Lecture 4

Divide and Conquer II: Counting inversions, counting intersections, max subarray, maxima set

CS 161 Design and Analysis of Algorithms

Ioannis Panageas

Divide and conquer method

Steps of method:

– **Divide** input into parts (smaller problems)

– **Conquer** (solve) each part recursively

– **Combine** results to obtain solution of original

$$
\begin{aligned}
T(n) = &\ \text{divide time} \\
&+ T(n_1) + T(n_2) + \ldots + T(n_k) \\
&+ \text{combine time}
\end{aligned}
$$

Case study I: Counting inversions

Given numbers $A_1, \ldots, A_n$ in an array A, compute the number of inversions.

(i, j) is an inversion: $A_i > A_j$ and $i < j$.

Example [18, 29, 12, 15, 32, 10] has 9 inversions:

(18,12), (18,15), (18,10), (29,12), (29,15), (29,10), (12,10), (15,10), (32,10)

Case study I: Counting inversions

- Solution: Use Divide and conquer. Tricky part the combine step.

Run a modification of Mergesort that has a counter that counts inversions during merge steps.

- Question: Assume that $B_1, \dots, B_k$ and $C_1, \dots, C_l$ are both sorted. Can you compute the number of inversions of the concatenated sequence $B_1, \dots, B_k, C_1, \dots, C_l$?

Case study I: Counting inversions

- Question: Assume that $B_1, \dots, B_k$ and $C_1, \dots, C_l$ are both sorted. Can you compute the number of inversions of the sequence $B_1, \dots, B_k, C_1, \dots, C_l$?

If $B_i > C_j \geq B_{i-1}$ there are

including C_j

$B_1, \dots, B_i, \dots, B_k$

i

$C_1, \dots, C_j, \dots, C_l$

j

Case study I: Counting inversions

- Question: Assume that $B_1, \dots, B_k$ and $C_1, \dots, C_l$ are both sorted. Can you compute the number of inversions of the sequence $B_1, \dots, B_k, C_1, \dots, C_l$?

If $B_i > C_j \geq B_{i-1}$ there are

$k - i + 1$ including C_j

$B_1, \dots, B_i, \dots, B_k$	$C_1, \dots, C_j, \dots, C_l$
i	j

Case study I: Counting inversions

If $B_i > C_j \geq B_{i-1}$ there are

$k - i + 1$ including C_j

$B_1, \ldots, B_i, \ldots, B_k$	$C_1, \ldots, C_j, \ldots, C_l$
i	j

B : 1 3 4 9 220 $\quad$ C : 2 3 5 7 8 10

Concatenated: 1 3 4 9 220 2 3 5 7 8 10

8 participates in 2 inversions. $k = 5, i = 4$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{aligned}
&\textbf{While } k < \text{len(B)} + \text{len(C)} \textbf{ do}\\
&\quad \textbf{If } B[i] <= C[j] \textbf{ then}\\
&\quad\quad A[k] \leftarrow B[i]\\
&\quad\quad i = i + 1, k = k + 1\\
&\quad \textbf{else}\\
&\quad\quad A[k] \leftarrow C[j]\\
&\quad\quad \text{counter} = \text{counter} + \text{len}(B) - i + 1\\
&\quad\quad j = j + 1, k = k + 1
\end{aligned}
$$

B : 1 3 4 9 220 $\qquad$ C : 2 3 5 7 8 10

i $\qquad$ j

A :

$\text{counter} = 0$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{array}{l}
\textbf{While } k < \text{len(B)} + \text{len(C)} \textbf{ do} \\
\quad \textbf{If } B[i] <= C[j] \textbf{ then} \\
\quad\quad A[k] \leftarrow B[i] \\
\quad\quad i = i + 1, k = k + 1 \\
\quad \textbf{else} \\
\quad\quad A[k] \leftarrow C[j] \\
\quad\quad \text{counter} = \text{counter} + \text{len}(B) - i + 1 \\
\quad\quad j = j + 1, k = k + 1
\end{array}
$$

B : 1 3 4 9 220 $\quad$ C : 2 3 5 7 8 10

i $\quad$ j

$A : 1$

$\text{counter} = 0$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

While $k < \text{len(B)} + \text{len(C)}$ **do**

  If $B[i] <= C[j]$ **then**

    $A[k] \leftarrow B[i]$

    $i = i + 1, k = k + 1$

  else

    $A[k] \leftarrow C[j]$

    $\text{counter} = \text{counter} + \text{len}(B) - i + 1$

    $j = j + 1, k = k + 1$

B : 1 3 4 9 220 C : 2 3 5 7 8 10

i j

A : 1 2

$\text{counter} = 4$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do}$$

$$\quad \textbf{If } B[i] <= C[j] \textbf{ then}$$

$$\quad\quad A[k] \leftarrow B[i]$$

$$\quad\quad i = i + 1, k = k + 1$$

$$\quad \textbf{else}$$

$$\quad\quad A[k] \leftarrow C[j]$$

$$\quad\quad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1$$

$$\quad\quad j = j + 1, k = k + 1$$

B : 1 3 4 9 220 $\quad$ C : 2 3 5 7 8 10

i $\quad$ j

A : 1 2 3

$\mathrm{counter} = 4$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{array}{l}
\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do} \\
\quad \textbf{If } B[i] <= C[j] \textbf{ then} \\
\quad\quad A[k] \leftarrow B[i] \\
\quad\quad i = i + 1, k = k + 1 \\
\quad \textbf{else} \\
\quad\quad A[k] \leftarrow C[j] \\
\quad\quad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1 \\
\quad\quad j = j + 1, k = k + 1
\end{array}
$$

$B : 1\ 3\ 4\ 9\ 220$ $\qquad$ $C : 2\ 3\ 5\ 7\ 8\ 10$

i $\qquad$ j

$A : 1\ 2\ 3\ 3$

$\mathrm{counter} = 7$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{aligned}
&\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do}\\
&\quad \textbf{If } B[i] <= C[j] \textbf{ then}\\
&\qquad A[k] \leftarrow B[i]\\
&\qquad i = i+1, k = k+1\\
&\quad \textbf{else}\\
&\qquad A[k] \leftarrow C[j]\\
&\qquad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1\\
&\qquad j = j+1, k = k+1
\end{aligned}
$$

$B: 1\ 3\ 4\ 9\ 220$ $\qquad$ $C: 2\ 3\ 5\ 7\ 8\ 10$

i $\qquad$ j

$A: 1\ 2\ 3\ 3\ 4$

$\mathrm{counter} = 7$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{array}{l}
\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do} \\
\quad \textbf{If } B[i] <= C[j] \textbf{ then} \\
\quad\quad A[k] \leftarrow B[i] \\
\quad\quad i = i + 1, k = k + 1 \\
\quad \textbf{else} \\
\quad\quad A[k] \leftarrow C[j] \\
\quad\quad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1 \\
\quad\quad j = j + 1, k = k + 1
\end{array}
$$

