

COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Individual worksheet
Week 5 – 10.07.14

Instructions: Answer at least 5-10 questions in class using **anonymous** comments. When you find the location of the video that is relevant to answering the question, make a comment, with the title **Question 1** (for example). Write the possible answer to the question (anonymous) and link any other videos that may be relevant to answering the question. You are not expected to answer all questions. Look for answers written by other “anonymous” students and vote for good answers so we can see which answers get the most votes. We will try to compile some of the best answers for Thursdays lecture.

1. State 3 applications where a symbol table data structure can be used.

There are many, implementing a dictionary, hash tables, database applications, or any application where keys and values need to be connected. That is, applications, where searching by a key is more efficient and values can be large data structures such a personal record

2. Suppose that a symbol table API (page 363) is implemented with a sorted array. What is the order of growth of each operation in symbol table API?

put - n , get - $\lg N$, delete- N , contains- $\lg N$, isempty() - constant, size() - constant
keys() - N

3. Suppose that a symbol table API (page 363) is implemented with an unsorted linked list. What is the complexity of each operation in symbol table API?

put - 1, get - N , delete- N , contains- N , isempty() - constant, size() - constant, keys()
- N

4. Give a trace of the process of inserting the keys E A S Y Q U E S T I O N into an initially empty table using SequentialSearchST. How many compares are involved? (compares are given in parenthesis)

E (0)
A E (1)
S A E (2)
Y S A E (3)
Q Y S A E (4)
U Q Y S A E (5)
U Q Y S A E (5)
U Q Y S A E (6)
T U Q Y S A E (6)
I T U Q Y S A E (7)
O I T U Q Y S A E (8)
N O I T U Q Y S A E (9)

5. What is a lazy implementation of delete and explain a situation where lazy implementation may not be the best way to do delete in a symbol table?

A lazy implementation of delete leaves keys in place with null. However, if you have many deletes in a symbol table, this can be a problem. One way to avoid this problem is to keep track of the "delete factor". This is an index where it can inform you when there are too many nulls. At that point you can reinitialize the table (linear cost) to remove all null elements.

6. In a symbol table which one is unique? keys or values? why?

keys have to be unique.

7. If a symbol table is implemented with a linked list, what is the complexity of each of the operations listed in the API (as in question 2)?

see answer to question #3

8. What is the definition of rank(key)?

number of keys in the ST that are less than key

9. List all operations in an ordered symbol table API.

see page 366 in the textbook

10. How much memory is required to hold a node of a BST that contains an Item and two pointers?

see page 375. You would need 3 references (24 bytes)

11. Insert the keys {B, D, A, Z, R, C} in alphabetical order into a BST. Explain all the operations performed (compare, insert, initialize) in the process of building the BST

B		B		B		B		B
		D		A	D	A	D	A
						Z		
							Z	
							R	
								C
								Z
								R

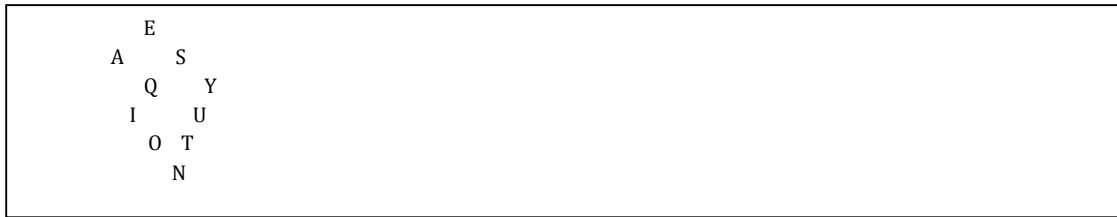
12. List 4 ordered ways that a BST can be traversed. Show the output when the BST in question #11 is traversed in two of the orders that was described.

inorder – Left, Root, Right	A B C D R Z
preorder – Root, Left, Right	B A D C Z R
postorder – left, right, Root	
level order – traverse by levels	

13. If keys {1,2,3,...n} are inserted into a BST into the order they come in (non-randomized) what is the worst case search complexity?

The BST will grow linearly. The search would cost: n

14. Draw a BST when keys E A S Y Q U E S T I O N is inserted in that order associating value i with i^{th} key.



15. In general BST's cannot handle duplicate keys. Suggest a way to design the data structure so that duplicate keys can be handled. What is the extra cost? memory and/or runtime?

One possibility is to store the key count in the node. This would cost extra integer for each node and a total of $4N$ memory

16. If n keys are inserted into a BST in random order what is the expected number of compares to search or insert in tilde notation?

$\sim 1.39 \lg N$

17. Design an algorithm to find the total number of nodes in BST. What is order of growth of your algorithm?

An idea would be to define a recursive formula: Let $\text{count}(T)$ be the total nodes in a tree T (T also represents the root). Then $\text{count}(T) = 1 + \text{count}(\text{left child of } T) + \text{count}(\text{right child of } T)$. This can be done in linear time.

