CS 32: Discussion 1D

TA: Shichang Zhang
LA: Stephanie Doan, Rish Jain

Administrative

1) FINAL ON Sunday, June 6, from 6:30 pm to 8:30 pm PST
2) Evaluations are due Saturday Morning, please fill it out

Final Review Questions

Credit: Spring 19 LA Final Review Workshop

Arabelle Siahaan

Varun Raju

Evan Lin

Question 1: Big-O

Given: `hash_map<student, set<class>> studentClass`

There are S students and an average of C classes per student.

1. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically?
2. What is the big-O complexity of determining who's taking CS32?
3. What is the big-O complexity of determining if Joe Smith is taking CS32?
4. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity?

Question 1: Big-O

Given: **`hash_map<student, set<class>> studentClass`**

There are S students and an average of C classes per student.

1. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically? **`SlogS + SC`**
2. What is the big-O complexity of determining who's taking CS32?
3. What is the big-O complexity of determining if Joe Smith is taking CS32?
4. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity?

Question 1: Big-O

Given: `hash_map<student, set<class>> studentClass`

There are S students and an average of C classes per student.

1. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically? **`SlogS + SC`**
2. What is the big-O complexity of determining who's taking CS32? **`SlogC`**
3. What is the big-O complexity of determining if Joe Smith is taking CS32?
4. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity?

Question 1: Big-O

Given: **`hash_map<student, set<class>> studentClass`**

There are S students and an average of C classes per student.

1. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically? **`SlogS + SC`**
2. What is the big-O complexity of determining who's taking CS32? **`SlogC`**
3. What is the big-O complexity of determining if Joe Smith is taking CS32? **`logC`**
4. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity?

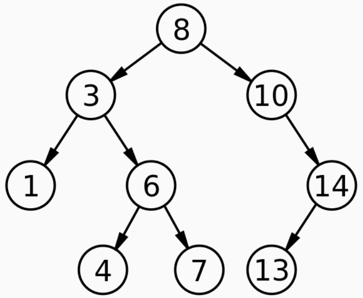
Question 1: Big-O

Given: `hash_map<student, set<class>> studentClass`

There are S students and an average of C classes per student.

1. What is the big-O complexity of printing all students alphabetically and for each student, listing each of their classes alphabetically? **`SlogS + SC`**
2. What is the big-O complexity of determining who's taking CS32? **`SlogC`**
3. What is the big-O complexity of determining if Joe Smith is taking CS32? **`logC`**
4. Suppose we have the data structure hash_map<string, set<int>> and we wish for a particular class L to print the ID numbers of all the students in that course in sorted order (assume there are S students in a class). What is the big-O complexity? **`S`**

Question 2: Binary Trees



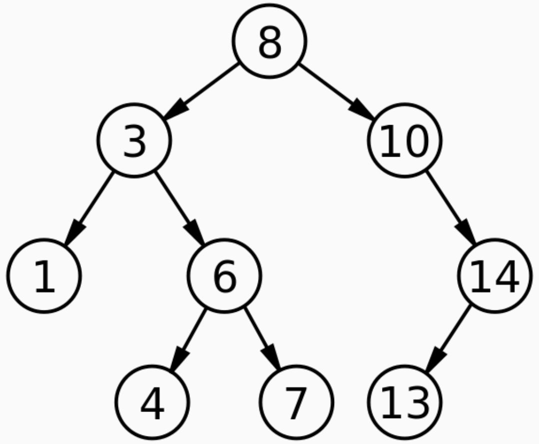
1. What are the preorder and inorder traversal for the tree?
2. Using the simplest binary search tree (BST) insertion algorithm (no balancing), what is the resulting tree after inserting the nodes 80, 65, 71, 15, 39 and 25, then deleting 10, 6 and 39 in that order.
3. What is the postorder traversal for the resulting tree in part 2?
4. Given the following postorder and inorder traversal of a binary tree (not necessarily a BST), draw the corresponding tree:

INORDER: 1 5 0 8 4 3 7 6 2

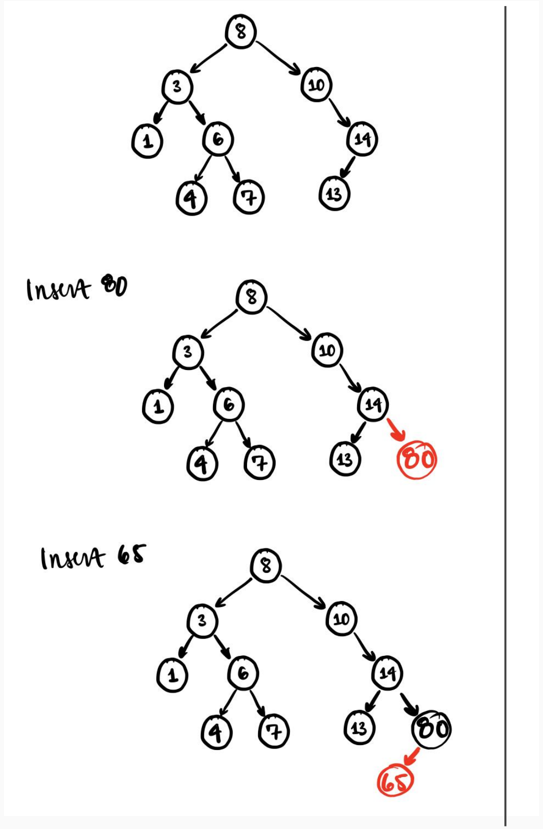
POSTORDER: 1 8 4 0 5 2 6 7 3

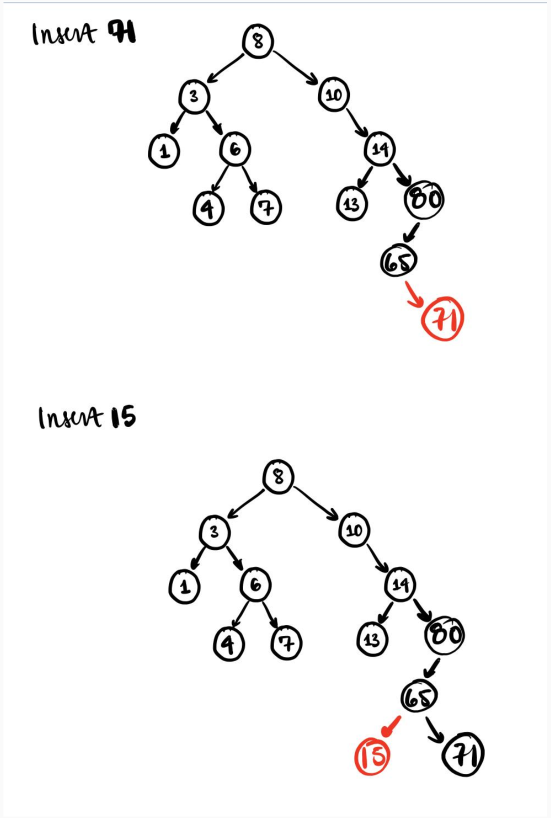
Question 2: Binary Trees

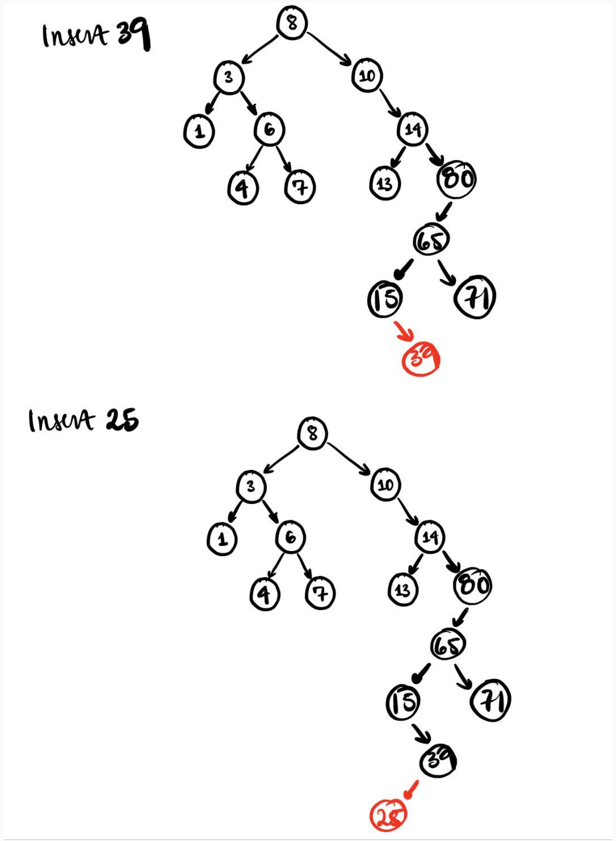
1. What are the preorder and inorder traversal for the tree?
   Preorder: 8, 3, 1, 6, 4, 7, 10, 14, 13
   Inorder: 1, 3, 4, 6, 7, 8, 10, 13, 14