$B : 1\ 3\ 4\ 9\ 220$ $\quad$ $C : 2\ 3\ 5\ 7\ 8\ 10$

i $\quad$ j

$A : 1\ 2\ 3\ 3\ 4\ 5$

$\mathrm{counter} = 9$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

While $k < \text{len(B)} + \text{len(C)}$ **do**

  If $B[i] <= C[j]$ **then**

    $A[k] \leftarrow B[i]$

    $i = i + 1, k = k + 1$

  else

    $A[k] \leftarrow C[j]$

    $\text{counter} = \text{counter} + \text{len}(B) - i + 1$

    $j = j + 1, k = k + 1$

B : 1 3 4 9 220 C : 2 3 5 7 8 10

i j

A : 1 2 3 3 4 5 7

$\text{counter} = 11$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

While $k < \mathrm{len(B)} + \mathrm{len(C)}$ **do**

  If $B[i] <= C[j]$ **then**

    $A[k] \leftarrow B[i]$

    $i = i + 1, k = k + 1$

  else

    $A[k] \leftarrow C[j]$

    $\mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1$

    $j = j + 1, k = k + 1$

$B: 1\ 3\ 4\ 9\ 220$ $C: 2\ 3\ 5\ 7\ 8\ 10$

i j

$A: 1\ 2\ 3\ 3\ 4\ 5\ 7\ 8$

$\mathrm{counter} = 13$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{aligned}
&\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do} \\
&\quad \textbf{If } B[i] <= C[j] \textbf{ then} \\
&\qquad A[k] \leftarrow B[i] \\
&\qquad i = i + 1, k = k + 1 \\
&\quad \textbf{else} \\
&\qquad A[k] \leftarrow C[j] \\
&\qquad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1 \\
&\qquad j = j + 1, k = k + 1
\end{aligned}
$$

$B : 1\ 3\ 4\ 9\ 220$ $\quad$ $C : 2\ 3\ 5\ 7\ 8\ 10$

i $\quad$ j

$A : 1\ 2\ 3\ 3\ 4\ 5\ 7\ 8\ 9$

$\mathrm{counter} = 13$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$
\begin{aligned}
&\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do} \\
&\quad \textbf{If } B[i] <= C[j] \textbf{ then} \\
&\qquad A[k] \leftarrow B[i] \\
&\qquad i = i + 1, k = k + 1 \\
&\quad \textbf{else} \\
&\qquad A[k] \leftarrow C[j] \\
&\qquad \text{counter} = \text{counter} + \mathrm{len}(B) - i + 1 \\
&\qquad j = j + 1, k = k + 1
\end{aligned}
$$

$B : 1\ 3\ 4\ 9\ 220$ $\qquad$ $C : 2\ 3\ 5\ 7\ 8\ 10$

i $\qquad$ j

$A : 1\ 2\ 3\ 3\ 4\ 5\ 7\ 8\ 9\ 10$

$\text{counter} = 14$

Case study I: Counting inversions

Problem: Given two sorted arrays B, C, merge them to a sorted array and count number of inversions simultaneously.

Solution: Index i for B, index j for C, index k for A, counter.

$$\textbf{While } k < \mathrm{len(B)} + \mathrm{len(C)} \textbf{ do}$$
$$\quad \textbf{If } B[i] <= C[j] \textbf{ then}$$
$$\quad\quad A[k] \leftarrow B[i]$$
$$\quad\quad i = i + 1, k = k + 1$$
$$\quad \textbf{else}$$
$$\quad\quad A[k] \leftarrow C[j]$$
$$\quad\quad \mathrm{counter} = \mathrm{counter} + \mathrm{len}(B) - i + 1$$
$$\quad\quad j = j + 1, k = k + 1$$

B : 1 3 4 9 220 $\quad$ C : 2 3 5 7 8 10

i $\quad$ j

A : 1 2 3 3 4 5 7 8 9 10 220

$\mathrm{counter} = 14$

Case study I: Counting inversions

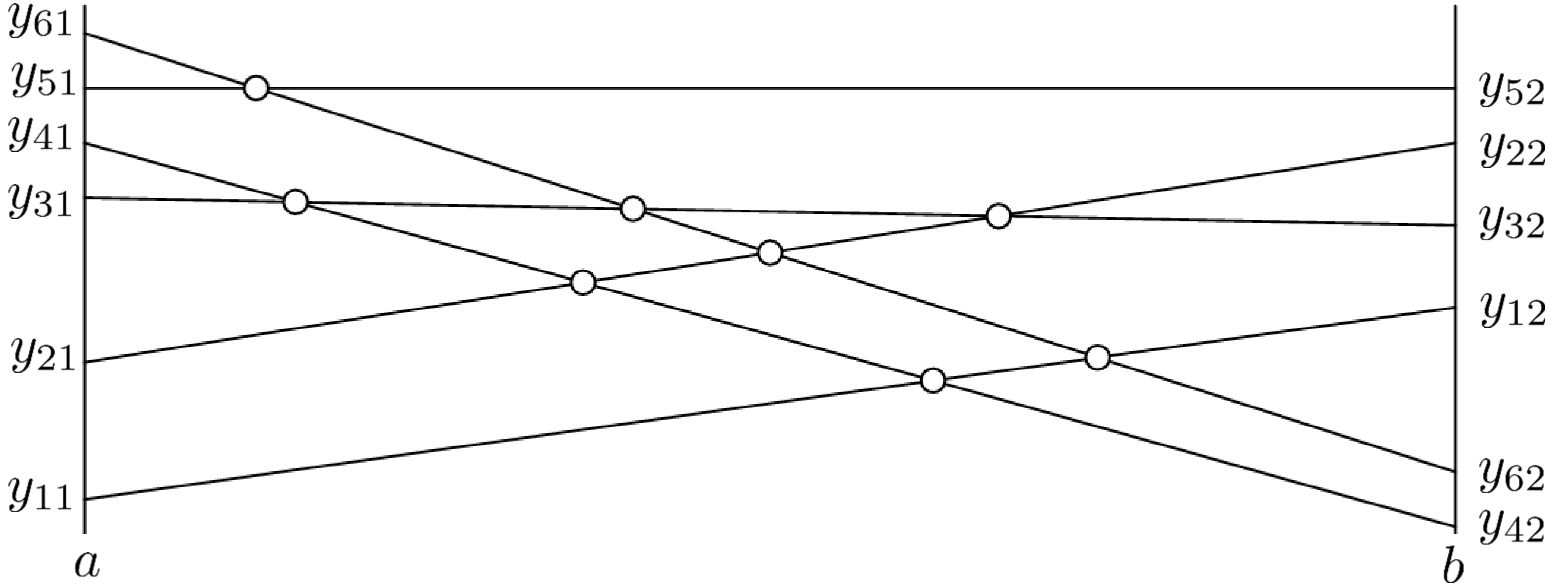
Pseudocode:

$$
\begin{array}{l}
\text{ModifiedMergesort}(A[1:n]) \\
\quad \textbf{If } n == 1 \textbf{ then} \\
\quad\quad \textbf{return } A, 0 \\
\quad B, \textbf{countL} \;=\; \text{ModifiedMergesort }(A[1:\frac{n}{2}]) \\
\quad C, \textbf{countR} \;=\; \text{ModifiedMergesort }(A[\frac{n}{2}+1:n]) \\
\quad A, \text{counterM} \leftarrow \text{ModifiedMerge}(B, C) \\
\quad \textbf{return } A, \text{countL} + \text{countR} + \text{counterM}
\end{array}
$$

Case study II: Counting intersections

Problem: Given n distinct lines in the plane, none of which are vertical and two vertical lines $x = a$ and $x = b$, find the number of intersections. We assume that each line i is described by its endpoints (a, y_{i1}) and (b, y_{i2}).

