Chapter 1
Divide and Conquer

Algorithm Theory
WS 2013/14

Fabian Kuhn

Number of Inversions

Formal problem:

- **Given**: array $A = [a_1, a_2, a_3, \dots, a_n]$ of distinct elements

- **Objective**: Compute number of inversions I

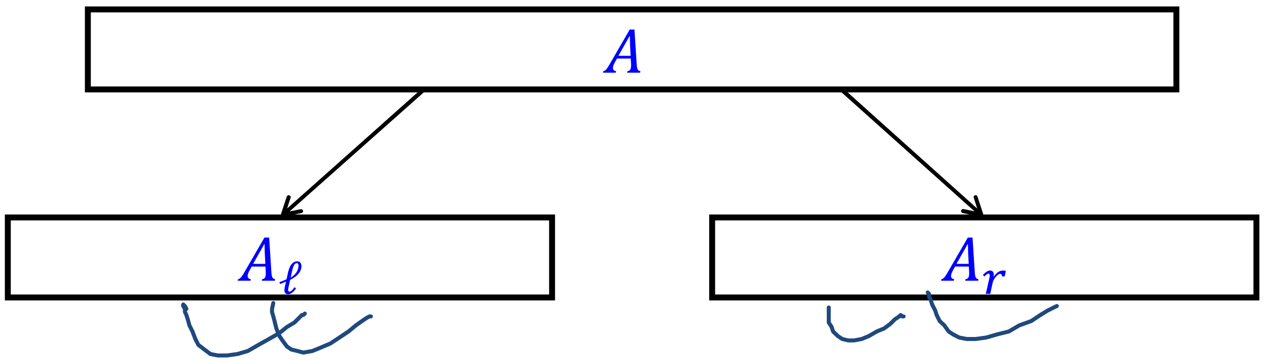
$$I := \left|\{0 \leq i < j \leq n \mid a_i > a_j)\}\right|$$

- **Example**: $A = [\ 4\ ,\ 1\ ,\ 5\ ,\ 2\ ,\ 7\ ,\ 10\ ,\ 6\ ]$

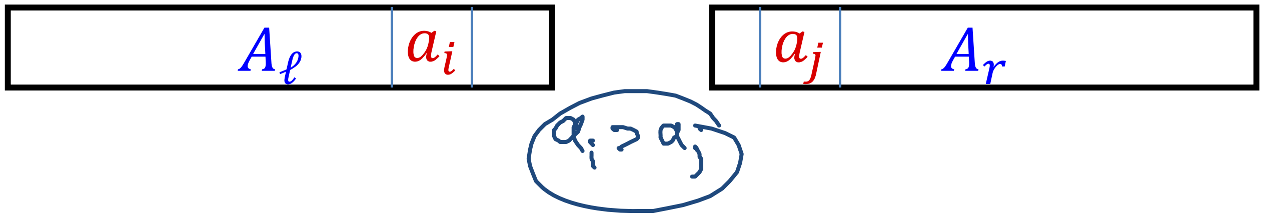
- **Naive solution**:

$O(n^2)$

Divide and conquer

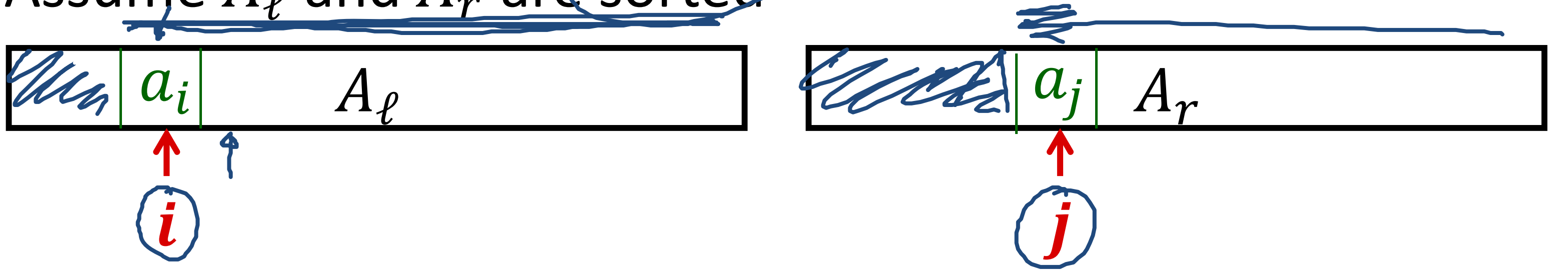


1. Divide array into 2 equal parts A_ℓ and A_r
2. Recursively compute #inversions in A_ℓ and A_r
3. Combine: add #pairs $a_i \in A_\ell$, $a_j \in A_r$ such that $a_i > a_j$



Combine Step

- Assume A_ℓ and A_r are sorted



- Pointers i and j, initially pointing to first elements of A_ℓ and A_r
- If $a_i < a_j$:
  - a_i is smallest among the remaining elements
  - No inversion of a_i and one of the remaining elements
  - Do not change count
- If $a_i > a_j$:
  - a_j is smallest among the remaining elements
  - a_j is smaller than all remaining elements in A_ℓ
  - Add number of remaining elements in A_ℓ to count
- Increment point, pointing to smaller element

$O(n)$ time

Combine Step

- Need sub-sequences in sorted order
- Then, combine step is like merging in merge sort

- **Idea**: Solve sorting and #inversions at the same time!
  1. Partition A into two equal parts A_ℓ and A_r
  2. Recursively compute #inversions and sort A_ℓ and A_r
  3. Merge A_ℓ and A_r to sorted sequence, at the same time, compute number of inversions between elements a_i in A_ℓ and a_j in A_r

$O(n)$

$O(n)$ time

Combine Step: Example

- Assume A_ℓ and A_r are sorted

3	5	8	13	14	18	24	25	30

6	7	9	19	21	23	28	32	33

i j

3	5	6	7	8	9	13	14	18	19	21	23	24	25	28	30	32	33

count: $\underline{0+7+7+6+3+3+3+1}$

Analysis, Guessing

Recurrence relation:

$$T(n) \leq 2 \cdot T(n/2) + c \cdot n, \qquad T(1) \leq c$$

Repeated substitution:

