Homework 0 (Introduction)
Due 9/8

1. (35%) Go to http://framingham.blackboard.com, “63.347 Analysis of Algorithms”. Read the message in thread “Welcome” of Discussion Board forum 0, and respond to it by leaving your own message.

2. (50%) Using the definition of a tree and the notion of induction, explain in your own words why every tree $G = (V, E)$ has exactly one more vertex than its number of edges. (If necessary refer to data-structures or discrete-math materials on trees; e.g., instructor’s Data Structures slides on trees, instructor’s Theory of Computing slides on induction (topic 0 or 1).

3. [TO DELETE]When you have a session with your Internet browser, is that one execution of an algorithm by the browser? Why or why not?

4. (15%) What are the domain and the range of the square-root function?

[WANT NEW Q’s THAT REALLY MK USE OF DS, DM I.]
Homework 1 (Verification)
Due 9/20

1. (35%) By use of preconditions, postconditions, and loop invariants, prove or argue persuasively that the pseudocode below will correctly count and return the number of spaces in a string, $s$. Note that a loop invariant is an assertion about values of data items, not a narrative of what happens.

```
Count-spaces(s)

n ← 0
i ← 1

while $i \leq \text{length}(s)$ do
    if $s[i] = \ ' \ ' \text{then}$
        $n ← n + 1$
    $i ← i + 1$

return $n$
```

2. (35%) Relate the above algorithm to the Hoare triple $<\phi>P<\psi>$ and explain how you would go about proving the correctness of $\text{Count-spaces}$. 

3. (30%) In Computation Tree Language, express the assertion that no path exists in which $\phi$ will never hold. (Practical example: In an operating system, no process is permanently blocked from access to a requested resource.)
Homework 2 (Recurrences; divide-and-conquer)
Due 10/6

1. Levitin, pp. 59-60, Ex. 1, 2(b, c), 3 (b, c, d), 5, 8(a). Show your work.

2. p. 67, Ex. 1 (g) (summations; show work),
   2 (d) (summation; show work),
   4 (mystery algorithm)

3. p. 76, Ex. 1 (b, c) (solve recurrences),
   3 (solve recurrences)

4. Consider the problem of finding the largest element of a sequence in an array. Signature:
   \text{Max}(A, \text{first}, \text{last}), \text{ where } A \text{ is an array, first and last are subscripts. Algorithm should return the}
   \text{largest element of the sequence from } A[\text{first}] \text{ to } A[\text{last}], \text{ inclusive.}

   a. Express an algorithm \emph{iteratively} (in pseudocode or a programming language) to solve this
      problem. Use verification methods to show correctness.

   b. Write a \emph{recursive} version of this algorithm.

   c. Write a recurrence that defines the function computed.

   d. Write a recurrence, based on (c), that defines the time function, i.e., the function on the array
      sequence size \( n = \text{last} – \text{first} + 1 \) that returns the running time.

   e. Solve the recurrence, giving a one-sentence explanation.

5. p. 128, Ex. 1 (array maximum)

6. p. 135, Ex. 7 (Quicksort average-case recurrence)

7. p. 139, Ex. 6 (binary search for elements in a range)

8. p. 143, Ex. 2 (number of leaves in a binary tree),
   8 (internal path length in binary tree)

9. p. 154, Ex. 3 (closest pair of points on plane) – pseudocode is OK

10. What is worst-case time for binary search, and why?

11. p. 170-171, Ex. 1 (depth-first search of graph)
Homework 3 (Brute force, greedy, transform-conquer)  
Due 10/25

**Brute force**

1. Name a brute-force algorithm for sorting an array, and one for searching an array, and state why they may be considered brute-force.

2. Levitin, p. 112, Ex. 4 (odd pie fight; show work)

3. p. 113, Ex. 9 (convex hull by brute force, using C, C++, Java, etc.)

4. p. 119, Ex. 1(a) (Running time for brute-force solution to traveling salesperson problem)

**Greedy**

5. p. 314, Ex. 7(a) (Apply Prim’s algorithm to pictured graph)

6. p. 322, Ex. 10 (Compare efficiencies of Prim and Kruskal algorithms experimentally)

7. What is the running time of the Dijkstra algorithm and why?

**Transform and conquer**

8. p. 201, Ex. 2(a) (closest pair of numbers in array)

9. p. 229, Ex. 2 (heap recognition)
Homework 4

Due 11/2

4. Levitin, p. 271, Ex. 7
5. p. 276, Ex. 3
6. p. 283, Ex. 4(a)
7. p. 292, Ex. 1 (See D. Keil C++ code, available in course docs)
8. p. 303, Ex. 2(a)
9. p. 350, Ex. 4(a)
10. p. 362, Ex. 4(a)
11. p. 369, Ex. 1(a)
12. p. 375, Ex. 2
13. p. 375, Ex. 5(a)
1. Concerning the following formula in propositional logic 
\((p \lor q \lor r) \land (p \lor \neg q \lor r) \land (\neg p \lor \neg q \lor \neg r)\)

(a) State whether the formula is satisfiable, showing your work (you may write a truth table);
(b) State and explain what is the time necessary to answer the question for arbitrary formulas with \(k\) variables.

2. Suppose someone found an \(O(n^4)\) solution to the Traveling Salesperson problem.
   (a) What implications would this have for other problems reducible to TSP in polynomial time?
   (b) What implications would it have for other problems to which TSP is reducible in polynomial time?

3. Levitin, p. 385, Ex. 3(a), 3(c) (Lower bound classes for maximum and subset problems)

4. p. 393, Ex. 9(a)-(b) (Tournament tree, number of games and rounds)

5. p. 402, Ex. 7(a) (State the decision version of knapsack problem)

6. p. 423, Ex. 3(a) (Backtracking for n-queens problem; pseudocode plus statement of rules is OK)

7. p. 432, Ex. 1 (Data structure to keep track of live nodes in best-first branch-and-bound algorithm)
Homework 6
Due 11/22

1. Levitin, p. 451, Ex. 5

2. Describe how you might write a program that writes programs by evolutionary computation; e.g., your program is given as input a postcondition that specifies program $P$’s output as a function of input, and your program will output program $P$. 
Homework 7
Due 12/1

1. Describe two models of parallel computation.

2. Argue for or against the possibility of super-linear speedup; i.e., parallel algorithms whose running times are less than $O(t/n)$, where $t$ is the running time of the best sequential algorithm to solve the problem and $n$ is the number of processors in the parallel system.

3. What factors other than those considered in serial algorithm analysis must be considered in performance analysis of parallel and distributed systems?
1. Argue that interaction is more powerful than algorithms.
2. Argue that interaction is not more powerful than algorithms.
3. Argue that multi-stream interaction is more powerful than sequential interaction.
4. Argue that an organization of many people will take longer to solve a problem than a team of two.