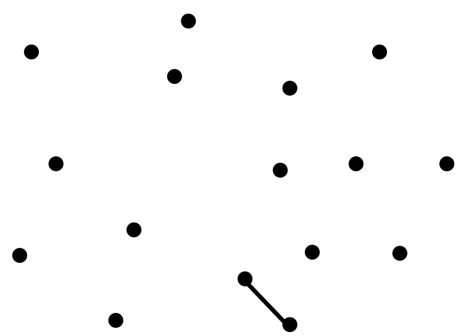
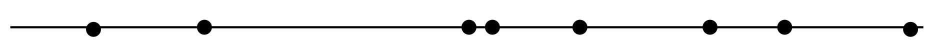
Nearest Neighbor Problem

- **Given:** n points in the plane.
- **Find:** closest pair.



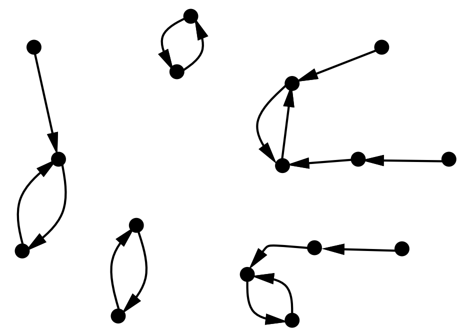
- Trivial algorithm $O(n^2)$
- Can it be improved?
- Yes, in 1 dimension.



- Sort. Closest pair is next to each other.
- Sorting $O(n \log n)$. Scanning $O(n)$ time.

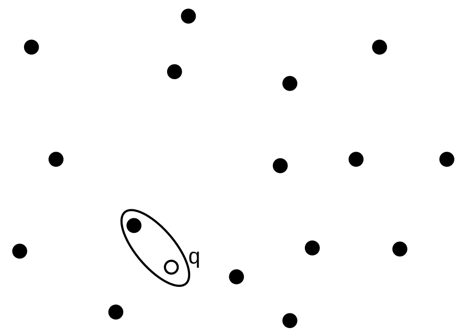
All Nearest Neighbor Problem

- **Given:** n points in the plane.
- **Find:** nearest neighbor for each.



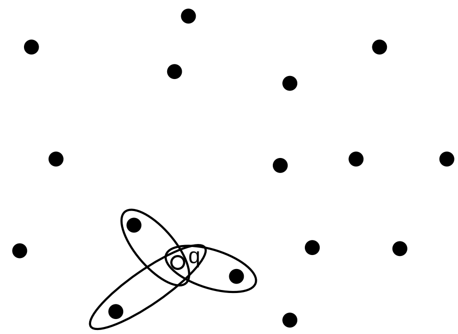
Nearest Neighbor Problem - Search

- **Given:** n points in the plane and a query point q.
- **Find:** nearest neighbor to q.



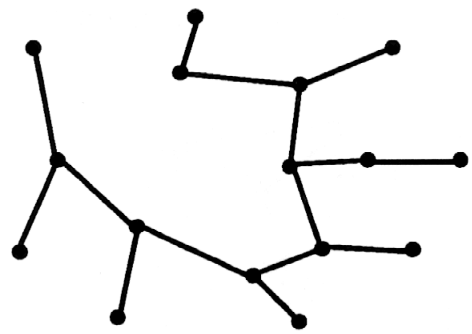
k-Nearest Neighbor Problem - Search

- **Given:** n points in the plane and a query point q.
- **Find:** k-th nearest neighbor to q.

