MADALGO

CENTER FOR MASSIVE DATA ALGORITHMICS

PhD Defense

Implicit Data Structures, Sorting, and Text Indexing

Jesper Sindahl Nielsen

Danmarks Grundforskningsfond
Danish National Research Foundation

STATSBIBLIOTEKET

AARHUS UNIVERSITY

Overview

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing

Overview

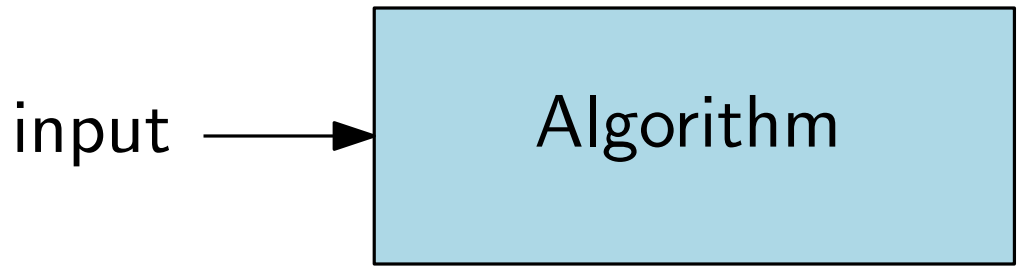
1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing
5. State and University Library

