# CSE 373: Practice Midterm

## 1 Short Answer

a) Let $f(n) = 5n^3 + 6n^2 - 4n$ and let $g(n) = 6n^3 - 9n^2 + 16$.
Show that $f(n) = O(g(n))$ or $g(n) = O(f(n))$ or both, if applicable, by providing the appropriate $c$ and $n_0$

$5n^3 + 6n^2 - 4n \leq c * (6n^3 - 9n^2 + 16)$
Let c be 10
$5n^3 + 6n^2 - 4n \leq 60n^3 - 90n^2 + 160$
$96n^2 \leq 55n^3 + 4n + 160$ for $n \geq 160$
Since we have found a $c$ and $n_0$, we can say that $f(n) = O(g(n))$

$6n^3 - 9n^2 + 16 \leq c * (5n^3 + 6n^2 - 4n)$
Let c be 10 again,
$6n^3 - 9n^2 + 16 \leq 50n^3 + 60n^2 - 40n$
$16 \leq 44n^3 + 69n^2 - 40n$ for $n \geq 1$
Since we have found a $c$ and $n_0$, we can say that $g(n) = O(f(n))$
Therefore, it is the case that $g(n) = O(f(n))$ and $f(n) = O(g(n))$

b) Suppose a heap is full and has $n$ elements. What is the bigO asymptotic runtime of an insert? Explain your answer

In this situation, the run would take $O(n)$ time because the array would have to be resized and copied over.

c) Write pseudocode, or explain briefly in paragraph form, an algorithm that will find the largest element in a BST. Include the worst-case runtime of this algorithm.

Define a pointer which keeps track of the current node and start it at the root. While the current pointer's right child is not null, move the pointer to point at its right child. Once there, return the pointer. This runs in worst-case $O(n)$ time. This occurs when the data is stored in an unbalanced BST where no node has left children.

d) What is the best-case and worst-case bigO complexities for `find(key k, value v)` in a half-full hash table implemented using linear probing. Briefly explain these best and worst case scenarios.

The best case is $O(1)$ and it occurs when the hashtable has elements in every other space in the table. The worst case is $O(n)$ and occurs when the hashtable is one cluster that spans half of the table. If a find or insert occurs in this cluster, it must search through the end to complete.

## 2 Big O notation

For the following functions, determine the tightest bigO upper bound in terms of $n$. Write your answers on the line provided.

a)
```
void silly(int n) {
    for (int i = 0; i < n; ++i) {
        j = n;
        while (j > 0) {
            System.out.println(j =  + j);
            j = j - 2;
        }
    }
}
```

$O(n^2)$

b)
```
void f2(int n) {
    for(int i=0; i < n; i++) {
        for(int j=0; j < 10; j++) {
            for(int k=0; k < n; k++) {
                for(int m=0; m < 10; m++) {
                    System.out.println("!");
    }   }   }   }
}
```

$O(n^2)$

c)
```
int f3(int n){
    if (n < 10) return n;
    else if(n < 1000) return f3(n-2);
    else return f3(n/2);
}
```

