Algorithms and Data Structures
Midterm Review
Part (2)

Keyi Yuan
Teaching assistant
Nov.2nd 2019

**Adapted from CS101 Midterm Review (Made by TA Keyi Yuan), 2018 Fall, ShanghaiTech University*

Agenda

Agenda

- Time:
  - Nov.7th, 8:15-10:10, 115 minutes in total
- Covers:
  - The lectures from *Lecture 1 Introduction* to *Lecture 14 Topological Sort*
  - The discussions from *week 2* to *week 9.*
- Policy:
  - Closed book. No cheating sheet.
  - No electronic devices. For example, calculators, PC, smartphones and so on.
  - And any other actions that violate the rule of the exam will be judged as plagiarism and zero score immediately.

Agenda

- Sample is released on Piazza.
- Covers:
  - 6 Multiple Choices (Choose one correct answer).
  - 12 Multiple Choices (Choose all correct answers): You should notice that if you choose the subset of the correct solution except choosing nothing can get the half point of this question.
  - 10 True or False.
  - 4 General Problems.
- OH has been adjusted.

Date	Time	Location	TA
Nov.2nd(Saturday)	21:15-22:15	SIST 1B-105	张雯、张尧
Nov.5th(Tuesday)	15:00-16:00	SIST 1B-105	王有佳、郑悦
	18:20-19:20	SIST 1B-105	许惟锴、王乐童
Nov.6th(Wednesday)	18:00-19:00	SIST 1B-105	沈尧、吕世栋
	21:00-23:00	Dormitory #9, First Floor (学生公寓9号楼一楼研讨室)	袁可一、徐寅韬、郑临风

Claim

1. Topics that reviewed in this discussion may not be covered in the Exam
2. Topics that not reviewed in this discussion may be covered in the Exam
3. ③ As we all know that Statements start with "keyi yuan thinks" usually won't be true. (X

What you have learned?

- Tree Structure
  - Binary Search Tree
    - **All objects in the left sub-tree to be less than the object stored in the root node.**
    - **All objects in the right sub-tree to be greater than the object in the root object.**
    - The two sub-trees are themselves binary search trees. (Recursive definition)
    - Unfortunately, the worst case is equivalent to a linked list, $O(n)$.
    - Insert: Average $O(\log n)$, only to find where to insert is enough.
    - Erase: Average $O(\log n)$, according to the following three cases:
      1. The node is a leaf node.
      2. It has exactly one child
      3. It has two children (it is a full node): Replace the current value with the minimum object in the right sub-tree, then erase that object from the right sub-tree.
    - Given pre-order, you can find a unique BST.

What you have learned?

- Tree Structure
  - AVL Tree
    - **Actually, it is a binary search tree.**
    - The difference in the heights between the left and right sub-trees is at most 1, and both sub-trees are themselves AVL trees. (Recursive definition)
    - Using formula of Fibonacci number, we can get the lower bound of nodes.
    - **Maintaining balance: LL / RR and LR / RL. (Important!!!)**
    - Insert: only cause one imbalance that must be fixed, so need Θ(1) time to correct. [Assuming we have found where to insert. For whole insertion, please see it in BST]
    - Erase: may cause O(h) imbalances that must be corrected, so need O(ln(n)) time.
    - You should concerned about how to maintain the balance of such a tree.

What you have learned?

- Disjoint Set
  - Disjoint Set
    - Important Function: find(), set_union(int i, int j)
    - Poor implementation: Using two arrays, but taking the union of two sets is Θ(n), because It would be necessary to check each array entry.
    - Optimization 1: Point the root of the shorter tree to the root of the taller tree, The height and average depth of the worst case are O(ln(n)). Time complexity: Θ(1).
    - Optimization 2: Path Compression. Whenever find is called, update the object to point to the root. The next call to find(n) is Θ(1), but the cost is O(h) memory.

If **a** and **b** are in the same set, **`find( a ) == find( b )`**

If **a** and **b** are not in the same set, **`find( a ) != find( b )`**

What you have learned?