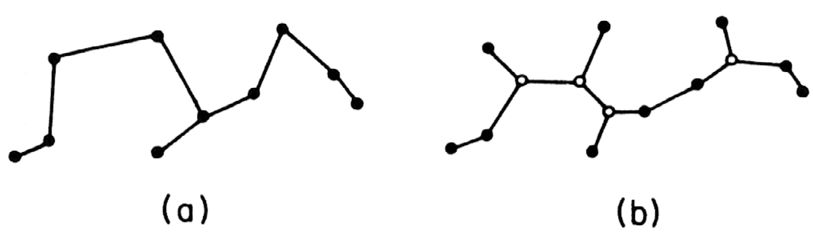


Minimum Spanning Tree Problem

- **Given:** n points in the plane.
- **Find:** minimum spanning tree



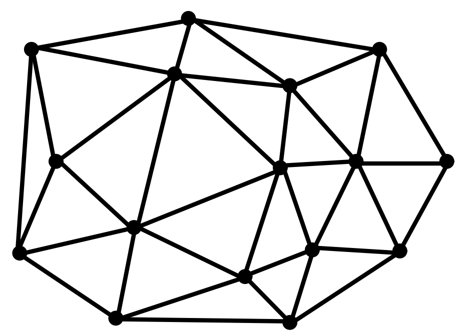
- Can be solved by well-known methods for minimum spanning trees in weighted graphs (in the complete graph K_n with distances as edge weights).
- Is it possible to prune K_n?
- Only pairs relatively close to each other need to be considered.



A Steiner Tree (b) may have smaller total length than the MST (a).

Delaunay Triangulation

- **Given:** n points in the plane.
- **Add:** non-crossing edges so that all faces are triangular. The exterior face is the convex hull of the point set.

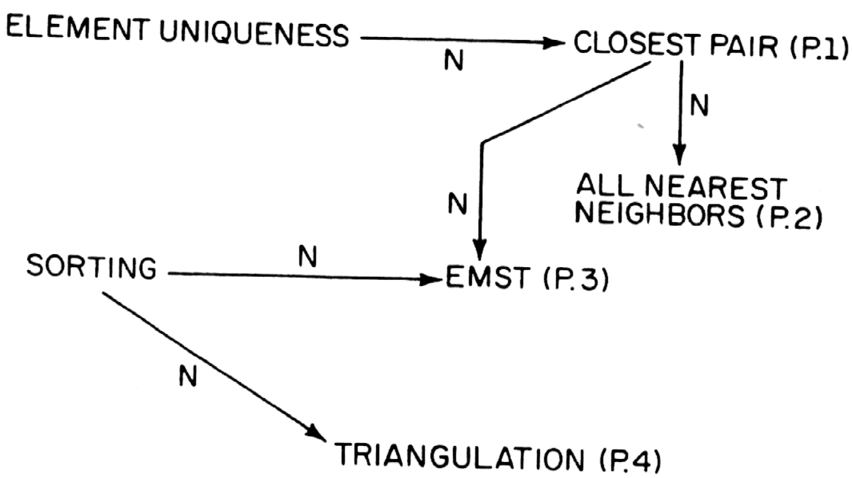


- Every triangulation has $3n - 6$ edges.
- There are many different triangulations:
  - minimum weight triangulation,
  - maximized smallest angle.

Lower Bounds

- Nearest neighbor problem is a generalization of the element uniqueness problem.
  - **Given:** n real numbers.
  - **Decide:** if two are identical.
- Transformation: $x \rightarrow (x, 0)$. Find two closest neighbors. If their distance is 0, then the set contains to identical numbers.
- Element uniqueness requires $\Omega(n \log n)$.

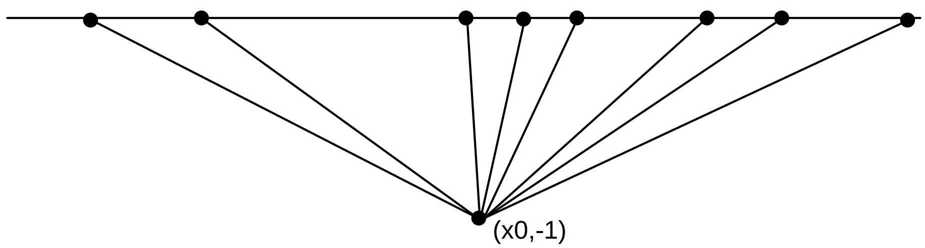
- All nearest neighbor problem is also $\Omega(n \log n)$ time. Its solution provides the solution to the nearest neighbor problem in additional $O(n)$ time.



