Rank

Rank of an element is its position in ascending key order.

[2,6,7,8,10,15,18,20,25,30,35,40]

rank(2) = 0

rank(15) = 5

rank(20) = 7

Selection Problem

- Given n unsorted elements, determine the k'th smallest element. That is, determine the element whose rank is k-1.
- Applications
  - Median score on a test.
    - k = ceil(n/2).
  - Median salary of Computer Scientists.
  - Identify people whose salary is in the bottom 10%. First find salary at the 10% rank.

Selection By Sorting

- Sort the n elements.
- Pick up the element with desired rank.
- O(n log n) time.

Divide-And-Conquer Selection

- Small instance has n <= 1. Selection is easy.
- When n > 1, select a pivot element from out of the n elements.
- Partition the n elements into 3 groups left, middle and right as is done in quick sort.
- The rank of the pivot is the location of the pivot following the partitioning.
- If k-1 = rank(pivot), pivot is the desired element.
- If k-1 < rank(pivot), determine the k'th smallest element in left.
- If k-1 > rank(pivot), determine the (k-rank(pivot)-1)'th smallest element in right.

D&C Selection Example

Find kth element of:

a

3	2	8	0	11	10	1	2	9	7	1

Use 3 as the pivot and partition.

a

1	2	1	0	2	3	10	11	9	7	8

rank(pivot) = 5. So pivot is the 6'th smallest element.

D&C Selection Example

a

1	2	1	0	2	3	10	11	9	7	8

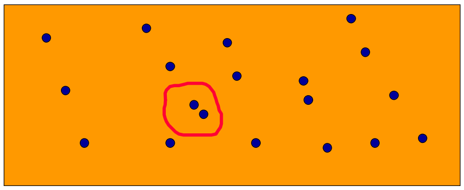
- If k = 6 (k-1 = rank(pivot)), pivot is the element we seek.
- If k < 6 (k-1 < rank(pivot)), find k'th smallest element in left partition.
- If k > 6 (k-1 > rank(pivot)), find (k-rank(pivot)-1)'th smallest element in right partition.

Time Complexity

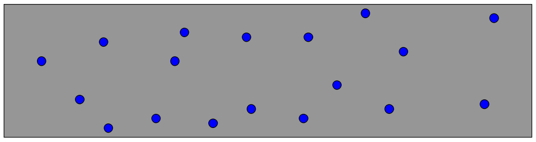
- Worst case arises when the partition to be searched always has all but the pivot.
  - $O(n^2)$
- Expected performance is $O(n)$.
- Worst case becomes $O(n)$ when the pivot is chosen carefully.
  - Partition into $n/9$ groups with 9 elements each (last group may have a few more)
  - Find the median element in each group.
  - pivot is the median of the group medians.
  - This median is found using select recursively.

Closest Pair Of Points

- Given n points in 2D, find the pair that are closest.



Applications



- We plan to drill holes in a metal sheet.
- If the holes are too close, the sheet will tear during drilling.
- Verify that no two holes are closer than a threshold distance (e.g., holes are at least 1 inch apart).

Air Traffic Control

- 3D -- Locations of airplanes flying in the neighborhood of a busy airport are known.
- Want to be sure that no two planes get closer than a given threshold distance.

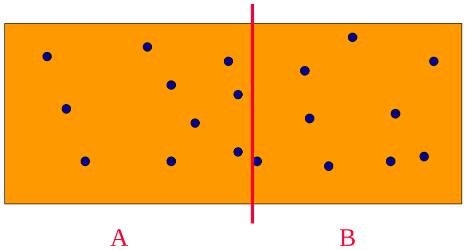
Simple Solution

- For each of the $n(n-1)/2$ pairs of points, determine the distance between the points in the pair.
- Determine the pair with the minimum distance.
- $O(n^2)$ time.

Divide-And-Conquer Solution

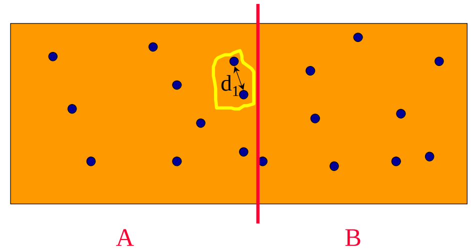
- When n is small, use simple solution.
- When n is large
  - Divide the point set into two roughly equal parts A and B.
  - Determine the closest pair of points in A.
  - Determine the closest pair of points in B.
  - Determine the closest pair of points such that one point is in A and the other in B.
  - From the three closest pairs computed, select the one with least distance.

