We describe here for our reference an algorithm we discussed in class for the following problem: Given a set \( P \) of \( n \) points, find, for each point \( q \in P \), its closest point in \( P \setminus \{q\} \). The following algorithm assumes that \( P = \{p_1, \ldots, p_n\} \) is input in increasing order of \( x \)-coordinate. It is moderately clever in avoiding inspecting certain pairs of points. Let \( d(p, q) \) denote the Euclidean distance between points \( p \) and \( q \). The algorithm stores the the nearest point to \( p_i \) in nearest\([i]\).

**Algorithm 1** All-Nearest\((P)\)

1: for all \( i \in \{1, \ldots, n - 1\} \) do
2: nearest\([i]\) \( \leftarrow p_{i+1} \)
3: for all \( j \in \{i + 2, \ldots, n\} \) do
4: if \( p_j.x - p_i.x > d(p_i, \text{nearest}[i]) \), break;
5: if \( d(p_i, p_j) < d(p_i, \text{nearest}[i]) \), nearest\([i]\) \( \leftarrow p_j \).
6: for all \( j \in \{i - 1, \ldots, 1\} \) do
7: if \( p_i.x - p_j.x > d(p_i, \text{nearest}[i]) \), break;
8: if \( d(p_i, p_j) < d(p_i, \text{nearest}[i]) \), nearest\([i]\) \( \leftarrow p_j \).
9: (For brevity, the computation of nearest\([n]\) is omitted.)