# Data Structures: Revision pack

June 5, 2018

## 1 Linked Lists

i. What is the difference between singly and doubly linked lists?

ii. Write a function to determine whether a given linked list contains loops

iii. Describe how you would insert a node at the beginning of a linkedlist

iv. For your description in (iii) write the recursive code to do so.

v. Describe how you would insert a node at the end of a linkedlist

vi. For your description in (v) write the recursive code to do so.

vii. Show how you can achieve the (iv) and (vi) above non-recursively.

viii. What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?

ix. Describe how you would reverse a singly linked list

x. Write the code to implement (ix) above.

## 2 True/False

Answer the following questions with either true or false. Assume there are n elements in the data structure. No explanation necessary

i. One can implement a stack based on a linked list so that EACH INDIVIDUAL push/pop operation is time O(1).

i. One can implement a stack (of unbounded size) based on an array so that each individual push/pop operation is time O(1).

ii. One can reverse the order of the elements in a linked list in time O(n).

iii. It is possible to append two linked lists in time O(1).

iv. Adding an element to a heap has worst-case time complexity $O(log(n))$.

v. Returning the maximum element in a max-heap (but not deleting it from the heap) can be done in time O(1).

# 3 Tree traversals

Give the preorder, inorder, and postorder traversal of the following tree.

Preorder: 9,5,3,1,4,8,6,20,12,10,11,30,21,31

Inorder: 1,3,4,5,6,8,9,10,11,12,20,21,30,31

Postorder: 1,4,3,6,8,5,11,10,12,21,31,30,20,9

# 4 Trees

## 4.1

Write a recursive function void mirrorTree(node root) that changes a given input tree so that it becomes the mirror image of the original tree. For example:

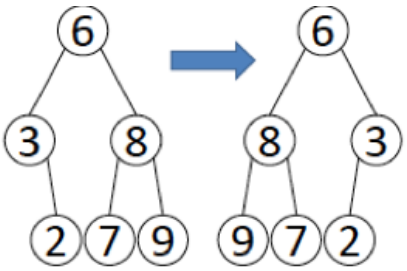


Figure 1: Mirror trees.

For this question, assume you have a node object that has the basic methods implemented: getLeft(), getRight(), setLeft(), setRight(), getValue(). All the values in a node are integers.

## 4.2

For Question 4.1 above, write a function that checks whether your mirror tree is the same as the original.

# 5 Binary Search Tree

Draw the binary search tree obtained when the keys 1, 2, 3, 4, 5, 6, 7 are inserted in the given order into an initially empty tree. What is the problem of the tree you get? Why is it a problem? How could you modify the insertion algorithm to solve this problem. Justify your answer.

# 6 AVL Tree

## 6.1

i. Insert the following sequence of elements into an AVL tree, starting with an empty tree: 10, 20, 15, 25, 30, 16, 18, 19.

ii. Delete 30 in the AVL tree that you got.

## 6.2

Draw an AVL tree that satisfies the following three conditions:

i. The tree has exactly 11 nodes.

ii. There are no pair of nodes that, if extracted one after the other, will cause the height decrease by 1.

iii. There is no key whose insertion will increase the height by 1.

# 7 Max -Heap

Consider the array A = {29, 18, 10, 15, 20, 9, 5, 13, 2, 4, 15}

i. Does A satisfy the max-heap property? If not, fix it by swapping elements

ii. Using array A (possibly corrected), illustrate the execution of the heap-extract-max algorithm, which extracts the max element and then rearranges the array to satisfy the max-heap property. For each iteration or recursion of the algorithm, write the content of the array A.

# 8 Min-Heap

## 8.1

Consider the array: A = {4, 33, 6, 90, 33, 32, 31, 91, 90, 89, 50, 33 }

i. Is A a min-heap? Justify your answer by briefly explaining the min-heap property.

ii. If A is a min-heap, then extract the minimum value and then rearrange the array with the min-heapify procedure. In doing that, show the array at every iteration of min-heapify. If A is not a min-heap, then rearrange it to satisfy the min-heap property

### 8.2

Draw the binary min heap that results from inserting: 77, 22, 9, 68, 16, 34, 13, 8 in that order into an initially empty binary min heap. You do not need to show the array representation of the heap. You are only required to show the final heap, although if you draw intermediate heaps, please circle your final result for ANY credit.

Once your heap is complete, answer the following question.

i. Draw the binary min heap that results from doing 2 deletemins on the heap you created in part a). You are only required to show the final heap, although if you draw intermediate heaps please circle your final result for ANY credit.

## 9 Search & Sorting techniques

i. Define and contrast the following search and sorting techniques: Insertion Sort, Heap Sort, Merge Sort, Quick Sort, Sequential Search, Binary Search

ii. For each of the mentioned search and sorting techniques, specify the worst-case and average-case Big-Oh complexity, assuming an input array of size N (You can draw a table with a column for worst-case and average-case):

## 10 Sorting

### 10.1

i. Describe the basic principle of the mergesort algorithm. Illustrate your answer by showing the steps involved in sorting the array

   A = $\{9, 3, 6, 2, 4, 1, 5\}$.

ii. Insertion sort can be considered as a mergesort where each step divides an array of size $n$ into two arrays; one of size 1 (the element to be inserted) and one of size *(n - 1)* for array length $n$. By solving an appropriate recurrence relation, show that this recursive version of insertion sort has a time complexity of O ($n^2$) Assume the time complexity for merging two arrays is O(n).

iii. A programmer is tasked with sorting both arrays and linked lists. For both data structures, they intend to use the mergesort algorithm.

   i. Show that the time complexity of a linked list mergesort is O(nlogn). Show also that the space complexity is O(1), taking care to demonstrate how this can be achieved.

ii. The programmer only knows how to merge two arrays in O(n) space and linked lists in O(1) space. They thus propose converting the arrays to linked lists before applying the mergesort algorithm to save on space. Comment on this strategy.

### 10.2

Give the result of Sorting the array below based on the following techniques:

N E W P A R T I TI O N Q U E S T I O N

i. Merge Sort
ii. Heap Sort
iii. Insertion Sort
iv. Selection sort
v. Quicksort (assuming the leftmost element is the pivot)

## 11 Data Structures

We have static data (the data is given initially and does not change) where the keys are natural numbers. We would like to answer queries of the following form: Given a number j, find the key k that is closest to j (i.e. $\|k-j\|$ is smallest) where k is a key in the data structure.

i. Assuming our entire data structure fits into memory, what data structure would you use? Explain why your solution is correct and compute its time complexity.

ii. Assuming the size of our data structure is 100GB, that the memory space is 2GB, that every read from the disc is a block of 1GB, and that the time for reading a block of data from the disc is equal to 2000 operations in memory. what data structure would you use? Describe an algorithm that answers the query. Explain why it is correct and compute its time complexity.