Example



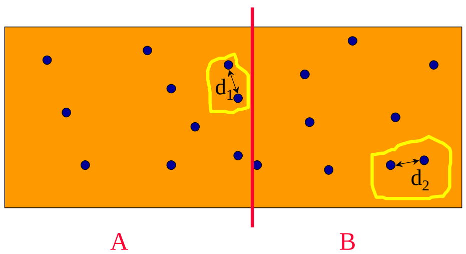
- Divide so that points in A have x-coordinate <= that of points in B.

Example



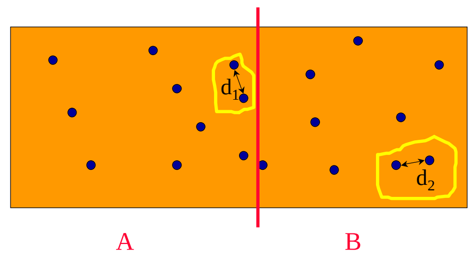
- Find closest pair in A.
- Let d_1 be the distance between the points in this pair.

Example



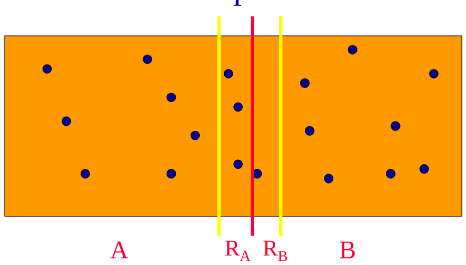
- Find closest pair in B.
- Let d_2 be the distance between the points in this pair.

Example



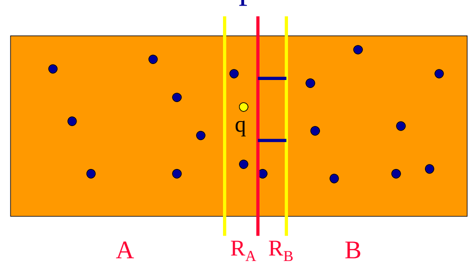
- Let $d = \min\{d_1, d_2\}$.
- Is there a pair with one point in A, the other in B and distance < d?

Example



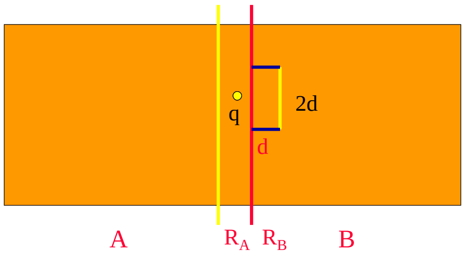
- Candidates lie within d of the dividing line.
- Call these regions R_A and R_B, respectively.

Example



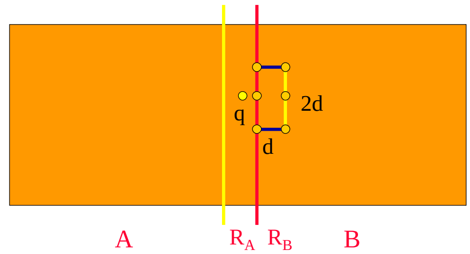
- Let q be a point in R_A.
- q need be paired only with those points in R_B that are within d of q.y.

Example



- Points that are to be paired with q are in a d x 2d rectangle of R_B (comparing region of q).
- Points in this rectangle are at least d apart.

Example



- So the comparing region of q has at most 6 points.
- So number of pairs to check is <= 6| R_A | = O(n).

Time Complexity

- Create a sorted by x-coordinate list of points.
  - O(n log n) time.
- Create a sorted by y-coordinate list of points.
  - O(n log n) time.
- Using these two lists, the required pairs of points from R_A and R_B can be constructed in O(n) time.
- Let n < 4 define a small instance.

Time Complexity

- Let t(n) be the time to find the closest pair (excluding the time to create the two sorted lists).
- t(n) = c, n < 4, where c is a constant.
- When n >= 4,

  t(n) = t(ceil(n/2)) + t(floor(n/2)) + an,

  where a is a constant.
- To solve the recurrence, assume n is a power of 2 and use repeated substitution.
- t(n) = O(n log n).