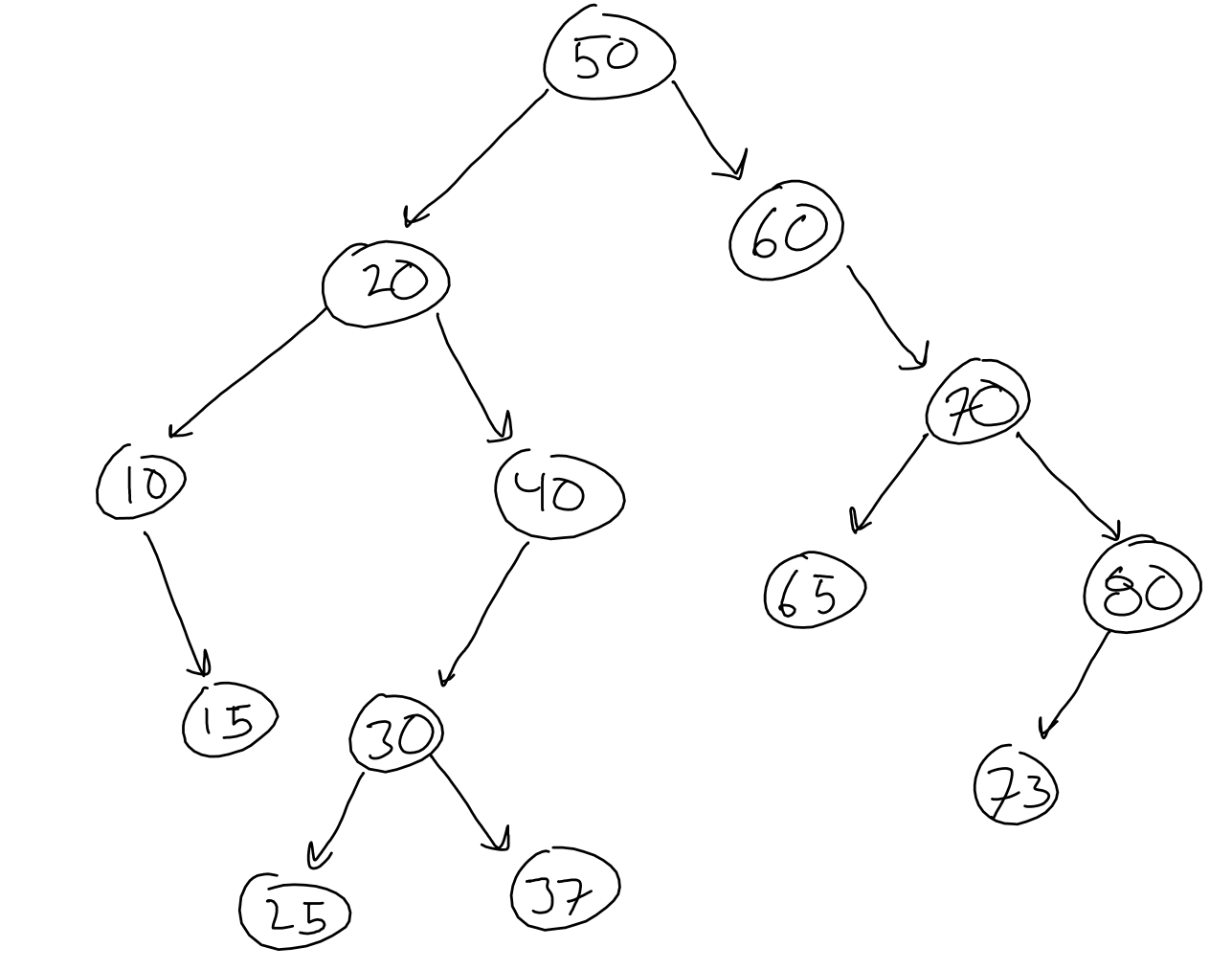
Angel Jimenez

604-667-905

Homework 5

CS 32 – Spring 18

**Problem 1a:**



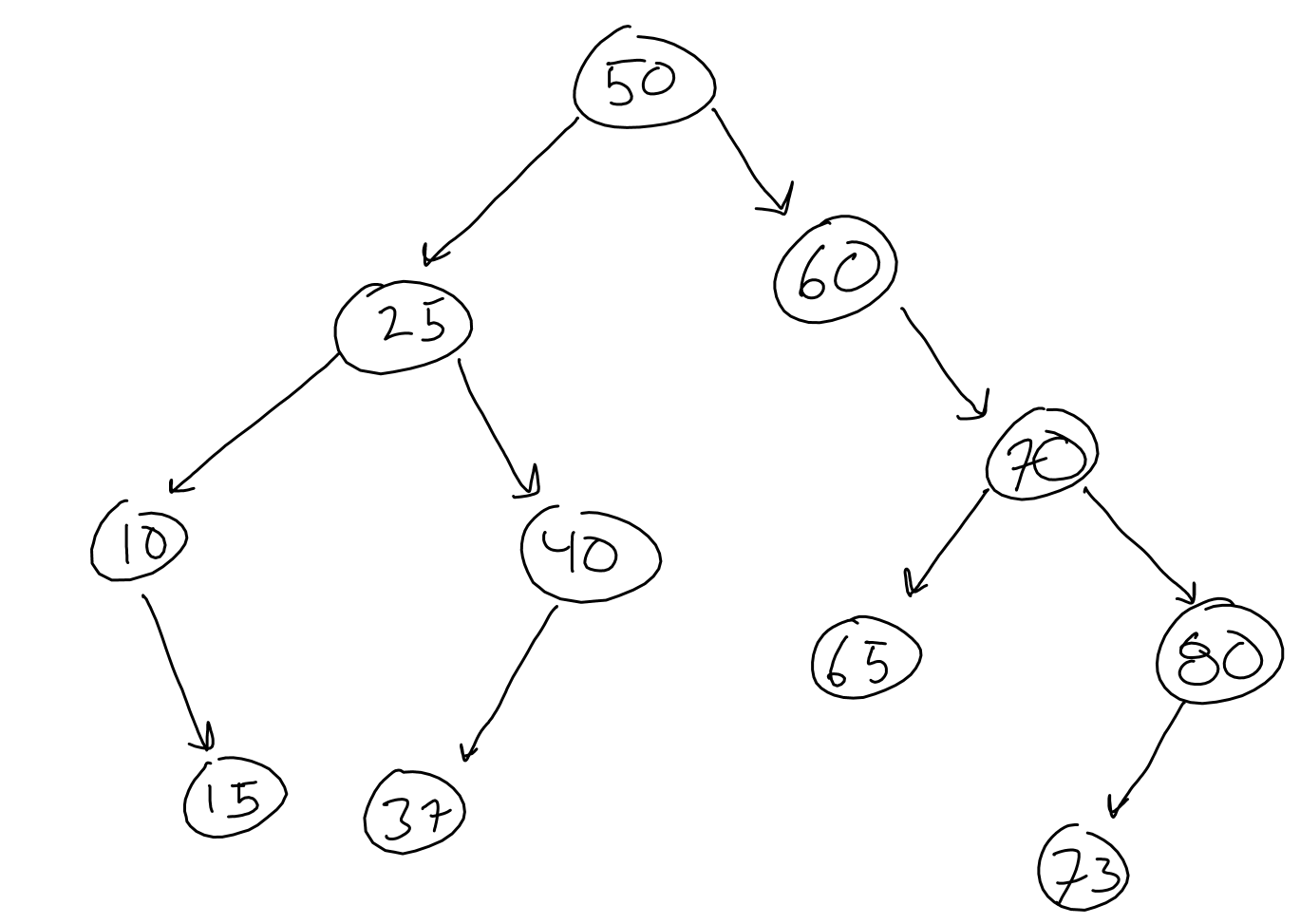
**Problem 1b:**

In-Order: 10, 15, 20, 25, 30, 37, 40, 60, 65, 70, 73, 80  
( left, middle, right)

Pre-Order: 50, 20, 10, 15, 40, 30, 25, 37, 60, 70, 65, 80, 73  
( middle, left, right)

Post-Order: 15, 10, 25, 37, 30, 40, 20, 65, 73, 80, 70, 60, 50  
( left, right, middle)

**Problem 1c:**



When we delete 20, we replace it with a leaf that is greater than 10 and less than 40

When we delete 30, we do the same thing.

**Problem 2a:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | struct Node  {          Node\* m\_left;          Node\* m\_right;          Node\* m\_parent;          int m\_data;            Node(Node\* parent, int data)                  : m\_data(data)                  {m\_left = m\_right = m\_parent = nullptr}          // not necessarily how I would do it          // but this looks cool  }; |

**Problem 2b:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | void treeInsert( Node\* parent, Node\* curr, int value)  {          if curr ptr is nullptr                  create new node                  set nodes data to value                  set parent pointer to previous node                  left and right child are set to nullptr          else if (value is less than curr->value)                  recursive call insert(curr, leftchild, value)          else if (vale us greater than curr->value)                  recursive call insert(curr, rightchild, value)  } |

Problem 3a:

6

4

2

0

3

8

**Problem 3b:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 8 | 3 | 6 | 0 | 2 | 4 |

**Problem 3c:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 0 | 2 |

**Problem 4a:**

Vector<pair<string, list<int>>>. Outer is course, inner is students in class, students sorted in order. No order in outer vector

