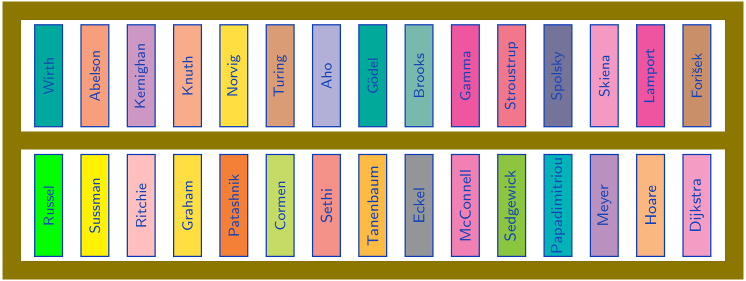
ITMO UNIVERSITY

How to Win Coding Competitions: Secrets of Champions

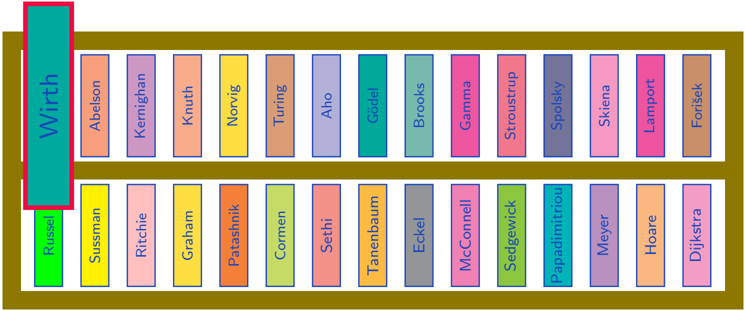
Week 3: Sorting and Search Algorithms
Lecture 1: Introduction to Sorting

