Graduate Comprehensive Exam
Data Structures and Algorithms
Fall 1999

Answer all questions on the exam. Total: 100 points. Good Luck.

1. (20 pts) Specify the worst-case running time for each of the following algorithms (assume n elements to be sorted):
   (a) Bubble sort
   (b) Insertion sort
   (c) Selection sort
   (d) Quicksort
   (e) Mergesort
   (f) Heapsort
   (g) Treesort

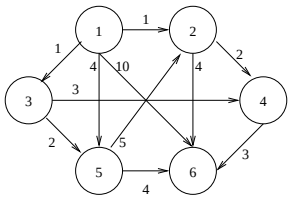
2. (20 pts)
   (a) Insert the following elements into a heap, one at a time. You should show the heap after each insertion, however, you do not need to show the heap as elements "percolate" through the heap. Simply show its contents after each insertion has been completed.

   3 19 1 12 11 15 0

   (b) Now perform three "DELETE" operations on the heap, and then insert the elements 10, 14, and 8. As in part (a), you must show the contents of the heap after each DELETE and after each INSERT.

3. (40 pts)
   (a) What property does an array have to have before *binary search* can be performed?
   (b) Using C, C++, Ada or pseudocode with sufficient details, write a function that performs *binary search* for a key on an array of integer elements and returns the array index of the location of the key. The function returns -1 if the key is not in the array. Provide two versions (be sure to specify all parameters):
      i. iterative (non-recursive) version
      ii. recursive version
   (c) For the iterative version, describe when the "best" case occurs and analyze the run-time complexity using big-O notation (assuming n is the number of elements in the array). Explain your answer.
   (d) For the iterative version, describe when the "worst" case occurs and analyze the run-time complexity using big-O notation (assuming n is the number of elements in the array). Explain your answer.
   (e) If the array does not satisfy the property in part (a), describe and analyze the best-case **and** worst-case run-time complexities that include both the worst-case run-time complexity to achieve the property in part (a) and binary search. Explain your answer.

4. (20 pts) Given the following graph:



   (a) Perform breadth-first search (BFS) starting from Node 1 (successors are visited in ascending order of the values), state the order of the nodes visted.
   (b) Perform depth-first search starting (DFS) from Node 1 (successors are visited in ascending order of the values), state the order of the nodes visted.
   (c) To find a path from Node 1 to 6, BFS can stop when Node 6 is visited. What is the found path and its length from Node 1 to Node 6?
   (d) Similar to part (c), DFS can stop when Node 6 is visited. What is the found path and its length from Node 1 to Node 6?
   (e) What property does a directed graph need to have so that the alogrithm in part (c) [BFS stops when the destination node is visited] always finds the shortest path between the source and destination nodes.