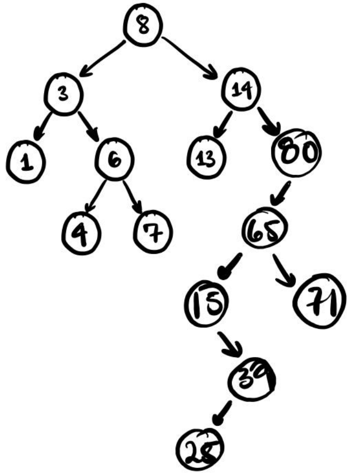
2. Using the simplest binary search tree (BST) insertion algorithm (no balancing), what is the resulting tree after inserting the nodes 80, 65, 71, 15, 39 and 25, then deleting 10, 6 and 39 in that order.



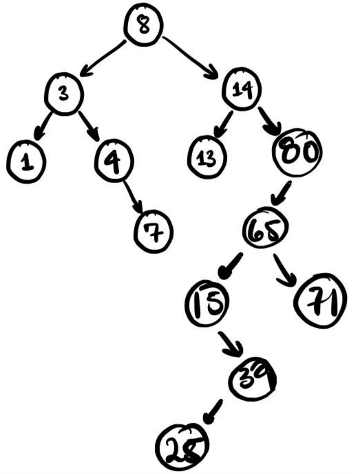




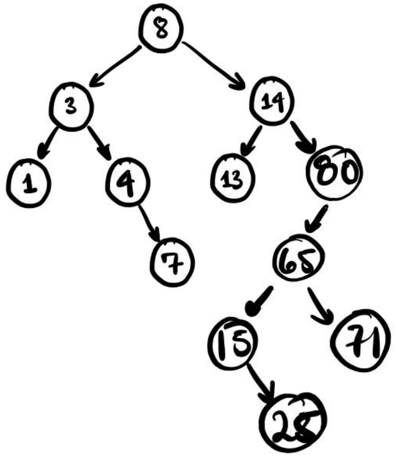
Delete 10



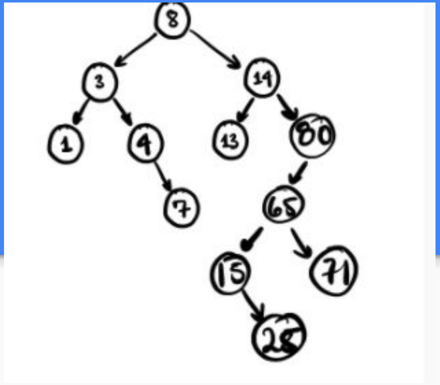
Delete 6



Delete 39



Question 2: Binary Trees



3. What is the postorder traversal for the resulting tree in part 2?
Postorder: 1, 7, 4, 3, 13, 25, 15, 71, 65, 80, 14, 8