Maxim Buzdalov
Saint Petersburg 2016

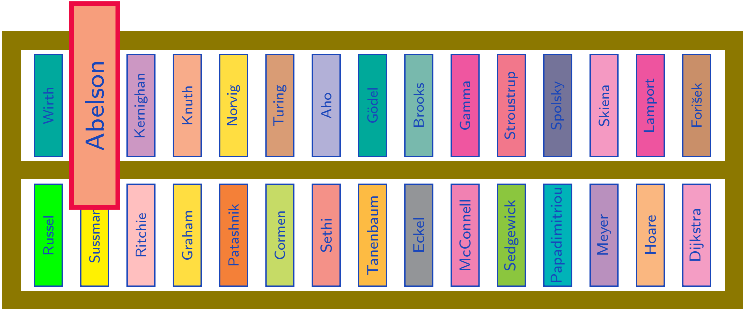
Find Cormen on this bookshelf



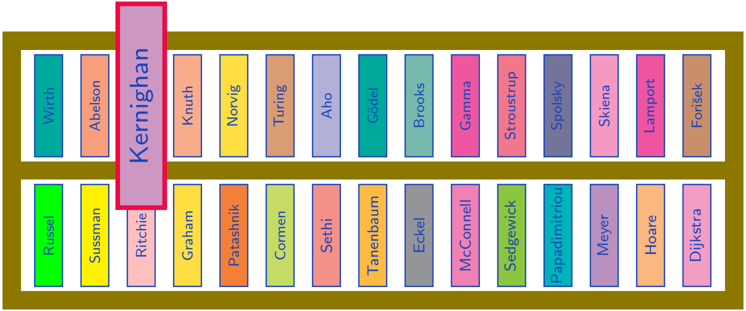
Find Cormen on this bookshelf



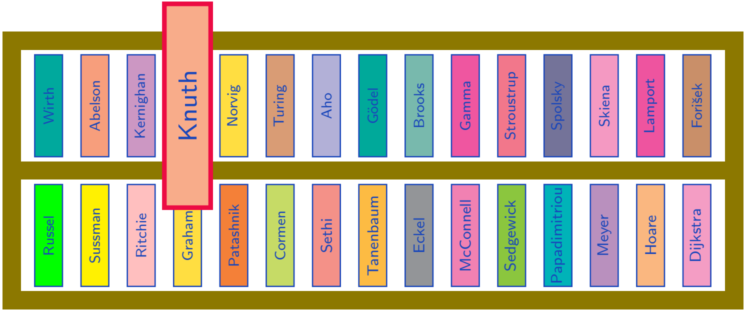
Find Cormen on this bookshelf



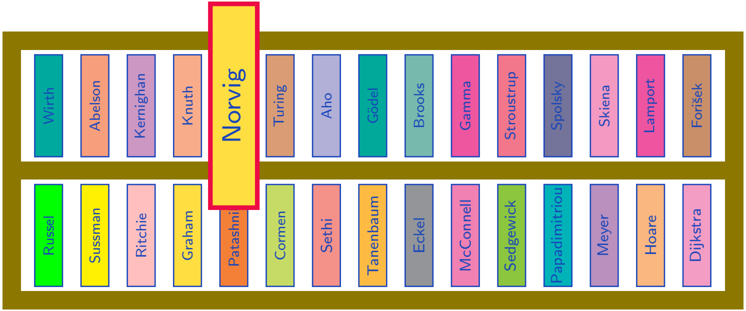
Find Cormen on this bookshelf



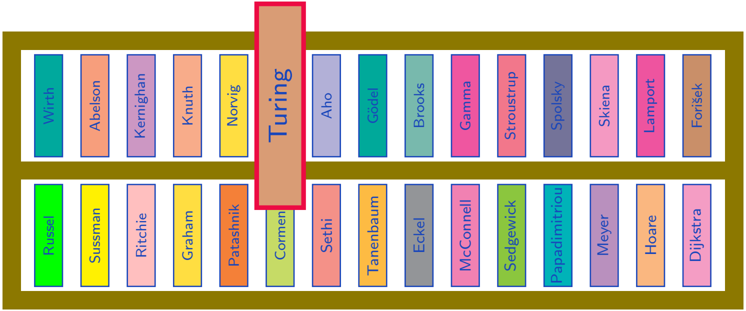
Find Cormen on this bookshelf



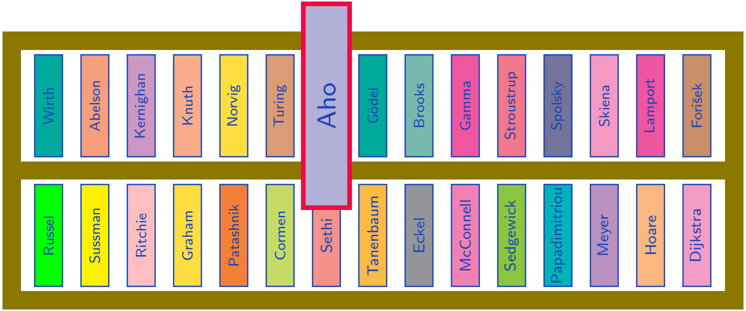
Find Cormen on this bookshelf



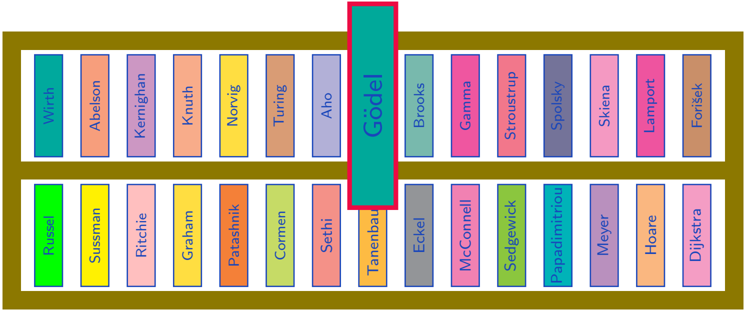
Find Cormen on this bookshelf



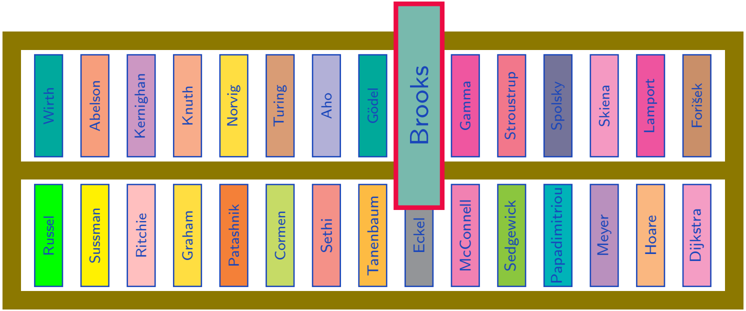
Find Cormen on this bookshelf



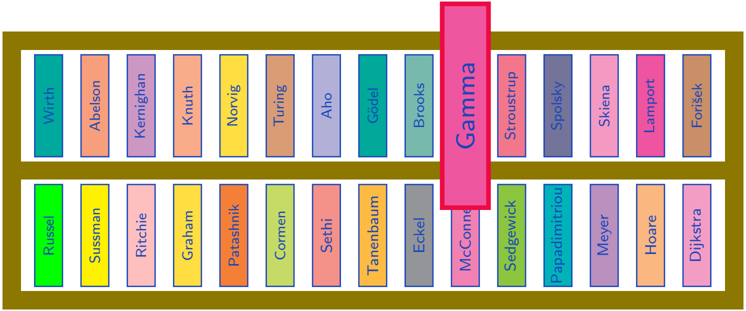
