Winter 2019 CS 32

Homework 5 Solution

Problem 1:

a.

```
                    50

          20                  60

    10          40                     70

      15    30                     65       80

           25  35                             75
```

b.

```
   In-order: 10 15 20 25 30 35 40 50 60 65 70 75 80
  Pre-order: 50 20 10 15 40 30 25 35 60 70 65 80 75
 Post-order: 15 10 25 35 30 40 20 65 75 80 70 60 50
```

c. One possibility is

```
                    50

          25                  60

    10          40                     70

      15    35                     65       80

                                              75
```

Other possibilities have the left subtree of 50 being

```
          15

    10          40

             35

           25
```

or

```
          15

    10          40

             25

               35
```

Problem 2:

a.

```
struct Node
{
    int data;
    Node* left;
    Node* right;
    Node* parent;
};
```

b.

```
void insertAuxiliary(Node*& n, int value, Node* par)
{
    if (n == nullptr)
        set n to point to a new Node whose data field is set to value,
          whose left and right children are null, and whose parent field
          is set to par.
    else if (value < n->data)
        insertAuxiliary(n->left, value, n);
    else
        insertAuxiliary(n->right, value, n);
}

void insert(Node*& n, int value)
{
  insertAuxiliary(n, value, nullptr);  // pass nullptr as parent of root
}
```

Problem 3:

a.

```
    8

  3   6

0 2   4
```

b. `8 3 6 0 2 4`

c. `6 3 4 0 2`

Problem 4:

a. O(C + S). We'd have to do a linear search through the outer vector to find the course, which is O(C), and then after that do a linear search of the S students in the list, which is O(S). We can't do a binary search in a linked list in logarithmic time, because it takes linear time just to get to the middle item of a list, sorted or not.
b. O((log C) + S). A BST-based map finds the course in O(log C) time, and after that we do a linear search of the S students in the list.
c. O(log C + log S). A BST-based map finds the course in O(log C) time, and a BST-based set finds the student in O(log S) time. Notice that a mathematically equivalent way to write this is O(log CS), but that form makes it harder to understand why it's right.
d. O(log S). A hash-based map finds the course in O(1) time, and a BST-based set finds the student in O(log S) time. Notice that a constant like 1 is dominated by a function that grows with S, so we write O(log S) instead of O(1 + log S).
e. O(1). A hash-based map finds the course in O(1) time, and a hash-based set finds the student in O(1) time.
f. O((log C) + S). A BST-based map finds the course in O(log C) time, and we can visit all S items in order in a BST-based set in O(S) time.
g. O(S log S). A hash-based map finds the course in O(1) time, and although we can visit all S items in a hash-based set in O(S) time, they'd be in whatever order the hash function caused them to be scattered across the buckets. We have to sort them. It takes O(S) steps to get them into and out of an auxiliary vector, say, and O(S log S) to sort them.

h. O(C log S). The hash-based map is keyed on the course, not the student, so it's not organized to look up students efficiently. Our only choice is to check every course and see in O(log S) time whether the student is in its set of students. Since we have to do the O(log S) operation for each of the C courses, it's O(C log S).