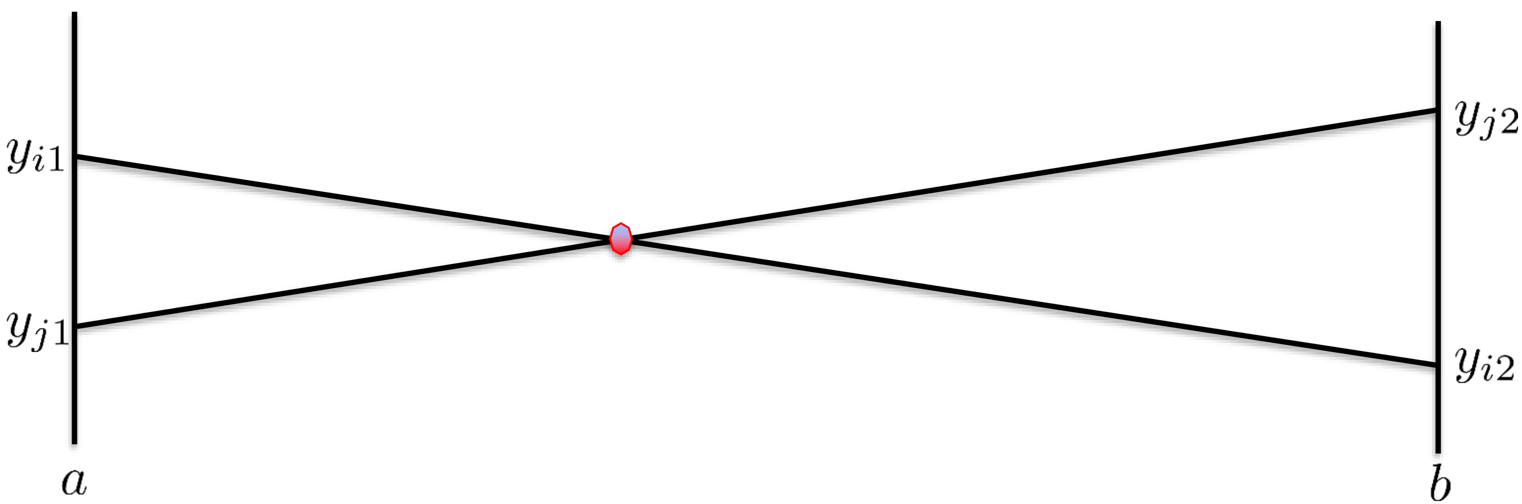
Example: 6 lines (8 intersections)



Case study II: Counting intersections

Problem: Given n distinct lines in the plane, none of which are vertical and two vertical lines $x = a$ and $x = b$, find the number of intersections. We assume that each line i is described by its endpoints (a, y_{i1}) and (b, y_{i2}).

Question: When two lines i, j intersect?

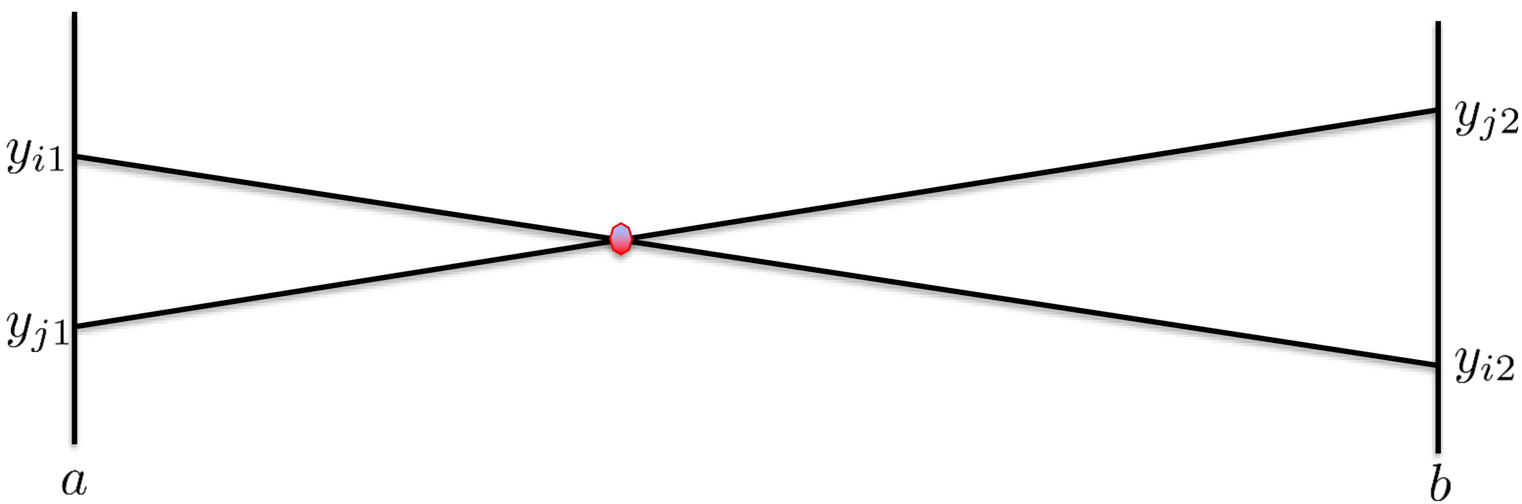


Case study II: Counting intersections

Problem: Given n distinct lines in the plane, none of which are vertical and two vertical lines $x = a$ and $x = b$, find the number of intersections. We assume that each line i is described by its endpoints (a, y_{i1}) and (b, y_{i2}).

Question: When two lines i, j intersect?

If $\boldsymbol{y_{i1} > y_{j1}}$ **then** $\boldsymbol{y_{i2} < y_{j2}}$ or If $\boldsymbol{y_{i1} < y_{j1}}$ **then** $\boldsymbol{y_{i2} > y_{j2}}$



Case study II: Counting intersections

For all pairs i, j with $i < j$, count number of intersections

Pseudocode:

$$
\begin{array}{l}
\text{counter} \leftarrow 0 \\
\textbf{For } i = 1 \text{ to } n \textbf{ do} \\
\quad \textbf{For } j = i+1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{If } (y_{i1} > y_{j1} \text{ and } y_{i2} < y_{j2}) \text{ or } (y_{i1} < y_{j1} \text{ and } y_{i2} > y_{j2}) \textbf{ then} \\
\quad\quad\quad \text{counter} \leftarrow \text{counter} + 1 \\
\textbf{return } \text{counter}
\end{array}
$$

Case study II: Counting intersections

For all pairs i, j with $i < j$, count number of intersections

Pseudocode:

$$
\begin{array}{l}
\text{counter} \leftarrow 0 \\
\textbf{For } i = 1 \text{ to } n \textbf{ do} \\
\quad \textbf{For } j = i + 1 \text{ to } n \textbf{ do} \\
\quad\quad \textbf{If } (y_{i1} > y_{j1} \text{ and } y_{i2} < y_{j2}) \text{ or } (y_{i1} < y_{j1} \text{ and } y_{i2} > y_{j2}) \textbf{ then} \\
\quad\quad\quad \text{counter} \leftarrow \text{counter} + 1 \\
\textbf{return } \text{counter}
\end{array}
$$