Find Cormen on this bookshelf



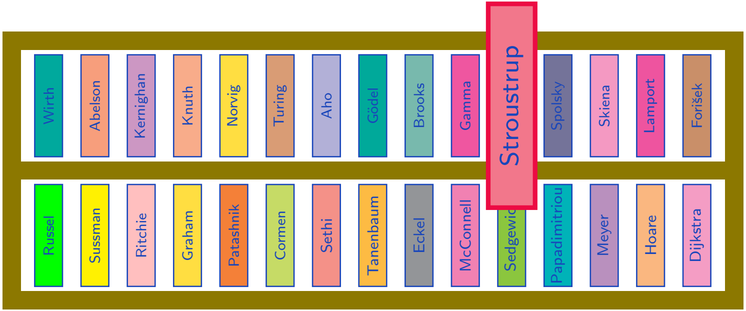
Find Cormen on this bookshelf



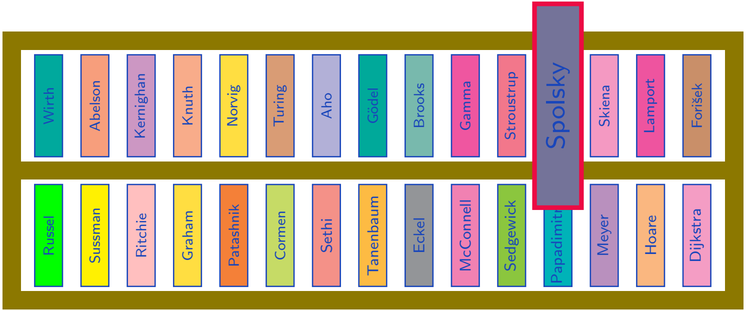
Find Cormen on this bookshelf



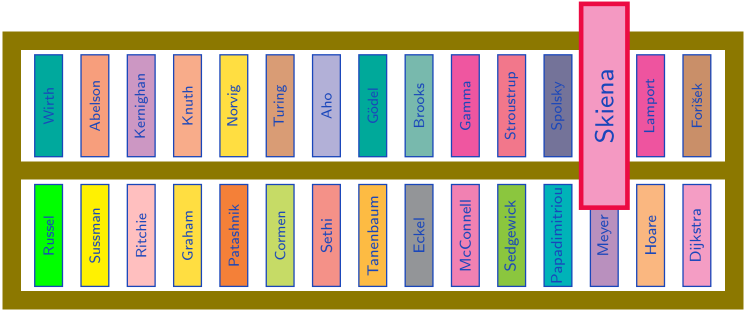
Find Cormen on this bookshelf



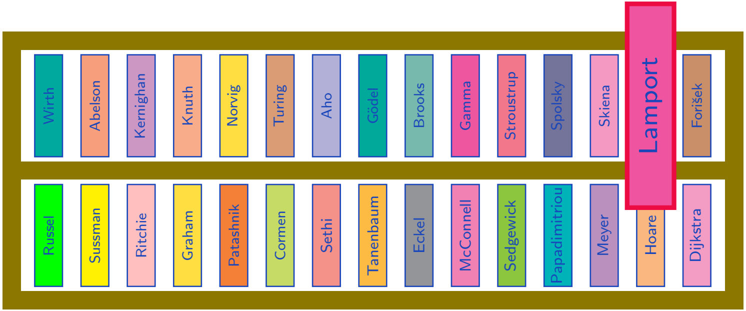
Find Cormen on this bookshelf



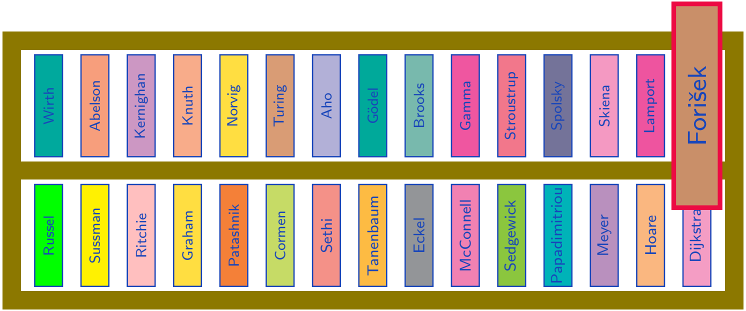
Find Cormen on this bookshelf



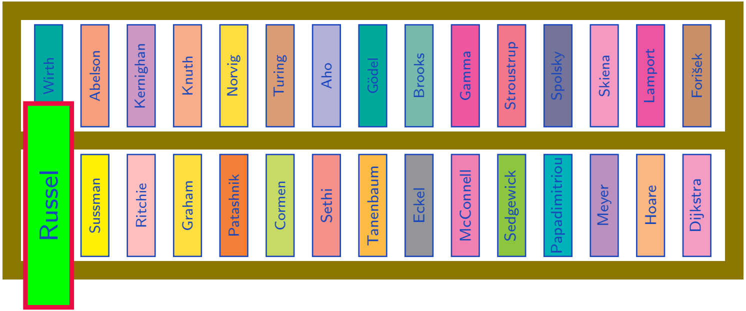
Find Cormen on this bookshelf



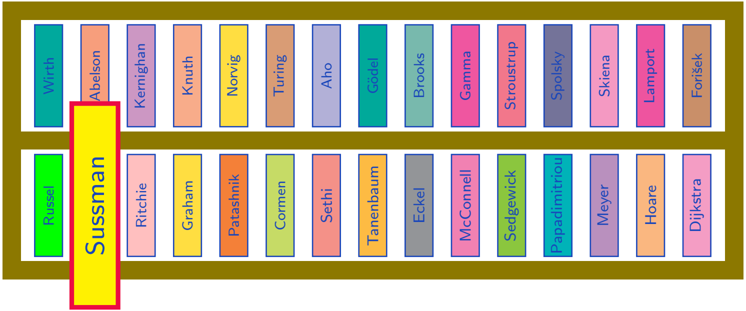
Find Cormen on this bookshelf



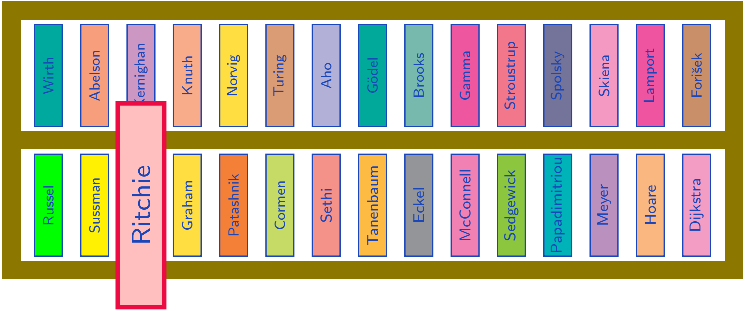
Find Cormen on this bookshelf



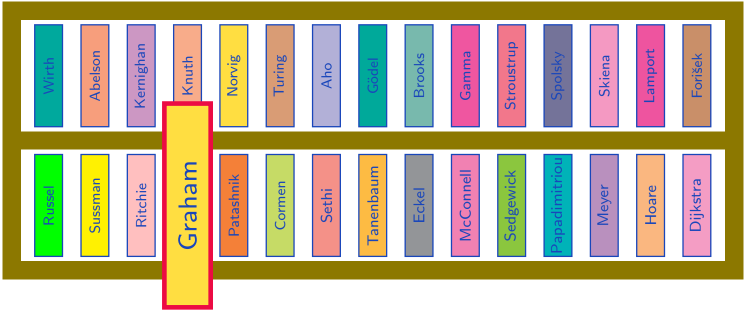
Find Cormen on this bookshelf