Algorithms

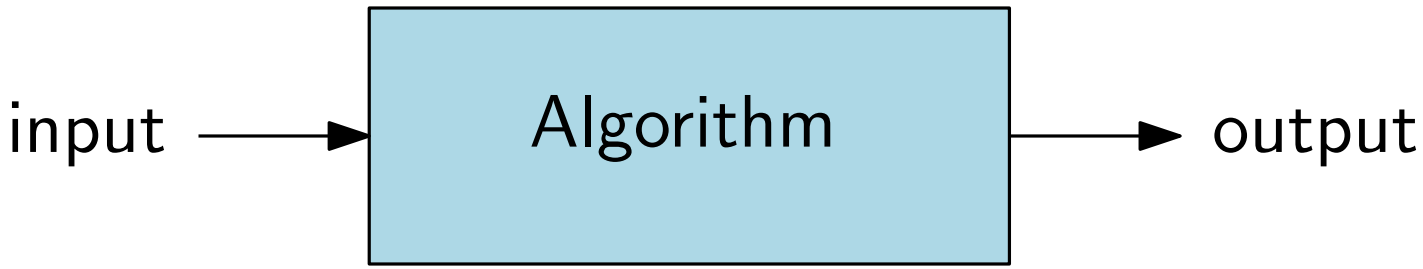
Algorithms

Algorithm

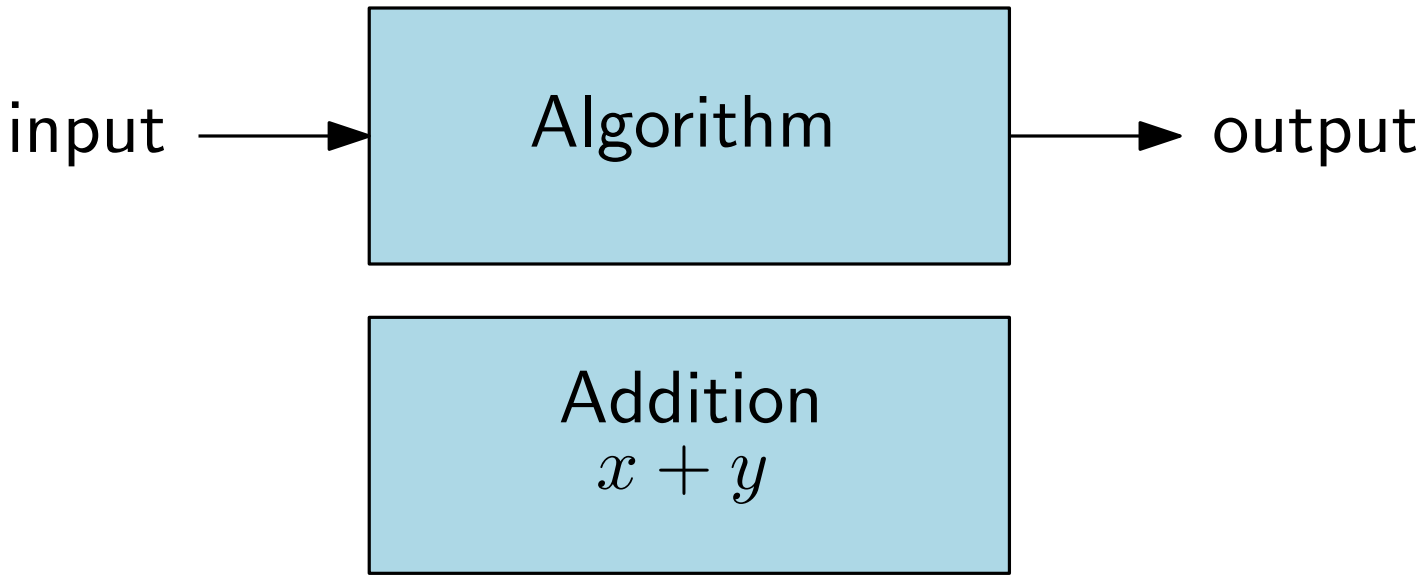
Algorithms



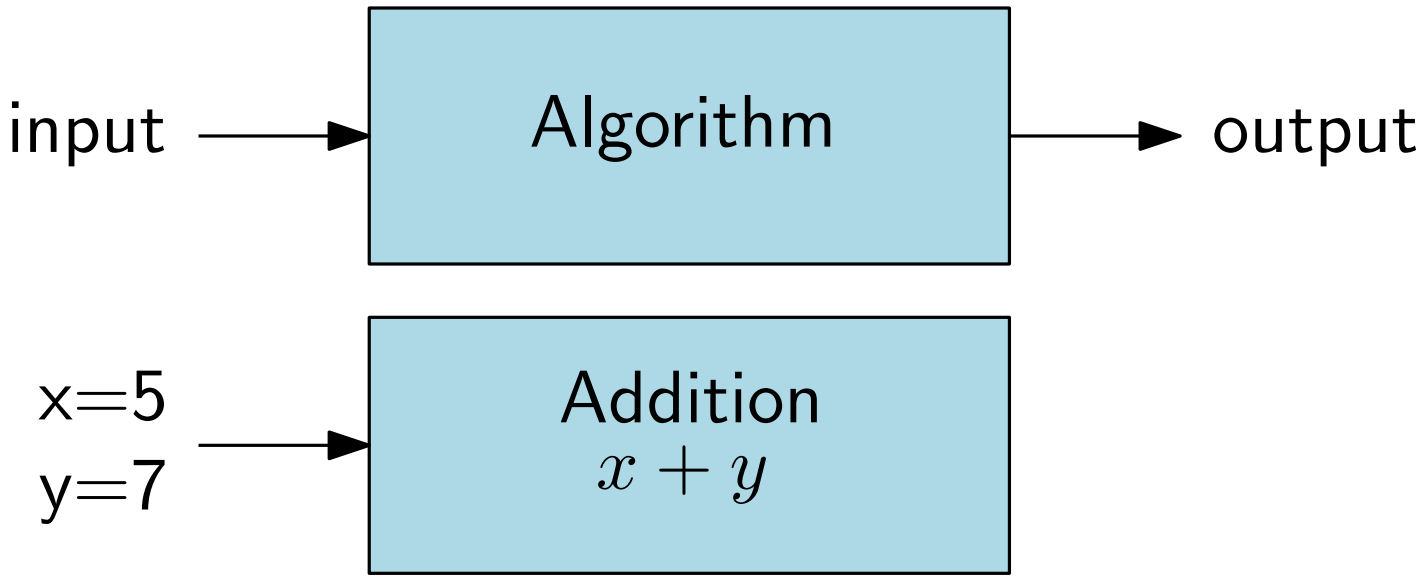
Algorithms



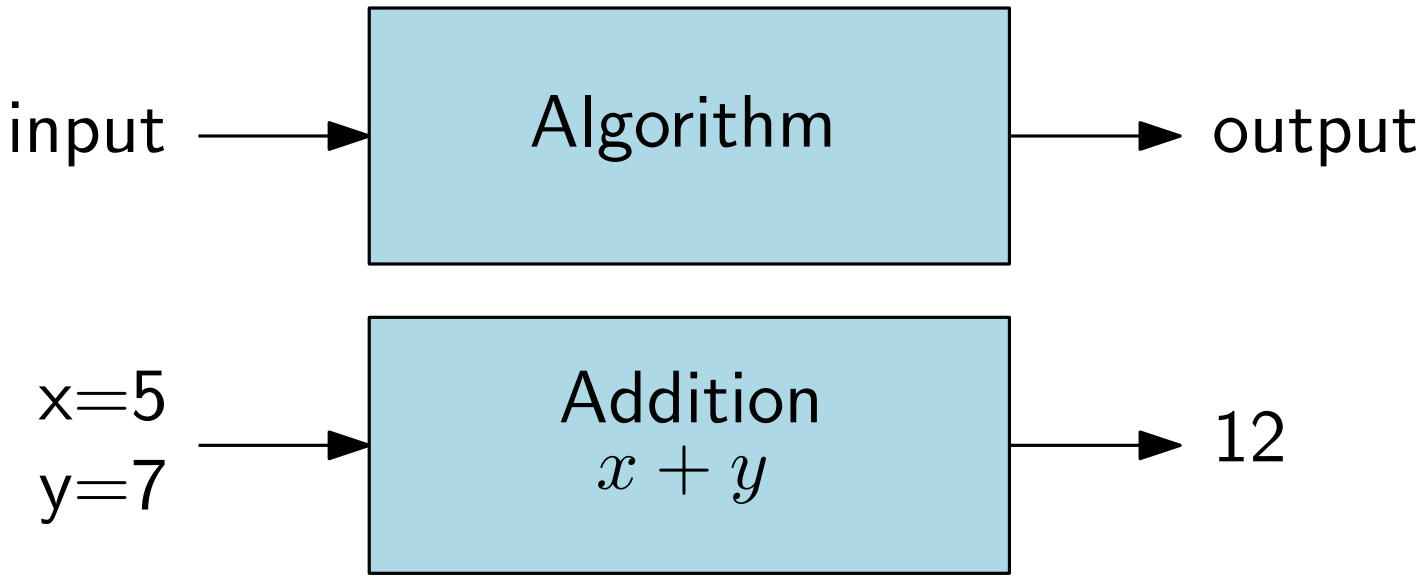
Algorithms



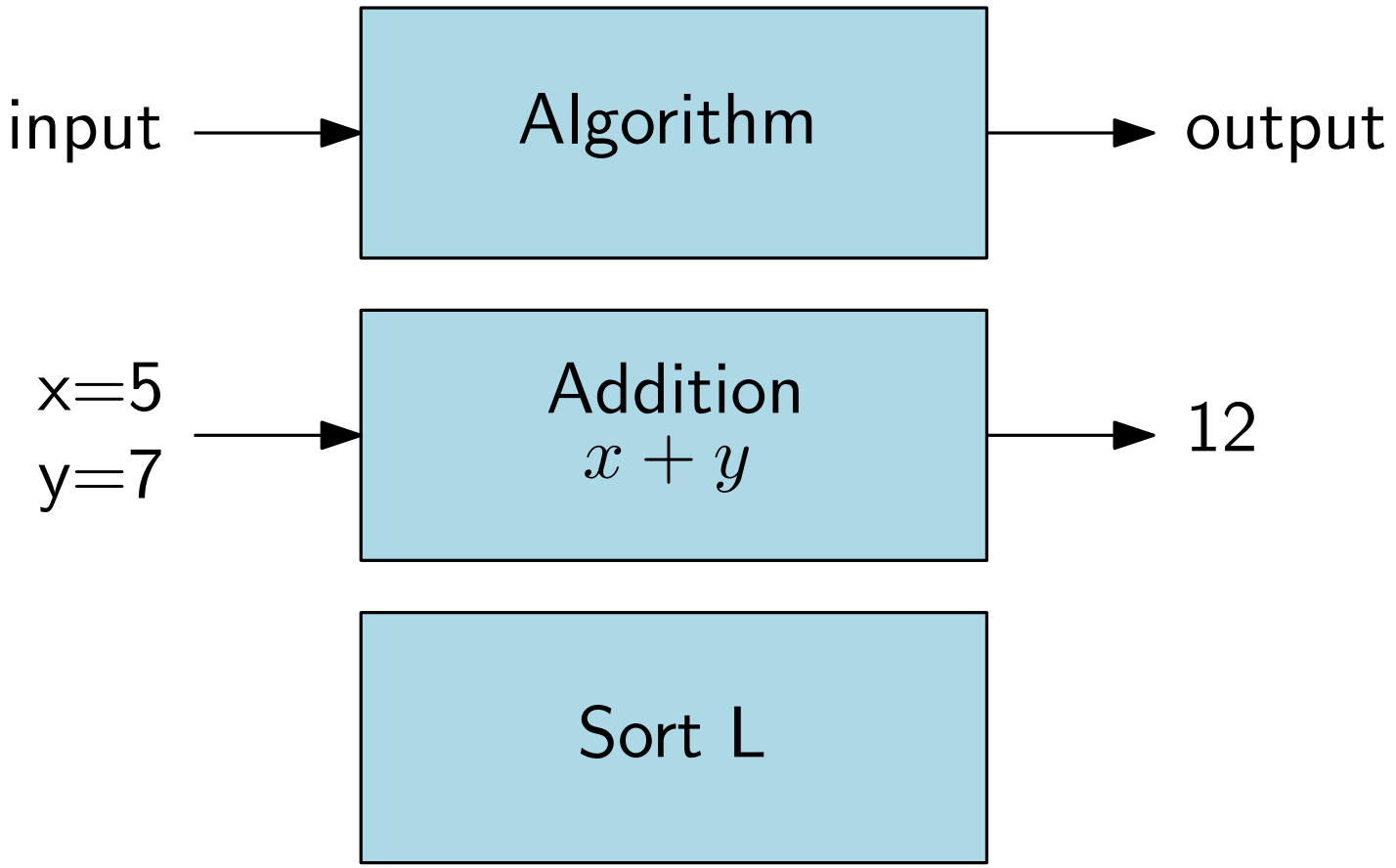
Algorithms



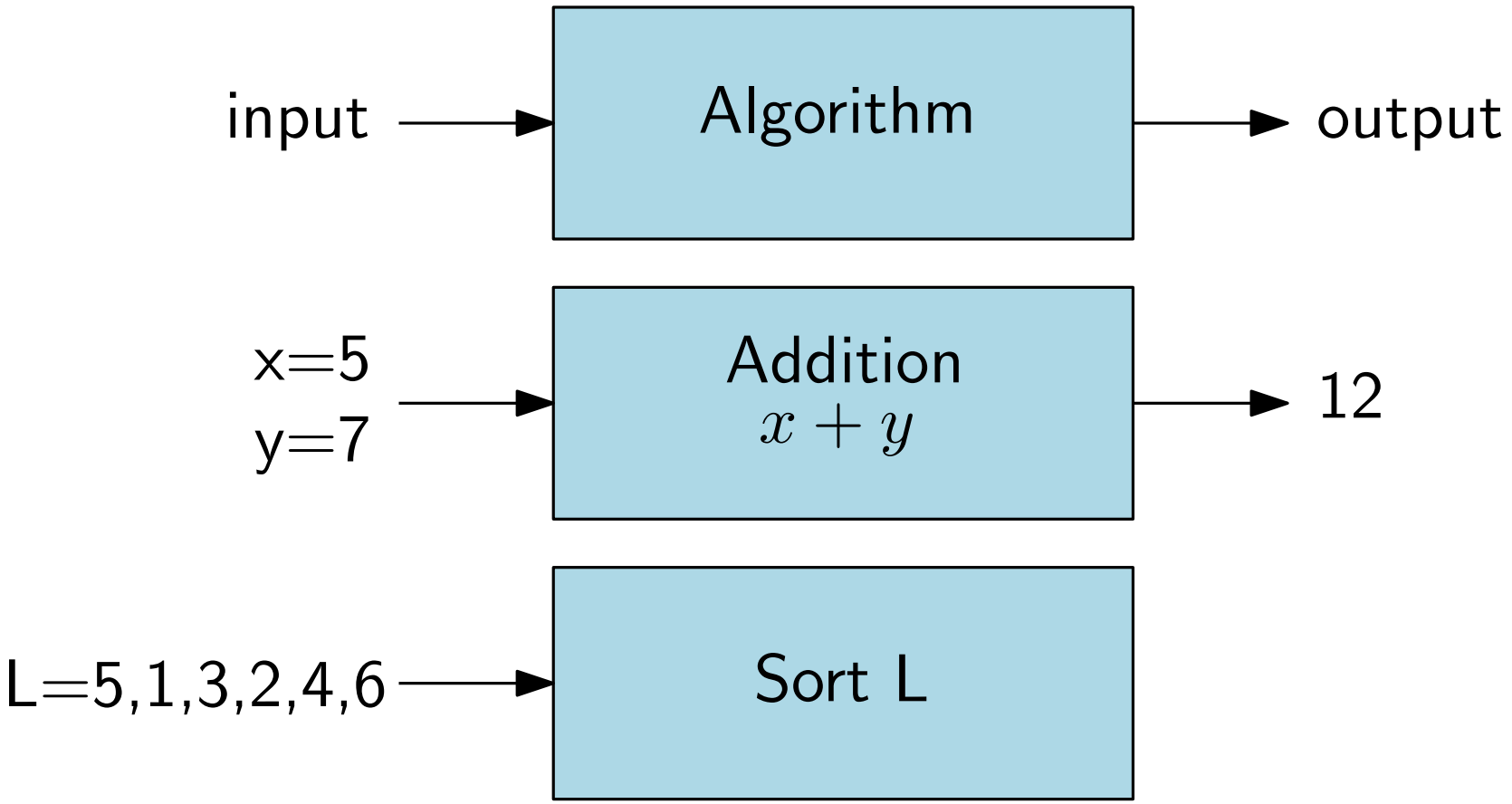
Algorithms



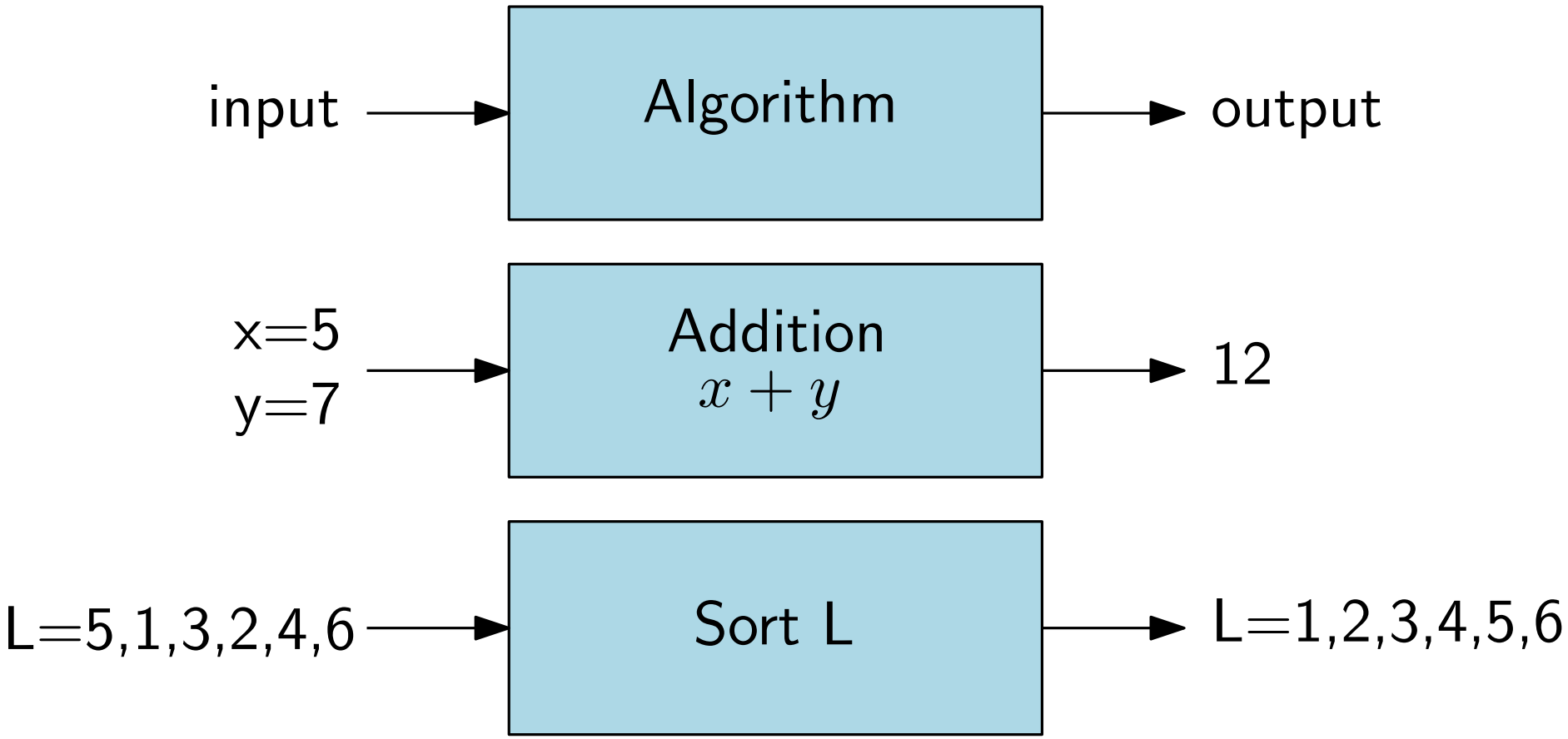
Algorithms



Algorithms



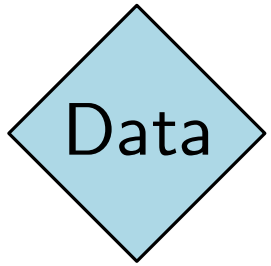
Algorithms



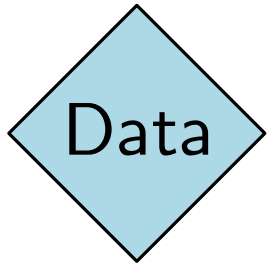
We want to design *fast* algorithms

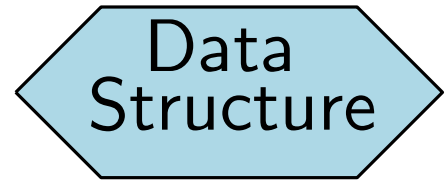
Data Structures

Data Structures

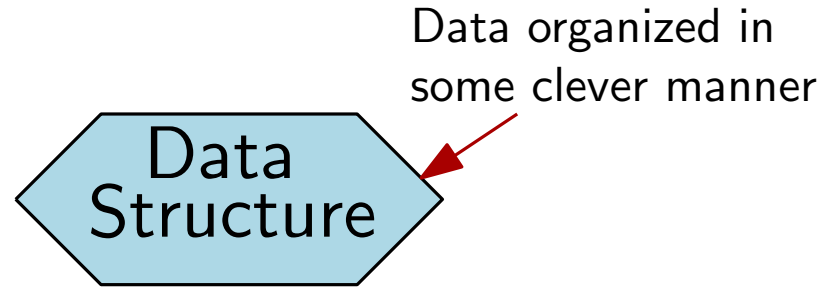


Data Structures

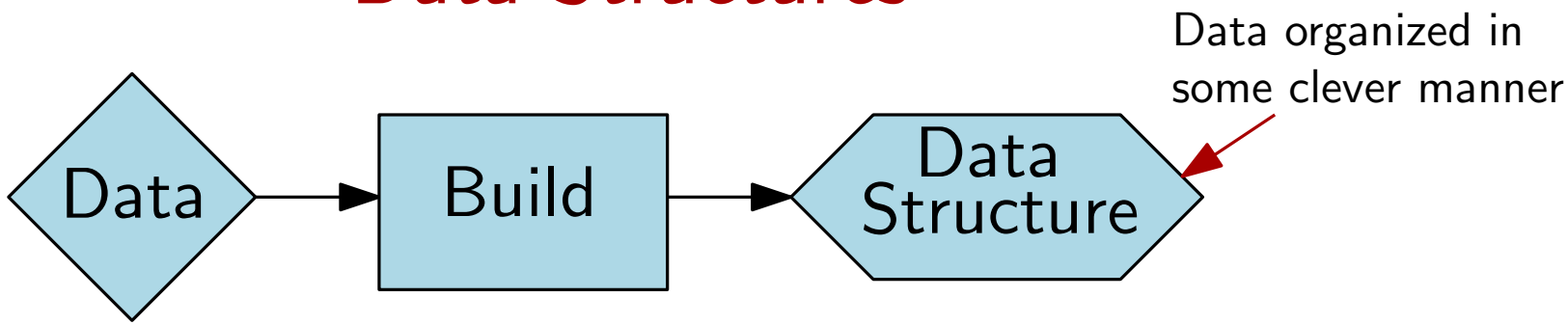




Data Structures

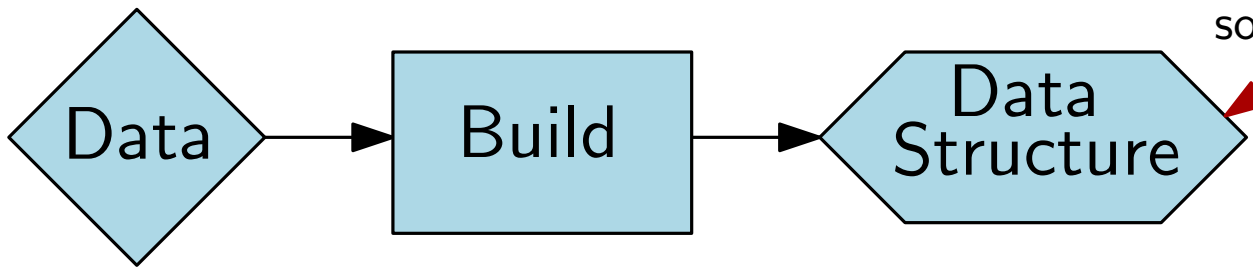


Data Structures



Data Structures

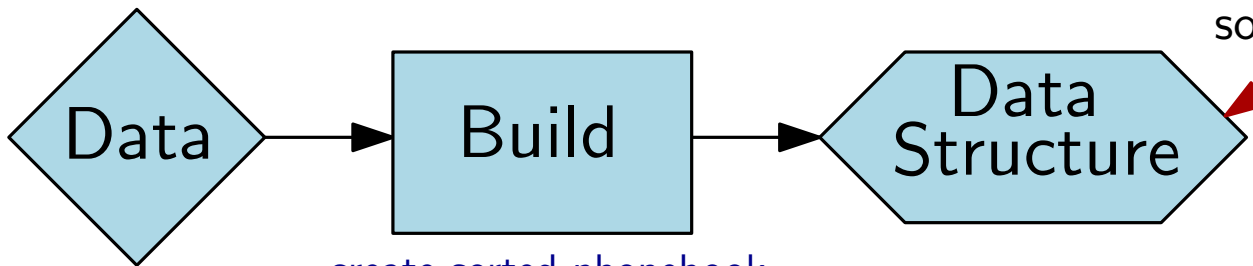
Mom,375874
Dad,278393
Brother,681958
Advisor,359617
Arnold,123456
Sylvester,987654
Jean-Claude,456321
Wesley,789123



Data organized in some clever manner

Data Structures

Mom,375874
Dad,278393
Brother,681958
Advisor,359617
Arnold,123456
Sylvester,987654
Jean-Claude,456321
Wesley,789123

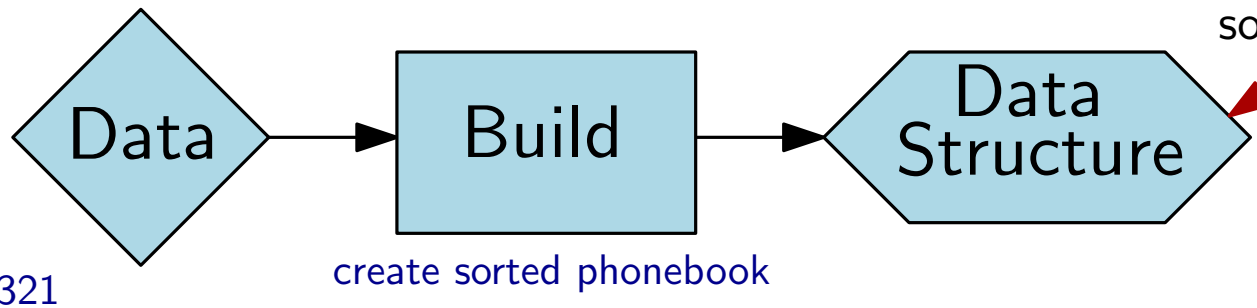


create sorted phonebook

Data organized in some clever manner

Data Structures

Mom,375874
Dad,278393
Brother,681958
Advisor,359617
Arnold,123456
Sylvester,987654
Jean-Claude,456321
Wesley,789123

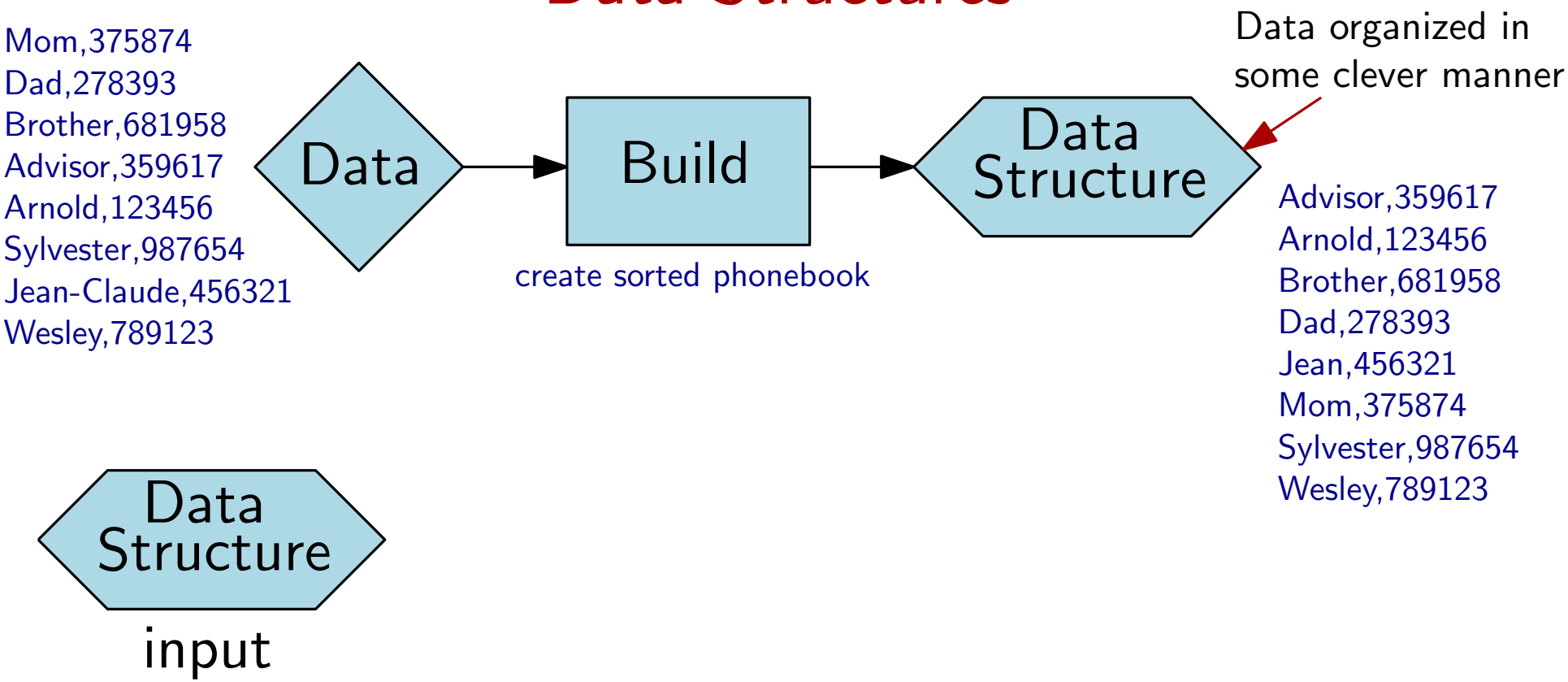


create sorted phonebook

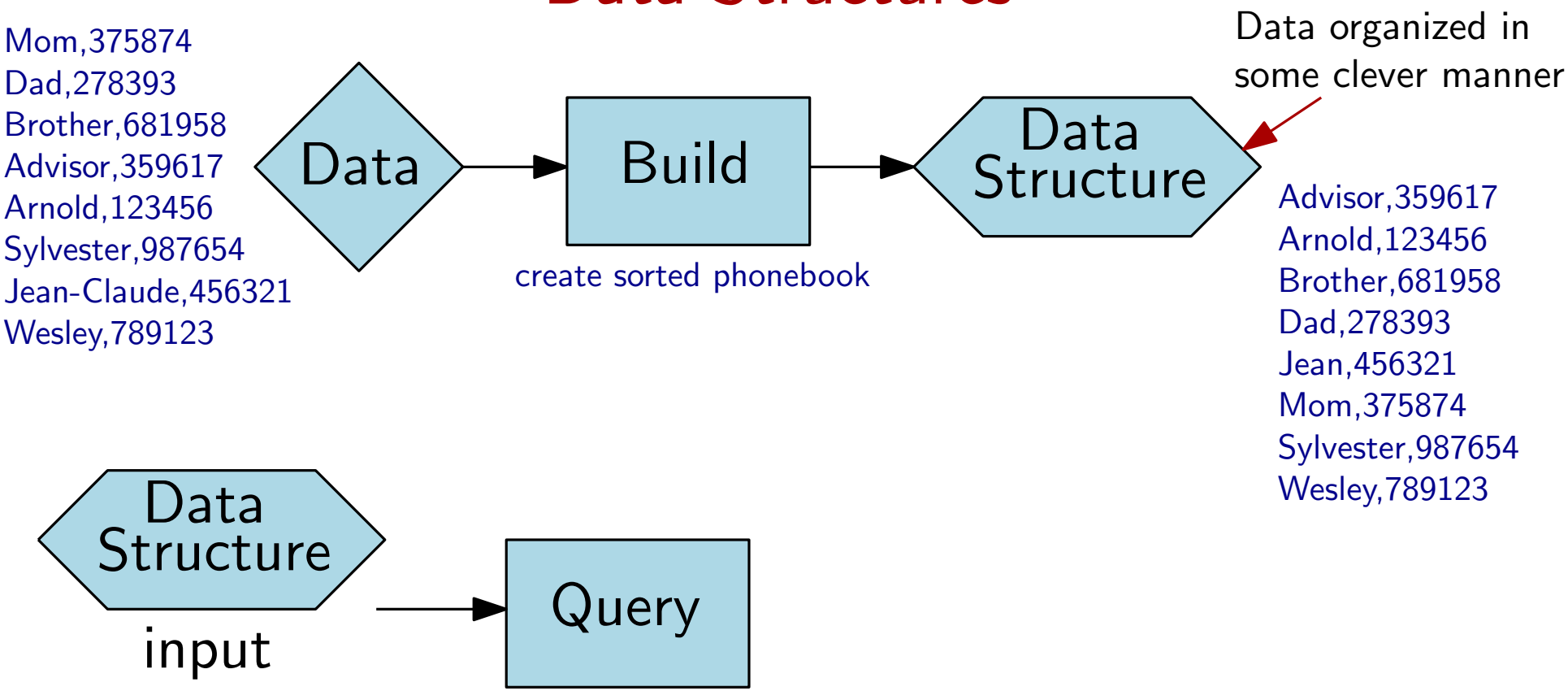
Data organized in some clever manner

Advisor,359617
Arnold,123456
Brother,681958
Dad,278393
Jean,456321
Mom,375874
Sylvester,987654
Wesley,789123

