COSC 2123/1285 Algorithms and Analysis
Tutorial 6
Transform and Conquer Algorithmic Paradigm

Objective

Students who complete this tutorial should:

- Be familiar with the three major variations of transform-and-conquer.
- Be able to apply transform-and-conquer strategies to different problem type

---

Questions

6.1.5 To sort or not to sort? Design a reasonably efficient algorithm for solving each of the following problems and determine its efficiency class.

a You are given n telephone bills and m checks sent to pay the bills ($n \geq m$). Assuming that telephone numbers are written on the checks, find out who failed to pay. (For simplicity, you may also assume that only one check is written for a particular bill and that it covers the bill in full.)

b You have a file of n student records indicating each student's number, name, home address, and date of birth. Find out the number of students from each of the 50 U.S. states.

Answer:

a The following algorithm will beat the brute-force comparisons of the telephone numbers on the bills and the checks: Using an efficient sorting algorithm, sort the bills and sort the checks. (In both cases, sorting has to be done with respect to their telephone numbers, say, in increasing order.) Then do a merging-like scan of the two sorted lists by comparing the telephone numbers b_i and c_j on the current bill and check, respectively: if $b_i < c_j$,add b_i to the list of unpaid telephone numbers and increment i; if $b_i > c_j$, increment j ; if $b_i = c_j$, increment both i and j. Stop as soon as one of the two lists becomes empty and append all the remaining telephone numbers on the bill list, if any, to the list of the unpaid ones.

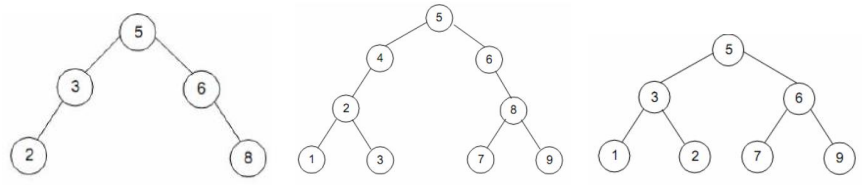
The time efficiency of this algorithm will be in:

$$\underbrace{O(n\log(n))}_{\text{sort numbers}} + \underbrace{O(m\log(m))}_{\text{sort bills}} + \underbrace{O(n+m)}_{\text{algorithm cost}} \overbrace{=}^{n\geq m} O(n\log(n))$$

This is superior to the $O(nm)$ efficiency of the brute-force algorithm (but inferior, for the average case, to solving this problem with hashing).

b Initialize 50 state counters to zero. Scan the list of student records and, for a current student record, increment the corresponding state counter. The algorithm's time efficiency will be in $T(n)$, which is superior to any algorithm that uses presorting of student records by a comparison-based algorithm.