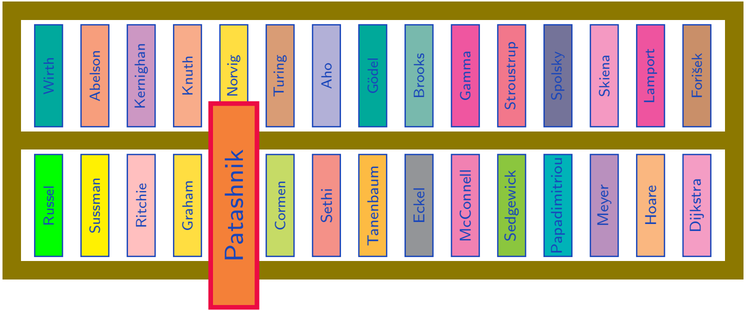
Find Cormen on this bookshelf



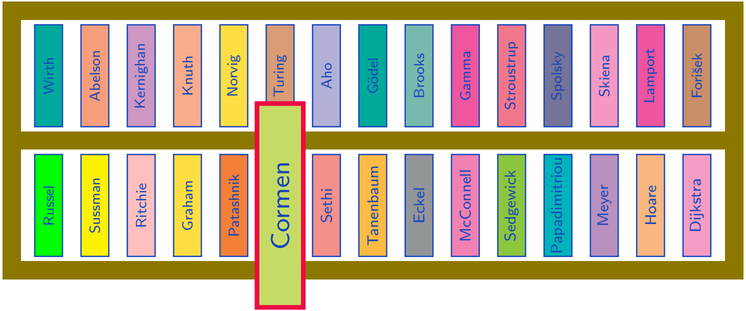
Find Cormen on this bookshelf



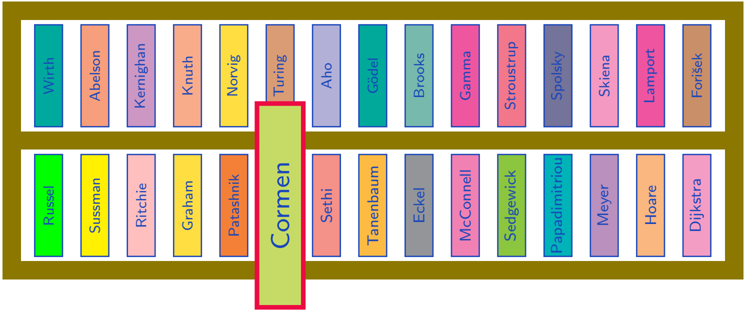
Find Cormen on this bookshelf



Find Cormen on this bookshelf

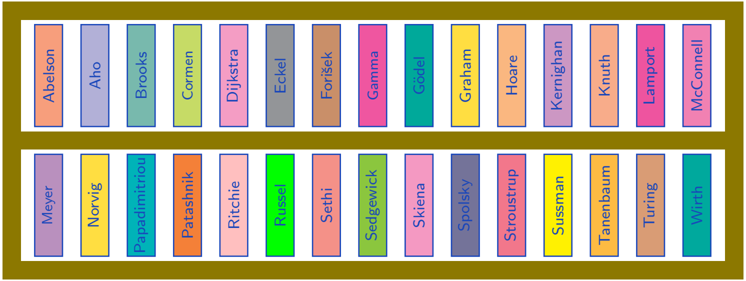


Find Cormen on this bookshelf

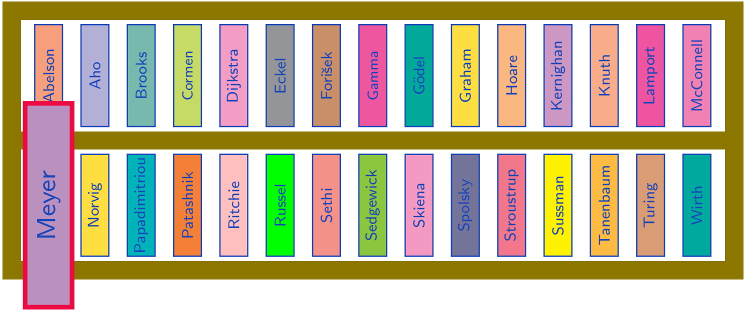


Too slow to be practical :/

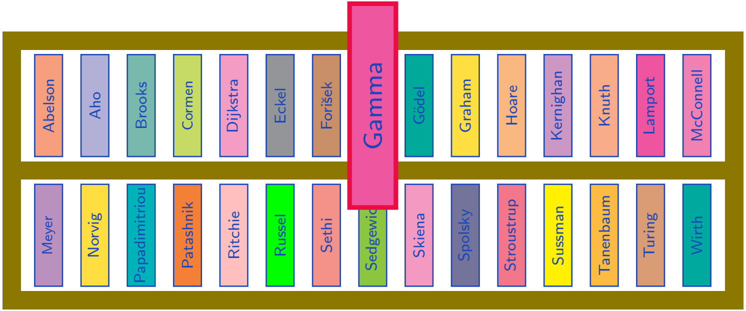
Find Cormen when books are in alphabetical order



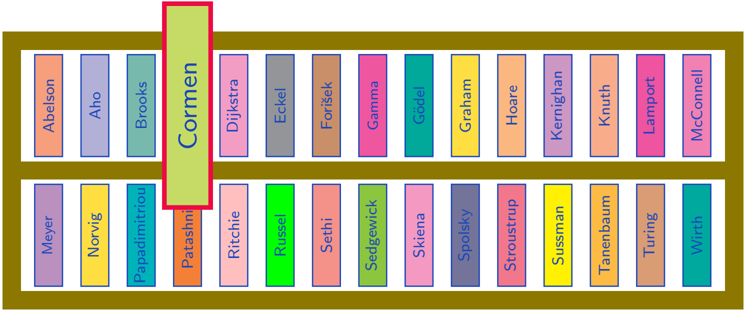
Find Cormen when books are in alphabetical order



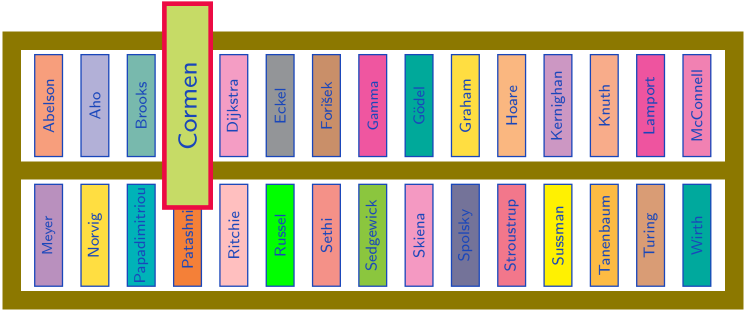
Find Cormen when books are in alphabetical order



