CSCE 221 Test 2

Topics Covered by Test 2

1. Singly and doubly linked list data structures.

   (a) Write C++ functions for linked lists: insert a new node, delete a node, copy constructor, copy assignment operator, destructor, search for a given item, get the number of items in a list.

   (b) Provide running times for the operations listed in (a).

2. Divide and conquer algorithms.

   (a) Write a running time recurrence equation for a recursive algorithm.

   (b) Provide a description or write a code for a recursive algorithm based on a given running time recurrence relation.

   (c) Solve a running time recurrence equation using the iterative and/or recursive tree method.

   (d) Solve a running time recurrence equation using the Master theorem, if it is applicable, to obtain a running time in Big-O asymptotic notation. The Master theorem will be provided on the test.

3. Recursive sorting algorithms.

   (a) Merge sort

      i. How is the division of an array/subarray done by this algorithm during the partition steps?

      ii. How many recursive steps are required by this algorithm to reach the base case (= an array with one element)?

      iii. What is a run time recurrence equation for the merge sort algorithm to sort any input of size n?

      iv. Use the Big-O notation to estimate the number of comparisons done by this algorithm at each level of the recursive tree during the conqueror steps.

      v. Is the merge sort an in-place algorithm?

      vi. Use the Big-O asymptotic notation to express the number of comparisons required to sort an input of the size n.

      vii. What is the number of comparisons done by the merge sort on an input which is already sorted? Provide a solution using Big-O asymptotic notation.

      viii. Use the Big-O notation to provide running times for the best, average and worst cases of this algorithm.

   (b) Quick sort

      i. What are recurrence relations for the best, average and worst cases of running times of the algorithm?

      ii. Use the Big-O notation to provide running times for the best, average, and worst cases of the algorithm.

      iii. Provide an input and select the pivot point to get the worst case.

      iv. Provide an input and select the pivot point to get the best case.

4. Applications of the stack and queue ADT.
    (a) Stack and queue ADT and their implementations using linked lists.
    (b) Convert an algebraic expressions from infix to its postfix form using queue and stack. For example, $(x+y)^2+(x-4)/3$.
    (c) Use postfix form to evaluate an expression for $x=2$ using a stack.
    (d) Build a binary expression tree starting from its postfix form using a stack. Illustrate all the merging steps during the construction of the tree.
    (e) Use the obtained binary tree to evaluate the expression for $x=2$. Which tree traversal operation should be used?
5. Binary trees.
    (a) Definitions and properties of proper and extended binary trees.
    (b) Tree traversal operations: in-order, pre-order and post-order.
    (c) Complexity of the worst and best cases for the insert, search, find, min/max and delete operations for balanced and unbalanced binary search trees.
    (d) The algorithms and their illustrations for the operations: create a tree; insert and delete a tree; find max/min value in a tree; remove max/min value from a tree.
    (e) What is the complexity of building a balanced and unbalanced binary search trees?
    (f) Provide a sorting algorithm based on binary search trees.
    (g) What is the complexity of the sorting algorithms based on a balanced and unbalanced binary search tree.
    (h) Provide running times for all the above operations.
6. Balancing techniques: AVL, Red Black, and and 2-4 trees. What is the motivation for them?
7. Skip lists.