Computer Science 3711 (Fall 2005):
Assignment #3, Questions #2–4
Due: 4:00 PM on Wednesday, October 26, 2005

2. (9 marks)

a) (6 marks) Give the recurrence underlying the operation of your dynamic programming algorithm for Question #1.

b) (3 marks) Derive the worst-case asymptotic, *i.e.*, big-Oh, running time of your algorithm for Question #1.

3. (26 marks)

a) (6 marks) Determine the longest common subsequence (LCS) of the strings GAAGCCTA and TATCGA using the algorithm given on page 353 of the textbook. In the style of Figure 15.6, show the filled-in dynamic programming matrix, all matrix-cell backpointers, the backpointer path that gives an optimal LCS, and the LCS associated with that path.

b) (8 marks) Determine the optimal 0/1 knapsack-load for the set of items $U = \{1, 2, 3, 4, 5, 6\}$ and $B = 6$ using the algorithm given in Lecture #11 of the class notes. The sizes and values of the items in U are as follows:

i	$s(i)$	$v(i)$
1	4	5
2	2	4
3	3	4
4	5	4
5	2	1
6	1	3

Show the filled-in dynamic programming value and backpointer matrices, the "backpointer path" in the backpointer matrix that gives an optimal knapsack-load, and the knapsack-load associated with that path.

c) (12 marks) Determine the optimal parenthesization of a matrix-chain product whose sequence of dimensions is $\langle 2, 5, 3, 4, 2, 3, 5\rangle$ using the algorithm given on page 336 of the textbook. In the style of Figure 15.3, show the filled-in dynamic programming matrices m and s , the "backpointer path" in s that gives an optimal parenthesization, and the parenthesization associated with that path.

4. (20 marks) For each of the problems below, give a pseudocode algorithm and a parameterized asymptotic worst-case time complexity for that algorithm. Note that these algorithms must run in parameterized polynomial time, *i.e.*, all terms excluding the variables denoting the time complexities of used operations must be polynomials in the input size.

a) (10 marks) Given a connected undirected graph $G = (V, E)$ and three non-overlapping vertex-subsets $S, I, F \subset V$, a **transit path** in G from $x \in S$ to $y \in F$ is a path in G from x to y that passes through at least one vertex in I. Given G, S, I, F, x, and y, compute the length of the shortest transit path (in terms of number of edges in the path) from x to y in G. You may use the following operations:

- `SP(G, x, y)`: Returns the length of the shortest path in G between x and y.
- `size(X)`: Returns number of vertices in vertex-set X.
- `getVertex(X, i)`: Returns the ith vertex in vertex-set X, where $1 \leq i \leq size(X)$.

b) (10 marks) Given an item-set I and a set S of subsets of I, compute the size of a minimum-size set cover for I. You may use the following operation:

- `SCDec(S, I, k)`: Returns `true` if there is set-cover of size $\leq k$ for I in S and `false` otherwise.