Note: All the questions in this session are taken from past CS473 midterms.

1. (Fall 2006) **Multiple Choice:** Each of the questions on this page has one of the following five answers: For each question, write the letter that corresponds to your answer.

A: $\Theta(1)$	B: $\Theta(\log n)$	C: $\Theta(n)$	D: $\Theta(n \log n)$	E: $\Theta(n)$

   (a) What is $\frac{5}{n} + \frac{n}{5}$?

   (b) What is $\sum_{i=1}^{n} \frac{n}{i}$?

   (c) What is $\sum_{i=1}^{n} \frac{i}{n}$?

   (d) How many bits are required to represent the nth Fibonacci number in binary?

   (e) What is the solution to the recurrence $T(n) = 2T(n/4) + \Theta(n)$?

   (f) What is the solution to the recurrence $T(n) = 16T(n/4) + \Theta(n)$?

   (g) What is the solution to the recurrence $T(n) = T(n-1) + \frac{1}{n^2}$?

   (h) What is the worst-case time to search for an item in a binary search tree?

   (i) What is the worst-case running time of quicksort?

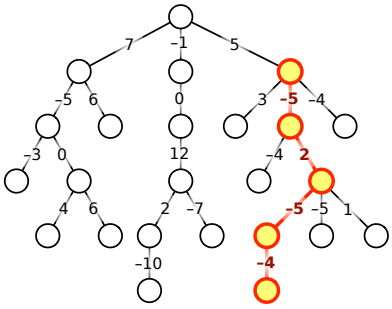
   (j) What is the running time of the fastest possible algorithm to solve Sudoku puzzles? A Sudoku puzzle consists of a 9×9 grid of squares, partitioned into nine 3×3 sub-grids; some of the squares contain digits between 1 and 9. The goal of the puzzle is to enter digits into the blank squares, so that each digit between 1 and 9 appears exactly once in each row, each column, and each 3×3 sub-grid. The initial conditions guarantee that the solution is unique.

2							4	
	7		5					
				1		9		
6		4			2			
	8						5	
			9			3		7
		1		4				
					3		8	
	5							6

A Sudoku puzzle. **Don't try to solve this during the exam!**

2. (Spring 2010) Let T be a rooted tree with integer weights on its edges, which could be positive, negative, or zero. The weight of a path in T is the sum of the weights of its edges. Describe and analyze an algorithm to compute the minimum weight of any path from a node in T down to one of its descendants. It is not necessary to compute the actual minimum-weight path; just its weight. For example, given the tree shown below, your algorithm should return the number -12.

3. (Fall 2006) Suppose you are given an array $A[1..n]$ of n distinct integers, sorted in increasing order. Describe and analyze an algorithm to determine whether there is an index i such that $A[i] = i$, in $o(n)$ time. [Hint: Yes, that's little-oh of n. What can you say about the sequence $A[i] - i$?]



The minimum-weight downward path in this tree has weight −12.

4. (Spring 2010 and Spring 2004) Describe and analyze efficient algorithms to solve the following problems:

   (a) Given a set of n integers, does it contain two elements a, b such that $a + b = 0$?

   (b) Given a set of n integers, does it contain three elements a, b, c such that $a + b = c$?