- Graph
  - Definitions
    - Kind: Undirected graph, Directed graph.
    - Degree: the number of adjacent vertices. Path: an ordered sequence of vertices $(v_0, v_1, v_2, \ldots, v_k)$
    - Simple path: no repetitions.
    - Tree: if it is **connected** and there is a unique path between any two vertices. N nodes with n-1 edges. Adding one more edge must create a cycle. Removing any one edge creates two unconnected sub-graphs.
    - Forest: any graph that has no cycles.
    - Sources, Sinks
    - In/Out degree

$$|E| \le 2\binom{|V|}{2} = 2\frac{|V|(|V|-1)}{2} = |V|(|V|-1) = O(|V|^2)$$

What you have learned?

- Graph Representation
  - Adjacency matrix
    - Memory: $\Theta(n^2)$
    - To store whether there is a path between them (or weight), using n x n matrix.
    - It will waste a lot of memory, but find the path only cost $\Theta(1)$
  - Adjacency list
    - Use an array of linked lists to store edges
    - It will save a lot of memory, but find the path needs $|E| \le 2\binom{|V|}{2} = 2\frac{|V|(|V|-1)}{2} = |V|(|V|-1) = O(|V|^2)$
    - Advantage and disadvantage?

What you have learned?

- Graph Traversal

$$|E| \le 2\binom{|V|}{2} = 2\frac{|V|(|V|-1)}{2} = |V|(|V|-1) = O(|V|^2)$$

  - Breadth-first requires a queue
    - To avoid visiting a vertex for multiple times, we have to track which vertices have already been visited. Requiring Θ(|V|) memory.
    - The time complexity of graph traversal cannot be better than and should not be worse than Θ(|V| + |E|)
  - Depth-first requires a stack based on recursion.
    - To avoid visiting a vertex for multiple times, we have to track which vertices have already been visited. Requiring Θ(|V|) memory.
    - The time complexity of graph traversal cannot be better than and should not be worse than Θ(|V| + |E|)
  - Three application
    - Connectedness, Unweighted path length, Identifying bipartite graphs

What you have learned?

- Topological Sort
  - Definitions
    - A graph is a DAG if and only if it has a topological sorting.
    - To show some certain order.
    - A DAG always has at least one vertex with in-degree zero.
    - Any sub-graph of a DAG is a DAG.
  - Algorithm
    - Use a queue (or other container) to temporarily store those vertices with in-degree zero.
    - Each time the in-degree of a vertex is decremented to zero, push it onto the queue.
    - $\Theta(|V| + |E|)$ if we use an adjacency list. (Best), determining if a graph has a cycle.
    - Extra $\Theta(|V|)$ Memory is needed (for queue).

What you have learned?

- Minimum Spanning Tree

$$|E| \le 2\binom{|V|}{2} = 2\frac{|V|(|V|-1)}{2} = |V|(|V|-1) = O(|V|^2)$$

  - Definitions
    - The weight of a spanning tree is the sum of the weights on all the edges which comprise the spanning tree.
    - All edge weights are distinct, there is a unique minimum spanning tree.
  - Prim's Algorithm
    - At each step, add the edge with least weight that connects the current minimum spanning tree to a new vertex. (Touch it).
    - Cut Property
    - Prove.
    - Using adjacency list or binary heap, the total run time is O(|V| ln(|V|) + |E| ln(|V|)) = O(|E|ln(|V|))
    - The time complexity is important. See it in Lecture 8.1, Slides 54.

What you have learned?

- Minimum Spanning Tree
  - Kruskal's Algorithm
    - Go through the edges from least weight to greatest weight (Global)
    - Add the edges to the spanning tree so long as the addition does not create a cycle
    - Until |V| – 1 edges have been added.
    - Consequently, the total run-time would be O(|E| ln(|E|) + |E|·|V|) = O(|E|·|V|).
    - Using disjoint set, we can make it to O(|E| ln(|V|)).

$$|E| \le 2\binom{|V|}{2} = 2\frac{|V|(|V|-1)}{2} = |V|(|V|-1) = O(|V|^2)$$

Exercise

Multiple Choices

- For an insertion of a single item into an n-item AVL tree, the maximum number of rotations is .
- (A) 1
- (B) 2
- (C) approximately 1.44 log n
- (D) none of the above