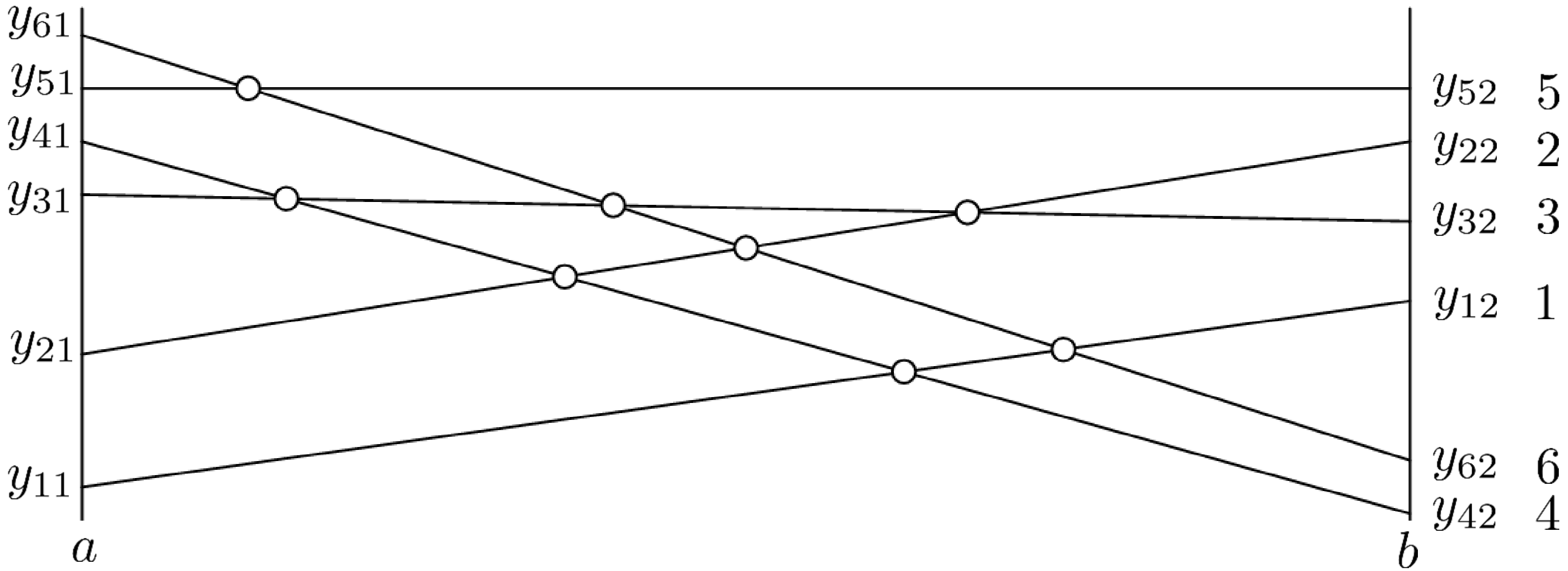
Running time $\Theta(n^2)$

Can we do better?

Case study II: Counting intersections

Idea: Let's sort the lines with respect to y on a. Check the inverse permutation of the indices of the lines on b.

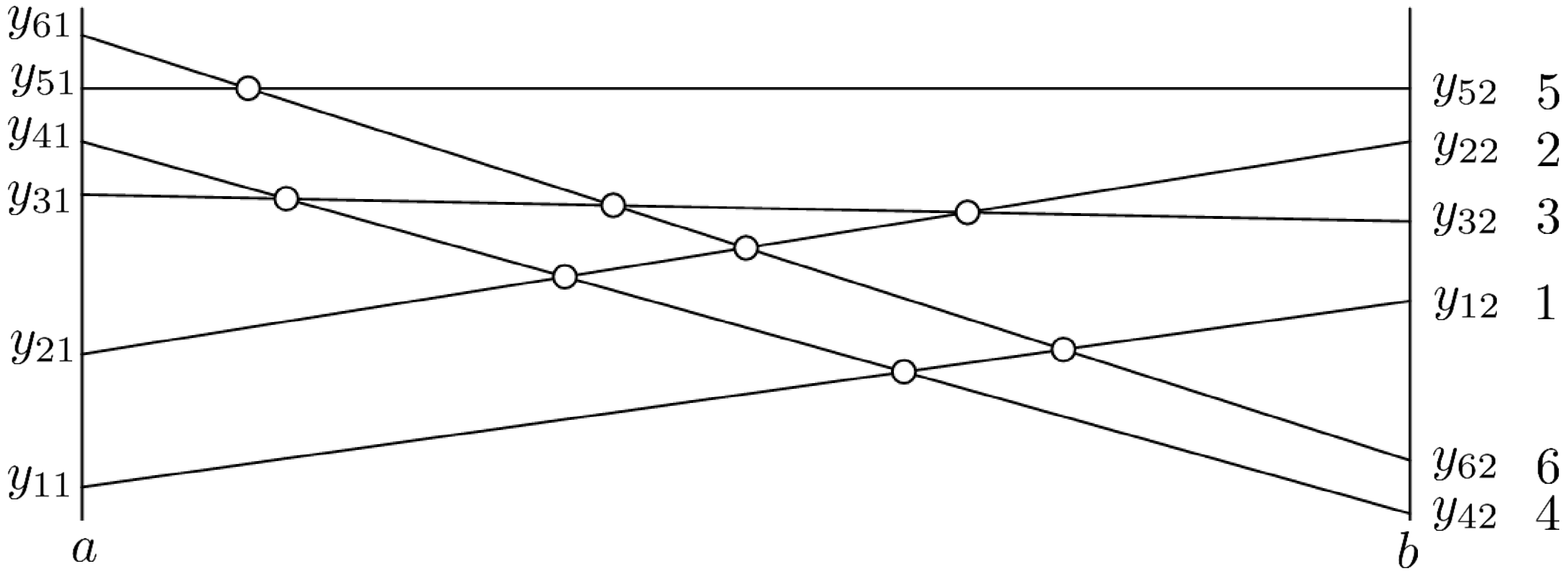
Example: 4, 6, 1, 3, 2, 5



Case study II: Counting intersections

Idea: Let's sort the lines with respect to y on a. Check the inverse permutation of the indices of the lines on b.

Example: 4, 6, 1, 3, 2, 5

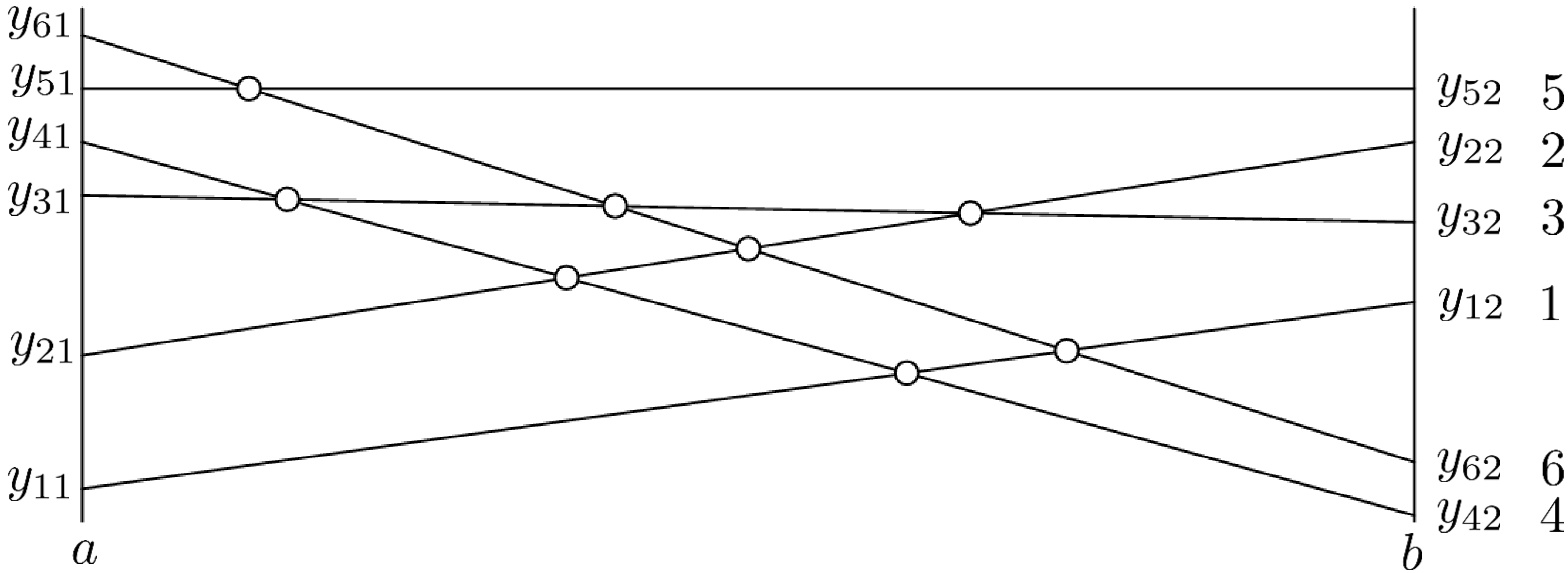


Key observation: Number of inversions is equal to number of intersections. In example (4, 1), (4,3), (4,2), (6,1), (6,3), (6,2), (6,5), (3,2)

Case study II: Counting intersections

Idea: Let's sort the lines with respect to y on a. Check the inverse permutation of the indices of the lines on b.

Example: 4, 6, 1, 3, 2, 5



Solution: Sort the lines with respect to y on a. Run modified mergesort to find number of inversions. Running time $\Theta(n \log n)$.

Case study III: Maximum subarray

Problem (Leetcode, question in interviews): Given an array A of n numbers (positive and negative), find the subarray with the maximum sum.

Example: $A = [-2, -5, 6, -2, -3, 1, 5, -6]$

Case study III: Maximum subarray

Problem (Leetcode, question in interviews): Given an array A of n numbers (positive and negative), find the subarray with the maximum sum.

Example: $A = [-2, -5, 6, -2, -3, 1, 5, -6]$

Solution of example:

$$[-2, -5, \mathbf{6}, \mathbf{-2}, \mathbf{-3}, \mathbf{1}, \mathbf{5}, -6] \text{ with sum 7.}$$

Case study III: Maximum subarray

For all i, j with $i \leq j$, compute $A_i + A_{i+1} + \cdots + A_j$. Keep the maximum from all sums. Total number of computations is…

Case study III: Maximum subarray

For all i, j with $i \leq j$, compute $A_i + A_{i+1} + \cdots + A_j$. Keep the maximum from all sums. Total number of computations is…

Pseudocode:

$$
\begin{array}{l}
\max \leftarrow 0 \\
\textbf{For } i = 1 \text{ to } n \textbf{ do} \\
\quad \textbf{For } j = i \text{ to } n \textbf{ do} \\
\qquad \text{sum} \leftarrow 0 \\
\qquad \textbf{For } k = i \text{ to } j \textbf{ do} \\
\qquad\quad \text{sum} = \text{sum} + A[k] \\
\qquad \textbf{If } \text{sum} > \max \textbf{ then} \\
\qquad\quad \max \leftarrow \text{sum} \\
\textbf{return } \max
\end{array}
$$

Case study III: Maximum subarray

For all i, j with $i < j$, compute $A_i + A_{i+1} + \cdots + A_j$. Keep the maximum from all sums. Total number of computations is

$$\sum_{i=1}^{n} \sum_{j=i}^{n} (j - i + 1) \text{ which is } \Theta(n^3)$$

Pseudocode:

$$
\begin{array}{l}
\max \leftarrow 0 \\
\textbf{For } i = 1 \text{ to } n \textbf{ do} \\
\quad \textbf{For } j = i \text{ to } n \textbf{ do} \\
\quad\quad \text{sum} \leftarrow 0 \\
\quad\quad \textbf{For } k = i \text{ to } j \textbf{ do} \\
\quad\quad\quad \text{sum} = \text{sum} + A[k] \\
\quad\quad \textbf{If } \text{sum} > \max \textbf{ then} \\
\quad\quad\quad \max \leftarrow \text{sum} \\
\textbf{return } \max
\end{array}
$$

Can we do better?

Case study III: Maximum subarray

Idea 1: Do first preprocessing. Compute partial sums $S_i = A_1 + \cdots + A_i$ for every i. Running time $\Theta(n)$.

Observe that $A_i + A_{i+1} + \cdots + A_j = S_j - S_{i-1}$

Case study III: Maximum subarray

Idea 1: Do first preprocessing. Compute partial sums

$S_i = A_1 + \cdots + A_i$ for every i. Running time $\Theta(n)$.

Observe that $A_i + A_{i+1} + \cdots + A_j = S_j - S_{i-1}$

Then for all i, j with $i \leq j$, compute the maximum among $S_j - S_{i-1}$.

Case study III: Maximum subarray

Idea 1: Do first preprocessing. Compute partial sums $S_i = A_1 + \cdots + A_i$ for every i. Running time $\Theta(n)$.

Observe that $A_i + A_{i+1} + \cdots + A_j = S_j - S_{i-1}$

Then for all i, j with $i \leq j$, compute the maximum among $S_j - S_{i-1}$.

$\max \leftarrow 0$
$S[0] \leftarrow 0$
For $i = 1$ to n **do**
  $S[i] \leftarrow S[i-1] + A[i]$
For $i = 1$ to n **do**
  For $j = i$ to n **do**
    If $S[j] - S[i-1] > \max$ **then**
      $\max \leftarrow S[j] - S[i-1]$
return $\max$

$[-2, -5, 6, -2, -3, 1, 5, -6]$

S = [0, -2, -7, -1, -3, -6, -5, 0, -6]

-5+6-2-3+1 = S[6] – S[1] = -3

Case study III: Maximum subarray

Idea 1: Do first preprocessing. Compute partial sums

$S_i = A_1 + \cdots + A_i$ for every i. Running time $\Theta(n)$.

Observe that $A_i + A_{i+1} + \cdots + A_j = S_j - S_{i-1}$

Then for all i, j with $i < j$, compute the maximum among $S_j - S_{i-1}$.

$\max \leftarrow 0$
$S[0] \leftarrow 0$
For $i = 1$ to n **do**
  $S[i] \leftarrow S[i-1] + A[i]$
For $i = 1$ to n **do**
  For $j = i$ to n **do**
    If $S[j] - S[i-1] > \max$ **then**
      $\max \leftarrow S[j] - S[i-1]$
return $\max$

Running time $\Theta(\mathbf{n}^2)$

Can we do better?

Case study III: Maximum subarray

Idea 2: Divide and conquer

$$[\mathbf{-2, -5, 6, -2, -3, 1, 5, -6}]$$

Find max in left half (e.g., green), find max in right half (e.g., black) and combine/merge. HOW?

Observe left part has maximum 6 and right part also 6. The solution is 7 though.

