

## CS325 Assignment #2, Summer 2009, OSU

This assignment is due on Wednesday, 7/8/2009. You are required to **type your solution**, and turn it in printed at the beginning of class. Also print your source code and program output.

1. Using any language you want, implement merge sort. Run it on the following list of numbers:

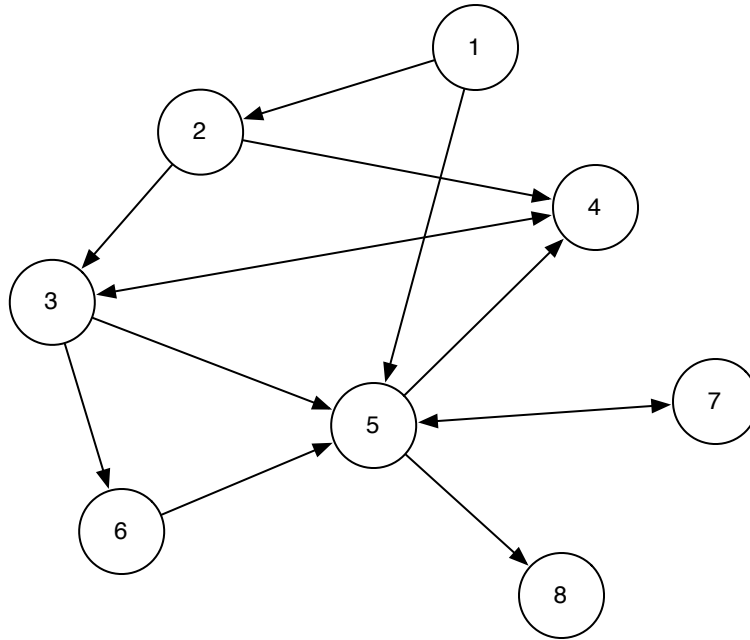
56, 93, 65, 34, 13, 19, 88, 84, 67, 48, 51, 21

Each time your main recursive function is called, output the following information (print each item on one line, and print “\*\*\*\*” at the very beginning of the function):

- The input list that the merge sort function is being called on
- The left and right lists it splits into, before recursively calling merge sort
- The left and right lists, after they are recursively sorted
- The result of merging the left and right lists

Turn in the printed output from this program, as well as the source code (give me a clean copy without all of the extra output, as well as the version with all of the code to output the above info). Print your source code on paper with ink!

2. Run your sorting algorithm on randomly generated lists of items of size 1 to 5,000. Measure the time in milliseconds it takes to sort each list. Give me a graph of the runtime vs. list size.  
Extra credit: Rather than just measuring the runtime of the algorithm once for each size of list, measure it 100 times on different random lists. Give me a graph of the average runtime, including error bars that represent the standard deviation.



3. Consider the graph above. Note that it is a directed graph. A double arrow between two nodes  $a$  and  $b$  indicates that there is an edge from  $a$  to  $b$ , and an edge from  $b$  to  $a$ .
  - a. Write the adjacency list representation of the graph. The elements of each list should be sorted in increasing order (i.e. if vertex 1 has neighbors 2,3 and 7, its list should be 2,3,7 and not 3,2,7, for example).
  - b. Using the representation from part a., find the DFS tree for the graph. Run the version covered in class using a stack, not the recursive version!
  - c. Using the representation from part a., find the BFS tree for the graph.
  - d. For each of DFS and BFS, list the order in which it first 'discovers' each node (i.e. when it marks it as visited). Your answer will be considered wrong if you do not use adjacency lists that are sorted in increasing order.

**For parts b. through d., assume you start at node 1.**