1. Best case run-time of Insertion Sort in terms of input size n

2. What is the **worst-case** run-time of Insertion Sort in terms of input size n?

3. If you want a guarantee that an algorithm will never take more than a certain amount of time for a given input size, which analysis do you need?

4. Express the following as Θ notation: $7n^3 + 4n + 99$

5. Define the three criterias for the loop invariant

6. Use induction to prove the following

$$\sum_{k=0}^{n} k^3 = \frac{(n)^2(n+1)^2}{4}$$

7. Express the function $\frac{n^3}{1000} - 100n^2 - 100n + 3$ in terms of θ notation.

8. What is the tightest bound on the worst case asymptotic cost of queue enqueue and dequeue operations under the array implementations discussed in the text and lecture?

9. Consider how Merge-Sort implements Divide and Conquer strategy. At a given call to Merge-Sort:

   (a) What is the cost to divide a problem of size n into sub-problems?

   (b) What is the cost to combine the solutions to subproblems?

   (c) How many subproblems does Merge Sort divide a problem of size n into?

   (d) What is the size of each subproblem as a function of n?

10. What is the worst case and the best case run time for Merge Sort, respectively?

11. What is the tightest bound on the worst case asymptotic cost of stack push and pop operations under the array implementations discussed in the text and lecture?

12. Now write the recurrence relation T(n) for the runtime of Merge Sort. We assume a constant time c to solve the base case of sorting one item:

$$T(n) = c$$

    if n < 2.
    Write the expression for n > 1 (the "otherwise" clause) using the formula T(n) = a($\frac{n}{b}$) + D(n) + C(n) and using θ to combine the terms for D(n) and C(n):

13. Suppose we have a binary tree of n nodes, each node having a key (their ordering is not important for this question), and using a linked node representation (e.g., nodes of class TreeNode as described in the lectures). What is the asymptotic run time of the fastest algorithm to print out the keys of all n nodes of the binary tree?

14. What is the tightest bound on the worst case asymptotic cost of the fastest algorithm for searching an ordered doubly linked list for a given key?

15. What is the tightest bound on the worst case asymptotic cost of the fastest algorithm for searching an unordered doubly linked list for a given key?

16. Given a pointer to a list cell in a singly-linked list, what is the expected asymptotic run time of the fastest algorithm to insert a new cell after that cell?

17. Given a pointer to a list cell in a singly-linked list, what is the expected asymptotic run time of the fastest algorithm to delete that cell?

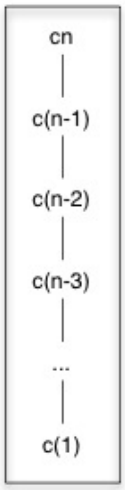
18. What is the worst case time to search for a single key in a direct address table of size m that stores n elements?

19. What is the expected (average) case time to search for a single key in a hash table with chaining of size m that stores n elements and uses a uniform hash function?

20. What is the tightest bound on the worst case asymptotic cost of the fastest algorithm for insertion of a new key into an unordered doubly linked list?

21. What is the worst-case time to search for a single key in a hash table with chaining of size m that stores n elements?

22. Write the recurrence relation for the runtime cost of the recursive procedure for which this recursion tree represents the recursion



23. Use the Master Method (see above) to solve the recurrence relation:

$$T(n) = T(n/2) + n^2$$

24. What is the recurrence that defines the runtime of this algorithm?

Consider the following algorithm.

```
whiskey_tango_foxtrot(A[1..n])
1     if n==1
2         return A[1]
3     whiskey_tango_foxtrot(A[1..(n/3)])
4     whiskey_tango_foxtrot(A[(n/3)+1..(2n/3)])
5     whiskey_tango_foxtrot(A[(2n/3)+1..n])
6     for i = 1 to n-1
7         for j = i+1 to n
8             if A[j] < A[j-1]
9                 exchange A[j] and A[j-1]
```

25. Use the Master Method (see above) to solve the recurrence relation:

$$T(n) = 2T(n/2) + n^2$$

26. How may leaves are there in a complete binary tree of height 10?

27. Which traversal would you use to get the keys of a binary search tree printed out in sorted order (smallest to largest)?

28. The Binary Search Tree property has two parts. Which of these two are correct statements of those parts of the BST property

29. How may internal vertices are there in a complete binary tree of height 10?

30. Give the upper and lower bounds for T(n) in each of the following recurrence relation.

(a) $T(n) = 4T(\frac{n}{3}) + n \log n$

(b) $T(n) = 3T(\frac{n}{3} - 2) + \frac{n}{2}$

(c) $T(n) = 4T(\frac{n}{2}) + n^2\sqrt{n}$

(d) $T(n) = T(n-1) + n$

(e) $T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + n$

31. Demonstrate what happens when we insert the keys 5, 28, 19, 15, 20,33, 12, 17, 10 into a hash table with collision resolved by chaining. Let the table have 9 slots and let the hash function be h(k) = $k \mod 9$.

32. For the set of $\{1, 4, 5, 10, 16, 17, 21\}$ of the keys draw the binary search tree of height 2.

33. Suppose that we have numbers between 1 and 1000 in BST, and we want to search for the number 363. Which of the following could <u>not</u> be the sequence of node examined?

    (a) 2, 252, 401, 398, 330, 344, 397, 363

    (b) 924, 220,911, 244, 898, 258, 362, 363

    (c) 2, 399, 387, 219, 240, 912, 245, 363

34. Fill in the table for these pairs of functions for the asymptotic growth.

f(n)	g(n)	O	0	Ω	ω	Θ
$2^{\log n}$	$\log^2(n)$					
$\sqrt{n}$	$n^{\cos(n)}$					
2^n	$2^{\frac{n}{2}}$					

35. Let f(n) and g(n) be asymptotically positive functions. Prove or disprove each of the following conjectures.

    (a) f(n) = O(g(n)) implies g(n) = O(f(n))

    (b) f(n) + g(n) = $\Theta(\min(f(n), g(n))$

    (c) If f(n) and g(n) are monotically increasing functions then so are the functions f(n) + g(n) f(g(n)), f(n) $\times$ g(n)