Case study III: Maximum subarray

Idea 2: Divide and conquer

$$[\mathbf{-2, -5, 6, -2, -3, 1, 5, -6}]$$

Find max in left half (e.g., green), find max in right half (e.g., black) and combine/merge. HOW?

Observe left part has maximum 6 and right part also 6. The solution is 7 though.

Key idea: The solution is either on **left** part, or **right** part or **crosses the midpoint (has at least one number in both parts).**

Case study III: Maximum subarray

Idea 2: Divide and conquer

$$[\mathbf{-2, -5, 6, -2, -3, 1, 5, -6}]$$

Question: How to get the maximum subarray that crosses the midpoint?

Case study III: Maximum subarray

Idea 2: Divide and conquer

$$[\mathbf{-2, -5, 6, -2, -3, 1, 5, -6}]$$

Question: How to get the maximum subarray that crosses the midpoint?

a) Find the maximum starting from mid and going left.
b) Find the maximum starting from mid+1 and going right.

Add them up. This can happen in $\Theta(n)$ time using partial sums $\boldsymbol{S}$.

In example above a) is 4 and b) is 3 for a total of 7.

Case study III: Maximum subarray

Pseudocode:

$S[0] \leftarrow 0$ **For** $i = 1$ to n **do**

$S[i] \leftarrow S[i-1] + A[i]$

$\text{Maxsum}(A[1:n])$

If $n == 1$ **return** $\max(A[1], 0)$

maxL $\leftarrow \text{Maxsum}(A[1:n/2])$

maxR $\leftarrow \text{Maxsum}(A[n/2+1:n])$

max1 $\leftarrow 0$

max2 $\leftarrow 0$

For $i = n/2$ downto 1 **do**

If max1 $< S[n/2] - S[i-1]$ **then max1** $\leftarrow S[n/2] - S[i-1]$

For $i = n/2 + 1$ to n **do**

If max2 $< S[i] - S[n/2]$ **then max2** $\leftarrow S[i] - S[n/2]$

return maximum of maxL, maxR and max1 + max2

Case study III: Maximum subarray

Pseudocode:

$S[0] \leftarrow 0$ **For** $i = 1$ to n **do**

$S[i] \leftarrow S[i-1] + A[i]$

$[-2, -5, 6, -2, \mathbf{-3}, \mathbf{1}, \mathbf{5}, \mathbf{-6}]$

maxL = 6, maxR = 6, max1+max2 = 4+3 = 7

$\mathrm{Maxsum}(A[1:n])$

If $n == 1$ **return** $\max(A[1], 0)$

maxL $\leftarrow \mathrm{Maxsum}(A[1:n/2])$

maxR $\leftarrow \mathrm{Maxsum}(A[n/2+1:n])$

max1 $\leftarrow 0$

max2 $\leftarrow 0$

For $i = n/2$ downto 1 **do**

If max1 $< S[n/2] - S[i-1]$ **then max1** $\leftarrow S[n/2] - S[i-1]$

For $i = n/2 + 1$ to n **do**

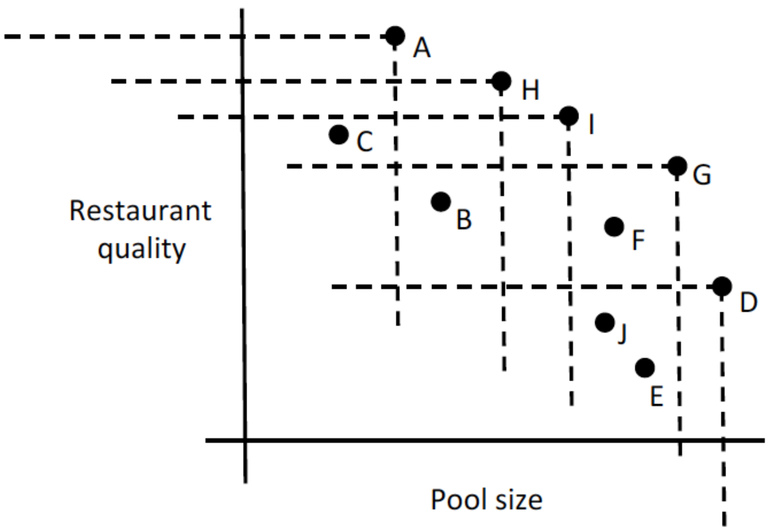
If max2 $< S[i] - S[n/2]$ **then max2** $\leftarrow S[i] - S[n/2]$

return maximum of maxL, maxR and max1 + max2

Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \ldots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

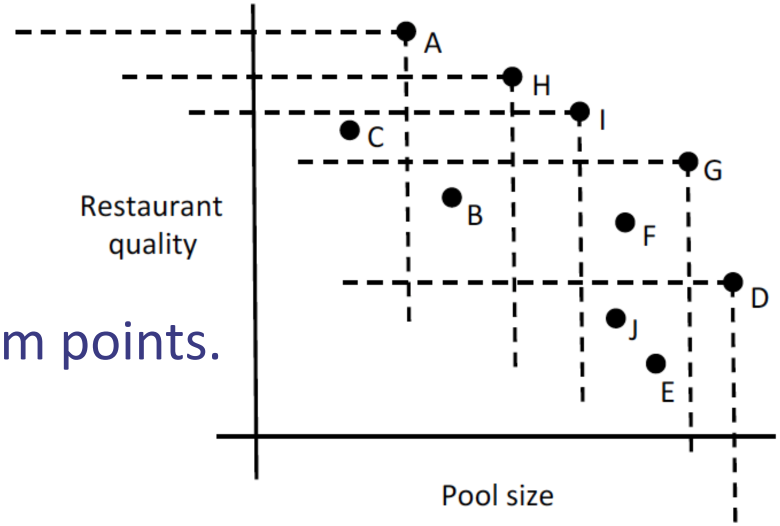
Example: x captures pool size and y restaurant quality. 10 hotels



Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \dots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Example: x captures pool size and y restaurant quality. 10 hotels



Explanation:

A, H, G, D are maximum points.

C, B, F, J, E are not.

Case study IV: Maxima Set

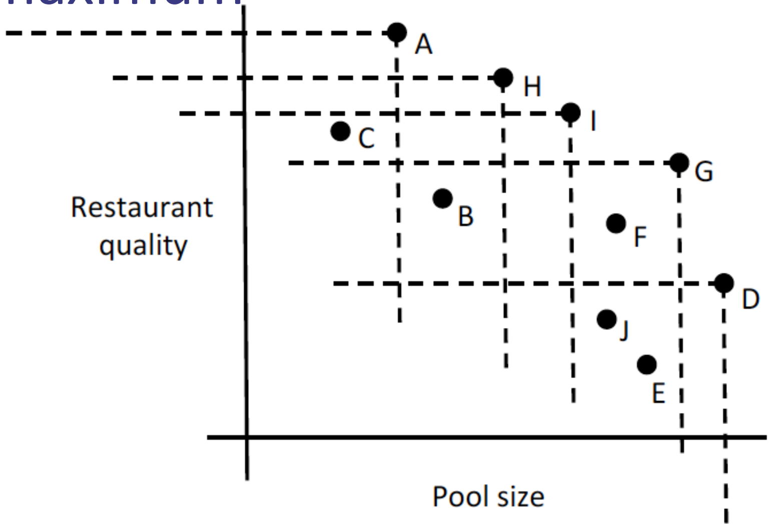
Problem: We are given n points $(x_1, y_1), \dots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Obvious approach:

For every point (x_i, y_i), check if it is maximum

To check if it is maximum, you check

the condition with all other points.



Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \ldots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Pseudocode:

counter $\leftarrow 0$
For $i = 1$ to n **do**
flag $\leftarrow 1$
  For $j = i + 1$ to n **do**
    If $(x_j > x_i$ and $y_j > y_i)$ **then** flag $\leftarrow 0$
counter $\leftarrow$ counter + flag
return counter

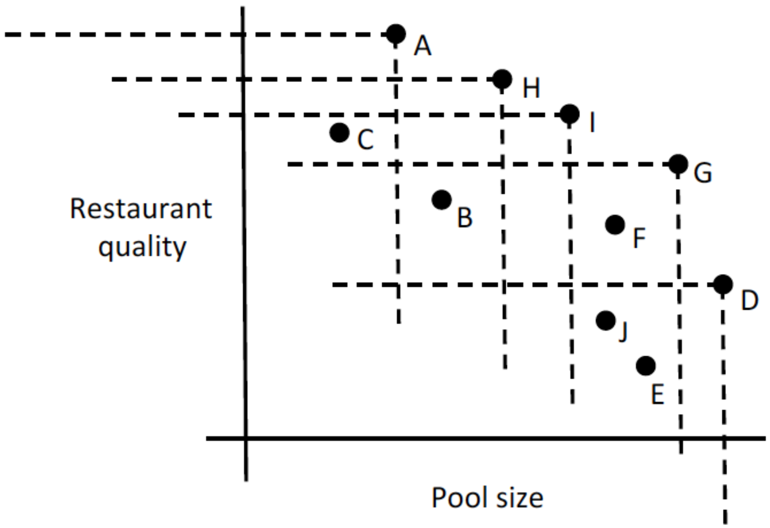
Running time $\Theta(n^2)$

Can we do better?

Case study IV: Maxima Set

Problem: We are given n points $(x_1, y_1), \ldots, (x_n, y_n)$ on the plane. A point (x_i, y_i) is called a maximum point if there is no other point (x_j, y_j) that $x_i \leq x_j$ and $y_i \leq y_j$.

Idea: Divide and conquer. Divide step and Combine step is challenging.



Case study IV: Maxima Set

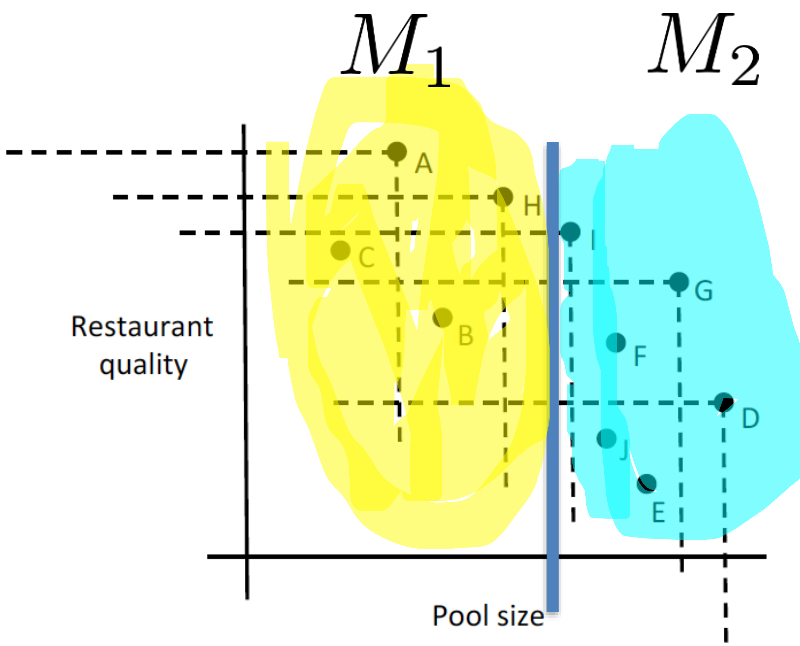
Divide step: It should split the points in two parts of equal size.

How?

Case study IV: Maxima Set

Divide step: It should split the points in two parts of equal size.

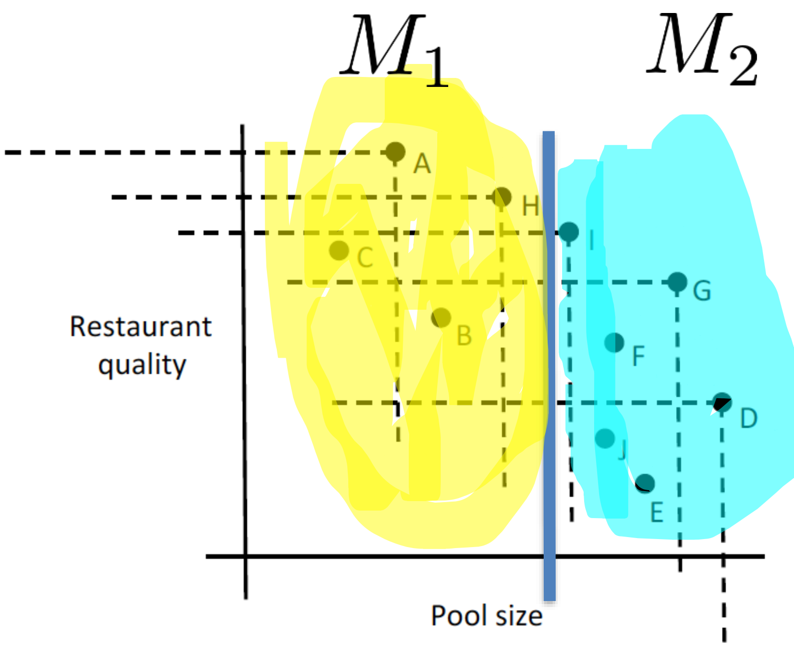
How? Choose the middle (median) point with respect to x coordinates.



Case study IV: Maxima Set

Divide step: It should split the points in two parts of equal size.