Looking through classes is O(C), looking through students is O(S) therefore O(C + S)

**Problem 4b:**

Map<string, list<int>>, students in list are not ordered. Big O of finding student S in course C?

Classes are stored in the map(in this case is BST). Searching through classes is O(log(C)), then looking through S students takes O(S) time, therefore together: O(log(C) + S)

**Problem 4c:**

Map<string, list<int>>, searching the classes takes O(log(C)) due to the BST. Searching through a set (also bst) of students takes O(log(S)). Therefore, combined is O(log(C) + log(S))

**Problem 4d:**

Unordered\_map<string, set<int>>. Finding particular student in course?

Unordered\_maps are organized as hash tables, thus O(1). Looking through a BST set is O(log(S)). Therefore, the time complexity is O(1+log(S)) which is O(log(S))

**Problem 4e:**

Unordered\_map<string, unordered\_set<int>>. Find particular student in course c? O(1) to search the class. Unordered sets are also organized as hash tables thus taking O(1) as well. In total, O(1+1) which is O(1)

**Problem 4f:**

Map<string, set<int>>. A BST map takes O(log(C)). For a BST ordered set, it takes O(S) to search. Therefore in total, it takes O(log(C) + S)

**Problem 4g:**

Unordered\_map<string, unordered\_set<int>> . It takes the hash table map O(1) to find the course, then we have to sort the items in the student set, which takes O(Slog(S)) time. To sort them we have to push them into a vector which takes O(S). Therefore we have O(S log(S) + S) which is just O(Slog(S))

**Problem 4h:**

Unoredered\_map<string, set<int>>. We have to search all the classes and then search all the students to find that particular student. We cannot take advantage of the map properties because it’s a hash based map and we cannot perform the reverse search. Thus, O(log(S)) to search the student in the set of students, and we have to check all courses which is O(C). Thus we get O(Clog(S))