Divide and Conquer Examples

Counting Inversion

Some web sites try to match your preference (for music, movies, books) with others.
How do you compare two rankings?

e.g.

	best			worst
I like	B	D	C	A
You like	A	D	B	C

Assignment Project Exam Help

Count: how many pairs do we rank differently?

i.e. how many pairs of lines cross (out of 6 pairs)

https://powcoder.com

$\binom{4}{2} = \frac{4 \cdot 3}{2} = 6$

Add WeChat powcoder

BD BA DA CA (the pairs that don't cross are BC DC)

Equivalently

my ranking	1 2 3 4
your ranking	4 2 1 3

and we count # inversions − # pairs out of order in 2nd list.

Brute Force: check all $\binom{n}{2}$ pairs: $O(n^2)$
Does sorting help? Doesn't seem to.

Counting inversions: divide and conquer

Given list $a_1, \ldots, a_n$, count # inversions.

$A = a_1 \ldots a_m \quad B = a_{m+1} \ldots a_n$

- Divide list in two: $m = \lceil \frac{n}{2} \rceil$
- Recursively count # inversions in each half, r_A, r_B
- Combine: answer $\longleftarrow r_A + r_B + r$

Assignment Project Exam Help

r = # inversions with one element in A, one in B

= # pairs a_i, a_j with $a_i \in A$, $a_j \in B$, $a_i > a_j$

https://powcoder.com

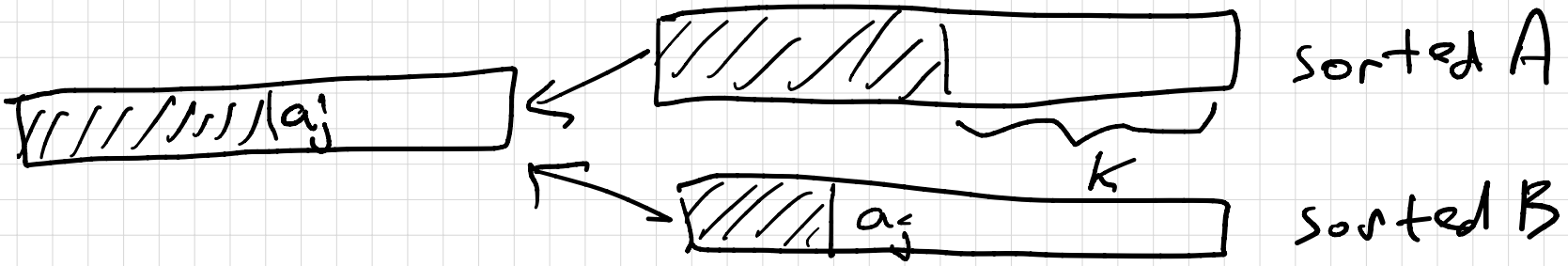
How do we find r?

Add WeChat powcoder

Can we count, for each $a_j \in B$, how many larger elements are there in A? — r_j

Then $r = \sum_{a_j \in B} r_j$

Think about mergesort: sort A, sort B, merge



when a_j is output to merged list: $r_j \leftarrow k$

Whole algorithm:

sort-and-count(L) – returns sorted L, # inverses

- divide L into A, B (first half, second half)
- $(r_A, A) \leftarrow$ sort-and-count(A)
- $(r_B, B) \leftarrow$ sort-and-count(B)
- $r \leftarrow 0$
- Do merge of A and B
  When elements is moved from B to output
  $r \leftarrow r+$ # elements remaining in A
- return $(r_A + r_B + r,$ merged list$)$

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

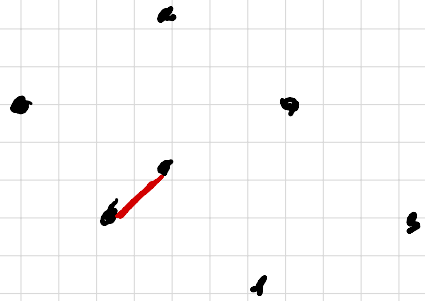
solution: $T(n) = O(n \log n)$ (as in mergesort)

Question: Is there a better algorithm?