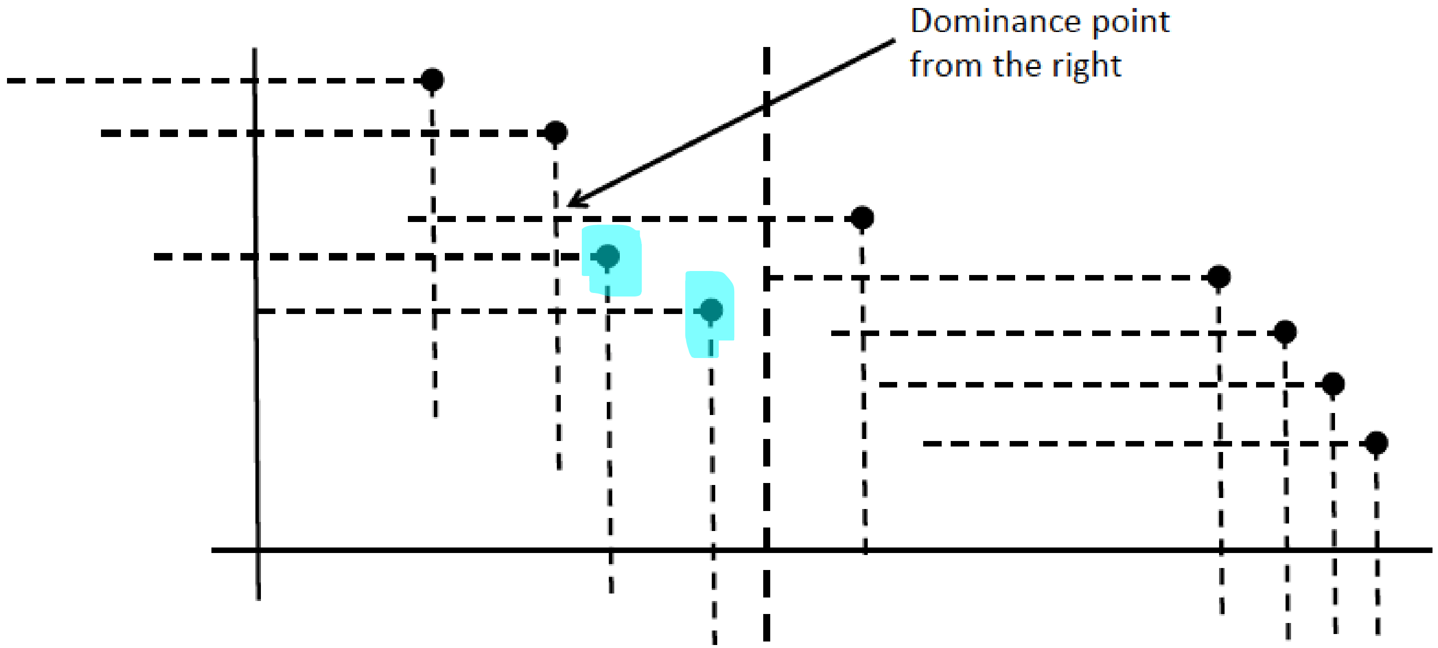
How? Choose the middle (median) point with respect to x coordinates.



Combine step: Return $M_1 \cup M_2$?

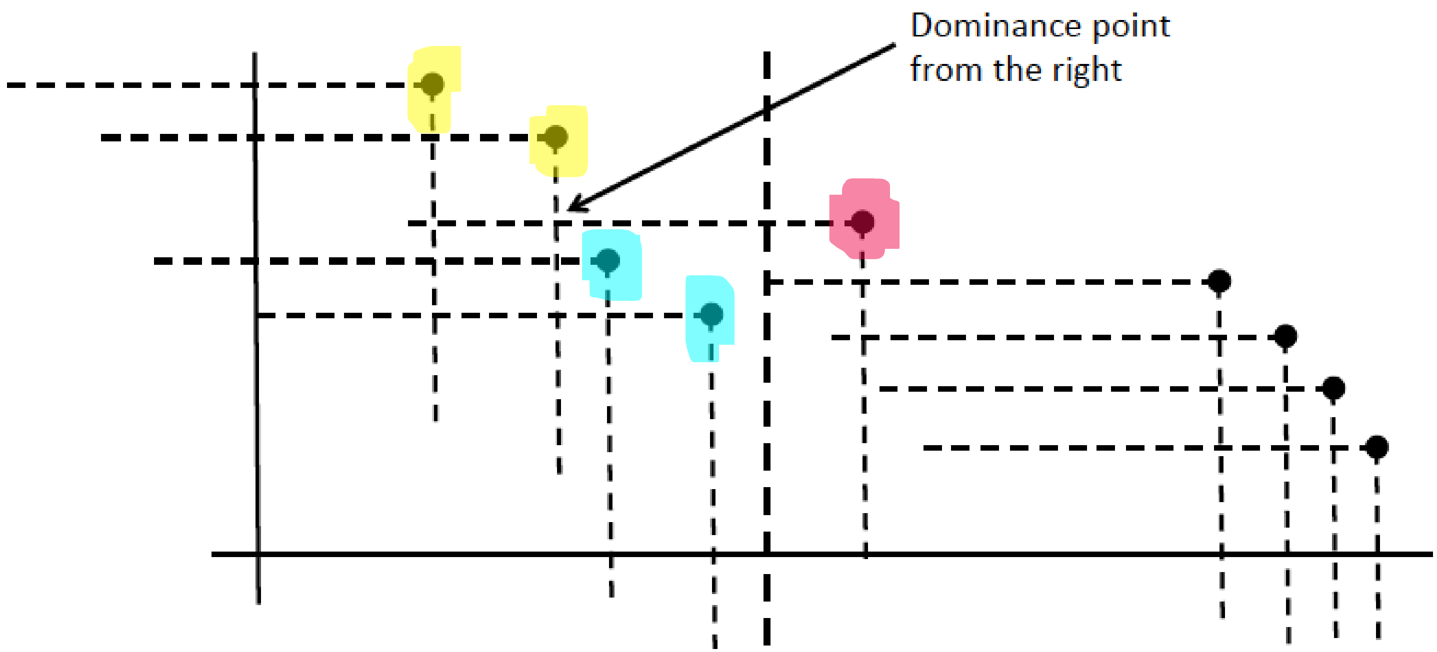
Case study IV: Maxima Set

Combine step: Return $M_1 \cup M_2$? Wrong: blue points below of M_1 are not part of the solution



Case study IV: Maxima Set

Combine step idea: M_2 points should part of the solution. From M_1, the points that are maximum should not be dominated by smallest with respect to x coordinates in M_2



Case study IV: Maxima Set

Pseudocode:

MaximaSet(S,n):
    if $n = 1$ **then**
        return S
    Let p be the median point in S, by x -coordinates
    Let L be the set of points less than p in S by x -coordinates
    Let G be the set of points greater than or equal to p in S by x -coordinates
    $M_1 \leftarrow$ MaximaSet(L)
    $M_2 \leftarrow$ MaximaSet(G)
    Let q be the smallest point in M_2
    for each point, r, in M_1 **do** by x -coordinates
        if $x(r) \leq x(q)$ **and** $y(r) \leq y(q)$ **then**
            Remove r from M_1
    return $M_1 \cup M_2$

Case study IV: Maxima Set

Pseudocode:

MaximaSet(S,n):
  if $n = 1$ **then**
    return S
  Let p be the median point in S, by x -coordinates
  Let L be the set of points less than p in S by x -coordinates
  Let G be the set of points greater than or equal to p in S by x -coordinates
  $M_1 \leftarrow$ MaximaSet(L)
  $M_2 \leftarrow$ MaximaSet(G)
  Let q be the smallest point in M_2
  for each point, r, in M_1 **do** by x -coordinates
    if $x(r) \le x(q)$ **and** $y(r) \le y(q)$ **then**
      Remove r from M_1
  return $M_1 \cup M_2$

$$\text{Running time is } T(n) = 2T(n/2) + T_{\text{media}}(n) + T_{\min}(n) + \Theta(n)$$
$$= 2T(n/2) + T_{\text{media}}(n) + \Theta(n)$$

Case study IV: Maxima Set

Pseudocode:

M

Next week we will see how to find the median in $\Theta(n)$ time!
This fact will yield Θ(nlog n) for Maxima Set.

rdinates

in S by x -coordinates

$M_2 \leftarrow$ MaximaSet(G)
Let q be the smallest point in M_2
for each point, r, in M_1 **do** by x -coordinates
  if $x(r) \leq x(q)$ **and** $y(r) \leq y(q)$ **then**
    Remove r from M_1
return $M_1 \cup M_2$

$$\text{Running time is } T(n) = 2T(n/2) + T_{\text{media}}(n) + T_{\min}(n) + \Theta(n)$$
$$= 2T(n/2) + T_{\text{media}}(n) + \Theta(n)$$