Multiple Choices

- Transfer from singly linked list to doubly linked list (both have a head and tail pointers), which of the following operations can definitely take less time complexity?
- (A) insert before front node
- (B) insert before tail node
- (C) erase front node
- (D) erase tail node

Multiple Choices

- Which of following algorithms is related to minimum spanning tree?
- (A) Kruskals algorithm
- (B) Prims algorithm
- (C) Dijkstras algorithm
- (D) Bellman ford algorithm

Multiple Choices

- Topological Sort can be carried out on what kinds of graphs:
- (A) All weight acyclic graphs
- (B) All directed graphs
- (C) All unweight directed acyclic graphs
- (D) All cyclic directed graphs

Multiple Choices

- Given a preorder of a binary search tree: 7 5 9 8 15 13 11 12. If we delete the key 9, which of the following could be the preorder of the tree after this operation?
- (A) 7 5 11 8 15 12 13
- (B) 7 5 8 11 12 13 15
- (C) 7 5 11 8 15 13 12
- (D) 7 5 8 15 13 12 11

Multiple Choices

- Which of the following statements is true?
- (A) All pairs of time complexity expression can be compared using Landau symbol.
- (B) An average-case exponential-time algorithm always runs slower than a worst-case polynomial-time algorithm.
- (C) If f(n) = o(g(n)); g(n) = o(h(n)), then f(n) = o(h(n)).
- (D) T(n) = $3 * 10^8$ has a much larger asymptotic time-complexity than T(n) = 1.

Multiple Choices

- Which of the following expressions is o(n log n)?
- (A) n
- (B) $n^{1.00001}$
- (C) $n^{\log_2 3}$
- (D) n^2

Multiple Choices

- Which statement(s) is(are) not true about binary tree?
- A If a binary tree has n leaf nodes, then the number of nodes of degree 2 is $n-1$.
- B In a complete non-empty binary tree with height h $(h \geq 2)$, there must exist a perfect sub-tree with height $h-1$ in this tree.
- C Suppose there are M $(M \geq 1)$ leaf nodes $l_1, l_2, \ldots, l_M$ in a binary tree. The depth of each node is $d_1, d_2, \ldots, d_M$ respectively. If $M \to \infty$, then $$\lim_{M \to \infty} \left[\max \left(\sum_{i=1}^{M} 2^{-d_i} \right) \right] = 2$$
- D None of the above.

Multiple Choices – Option C

- Which statement(s) is(are) not true about binary tree?
- C Suppose there are M $(M \geq 1)$ leaf nodes $l_1, l_2, \ldots, l_M$ in a binary tree. The depth of each node is $d_1, d_2, \ldots, d_M$ respectively. If $M \to \infty$, then $$\lim_{M \to \infty} \left[\max \left(\sum_{i=1}^{M} 2^{-d_i} \right) \right] = 2$$

Take any arbitrary tree, and add a child leaf to any non-leaf node with only one child. This can only increase $\sum 2^{-d_i}$, because you're adding leaves without changing the degree of any existing leaf, or changing any existing leaf to a non-leaf.

Now find each node with two child leaf nodes. From each of these, remove both leaves. This gets rid of two leaf nodes with degree d and creates a new leaf node with degree $d - 1$, leaving $\sum 2^{-d_i}$ unchanged.

Repeat until all that's left is the trivial one-node tree, for which $\sum 2^{-d_i} = 1$.

Prove it by structural induction on the tree (or if you like, by induction on the depth of the tree). Let $S(T)$ denote $\sum 2^{-d_i}$ for leaf nodes of the tree T. If T is a single leaf node, then $S(T) = 2^{-0} = 1$. Otherwise, both the left and right subtrees, T_l and T_r, satisfy $S(T_l) \leq 1$ and $S(T_r) \leq 1$ by inductive hypothesis. However, the depth of each leaf node in the tree is one more than the depth of the leaf node in the corresponding child tree. Therefore,

$$S(T) = \sum 2^{-d_i} = \sum_{i \text{ for left subtree}} 2^{-d_i} + \sum_{i \text{ for right subtree}} 2^{-d_i} = \frac{1}{2}(S(T_l) + S(T_r)) \leq \frac{1}{2}(1+1) = 1.$$

(And in fact, it's easy to see by a very similar proof that for any *full* binary tree T, $S(T) = 1$.)

