COMP 285 Practice Midterm Questions

The following are questions meant to help you practice, and cannot be submitted for a grade.

Important Notes

- *It is meant to give you a chance to do some practice questions after having reviewed the slides, quizzes, in-class exercises, homeworks, etc.*
- *It should give a rough sense of some ways questions might be posed, though there's no guarantee that the actual midterm will have the exact same format (at a minimum, one difference is that the actual midterm will show the point values associated with the questions).*
- *It should give a rough idea of the level of mastery expected generally, though more/less mastery may be expected for any given topic.*

Thanks for reading the notes above - the big picture thing is that I want to be sure you use this resource appropriately, while at the same time **do not neglect the many other more comprehensive resources**!

Asymptotic Analysis

1. O(n/100 + log(n) + 200) can be simplified to O(n). True or False?

2. $2x + x^2/2 = \Theta(x^2 + 2x + x \log(x))$. True or False?

3. $x + 20 = \Omega(999)$. True or False?

For questions 4 - 6, refer to the containsDuplicates pseudocode.

```
algorithm containsDuplicates
  input: size n vector of ints called vec
  output: true if vec contains duplicates, false otherwise

for i = 0...n-1
  for j = i + 1...n-1  // Notice we start at i + 1, not j
    if vec[i] == vec[j]
      return true
return false
```

4. What is the **worst-case runtime** of containsDuplicates? Define n, provide a tight upper bound with Big-O, and justify your answer.

5. What is the **worst-case space complexity** of containsDuplicates? Define n, provide a tight upper bound with Big-O, and justify your answer.

Using the Right Tools

6. Which of the data structure implementations below have O(1) runtime on average for element insertions? Select **ALL** that apply.
   - ☐ A stack (C++: std::stack)
   - ☐ A queue (C++: std::queue)
   - ☐ A hash set (C++: std::unordered_set)
   - ☐ A hash map (C++: std::unordered_map)
   - ☐ A priority queue (C++: std::priority_queue)

7. Given a vector of n integers, where each integer is at most d away from its correct position in the sorted vector, complete the pseudocode in the box below that returns a sorted array in O(n log(d)) time.

```
algorithm sort
  input: d and an almost sorted vector vec of ints as described above
  output: the sorted vector

m = new min priority queue of size d + 1
for i = 0...d
  m.push(vec[i])
ret = new empty vector to be returned
i = d + 1
while !m.empty()
  ┌──────────────────────────────────┐
  │                                  │
  └──────────────────────────────────┘
  if i < vec.size()
    m.push(vec[i])
    i++
return ret
```

Sorting

8. Which array of the following will RadixSort take the most number of steps on? Select **ONE.**
    a. [1, 2, 3, 4, 5, 6]
    b. [5, 43, 3, 11, 6, 9]
    c. [3, 1, 34, 3, 4, 81]
    d. [4, 4754, 4, 24, 1, 33]

For questions 9 - 10, refer to quickSort provided.

```
algorithm quickSort
  Input: vector<int> vec of size N
  Output: vector<int> with sorted elements

  if N < 2
    return vec
  pivot = findPivot(vec)
  left = new empty vec
  right = new empty vec
  for index i = 0, 1, 2, ... N-2
    if vec[i] <= pivot
      left.push_back(vec[i])
    else
      right.push_back(vec[i])
  return quickSort(left) + [pivot] + quickSort(right)
```

9. Suppose findPivot is a function which finds the element that will partition the list in two (nearly) equal halves in linear time while using constant space. What is the **worst-case runtime** of quickSort in this case? Justify your answer.

10. Challenge: what is the **worst-case space complexity** of quickSort in this case? Justify your answer.

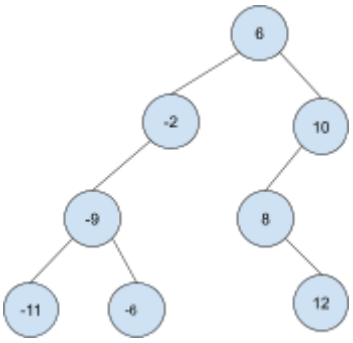
Master Theorem

11. Find the runtime of an algorithm described by the following recurrence relation: $T(n) = T(n/2) + O(1)$. You may show your work for partial credit.

12. Write a recurrence relation for MergeSort. You may show your work for partial credit.

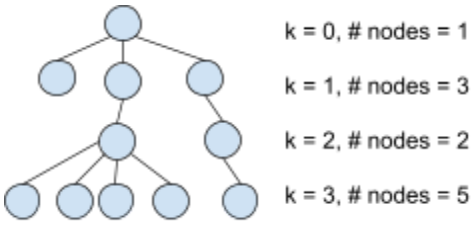
Trees

13. Is the tree on the right a Binary Search Tree? Explain.



---

14. Complete the recursive case of countAtLevel in the box, which counts the number of nodes at each level in a Tree.




```
algorithm countAtLevel
  input: TreeNode root and a level k
  output: the number of nodes at level k in root

if level == 0  // base case
  return 1

total = 0
for each element child in root->getChildren()
   total +=
return total
```

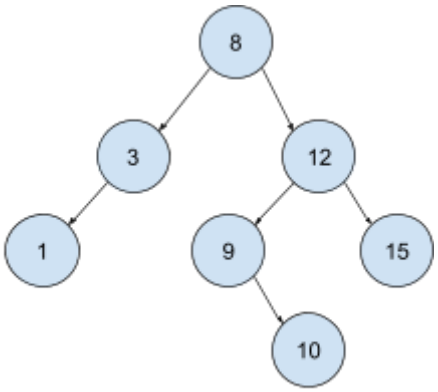
15. What is the **best-case runtime** of **removing** a node for a "**regular**" (i.e. not Red-Black) BST. Give an example of when the best-case happens.

16. What is the **worst-case runtime** of **searching** for a node in a **balanced** (e.g. AVL) BST. Give an example of when the worst-case happens.

For question 21, use the following pseudocode

```
BSTremove(t, v) // from visualgo.net
  search for v
  if v is a leaf
    delete leaf v
  else if v has 1 child
    bypass v
  else replace v with successor
```

17. Draw what the BST t below will look like after BSTremove(t, 8)



Graphs

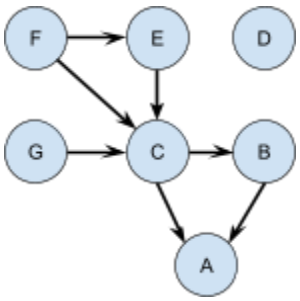
Use the following adjacency list, to answer questions 22 - 23

```
0: {1}
1: {3, 4}
2: {0}
3: {1, 2}
4: {}
```

18. Represent this Graph as an adjacency matrix. Recall that `m[a][b] = 1` means that there's a directed edge from Node(a) to Node(b)

19. Does this graph have cycles? If yes, identify them.

Use the following DAG to answer questions 20



20. Provide one valid topological sort for this DAG

For questions 21 - 23, use the below.

Wildlife scientists observe elephants in Serengeti National Park. Although the elephant herds may rarely be seen altogether, the scientists want to understand the **average herd size**, so they record all elephant "interactions" they observe over 3 months. Assume:

- The scientists can uniquely identify each elephant.
- Elephants will only ever "interact" with other elephants in their same herd.
- Elephants can only belong to one herd.

21. In order to solve this problem, we can represent this as a graph. What are the nodes and edges?

22. Which graph properties apply to this graph? Select **ALL** that apply.
    - ☐ Undirected
    - ☐ Acyclic
    - ☐ Weighted

23. How would you solve this problem leveraging graph algorithms we've covered? Explain which algorithm you would use in words AND how you would use it to produce the **average herd size** amongst all observed elephants**.**