CS 410/584: Test 1, 29 April 2009 Name:____________

You may use the course text and course materials. You may not use computers, other books or other materials. Work individually.

100 points possible. Give your answers on these sheets. (You can ask for extra blank sheets if needed, or continue on the back. Put your name on any extra sheets.)

1. **Sorting** (15 points) Suppose SRT is a comparison-based sorting method that never exchanges items more than k positions away. Explain why SRT has worst-case time complexity (counting comparisons) in $\Omega(n^2/k)$.

2**. Minimum Spanning Tree** (10 points) Let G(N, E) be an undirected, connected graph with edge costs. Suppose e is an edge in E with higher cost than any other edge in E. Explain why e will be in a minimum spanning tree for G exactly when there is no simple cycle that contains e.

3. **Order Notation** (20 points) In each of the parts below, assume f(n) and g(n) are positive functions that are both in $\Theta(n^2)$.

a. Give an example where $g(n) - f(n) \in \Theta(n \lg n)$.

b. Give an example where $g(n) - f(n) \in \Theta(n)$.

c. Give an example where $g(n) - f(n) \in \Theta(1)$.

d. Explain why there is no example where $g(n) - f(n) \in \Theta(n^3)$.

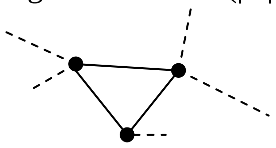
4. **Random** (0 points) Did you study for this question?

5. **Amortized Cost** (10 points) Recall the example from class of implementing a queue **Q** with two stacks **SI** and **SO**. **SI** represents the back of **Q**, and **SO** the front of **Q**. An *enqueue* operation on **Q** is implemented as a *push* on **SI**, and a *dequeue* operation as a *pop* on **SO**, moving the contents of **SI** to **SO** if necessary.

a. Suppose we also support a "*requeue*(x)" operation on **Q** that adds element x to the front, by performing *push*(x) on **SO**. Explain why a sequence of n *enqueue*, *dequeue* and *requeue* operations can still be done with O(n) stack operations.

b. Now suppose we also support an "*unqueue*" operation on **Q** that removes an element from the back, by performing a *pop* on **SI**, with a move of the contents of **SO** to **SI** if **SI** is empty. Explain why a series of n *enqueue*, *dequeue*, *requeue* and *unqueue* operations might now take more than O(n) time.

6. **Graph Algorithms** (20 points) A *triangle* in an undirected graph G = (N, E) is a cycle of length 3. One could search for a triangle by considering all triples of nodes in N, giving an algorithm with $\Omega(|N|^3)$ time complexity.



Explain how to find if G has a triangle in O(|N| + |E|) time. Illustrate your method on the graph G = ({a, b, c, d, e, f, g}, {{a,b}, {a,e}, {b,d}, {c,d}, {c,f}, {c,g}, {d,e}, {e,f}, {f,g}}).

410 Only

7. **Dynamic Programming** (25 points) Let A[1..n] be an array of floating-point numbers.
a. Describe an O(n) algorithm to find values i and j that maximize the sum of the values A[i] through A[j], that is A[i] + A[i + 1] + … + A[j]. If you give pseudocode, also provide an English explanation.

b. Illustrate your method on array A below.

i	1	2	3	4	5	6	7	8
A[i]	3.1	-4.2	6.7	-3.2	7.5	2.0	-5.4	4.7

584 Only

7. **Dynamic Programming** (25 points) We have n people with integer weights w_1, w_2, …, w_n. There are two elevators, and we want to know if we can divide the people into two groups that will put equal weights on each elevator. (For example, with weights 3, 4, 6, 2, 11, 4 we can split them into Group 1 = 4, 11 and Group 2 = 3, 6, 2, 4, each with total weight 15).

a. Describe a method to solve the problem in $O(n^2W)$ time, where W is the maximum weight of any person. If you give pseudocode, also provide an English explanation.

b. Illustrate your method with the weights 2, 3, 2, 7, 2.