True or False

- All sorting algorithms whose time complexity is less than □(n) cannot be a comparison based sorting.

True or False

- In a binary max heap containing n numbers, the smallest element can be found in time is O(log(n)).

True or False

- An undirected graph with n vertices and n-1 edges is a tree.

True or False

- Breadth first search and depth first search on a tree always give different traversal sequences.

General Question

- Check your homework, quiz, …

How to review?

How to review?

- What you have learned
  - Lecture: Definition, Algorithm, Time Complexity …
  - Discussion Slides
- What you have done
  - Homework
  - Quiz
- What you need to do
  - Simulate the algorithm by hand
  - Compare the similar algorithm/problem
  - DO it by hand.

Common Mistakes in Quiz

- Quiz1
  - Problem 1: Operation on Linked List
  - Problem 2: Time Complexity of Linked List
  - Problem 3: Time Complexity of Array

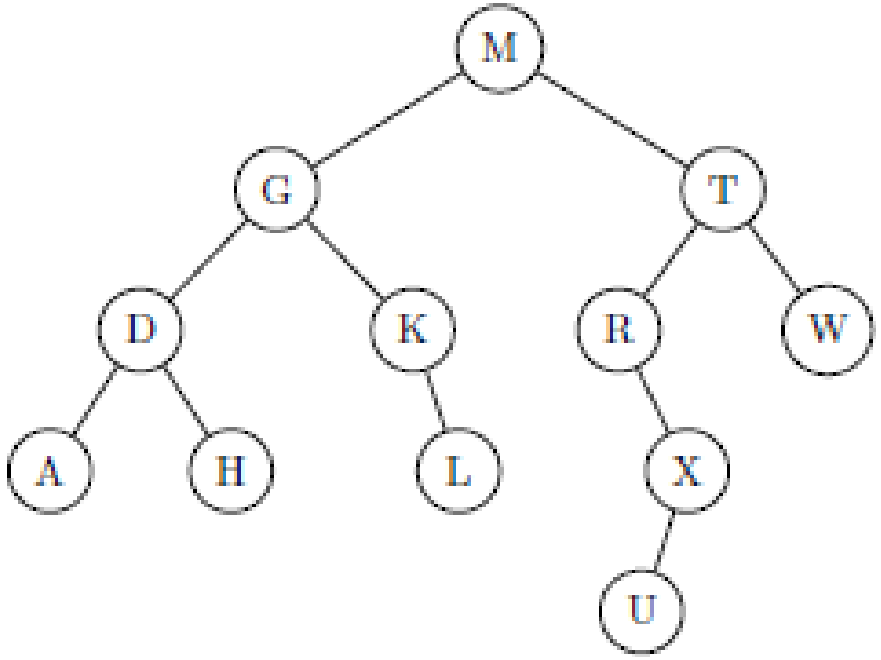
- Quiz2
  - Problem 1: Operation and Result on Stack (sequential)
  - Problem 2: Operation and Result on Array
  - Problem 3: Solve Recursive Equations (Difficult!)

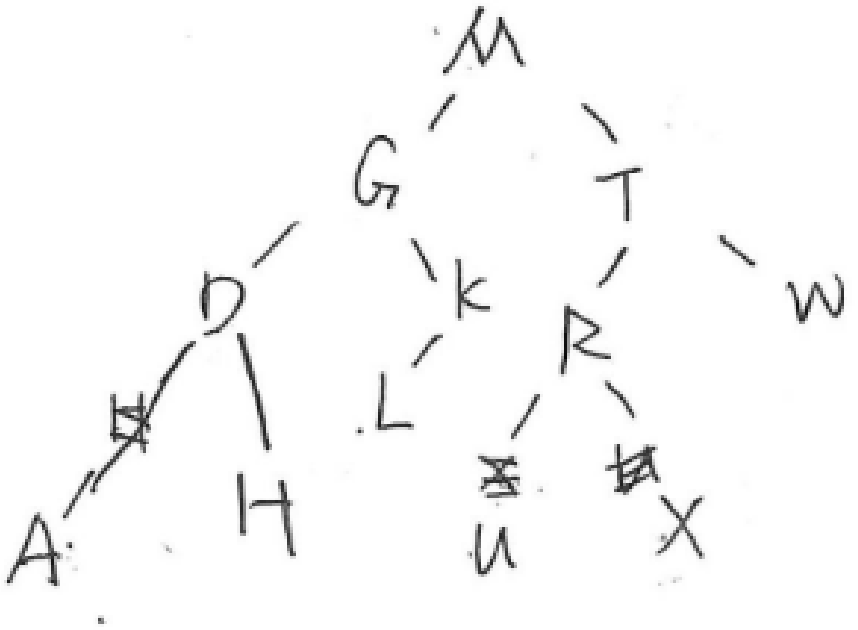
Common Mistakes in Quiz

- Quiz3
  - Problem 1: Bubble Sort Algorithm
  - Problem 2: Optimization on Bubble Sort
  - Problem 3: Intermediate steps of Insert Sort
  - Problem 4: The Pre-order and Post-order of a tree.
- Quiz4
  - Problem 1-1: Definition of full, complete, perfect binary tree.
  - Problem 1-2: Time Complexity of Build Heap
  - Problem 1-3: Tree node computation

Common Mistakes in Quiz

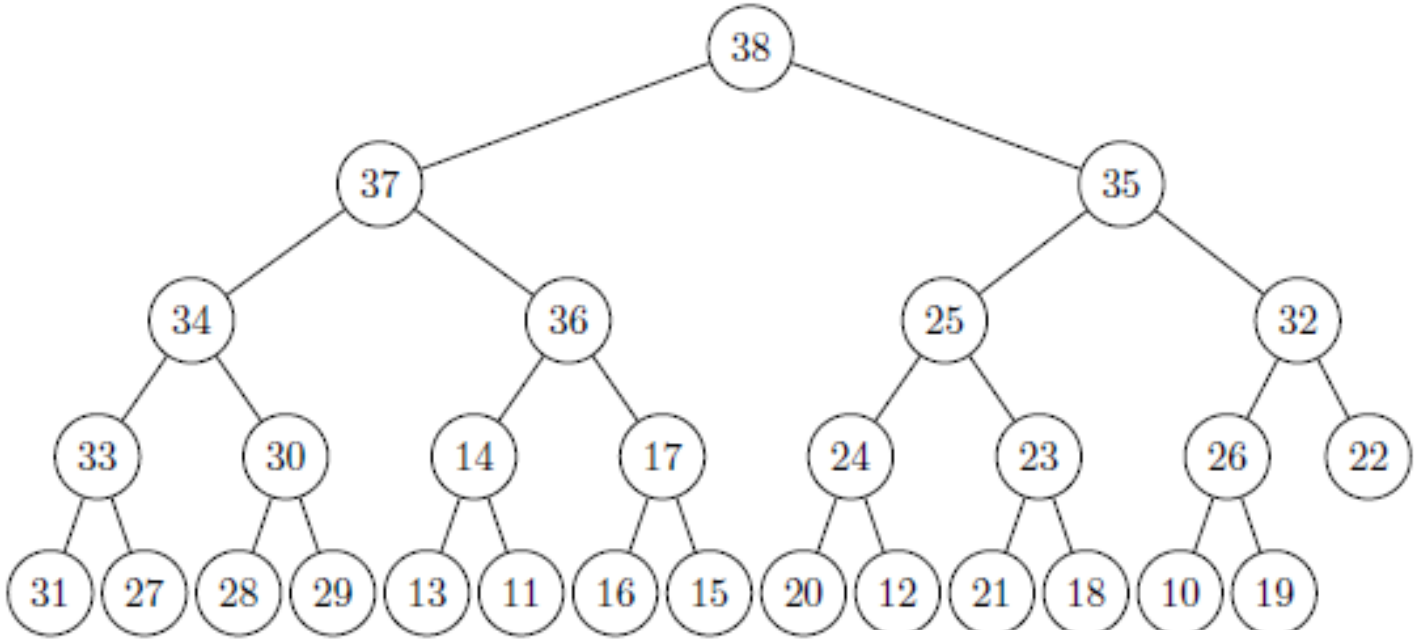
- Quiz4
  - Problem 2: Given in-order and post-order, traverse to pre-order and reconstruct the tree.