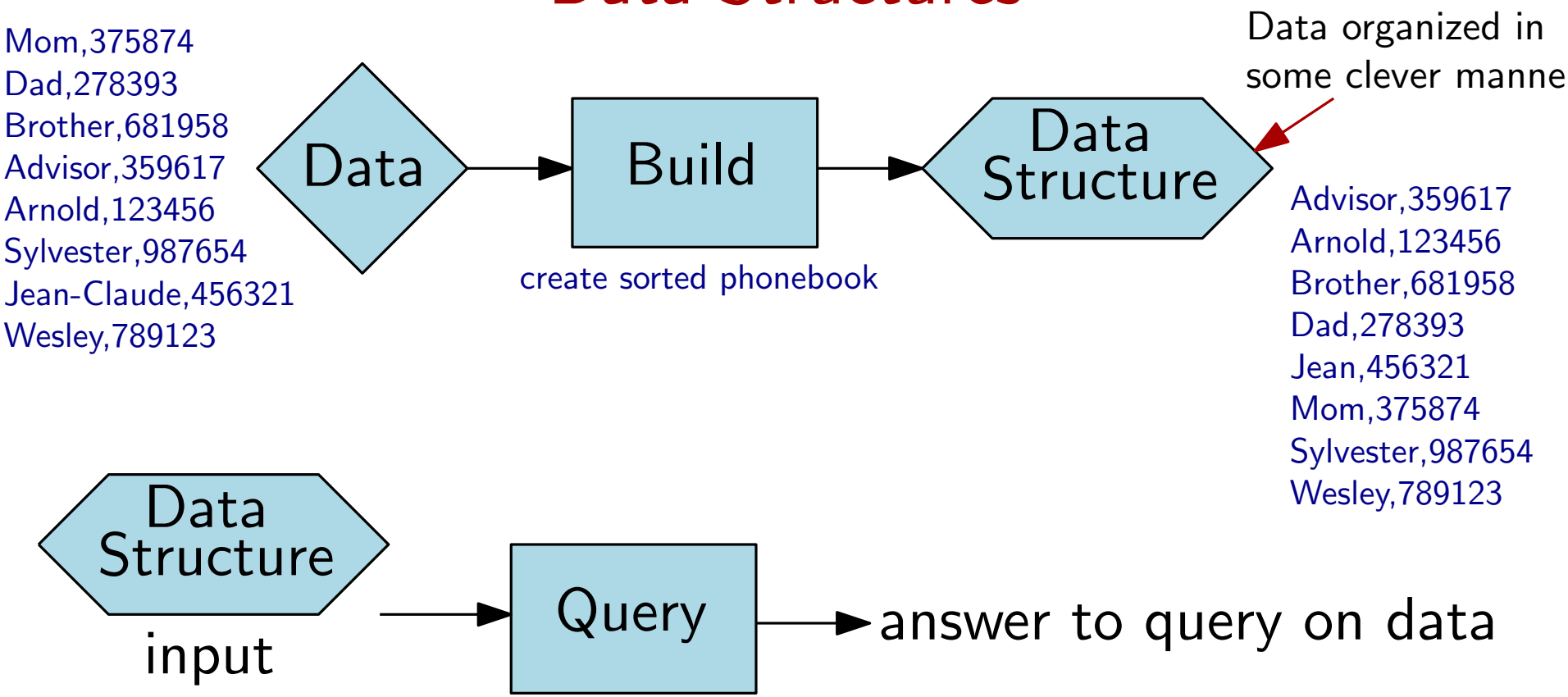
Data Structures



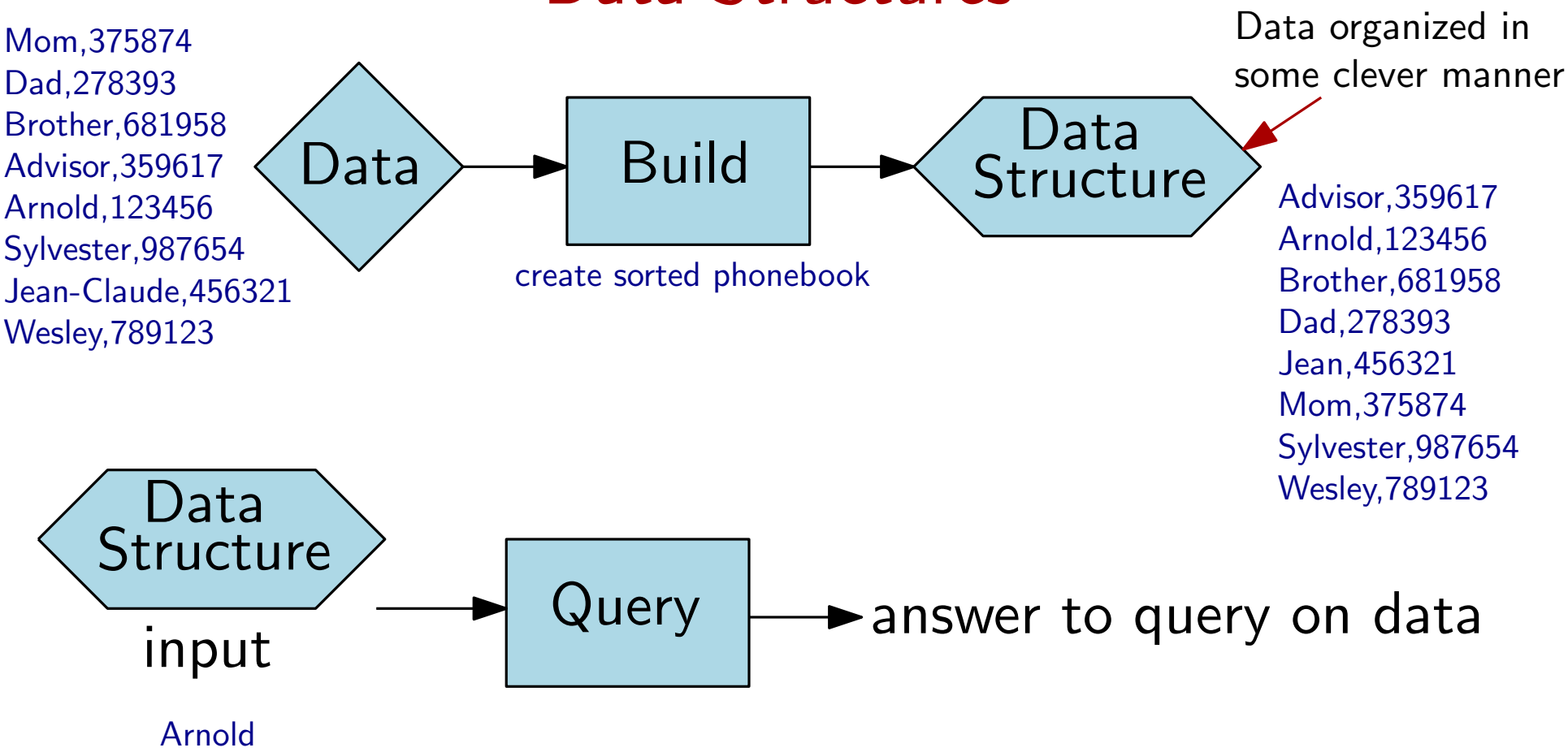
Data Structures



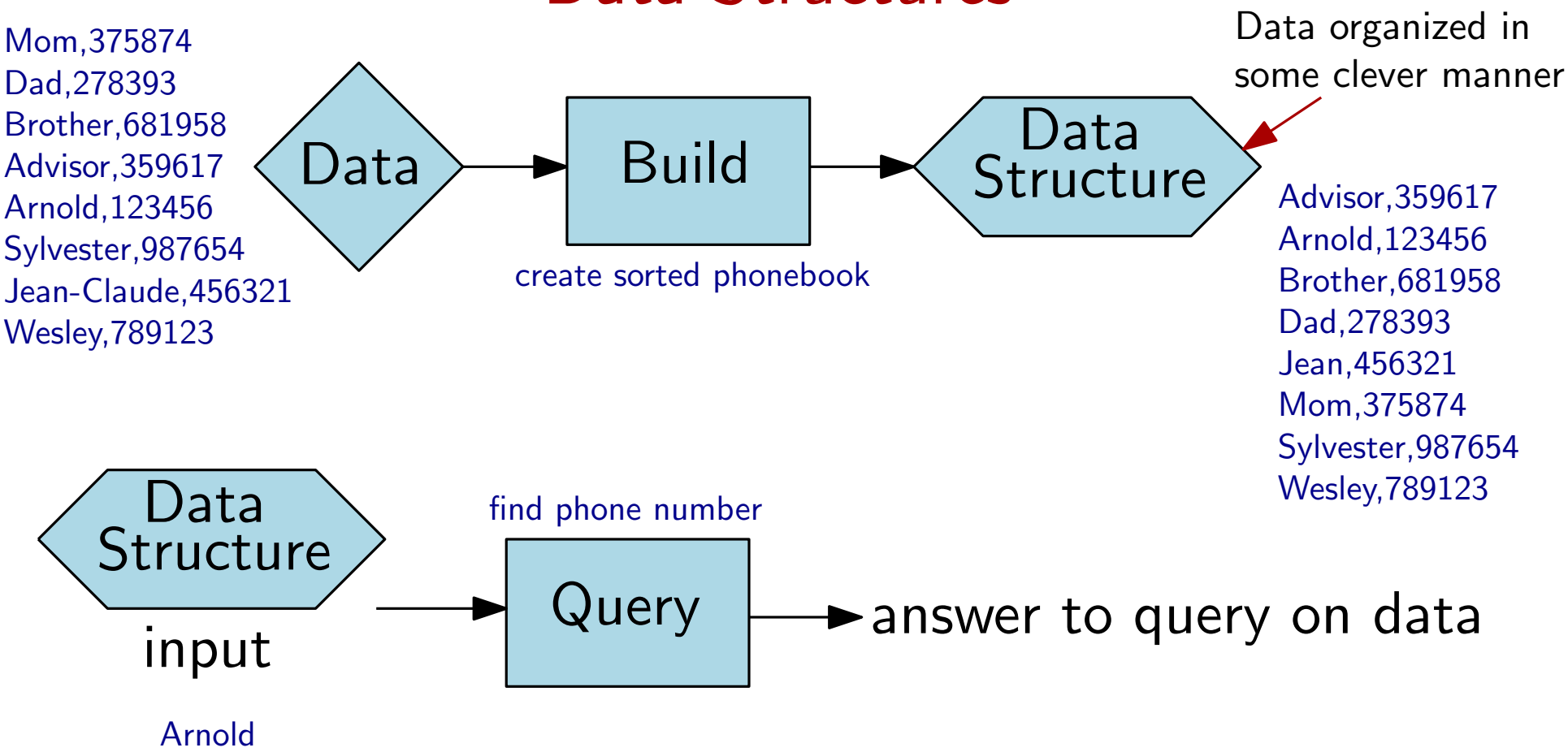
Data Structures



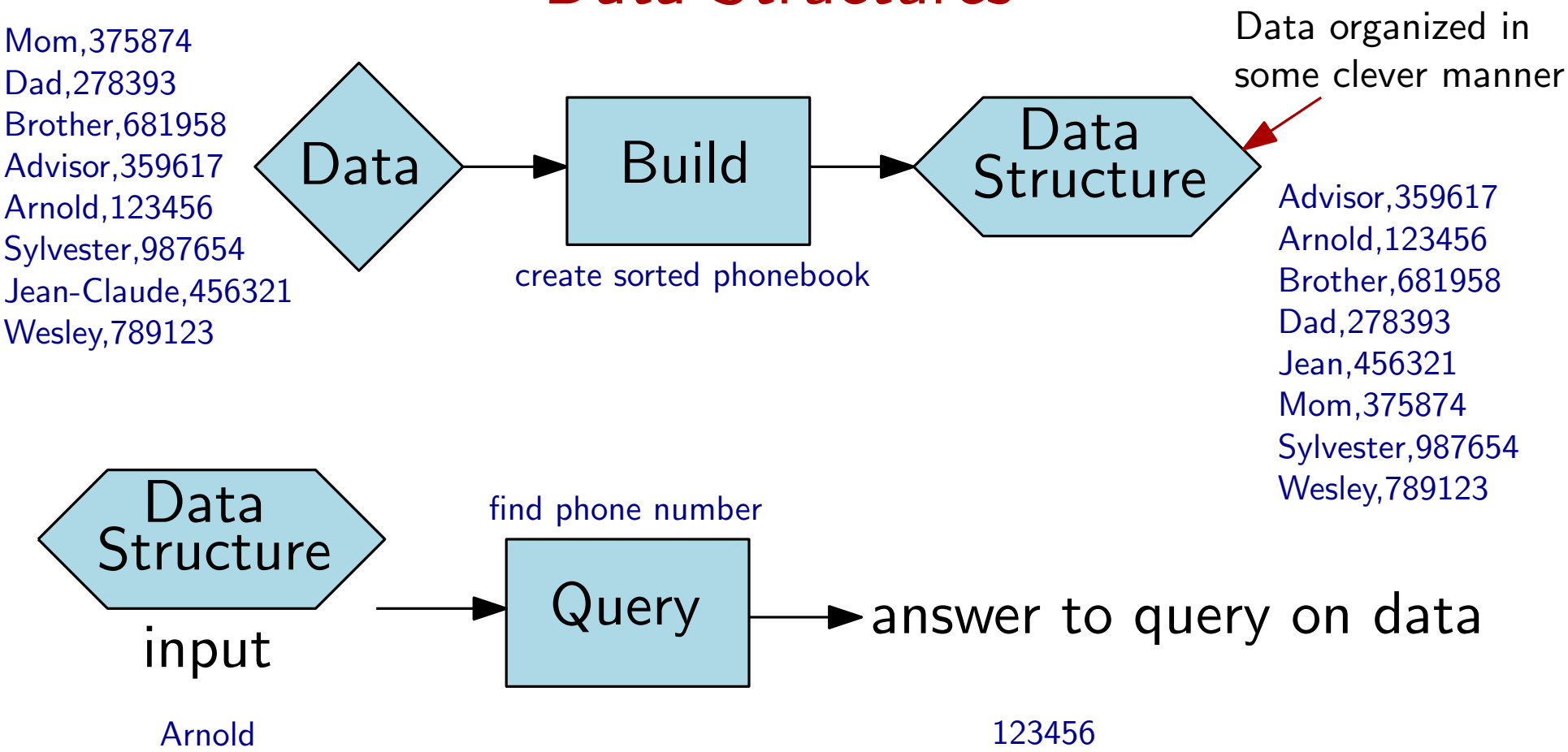
Data Structures



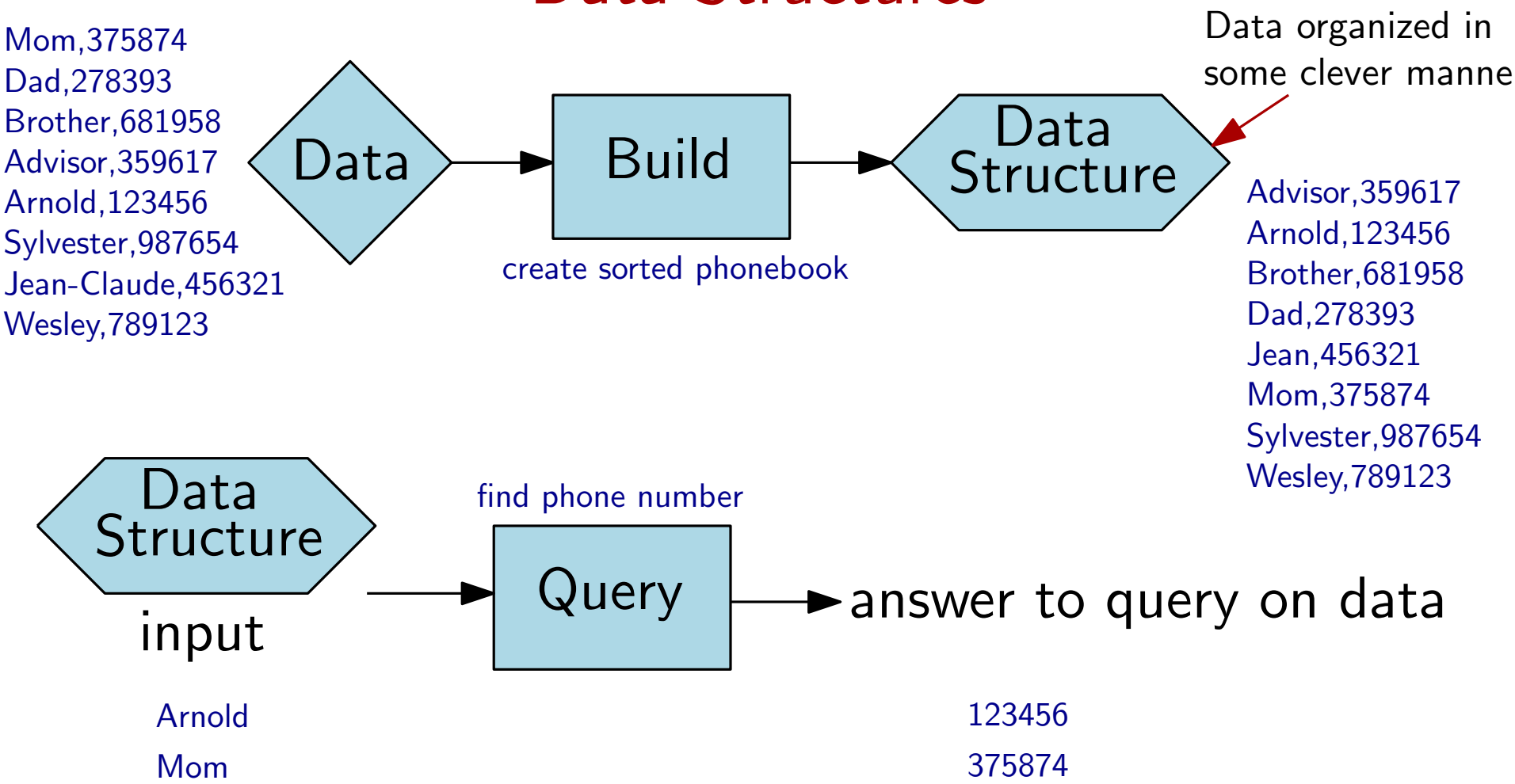
Data Structures



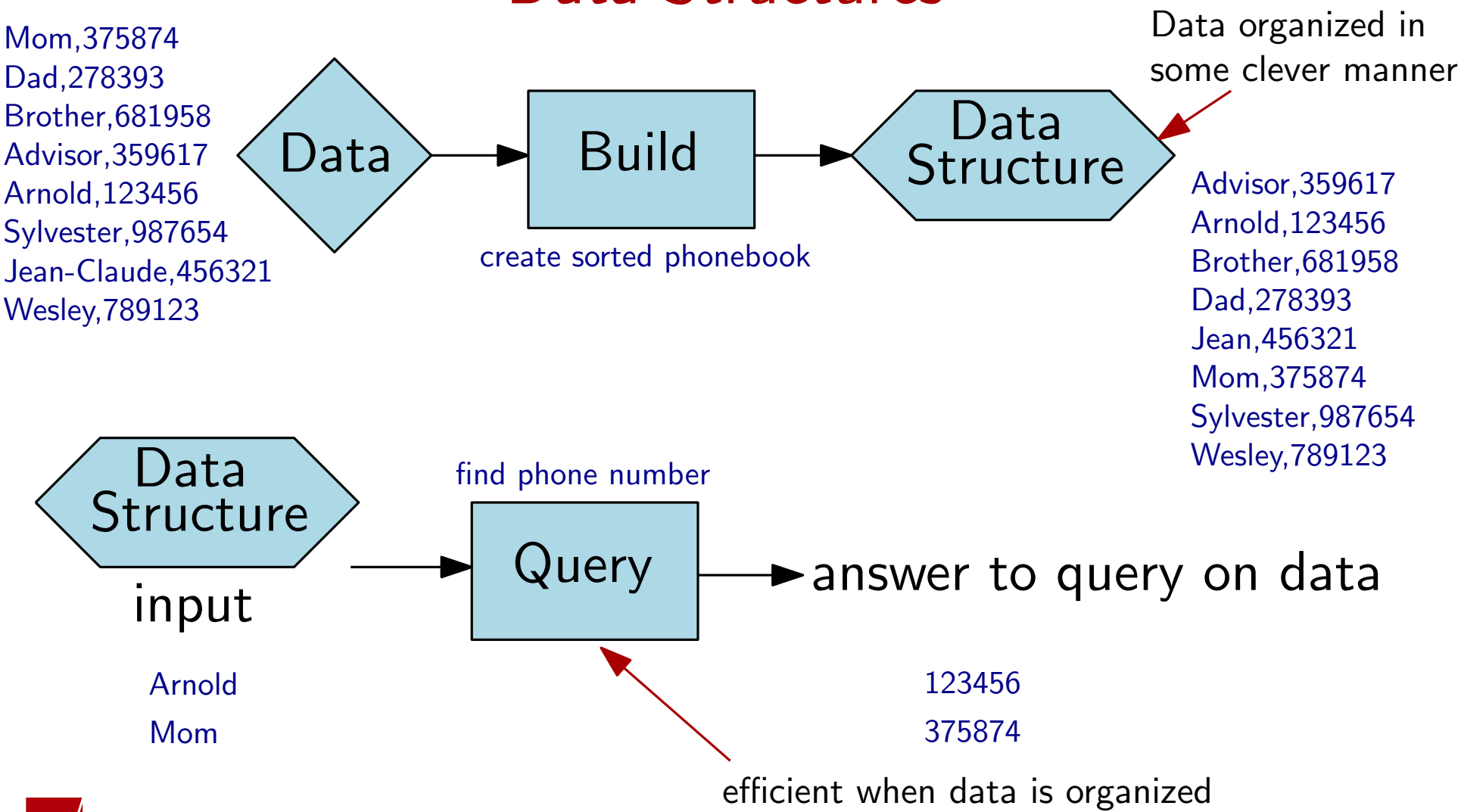
Data Structures



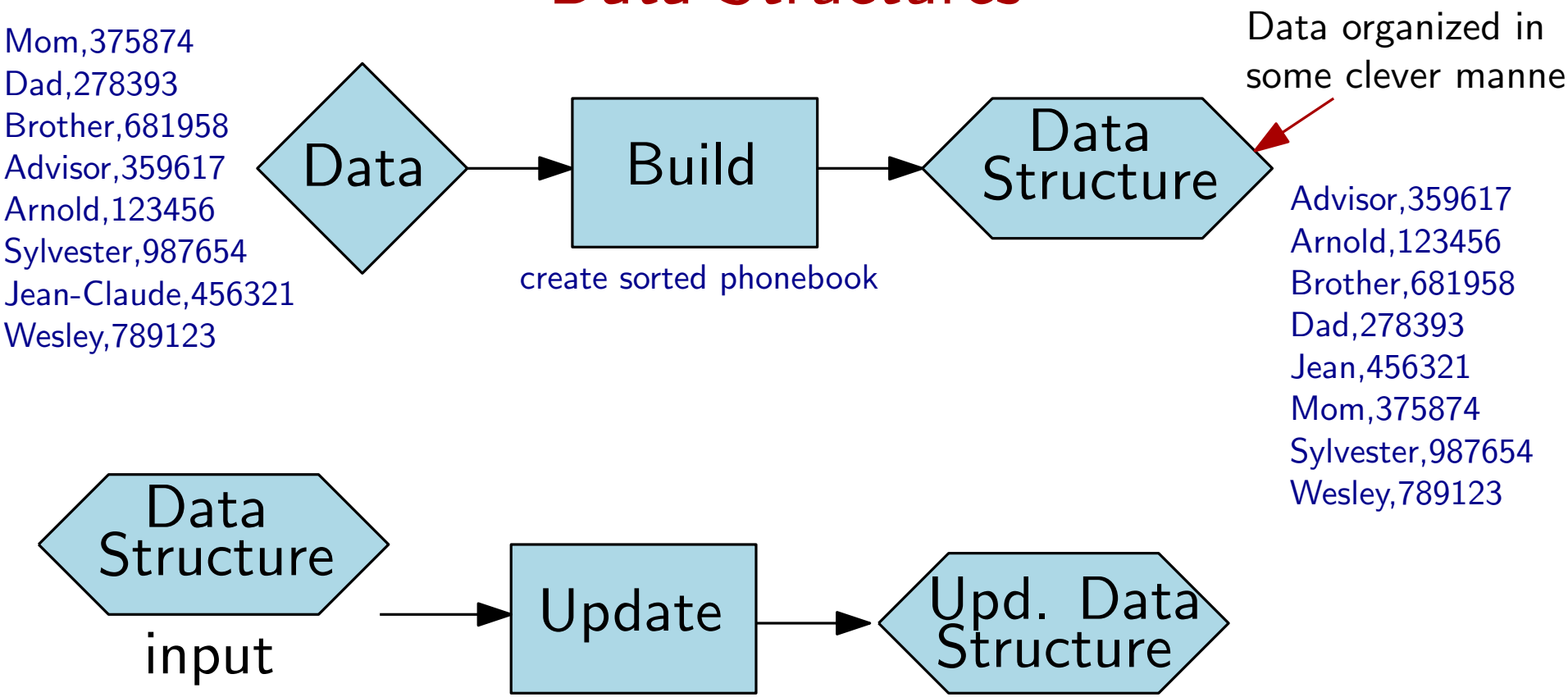
Data Structures



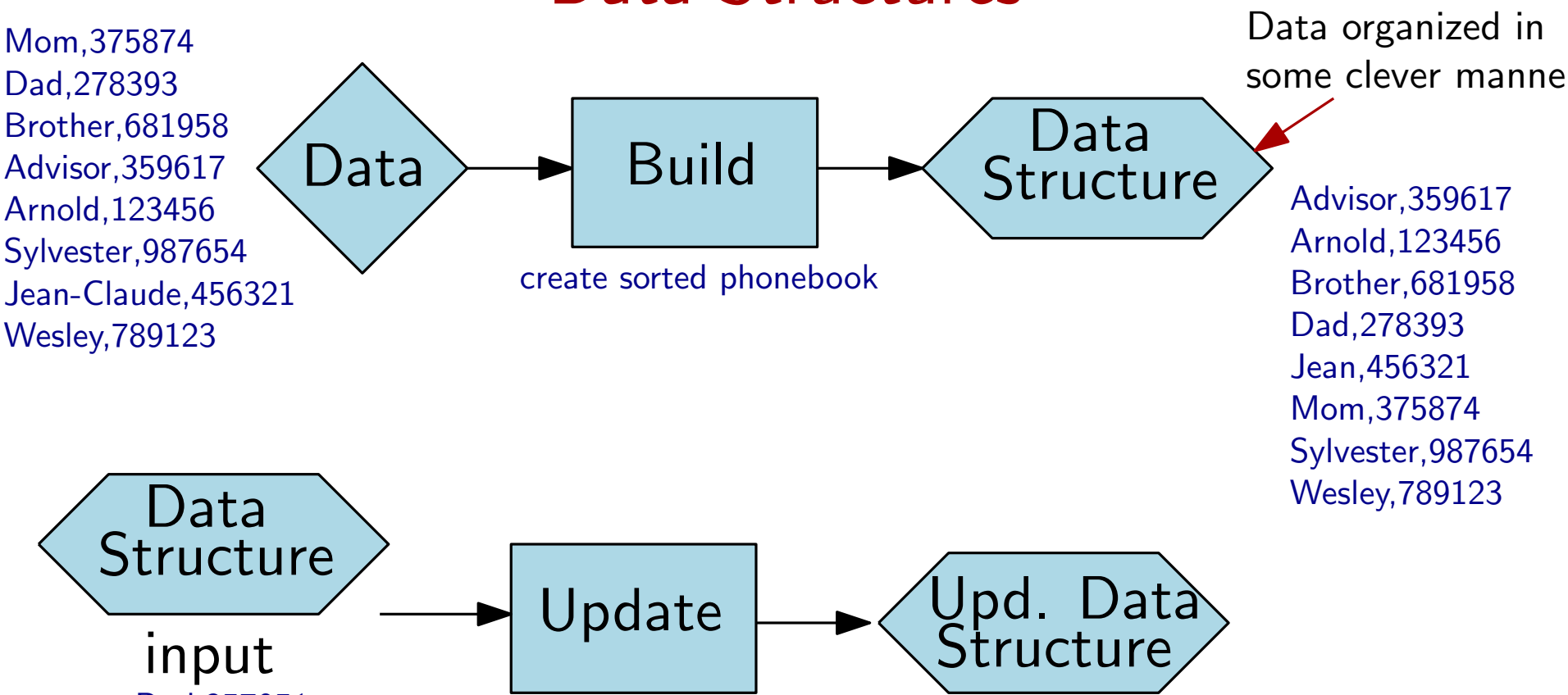
Data Structures



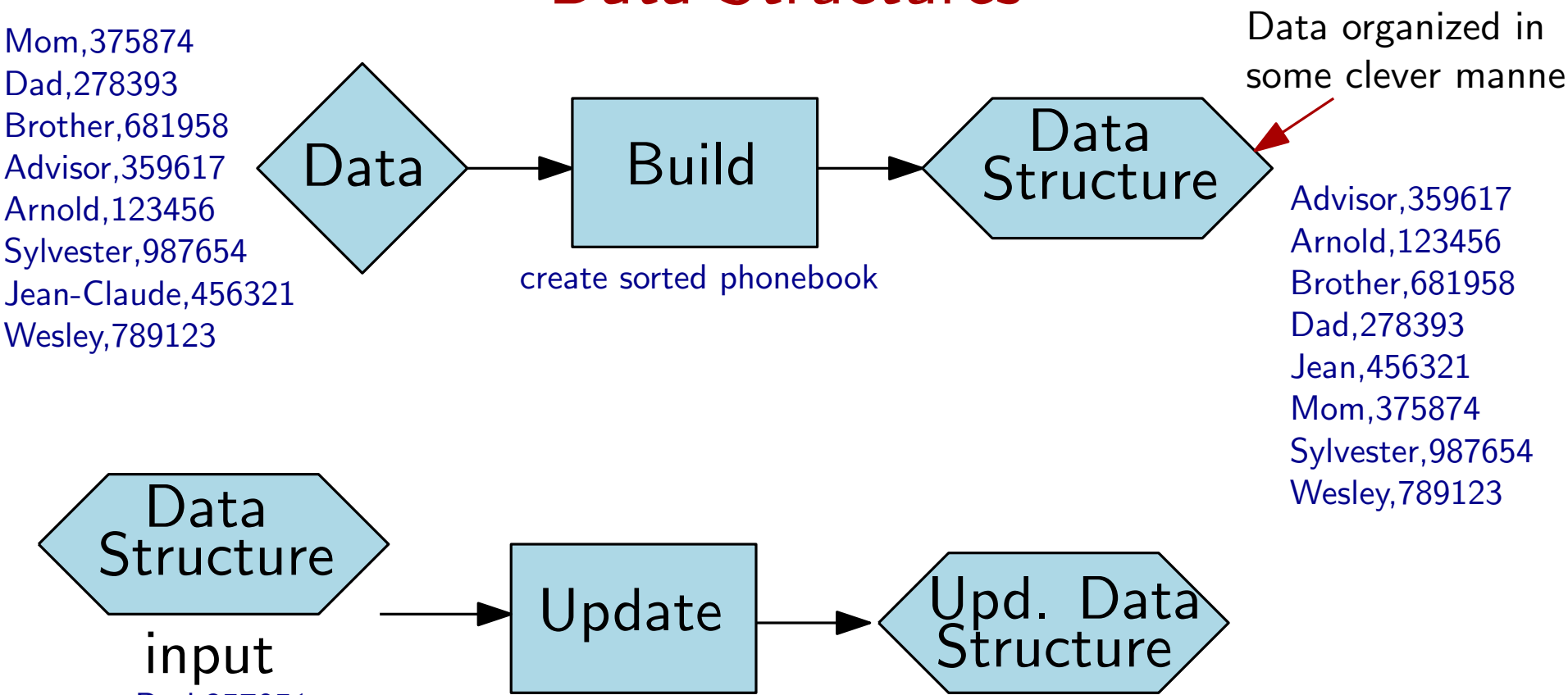
Data Structures



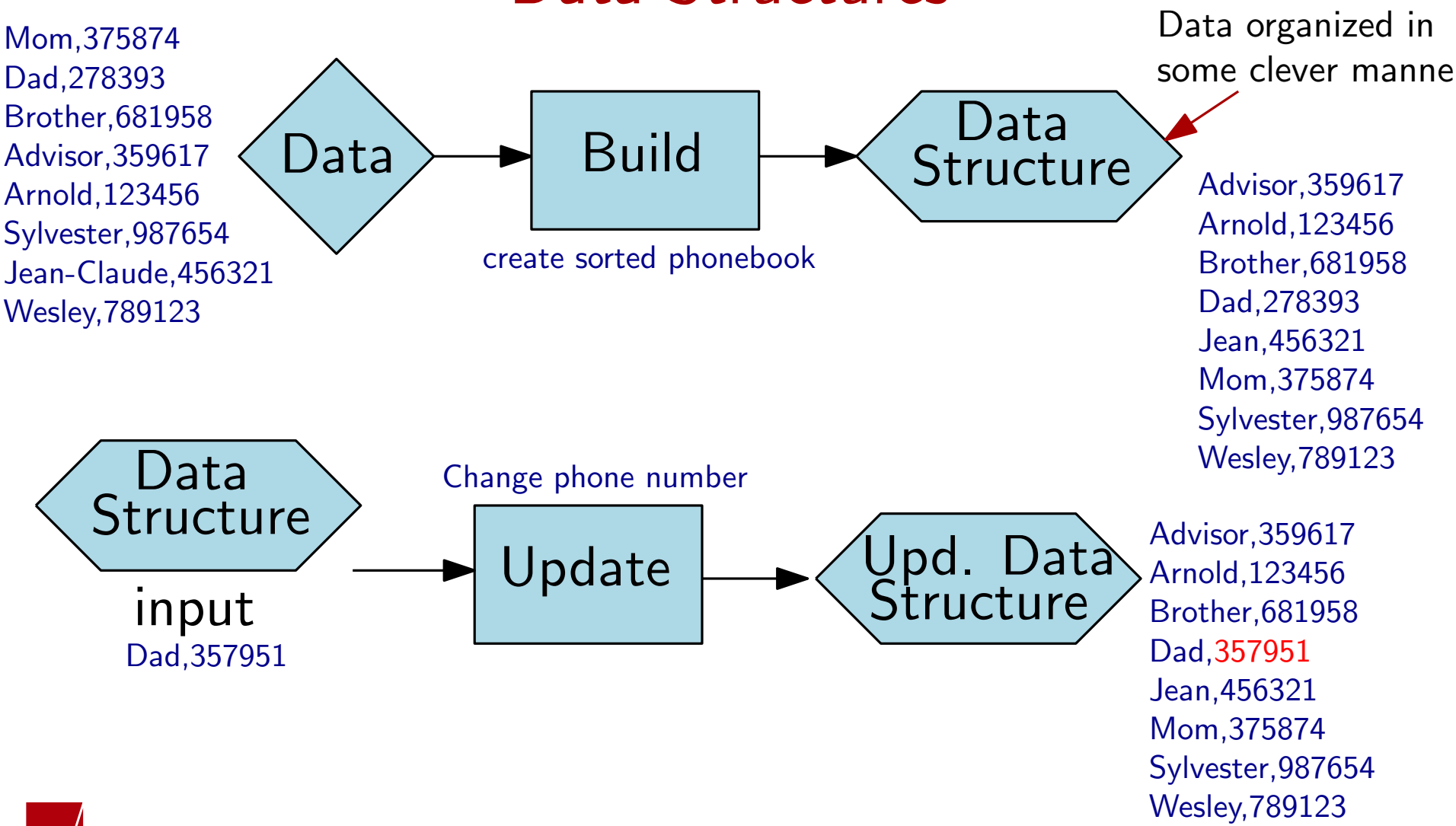
Data Structures



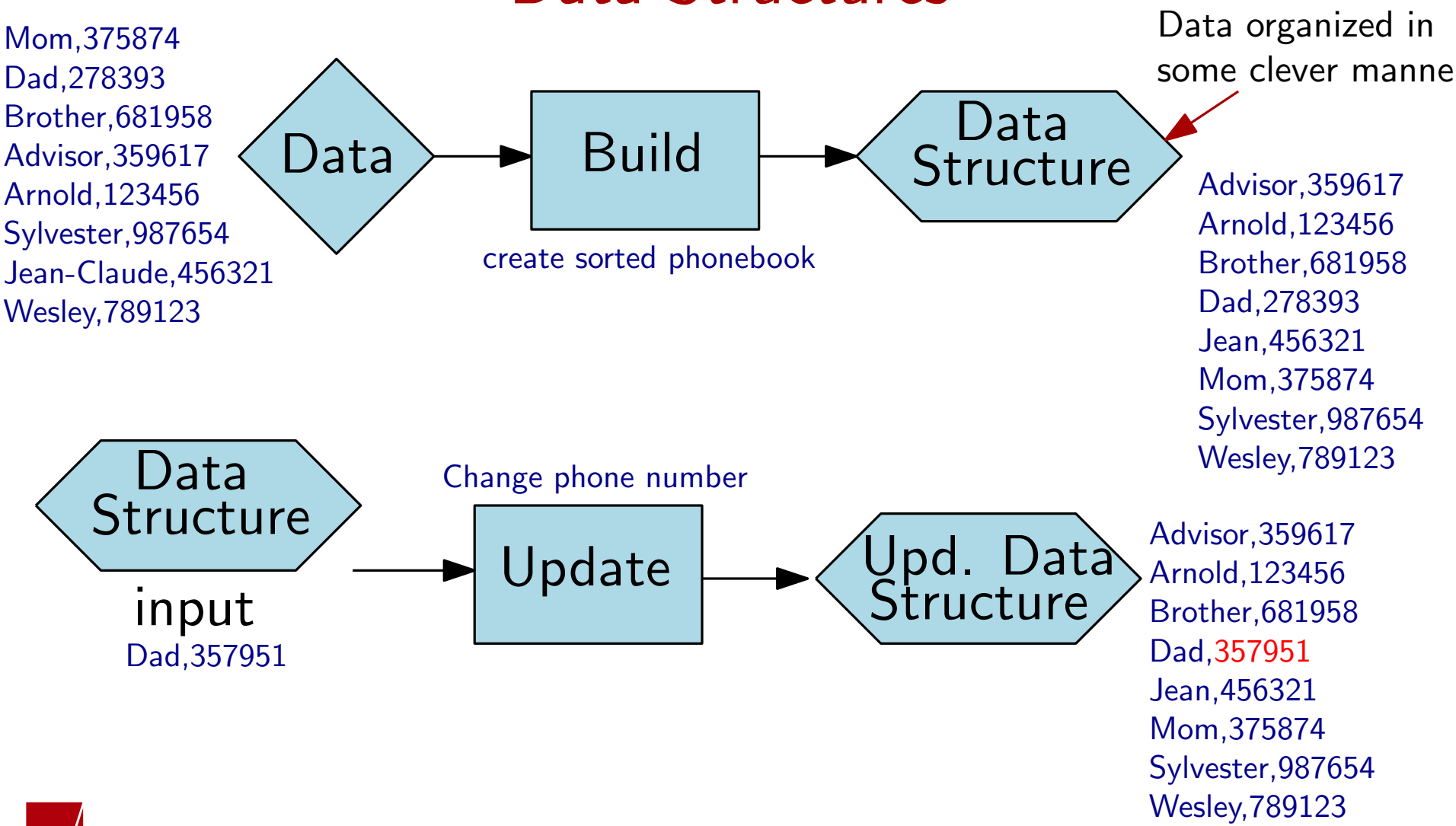
Data Structures



Data Structures

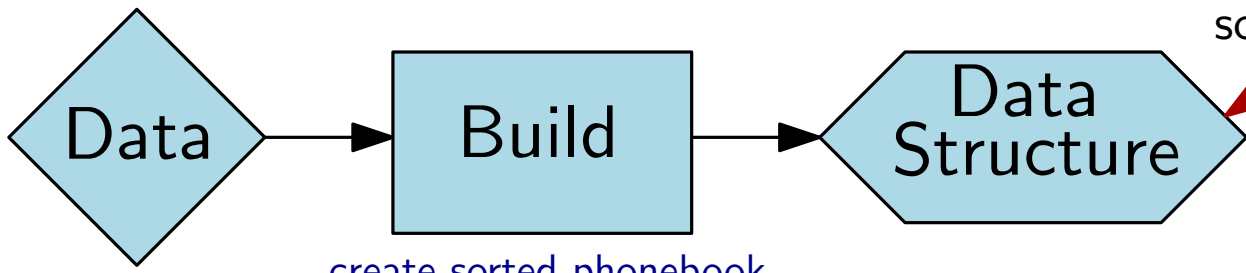


Data Structures



Data Structures

Mom,375874
Dad,278393
Brother,681958
Advisor,359617
Arnold,123456
Sylvester,987654
Jean-Claude,456321
Wesley,789123

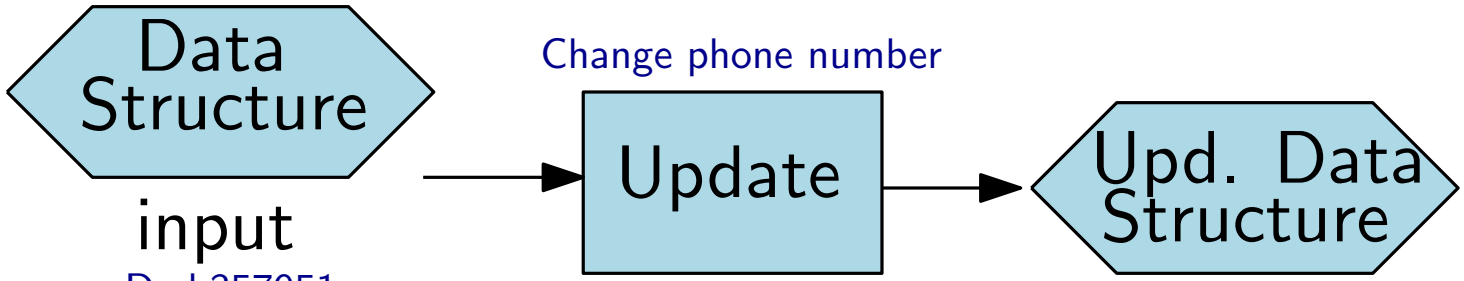


create sorted phonebook

Data organized in some clever manner

Advisor,359617
Arnold,123456
Brother,681958
Dad,278393
Jean,456321
Mom,375874
Sylvester,987654
Wesley,789123

We want space efficient structures, fast queries, and fast updates



Advisor,359617
Arnold,123456
Brother,681958
Dad,357951
Jean,456321
Mom,375874
Sylvester,987654
Wesley,789123

Outline

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example

Outline

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing
5. State and University Library

Implicit Data Structures

Implicit Data Structures

Primary goal: space efficiency

Implicit Data Structures

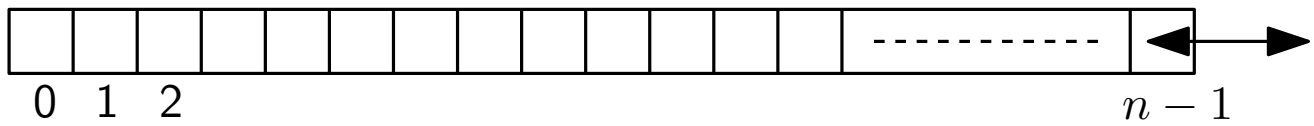
Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.

Implicit Data Structures

Primary goal: space efficiency

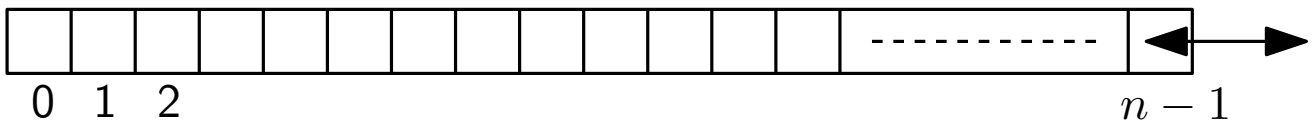
Data Structure has n comparable elements, laid out in an array of size n.



Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.

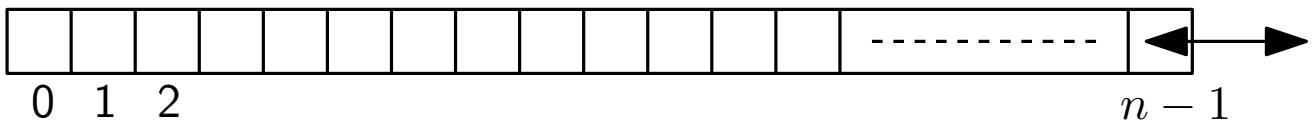


Only the array and n is maintained between operations.

Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



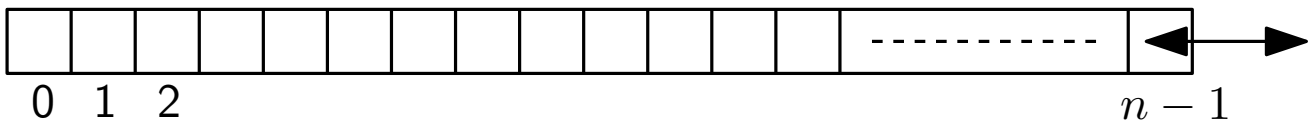
Only the array and n is maintained between operations.

Strict implicit model: *Nothing* but elements and n stored

Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



