Assignment #3 CS4/531, Fall 2013

Due Date: Monday, Oct. 21, 2013

- UNSUPPORTED SOLUTIONS RECEIVE NO CREDIT.
- Total points: 30

  Note: The problems marked 0 points **will not be collected nor graded**. However, it is **very important** for you to do these problems as if they were to be graded. (Because similar problems **will** appear in exams). Detailed solutions to **all** problems (including 0 point problems) will be provided.

- Guideline for all homework assignments
  - **HW must be handed-in by $\boxed{\textbf{9:05:00 am, Oct 21.}}$**
    (Don't bother to turn it in after 9:05:01.)
  - **Print your name, and UB number on the first page**.
  - Staple all sheets together. Do not use notebooks and folders.
  - The solutions can be hand-written, but must be legible.
  - Present the solutions in sequential order. Make sure the problem number is **clearly marked**, and leave space between problems.
  - If you do not follow these guidelines, TA may deduct up to 20% of points.

1. (0 pts) The following is a heuristic strategy for solving the Matrix Chain Product problem. Let $p_0, p_1, \dots, p_n$ be the input integers. Let $p_i = \max\{p_j \mid 1 \leq j \leq n-1\}$. We first multiply the two matrices A_i and A_{i+1} (p_i is the column dimension of A_i and the row dimension of A_{i+1}). After this, we get a sequence of $n-1$ matrices $A_1, A_2, \dots, A_{i-1}, A_i \times A_{i+1}, A_{i+2}, \dots, A_n$, with dimension sequence $p_0, p_1, \dots p_{i-1}, p_{i+1}, \dots, p_n$. We repeat above steps until the chain product is obtained. The idea of the heuristic is that after $A_i \times A_{i+1}$ is computed, the dimension p_i (which is the largest dimension shared by any two matrices) disappears, and hopefully, the total cost would be minimum.

   This strategy does not work. Find a counter-example of this strategy.

2. (3 pts) Find the optimal parenthesization of a matrix chain product whose sequence of dimensions is $< 5, 10, 4, 13, 4, 60, 8 >$. (Namely, the dim of the matrix A_1 is 5×10, the dim of A_2 is $10 \times 4, \dots$, the dim of A_6 is 60×8).

3. (8+3= 11 pts) **Editing Distance** Problem.

   Let $X[1..m]$ and $Y[1..n]$ be two character strings. We want to *edit* the string X to the string Y. The operations we can use are (x is any character in the alphabet set):

- Insert(i, x): insert the character x after $X[i]$ in X.
- Delete(i): delete the character $X[i]$ in X.
- Replace(i, x): replace the character $X[i]$ in X by the character x.

We have cost(Insert), cost(Delete), and cost(Replace) defined for these operations.

An *edit sequence* for X and Y is a sequence S of above operations that changes X into Y. Define cost(S) to be the sum of the costs of the operations in S. Note that there is at least one edit sequence: delete all characters in X one-by-one; then insert the characters in Y one-by-one. In general, there are exponentially many editing sequences for X and Y.

The *editing distance* between X and Y is defined to be:

$$\text{dist}(X,Y) = \min \{ \text{ cost}(S) \mid S \text{ is an editing sequence for } X \text{ and } Y.\}$$

(1) Describe a dynamic programming algorithm for computing dist(X, Y). The run time of the algorithm should be $O(nm)$.

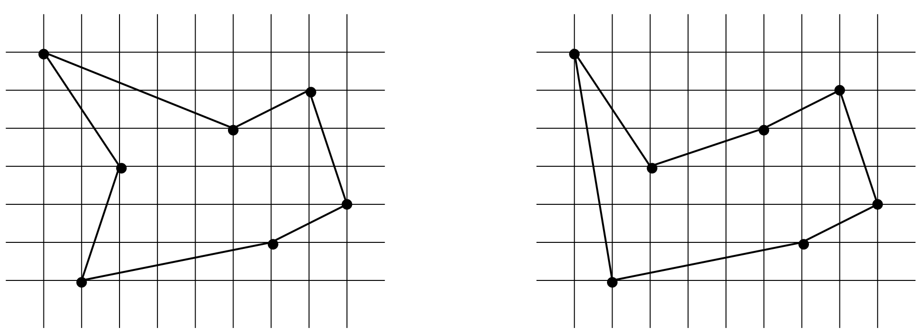
(2) Let cost(Insert)=cost(Delete)= 2 and cost(Replace)=3. Let X =“algorithm” and Y = “altruistic”. Compute dist(X, Y) by using the algorithm you designed in (1).

Note 1: Suppose that X represents an address. It may contains errors in it (such as misspelling, omission of letters, insertion of extra letters). We want to search a address database to find an address Y that “best matches” X. Then, $dist(X, Y)$ is the most logical definition of “closeness” of X and Y.

4. (8 pts) Yuckdonald’s is considering to open a series of restaurants along Great Valley Highway (GVH). The n possible locations are along the GVH, and the distances of these locations from the start of GVH are, in miles and increasing order, $m_1, m_2, \ldots, m_n$. The constraints are as follows:

- At each location, Yuckdonald’s may open at most one restaurant. The expected profit from open a restaurant at location m_i is $p_i > 0$, for $i = 1, \ldots n$.
- Any two restaurants should be at least k miles apart, where k is a positive integer.

Find an efficient dynamic programming style algorithm to compute the maximum expected total profit subject to the above constraints.



(1) (2)

Figure 1: (1) The shortest closed tour, with length ≈ 24.89. This tour is not bitonic. (2) The shortest closed bitonic tour, with length ≈ 25.58.

5. (8 pts) The **Euclidean Traveling Salesman Problem** (ETSP) is the problem of determining the shortest closed tour that connects a given set of n points in the 2D plane. Fig 1 (1) shows a solution to a 7-point problem. This problem, in its general form, is NP-complete, (meaning that it is highly unlikely solvable in polynomial time.

We consider a restricted form of this problem: We only consider the **bitonic tours**, that is, tours start at the leftmost point, go strictly left to right to the rightmost point, and then go strictly right to left back to the starting point. Fig 1 (2) shows the shortest bitonic tour of the same 7 points. In this case, a polynomial time algorithm is possible.

Describe an $O(n^2)$ time dynamic programming style algorithm for solving the bitonic ETSP. You may assume that no two points have the same x-coordinates.

Hint: Scan left to right, maintaining optimal possibilities for the two parts of the tour.