Relationship among computational prototypes and proximity problems.

Lower Bounds

- Minimum spanning tree problem is $\Omega(n \log n)$; it is a generalization of sorting of n numbers.
  - Transformation $x \rightarrow (x, 0)$. Minimum spanning tree for this point-set is a path (defining the ordering).
- Triangulation is a generalization of sorting.



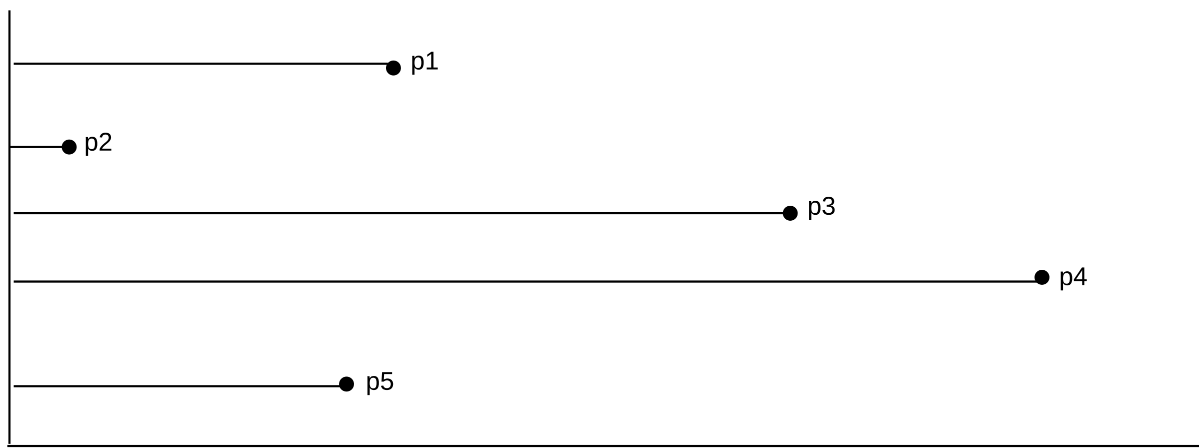
- edges incident with $(x_0, -1)$ give the ordering.

Lower Bounds

- Nearest neighbor search is a generalization of binary search. Transformation: $x \rightarrow (x, 0)$. Search for the nearest neighbor to $(x_0, 0)$. Binary search is $\Omega(\log n)$.
- k-nearest neighbor search is obviously a generalization of nearest neighbor search. Hence $\Omega(\log n)$

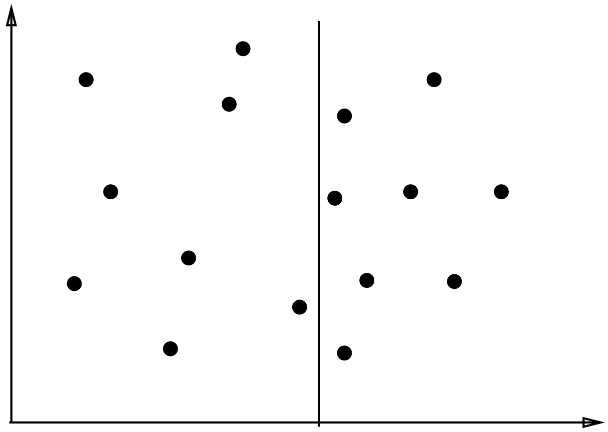
Nearest Neighbor Problem

- Sorting provides an optimal $\Theta(n \log n)$ algorithm in 1-dimensional space.
- Can this be generalized to higher dimensions?
- Project onto one of the axes and then sort.



- Does not work. p_1 and p_5 are nearest neighbors but their projections are farthest away on the y-axis.

