Room 309 HAV SEAT: 271

COMS W3134 Data Structures in Java – Section 1

Final Exam, Spring 2018

May 7, 2018

NAME:____________________

UNI:______________________

SECTION (1 or 2):__________

<u>YOU MUST SIT IN THE SEAT DESIGNATED AT THE TOP OF THE EXAM</u>. Failure to do so may result in a failing grade.

There are **11** questions on this exam totaling 150 points. The exam is closed book and closed notes. No calculators or computers are allowed. You will have 2 hours and 50 minutes to complete this exam. Do not open this exam packet until instructed.

Print your name and UNI on the exam. Read and sign the academic honesty statement below.

Place all Answers in this booklet. You may use the blank back of pages if you need additional space.

Academic Honesty Statement:

I certify that I have neither given nor received unauthorized help on this exam and that I did not use any notes, electronic devices, or other aids not specifically permitted. I will not discuss the content of this exam with anyone who is not taking the final at this time. I understand that any violation of this policy can result in an exam grade of zero and will be reported.

Signature:________________________

1. (12 points total) Using **induction** (other proof strategies will not be accepted), prove that a complete undirected graph of N vertices has N*(N-1)/2 edges. Recall that a complete graph is a graph in which each vertex is connected to every other vertex.

2. (20 points total) Run times (for big-O costs, provide as tight a bound as you can get) -

   a. (12 points) Give the worst-case big-O cost for the following four algorithms:

      i. Determining that a value is not in an AVL tree containing N nodes.

      ii. Construct a max-heap of N values, by performing N inserts on an initially empty heap.

      iii. Perform the topological sort algorithm on a DAG (V is the set of vertices, E is the set of edges). Consider the simple **non-queue** based implementation of the algorithm

      iv. Merging two independently sorted subarrays with a total size of N into a single sorted array using the **merge** operation from the merge sort algorithm.

   b. (4 points) Consider the Quicksort algorithm. Assume you have a pivot selection strategy that chooses the first element of the subarray as the pivot. Provide the big-O cost for running this algorithm until it terminates on

      i. an array of N distinct numbers already sorted in **increasing** order.

      ii. an array of N distinct numbers already sorted in **decreasing** order.

c. (4 points) Consider the Selection Sort algorithm. Provide the big-O cost for running this algorithm until it terminates on

i. an array of N distinct numbers already sorted in **increasing** order.

ii. an array of N distinct numbers already sorted in **decreasing** order.

3. (12 points) The following hash table uses quadratic probing with $f(i) = i^2$.

a. (7 points) Fill in the table by inserting the following integer keys in order using the hash function $hash(x) = x\%11$.

44, 22, 7, 21, 32, 12, 34

Index:	0	1	2	3	4	5	6	7	8	9	10
Key:											

b. (5 points) If I deleted the value 44 from the above hash table by just setting its location to NULL, identify one number that remains in the hash table but would no longer be findable using the contains method.

4. (16 points total) Implement the partition operation of the Quicksort algorithm in-place. You are given the array, as well as the left and right most index of the portion to partition. The code provided selects the center element as the pivot and moves it **to the right most element of the array**. You can assume that each element is unique. The provided code also swaps the pivot back into place for you after the partitioning is complete. Since you are directly swapping elements within the array, there is no need for a return value. Please fill in the following partition method:

```
void partition(int[] a, int left, int right) {
  // locate the pivot and swap it to the right
  int pivot = a[(left + right)/2];
  a[(left+right)/2] = a[right];
  a[right] = pivot;
  int i = left; int j = right - 1;
  // add your code to partition the array a here
```



```
  // swap the pivot back into place
  a[right] = a[i];
  a[i] = pivot;
}
```

5. (16 points total) A data structure to store trees with an arbitrary number of children at each node is:

```
public class TreeNode {
  Integer data;

  TreeNode firstChild;

  TreeNode nextSibling;

}
```

Write a **recursive** Java method to find the total number of leaf nodes in such a tree given the root:

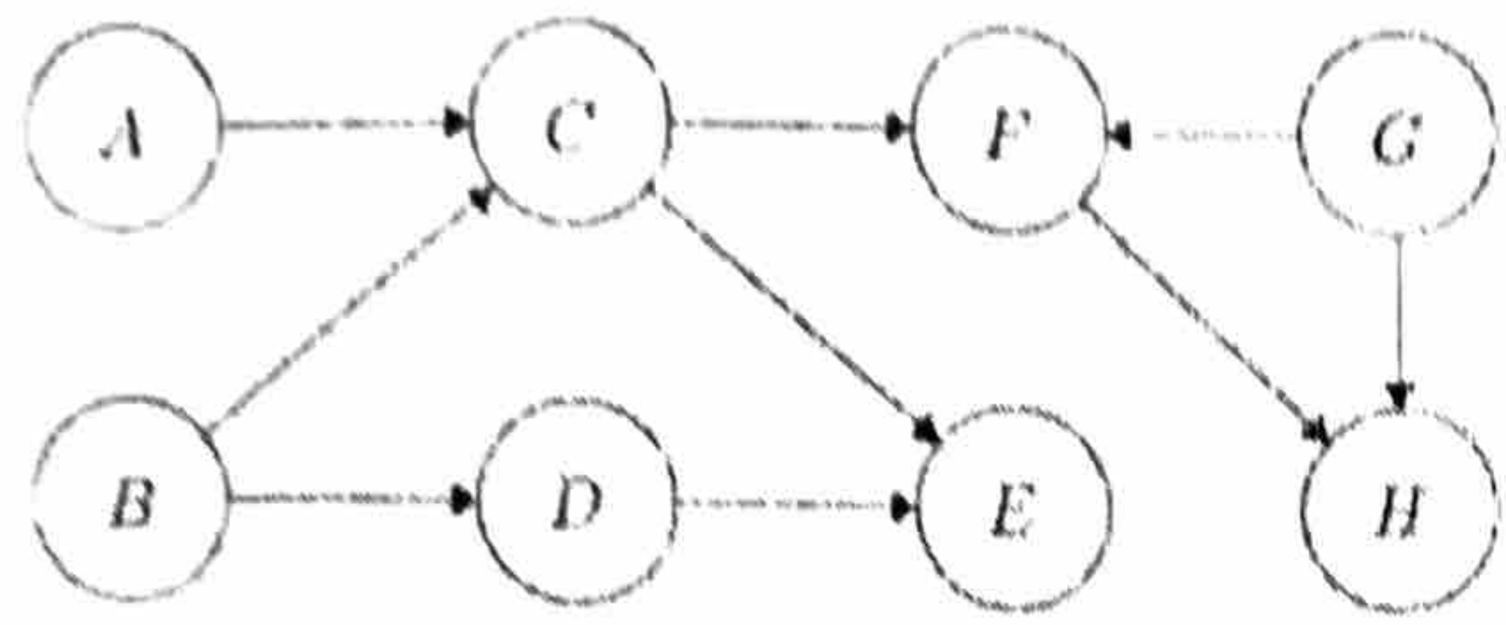
```
int countLeaves(TreeNode root)
```

6. (17 points total) Starting with an empty AVL tree, insert the following items one by one: 4, 6, 1, 8, 7, 5, 9. Show the tree before and after each required rotation (double rotations should show both single rotations involved).