Only the array and n is maintained between operations.

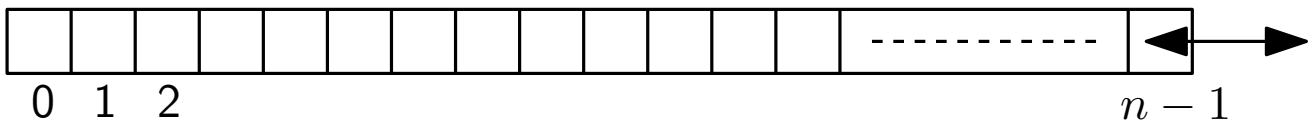
Strict implicit model: *Nothing* but elements and n stored

Weak implicit model: Allow a constant number of extra memory cells

Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



Only the array and n is maintained between operations.

Strict implicit model: *Nothing* but elements and n stored

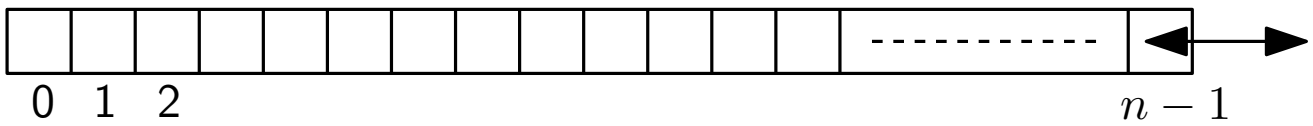
Weak implicit model: Allow a constant number of extra memory cells

We work with the strict implicit model

Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



Only the array and n is maintained between operations.

Strict implicit model: *Nothing* but elements and n stored

Weak implicit model: Allow a constant number of extra memory cells

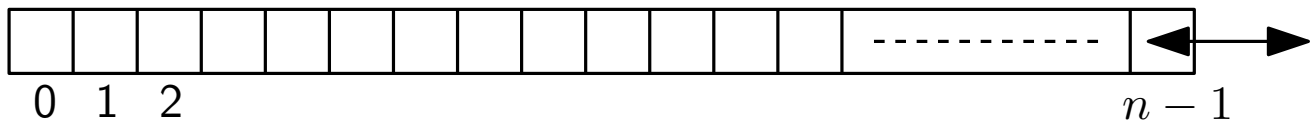
We work with the strict implicit model

Fundamental trick: pair encoding bits.

Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



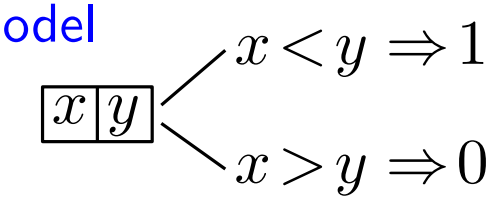
Only the array and n is maintained between operations.

Strict implicit model: *Nothing* but elements and n stored

Weak implicit model: Allow a constant number of extra memory cells

We work with the strict implicit model

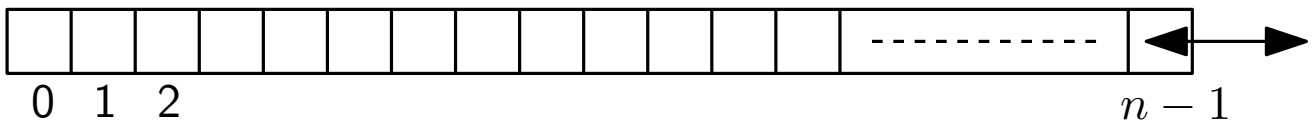
Fundamental trick: pair encoding bits.



Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



Only the array and n is maintained between operations.

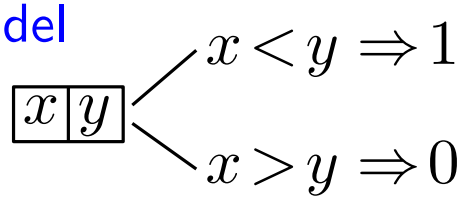
Strict implicit model: *Nothing* but elements and n stored

Weak implicit model: Allow a constant number of extra memory cells

We work with the strict implicit model

Fundamental trick: pair encoding bits.

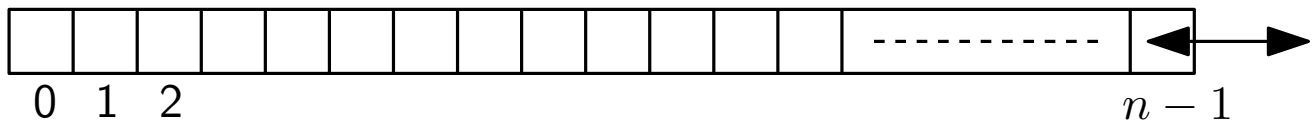
Requires distinct elements.



Implicit Data Structures

Primary goal: space efficiency

Data Structure has n comparable elements, laid out in an array of size n.



Only the array and n is maintained between operations.

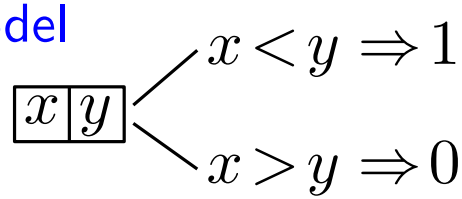
Strict implicit model: *Nothing* but elements and n stored

Weak implicit model: Allow a constant number of extra memory cells

We work with the strict implicit model

Fundamental trick: pair encoding bits.

Requires distinct elements.



