

**CS 410/586: Test 1, 30 April 2008 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. Put your name on any extra sheets.)

1. **Order Notation** Given functions  $f(n)$  and  $g(n)$  with  $g(n) \in O(f(n))$ , define  $h(n) = g(n) - f(n)$ .

a. (5 points) Give examples of  $f(n)$  and  $g(n)$  where  $O(h(n)) = O(f(n))$

b. (5 points) Give examples of  $f(n)$  and  $g(n)$  where  $O(h(n)) \neq O(f(n))$ .

2. **Heapsort** (10 points) Heapsort of  $n$  elements is based on a binary tree of depth  $\log_2 n$ . We could modify Heapsort to work on a 3-way tree of depth  $\log_3 n$ . Will using a 3-way tree reduce the number of comparisons for Heapsort by a factor of  $(\log_3 n / \log_2 n)$ ? Why not?

3. **Heap Deletion** Let array  $A$  hold a binary heap (represented implicitly), where  $A[k]$  is currently the last element. Consider the following  $O(\log k)$  method to delete the item at  $A[i]$  from the heap.

1.  $A[i] \leftarrow A[k]$
2. Set the end of the heap to be  $A[k - 1]$
3. Call  $\text{max-heapify}(A, i)$ .

a. (10 points) Give an example of how this method can produce a result that is **not** a heap.

b. (10 points) Explain how to fix the method while still getting  $O(\log k)$  time complexity.

4. (0 points) Don't you wish this question was worth 10 points?

**5. Dynamic Programming** (15 points) Consider an array  $B[1 .. n]$  of integers (positive or negative). For example

B[1]	B[2]	B[3]	B[4]	B[5]	B[6]	B[7]	B[8]	B[9]	B[10]	B[11]	B[12]
6	3	-4	2	3	-6	2	1	-5	-3	2	5

Define  $SUM[i, j] = B[i] + B[i+1] + \dots + B[j]$ . For the example above,  $SUM[1, 4] = 7$  and  $SUM[6, 9] = -8$ .

Suppose we want to find  $i$  and  $j$  that maximize  $SUM[i, j]$ . Directly computing  $SUM[i, j]$  for all  $i < j$  is  $O(n^3)$ .

Describe an algorithm to find the maximum  $SUM[i, j]$  that has better time complexity than  $O(n^3)$ . [You do not need to give an example execution.]

6. **Greedy Algorithms** Consider an elevator with maximum capacity  $M$  (say, 1000 pounds). We have a collection  $P$  of people of various weights, and we want to know who to put on the elevator, in order to maximize some goal. Consider two different goals, and possible greedy choice steps for them:

a. (10 points) The goal is to put the **most weight** on the elevator without exceeding the maximum load  $M$ . Consider the following greedy step:

*Add the **heaviest** person from  $P$  to the elevator who won't cause the total weight to go over  $M$ .*

Will a greedy algorithm based on repeating this step achieve the goal? Why or why not?

b. (10 points) The goal is to put the **most people** on the elevator without exceeding the maximum load  $M$ . Consider the following greedy step:

*Add the **lightest** person from  $P$  to the elevator who won't cause the total weight to go over  $M$ .*

Will a greedy algorithm based on repeating this step achieve the goal? Why or why not?

8. **Graph Algorithms** A vine in an undirected graph is a path  $n_1, n_2, \dots, n_k, k \geq 3$ , where each of  $n_2, \dots, n_{k-1}$  has exactly 2 edges (namely the ones on the path). In the graph below:

G, I, J is a vine

A, L, M, N, H is a vine

A, K is not a vine (too short)

A, B, C, E is not a vine (C has 3 edges)

a. (15 points) Describe an  $O(|E|)$  algorithm to find the length of the longest vine in a connected graph  $G = (N, E)$ . [Or, if you want, an algorithm to find all the maximal vines – vines that can't be extended.]

b. (10 points) Illustrate the operation of your algorithm on the graph above.