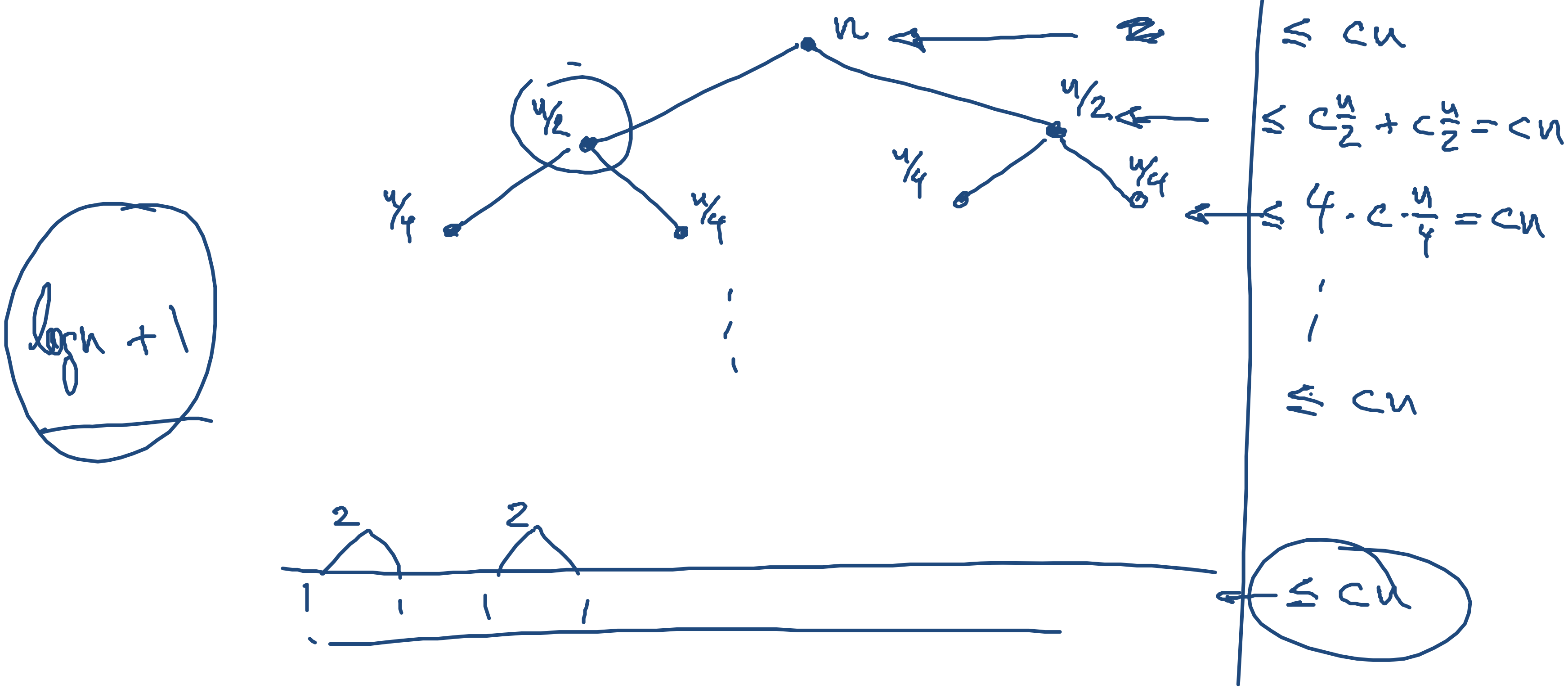
$$T(n) \leq 2T(n/2) + c \cdot n$$
$$\leq 2\left(2T(n/4) + \frac{cn}{2}\right) + cn = 4T(n/4) + 2cn$$
$$\leq 4\left(2T(n/8) + \frac{cn}{4}\right) + 2cn = 8T(n/8) + 3cn$$
$$\vdots$$
$$\leq nT(1) + c \cdot n \cdot \log_2 n \leq cn(\log_2 n + 1)$$

Analysis, Alternative Guessing

Recurrence relation:

$$T(n) \leq \underline{2 \cdot T(n/2)} + c \cdot n, \qquad T(1) \leq c$$

Recursion Tree:



Analysis, Induction

Recurrence relation:

$$T(n) \le 2 \cdot T(n/2) + c \cdot n, \qquad T(1) \le c$$

Verify by induction:

guess: $T(n) \le cn(\log n + 1)$

base: $T(1) \le c(\log 1 + 1) = c$ ✓

ind. step: $T(n) \le 2T(n/2) + cn$

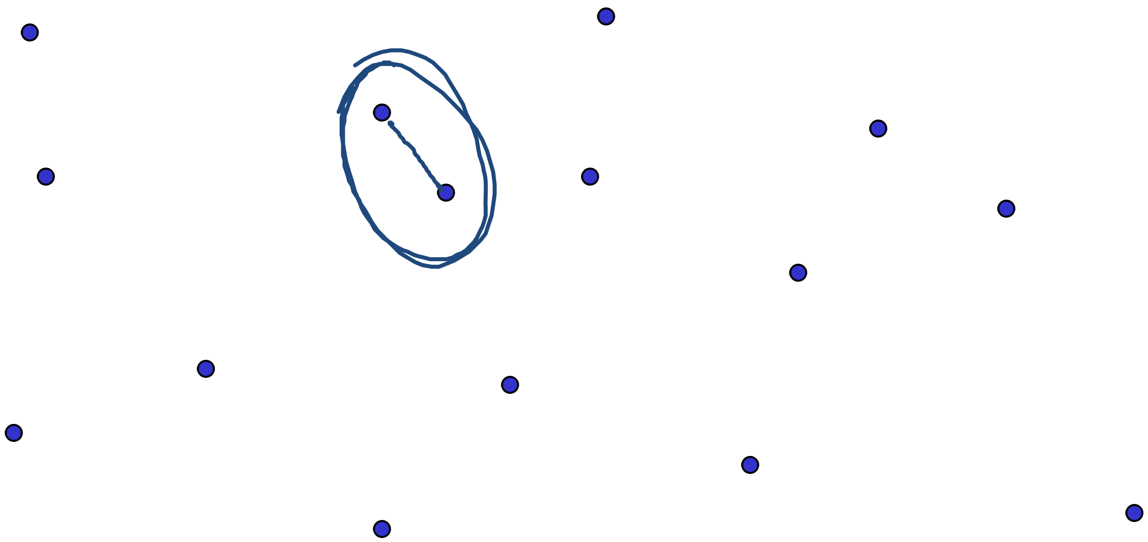
$\overset{\text{I.H.}}{\le} 2\left(c \cdot \frac{n}{2}\left(\log \frac{n}{2} + 1\right)\right) + cn$

$= cn \log n + cn$

$= cn(\log n + 1)$ □

Geometric divide-and-conquer

Closest Pair Problem: Given a set S of n points, find a pair of points with the smallest distance.

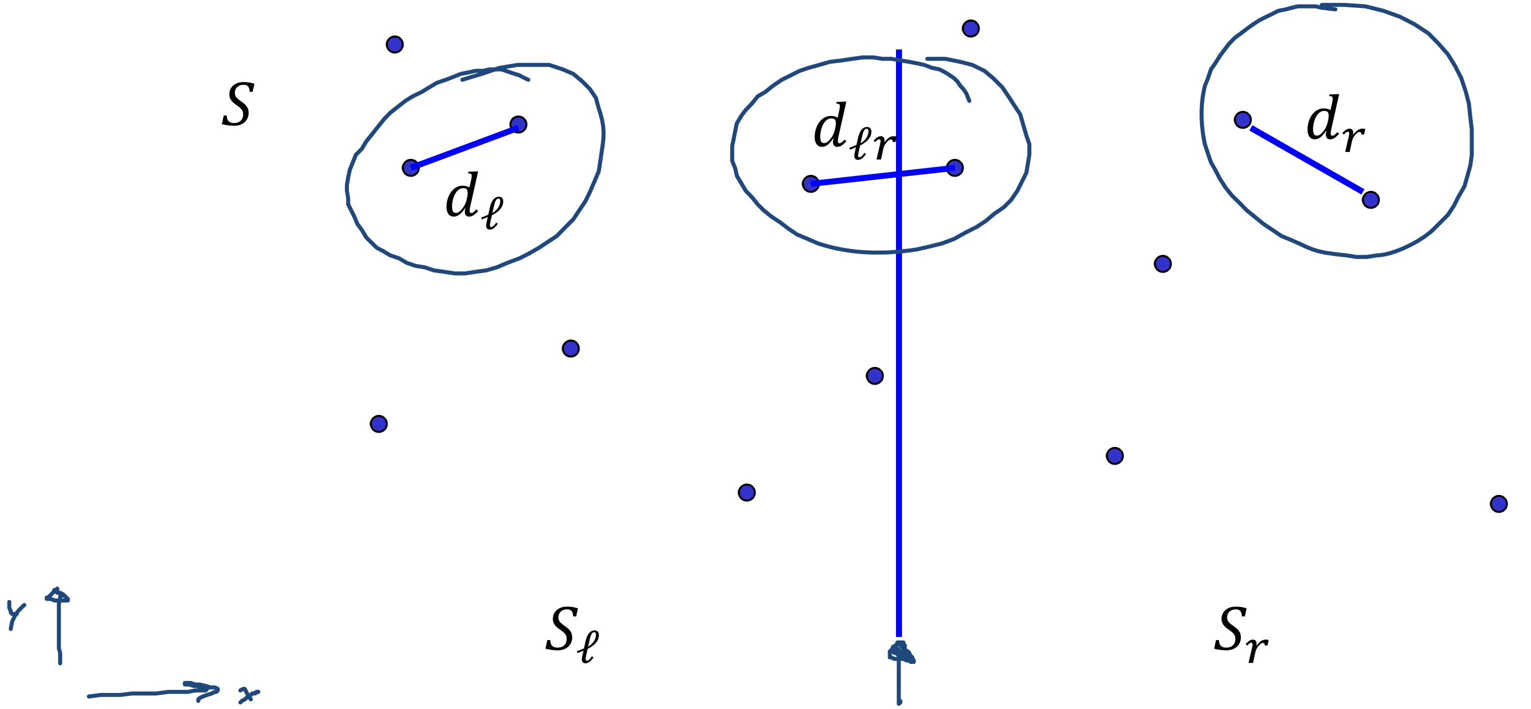


Naive solution: check all pairs

$O(n^2)$

Divide-and-conquer solution

1. **Divide:** Divide S into two equal sized sets S_ℓ und S_r.
2. **Conquer:** $d_\ell = \text{mindist}(S_\ell)$ $\quad d_r = \text{mindist}(Sr)$
3. **Combine:** $d_{\ell r} = \min\{d(p_\ell, p_r) \mid p_\ell \in S_\ell, p_r \in S_r\}$
   return $\min\{d_\ell, d_r, d_{\ell r}\}$

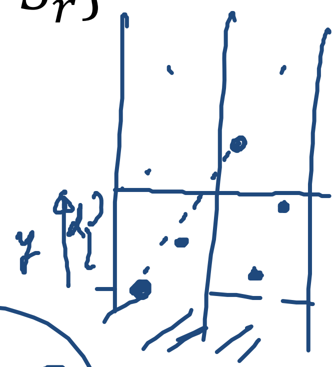
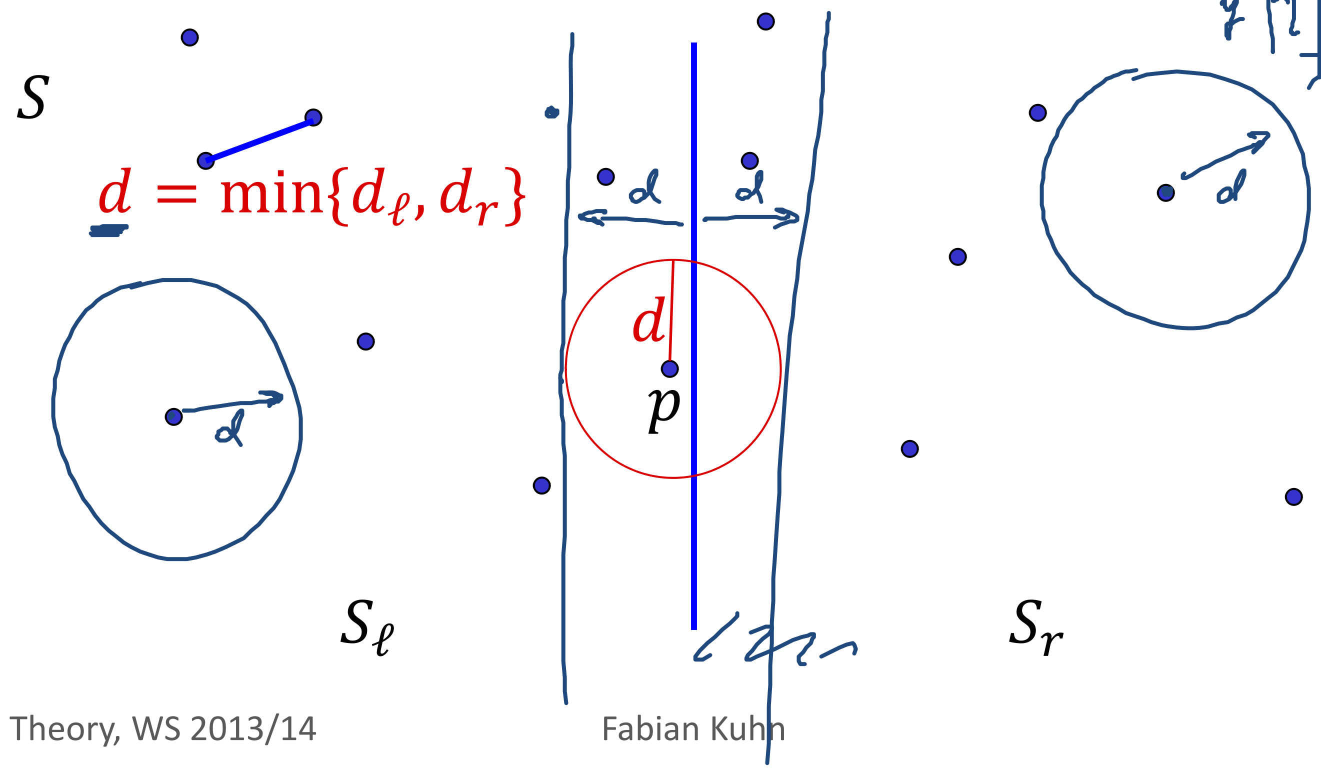