- $O(n \log n/(\text{loglog}\, n))$ '89
- $O(n\sqrt{\log n})$ 2010 Timothy Chan et al. using techniques/model where sorting is $o(n \log n)$

Finding the closest pair of point: an example with the "conquer" step more complicated

Problem: Given n points in the plane, find the pair that is closest together.



$$d(p,q) = \text{distance between } p \text{ and } q$$
$$= \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Note: we can compare $d(p,q)$ to $d(r,s)$ without $\sqrt{}$.

Assignment Project Exam Help

Brute force: try all pairs $O(n^2)$

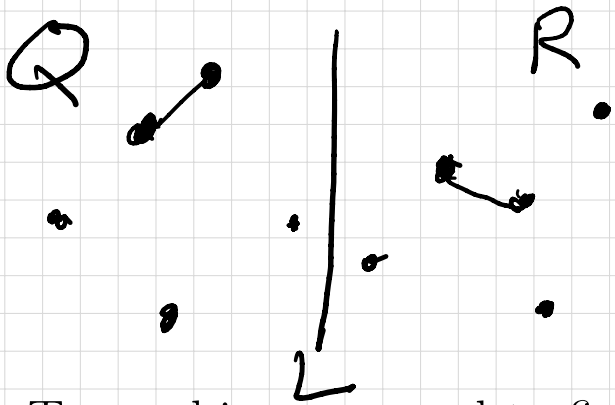
In 1D (points on a line: sort and compare consecutive pairs: $O(n \log n)$

https://powcoder.com

Note that this does not work in $2D$.

Add WeChat powcoder

Divide and Conquer (after sorting by x-coord.)



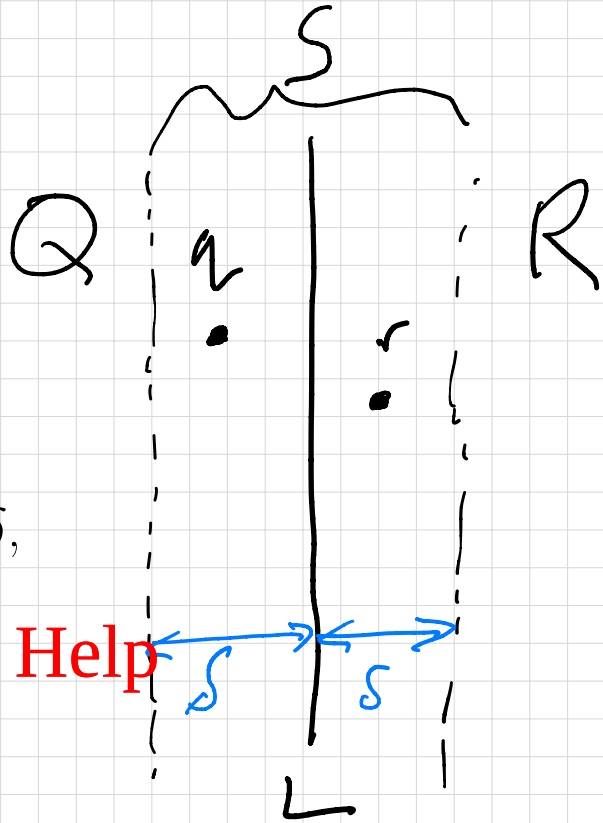
- divide points in left half, Q, right half, R, dividing line L
- recursivley find closest pair in Q, closest pair in R
- combine

To combine, we need to find close pairs crossing L.

This is the tricky part.

Let $\delta = \min$ distance of $\begin{cases} \text{closest pair in } Q \\ \text{closest pair in } R \end{cases}$

Must check pairs $q \in Q$, $r \in R$ with $d(q, r) < \delta$.



Claim: Such points satisfy $d(q, L) < \delta$, $d(r, L) < \delta$

Pf. Otherwise horizontal distance $\geq \delta$, so distance $\geq \delta$

Assignment Project Exam Help

Let S = points in this vertical strip of width 2δ.

https://powcoder.com

We can restrict our search to S (might be all n points!)

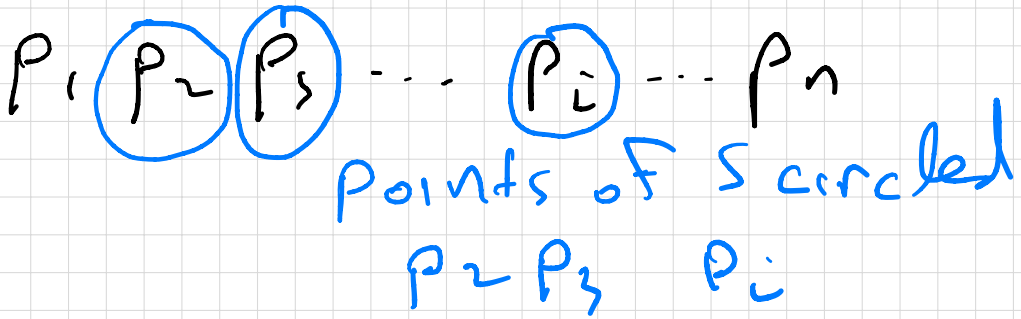
Add WeChat powcoder

This problem seems more 1-dimensional.

Sort by y-coordinate.

Note: Do not do this in each recursive call. Do it once for all points $O(n \log n)$ and extract the sorted sublist for and S in linear time.

This is like "unmerging". E.g., points sorted by y-coordinate



Note: each recursive call needs to know its points sorted by y-coordinate.

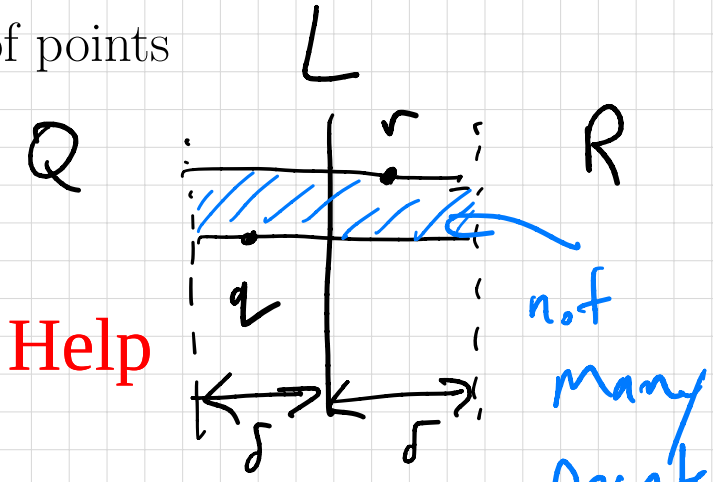
Overall structure of the algorithm:

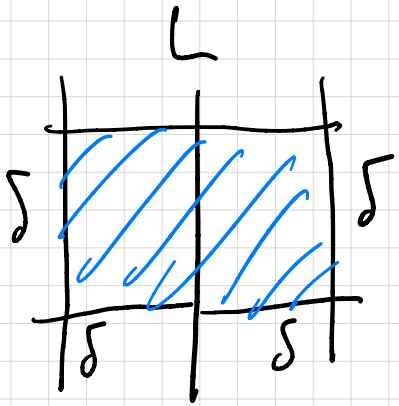
$X \leftarrow$ sort points by x-coord
$Y \leftarrow$ sort points by y-coord
closest(X, Y) — returns distance between closest pair of points

closest(X, Y)

