Week: 07 Date: 3/18/2021

|  |
| --- |
| 15-110 Recitation Week 7 |

**Reminders for Students**

* Check 4 due Monday 3/22 @ Noon EDT
* Check 3 and HW 3 due Tuesday 3/23 @ Noon EDT
* Small Groups for Quiz 3 which is on 3/24
* See Piazza for mid-semester surveys (& HW4 bonus points) due by Mon 3/29 noon!
* Usual recitation feedback form: <https://forms.gle/WKrrbawKktmRu1xp9>

**Overview**

* Sorting
* Hashing
* Trees, Binary Trees
* Graphs

|  |
| --- |
| Problems |

# **HASHING REVIEW**

(a) How are the locations for data chosen in a hashtable?

Answer:

(b) How can we check if an element is in a hash table?

Answer:

(c) What is the difference between a good and bad hash function?

Answer:

(d) Describe one concrete situation in which a hash table would **not** be an appropriate choice of data structure. Explain why a hash table would not be appropriate in that situation.

Answer:

# 

# **MERGESORT REVIEW**

**Visually trace mergesort on the list [7, 5, 2, 3, 9, 1].**

Trace:

Does the runtime improve if the list is already sorted in ascending order? Descending order?

Answer:

**Runtime Analysis:**

How many steps per split-pass + merge-pass?

Answer:

How many split passes + merge passes?

Answer:

Total # steps?

Answer:

Total complexity?

Answer:

# **BINARY TREE CODE WRITING:**

**Binary Tree Review:**

* A binary tree is represented as a dictionary with 3 key-value pairs
  1. “contents” - corresponds to the value of the current node
  2. “left” - the left subtree extending from the current node
  3. “right” - the right subtree extending from the current node
* If a given node has no left subtree, the “left” key will have a value of None - same goes for the right subtree of a given node
* A leaf node is a node in the tree which does not have a left subtree (has a value of None for its “left” key) and does not have a right subtree (has a value of None for its “right” key)

Write the function **addLeaves(tree)** which takes in a tree represented as a dictionary and adds the value of all leaf nodes together, where each leaf node is an integer, and returns the sum. Make sure not to include the values of the internal nodes in the sum.

def addLeaves(tree):

# 

# 

# 

# **GRAPHS PRACTICE**

In 15-110, we represent graphs as dictionaries. Try drawing the graph that is represented by the dictionary below.

graph = {

“A”: [[“B”, 4], [“C”, 2], [“E”, 6]],

“B”: [[“A”, 4], [“D”, 1], [“E”, 7]],

“C”: [[“A”, 2]],

“D”: [[“B”, 1]],

“E”: [[“B”, 7], [“A”, 6]]

}

Graph:

**Graph Code Writing:**

How can we print every node in this graph G? Write some code to iterate through every node in the graph and print the node’s value.

Code:

Now, how can we print the value of each neighbor of a node x?

Code: