CSE 202 Homework 2
Combinatorial Search and Optimization:
Back-tracking and Dynamic Programming
Due Thursday, May 6

1. **Hamiltonian path** The s, t Hamiltonian path problem is: given an undirected graph $G = (V, E)$, is there a simple (i.e., non- repeating) path that starts at s, goes through each node $y \in V$ and ends at t? Give a back-tracking algorithm for this problem. Give an upper bound on the time for your algorithm on graphs where all nodes have degree at most 3.

2. **Number of rectangular prisms of given diagonal** . Consider a *rectangular prism* of dimension d, with sides $s_1, ..s_d$. Then the diagonal will have length D so that $D^2 = \sum_i s_i^2$. Give an algorithm which, given integers d and m, returns the number of prisms with *positive integer-valued* sides $s_1, ..s_d$ and with diagonal D so that $D^2 = m$ Your algorithm should be poly-time in the values m and d. (7 points correct poly-time algorithm, 3 pts efficiency. My best time is $O(md)$.)

3. **Library storage** Consider the following problem. A library has n books that must be stored in alphabetical order on adjustable height shelves. Each book has a height and a thickness. The width of each shelf is an input W, so the sum of the thicknesses of books on a single shelf must be at most W. The next shelf will be placed on top, at a height equal to the maximum height of a book in the shelf. Give an algorithm that minimizes the total height of shelves used to store all the books. You are given the list of books in alphabetical order, $b_i = (h_i.t_i)$, where h_i is the height and t_i is the thickness.

4. **Viterbi** Problem 15-5 from text (p. 367)

5. **Dynamic Programming vs. Memoization** Implement the bottom-up dynamic programming algorithm for the length of the longest common subsequence (page 353), and implement a top-down, memoized version of the same recurrence. For a range of sizes n (say, $n = 2^4$ to $n = 2^15$ or as high as time permits), compare the performance of these two algorithms on inputs consisting of two random strings of length n over an alphabet of size 4. (You can present times with a log-log chart. Use all times, not just cycles.) Which approach is better and why? Is there a cross-over point?

When describing an algorithm, don't write out an entire pseudo-code; just describe it at a high level. Be sure to specify completely all data structures used in the algorithm. Include time analysis for all algorithms. Some problems will refer to or be easier if you know material from the book not covered in class. Read the preceding sections.