18. Design an algorithm to find the height of a BST. What is order of growth of your algorithm?

An idea would be to define a recursive formula: Let $\text{height}(T)$ be the total nodes in a tree T (T also represents the root). Then $\text{height}(T) = 1 + \text{Math.max}(\text{height}(\text{left child of } T), \text{height}(\text{right child of } T))$ - linear time

19. Design an algorithm to find the floor and ceiling of a given key. What is the order of growth of your algorithm?

In an ordered list floor entry ceiling
Find the entry in $\lg N$ time (balanced BST). To find the floor we may need N work. To find the ceiling a

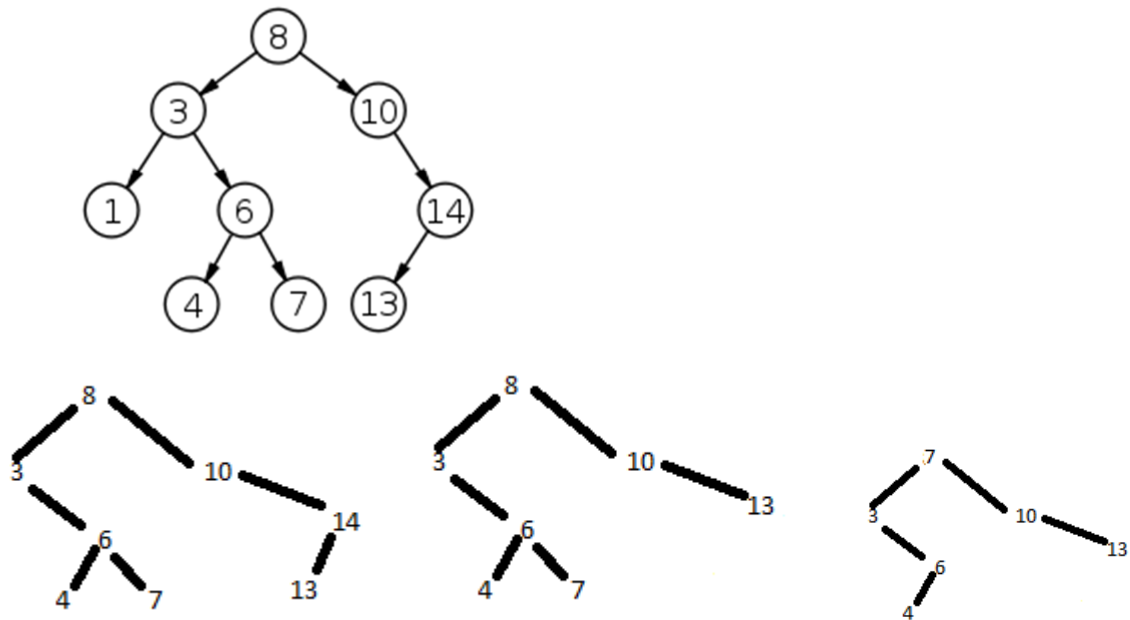
20. Given 4 comparable keys $\{1, 2, 3, 4\}$ how many BST's are possible? Draw all the trees.

you should draw all possible BST's. You can actually show (not in this class) the number is $1/(n+1) \cdot (2n \text{ C } n)$ where n is the number of keys

21. Why is it that the lazy approach of deleting keys from a BST is not desirable? State a situation where the lazy approach can be a problem.

same as the argument we made with ST's. Just leaving a lots of dead nodes can waste space. You can rebuild the tree when the dead node percentage goes up.

22. Consider the following BST (source: Wikipedia). Show the tree after deleting the keys 1, 14 and 8 successively.



23. Show that it is not possible to build a BST in linear time given n keys

Suppose it is possible to build a BST in linear time. Then we have a sorting algorithm that can sort a set of comparable keys in linear time. But we showed that we need at least $N \lg N$ comparisons to sort a list of size n . So this is a contradiction

24. What is the worst case complexity of the following operations. State your answer in order of growth notation.
- traverse a BST inorder
 - traverse a BST in level order
 - Find the max of a BST
 - Find the floor of a key
 - Find all keys in a given range $[k_1, k_2]$ including k_1 and k_2

- N
- N
- $\lg N$ (if the BST is balanced)
- N
- N

25. Given n horizontal and n vertical line segments, design a naive algorithm for each of the following operations and determine worst case order of growth for your algorithm
- Find all intersection points
 - Find all intersection points between two horizontal or two vertical line segments

A naive algorithm can scan all pairs (there are n^2 of them) of and find if they intersect. You can find if two lines intersect in linear time (how). So the order of growth is n^2

26. Explain an efficient algorithm for finding intersection of n horizontal and n vertical line segments. What is the order of growth of your algorithm for finding all intersection points?

see salon video

http://www.classroomsalon.com/video/view_video.aspx?mode=view&document_id=3418

for sweep algorithm

27. Given n points (x,y) form, design an algorithm to find all points that are inside a given rectangle? What is the worst case order of growth for your algorithm?

A naïve algorithm for doing this is to scan all points (N of them) and call the method `contains(point, rectangle)` that checks if the point is inside the rectangle. The order of growth is N

28. Insert the points $(2,3)$, $(2,4)$, $(1,1)$, $(5,2)$, $(3,3)$ into a 2d tree. Show the tree and corresponding bounding box for each point.

(2,3)
(1,1) (2,4)
 (5,2)
 (3,3)

$(2,3) \rightarrow (-inf, -inf) (inf, inf)$
 $(2,4) \rightarrow (2, -inf) (inf, inf)$
 $(1,1) \rightarrow (-inf, -inf) (2, inf)$
 $(5,2) \rightarrow (1, -inf) (2, inf)$
 $(3,3) \rightarrow (1, -inf), (2,2)$

29. Describe the operations, nearest neighbor search and range search in 2d trees. What is the worst case order of growth in each case?

Nearest neighbor search requires finding the closet bounding box for a given point. The range search requires pruning the tree until we find the points that are contained in the bounding box.