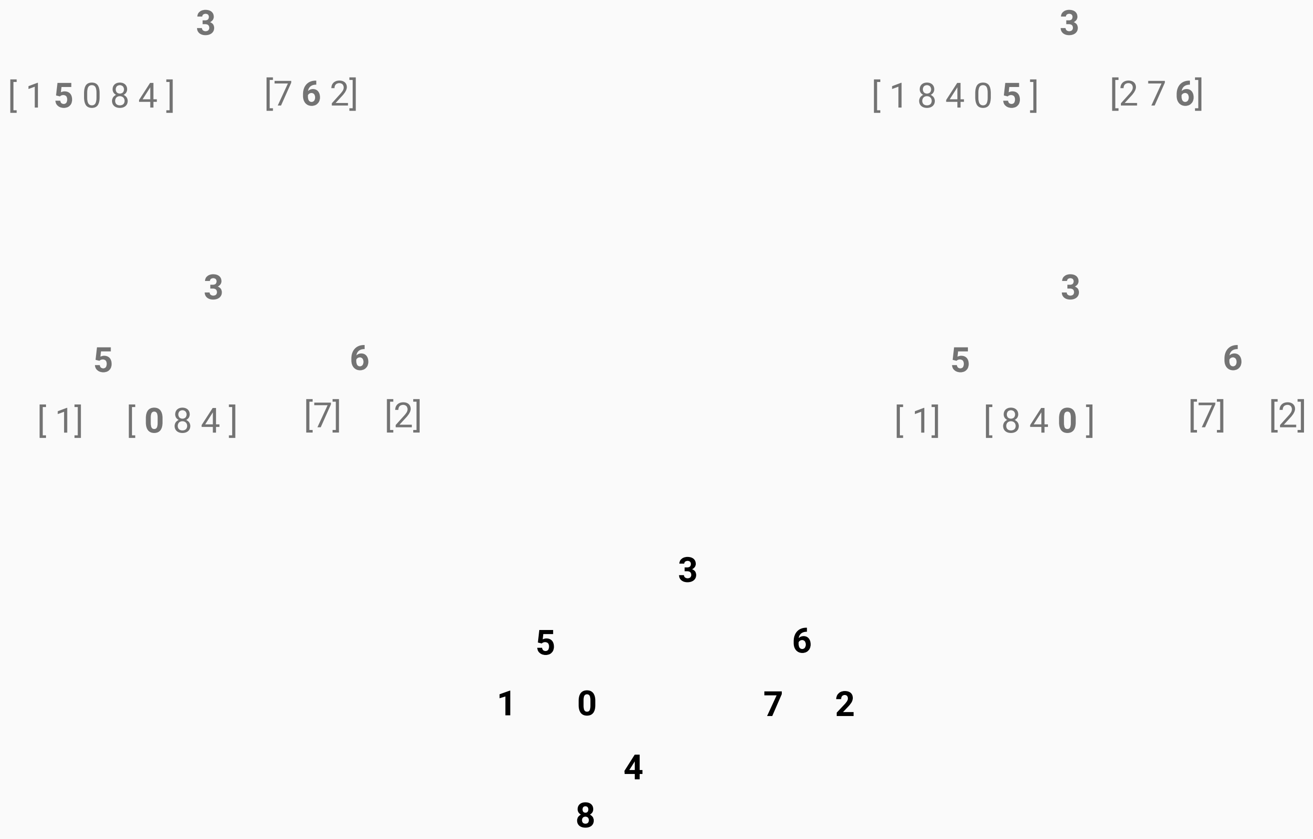
4. Given the following postorder and inorder traversal of a binary tree (not necessarily a BST), draw the corresponding tree:

INORDER: 1 5 0 8 4 3 7 6 2

POSTORDER: 1 8 4 0 5 2 6 7 3

INORDER: 1 5 0 8 4 **3** 7 6 2

POSTORDER: 1 8 4 0 5 2 6 7 **3**



Question 3: Time Complexity

```
void func(int arr[], int n){
   set<int> s;
   for (int i=0;i<n;i++){
       for (int j=0;j<i;j++)
            s.insert(arr[i] * arr[j]);
       }
}
```

1. What is the time complexity of the code?
2. If we implement s with an unordered map instead, what is the time complexity of the code?
3. What if we just implement it using an array that’s sorted? What is its time complexity?

Question 3: Time Complexity

```
void func(int arr[], int n){
   set<int> s;
   for (int i=0;i<n;i++){
       for (int j=0;j<i;j++)
            s.insert(arr[i] * arr[j]);
       }
}
```

1. What is the time complexity of the code? `O(N^2 logN)`
2. If we implement s with an unordered map instead, what is the time complexity of the code?
3. What if we just implement it using an array that's sorted? What is its time complexity?

Question 3: Time Complexity

```
void func(int arr[], int n){
   set<int> s;
   for (int i=0;i<n;i++){
       for (int j=0;j<i;j++)
            s.insert(arr[i] * arr[j]);
       }
}
```

1. What is the time complexity of the code? `O(N^2 logN)`
2. If we implement s with an unordered map instead, what is the time complexity of the code?

   `O(N^2)`
3. What if we just implement it using an array that's sorted? What is its time complexity?