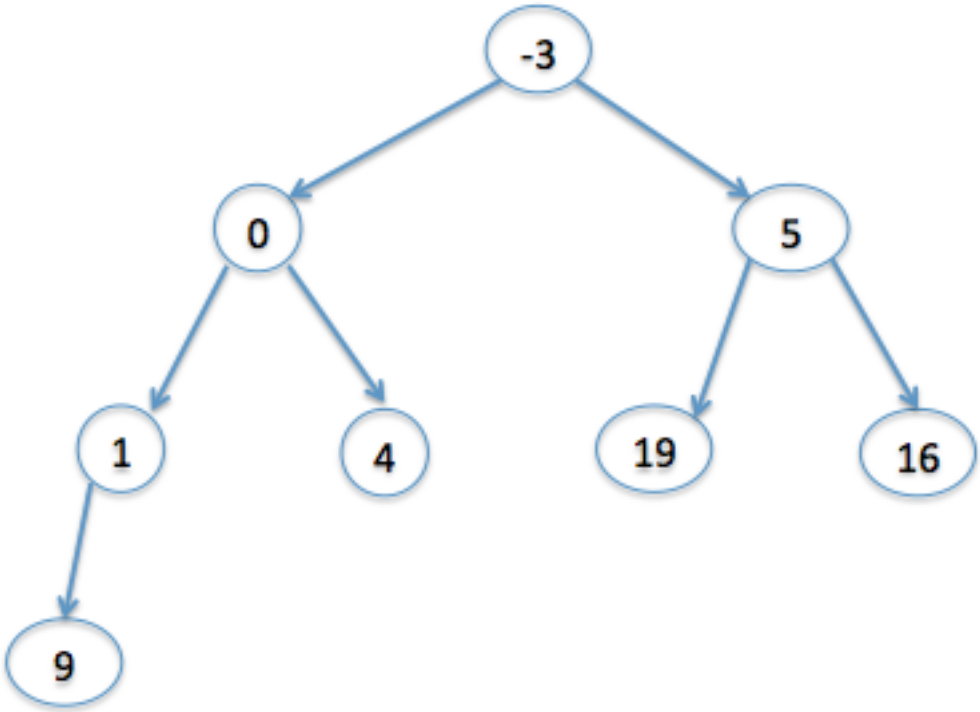
$O(logn)$

# 3 Heaps

Show a min-heap after performing the following inserts:

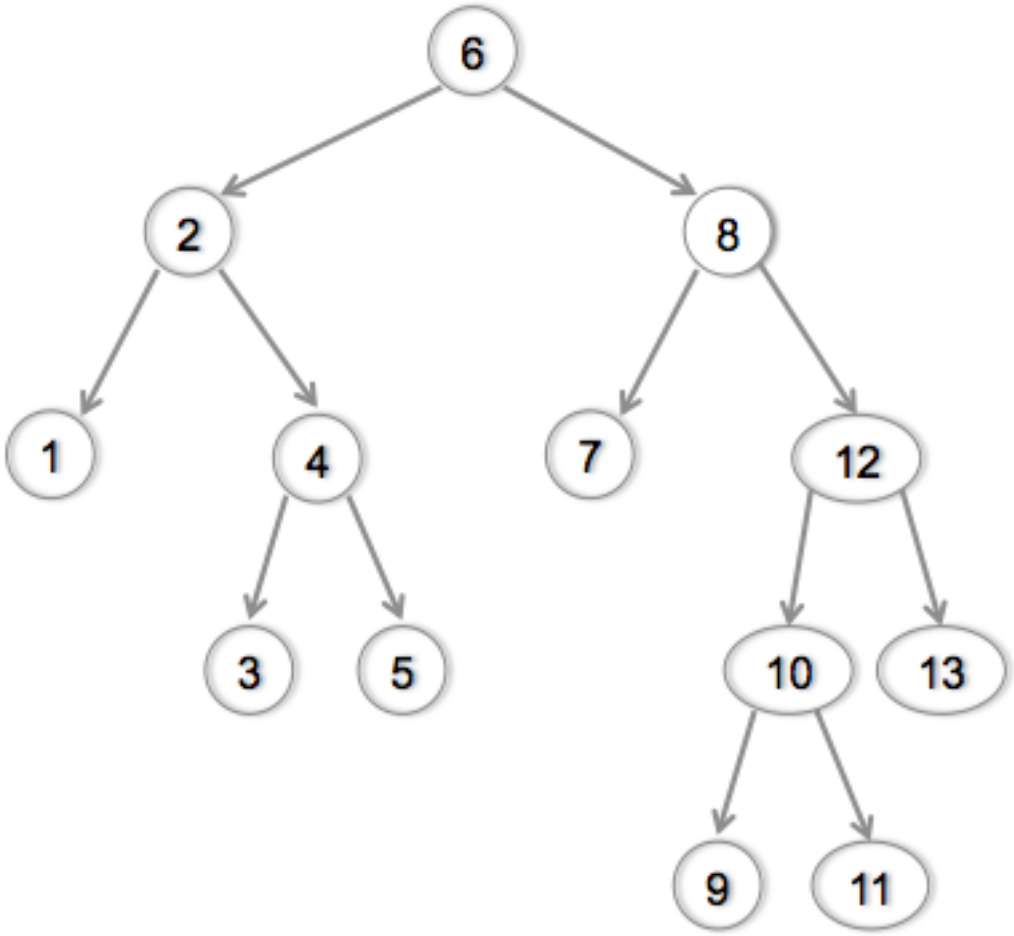
$$[1, 9, 16, 4, -3, 19, 5, 0]$$

You only need to show the final result, but showing intermediate steps may earn you partial credit if a mistake is made. You may show either the tree or the array in your result.



# 4 Traversals

Provide the pre-order, in-order, post-order and BFS traversal of the following Binary Search Tree.