Common Mistakes in Quiz

- Quiz4
  - Problem 3: Insert and Pop Operation on Heap



(3)(2pts) Suppose that the figure above was the last operation performed in the binary heap when inserting the key x. Write down all possible value of x.

19,26,32,35. To insert a node in a binary heap, we place it in the next available lead node and swim it up. Thus, 19,26,32,35, and 38 are the only keys that we might move. But, the last inserted key could not have been 38, because, then, 35 would have been the old root (which would violate heap order because the left child of the root is 37)

Common Mistakes in Quiz

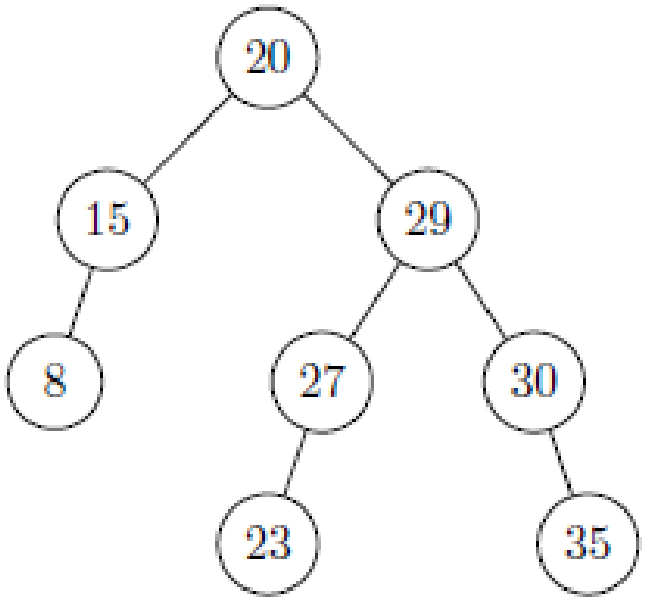
- Quiz5
  - Problem 1-1: Search Step in BST.
  - Problem 1-2: AVL Tree Property
  - Problem 1-3: AVL height/nodes computation

Common Mistakes in Quiz

- Quiz5
  - Problem 2: Insertion of BST and preorder of a tree.

Problem 2(3pts) The preorder traversal sequence of a BST is $20, 15, 8, 29, 27, 23, 30, 35$.

(a)(2pts) Construct the BST.



(b)(1pts) If we orderly insert 18,17,32 to the above BST. The preorder of the tree is 20,15,8,18,17,29,27,23,30,35,32

Common Mistakes in Quiz

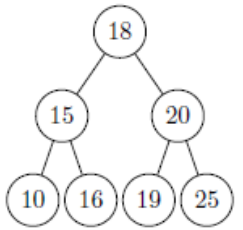
- Quiz5
  - Problem 3: Construct an AVL Tree.

Problem 3(6pts) You should use the methods taught in class to implement insertion and deletion.
1)LL/RR-rotation is prior to LR/RL-rotation in this problem.
2)If there exits an imbalance after you insert an integer, you should maintain the balance of AVL tree before you insert the next node.
(a)(2pts) Given an empty AVL tree, insert the sequence of integers 10, 20, 15, 25, 30, 16, 18 into the AVL tree. Write the pre-order traversal of the final AVL tree.

Pre-order: 20, 15, 10, 16, 18, 25, 30.

(b)(2pts) Continue with the final AVL tree in (a), insert integer 19 into the AVL tree. Write the pre-order traversal of the final AVL tree.
Pre-order: 20, 15, 10, 18, 16, 19, 25, 30.

(c)(2pts) Continue with the final AVL tree in (b), delete integer 30 from the AVL tree. Construct the AVL tree.



Good Luck

伟大学长：

我不想明年
再见到你