Find Cormen when books are in alphabetical order



Find Cormen when books are in alphabetical order



Much faster if sorted! :)

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \to \{\texttt{false}, \texttt{true}\}$

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \to \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \rightarrow \{\mathtt{false}, \mathtt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \rightarrow \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$
- Find a permutation $P = [p_1, p_2, \ldots, p_N]$, such that:
  - for all $i \in [1; N-1]$ it holds that $A_{p_i} \preceq A_{p_{i+1}}$

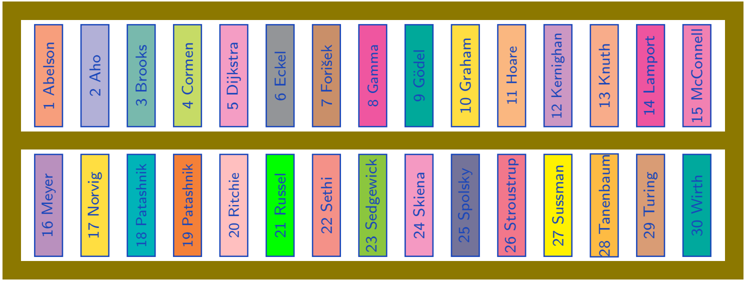
- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \rightarrow \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$
- Find a permutation $P = [p_1, p_2, \ldots, p_N]$, such that:
  - for all $i \in [1; N-1]$ it holds that $A_{p_i} \preceq A_{p_{i+1}}$
- In other words, construct a sequence $B_1, B_2, \ldots B_N$, such that:
  - Every B_i has exactly one corresponding A_j, and vice versa
  - $B_i \preceq B_{i+1}$

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \to \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$
- Find a permutation $P = [p_1, p_2, \ldots, p_N]$, such that:
  - for all $i \in [1; N-1]$ it holds that $A_{p_i} \preceq A_{p_{i+1}}$
- In other words, construct a sequence $B_1, B_2, \ldots B_N$, such that:
  - Every B_i has exactly one corresponding A_j, and vice versa
  - $B_i \preceq B_{i+1}$
- Note:
  - Strict ordering: $A \prec B := A \preceq B$ and $A \neq B$

- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \to \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$
- Find a permutation $P = [p_1, p_2, \ldots, p_N]$, such that:
  - for all $i \in [1; N-1]$ it holds that $A_{p_i} \preceq A_{p_{i+1}}$
- In other words, construct a sequence $B_1, B_2, \ldots B_N$, such that:
  - Every B_i has exactly one corresponding A_j, and vice versa
  - $B_i \preceq B_{i+1}$
- Note:
  - Strict ordering: $A \prec B := A \preceq B$ and $A \neq B$
  - For numbers $(\mathbb{S} = \mathbb{Z}, \mathbb{R}, \ldots)$, $\preceq$ is often $\leq$

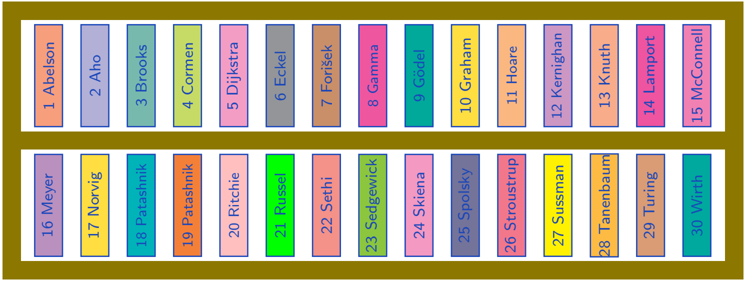
- Given a set $\mathbb{S}$ with total ordering $\preceq : \mathbb{S} \times \mathbb{S} \to \{\texttt{false}, \texttt{true}\}$
  - Reflexive: $a \preceq a$
  - Antisymmetric: $(a \preceq b), (b \preceq a) \Rightarrow (a = b)$
  - Transitive: $(a \preceq b), (b \preceq c) \Rightarrow (a \preceq c)$
  - Total: if not $(a \preceq b)$ then $(b \preceq a)$
- Given a sequence of N items $A_1, A_2, \ldots, A_N$, each from the set $\mathbb{S}$
- Find a permutation $P = [p_1, p_2, \ldots, p_N]$, such that:
  - for all $i \in [1; N-1]$ it holds that $A_{p_i} \preceq A_{p_{i+1}}$
- In other words, construct a sequence $B_1, B_2, \ldots B_N$, such that:
  - Every B_i has exactly one corresponding A_j, and vice versa
  - $B_i \preceq B_{i+1}$
- Note:
  - Strict ordering: $A \prec B := A \preceq B$ and $A \neq B$
  - For numbers $(\mathbb{S} = \mathbb{Z}, \mathbb{R}, \ldots)$, $\preceq$ is often $\leq$
  - We will denote $\preceq$ as $\leq$, and $\prec$ as $<$, in the rest of the week materials

Why do we need sorting? – Prepare to fast query answering



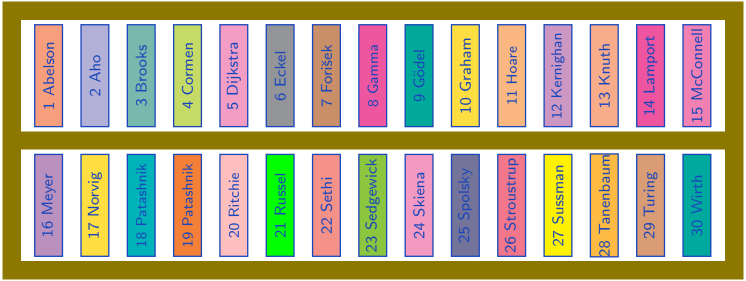
Why do we need sorting? – Prepare to fast query answering

