These problems will be solved in precept.

1. **Bitonic TSP.** (CLRS 15-3) Given \(n\) points in the plane \(p_1, p_2, \ldots, p_n\), the *Euclidean traveling salesperson* problem is to find a shortest closed tour that connects all \(n\) points, where the distance between two points is the Euclidean distance between them. The general problem is NP-hard, so we will restrict our attention to *bitonic* tours—tours that start at the rightmost point, go strictly leftward to the leftmost point, and then go strictly rightward back to the starting point. Design a dynamic-programming algorithm that finds an bitonic tour of minimum length. Your algorithm should take \(O(n^2)\) time and use \(O(n^2)\) space.

Assume that no two points have the same \(x\)-coordinate and that all operations on real numbers (such as computing the Euclidean distance between two points) take unit time.

![A bitonic TSP tour of 33 points in the plane.](image)

2. **Local sequence alignment.** Often two DNA sequences are significantly different, but contain regions that are very similar and are highly conserved. Design an algorithm that takes two strings \(x = x_1 x_2 \ldots x_m\) and \(y = y_1 y_2 \ldots y_n\) as input and finds a min-cost alignment between any substring of \(x\) with any substring of \(y\). As usual, let \(\delta\) denote the gap cost and \(\alpha_{pq}\) denote the cost of aligning character \(p\) with character \(q\).

For example, consider the two strings

\[
x = TTAAAACCTTTGTTTTT, \quad y = CCCAAAAATTGGCCCC
\]

and suppose the gap cost is 2, the mismatch cost for aligning two different characters is 1, and match cost is \(-1\). Then, the minimum cost of a local alignment of \(x\) and \(y\) is \(5 = 8 - 1 - 2\) (align \(AAAACCTTTG\) with \(AAAAATTGG\)).

(a) Design an algorithm that takes \(O(m^3 n^3)\) time by solving many global sequence alignment problems.

(b) Design a dynamic programming algorithm that takes \(O(mn)\) time.