Divide-and-conquer solution

1. **Divide:** Divide S into two equal sized sets S_ℓ und S_r.
2. **Conquer:** $d_\ell = \text{mindist}(S_\ell)$ $\quad d_r = \text{mindist}(Sr)$
3. **Combine:** $d_{\ell r} = \min\{d(p_\ell, p_r) \mid p_\ell \in S_\ell, p_r \in S_r\}$
   return $\min\{d_\ell, d_r, d_{\ell r}\}$

Computation of $d_{\ell r}$:

only if $d_{\ell r} < d$

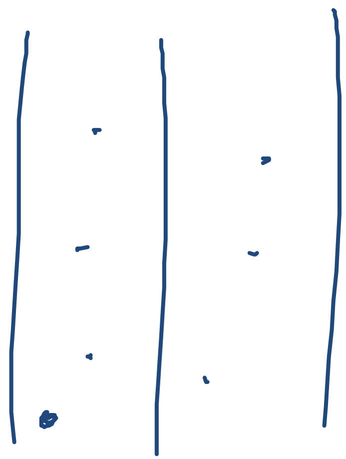



Merge step

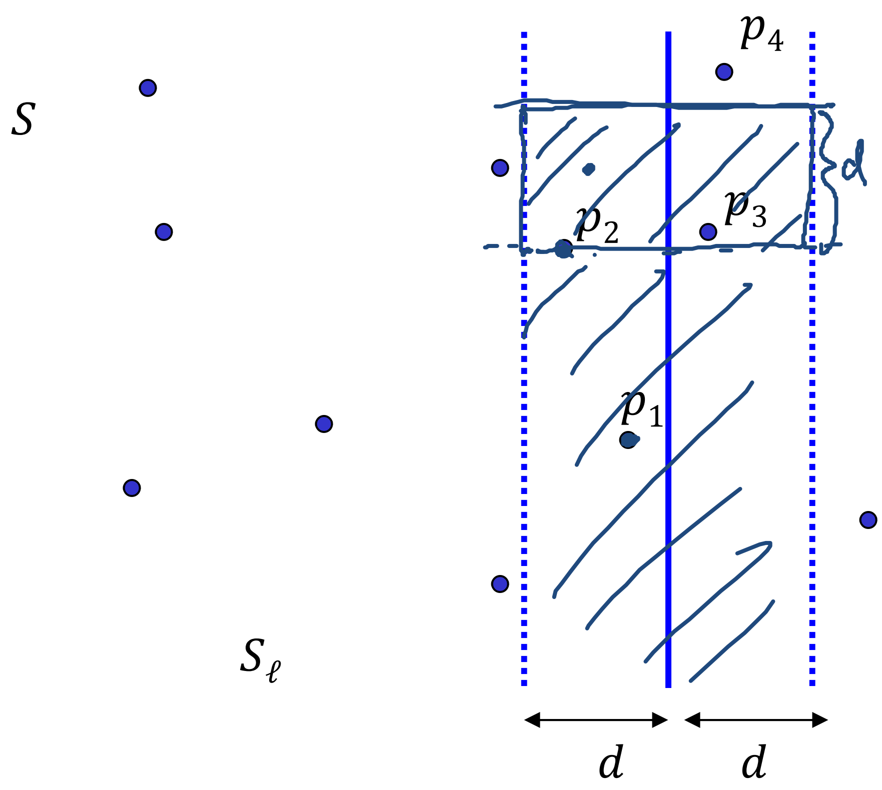
1. Consider only points within distance $< d$ of the bisection line, in the order of increasing y-coordinates.

