Sorting

The eight numbers below are stored in an array. They are to be sorted using one of the following sorting algorithms. For each algorithm, show the state of the array after **each swap** that occurs, until the array is entirely sorted.

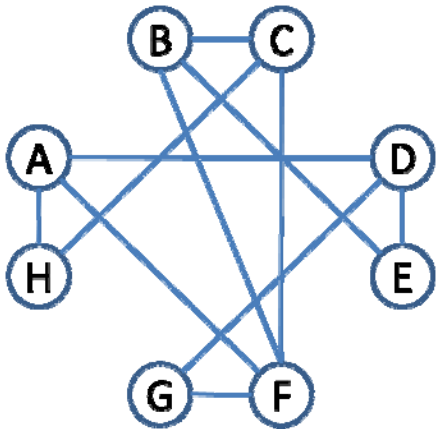
10	5	25	20	25	35	30	15

a. Insertion Sort

b. Heap Sort

Graphs

Give the order in which the nodes would be visited, for the undirected graph at right under the following graph traversal algorithms. To break ties, choose the node that comes first in alphabetical order.



a. Breadth-first traversal, staring at A.

b. Depth-first traversal, starting at F.

c. Dijkstra's Algorithm, starting at C and assuming that all edge weights are 1 except for CH = 7, AF = 4, and DG = 3.

Java References

Suppose that the diagram below shows the state of a graph data structure similar to the one we have worked with in class. For each of the following items, describe how you would reference the specified piece of data starting with the variable described.

Binary Search Trees

a. Draw the tree that would result from adding the following numbers, in order, to a BST that is initially empty, without rotations or rebalancing:

32, 16, 48, 24, 28, 2, 64, 10, 56, 4, 8, 52

b. Now draw the tree that would result from a right rotation of the root node, in your answer to part a.

c. Finally, please draw the result of deleting the root node of your answer to part b, using a **copy left** protocol to fill in the gap.

Hash Tables

You want to create a new hash function for strings, for an application where you will be storing people's names in "Firstname Lastname" format. You have three partners on the project, and each suggests a possible hash function, as follows:

a. Take the first four characters of the string (pad with zeros if necessary). This is four bytes, which is equivalent to an integer; the hashed key is this integer mod n.

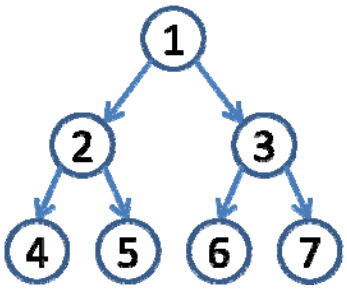
b. Like the above, on ly take the last four characters of the string (again, padding if necessary). Then take the remainder of this integer mod n.

c. Take all characters of the string in blocks of four, padding if necessary, to get multiple integers. Add them all together, and take the remainder mod n.

Your partners want you to make the final decision. Assess the three proposals in terms of their likely performance for your application. Finally, rank them from best to worst (or, if one or more are tied, say so).

Recursion

What would be the output of the following piece of Java code, assuming `tree` represents the data structure at right?




```
function parityTraverse(Tree<Integer> tree) {
   if (tree != null) {
      System.out.println(tree.getData());
      if (tree.getData()%2 == 0) {
         parityTraverse(tree.getRight());
      } else {
         parityTraverse(tree.getLeft());
      }
   }
}
```

Java Core

Match each description with a class or interface from the Java core. (Some may have more than one applicable answer; you only need to identify one correctly.)

a. A class that stores a sequence of entities of some type using a resizeable array as the underlying data type; part of the Java Collections framework.

b. Items to be displayed in a GUI window will be subclasses of this class. Several of its methods should be overridden by the subclasses, including `paintComponent`.

c. This class allows one to store data using an identifying key, and to quickly retrieve the data later using the same key.

d. This class contains all the methods used to draw graphical elements in a window, including `fillOval` and `drawLine`.

e. A class that stores a sequence of entities of some type in a manner optimized for fast insertion or deletion anywhere in the sequence; part of the Java Collections framework.

f. A class used to identify the points between elements in (a) or (e); its methods can be used to step through the sequence either forward or backward, and to insert or delete elements.

g. A Java Collections class with methods for last in, first out (LIFO) access.

h. A Java Collections class with methods for first in, first out (FIFO) access.