Procedures can use a constant number of working memory/registers

Dictionary and Fingersearch

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Find(k)

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Find(k)

Predecessor(k)

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Find(k)

Predecessor(k)

Successor(k)

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Find(k)

Predecessor(k)

Successor(k)

Time: $O(\log n)$

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

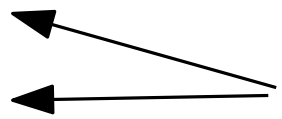
Insert(k)
Delete(k) — Only in dynamic structures
Find(k)
Predecessor(k)
Successor(k)

Time: $O(\log n)$

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)
Delete(k)
Find(k)
Predecessor(k)
Successor(k)



Only in dynamic structures

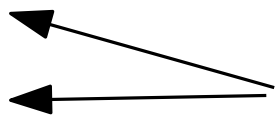
Special element: the finger element

Time: $O(\log n)$

Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

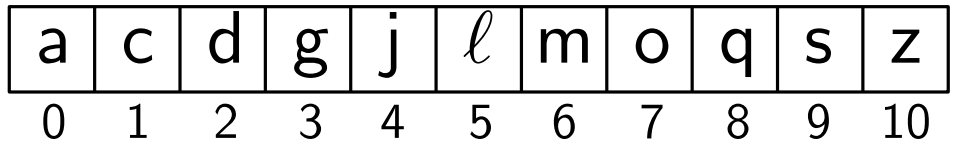
Insert(k)
Delete(k)
Find(k)
Predecessor(k)
Successor(k)



Only in dynamic structures

Special element: the finger element

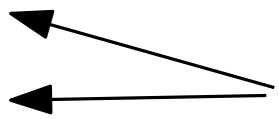
Time: $O(\log n)$



Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

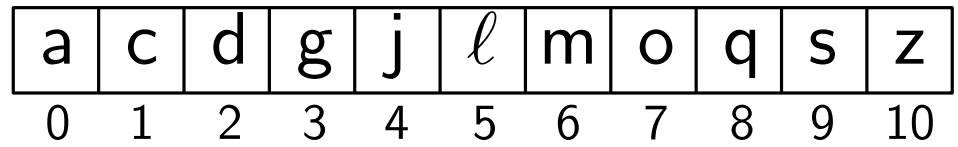
Insert(k)
Delete(k)
Find(k)
Predecessor(k)
Successor(k)



Only in dynamic structures

Special element: the finger element

Time: $O(\log n)$



n=11

Dictionary and Fingersearch

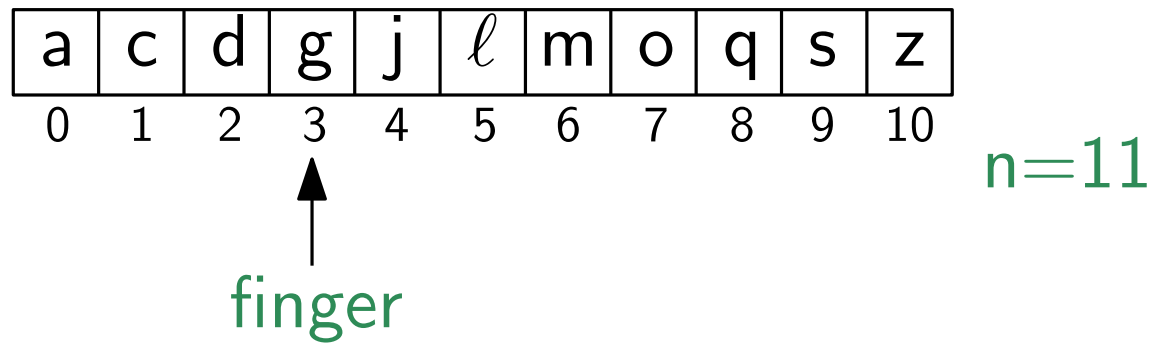
Maintain a (dynamic) set of n keys, while supporting:

Insert(k)
Delete(k)
Find(k)
Predecessor(k)
Successor(k)

Only in dynamic structures

Special element: the finger element

Time: $O(\log n)$



Dictionary and Fingersearch

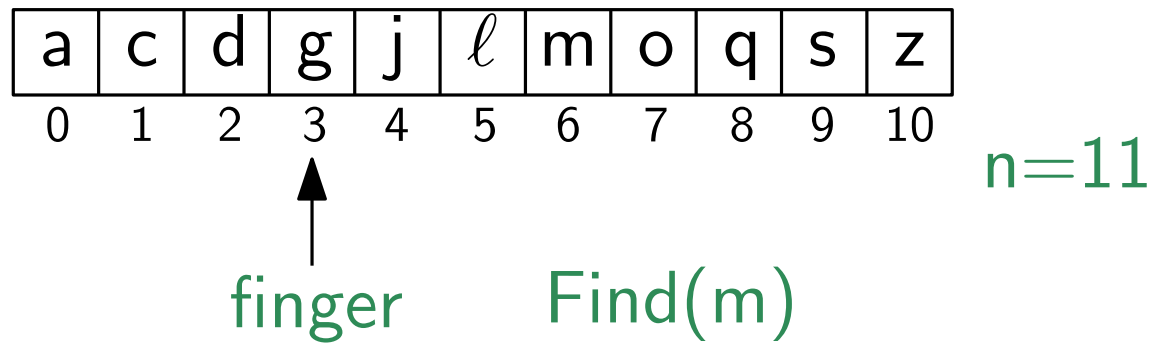
Maintain a (dynamic) set of n keys, while supporting:

Insert(k)
Delete(k)
Find(k)
Predecessor(k)
Successor(k)

Only in dynamic structures

Special element: the finger element

Time: $O(\log n)$



Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Only in dynamic structures

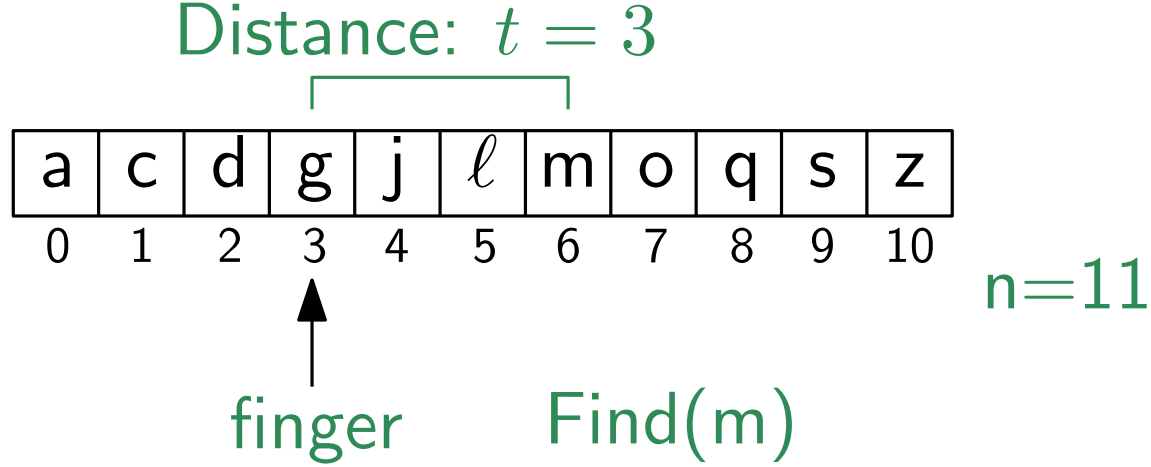
Find(k)

Predecessor(k)

Successor(k)

Time: $O(\log n)$

Special element: the finger element



Dictionary and Fingersearch

Maintain a (dynamic) set of n keys, while supporting:

Insert(k)

Delete(k)

Only in dynamic structures

Find(k)

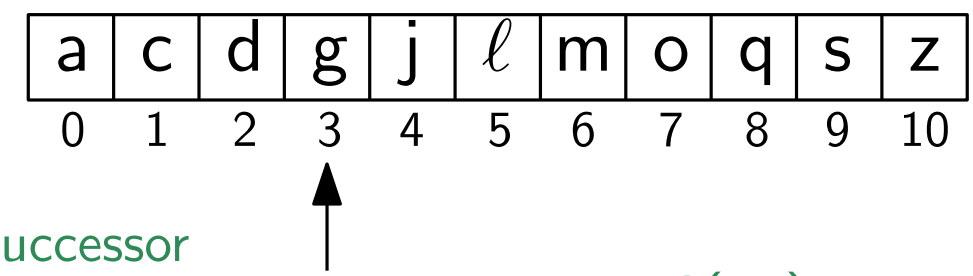
Predecessor(k)

Successor(k)