$L \leftarrow$ dividing line (middle of X)
"umerge" to get X_Q, X_R, Y_Q, Y_R
sorted lists for leftside (Q), rightside (R)
$\delta_Q \leftarrow$ closest(X_Q, Y_Q)
$\delta_R \leftarrow$ closest(X_R, Y_R)
$\delta \leftarrow \min\{\delta_Q, \delta_R\}$
find set S as above
$Y_S \leftarrow S$ sorted by y-coord (extract from Y)

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

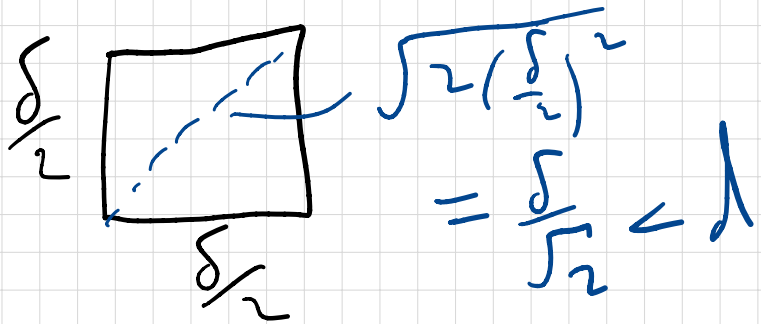




Claim: At most 8 points here

Finally, what do we do with S, Y_S?
Our hope: if $q, r \in S$, $q \in Q$, $r \in R$ and $d(q, r) < \delta$ then q and r are near each other in the sorted S.

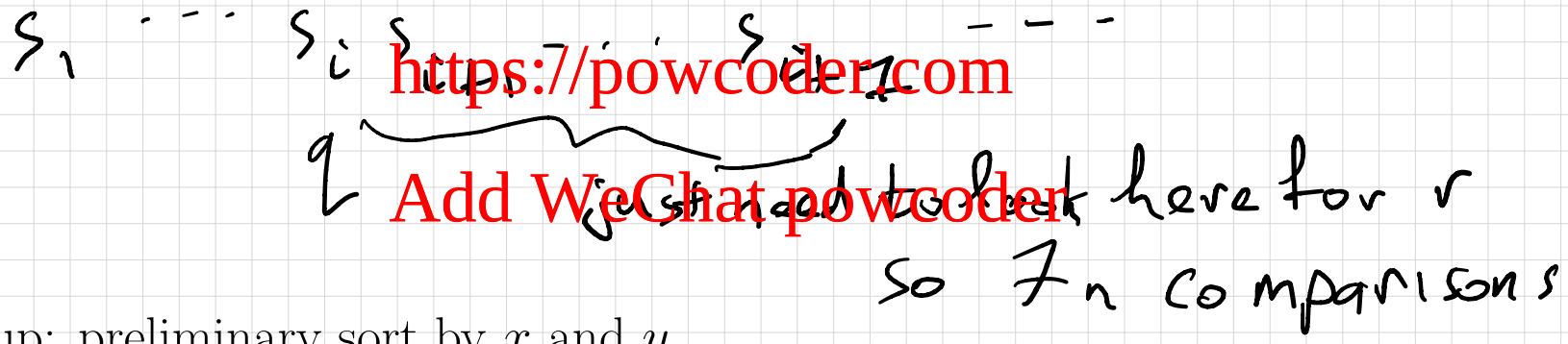
Why? We have 8 squares $\frac{\delta}{2} \times \frac{\delta}{2}$ and each



square has ≤ 1 point
(in fact, can prove 6 instead of 8)

Thus if $q, r \in S \quad q \in Q, r \in R \quad d(q,r) < \delta$
then q and r are at most 8 positions apart in sorted S.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder



Wrapping up: preliminary sort by x and y
then $T(n) = 2T(n/2) + cn$ so $T(n) \in O(n \log n)$

This algo. was due to Preparata & Shamos in the early days of computational geometry (70's and 80's).

In fact, one can do much more in $O(n \log n)$ — find closes neighbours of all points

indicates b is a's closest neighbour