15-110 Recitation Week 8

Reminders

- Hw4 due Monday 10/30 at Noon
- Fill out mid-semester surveys for HW4 **extra points**
- Recitation Feedback Form

Overview

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

Problems

Hashing Code Trace

Muzaffar wants to keep track of what fruits his friends like, and he decides a dictionary would be the best way to do this. He has collected data from his friends and has run these lines of code to initialize the dictionary using the hash function below. The corresponding hash table of length 5 is also shown below:

Note: Remember we do hash(value) % size of table 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) Muzaffar realizes he forgot to account for the 50 of his bros that love figs.
    a) What line of code should he use to add this 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) Muzaffar comes back to this dictionary later and wants to check if any of his friends like apples.
    a) What line of code should Muzaffar run?

b) How do we check if an element is in the hashtable?

-

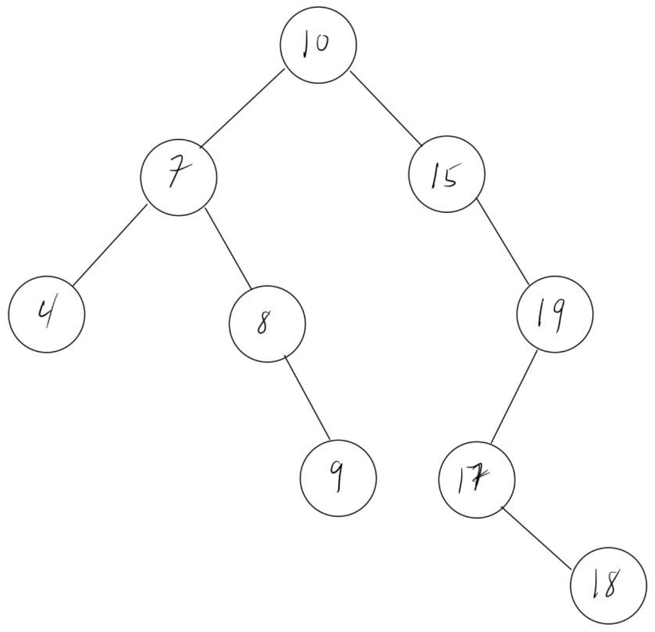
4) What would happen if Muzaffar tried to run these lines of code?

a) `print(d["bananas"])`

b) `print(d[“pomegranate”])`

5) 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.

Binary Search Tree Tracing

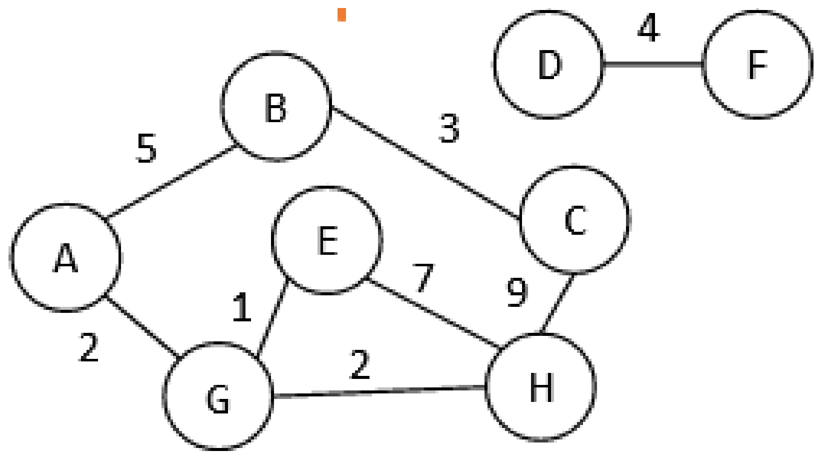


What constraints does a binary search tree follow ?

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

What elements would you look through to find the value of 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] ]
```

Take 7 minutes to try this on your own. Given a graph g, write the function **sumWeights** that returns the sum of adding each weight in the graph. Ex: sumWeights(g) = 33

Tractability & P VS. NP

Notes:

	P	NP
Verifying		
Solving		

Quick True/False Questions:

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