Time: $O(\log n)$

Special element: the finger element

Distance: $t = 3$



Time: $O(\log t)$ for
Find, Predecessor, and Successor

Implicit Finger Search Idea

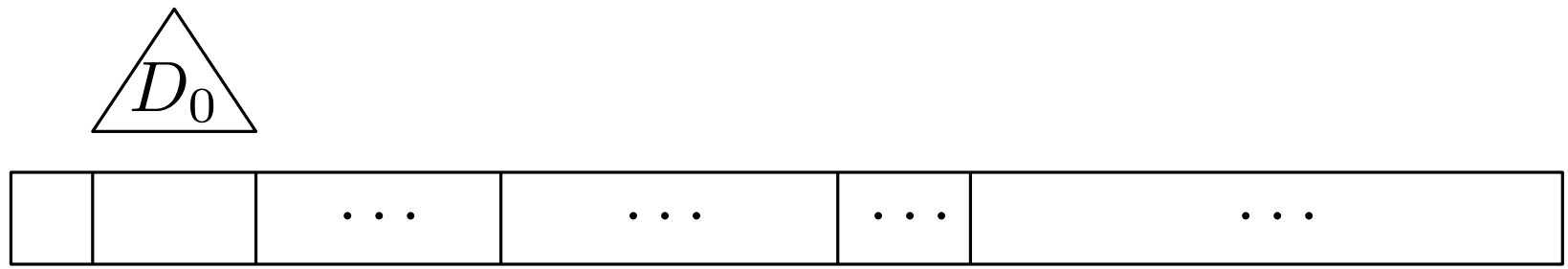
Implicit Finger Search Idea

Implicit Finger Search Idea

2^{2^0} 2^{2^1} 2^{2^2} 2^{2^ℓ}

Implicit Finger Search Idea

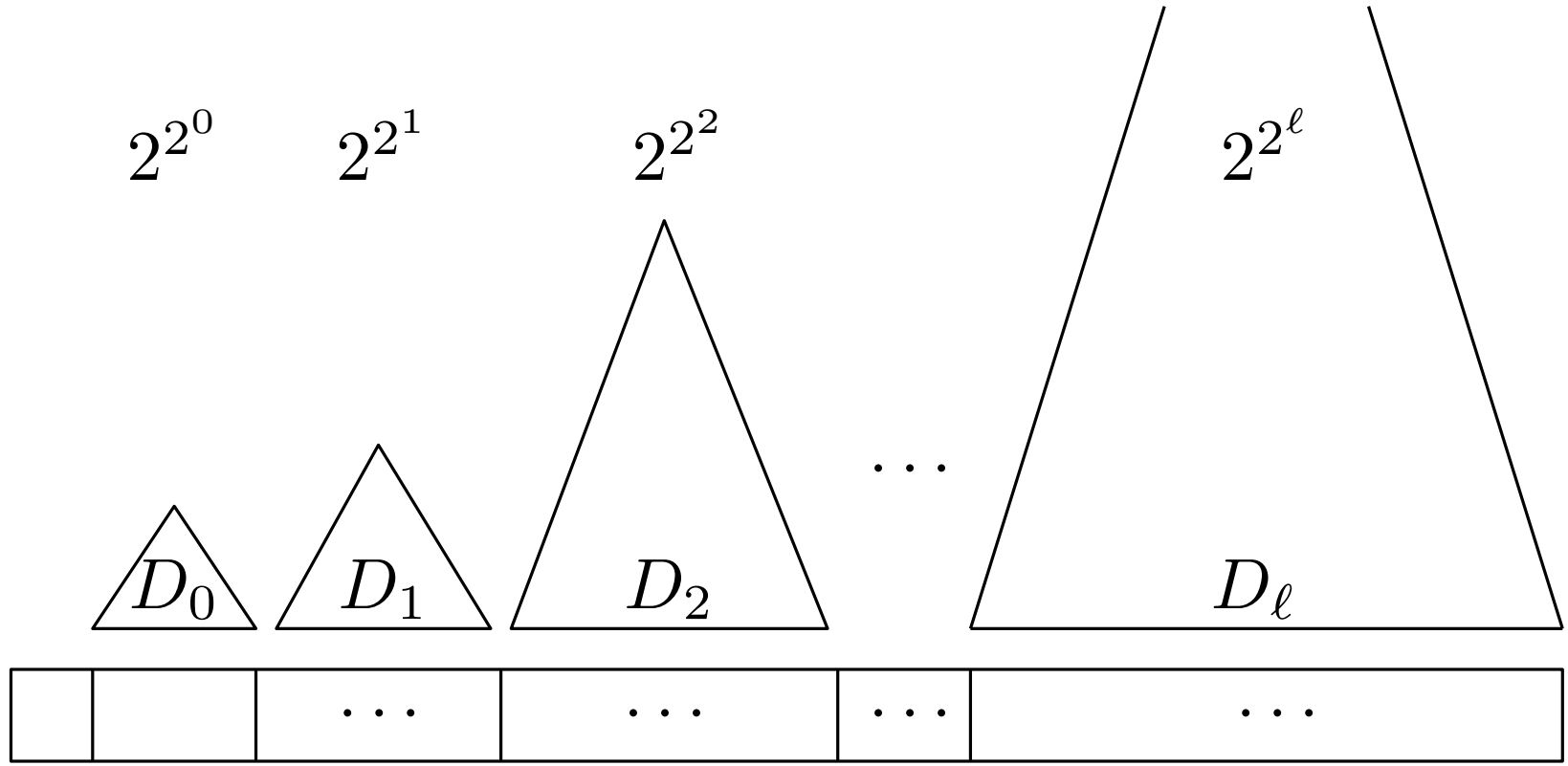
2^{2^0} 2^{2^1} 2^{2^2} 2^{2^ℓ}



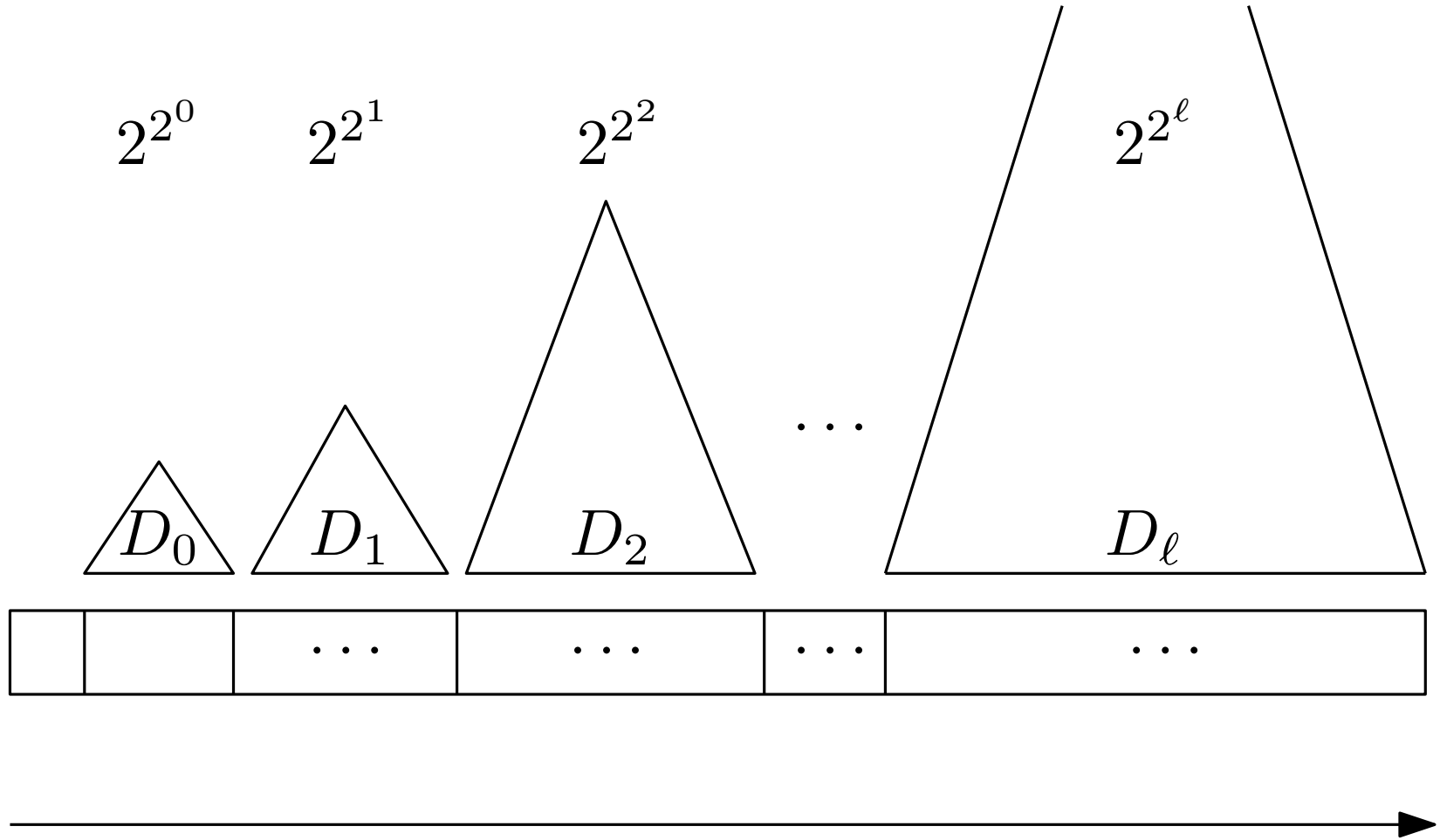
Implicit Finger Search Idea

2^{2^0} 2^{2^1} 2^{2^2} 2^{2^ℓ}

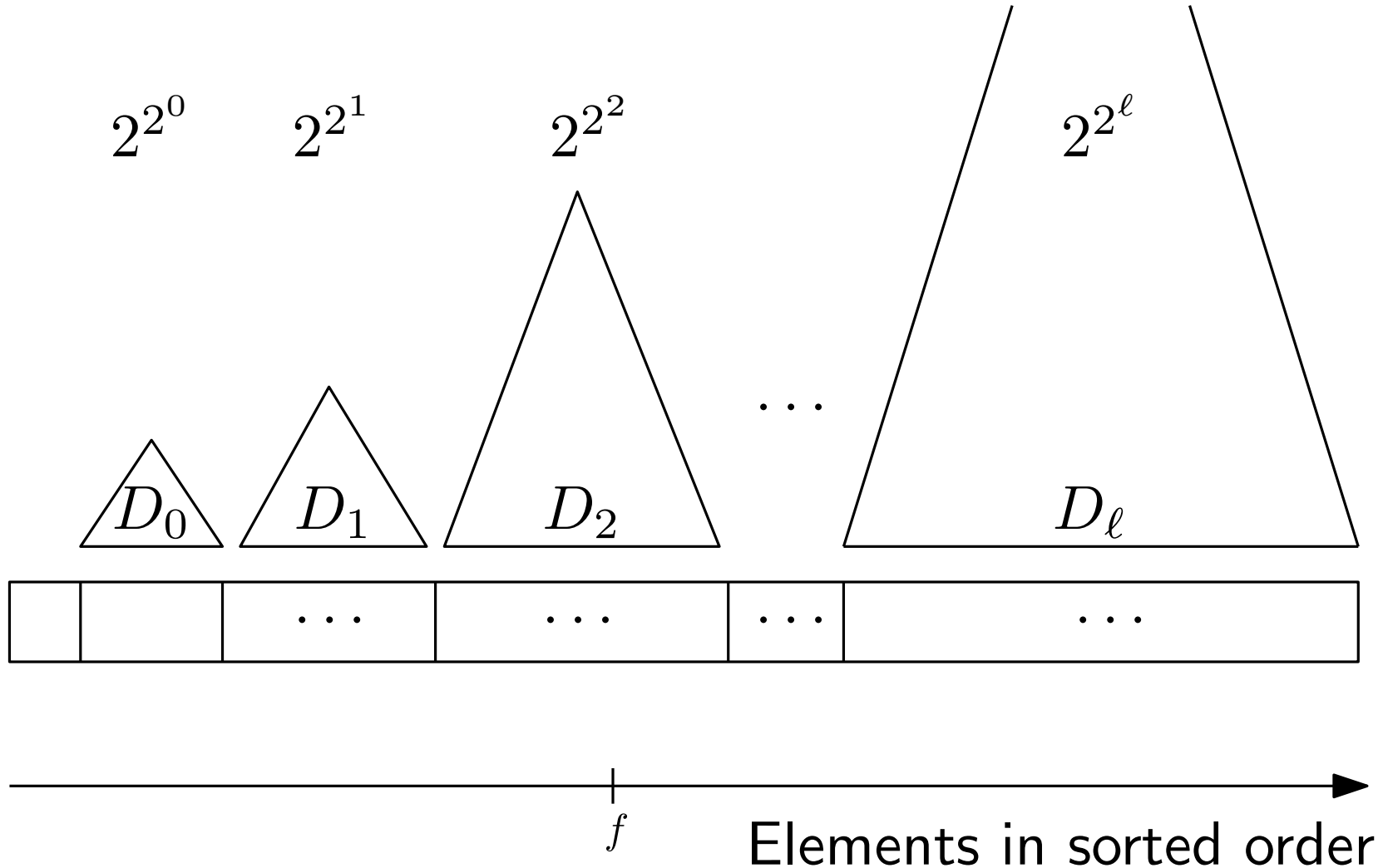
Implicit Finger Search Idea



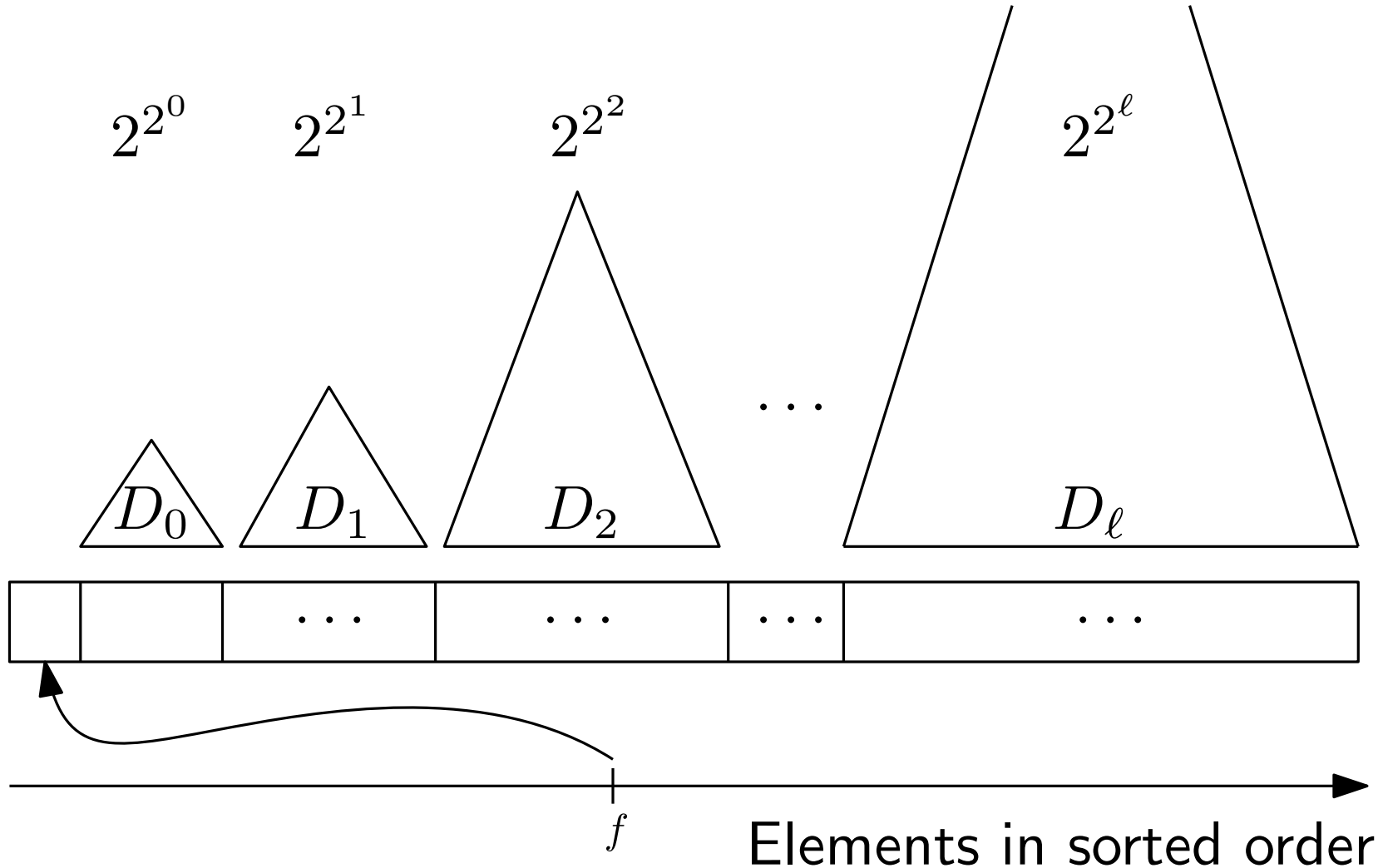
Implicit Finger Search Idea



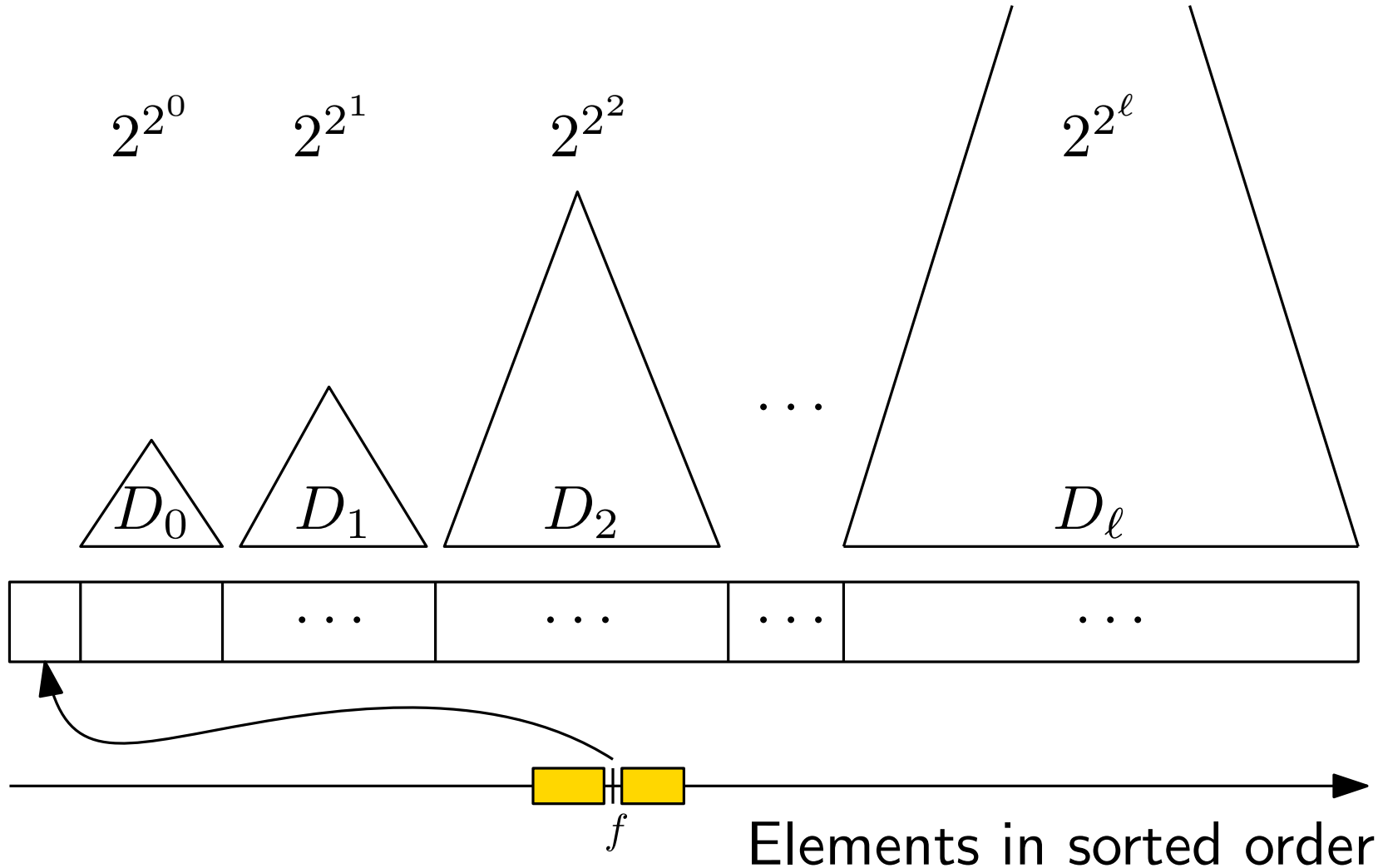
Implicit Finger Search Idea



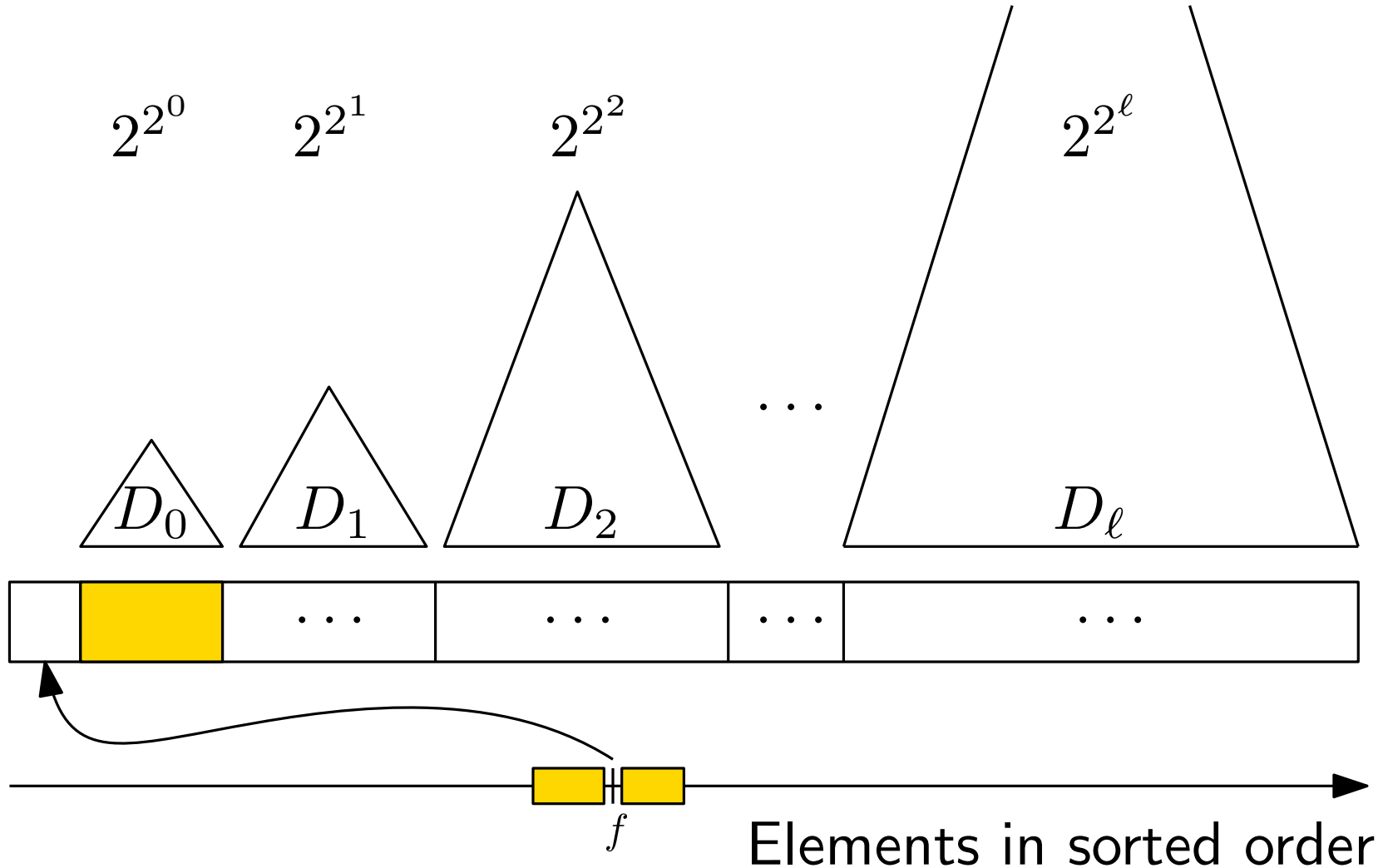
Implicit Finger Search Idea



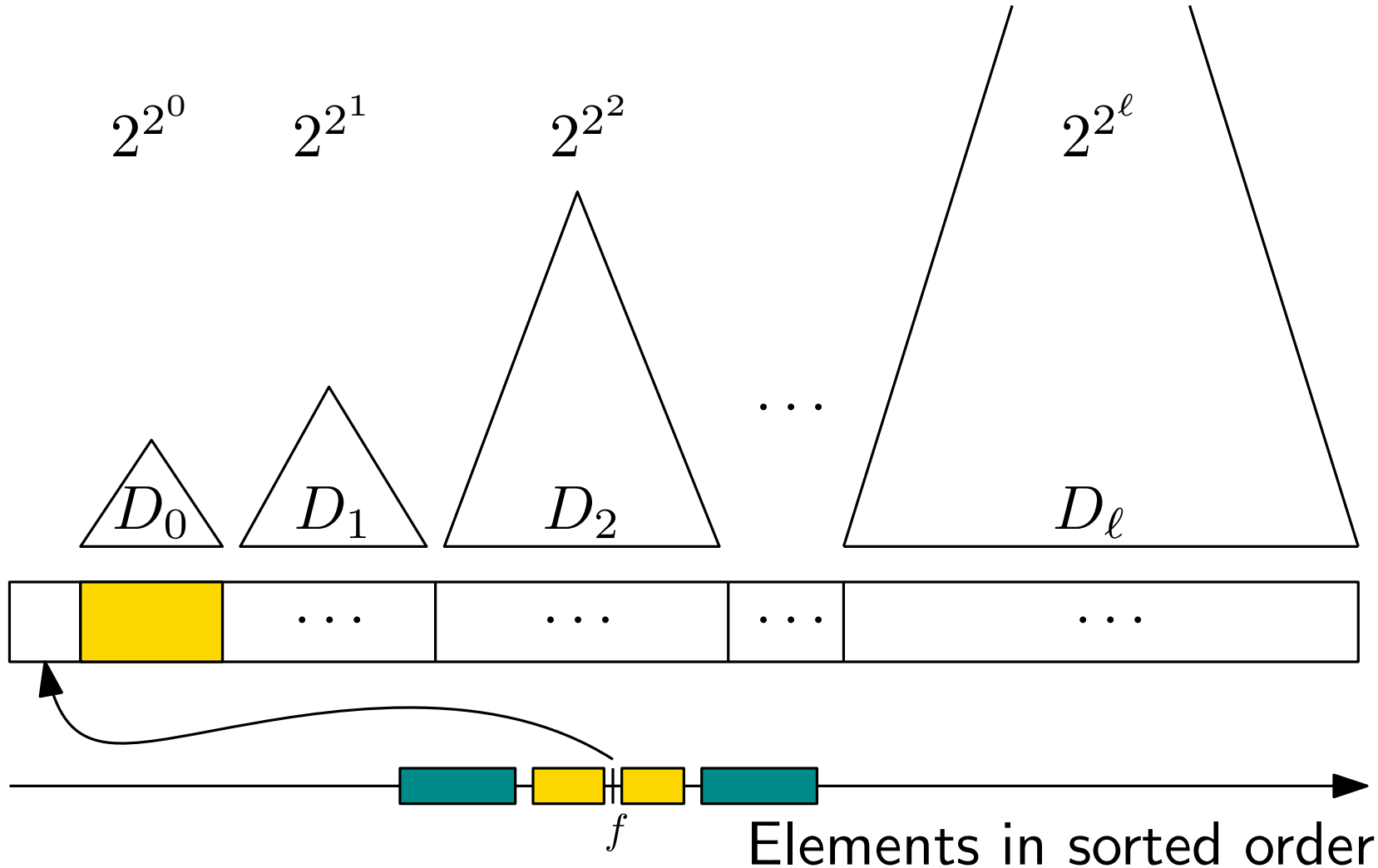
Implicit Finger Search Idea



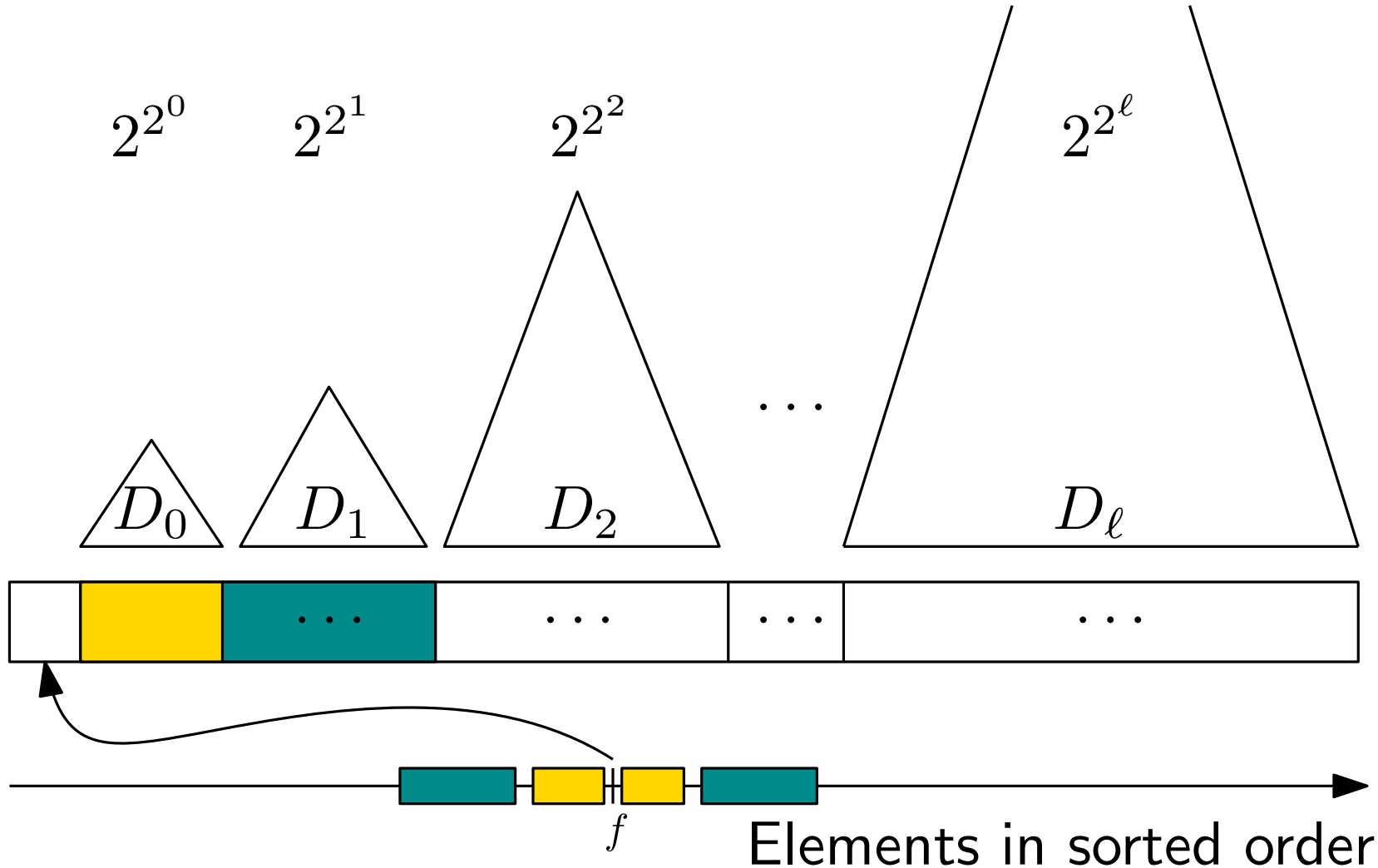
Implicit Finger Search Idea



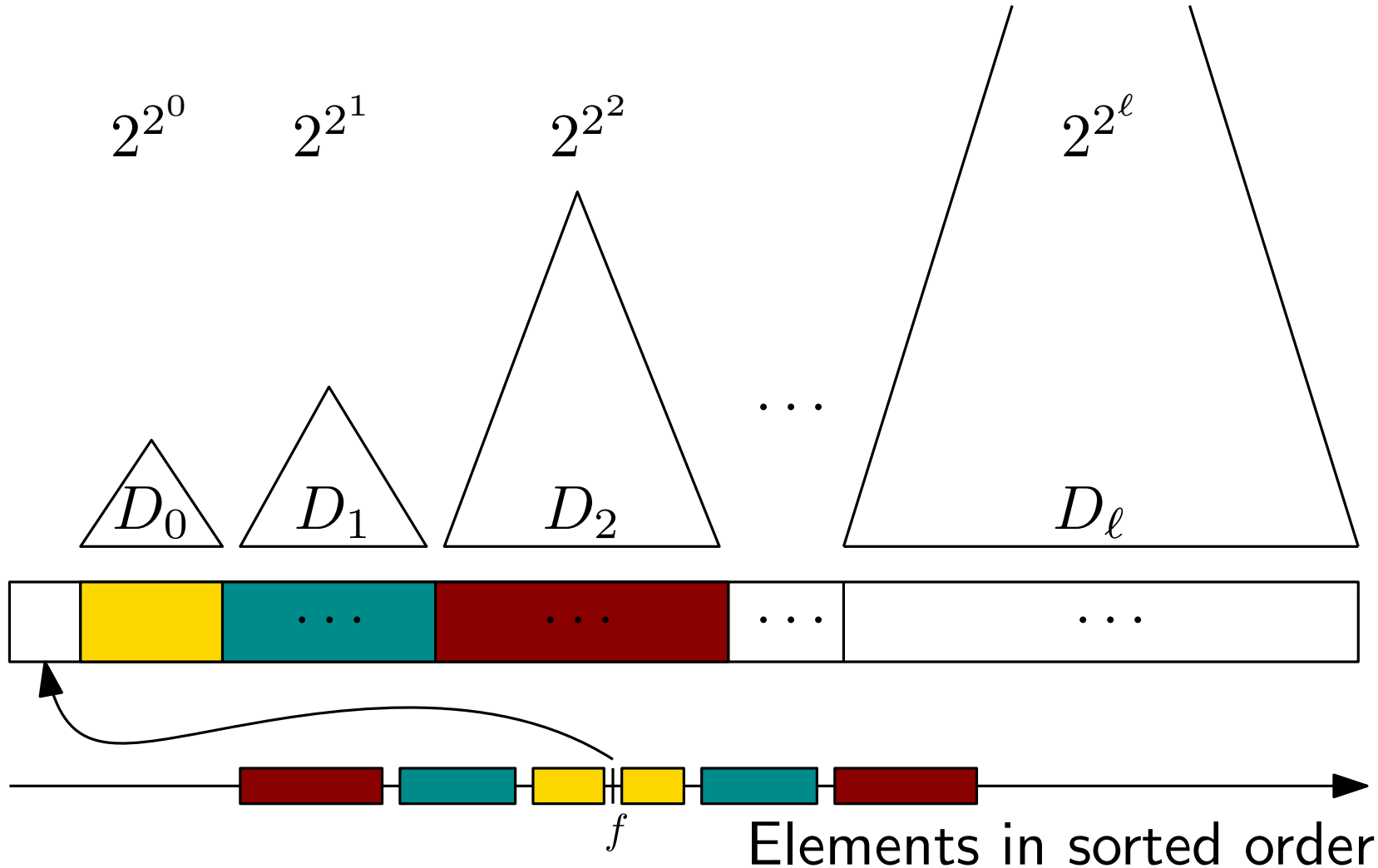
Implicit Finger Search Idea



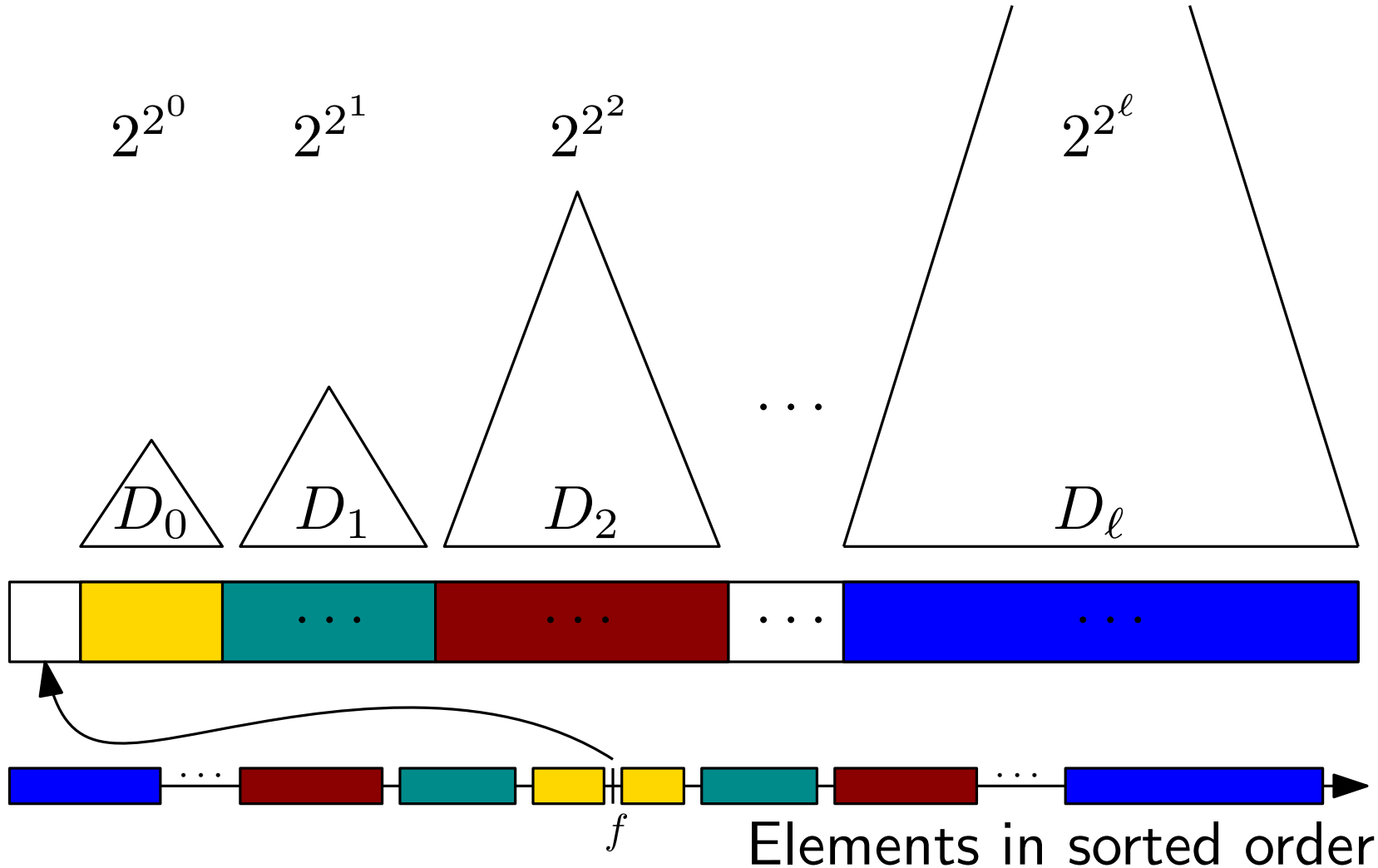
Implicit Finger Search Idea



Implicit Finger Search Idea

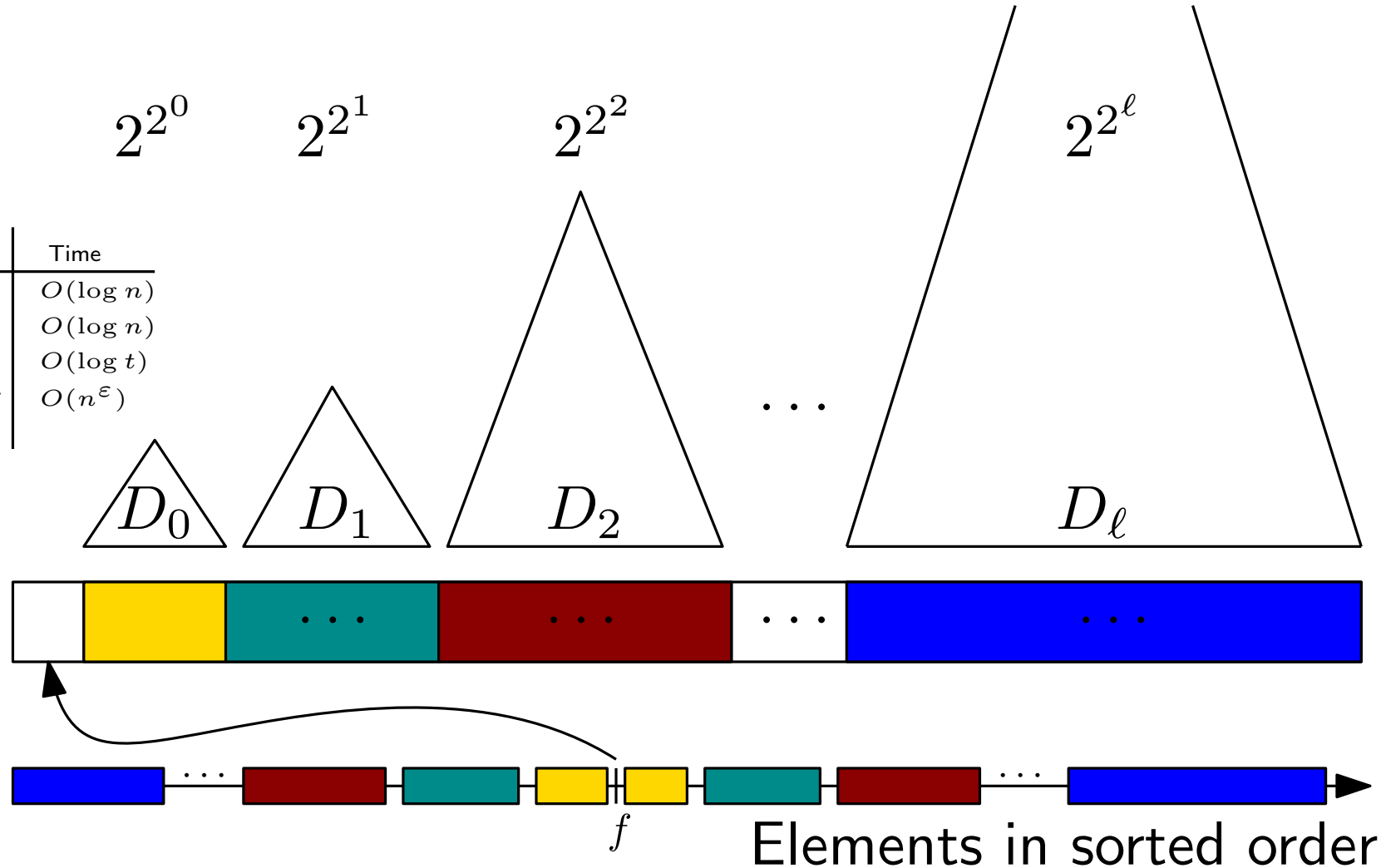


Implicit Finger Search Idea

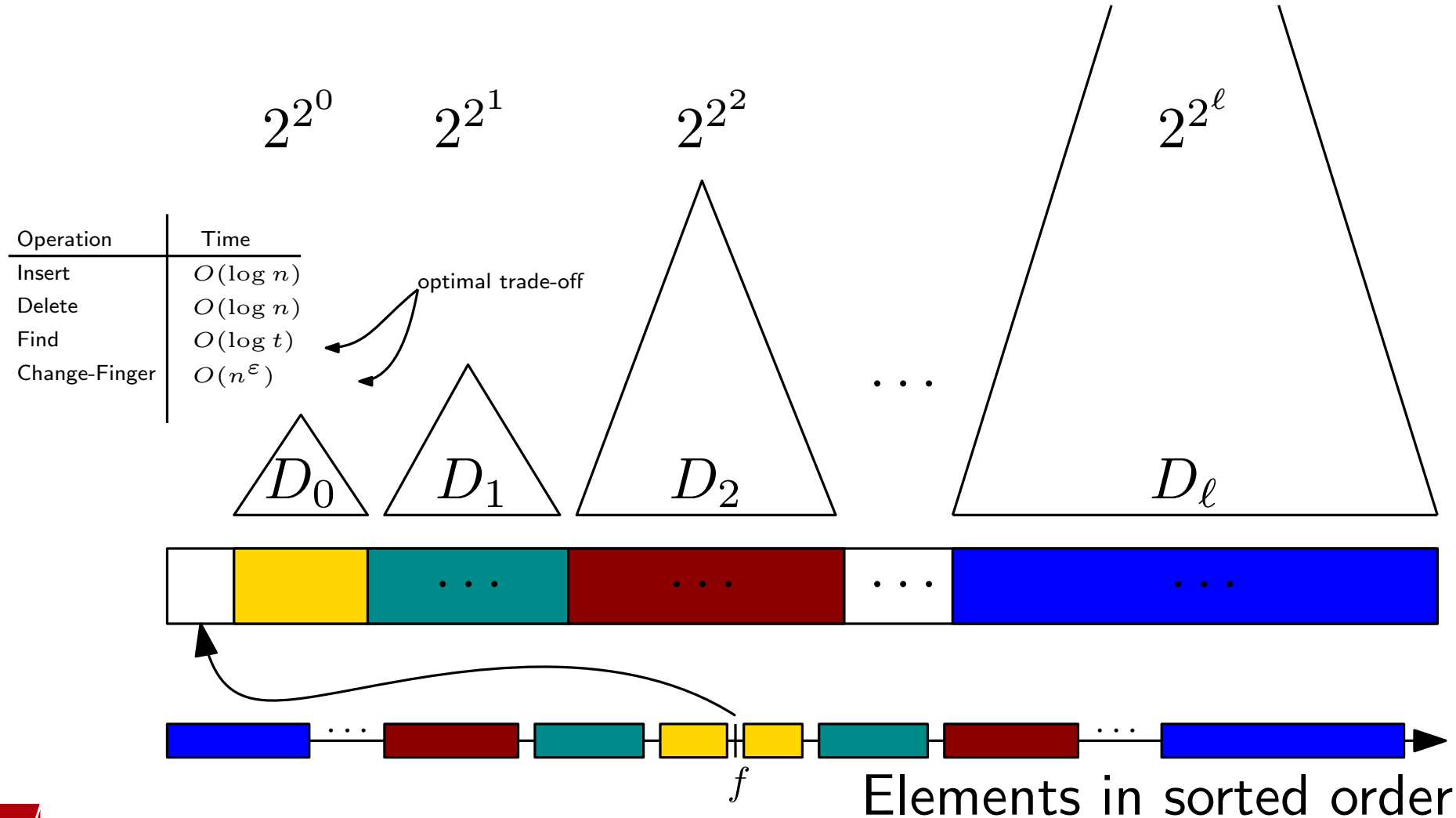


Implicit Finger Search Idea

Operation	Time
Insert	$O(\log n)$
Delete	$O(\log n)$
Find	$O(\log t)$
Change-Finger	$O(n^{\varepsilon})$



Implicit Finger Search Idea



Priority Queues

Priority Queues

Maintain a set of n keys and support:

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes
Edelkamp *et al.*	1	$\log n$	$\log n$	no	yes

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes
Edelkamp *et al.*	1	$\log n$	$\log n$	no	yes
Harvey & Zatloukal	$\star$ 1	$\star$ $\log n$	$\star$ $\log n$	yes	yes

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes
Edelkamp *et al.*	1	$\log n$	$\log n$	no	yes
Harvey & Zatloukal	$\star$ 1	$\star$ $\log n$	$\star$ $\log n$	yes	yes
Franceschini & Munro	$\star$ $\log n$	$\star$ $\log n$	$\star$ 1	yes	no

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes
Edelkamp *et al.*	1	$\log n$	$\log n$	no	yes
Harvey & Zatloukal	$\star$ 1	$\star$ $\log n$	$\star$ $\log n$	yes	yes
Franceschini & Munro	$\star$ $\log n$	$\star$ $\log n$	$\star$ 1	yes	no
New	$\star$ 1	$\star$ $\log n$	$\star$ 1	yes	yes

Amortized bounds are marked with $\star$

Priority Queues

Maintain a set of n keys and support:

- FindMin: Return the minimum element
- ExtractMin: Remove the minimum element from the set
- Insert(k): Add the key k to the set

	Insert	ExtractMin	Moves	Strict	Identical Elements
Williams	$\log n$	$\log n$	$\log n$	yes	yes
Carlsson *et al.*	1	$\log n$	$\log n$	no	yes
Edelkamp *et al.*	1	$\log n$	$\log n$	no	yes
Harvey & Zatloukal	$\star$ 1	$\star$ $\log n$	$\star$ $\log n$	yes	yes
Franceschini & Munro	$\star$ $\log n$	$\star$ $\log n$	$\star$ 1	yes	no
New	$\star$ 1	$\star$ $\log n$	$\star$ 1	yes	yes
New	1	$\log n$	$\log n$	yes	no

Amortized bounds are marked with $\star$

Outline

Outline

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues

Outline

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing
5. State and University Library

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

conclude $A > B$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need
$\Theta(n \log n)$ scalings/comparisons

conclude $A > B$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need
$\Theta(n \log n)$ scalings/comparisons

conclude $A > B$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need
$\Theta(n \log n)$ scalings/comparisons

Question: Can we sort faster if
we have access to a digital scale?

conclude $A > B$