Question 3: Time Complexity

```
void func(int arr[], int n){
    set<int> s;
    for (int i=0;i<n;i++){
        for (int j=0;j<i;j++)
            s.insert(arr[i] * arr[j]);
        }
}
```

1. What is the time complexity of the code? `O(N^2 logN)`
2. If we implement s with an unordered map instead, what is the time complexity of the code?

   `O(N^2)`
3. What if we just implement it using an array that's sorted? What is its time complexity?

   `O(N^3)`

Question 4: Sorting

1. Given the following array of integers {-4, 10, 8, 2, -7, 3, 1, 0}. Show the resulting array after three iterations of a) insertion sort, b) bubble sort, and c) quicksort with the first element as the pivot

Question 4: Sorting

1. Given the following array of integers {-4, 10, 8, 2, -7, 3, 1, 0}. Show the resulting array after three iterations of a) insertion sort, b) bubble sort, and c) quicksort with the first element as the pivot

a) Insertion sort:

```
-4 10 8 2 -7 3 1 0
-4 8 10 2 -7 3 1 0
-4 2 8 10 -7 3 1 0
```

b) Bubble sort:

```
-4 10 8 2 -7 3 1 0
-4 8 10 2 -7 3 1 0
-4 8 2 10 -7 3 1 0
```

c) Quicksort (by swapping):

```
-7 -4 8 2 3 1 0 10   // initial pivot -4
-7 -4 2 3 1 0 8 10   // pivot1 = -7, pivot2 = 8
-7 -4 1 0 2 3 8 10   // pivot1 = -7, pivot2 = 2, pivot3 = 10
```

Question 4: Sorting

2. Now given the following array of integers {-2, -1, 0, 3, 10, 5, 7, 9}. Which sorting algorithm is most efficient for you to use? Insertion sort, merge sort or heapsort?

Question 4: Sorting

2. Now given the following array of integers {-2, -1, 0, 3, 10, 5, 7, 9}. Which sorting algorithm is most efficient for you to use? Insertion sort, merge sort or heapsort?

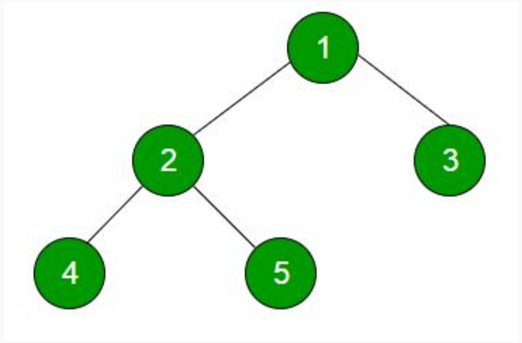
Insertion sort is the best algo for nearly sorted arrays, and arrays with small number of elements

Question 5: Recursion & Trees

Given a binary tree, write a code that returns all root-to-leaf paths. Use the function header:

vector<string> rootToLeaf(Node* head)

<u>Example</u>:



All root-to-leaf paths are: `["1->2->4", "1->2->5", “1->3]`

Note: Use this Node definition

```
struct Node {

        int val;

        Node* left, right;

};
```

Question 5: Recursion & Trees

```
void rootToLeaf(Node* root, vector<std::string>& paths, std::string curr) {
        if (root->left == nullptr  &&  root->right == nullptr) {
                paths.add(curr);
                return;
        }
        if (root->left != nullptr) {
                rootToLeaf(root->left, paths, curr + "->" + std::to_string(root->left->val));
        }
        if (root->right != nullptr) {
                rootToLeaf(root->right, paths, curr + "->" +  std::to_string(root->right->val));
        }
}
vector<std::string> rootToLeaf(Node* root) {
        vector<std::string> paths;
        if (root != nullptr)
                rootToLeaf(root, paths, std::to_string(root->val));
        return paths;
}
```

Question 6: Stacks & Queues

1. **Consider the following code:**

```
void mystery(queue<int>& q) {
    if (q.empty()) {return;}
    int data = q.front();
    q.pop();
    mystery(q);
    q.push(data);
}
```

Given the queue containing [3, 7, 5, 9, 2, 0, 8]

a. **What is the queue being passed in on the first and last recursive call?**
b. **What does this function do?**

