15-110 Recitation Week 8

Reminders

- Hw4 due Monday, October 27 at noon
    - Fill out the midsemester surveys by this date for bonus points!
        - Midsemester Survey - Course
        - Midsemester Survey - TAs
- Recitation feedback form

Overview

- Hashing Code Tracing
- Binary Search Tree Tracing
- Graphs Code Writing
- Tractability, P vs. NP

Problems

Hashing Code Tracing

Prof. Rivers wants to keep track of what fruits her friends like, and she decides a dictionary would be the best way to do this. She has collected data from her friends and has run these lines of code to initialize the dictionary using the hash function below. She stores the keys and values in a hashtable of length 5, as shown below.

Note: Remember we do hash(value) % size if the hash value is larger than the size of the table.

```
def hash(s):
    return len(s)

d = {}
d["pineapple"] = 10
d["pears"] = 7
d["bananas"] = 14
```

“pears” : 7		“bananas” : 14		“pineapple” : 10
Bucket 0	Bucket 1	Bucket 2	Bucket 3	Bucket 4

1) Prof. Rivers realizes she forgot that she also has 50 friends who love figs.
    a) What line of code should she use to add this information to the dictionary?

    b) Where will it be added in the hash table?

2) What makes this a bad hash function, and how can we make this hash function better?

3) Prof. Rivers comes back to this dictionary later and wants to check if any of her friends like apples.
    a) What line of code should she run?

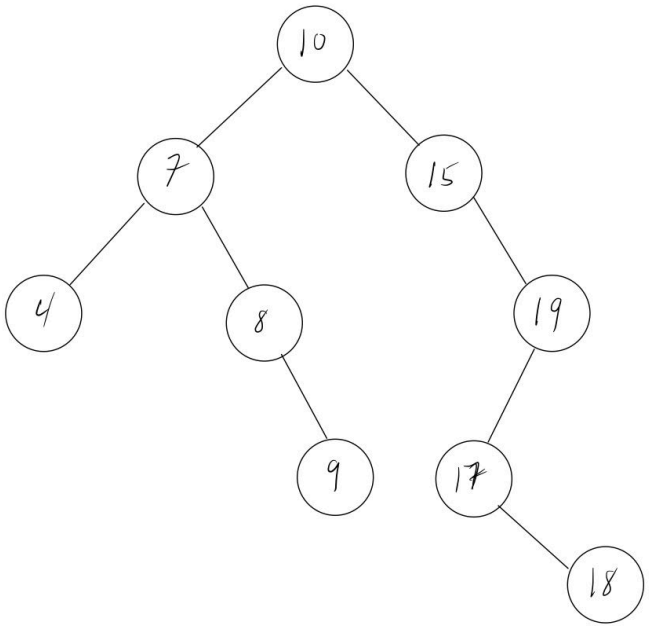
    b) Conceptually, how do we check if an element is in the hashtable?
    hash(“apples”) = 6 % 5 = 1

4) What would happen if Prof. Rivers tried to run these lines of code?
    a) `print(d["bananas"])`

    b) `print(d[“pomegranate”])`

5) Describe a situation in which it would not be appropriate to use a hash table, and explain why.

Binary Search Tree Tracing

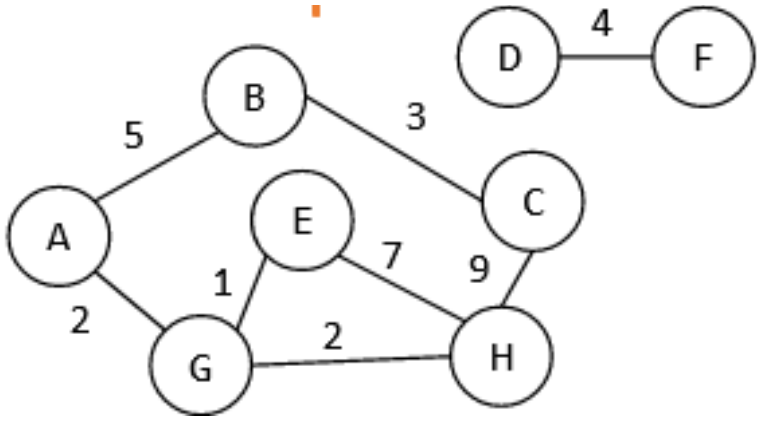


What constraints does a binary search tree follow ?

What elements would you look through to find the value 17?

What elements would you look through to find the value 11?

Graphs Code Writing




```
g = {
      "A" : [ ["B", 5], ["G", 2] ],
      "B" : [ ["A", 5], ["C", 3] ],
      "C" : [ ["B", 3], ["H", 9] ],
      "D" : [ ["F", 4] ],
      "E" : [ ["G", 1], ["H", 7] ],
      "F" : [ ["D", 4] ],
      "G" : [ ["A", 2], ["E", 1], ["H", 2] ],
      "H" : [ ["C", 9], ["E", 7], ["G", 2] ]
```

Given an undirected, weighted graph g, write the function **sumWeights** that returns the sum of all the weights in the graph. For example, sumWeights(g) = 33.

Tractability and P vs. NP

Notes:

	P	NP
Verifying		
Solving		

True/False Questions:

We can solve problems in P in polynomial time [T / F]

We can solve problems in NP in polynomial time [T / F]

Some intractable problems are in P [T / F]

All problems in P are in NP [T / F]

Linear Search is in NP [T / F]

Puzzle solving is in NP [T / F]

Exam Scheduling is in P [T / F]

If you can find a tractable solution to any useful NP problem, you prove P=NP [T / F]