hints:

3. Recall (see Section 5.1) that the number of comparisons made by mergesort in the worst case is $C_{worst}(n) = n \log_2 n - n + 1$ (for $n = 2^k$). You may use just the highest-order term of this formula in the recurrence you need to set up.

Problem 5: **(5)** 5.5.7 [4.6.6]

7. Explain how one can find point $p_{\max}$ in the quickhull algorithm analytically.

Author's Hints:

2. We traced the algorithms on smaller instances in the section.

Problems 6-7: Not from textbook

6. (5) If the permutations of the numbers 0-7 are numbered from 0 to 8!-1, what is the (lexicographic ordering) sequence number of the permutation 37246510?

Example sequence numbers: 01234567 has sequence number 0, 01234576 has sequence number 1, 01234657 has sequence number 2, …, 76543210 has sequence number 8! - 1.

7. (5) Which permutation of 01234567 is number 25000 in lexicographic order?

Problem 8: (15) 2.4.6 [Not in 2nd ed.]

6. *Restricted Tower of Hanoi* Consider the version of the Tower of Hanoi puzzle in which n disks have to be moved from peg A to peg C using peg B so that any move should either place a disk on peg B or move a disk from that peg. (Of course, the prohibition of placing a larger disk on top of a smaller one remains in place, too.) Design a recursive algorithm for this problem and find the number of moves made by it.

Author's Hints:

6. The required algorithm and the method of its analysis are similar to those of the classic version of the puzzle. Because of the additional constraint, more than two smaller instances of the puzzle need to be solved here.

Problems 9-11: (5, 10, 6) 4.5.4-4.5.6 [5.6.4-5.6.6]

4. Derive the formula underlying interpolation search.

5. ▷ Give an example of the worst-case input for interpolation search and show that the algorithm is linear in the worst case.

6. a. Find the smallest value of n for which $\log_2 \log_2 n + 1$ is greater than 6.

   b. Determine which, if any, of the following assertions are true:

   i. $\log \log n \in o(\log n)$ ii. $\log \log n \in \Theta(\log n)$ iii. $\log \log n \in \Omega(\log n)$.

Author's Hints:

4. Write an equation of the straight line through the points $(l, A[l])$ and $(r, A[r])$ and find the x coordinate of the point on this line whose y coordinate is v.

5. Construct an array for which interpolation search decreases the remaining subarray by one element on each iteration.

6. a. Solve the inequality $\log_2 \log_2 n + 1 > 6$.

   b. Compute $\lim_{n\to\infty} \frac{\log \log n}{\log n}$. Note that to within a constant multiple, you can consider the logarithms to be natural, i.e., base e.

Claude's notes:

4.5.5 Show that the worst case is Θ(N)

4.5.6 For part b, use the "limit of the ratio" approach