Question 6: Stacks & Queues

1. **Consider the following code:**

```
void mystery(queue<int>& q) {
    if (q.empty()) {return;}
    int data = q.front();
    q.pop();
    mystery(q);
    q.push(data);
}
```

Given the queue containing [3, 7, 5, 9, 2, 0, 8]

a. **What is the queue being passed in on the first and last recursive call?**

b. **What does this function do?**

a) Calls mystery on the queue [7,5,9,2,0,8] on the first recursive call, and the queue [] on the last recursive call

b) It reverses the queue

Question 6: Stacks & Queues

2. Evaluate the following *prefix* expression using integer arithmetic. The answer is a single integer:

+ / * 2 - 8 2 4 * + 1 0 - 6 2

Question 6: Stacks & Queues

2. Evaluate the following *prefix* expression using integer arithmetic. The answer is a single integer:

+ / * 2 - 8 2 4 * + 1 0 - 6 2

To evaluate prefix expressions:

1. **Reverse to be postfix, i.e. 2 6 - 0 1 + * 42 8 - 2 * / +**
2. **Push every digit into the stack, and once you hit an operator, evaluate the first 2 digits on the stack and push the result back**

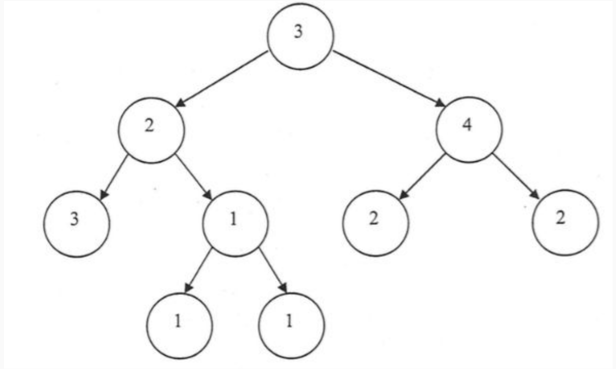
- [4]
+ [4, 1]
*** [4]**
- [4, 4, 6]
*** [4, 4, 12]**
/ [4, 3]
+ [7]

FINAL ANSWER: 7

Question 7: Trees & Recursion (10 mins)

We say that a tree is balanced if the tree is empty, or the *weight* (defined to be the total of the vals in the nodes of the tree) of the left and right subtrees are equal, and each of those subtrees is itself in balance. Write a function called isInBalance that takes a pointer to the root node of a tree and returns true if the tree is balanced, or false otherwise. The node is defined as:

```
struct Node{

        int val;

        Node* left, right;

};
```



You may write additional functions, but you must not use any global variables, not any variables other than `bool`, `bool&`, `int`, `int&`, or `Node*`. You must not use the keywords `while`, `for`, `goto` or `static`. Your solution shouldn't be more than 30 lines of code. Use the function header: `bool isInBalance(Node* root)`

Question 7: Trees & Recursion (10 mins)

// helper func that calculates the sum of a tree recursively

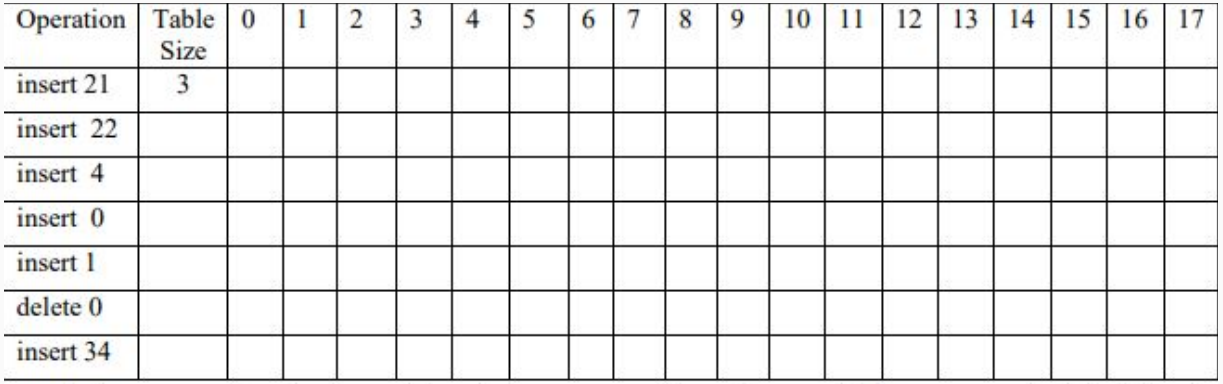
```
int sum(struct node *root)
{
   if(root == NULL)
     return 0;
   return sum(root->left) + root->data +
sum(root->right);
}
```

```
bool isInBalanced(Node* root){

    int ls, rs;

    /* If node is NULL or it's a leaf node then return true */
    if(node == NULL ||
            (node->left == NULL && node->right == NULL))
        return true;

   /* Get sum of nodes in left and right subtrees */
   ls = sum(node->left);
   rs = sum(node->right);

   /* if the node and both of its children satisfy the property return 1 else
0*/
    if((ls == rs)&&
            isInBalance(node->left) &&
            isInBalance(node->right))
        return true;

   return false;
}
```