2. For each point p consider all points q within y-distance less than d

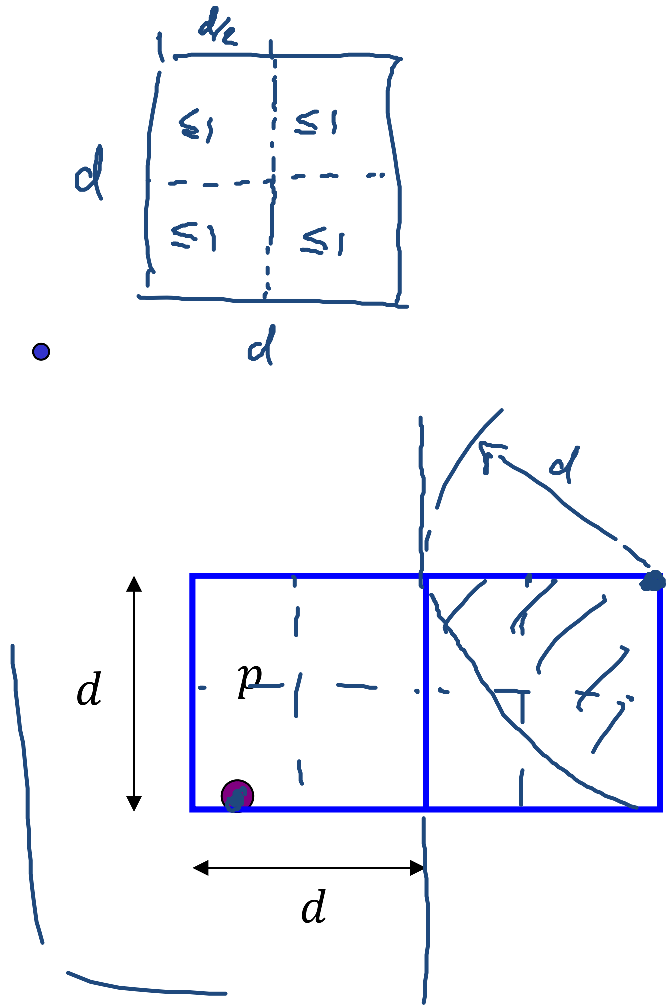
3. There are at most 7 such points.



Combine step



$d = \min\{d_\ell, dr\}$



Implementation

- Initially sort the points in S in order of increasing x-coordinates

  $O(n \log n)$

- While computing closest pair, also sort S according to y-coord.
  - Partition S into S_ℓ and S_r, solve and sort sub-problems recursively
  - Merge to get sorted S according to y-coordinates

    $O(n)$
  - Center points: points within x-distance $d = \min\{d_\ell, d_r\}$ of center
  - Go through center points in S in order of incr. y-coordinates

combine: $O(n)$

Running Time

Recurrence relation:

$$T(n) = 2 \cdot T(n/2) + c \cdot n, \qquad T(1) = a$$

Solution:

- Same as for computing number of number of inversions, merge sort (and many others…)

$$T(n) = O(n \cdot \log n)$$

Recurrence Relations: Master Theorem

Recurrence relation

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

const

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n), \qquad T(n) = O(1) \text{ for } n \le n_0$$

$T(1) = O(1)$

Cases

- $f(n) = O(n^c),\ c < \log_b a$

$$T(n) = \Theta\left(n^{\log_b a}\right)$$

- $f(n) = \Omega(n^c),\ c > \log_b a$ $\leftarrow$ $T(n) = 2T(n/2) + c \cdot n^2$

$$T(n) = \Theta\left(f(n)\right)$$

- $f(n) = \Theta\left(n^c \cdot \log^k n\right),\ c = \log_b a$

$$T(n) = \Theta\left(n^c \cdot \log^{k+1} n\right)$$