Nearest Neighbor Problem - Divide and Conquer



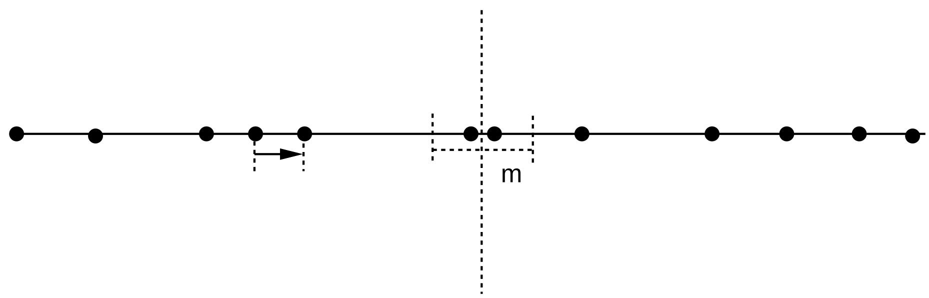
- Nearest neighbors in S_1.
- Nearest neighbors in S_2.
- Nearest neighbors, one in S_1 other in S_2.
- Time complexity:

$$T(n) = 2T(n/2) + O(n^2/4)$$

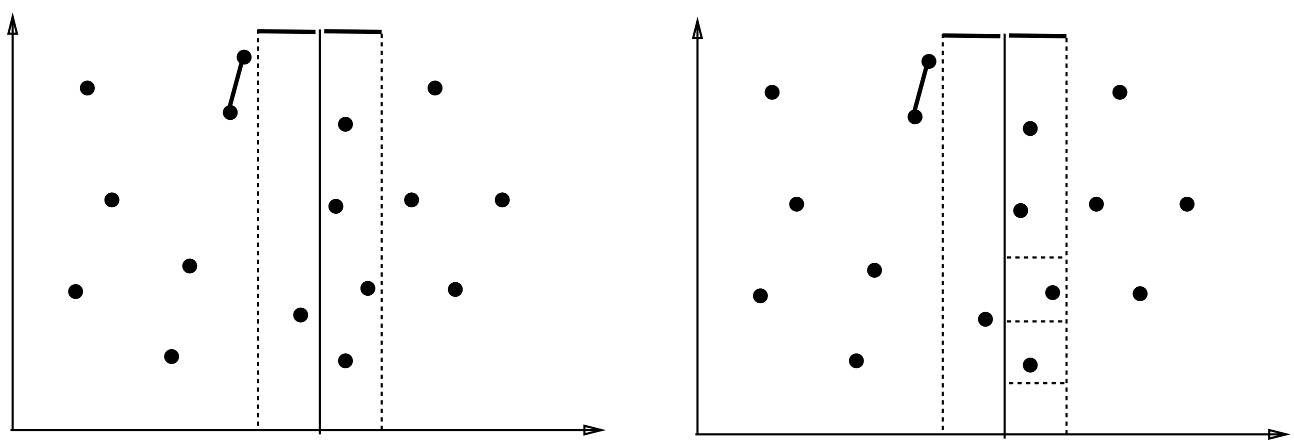
is $O(n^2)$

Nearest Neighbor Problem - Divide and Conquer

- Is it necessary to check all $n^2/4$ pairs with one point in S_1 and the other point in S_2?
- In 1-dimensional space.

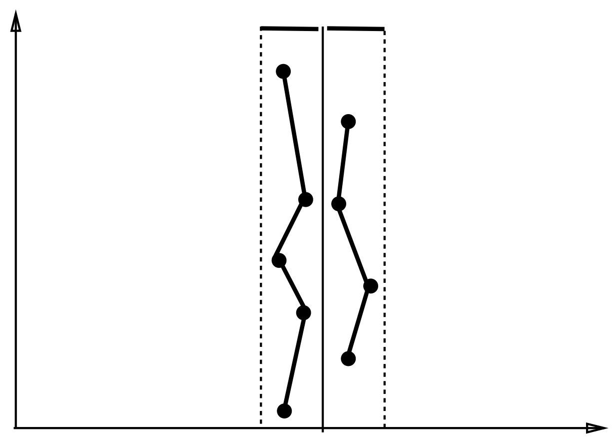


- Let $\sigma = \min\{|p_i p_j|, |q_k q_l|\}$
- Only points at distance σ need to be checked.
- There is at most one point in S_1 at distance σ from m. Similarly for S_2.
- In 2-dimensional space.