Question 8: Hash Table

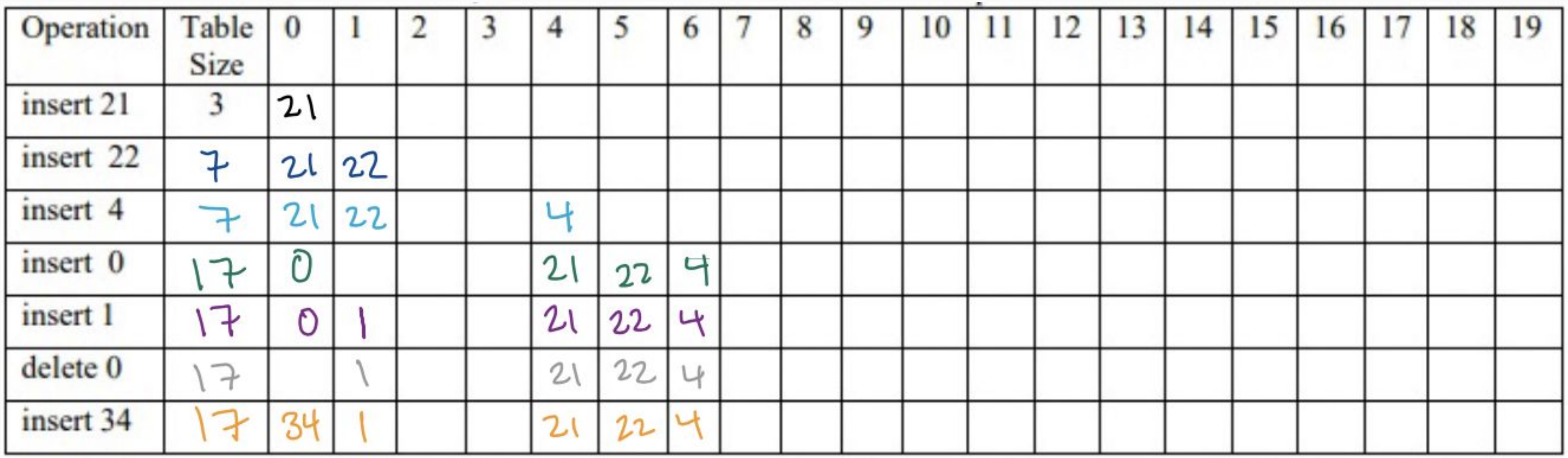
Given that:

- Hash Table uses Linear Probing
- Rehash rules: double the table size and find the next prime number.
    - i.e) 3 * 2 = 6, next prime number is 7
- Maximum Load factor is 0.5
- Hash Function is just modulo Table size