1. Does this book exist?



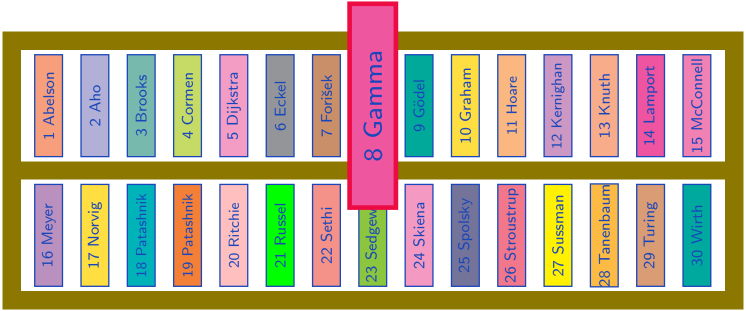
Why do we need sorting? – Prepare to fast query answering

1. Does this book exist? Gamma



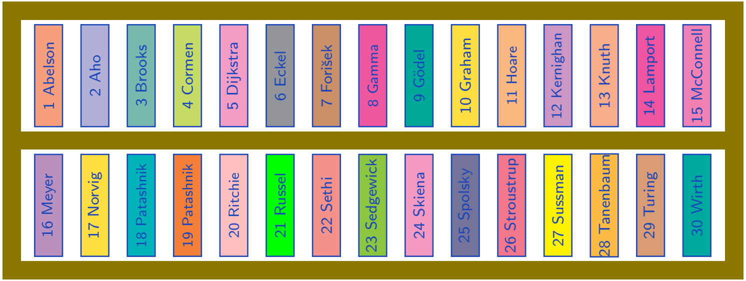
Why do we need sorting? – Prepare to fast query answering

1. Does this book exist? Gamma → YES



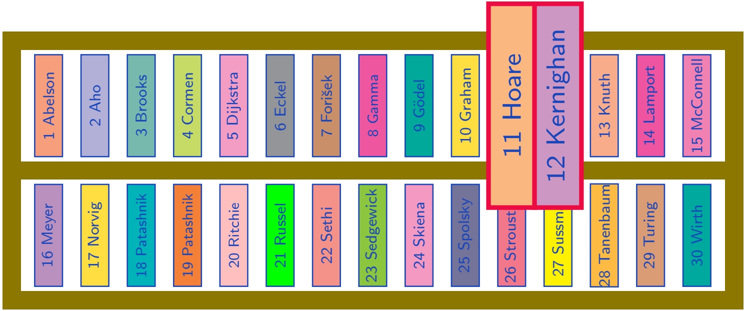
Why do we need sorting? – Prepare to fast query answering

1. Does this book exist? Kant



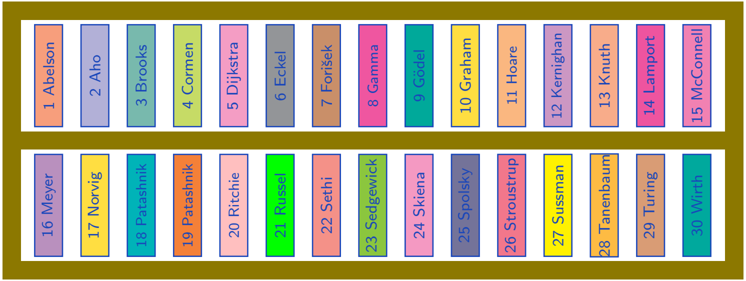
Why do we need sorting? – Prepare to fast query answering

1. Does this book exist? Kant → NO



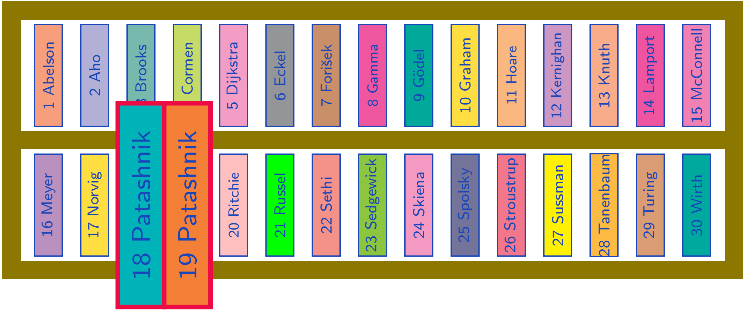
Why do we need sorting? – Prepare to fast query answering

2. How many books of Patashnik is there?



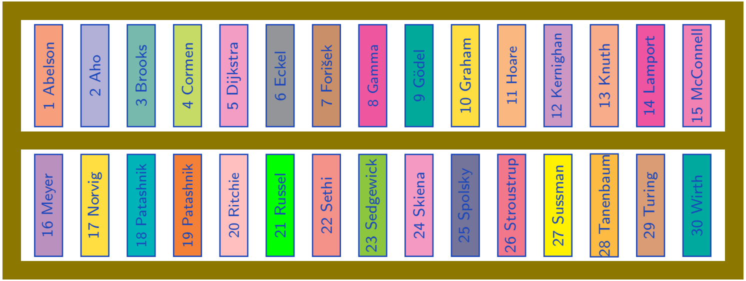
Why do we need sorting? – Prepare to fast query answering

2. How many books of Patashnik is there? Two



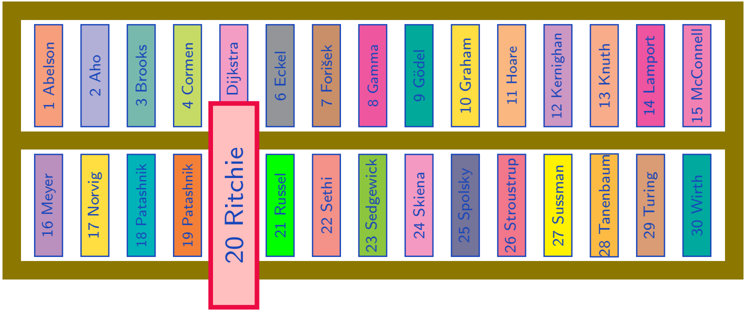
Why do we need sorting? – Prepare to fast query answering