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need $\Theta(n \log n)$ scalings/comparisons

conclude $A > B$

Question: Can we sort faster if we have access to a digital scale?

Answer: Yes.

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need $\Theta(n \log n)$ scalings/comparisons

conclude $A > B$

Question: Can we sort faster if we have access to a digital scale?

Answer: Yes.

Question: How much faster?

Integer Sorting

Comparison based sorting: only allowed operation $\leq$

To sort n gold pieces we need $\Theta(n \log n)$ scalings/comparisons

conclude $A > B$

Question: Can we sort faster if we have access to a digital scale?

Answer: Yes.

Question: How much faster?

Depends on accuracy of the scale

Integer Sorting

Integer Sorting

Algorithm	Time	Space	Notes

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie
Kirkpatrick & Reisch (1984)	$O(n \log \frac{w}{\log n})$		Only asymptotically better than vEB when $w = O(\log n)$

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie
Kirkpatrick & Reisch (1984)	$O(n \log \frac{w}{\log n})$		Only asymptotically better than vEB when $w = O(\log n)$
Signature sort (1995)	$O(n)$	$O(n)$	Only for $w = \Omega(\log^{2+\varepsilon} n)$

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie
Kirkpatrick & Reisch (1984)	$O(n \log \frac{w}{\log n})$		Only asymptotically better than vEB when $w = O(\log n)$
Signature sort (1995)	$O(n)$	$O(n)$	Only for $w = \Omega(\log^{2+\varepsilon} n)$
Han & Thorup (2002)	$O(n\sqrt{\log(w/\log n)})$	$O(n)$	Randomized with mulitplication

w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie
Kirkpatrick & Reisch (1984)	$O(n \log \frac{w}{\log n})$		Only asymptotically better than vEB when $w = O(\log n)$
Signature sort (1995)	$O(n)$	$O(n)$	Only for $w = \Omega(\log^{2+\varepsilon} n)$
Han & Thorup (2002)	$O(n\sqrt{\log(w/\log n)})$	$O(n)$	Randomized with mulitplication
Thorup (2002)	$O(n \log \log n)$	$O(n)$	Randomized, AC^0

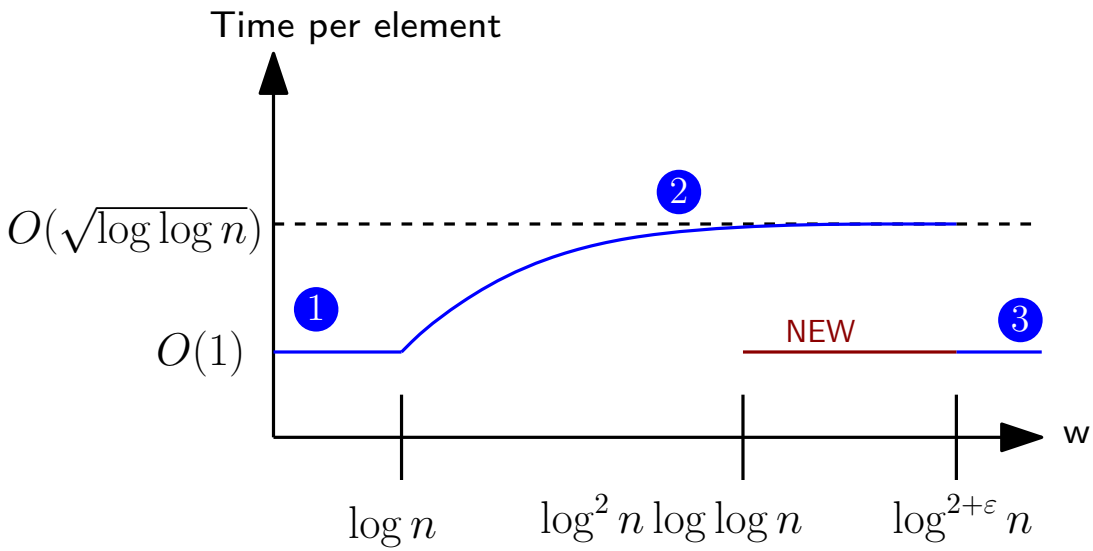
w is the number of bits per word.

Integer Sorting

Algorithm	Time	Space	Notes
Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	$O(n)$	optimal for $w = O(\log n)$
vEB sort (1975)	$O(n \log w)$	$O(2^w)$	Only $O(n)$ space with Y-fast trie
Kirkpatrick & Reisch (1984)	$O(n \log \frac{w}{\log n})$		Only asymptotically better than vEB when $w = O(\log n)$
Signature sort (1995)	$O(n)$	$O(n)$	Only for $w = \Omega(\log^{2+\varepsilon} n)$
Han & Thorup (2002)	$O(n\sqrt{\log(w/\log n)})$	$O(n)$	Randomized with mulitplication
Thorup (2002)	$O(n \log\log n)$	$O(n)$	Randomized, AC^0
Han (2004)	$O(n \log\log n)$	$O(n)$	Deterministic, with multiplication

w is the number of bits per word.

Integer Sorting



	Algorithm	Time	Notes
1	Radix sort (1887?)	$O(n \cdot \frac{w}{\log n})$	Optimal for $w = O(\log n)$
2	Signature sort (1995)	$O(n)$	Only for $w = \Omega(\log^{2+\varepsilon} n)$
3	Han & Thorup (2002)	$O\left(n\sqrt{\log(w/\log n)}\right)$	Randomized with mulitplication

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing
5. State and University Library

Text Indexing

Text Indexing

Most of us daily use services like Google, Bing, etc.

Text Indexing

Most of us daily use services like Google, Bing, etc.

Text Indexing

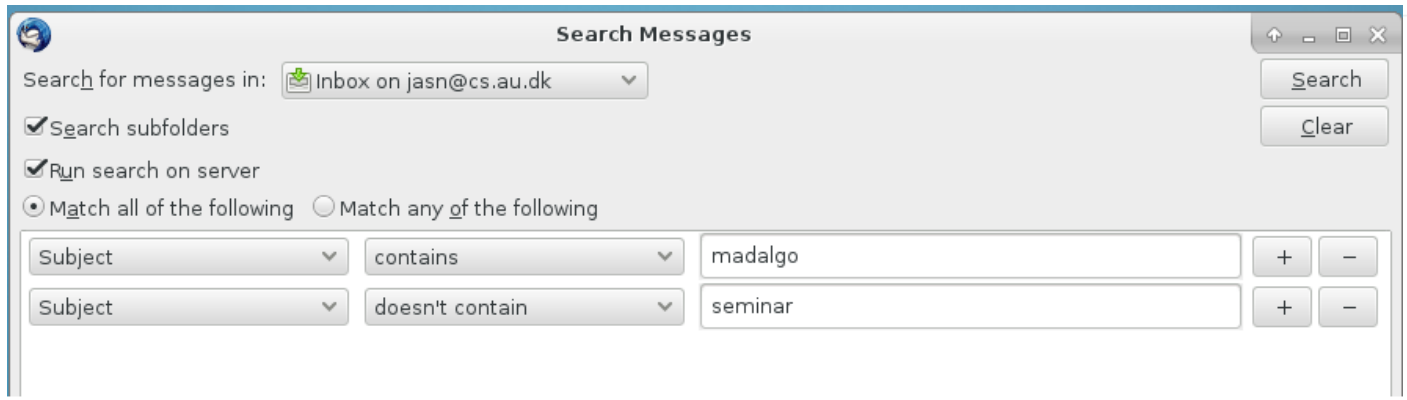
Most of us daily use services like Google, Bing, etc.

We want documents that contain both terms

Text Indexing

Most of us daily use services like Google, Bing, etc.

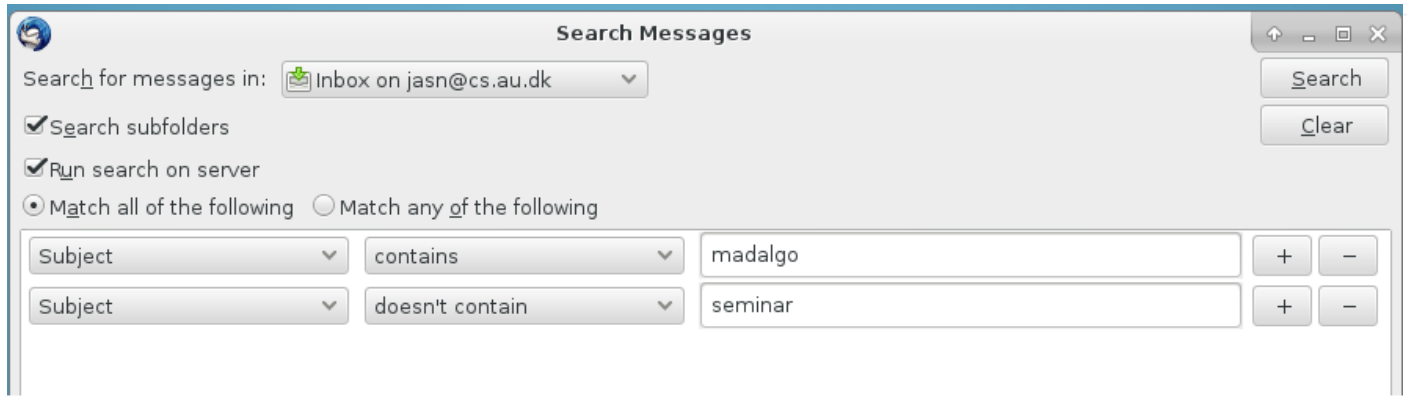
We want documents that contain both terms



Text Indexing

Most of us daily use services like Google, Bing, etc.