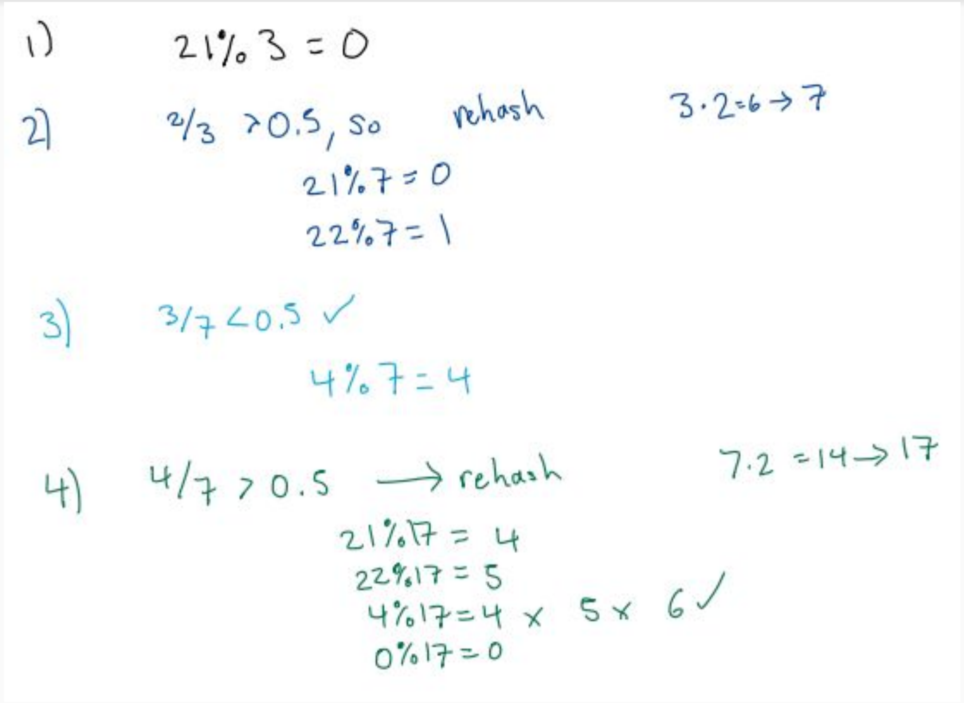
Operation	Table Size	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
insert 21	3																		
insert 22																			
insert 4																			
insert 0																			
insert 1																			
delete 0																			
insert 34																			

Hash Table solution



Operation	Table Size	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
insert 21	3	21																			
insert 22	7	21	22																		
insert 4	7	21	22			4															
insert 0	17	0				21	22	4													
insert 1	17	0	1			21	22	4													
delete 0	17		1			21	22	4													
insert 34	17	34	1			21	22	4													

Hash Table solution (Step by Step)



1) $21 \% 3 = 0$

2) $2/3 > 0.5$, so rehash $3 \cdot 2 = 6 \to 7$

$21 \% 7 = 0$

$22 \% 7 = 1$

3) $3/7 < 0.5$ ✓

$4 \% 7 = 4$

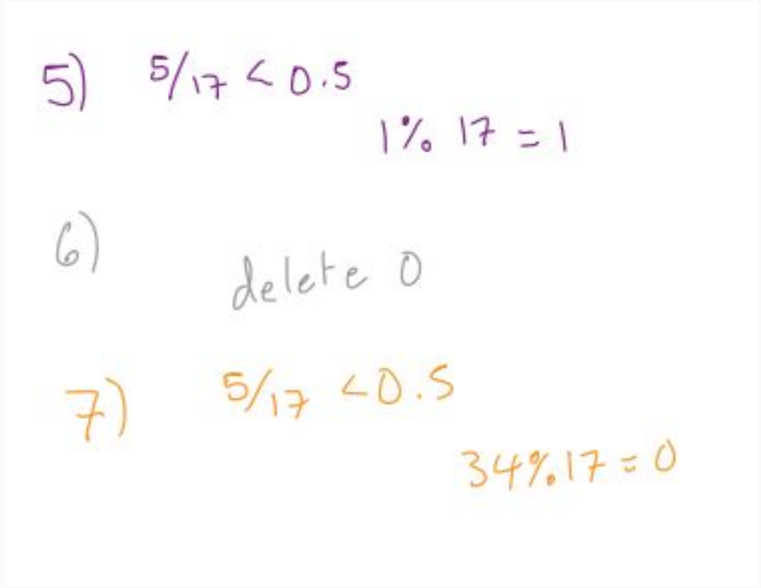
4) $4/7 > 0.5 \longrightarrow$ rehash $7 \cdot 2 = 14 \to 17$

$21 \% 17 = 4$

$22 \% 17 = 5$

$4 \% 17 = 4$ × 5 × 6 ✓

$0 \% 17 = 0$



5) $5/17 < 0.5$

$1 \% 17 = 1$

6) delete 0

7) $5/17 < 0.5$

$34 \% 17 = 0$

Best Luck to Everyone!
Q & A