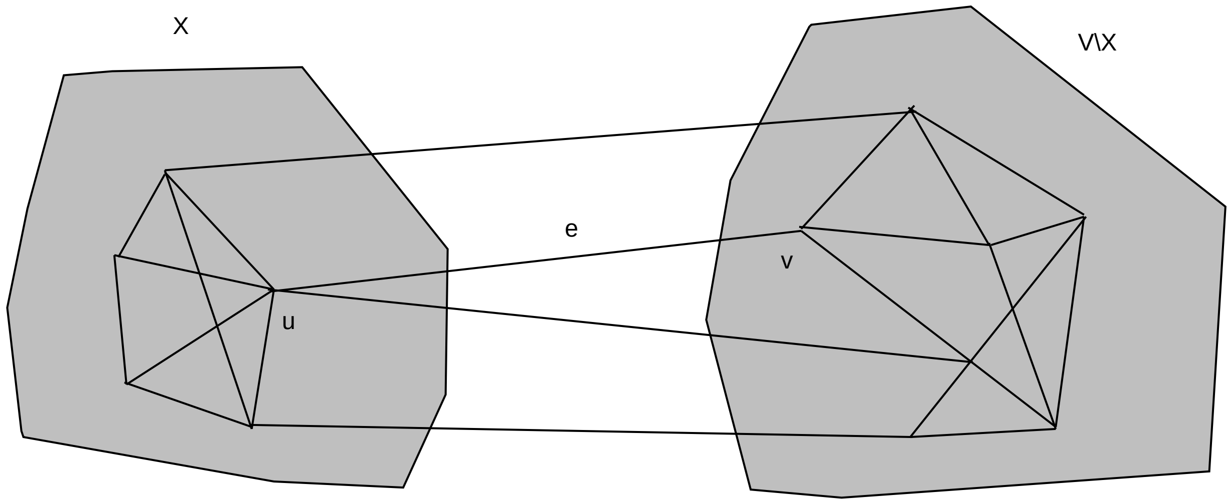
Nearest Neighbor Problem - Divide and Conquer

- Preprocessing: Sort S by y-coordinates.
- Divide S into two equal size subsets S_1 and S_2 by a vertical median.
- Solve (recursively) for S_1 and S_2. Let $\delta = \min\{\delta_1, \delta_2\}$ where δ_i is the smallest distance in S_i, $i = 1, 2$.
- Determine the upward chain P_i through points of S_i at distance δ from the median. Can be done in $O(n)$ time.

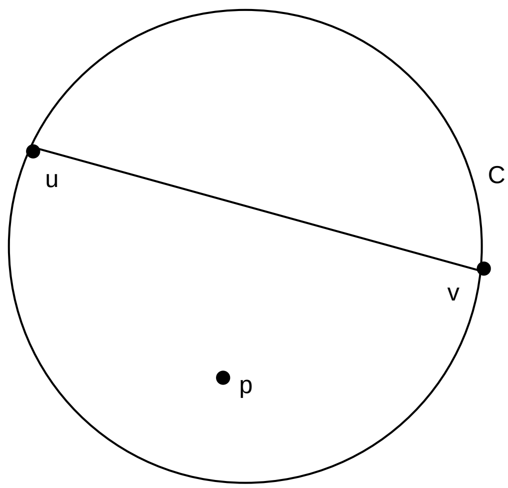


- In total $\Theta(n \log n)$.
- This method cannot be generalized to solve other problems.

Minimum Spanning Tree



- e is shortest crossing edge,
- e is not in DT



- p is either in X or in $V - X$
- $|up| < |uv|$ and $|vp| < |uv|$
- uv cannot be a crossing edge.