We want documents that contain both terms



Find all emails sent to madalgo that are not seminars

Text Indexing - Two Patterns

Text Indexing - Two Patterns

Problem definition:

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

when given two text strings P_1, P_2 (online)

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

when given two text strings P_1, P_2 (online)

we can report all d_i where both P_1 and P_2 occur

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

when given two text strings P_1, P_2 (online)

we can report all d_i where both P_1 and P_2 occur

(The two-pattern problem)

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

when given two text strings P_1, P_2 (online)

we can report all d_i where both P_1 and P_2 occur

(The two-pattern problem)

we can report all d_i where both P_1 occurs but P_2 does not occur

Text Indexing - Two Patterns

Problem definition:

Given documents $\mathcal{D} = d_1, d_2, \ldots, d_D$

preprocess and store $\mathcal{D}$ such that

when given two text strings P_1, P_2 (online)

we can report all d_i where both P_1 and P_2 occur

(The two-pattern problem)

we can report all d_i where both P_1 occurs but P_2 does not occur

(The forbidden pattern problem)

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^{\alpha} + t$	Ferragina et al. '03

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10
Two-Pattern	n	$\sqrt{nt \log n} \log n + t$	Hon et al. '10

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10
Two-Pattern	n	$\sqrt{nt \log n} \log n + t$	Hon et al. '10
Forbidden Pattern (FP)	$n^{3/2} / \log n$	$\sqrt{n} + t$	Fischer et al. '12

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10
Two-Pattern	n	$\sqrt{nt \log n} \log n + t$	Hon et al. '10
Forbidden Pattern (FP)	$n^{3/2} / \log n$	$\sqrt{n} + t$	Fischer et al. '12
Forbidden Pattern	n	$\sqrt{nt \log n} \log^2 n + t$	Hon et al. '12

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10
Two-Pattern	n	$\sqrt{nt \log n} \log n + t$	Hon et al. '10
Forbidden Pattern (FP)	$n^{3/2} / \log n$	$\sqrt{n} + t$	Fischer et al. '12
Forbidden Pattern	n	$\sqrt{nt \log n} \log^2 n + t$	Hon et al. '12
Both 2P and FP	n	$\sqrt{nt \log n} \log^\varepsilon n + t$	**New**

Text Indexing - Two Patterns

Problem	Space	Query Time	Authors
Two-Pattern (2P)	$n^{2-\alpha} \log^c n$	$n^\alpha + t$	Ferragina et al. '03
Two-Pattern	$n \log n$	$\sqrt{nt \log n} \log^2 n + t$	Cohen and Porat '10
Two-Pattern	n	$\sqrt{nt \log n} \log n + t$	Hon et al. '10
Forbidden Pattern (FP)	$n^{3/2} / \log n$	$\sqrt{n} + t$	Fischer et al. '12
Forbidden Pattern	n	$\sqrt{nt \log n} \log^2 n + t$	Hon et al. '12
Both 2P and FP	n	$\sqrt{nt \log n} \log^\varepsilon n + t$	**New**

Is $\sqrt{n}$ query time really necessary?

Text Indexing - Two Patterns

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

Text Indexing - Two Patterns

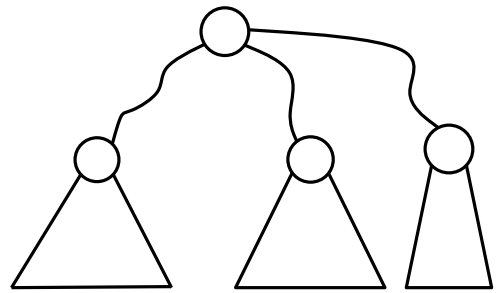
Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

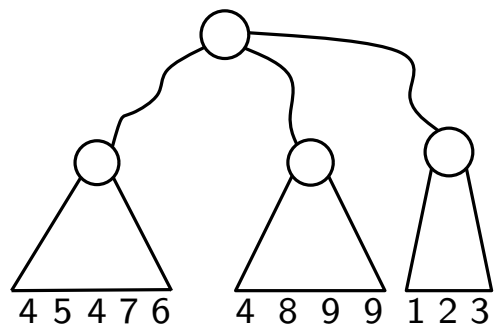
$$S = d_1\$d_2\$\cdots\$d_D\$$$



Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$

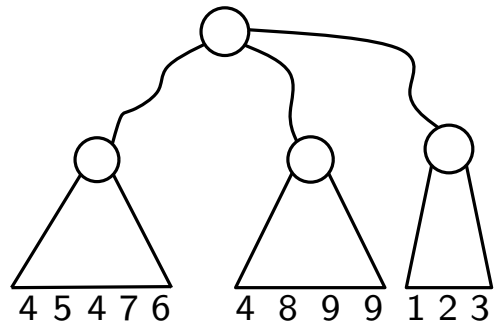


Annotate each leaf with the document id the suffx starts in

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



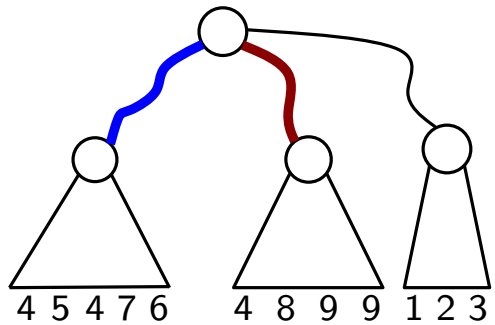
Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



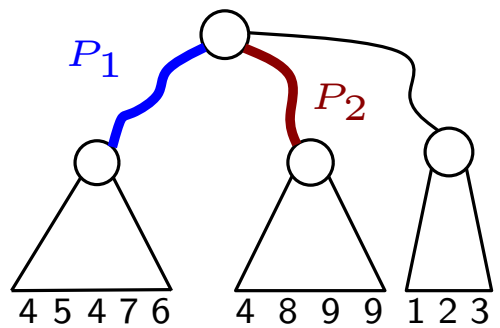
Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



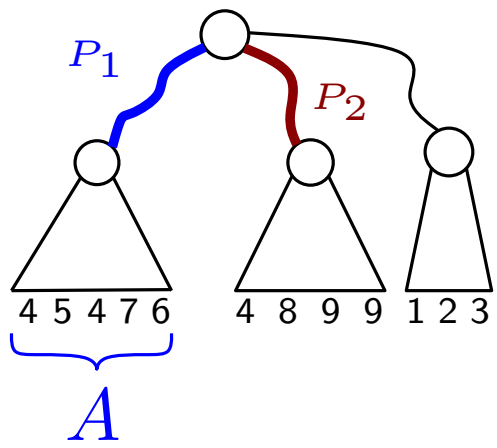
Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



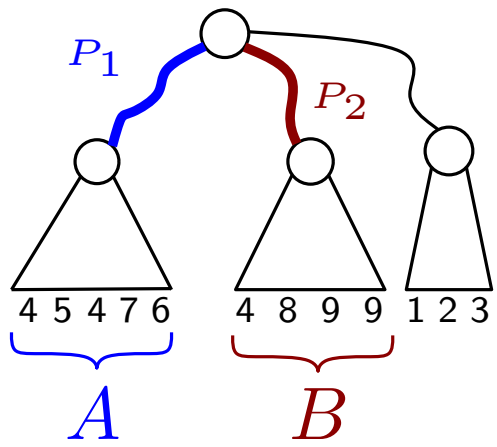
Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



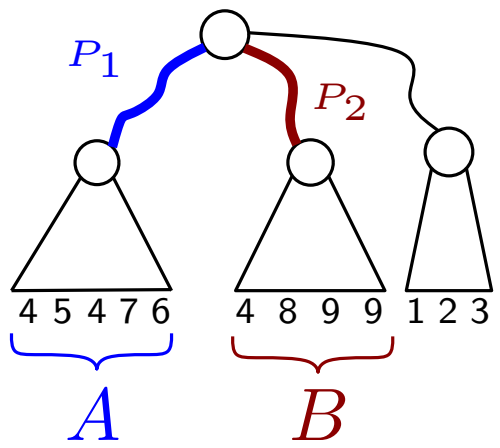
Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



Annotate each leaf with the document id the suffx starts in

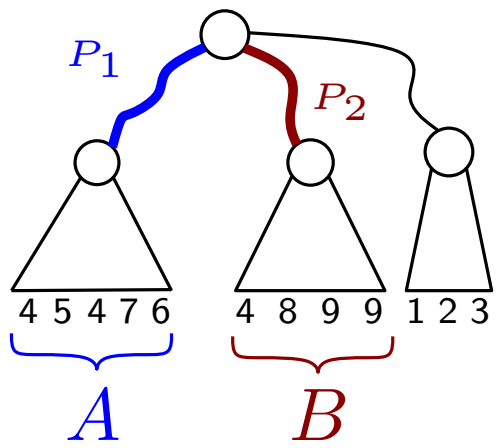
Query: P_1 and P_2

Report: $A \cap B$

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



Annotate each leaf with the document id the suffx starts in

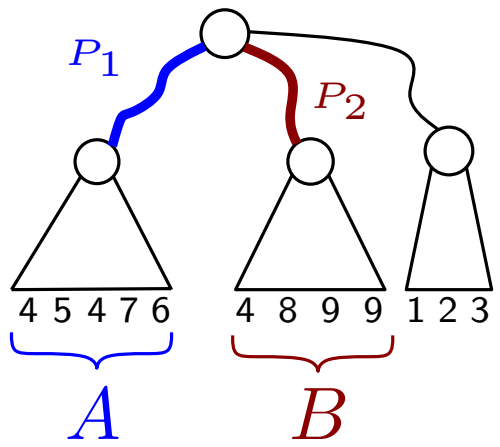
Query: P_1 and P_2

Report: $A \cap B = \{4\}$

Text Indexing - Two Patterns

Build suffix tree on concatenation of the documents

$$S = d_1\$d_2\$\cdots\$d_D\$$$



Annotate each leaf with the document id the suffx starts in

Query: P_1 and P_2

Report: $A \cap B = \{4\}$

set intersection is usually hard

Text Indexing - Two Patterns

Text Indexing - Two Patterns

Problem	Lower Bound	Notes

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)
Two-Pattern	$P(n) + n^{3/4+\varepsilon}Q(n) = \Omega(n^{4/3-\varepsilon})$	Kopelowitz et al (3SUM)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)
Two-Pattern	$P(n) + n^{3/4+\varepsilon}Q(n) = \Omega(n^{4/3-\varepsilon})$	Kopelowitz et al (3SUM)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)
Two-Pattern	$P(n) + n^{3/4+\varepsilon}Q(n) = \Omega(n^{4/3-\varepsilon})$	Kopelowitz et al (3SUM)
Both 2P and FP	$S(n)Q(n) = \Omega(n^{2-o(1)})$	Pointer Machine Model (**new**)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)
Two-Pattern	$P(n) + n^{3/4+\varepsilon}Q(n) = \Omega(n^{4/3-\varepsilon})$	Kopelowitz et al (3SUM)
Both 2P and FP	$S(n)Q(n) = \Omega(n^{2-o(1)})$	Pointer Machine Model (**new**)
Both 2P and FP	$S(n)Q(n)^2 = \Omega(n^{2-o(1)})$	Semi group model (**new**)

Text Indexing - Two Patterns

Problem	Lower Bound	Notes
Both 2P and FP	$P(n) + nQ(n) = \Omega(n^{\omega})$	ω is matrix multiplication constant (**new**)
Two-Pattern	$P(n) + n^{1+\varepsilon}Q(n) = \Omega(n^{4/3-o(1)})$	Kopelowitz et al (3SUM)
Two-Pattern	$P(n) + n^{3/4+\varepsilon}Q(n) = \Omega(n^{4/3-\varepsilon})$	Kopelowitz et al (3SUM)
Both 2P and FP	$S(n)Q(n) = \Omega(n^{2-o(1)})$	Pointer Machine Model (**new**)
Both 2P and FP	$S(n)Q(n)^2 = \Omega(n^{2-o(1)})$	Semi group model (**new**)
Both 2P and FP	$Q(n) = (nt)^{\frac{1}{2}-\alpha} \Rightarrow$ $S(n) = \Omega(n^{\frac{1+6\alpha}{1+2\alpha}-o(1)})$	Pointer Machine Model t is the output size (**new**)

Text Indexing - How to prove the lower bounds?

We use two different techniques:

1) Reduction from Matrix Multiplication

2) Find a hard input for a Pointer Machine Data Structure

---

1) Given two matrices, produce a set of documents and queries such that the product of the two matrices can be found

Text Indexing - How to prove the lower bounds?

Text Indexing - How to prove the lower bounds?

$$\begin{matrix} & A & \\ \end{matrix}\begin{bmatrix} T & F & T \\ T & F & F \\ T & T & T \end{bmatrix} \times \begin{matrix} & B & \\ \end{matrix}\begin{bmatrix} F & F & T \\ F & T & T \\ T & F & T \end{bmatrix} = \begin{matrix} & C & \\ \end{matrix}\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t.
$A_{i,\ell} = B_{\ell,j} = T$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$$d_1^A = 1\#2\#3\#$$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$$d_1^A = 1\#2\#3\#$$

$$d_2^A = 3\#$$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$$d_1^A = 1\#2\#3\#$$

$$d_2^A = 3\#$$

$$d_3^A = 1\#3\#$$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \\ 4 & 5 & 6 \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$

$d_3^A = 1\#3\#$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$ $\quad$ $d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ \boxed{T} & \boxed{F} & \boxed{T} \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$$d_1^A = 1\#2\#3\# \qquad d_1^B = 6\#$$

$$d_2^A = 3\# \qquad d_2^B = 5\#6\#$$

$$d_3^A = 1\#3\# \qquad d_3^B = 4\#6\#$$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$ $\quad d_1 = 1\#2\#3\#6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$ $\quad d_2 = 3\#5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$ $\quad d_3 = 1\#3\#4\#6\#$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$$d_1^A = 1\#2\#3\# \qquad d_1^B = 6\#$$

$$d_2^A = 3\# \qquad d_2^B = 5\#6\#$$

$$d_3^A = 1\#3\# \qquad d_3^B = 4\#6\#$$

$$d_1 = 1\#2\#3\#6\#$$

$$d_2 = 3\#5\#6\#$$

$$d_3 = 1\#3\#4\#6\# \quad D$$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 1 $P_2 =$ 4

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 1 $P_2 =$ 4

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & & \\ & & \\ & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 1 $P_2 =$ 5

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{cc} T & F \\ & \\ & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$ $\quad$ $d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad$ $d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 1 $P_2 =$ 5

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{cc} T & F \\ & \\ & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$ $\quad$ $d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad$ $d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 1 $P_2 =$ 5

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & F & T \\ F & F & T \\ T & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$ $\quad$ $d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad$ $d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Query:

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & F & T \\ F & F & T \\ T & & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 3 $P_2 =$ 5

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & F & T \\ F & F & T \\ T & T & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 3 $P_2 =$ 5

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & F & T \\ F & F & T \\ T & T & \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad$ $d_1^B = 6\#$

$d_2^A = 3\#$ $\quad$ $d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad$ $d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 3 $P_2 =$ 6

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Text Indexing - How to prove the lower bounds?

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} A \\ \left[\begin{array}{ccc} T & F & T \\ T & F & F \\ T & T & T \end{array}\right] \end{array} \times \begin{array}{c} B \\ \left[\begin{array}{ccc} F & F & T \\ F & T & T \\ T & F & T \end{array}\right] \end{array} = \begin{array}{c} C \\ \left[\begin{array}{ccc} T & F & T \\ F & F & T \\ T & T & T \end{array}\right] \end{array}$$

$C_{i,j} = T \Leftrightarrow \exists \ell$ s.t. $A_{i,\ell} = B_{\ell,j} = T$

$d_1^A = 1\#2\#3\#$ $\quad d_1^B = 6\#$

$d_2^A = 3\#$ $\quad d_2^B = 5\#6\#$

$d_3^A = 1\#3\#$ $\quad d_3^B = 4\#6\#$

$d_1 = 1\#2\#3\#6\#$

$d_2 = 3\#5\#6\#$

$d_3 = 1\#3\#4\#6\#$ D

Query: $P_1 =$ 3 $P_2 =$ 6

$i \in d_\ell \Leftrightarrow A_{i,\ell} = T$

$j + 3 \in d_\ell \Leftrightarrow B_{\ell,j} = T$

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing

Overview

1. Algorithms and Data Structures
   (a) Abstract description
   (b) Example
2. Implicit Data Structures
   (a) Finger Search
   (b) Priority Queues
3. Integer Sorting
4. Text Indexing
5. State and University Library

State and University Library

State and University Library

State and University Library

State and University Library

State and University Library

"Old Days"

State and University Library

"Old Days"

State and University Library

"Old Days"

State and University Library

"Old Days"

State and University Library

"Old Days"

State and University Library

"Old Days"

EUROVISION
SONG CONTEST
COPENHAGEN 2014

State and University Library

"Old Days"

EUROVISION
SONG CONTEST
COPENHAGEN 2014

State and University Library

"Old Days"

EUROVISION
SONG CONTEST
COPENHAGEN 2014

LOTS of data

A nuisance in a collection

A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started

A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started

$\Rightarrow$ overlap

A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started
$\Rightarrow$ overlap

Annoying for customers

A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started
$\Rightarrow$ overlap

Annoying for customers

Task: find overlap and eliminate it

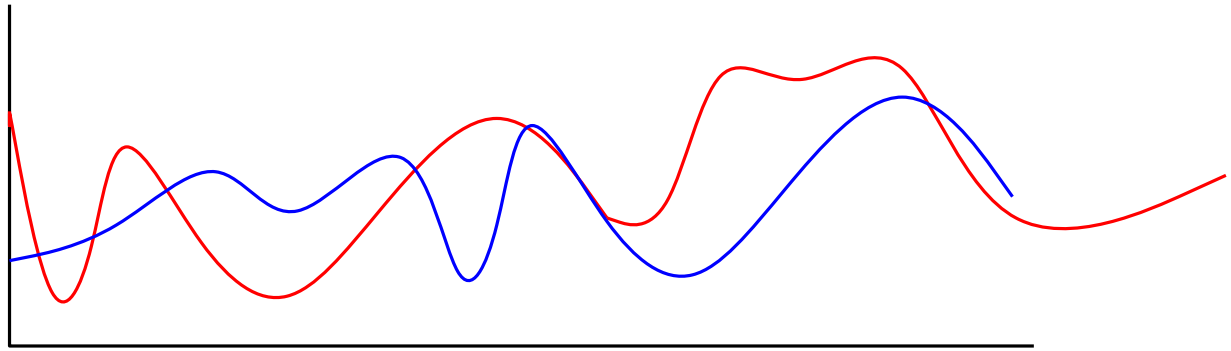
A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started
$\Rightarrow$ overlap

Annoying for customers

Task: find overlap and eliminate it



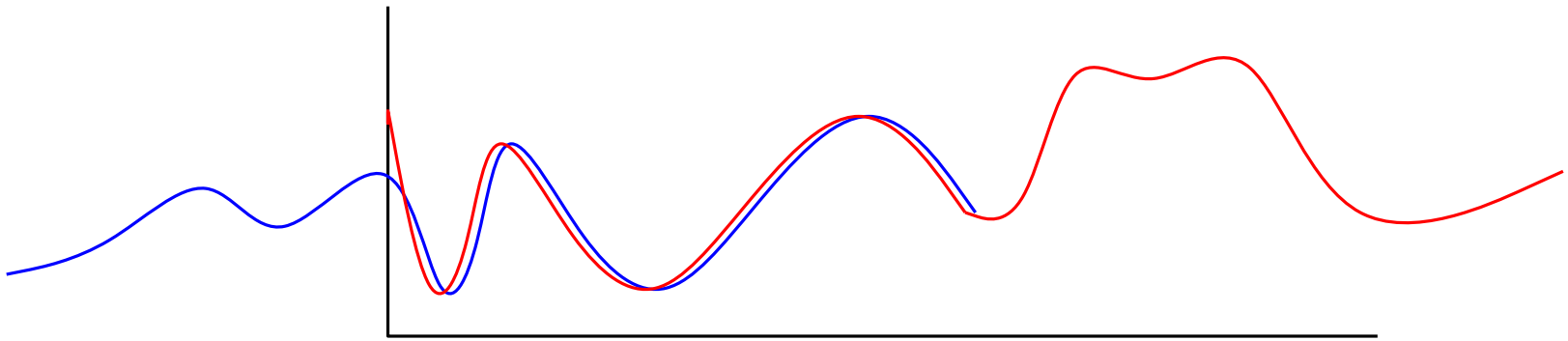
A nuisance in a collection

Danish Radio collection 2 hour chunks

Before one runs out another is started
$\Rightarrow$ overlap

Annoying for customers

Task: find overlap and eliminate it



A nuisance in a collection

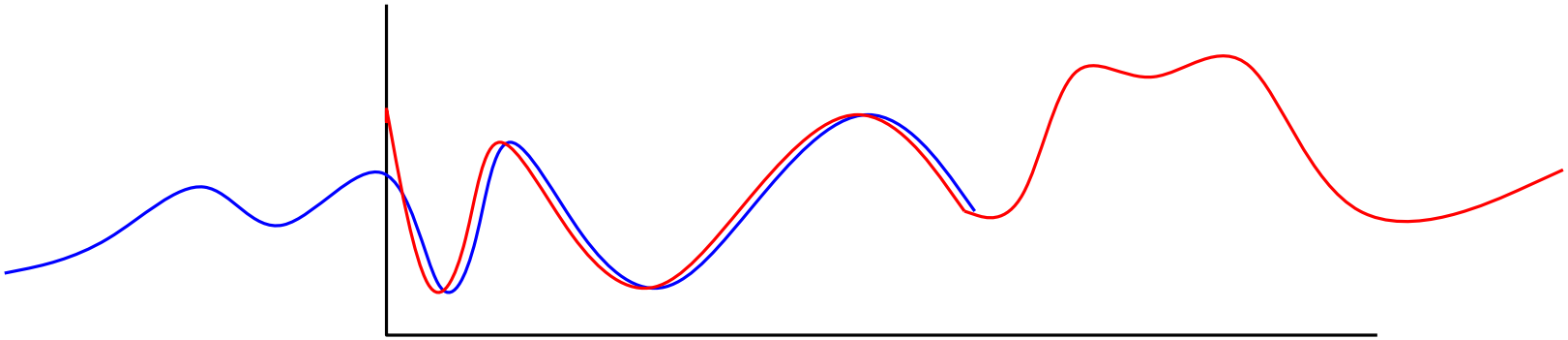
Danish Radio collection 2 hour chunks

Before one runs out another is started
$\Rightarrow$ overlap

Annoying for customers

Task: find overlap and eliminate it

Use Cross Correlation.



Thank You