3. How many books have a name smaller than Ritchie?



Why do we need sorting? – Prepare to fast query answering

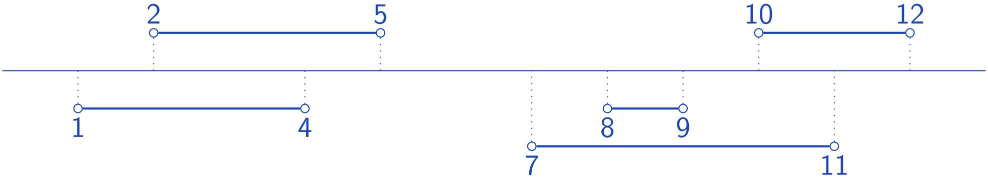
3. How many books have a name smaller than Ritchie? 19 (the index minus 1)



Why do we need sorting? – Prepare to fast data processing

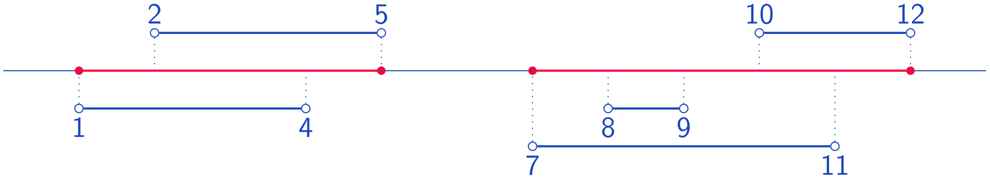
Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied.



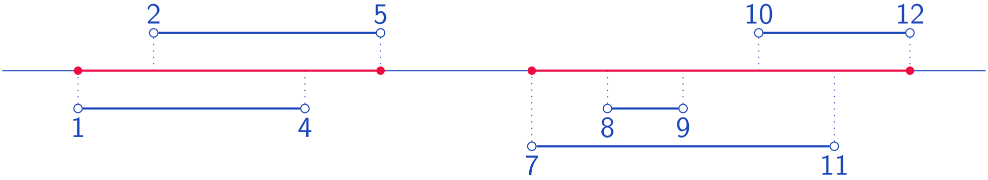
Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted



Why do we need sorting? – Prepare to fast data processing

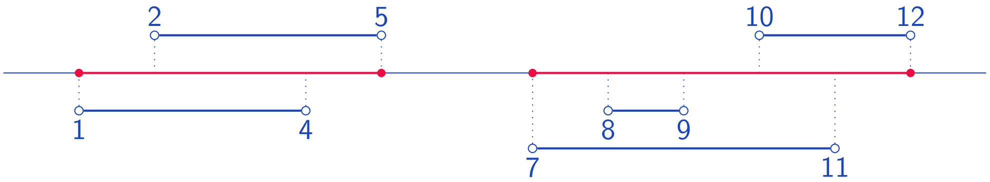
Given segment endpoints where paint was applied. Find which length was painted



Solution:

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

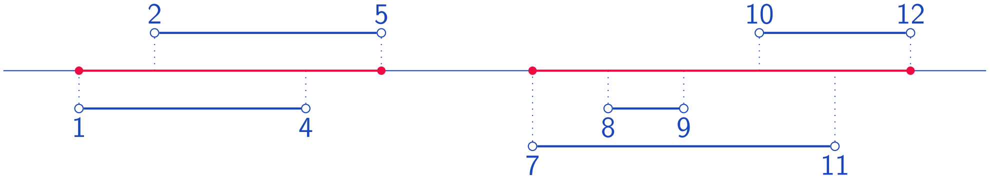


Solution:

- Sort segments by the left coordinate

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

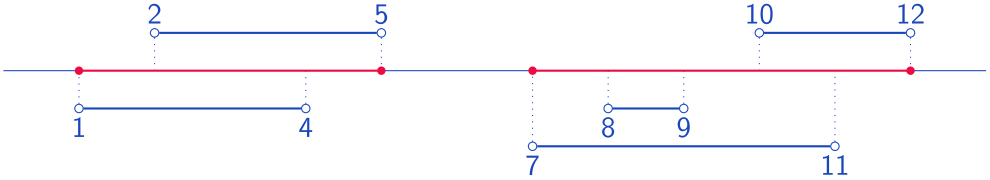


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

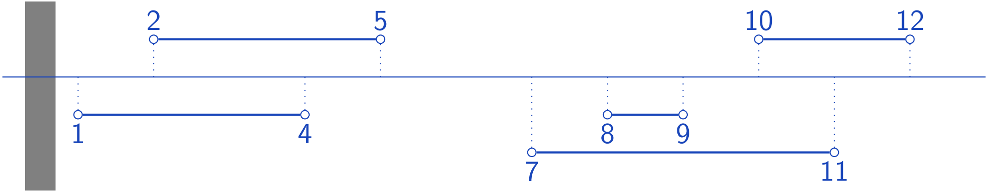


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

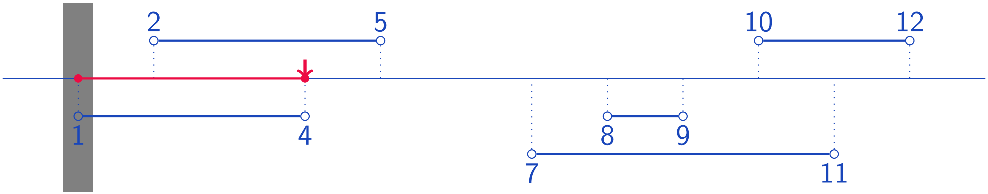


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

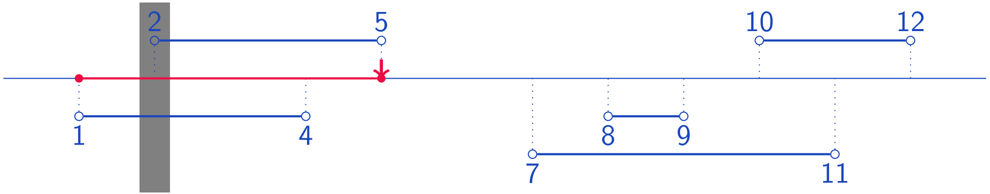
Given segment endpoints where paint was applied. Find which length was painted



Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing
Given segment endpoints where paint was applied. Find which length was painted

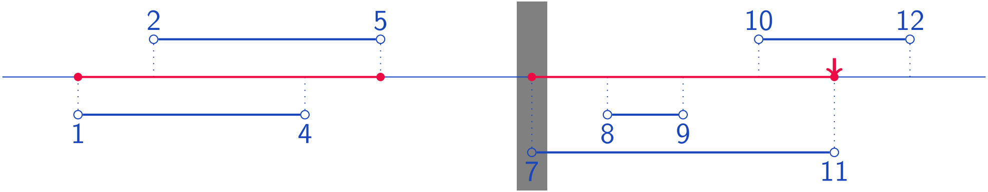


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

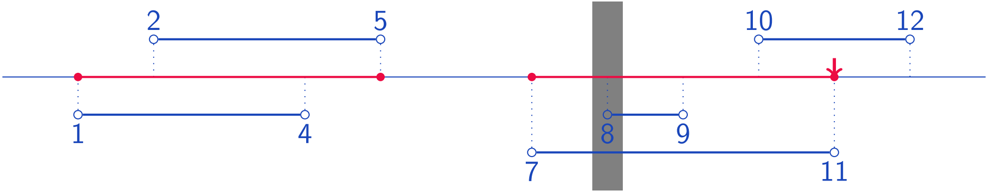


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

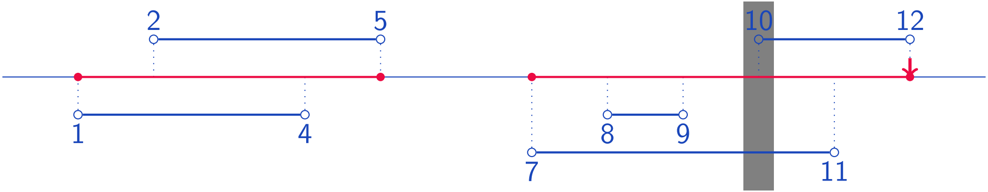


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted

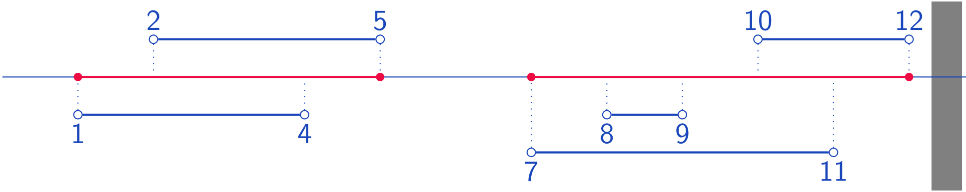


Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Prepare to fast data processing

Given segment endpoints where paint was applied. Find which length was painted



Solution:

- Sort segments by the left coordinate
- Traverse segments in the sorted order
  - Track the endpoint of the current segment cluster
  - Finish the cluster when the next segment is beyond this point

Why do we need sorting? – Construct an optimal solution

7	1	4	6	8	2	9	3	1	4	3	5	9	8	2
3	1	2	8	6	5	4	7	4	9	4	5	1	3	8

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible

7	1	4	6	8	2	9	3	1	4	3	5	9	8	2
3	1	2	8	6	5	4	7	4	9	4	5	1	3	8

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible

Solution:

7	1	4	6	8	2	9	3	1	4	3	5	9	8	2
3	1	2	8	6	5	4	7	4	9	4	5	1	3	8

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible

Solution:

- Sort A in non-decreasing order

1	1	2	2	3	3	4	4	5	6	7	8	8	9	9
3	1	2	8	6	5	4	7	4	9	4	5	1	3	8

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is maximum possible

Solution:

- Sort A in non-decreasing order
- Sort B in non-decreasing order

1	1	2	2	3	3	4	4	5	6	7	8	8	9	9
1	1	2	3	3	4	4	4	5	5	6	7	8	8	9

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \ldots A_N]$ and $B = [B_1, \ldots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is minimum possible

Solution:

- Sort A in non-decreasing order
- Sort B in non-increasing order

1	1	2	2	3	3	4	4	5	6	7	8	8	9	9
9	8	8	7	6	5	5	4	4	4	3	3	2	1	1

Why do we need sorting? – Construct an optimal solution

Example:

- Given two sequences $A = [A_1, \dots A_N]$ and $B = [B_1, \dots B_N]$
- Find permutations P and Q such that $\sum_{i=1}^{N} A_{P_i} \cdot B_{Q_i}$ is minimum possible

Solution:

- Sort A in non-decreasing order
- Sort B in non-increasing order
- These facts will be proven in a this week's video

1	1	2	2	3	3	4	4	5	6	7	8	8	9	9
9	8	8	7	6	5	5	4	4	4	3	3	2	1	1

- Insertion sort: a simple sorting algorithm
- If a sorted sequence yields an optimal result, how to prove it?
- Quick sort: a very fast algorithm
- Merge sort: can never be too slow
- Stable and unstable sorting algorithms
- How to compare various objects
- An $\Omega(n \log n)$ bound on the complexity of comparison-based algorithms
- Bucket sort and radix sort: Linear non-comparison sorting algorithms
- Sorting algorithms in standard libraries