6.3.1 Which of the following binary tress are AVL trees?

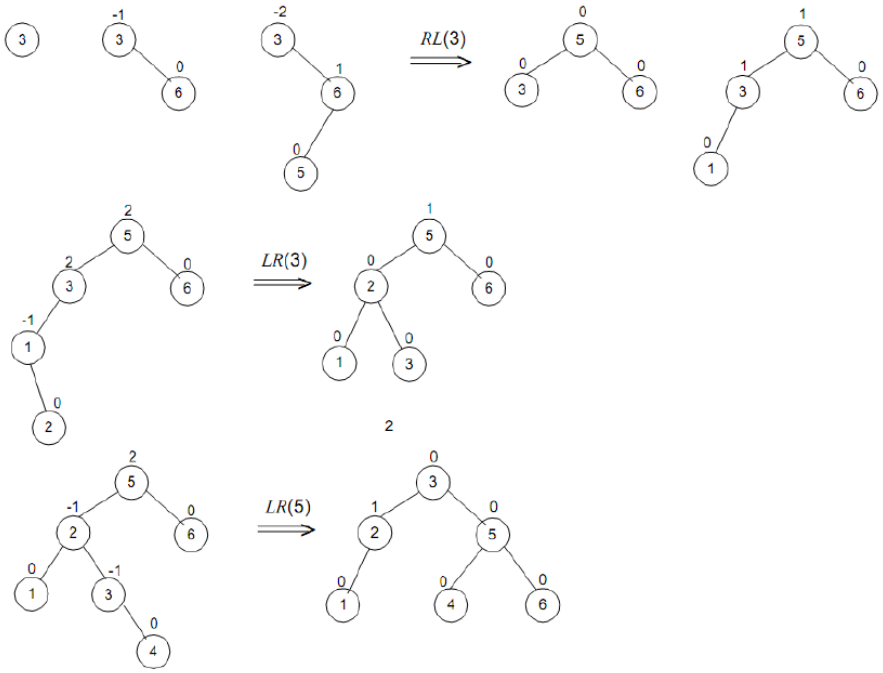


(a) (b) (c)

Answer: Only (a) is an AVL tree; (b) has a node (in fact, there are two of them: 4 and 6) that violates the balance requirement; (c) is not a binary search tree because 2 is in the right subtree of 3 (and 7 is in the left subtree of 6).

6.3.4 Construct an AVL tree for the list `3, 6, 5, 1, 2, 4`.

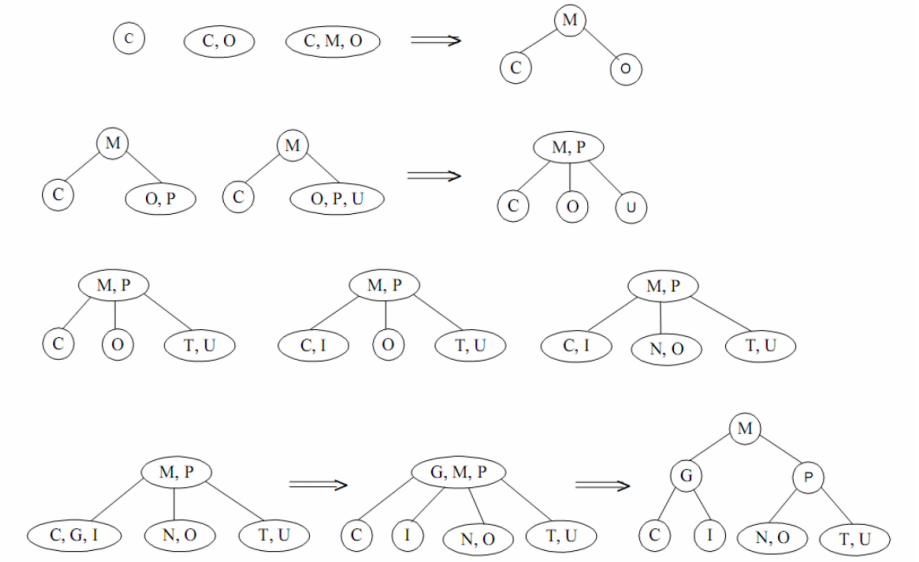
Answer:



6.3.7a Construct a 2–3 tree for the list `C, O, M, P, U, T, I, N, G` (Use the alphabetical order of the letters and insert them successively starting with the

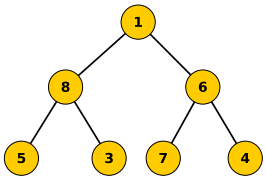
empty tree.)

Answer:

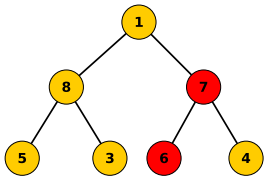


6.4.1 Construct a heap for the list `1, 8, 6, 5, 3, 7, 4` using the algorithm described in the lecture notes.

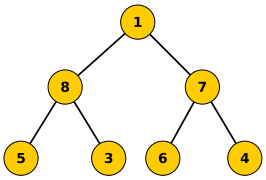
Answer:



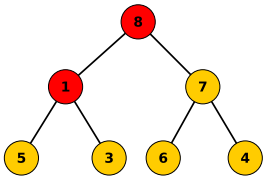
(a) Initial



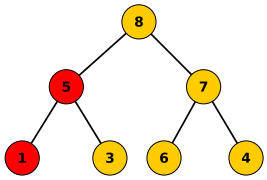
(b) Update parent node 6



(c) Consider parent node 8 but no changes



(d) Update parent node 1



(e) Propagate and update parent node 1