Sample Solution

Assignment 8

CSCI 3110 — Summer 2014

(a) Observe that one of the neighbours of $p_n$ must be $p_{n-1}$. Indeed, if $p_n$ had neighbours $p_i$ and $p_j$ with $i < n - 1$ and $j < n - 1$, the tour would have to visit $p_{n-1}$ on its way from $p_i$ to $p_1$ or on its way from $p_1$ to $p_j$, but then one of these two paths wouldn’t be $x$-monotone.

Removing the edge from $p_{n-1}$ to $p_n$ gives us a tour from $p_{n-1}$ to $p_n$ that visits all $n$ points, and this tour must of course be optimal, that is, its length is $L(n-1,n)$. This gives us the equation

$$L^* = L(n-1,n) + \|p_{n-1} - p_n\|.$$

(b) For a tour that starts at $p_i$, ends at $p_j$, and visits every point in $\{p_1, p_2, \ldots, p_j\}$ exactly once, observe that this tour consists of the edge $p_1 p_2$ if $(i,j) = (1,2)$. Thus, its length is $L(1,2) = \|p_1 - p_2\|$. If $j > 2$, we distinguish two cases:

If $j > i + 1$, then the neighbour of $p_j$ on this tour must be $p_{j-1}$ for the same reason why $p_{n-1}$ is the neighbour of $p_n$ in an optimal $x$-monotone tour that visits all points. Thus, we have

$$L(i,j) = L(i,j-1) + \|p_{i-1} - p_j\| \quad \text{if } j > i + 1.$$

If $j = i + 1$, then the neighbour of $p_j$ must be a point $p_h$ with $h < i$. We do not know which one it is, but whichever it is, the part of the tour from $p_h$ to $p_i$ must be an optimal tour that visits all points in $\{p_1, p_2, \ldots, p_i\}$. Thus, we have

$$L(j-1,j) = \min_{1 \leq h < j-1} (L(h,j-1) + \|p_h - p_j\|).$$

(c) The following is the algorithm, which takes the input points in an array.
MonotoneTravelingSalesman(P)

1 Sort the points in P by their x-coordinates
2 \( L[1,2] = \|P[1] - P[2]\| \)
3 for \( j = 3 \) to \( n \)
4 for \( i = 1 \) to \( j - 2 \)
5 \( L[i,j] = L[i,j-1] + \|P[j-1] - P[j]\| \)
6 \( L[j-1,j] = \infty \)
7 for \( h = 1 \) to \( j - 2 \)
8 \( L = L[h,j-1] + \|P[h] - P[j]\| \)
9 if \( L < L[j-1,j] \)
10 \( L[j-1,j] = L \)
11 \( L^* = L[n-1,n] + \|P[n-1] - P[n]\| \)
12 return \( L^* \)

The running time of this algorithm is clearly \( O(n^2) \) because line 1 takes \( O(n \lg n) \) time, lines 2, 11, and 12 take constant time, and lines 3–10 consist of an outer loop with \( n - 2 \) iterations with two inner loops with at most \( n - 2 \) iterations nested inside it. Its correctness follows because it simply evaluates the recurrence from part (b).

(d) To be able to construct an optimal tour, we only need to remember the optimal predecessor of \( P[j] \) in an optimal tour from \( P[j-1] \) to \( P[j] \). The following algorithm constructs a table \( Q[1..n] \) with that stores this information:

MonotoneTravelingSalesman(P)

1 Sort the points in P by their x-coordinates
2 \( L[1,2] = \|P[1] - P[2]\| \)
3 for \( j = 3 \) to \( n \)
4 for \( i = 1 \) to \( j - 2 \)
5 \( L[i,j] = L[i,j-1] + \|P[j-1] - P[j]\| \)
6 \( L[j-1,j] = \infty \)
7 for \( h = 1 \) to \( j - 2 \)
8 \( L = L[h,j-1] + \|P[h] - P[j]\| \)
9 if \( L < L[j-1,j] \)
10 \( L[j-1,j] = L \)
11 \( Q[j] = h \)
12 \( L^* = L[n-1,n] + \|P[n-1] - P[n]\| \)
13 return \( L^* \)

(e) Using the table \( Q \) constructed by the previous algorithm, we can construct the optimal tour in linear time as follows. There are many possible ways to represent this tour. For simplicity, we output its edges in no particular order here:
**BuildTour(P,Q)**

1. $T = \emptyset$ // The edge set of the tour
2. Add edge $(P[n-1], P[n])$ to $T$
3. Add edge $(P[1], P[2])$ to $T$
4. $i = n - 1$
5. for $j = n$ downto 3
6. if $i == j - 1$
7. $h = Q[j]$
8. Add edge $(P[h], P[j])$ to $T$
9. $i = h$
10. else Add edge $(P[j - 1], P[j])$ to $T$
11. return $T$