

Fall 2010

C343 Midterm

Name:

This exam is closed-book. There are 10 questions worth a total of 100 points.

Question 1. (10 points) Find reasonable values of c and n_0 to show that:

a) $3n + n^2 + 2n^3 = O(n^3)$

b) $\sqrt{n} + 2\lg\left(\frac{n}{4}\right) = O(n \lg n)$.

Question 2. Suppose that you start with an empty growable array, which has an initial capacity of 1 and doubles in size each time it fills up. You then insert n elements into it, one at a time. Assume that n is a power of two.

a) (5 points) How many times will the array grow?

b) (5 points) How many array item assignments (including copies) will take place in total?

c) (5 points) In his undergraduate days, when Professor Bunyan learned that adding to a growable array could take $O(n)$ time in the worst case, he exclaimed: "Aha! Then if I add n items, it will take $O(n^2)$ time." His teacher promptly struck him with a rolled-up newspaper. What was wrong with his statement?

Question 3. Suppose you are given a doubly-linked list of integers named *list* which is sorted in ascending order. The list has a head pointer *list.head* and a tail pointer *list.tail*, and it may or may not be empty. If it is empty, then its head and tail will both point to null.

a) (10 points) Write a pseudocode procedure *insertInOrder(list, value)* which takes this list and inserts a value into it, maintaining sorted order.

b) (5 points) When implementing a doubly linked list, you may choose to mark both ends of the list with empty "sentinel" nodes rather than null pointers. How could those nodes be used to simplify your code from part a?

Question 4.

a) (5 points) Give an advantage that a growable array has over a linked list, and a situation where that would be useful.

b) (5 points) Give an advantage that a linked list has over a growable array, and a situation where that would be useful.

Question 5. (5 points) Java's collection hierarchy includes a *Stack* class which extends *Vector*. Explain why the name of this class is misleading, and what could have been done instead.

Question 6.

a) (10 points) Explain briefly how you could implement a queue using two stacks. Specifically, you should describe how to both *enqueue* and *dequeue* items from the queue, and how to tell when the queue is empty. You may use diagrams to help your explanation if you wish.

b) (5 points) Give the running times of the *enqueue* and *dequeue* operations on your queue when the stacks are implemented with linked lists.

Question 7. (5 points) The running times of binary search tree operations are typically given in terms of h (the height of the tree) rather than n (the number of nodes). Why is this?

Question 8. (5 points) Prove by contradiction that the leftmost item in a binary search tree is the smallest one.

Question 9. (5 points) Give a formal definition of what it means for a binary search tree to be *height-balanced*, and explain why it is a useful property for a tree to have.

Question 10. The following is pseudocode for the quicksort algorithm, per Wikipedia:

```
function quicksort(inputList)
  var list less, greater
  if length(inputList) ≤ 1
    return inputList
  select and remove a pivot value pivot from inputList
  for each x in inputList
    if x ≤ pivot then append x to less
    else append x to greater
  return concatenate(quicksort(less), pivot, quicksort(greater))
```

The performance of this algorithm is highly dependent on how successful the underlined step is at choosing a good pivot value. For each of the following cases, give a recurrence relation $C(n)$ for the number of **comparisons** performed on a list of n items, then solve the recurrence and give its big- O . For simplicity, assume that concatenate() and selecting a pivot both require no comparisons.

a) (10 points) Worst case: the pivot splits the input list into chunks of size 1 and $(n-1)$ elements.

b) (5 points) Best case: the pivot value splits the list into two equal chunks.

Extra credit. (10 points) Consider a complete binary search tree containing the keys: {1, 2, 3, 4, 5, 6, 7}. How many different ways could the keys have been inserted to produce this specific tree?

Scratch Space