Pre-order:
6,2,1,4,3,5,8,7,12,10,9,11,13

In-order:
1,2,3,4,5,6,7,8,9,10,11,12,13

Post-order:
1,3,5,4,2,7,9,11,10,13,12,8,6

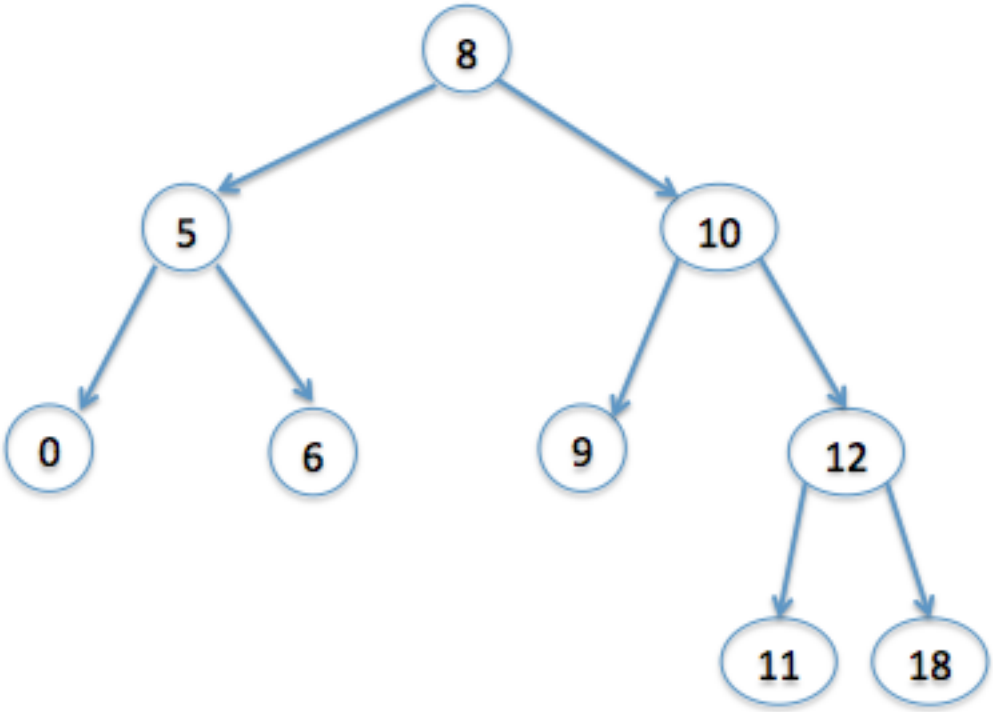
Breadth-first Search:
6,2,8,1,4,7,12,3,5,10,13,9,11

# 5 AVL insertions

Show an AVL tree after performing the following inserts:

$$[5, 8, 6, 10, 18, 9, 0, 11, 12]$$

You only need to show the final result, but showing intermediate steps may earn you partial credit if a mistake is made.



# 6 Design Decision

A client is trying to provide a data structure which logs the locations of process jobs on a small server farm. There are 64 servers at the farm and there is no limit to the number of jobs that should be assigned to each server. For each process job, there are four pieces of information which need to be tracked:

- An int ID number which is unique to each job
- An int which indicates to which server the job has been assigned
- A long which is the time that the process was assigned to the server
- A String which identifies the owner of the job

Because this is a log, the client will only delete records when they were created in error. Because of this, fast delete times are not important. Additionally, the client will be calling find much more frequently than insert, so any speed benefits should prioritize speeding up find, if possible.

Provide a data structure and implementation that would meet the clients demands. Explain what data will be stored where and how it will be accessed. Then, justify any decisions you made using material from the course. This includes, but is not limited to: asymptotic runtimes, memory usage, experimental results and data structure properties.

An easy solution here is an AVL tree. Because it provides fast O(log n) finds because of the balance operation without much maintenance it is desirable. Since delete isn't important in the specifications, AVL also seems to be a good fit.
Our AVL tree data structure implements the Dictionary ADT. Here, we use the ID number as the keys and use a combination of the server number, process time and owner variables as the key.
The benefits of the AVL tree become more apparent when the data structure becomes larger. Given that there is a significant amount of memory in our key and values, memory efficiency is likely to be high.

Because the input size is unbounded, an array may not be ideal. But a sorted array data structure may also be applicable here. Because insert times are not important, the time to sort and shift in an array may not be as important. Additionally, because memory usage is not constrained, a hashtable is an acceptable choice as well.