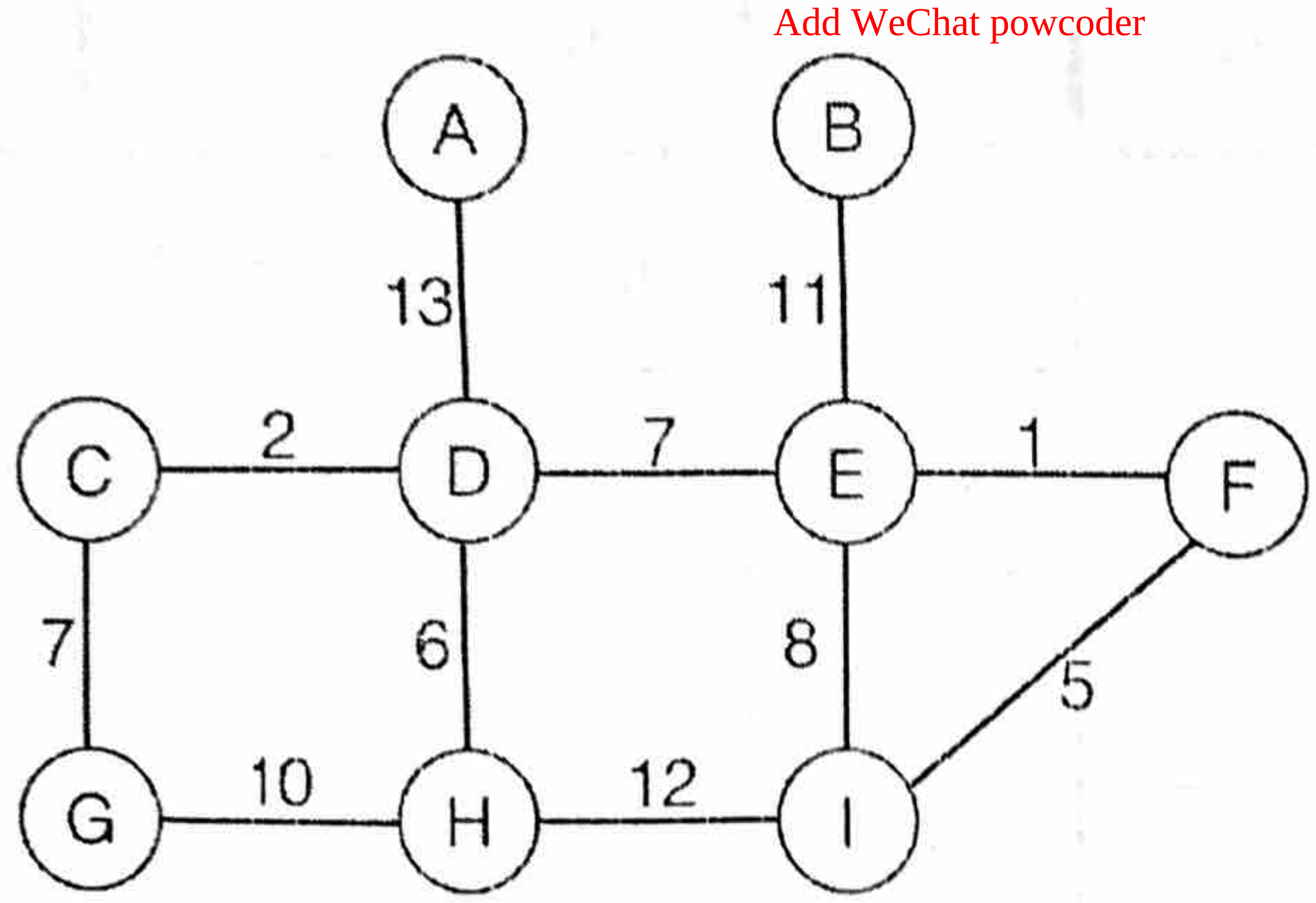
7. (15 points) Given an array of Integers A[1] through A[N], write a Java method that returns true if the array is max-heap ordered, false otherwise. (Remember that the location A[0] is unused in this implementation.)

8. (18 points total) For the following graphs:

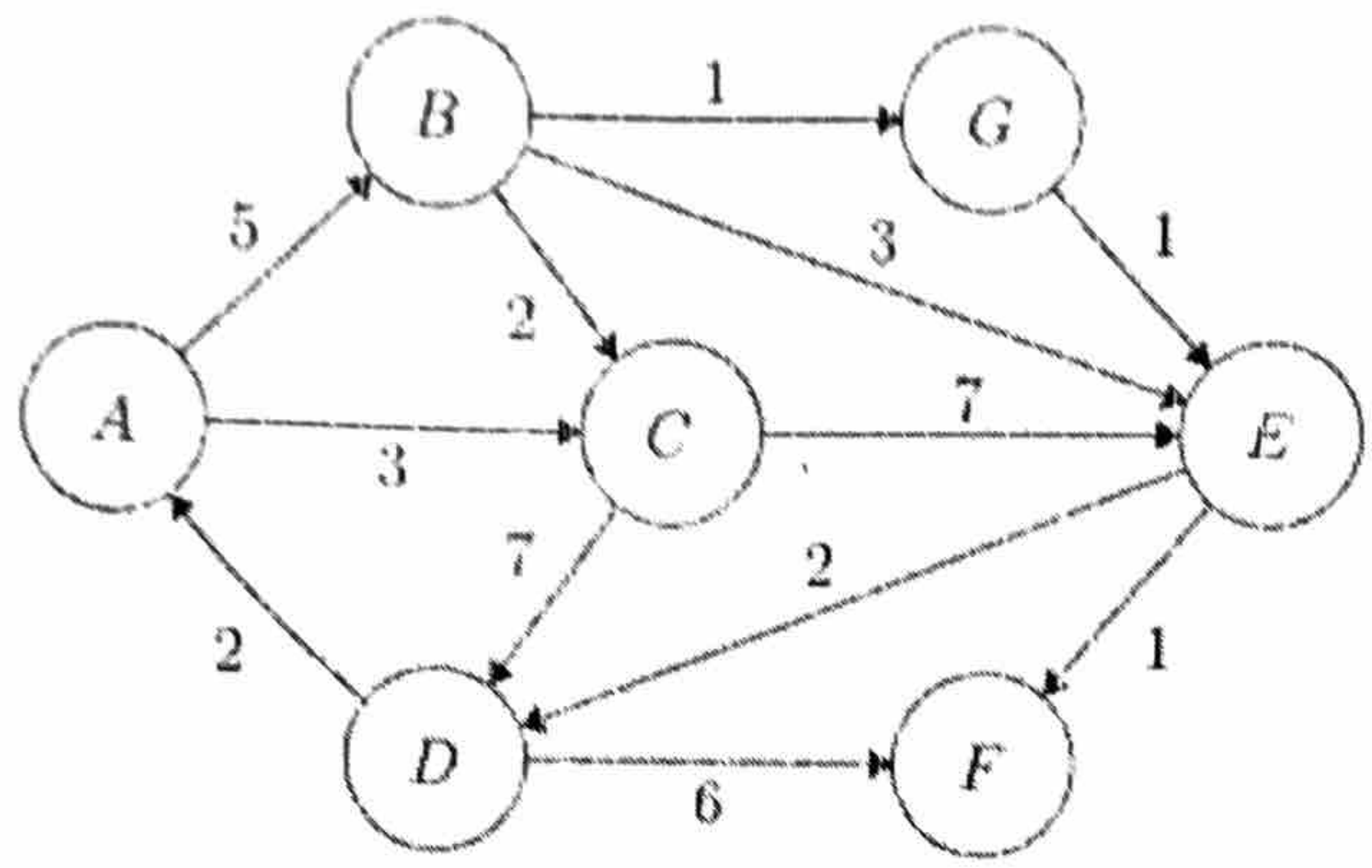
a. (8 points) Is the following graph weakly connected? Why or why not?



b. (10 points) Draw a minimum spanning tree of this graph using Kruskal's algorithm. You only need to show the resulting tree plus the order in which the edges were examined, and whether each edge was accepted or rejected.



9. (15 points) Given the directed graph below, show how Dijkstra's chooses a vertex at each step and updates the distances from the starting vertex. For your answer, fill in the vertices and distance information (what the textbook referred to as d_v), you do not have to list the previous field or the known field. Start at vertex **D** (I've provided the first step for you).



Step	**Vertex Chosen**	**A**	**B**	**C**	**D**	**E**	**F**	**G**
0	---	Infinity	Infinity	Infinity	0	Infinity	Infinity	Infinity
1	D	2	Infinity	Infinity	0	Infinity	6	Infinity

10. (7 points) If we were able to prove that the bounded Traveling Salesperson Problem (the version where we are looking for a tour of length < k) cannot be solved in better than $O(n^3 2^n)$ time, what would this say about the relationship between the class of problems P and the class of problems NP. Why?

11.(2 points) Describe the following graph: