Succinct Data Structures

Auto-completion as our target application

Rossano Venturini

rossano@di.unipi.it

auto trader

autotrader
autozone
auto loan calculator
autodesk

Learn more

+Rossano Share

Cookies help us deliver our services. By using our services, you agree to our use of cookies.

OK Learn more

New Cars, Used Cars - Find Cars at AutoTrader.com

www.autotrader.com/

Find used cars and new cars for sale at **AutoTrader**.com. With millions of cars, finding your next new car or used car and the car reviews and information you're ...

Used Car Research - Find Cars for Sale - Certified Pre-Owned Car - Sell a Car

Auto Trader UK – New & used cars for sale

www.autotrader.co.uk/

The UK's #1 site to buy and sell new and used cars, bikes, vans, trucks and caravans with over 350000 vehicles online. Check Car news, reviews and obtain ...

Used cars - Vans - Bikes - Used cars UK

Used cars - Find a used car for sale on Auto Trader

www.autotrader.co.uk/used-cars

Used cars for sale on **Auto Trader**, find the right used car for you at the UK's No.1 destination for motorists.

Used Cars for Sale – autoTRADER.ca – Auto Classifieds

www.autotrader.ca/

Visit Canada's largest auto classifieds site for new and used cars for sale. Buy or sell your car for free, compare car prices, plus reviews, news & pictures.

Auto Trader South Africa - Used Cars for sale

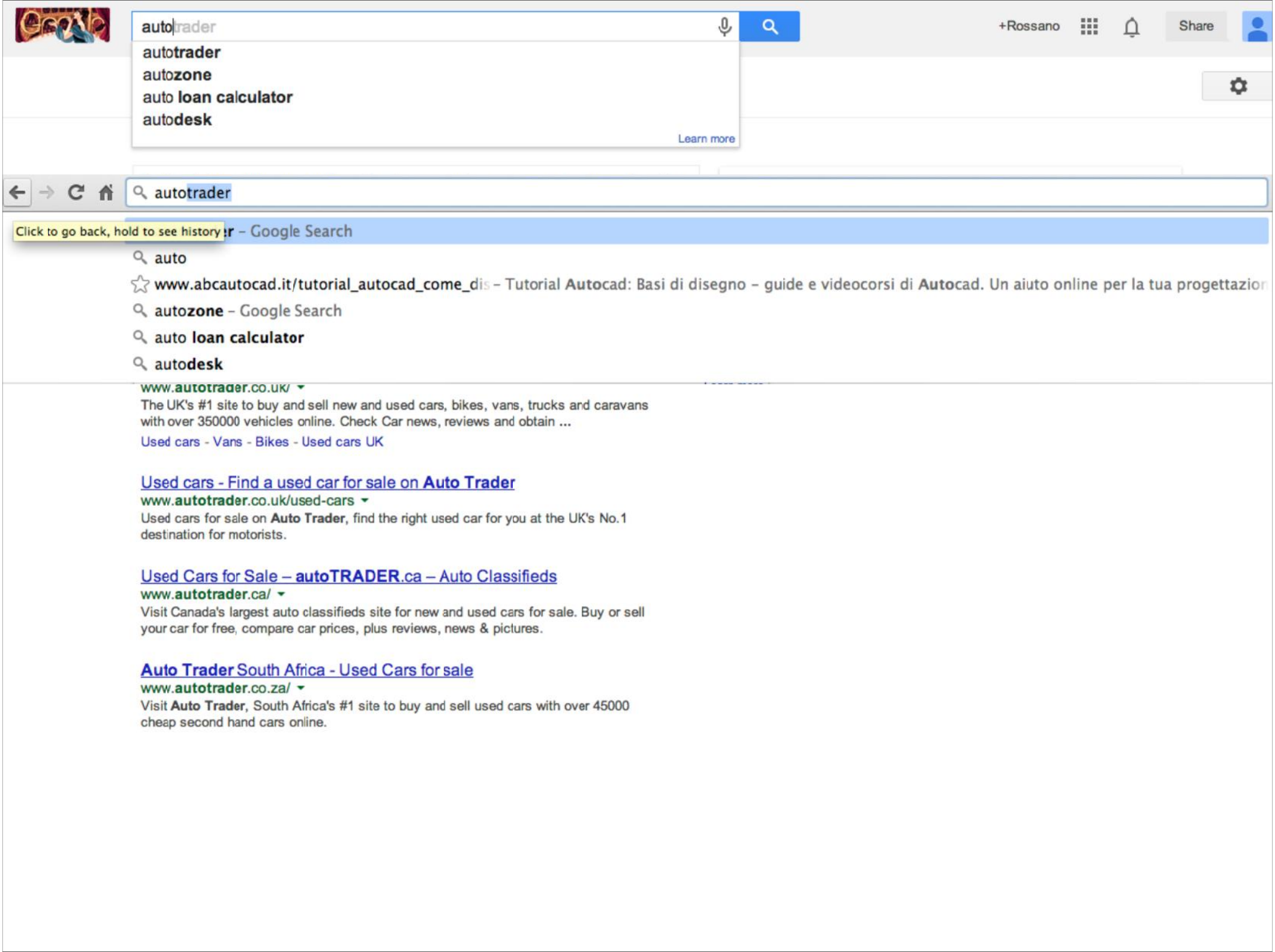
www.autotrader.co.za/

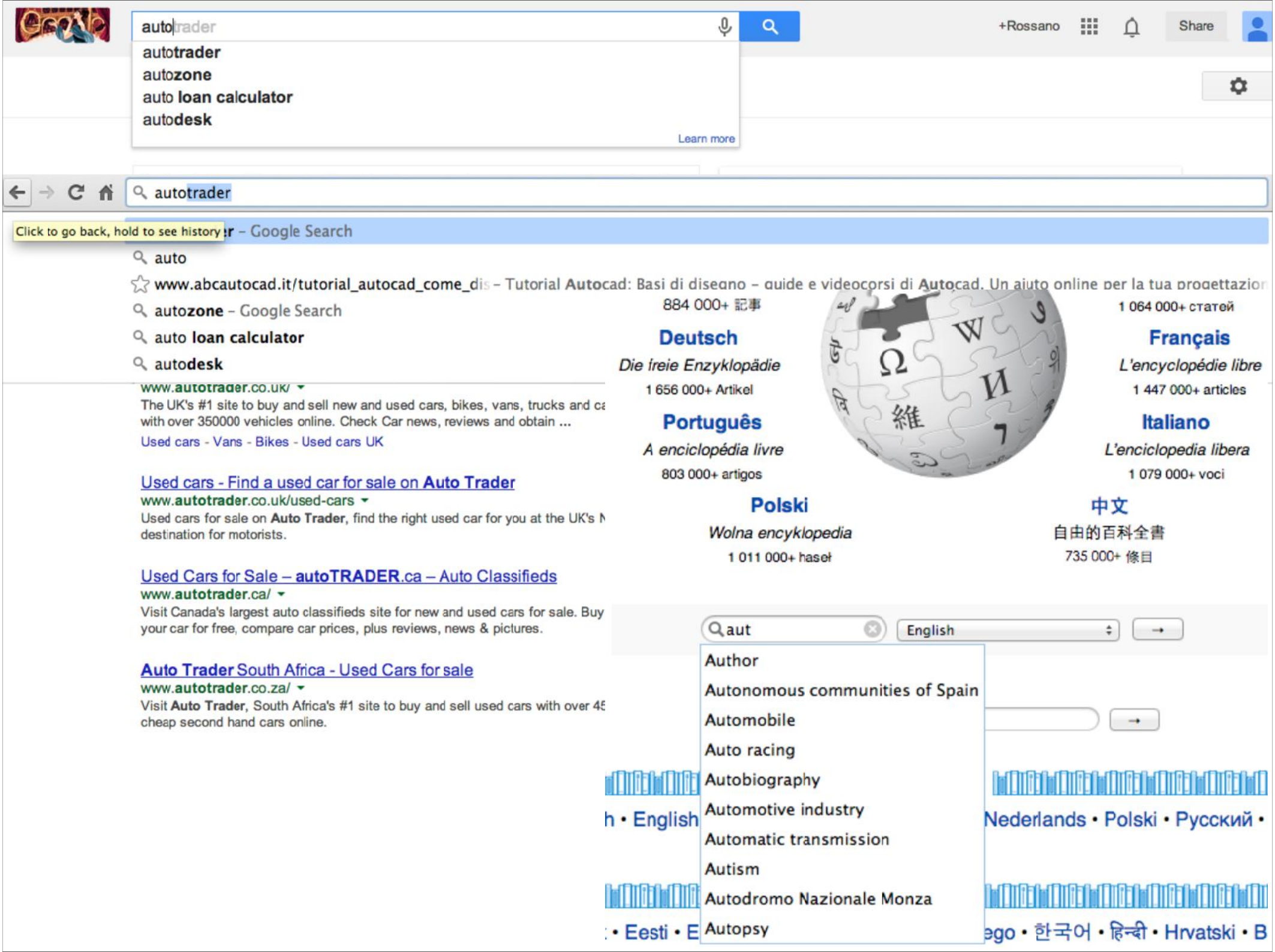
Visit **Auto Trader**, South Africa's #1 site to buy and sell used cars with over 45000 cheap second hand cars online.

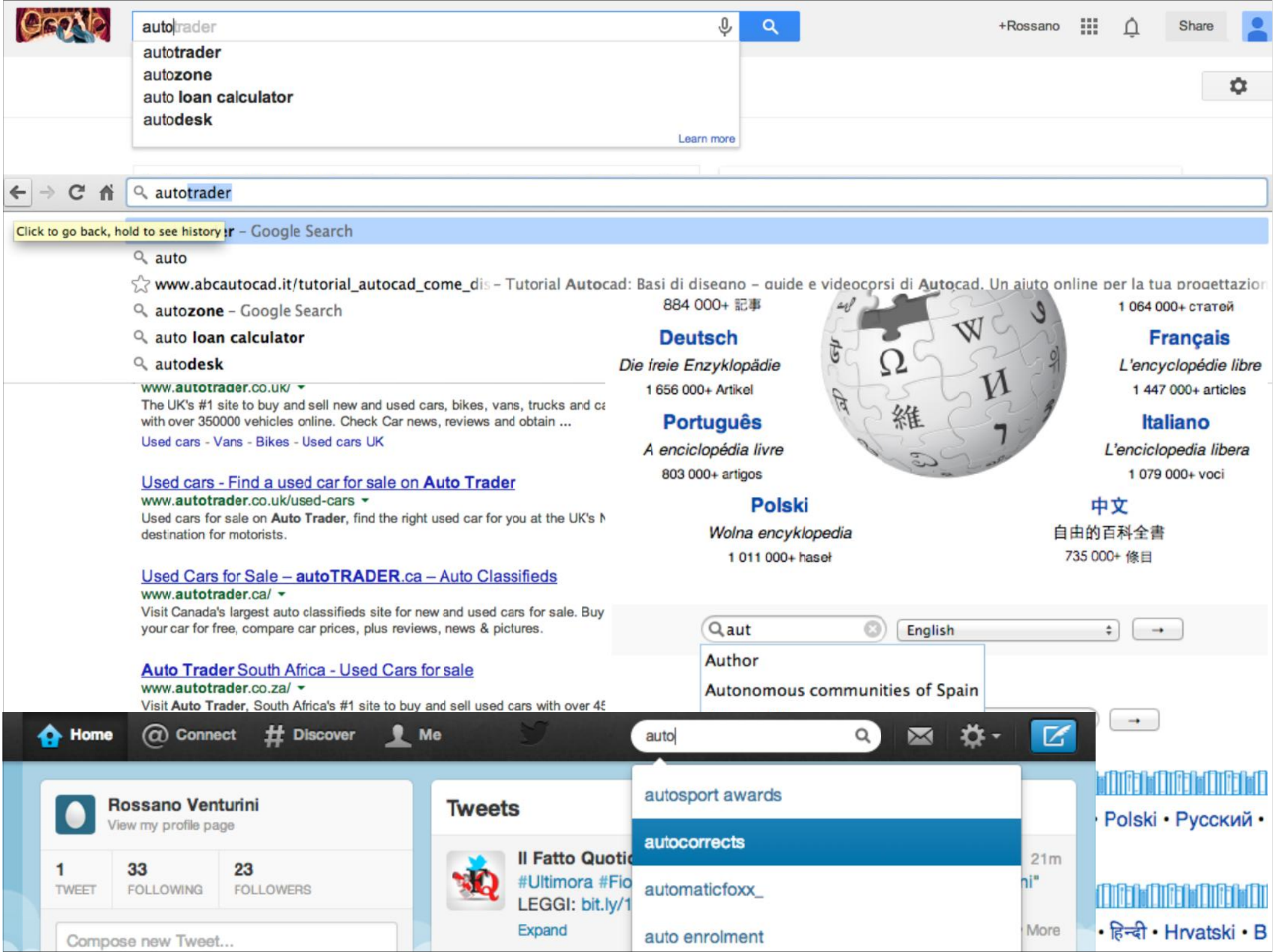
See results about

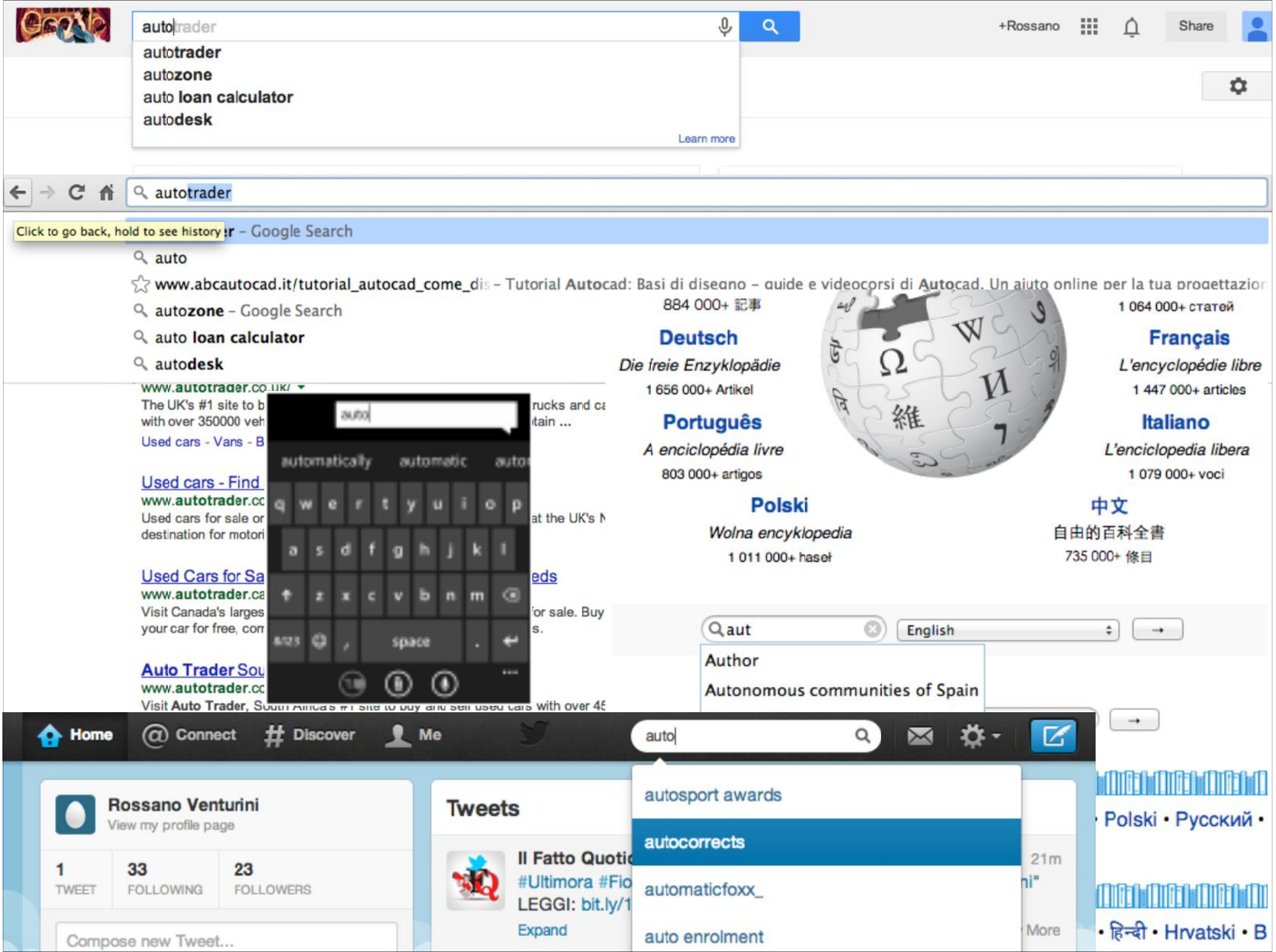
AutoTrader.com
Corporation
AutoTrader.com, Inc. is an online marketplace for car shoppers and sellers. It aggregates millions of new, ...

Feedback/More info


auto trader
autotrader
autozone
auto loan calculator
autodesk
Learn more
+Rossano
Share
autotrader
Click to go back, hold to see history
r – Google Search
auto
www.abcautocad.it/tutorial_autocad_come_dis – Tutorial Autocad: Basi di disegno – guide e videocorsi di Autocad. Un aiuto online per la tua progettazion
autozone – Google Search
auto loan calculator
autodesk
www.autotrader.co.uk/
The UK's #1 site to buy and sell new and used cars, bikes, vans, trucks and caravans with over 350000 vehicles online. Check Car news, reviews and obtain ...
Used cars - Vans - Bikes - Used cars UK
Used cars - Find a used car for sale on Auto Trader
www.autotrader.co.uk/used-cars
Used cars for sale on Auto Trader, find the right used car for you at the UK's No.1 destination for motorists.
Used Cars for Sale – autoTRADER.ca – Auto Classifieds
www.autotrader.ca/
Visit Canada's largest auto classifieds site for new and used cars for sale. Buy or sell your car for free, compare car prices, plus reviews, news & pictures.
Auto Trader South Africa - Used Cars for sale
www.autotrader.co.za/
Visit Auto Trader, South Africa's #1 site to buy and sell used cars with over 45000 cheap second hand cars online.


autotrader
autotrader
autozone
auto loan calculator
autodesk
Learn more
+Rossano
Share
autotrader
Click to go back, hold to see history
– Google Search
auto
www.abcautocad.it/tutorial_autocad_come_dis – Tutorial Autocad: Basi di disegno – guide e videocorsi di Autocad. Un aiuto online per la tua progettazion
autozone – Google Search
auto loan calculator
autodesk
www.autotrader.co.uk/
The UK's #1 site to buy and sell new and used cars, bikes, vans, trucks and ca
with over 350000 vehicles online. Check Car news, reviews and obtain ...
Used cars - Vans - Bikes - Used cars UK
Used cars - Find a used car for sale on Auto Trader
www.autotrader.co.uk/used-cars
Used cars for sale on Auto Trader, find the right used car for you at the UK's N
destination for motorists.
Used Cars for Sale – autoTRADER.ca – Auto Classifieds
www.autotrader.ca/
Visit Canada's largest auto classifieds site for new and used cars for sale. Buy
your car for free, compare car prices, plus reviews, news & pictures.
Auto Trader South Africa - Used Cars for sale
www.autotrader.co.za/
Visit Auto Trader, South Africa's #1 site to buy and sell used cars with over 45
cheap second hand cars online.
884 000+ 記事
Deutsch
Die freie Enzyklopädie
1 656 000+ Artikel
Português
A enciclopédia livre
803 000+ artigos
Polski
Wolna encyklopedia
1 011 000+ haseł
1 064 000+ статей
Français
L'encyclopédie libre
1 447 000+ articles
Italiano
L'enciclopedia libera
1 079 000+ voci
中文
自由的百科全書
735 000+ 條目
aut
English
Author
Autonomous communities of Spain
Automobile
Auto racing
Autobiography
Automotive industry
Automatic transmission
Autism
Autodromo Nazionale Monza
Autopsy
h • English
Nederlands • Polski • Русский •
• Eesti • E
ego • 한국어 • हिन्दी • Hrvatski • B


auto trader
autotrader
autozone
auto loan calculator
autodesk
Learn more
+Rossano
Share
autotrader
Click to go back, hold to see history
– Google Search
auto
www.abcautocad.it/tutorial_autocad_come_dis – Tutorial Autocad: Basi di disegno – guide e videocorsi di Autocad. Un aiuto online per la tua progettazion
autozone – Google Search
auto loan calculator
autodesk
www.autotrader.co.uk/
The UK's #1 site to buy and sell new and used cars, bikes, vans, trucks and ca
with over 350000 vehicles online. Check Car news, reviews and obtain ...
Used cars - Vans - Bikes - Used cars UK
Used cars - Find a used car for sale on Auto Trader
www.autotrader.co.uk/used-cars
Used cars for sale on Auto Trader, find the right used car for you at the UK's N
destination for motorists.
Used Cars for Sale – autoTRADER.ca – Auto Classifieds
www.autotrader.ca/
Visit Canada's largest auto classifieds site for new and used cars for sale. Buy
your car for free, compare car prices, plus reviews, news & pictures.
Auto Trader South Africa - Used Cars for sale
www.autotrader.co.za/
Visit Auto Trader, South Africa's #1 site to buy and sell used cars with over 45
884 000+ 記事
Deutsch
Die freie Enzyklopädie
1 656 000+ Artikel
Português
A enciclopédia livre
803 000+ artigos
Polski
Wolna encyklopedia
1 011 000+ haseł
1 064 000+ статей
Français
L'encyclopédie libre
1 447 000+ articles
Italiano
L'enciclopedia libera
1 079 000+ voci
中文
自由的百科全書
735 000+ 條目
aut
English
Author
Autonomous communities of Spain
Home
Connect
Discover
Me
auto
autosport awards
autocorrects
automaticfoxx_
auto enrolment
Rossano Venturini
View my profile page
1
TWEET
33
FOLLOWING
23
FOLLOWERS
Compose new Tweet...
Tweets
Il Fatto Quoti
#Ultimora #Fio
LEGGI: bit.ly/1
Expand
21m
More
Polski • Русский •
• हिन्दी • Hrvatski • B


auto trader
autotrader
autozone
auto loan calculator
autodesk
Learn more
+Rossano
Share
autotrader
Click to go back, hold to see history
– Google Search
auto
www.abcautocad.it/tutorial_autocad_come_dis – Tutorial Autocad: Basi di disegno – guide e videocorsi di Autocad. Un aiuto online per la tua progettazion
autozone – Google Search
auto loan calculator
autodesk
884 000+ 記事
Deutsch
Die freie Enzyklopädie
1 656 000+ Artikel
Português
A enciclopédia livre
803 000+ artigos
Polski
Wolna encyklopedia
1 011 000+ haseł
1 064 000+ статей
Français
L'encyclopédie libre
1 447 000+ articles
Italiano
L'enciclopedia libera
1 079 000+ voci
中文
自由的百科全書
735 000+ 條目
automatically
automatic
aut
aut
English
Author
Autonomous communities of Spain
Home
Connect
Discover
Me
auto
autosport awards
autocorrects
automaticfoxx_
auto enrolment
Rossano Venturini
View my profile page
1
TWEET
33
FOLLOWING
23
FOLLOWERS
Compose new Tweet...
Tweets
Il Fatto Quoti
#Ultimora #Fio
LEGGI: bit.ly/1
Expand
21m
More
Polski • Русский •
• हिन्दी • Hrvatski • B

Search the web using Google!

Google Search I'm feeling lucky

Special Searches Stanford Search Linux Search	Why use Google! Press about Google! Help! Company Info Jobs at Google Google! Logos Making Google! the Default	Get Google! updates monthly: your e-mail Subscribe Archive


Google!
BETA
Search the web using Google!
Google Search
I'm feeling lucky
Special Searches
Stanford Search
Linux Search
Why use Google!
Press about Google!
Help!
Company Info
Jobs at Google
Google! Logos
Making Google! the Default
Get Google!
updates monthly:
your e-mail
Subscribe
Archive
Copyright ©1999 Google Inc.

Dataset?

Dataset?

All the past queries

Dataset?

Searches?

All the past queries

Dataset?

Searches?

All the past queries

Prefix search

Dataset?

Searches?

Data structure?

All the past queries

Prefix search

Dataset?

Searches?

Data structure?

All the past queries

Prefix search

Trie

Dataset? All the past queries

Searches? Prefix search

Data structure? Trie

How to find top-k efficiently?

Trie

Trie

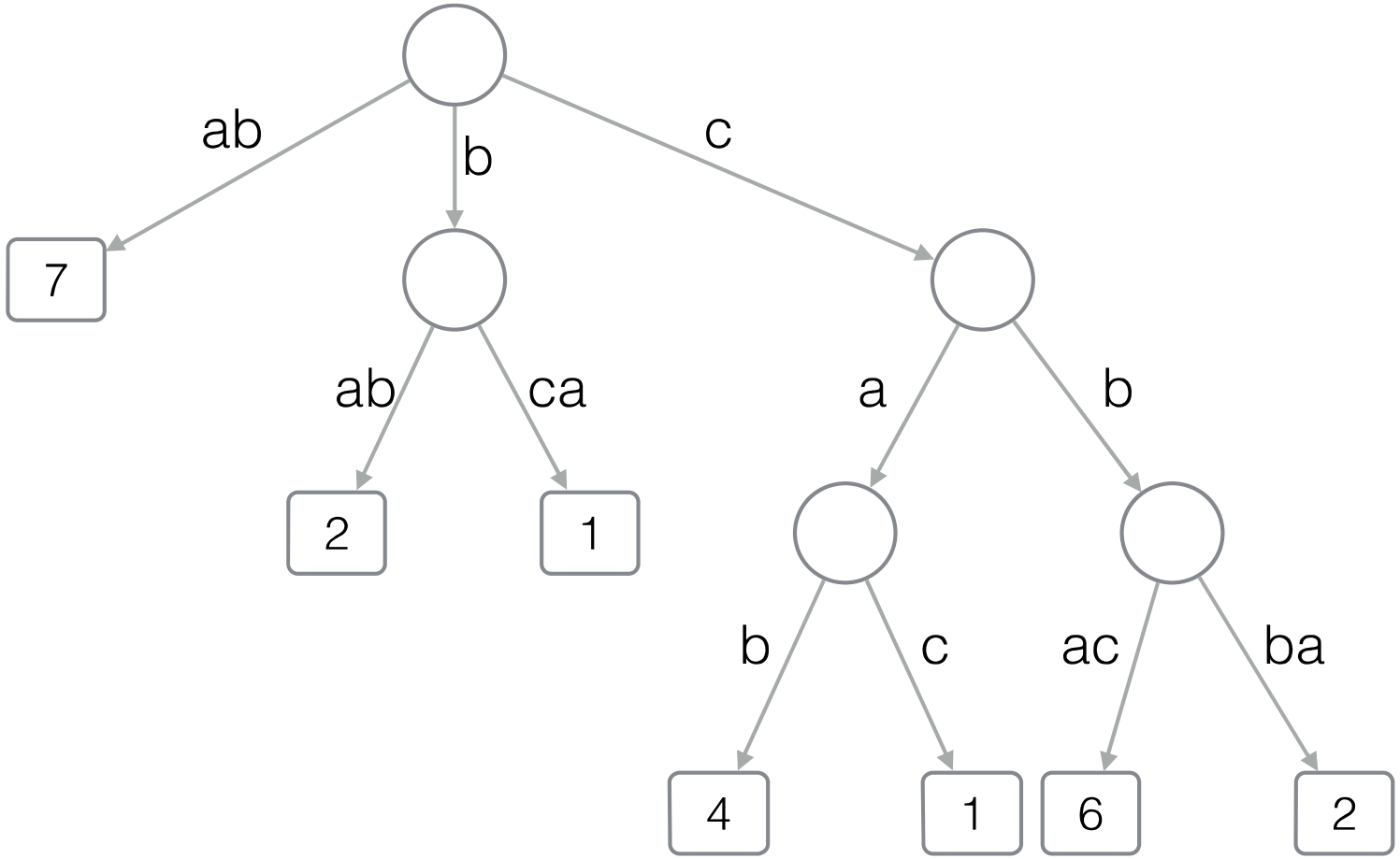
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

Trie

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

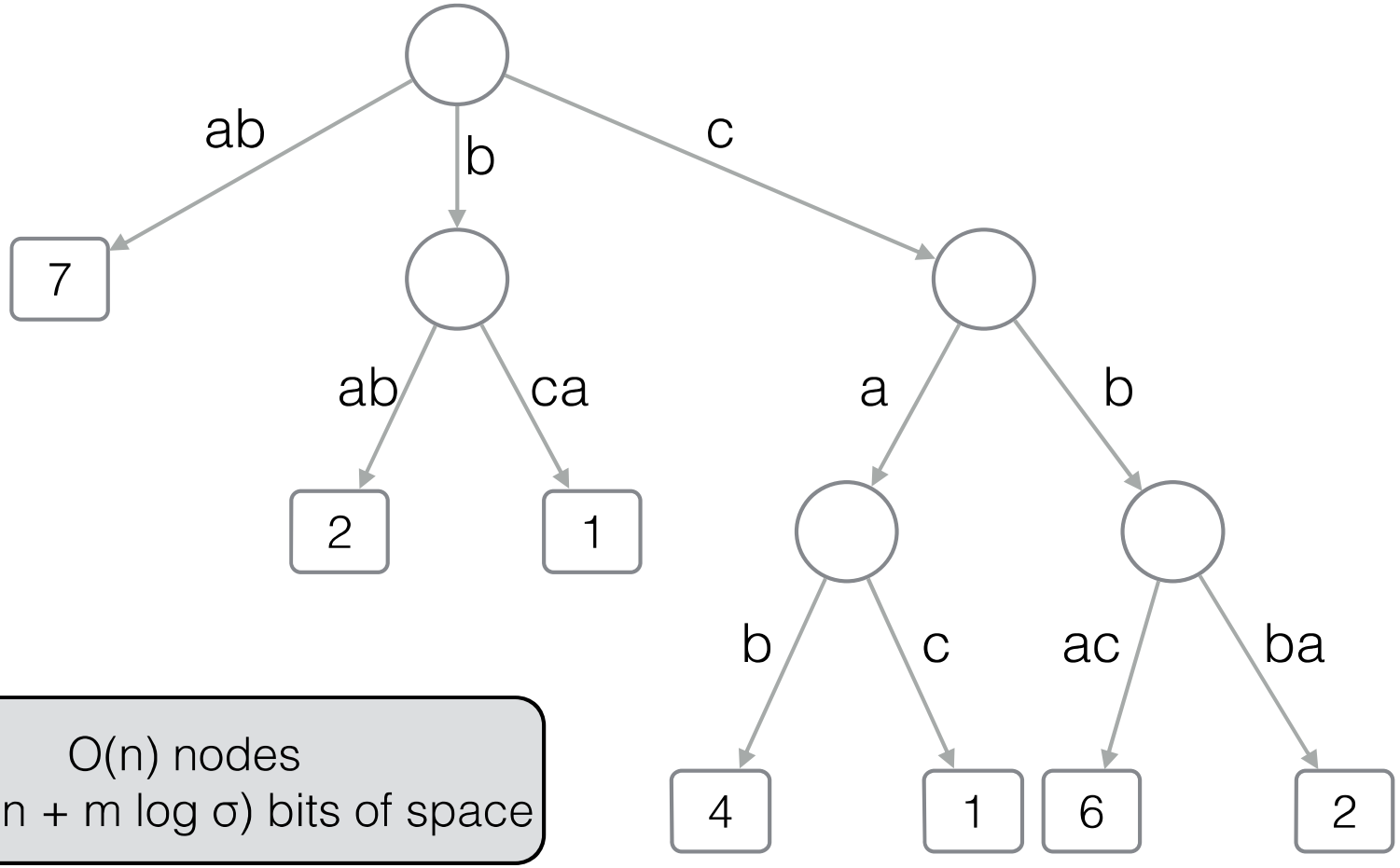
Trie



D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Trie

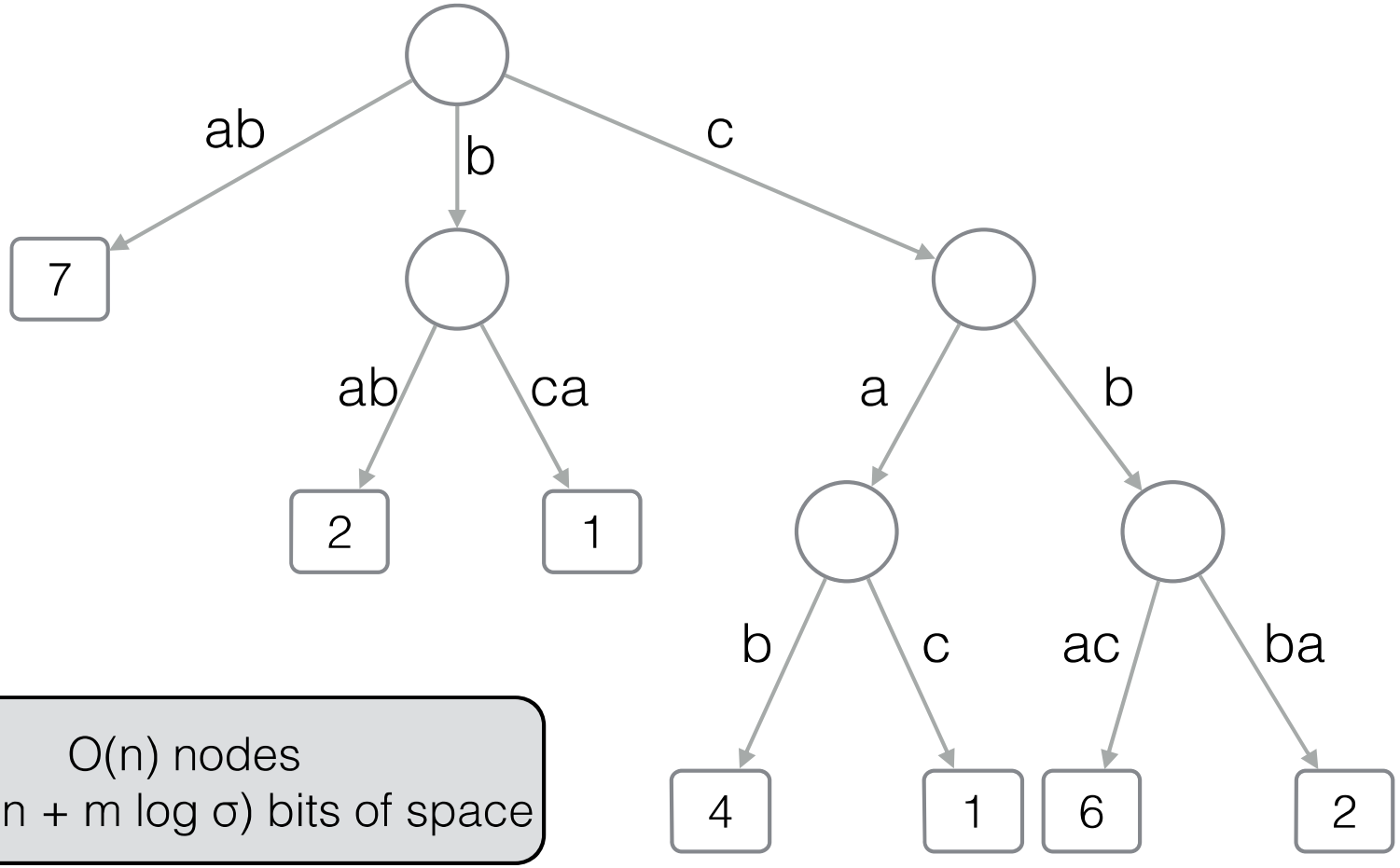


O(n) nodes
O(n log n + m log σ) bits of space

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Trie



O(n) nodes
O(n log n + m log σ) bits of space

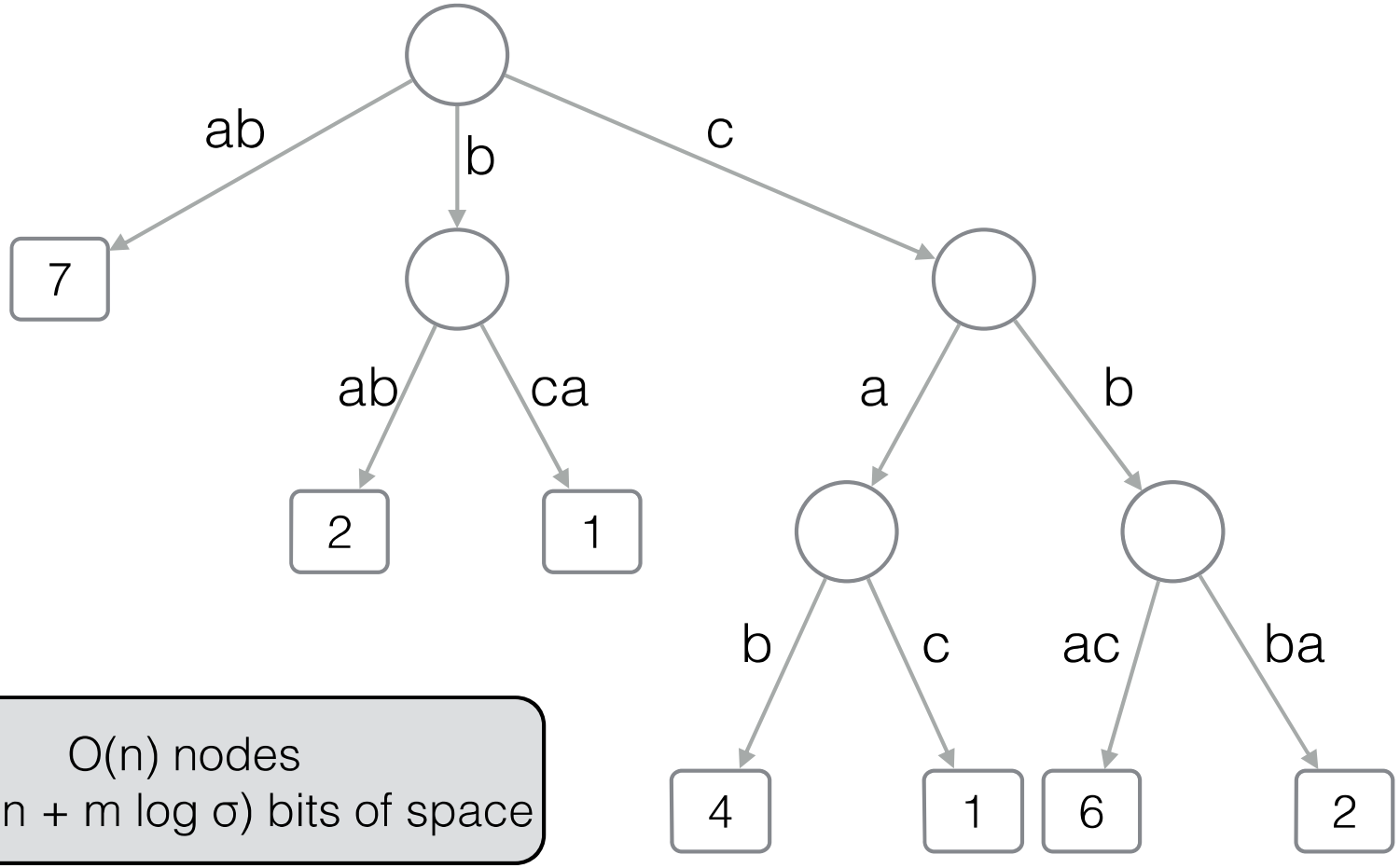
Find all the strings prefixed by any pattern P in O(|P|) time

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Trie

P = c



O(n) nodes
O(n log n + m log σ) bits of space

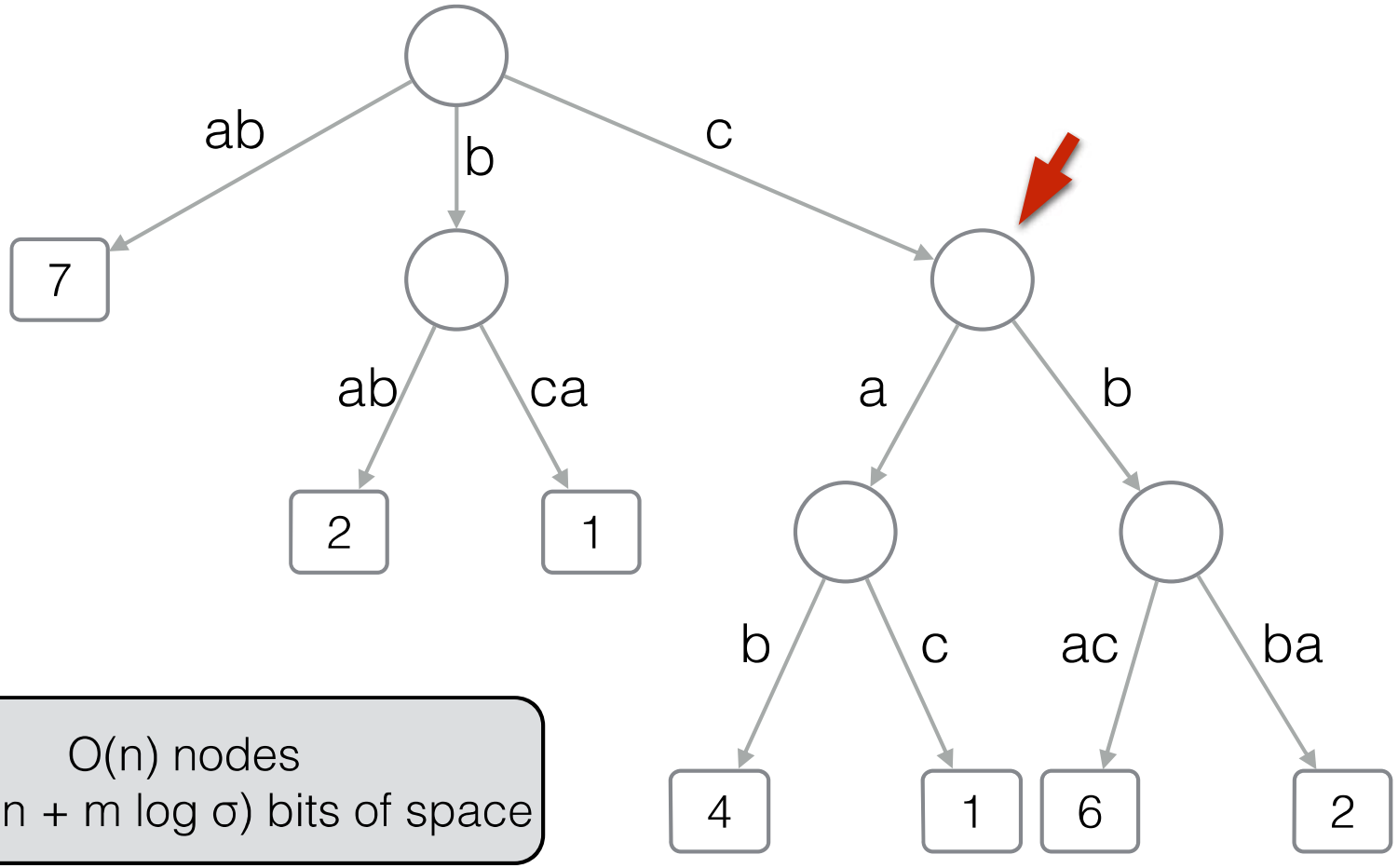
Find all the strings prefixed by any pattern P in O(|P|) time

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Trie

P = c



O(n) nodes
O(n log n + m log σ) bits of space

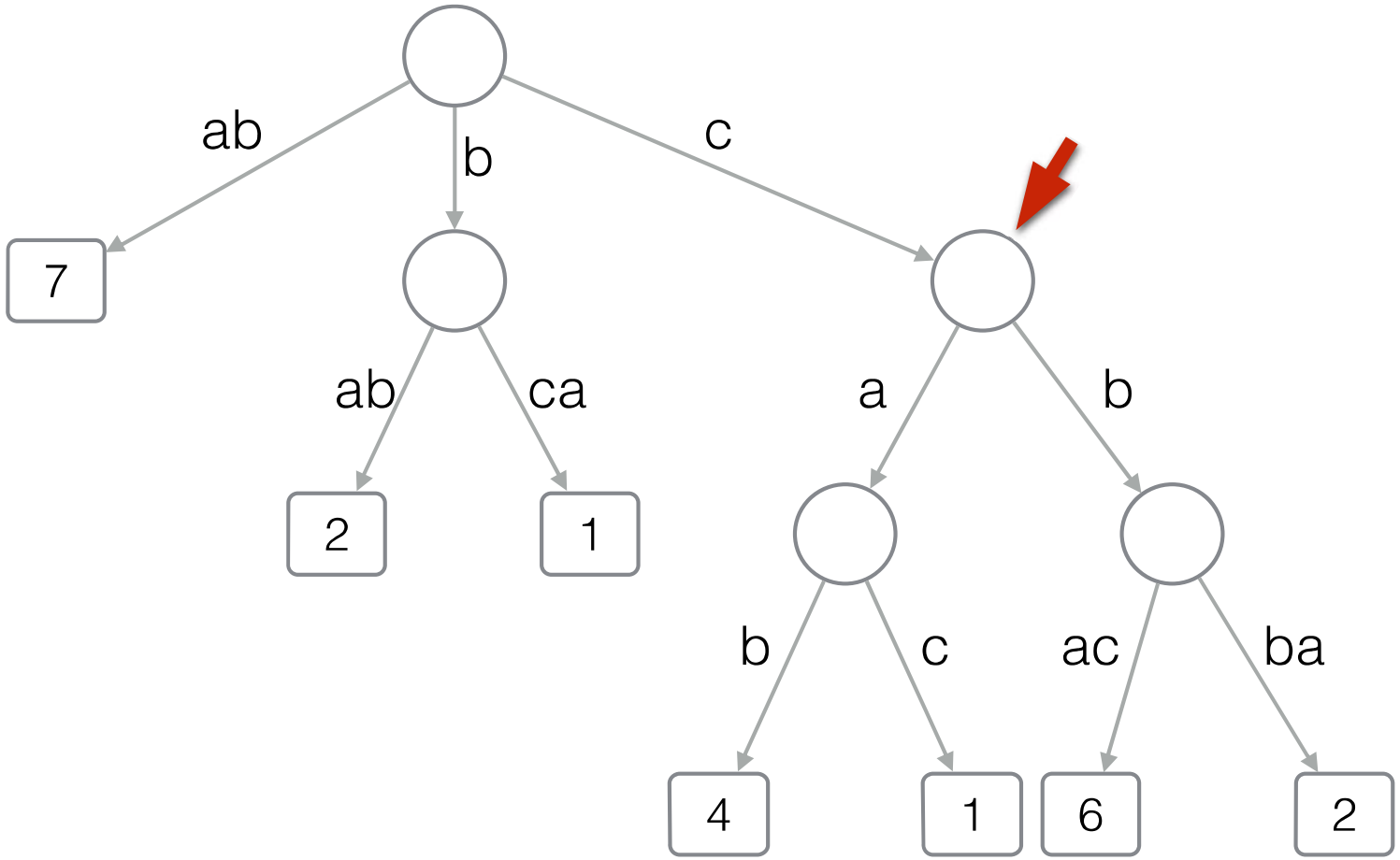
Find all the strings prefixed by any pattern P in O(|P|) time

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c



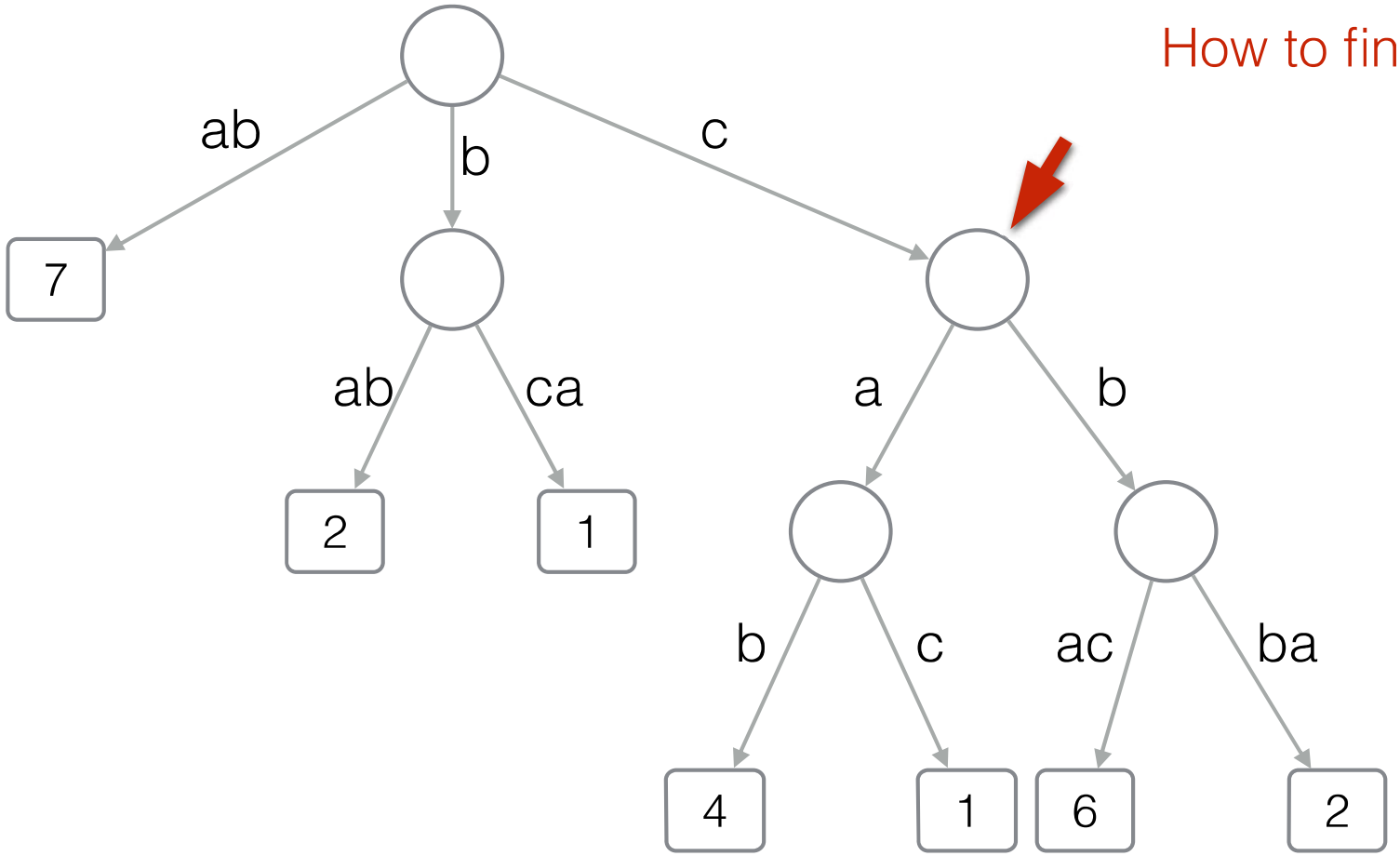
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



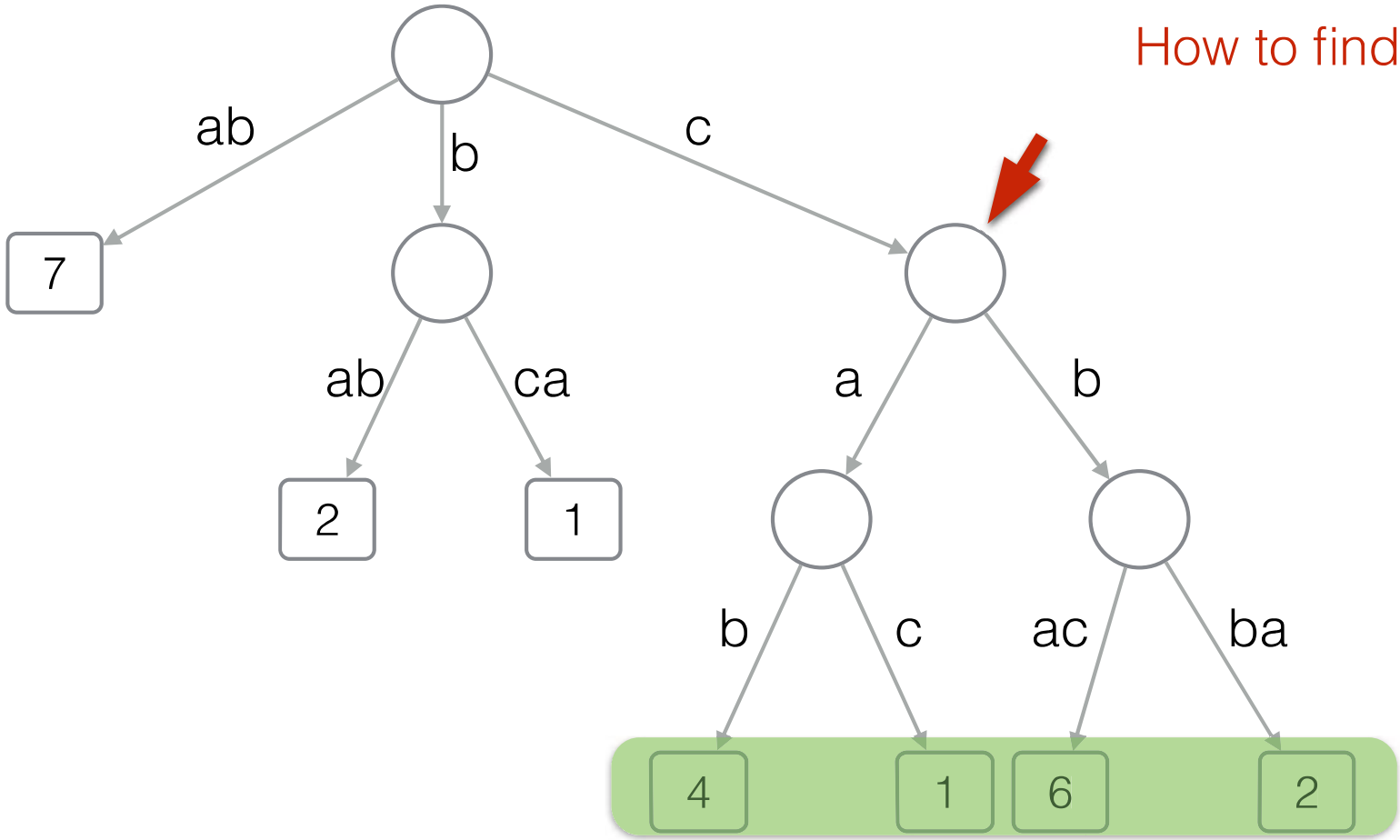
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



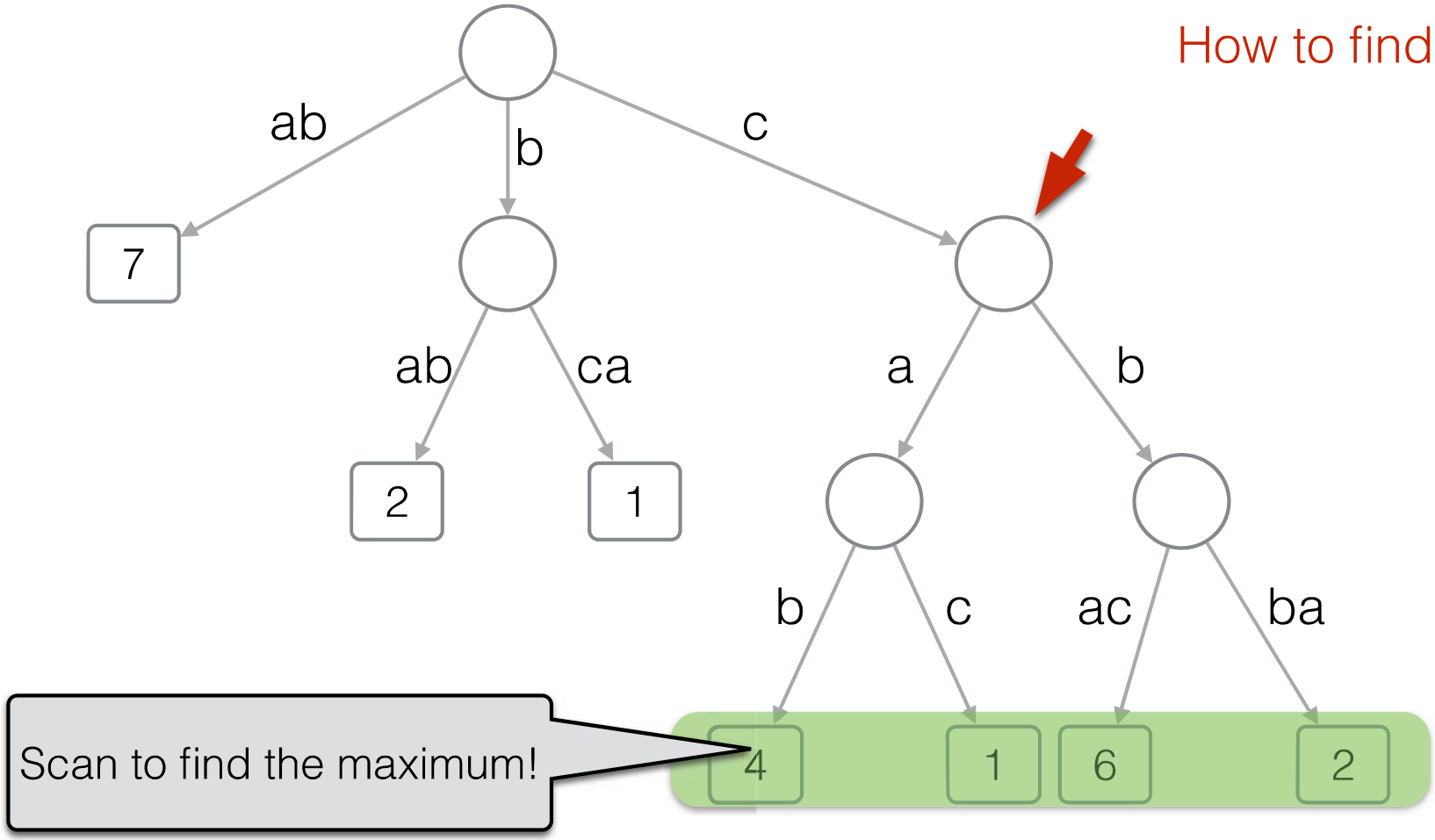
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



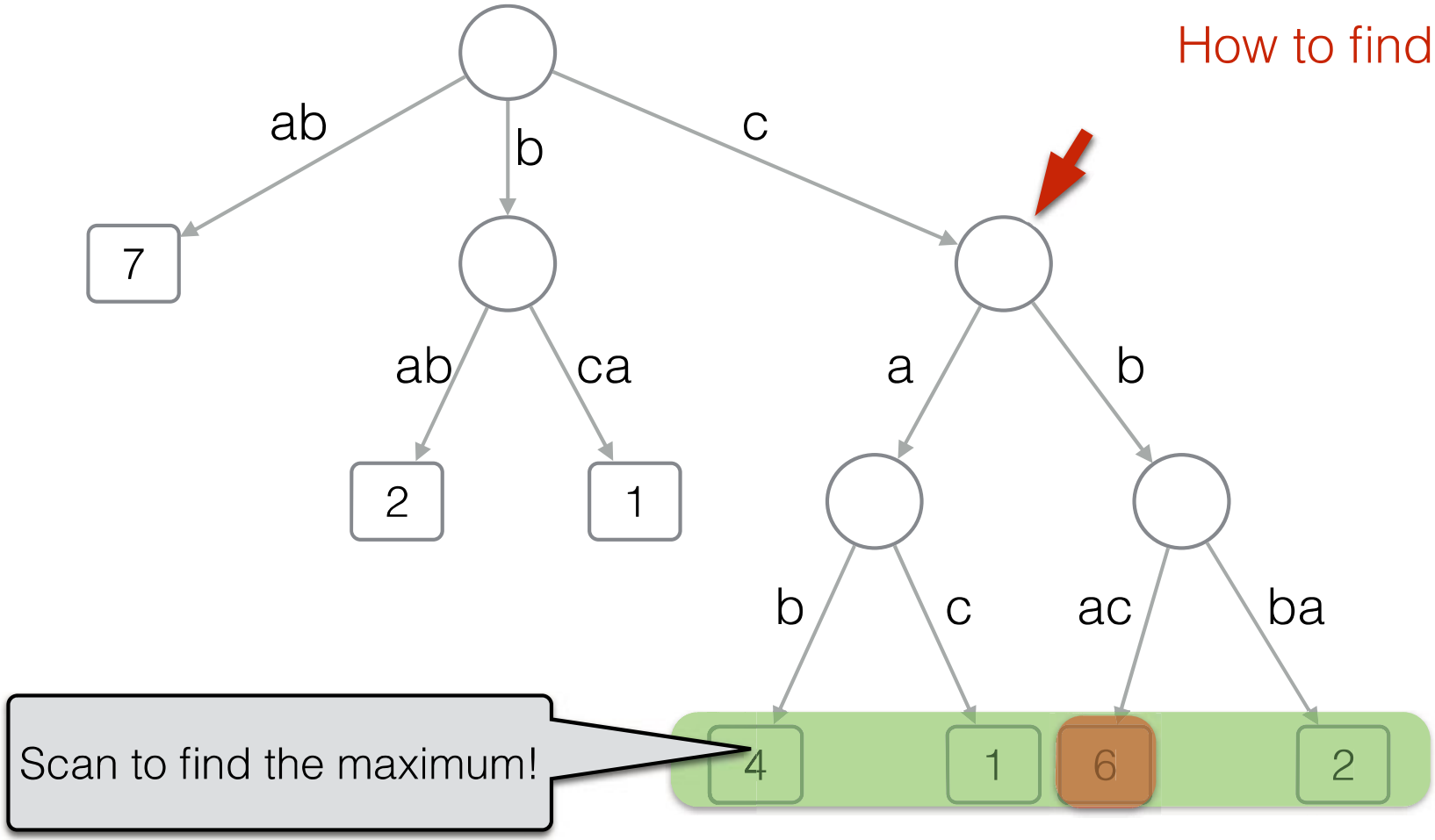
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



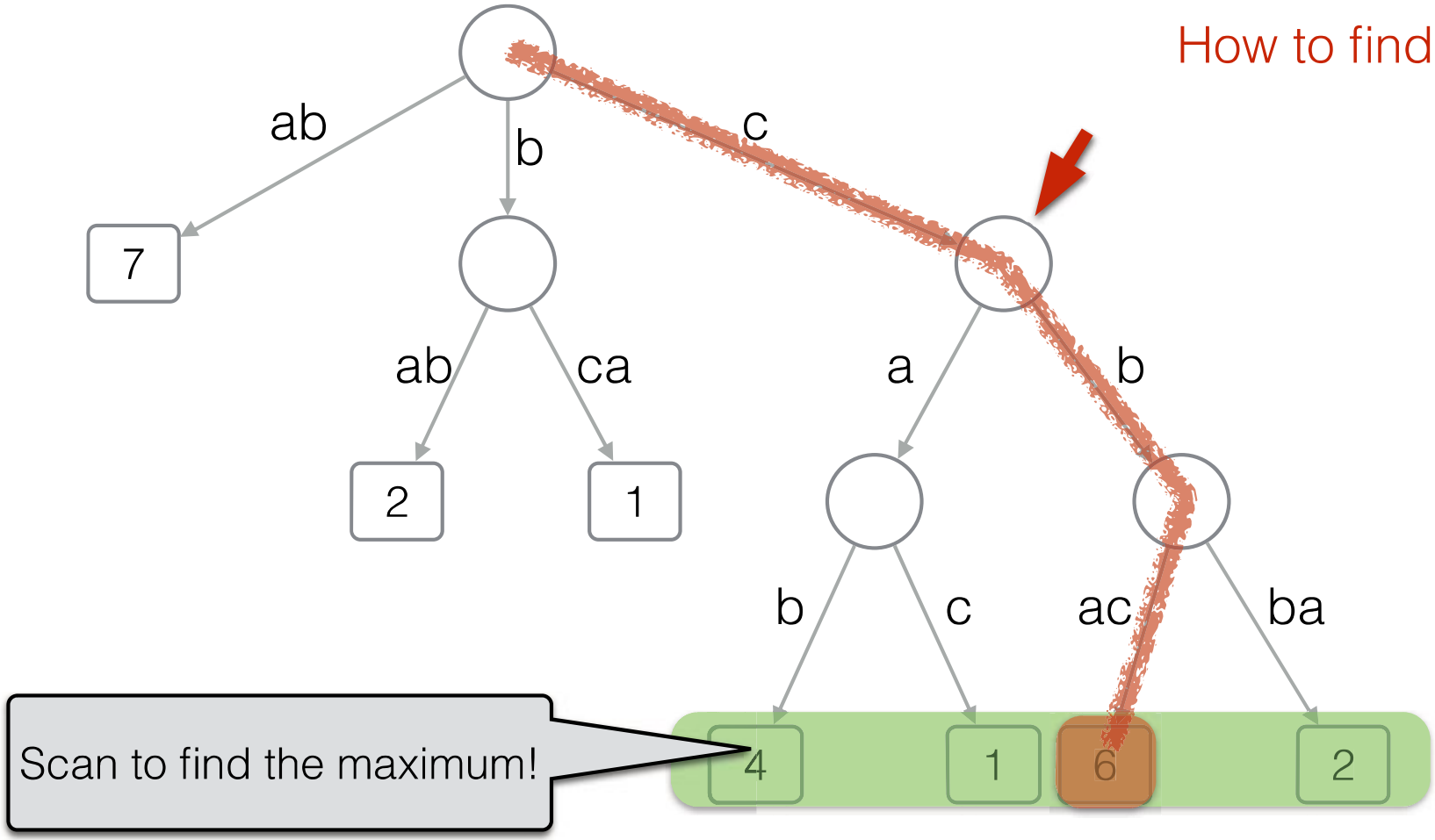
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



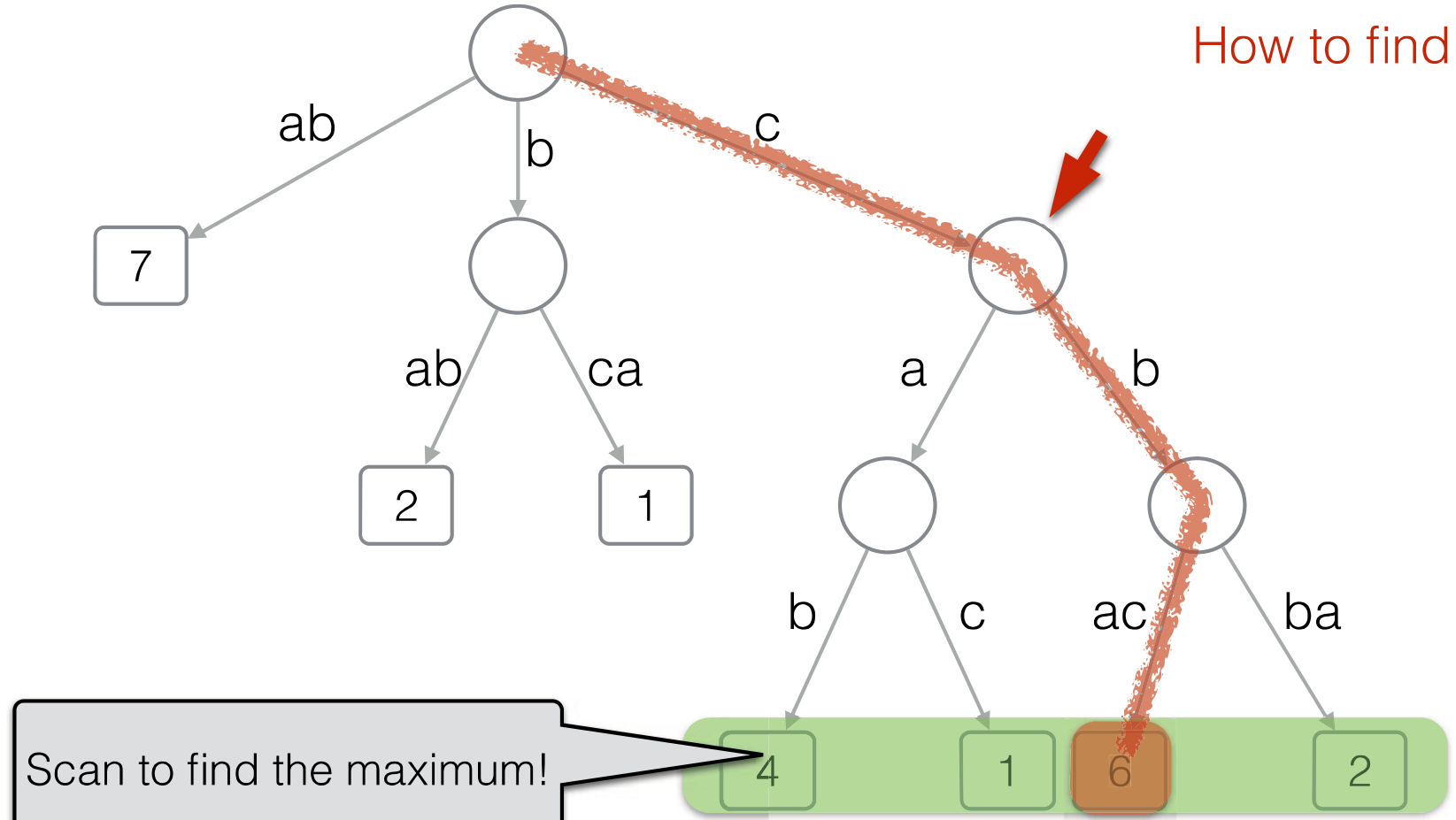
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Scan to find the maximum!

O(n) query time :-(

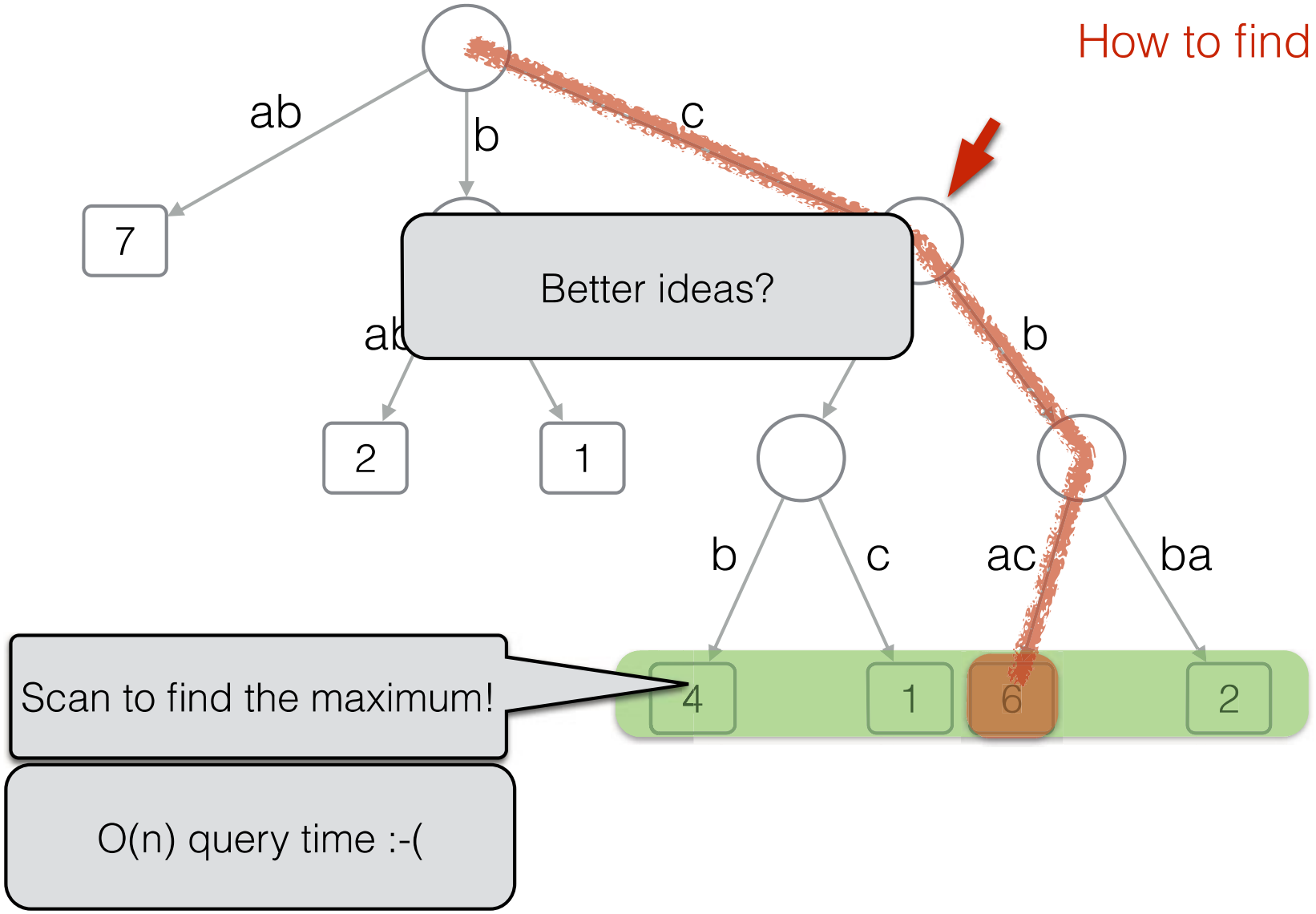
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



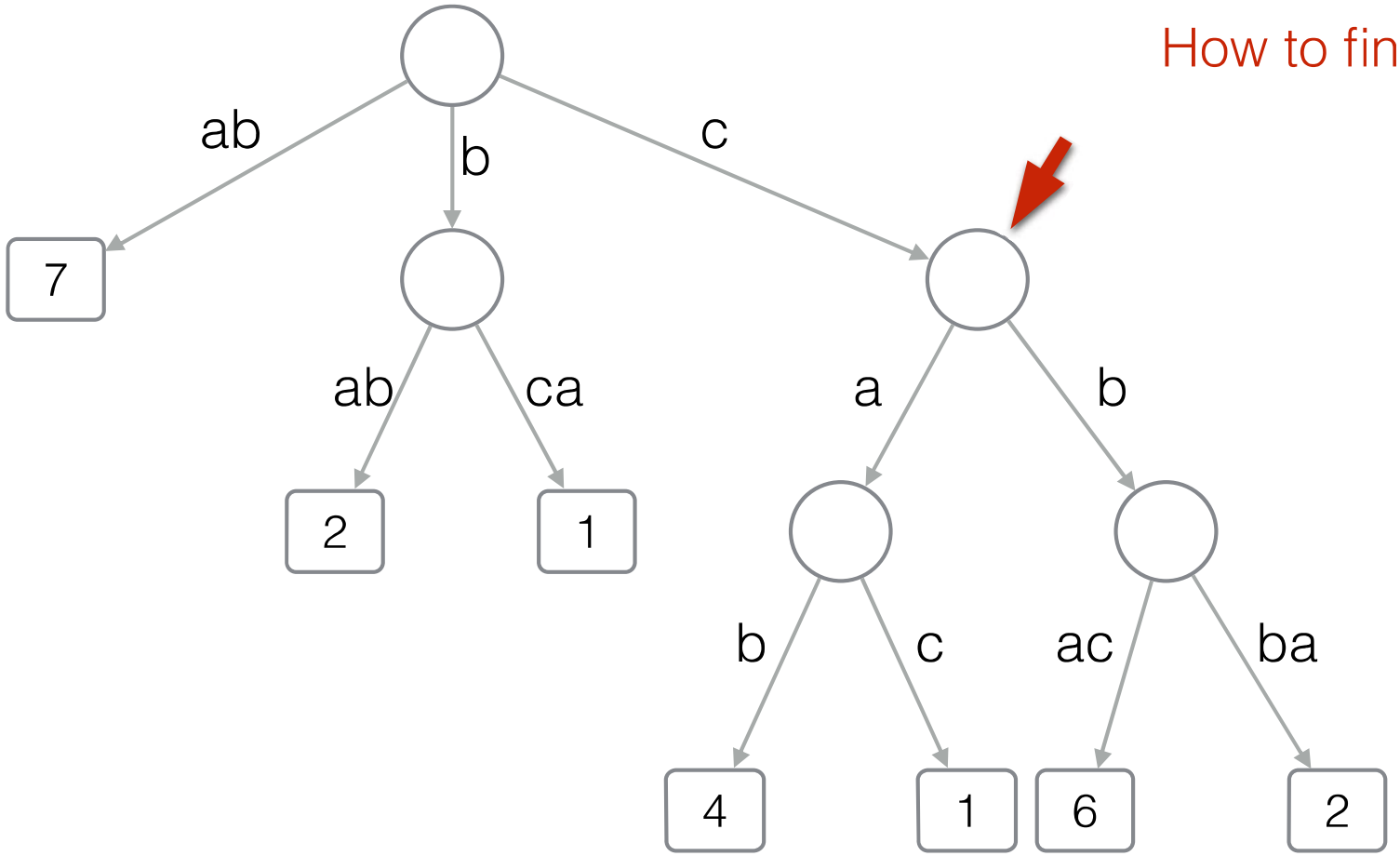
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



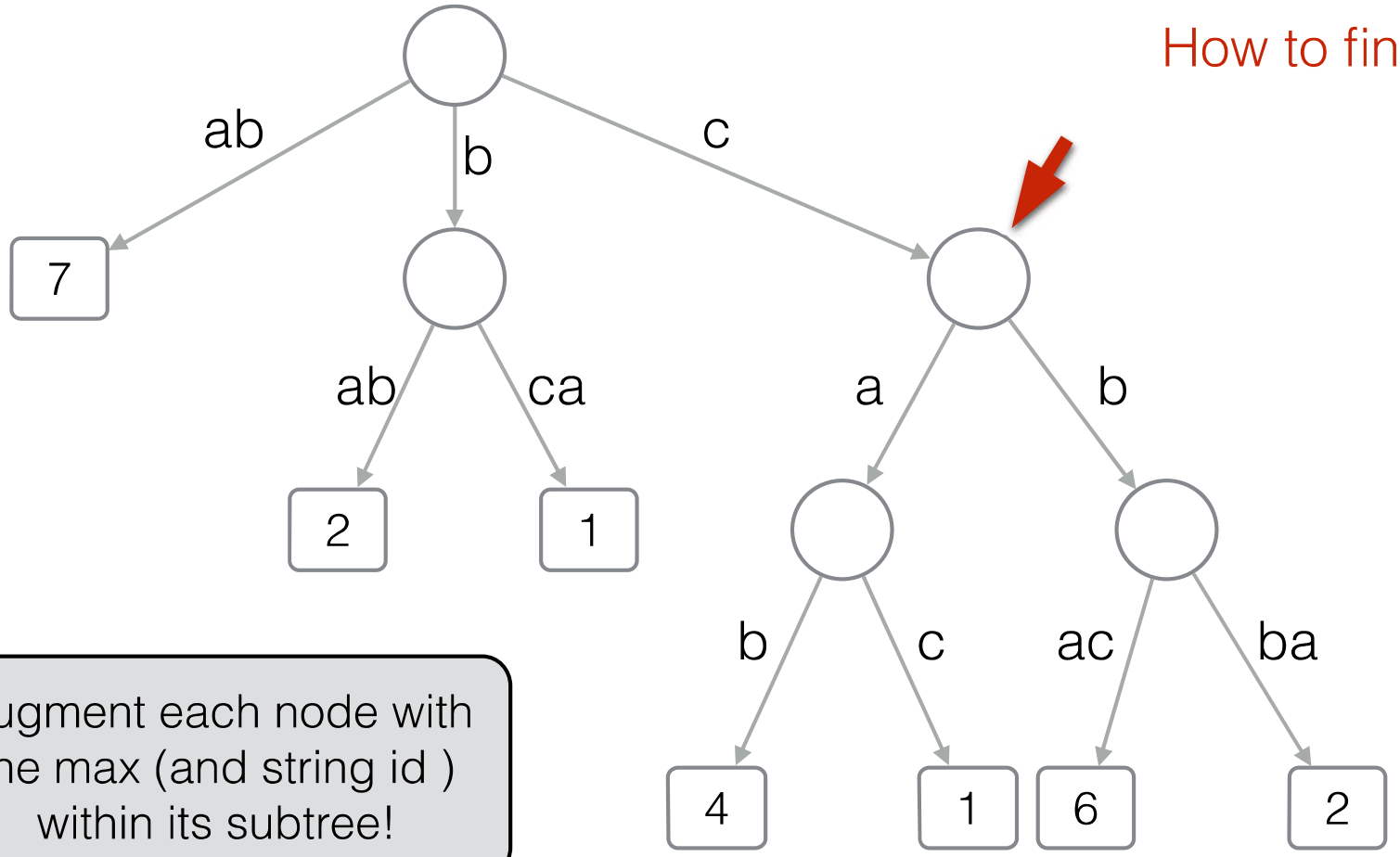
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

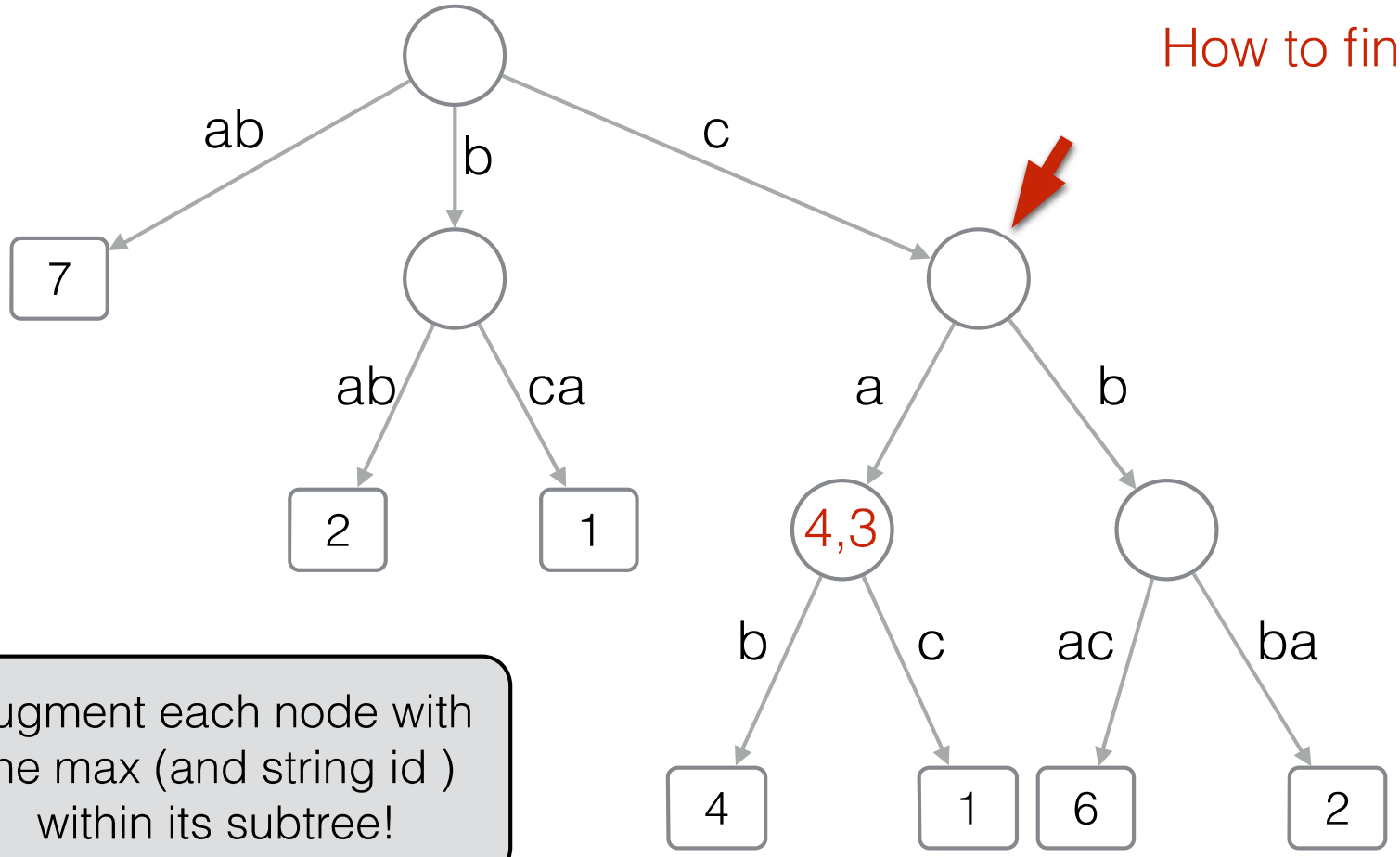
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

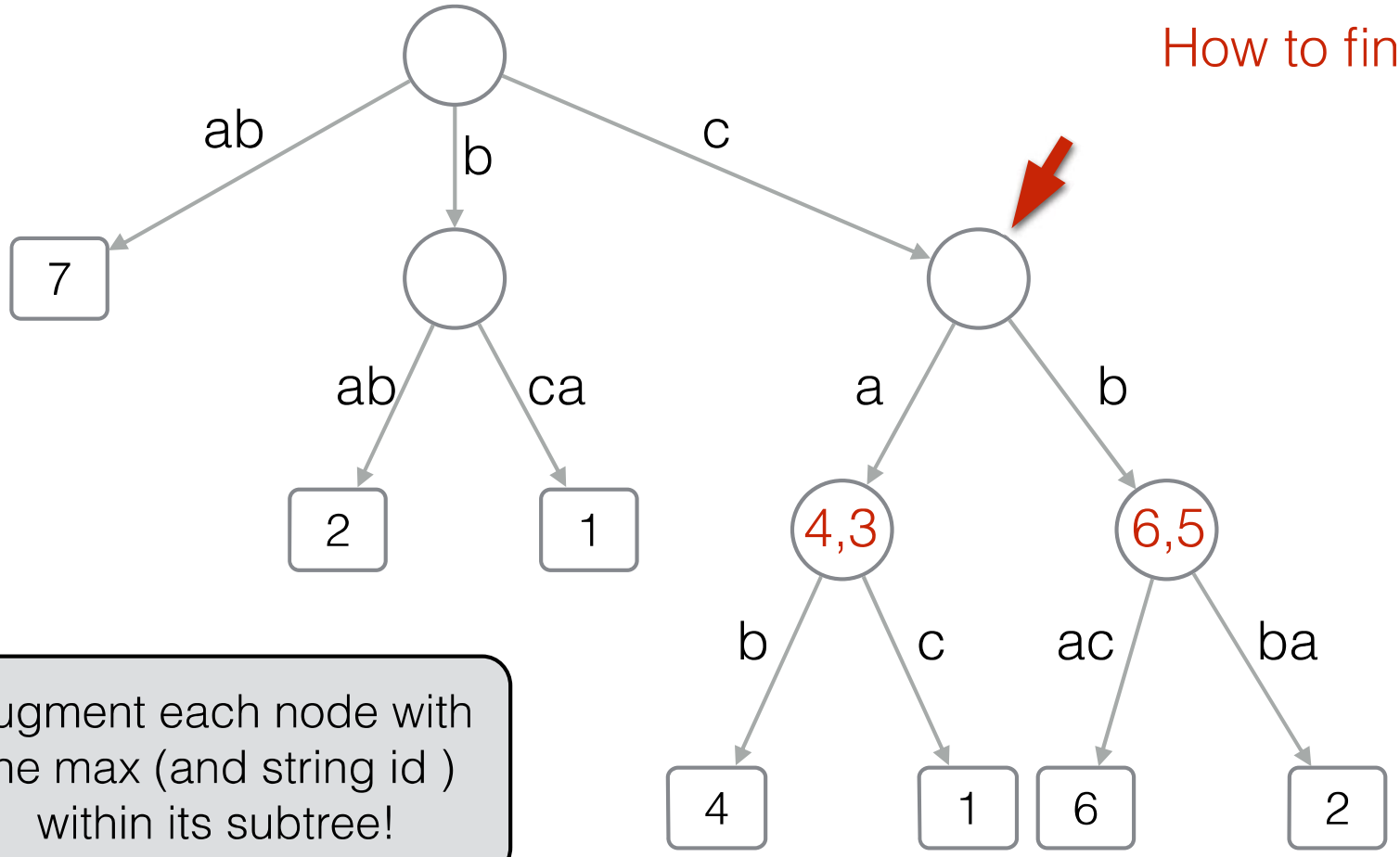
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

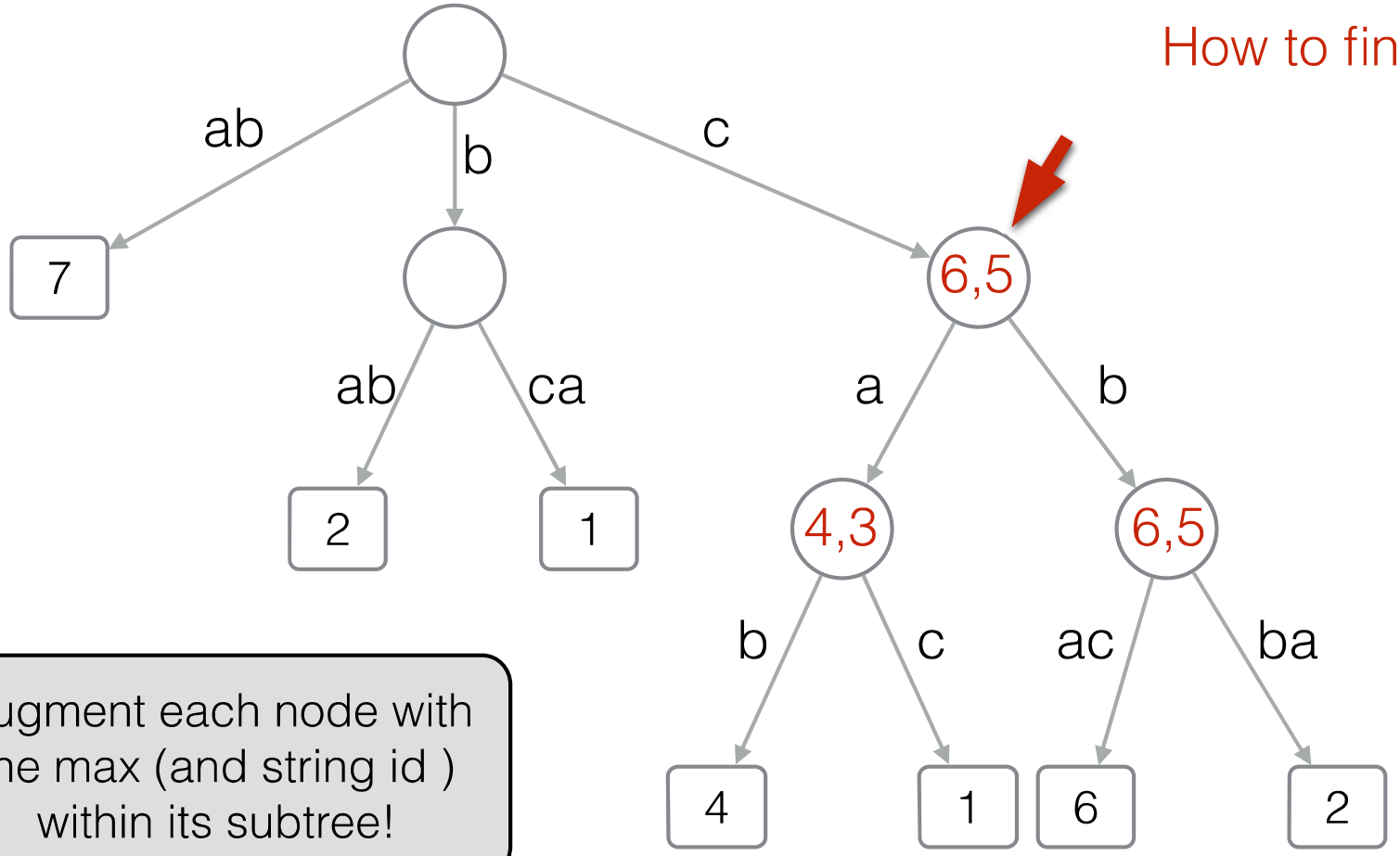
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

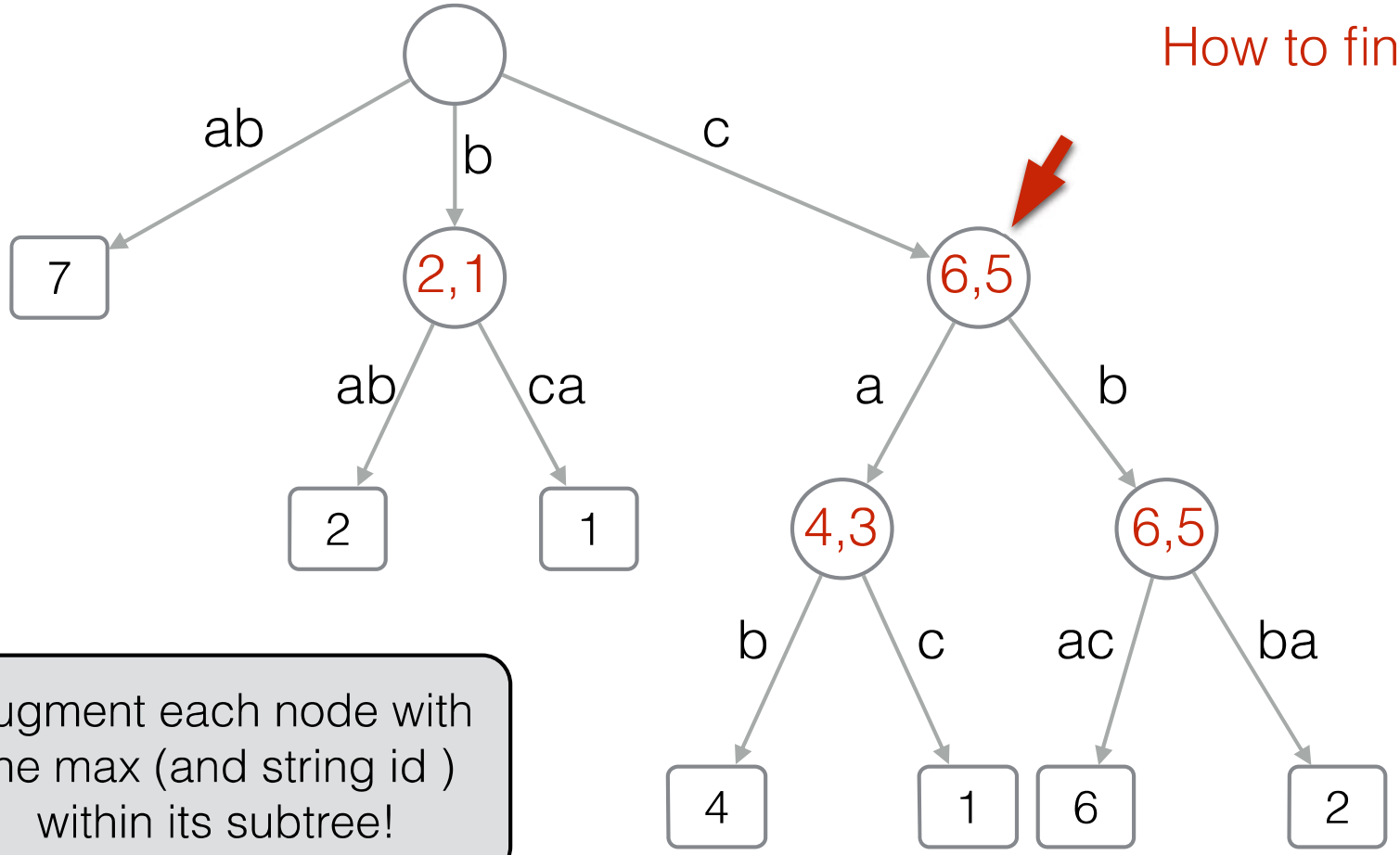
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

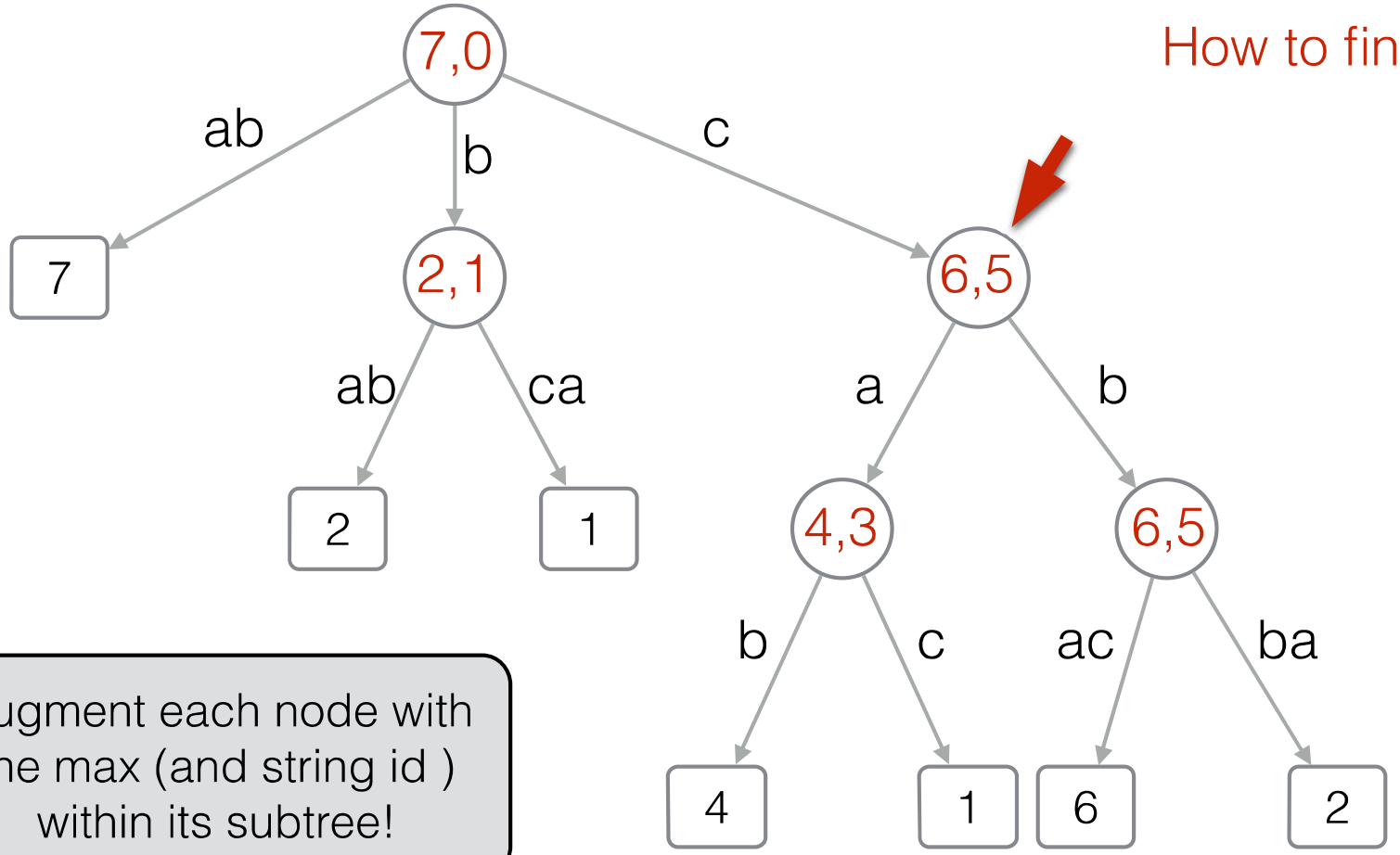
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

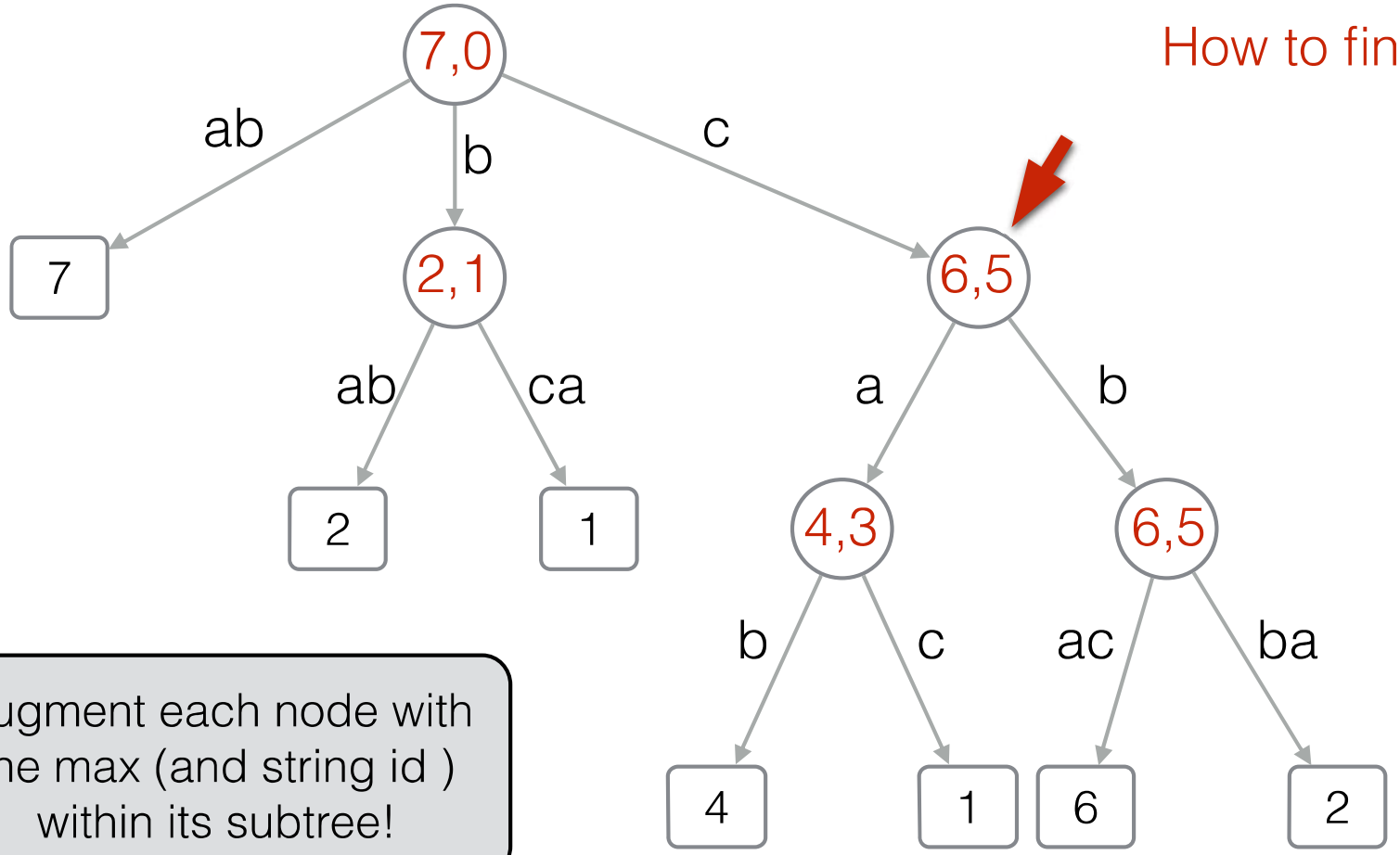
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

Preprocessing time: O(n)

Extra space: O(n log n) bits

Query time: O(1)

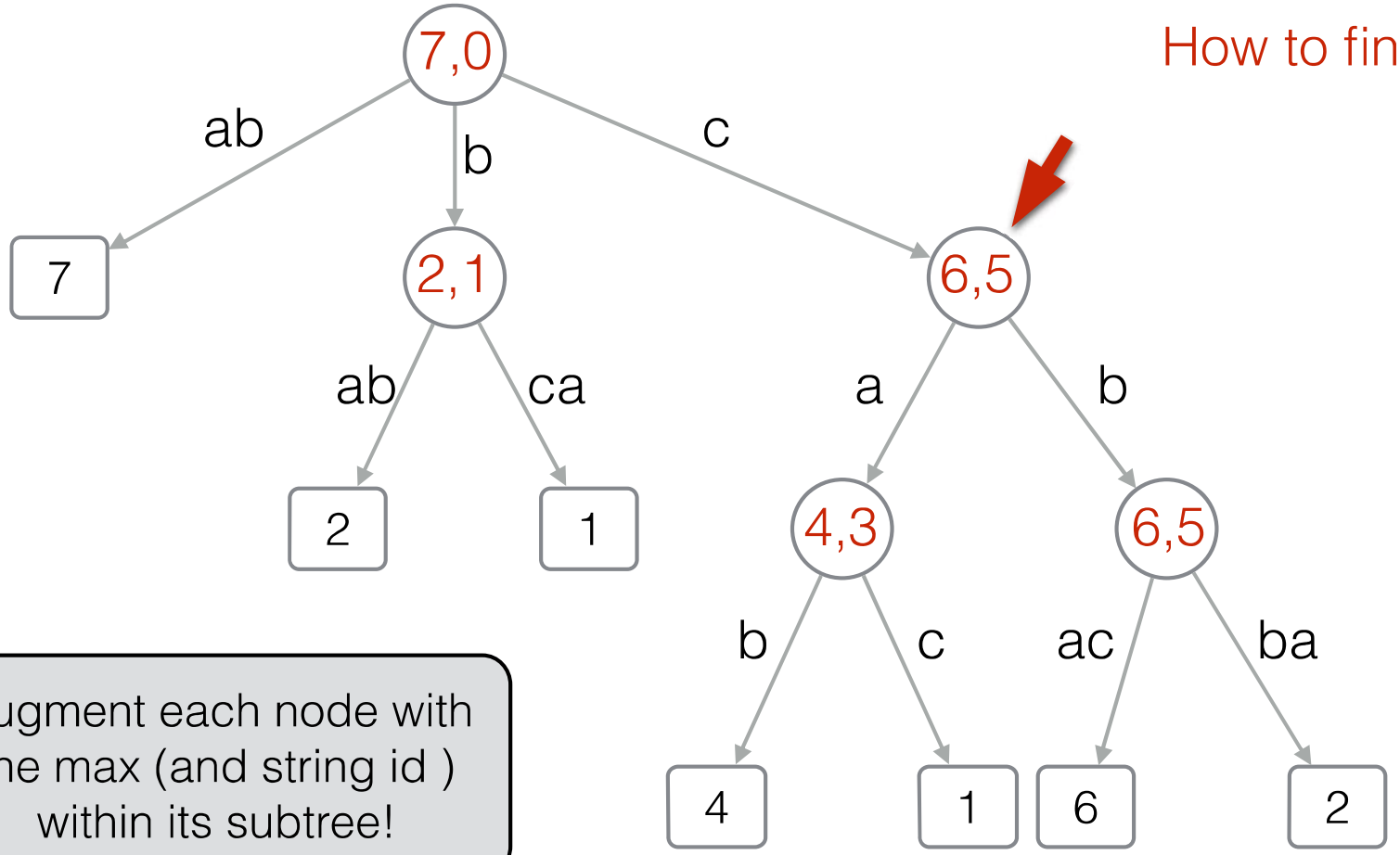
D (1), cab (4), cac (1), cbac (6), cbba (2) }

l length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

Preprocessing time: O(n)

Extra space: O(n log n) bits

Query time: O(1)

Solving Top-k?

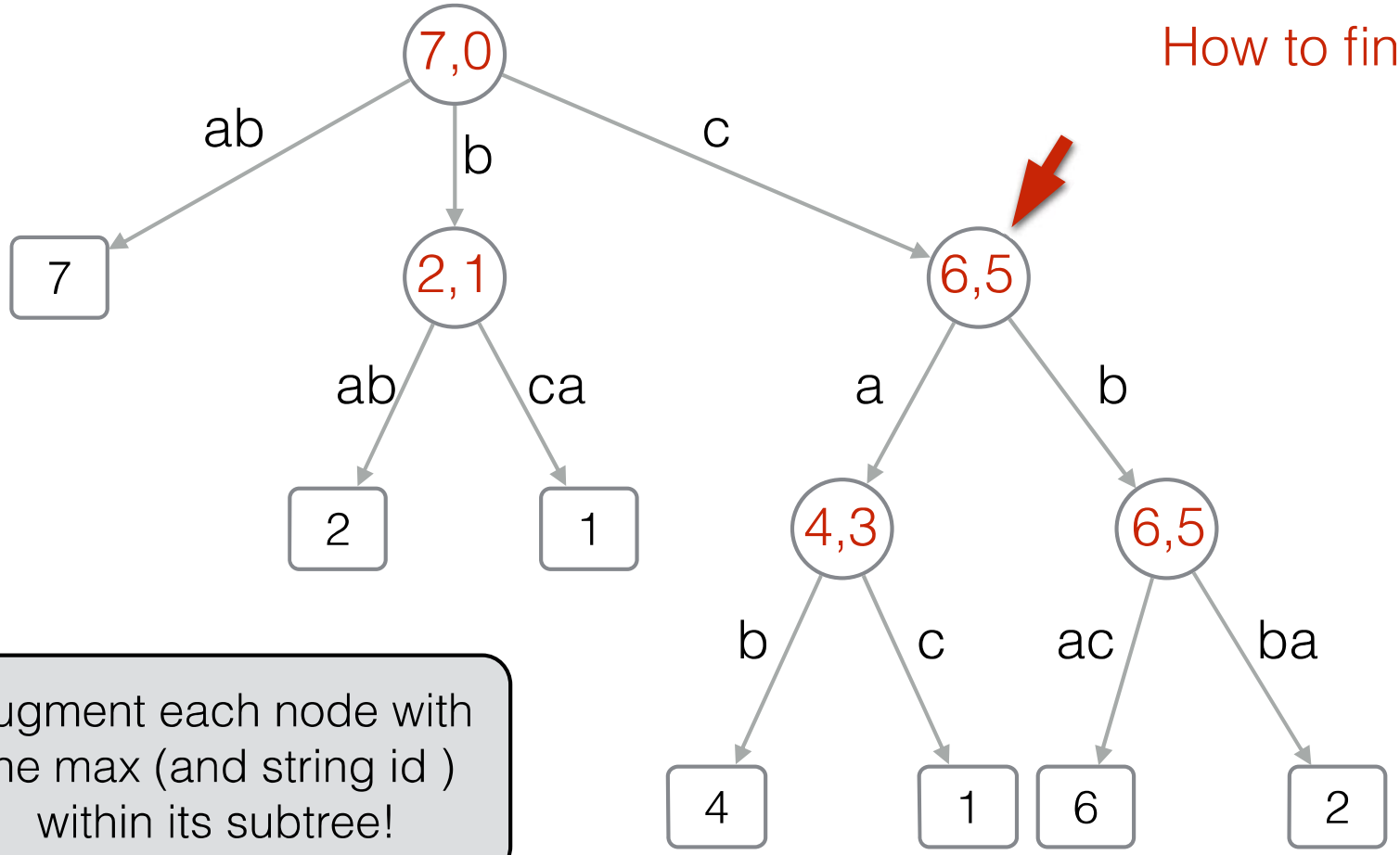
D (1), ca

l length of strings in D

Finding Top-1

P = c

How to find Top-1?



Augment each node with the max (and string id) within its subtree!

Preprocessing time: O(n)

Extra space: O(n log n) bits

Query time: O(1)

Solving Top-k?

- Extra space: O(k*n*log n) bits :-(
- You must know k at building time! :-(

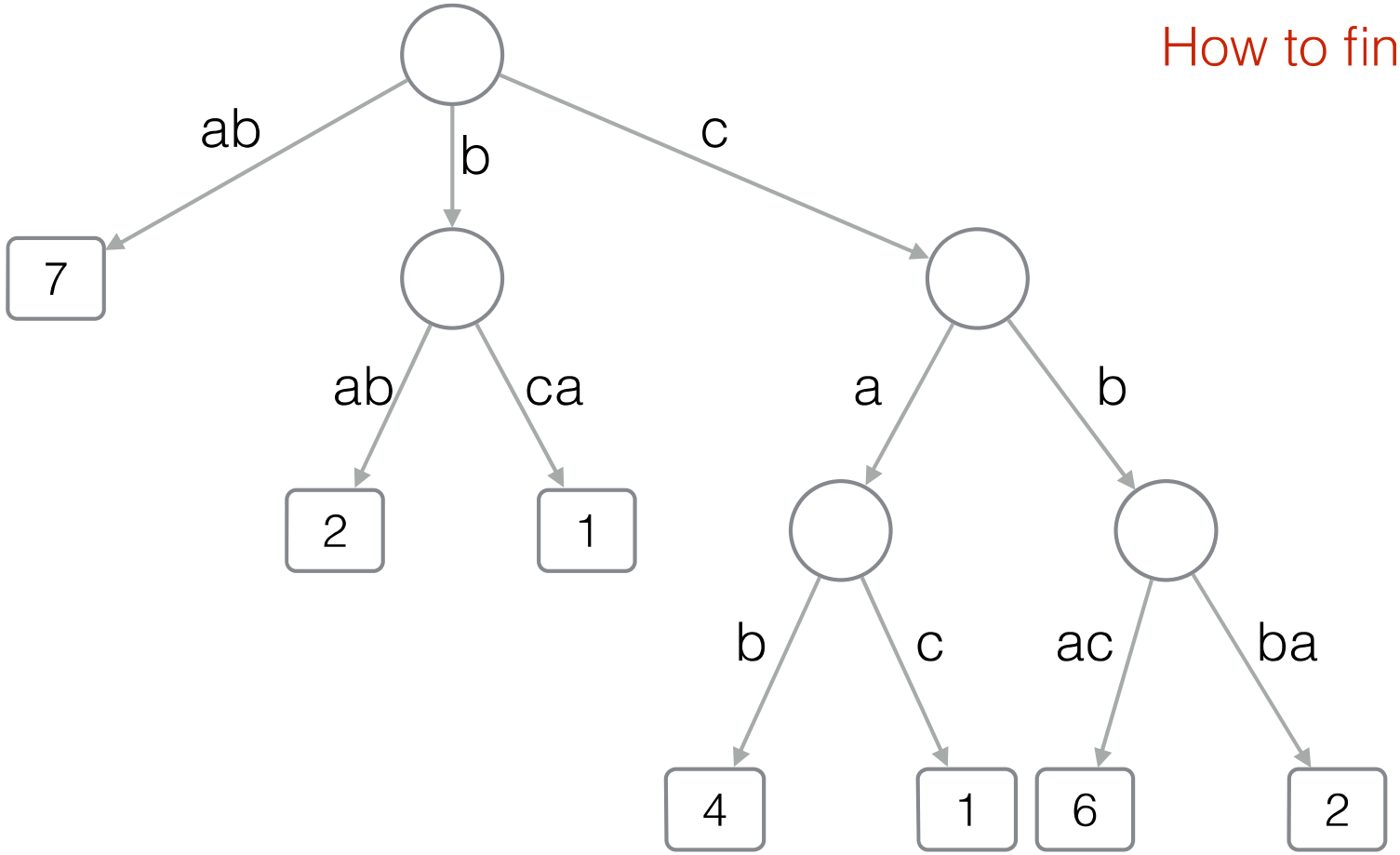
D (1), ca

l length of strings in D

Finding Top-1

P = c

How to find Top-1?



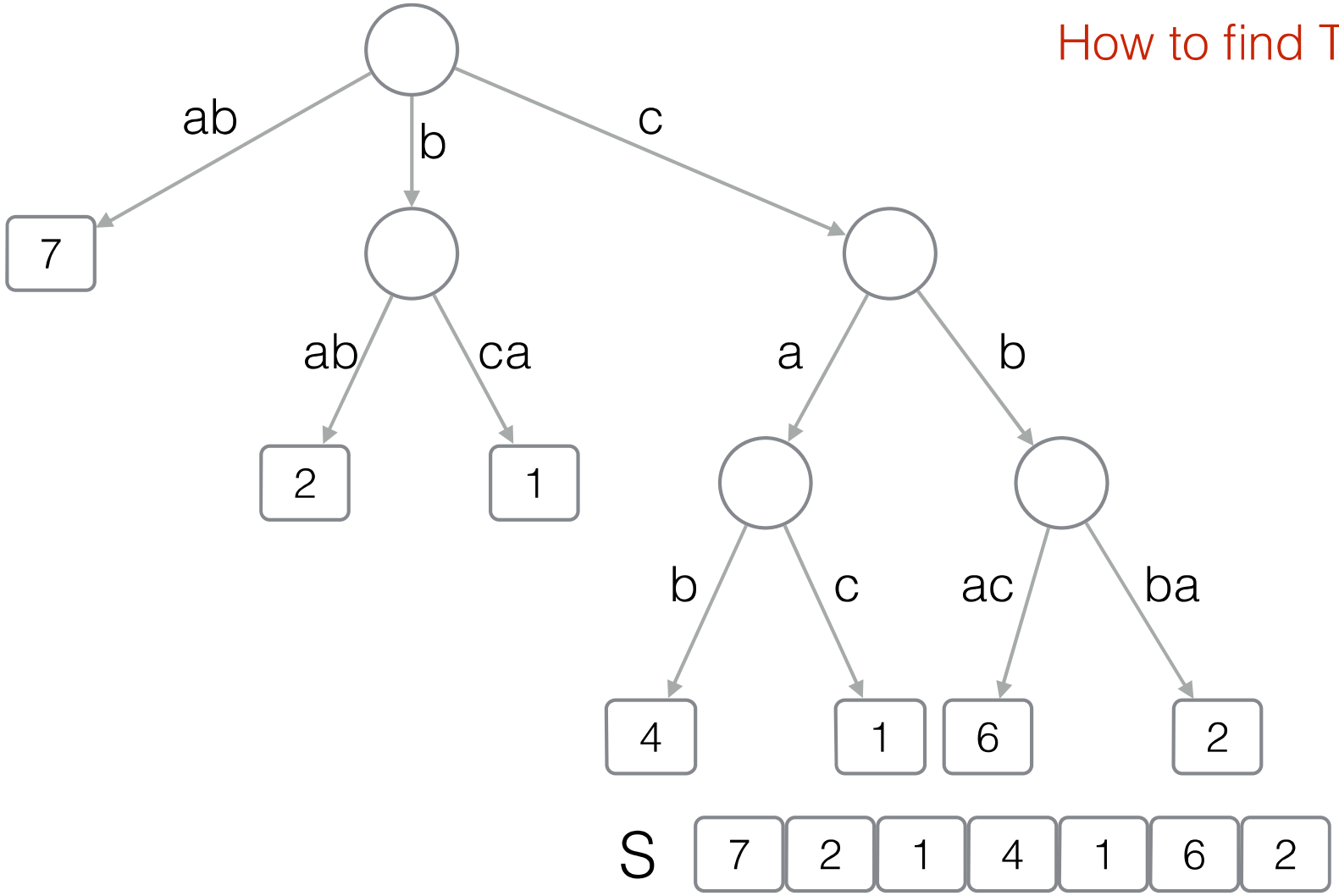
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



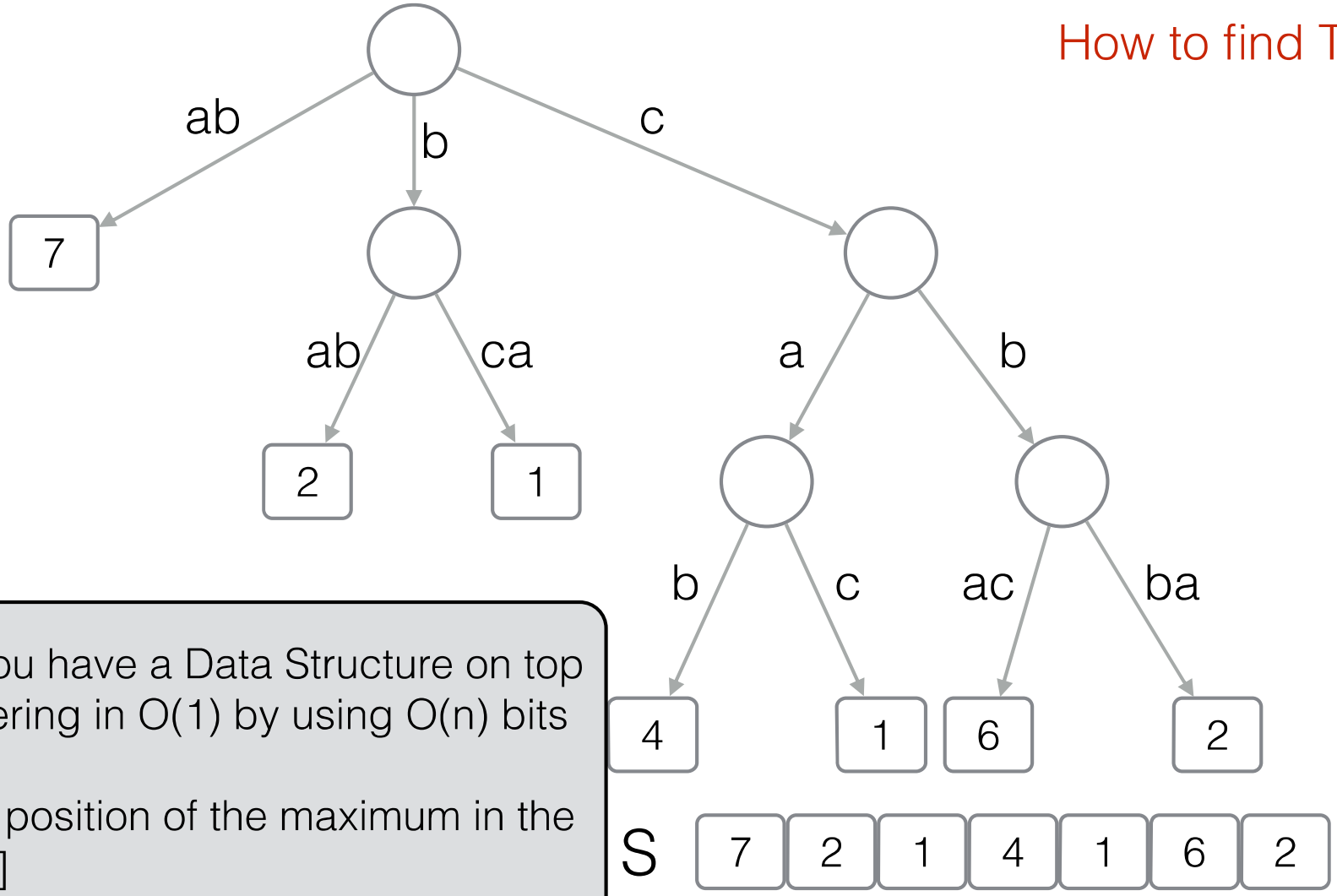
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of the maximum in the range S[i,j]

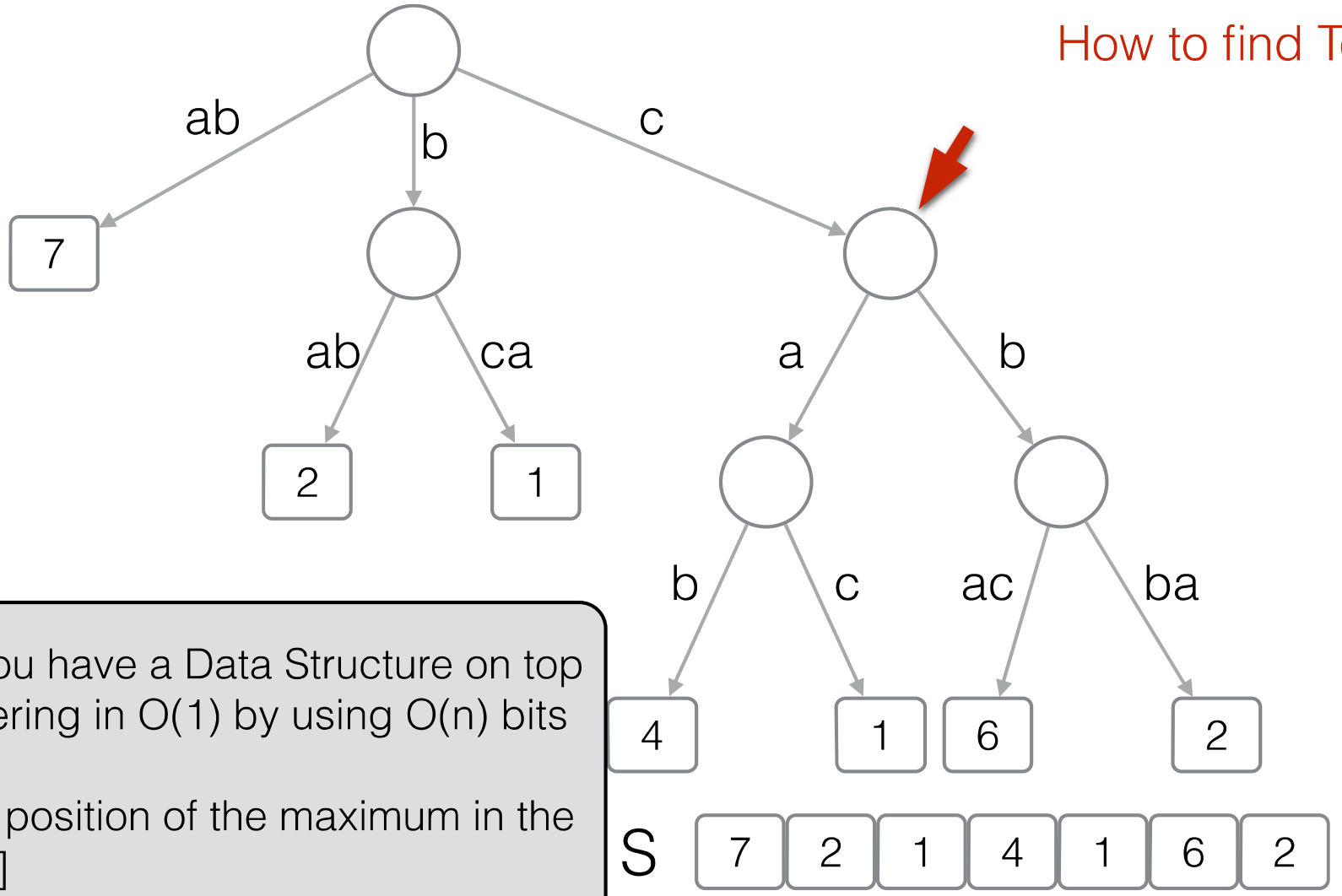
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of the maximum in the range S[i,j]

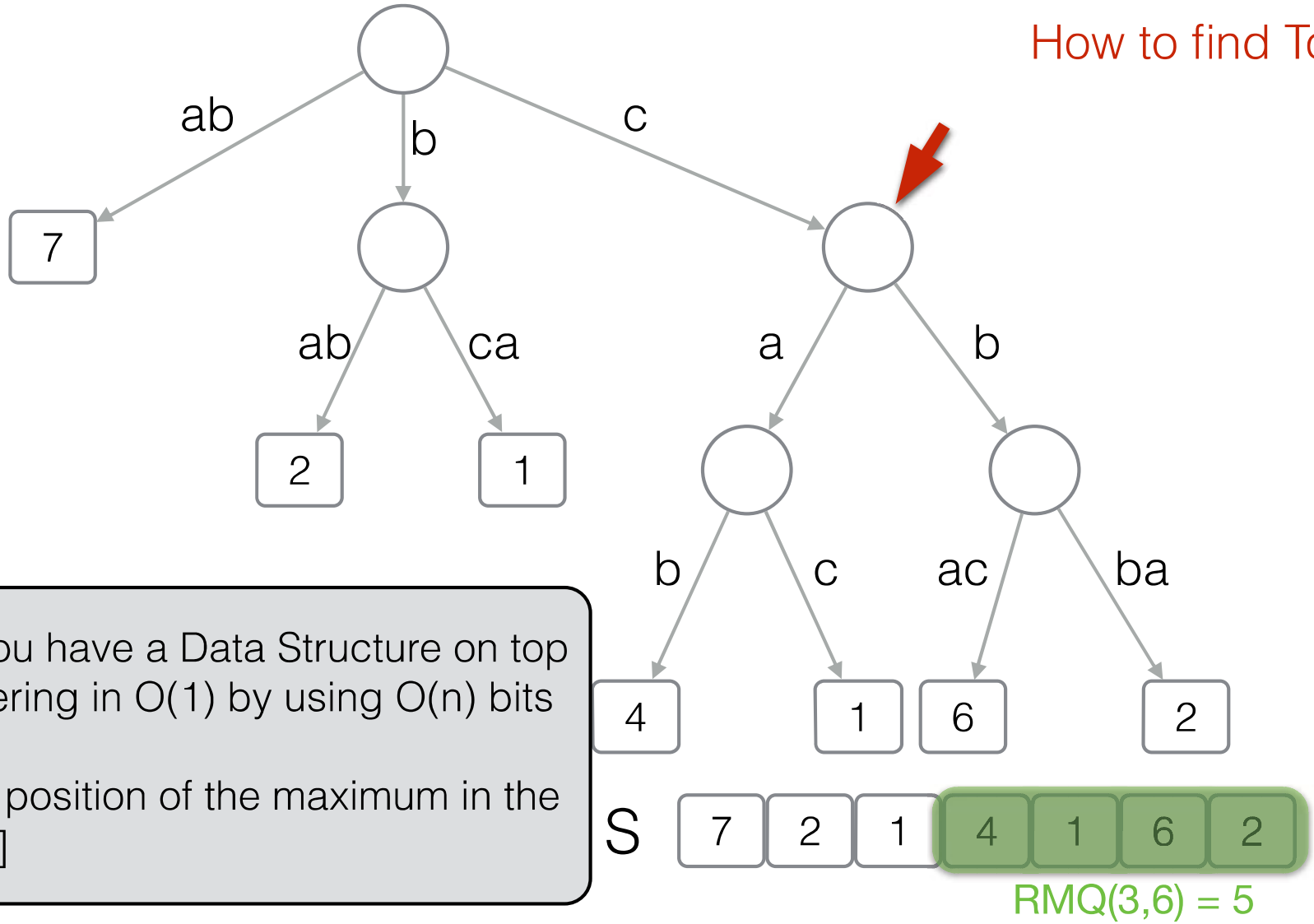
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of the maximum in the range S[i,j]

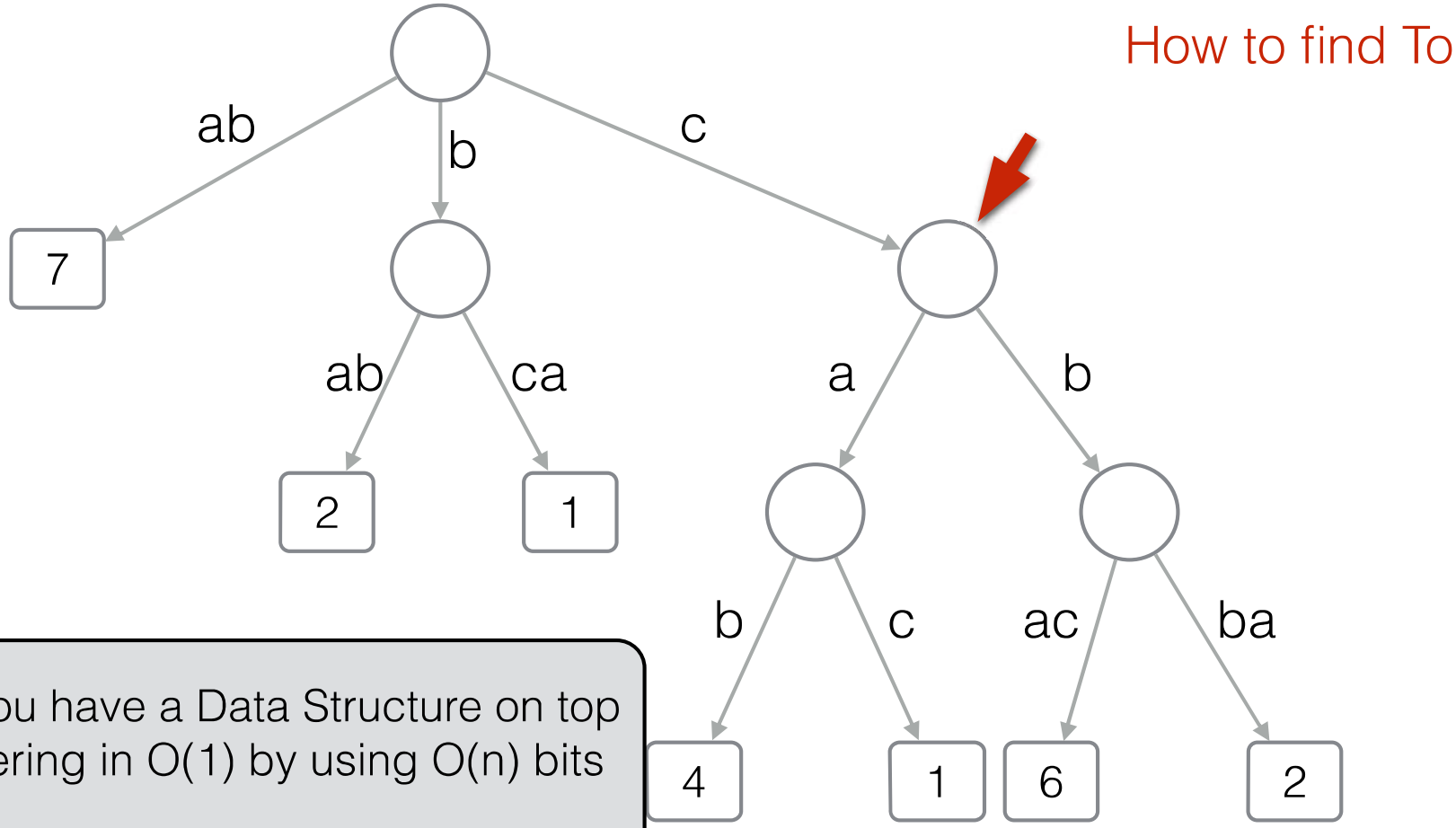
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of th
range S[i,j]

Can you solve Top-2?

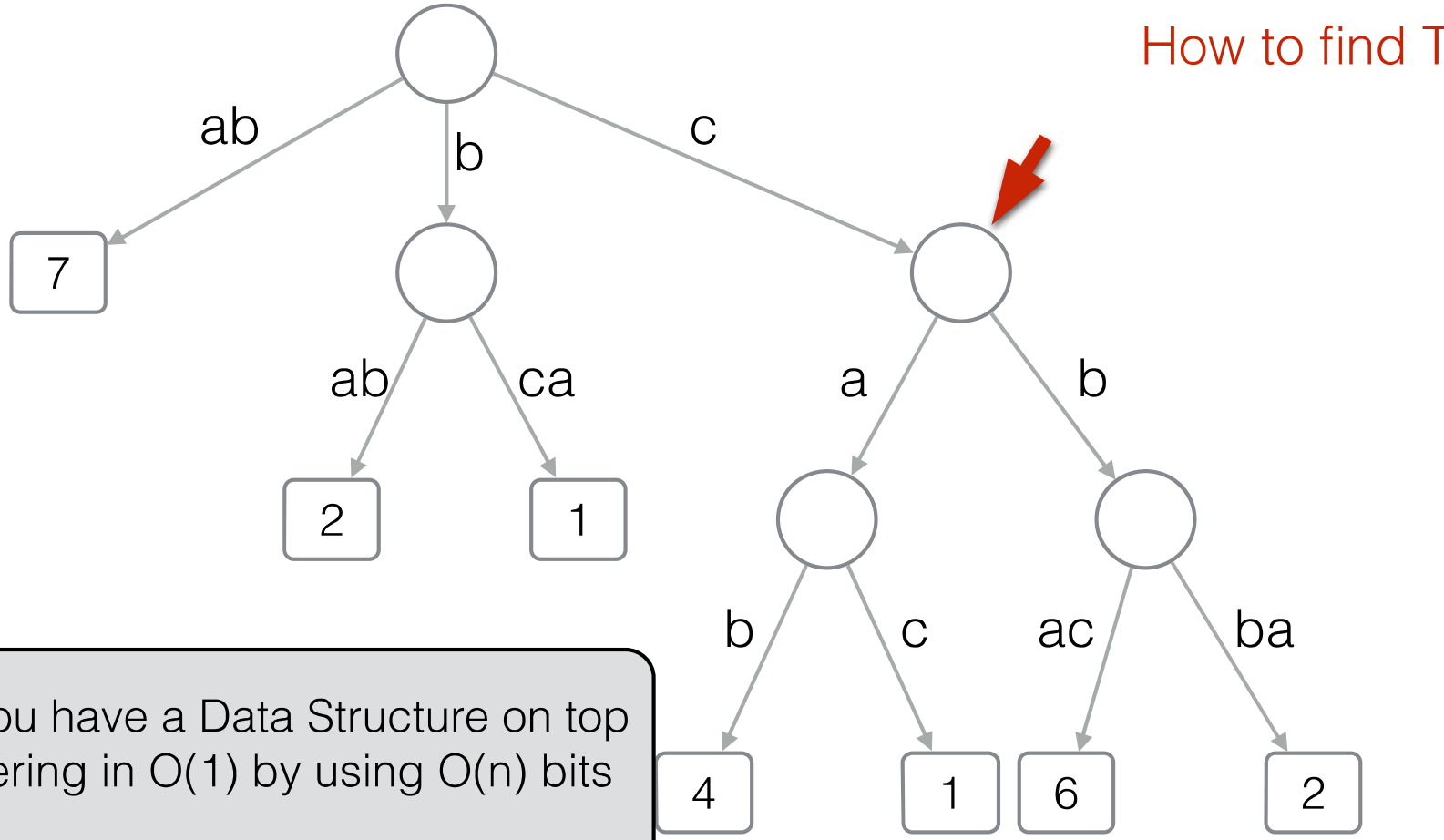
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of th
range S[i,j]

Can you solve Top-2?

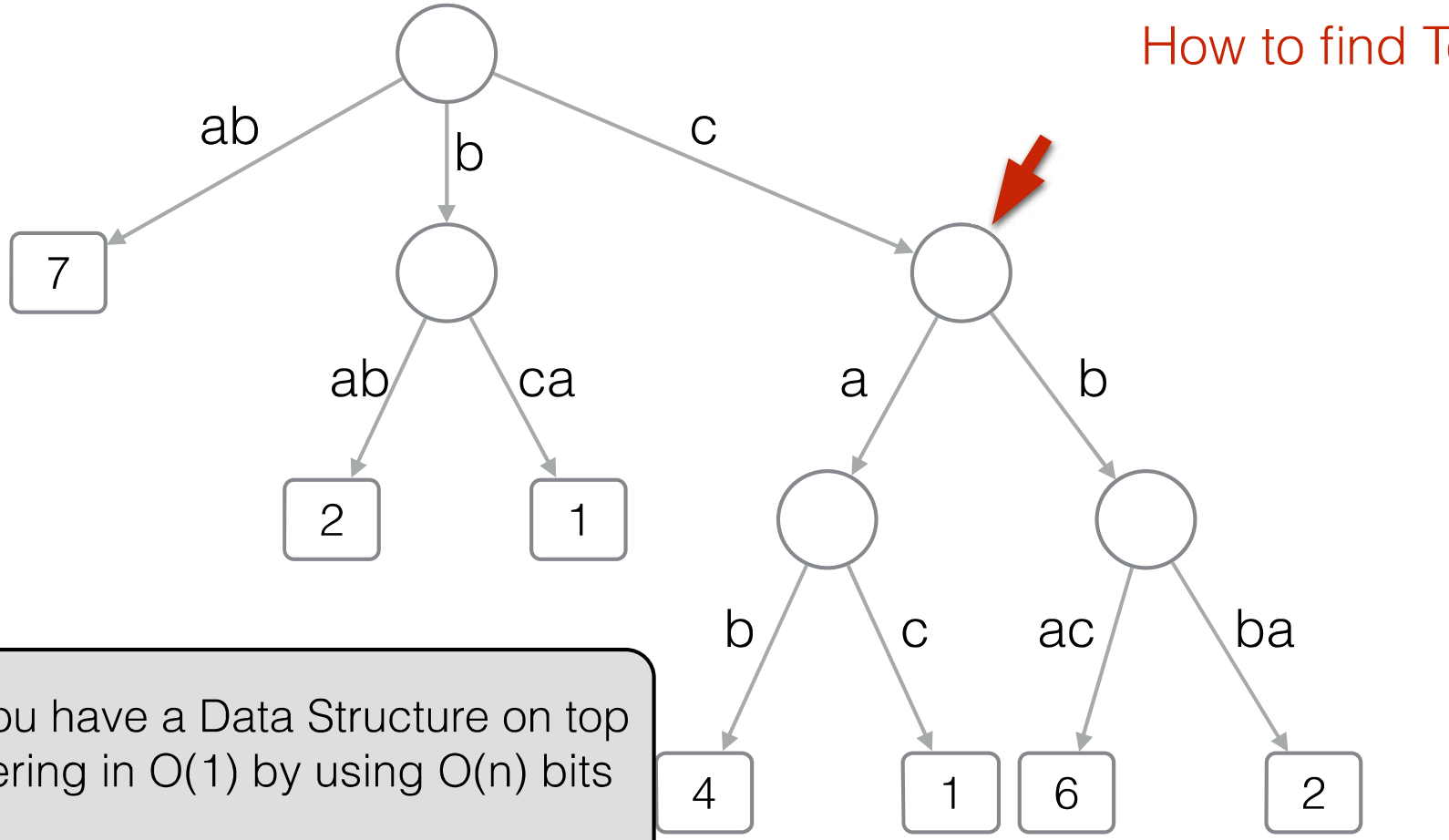
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Finding Top-1

P = c

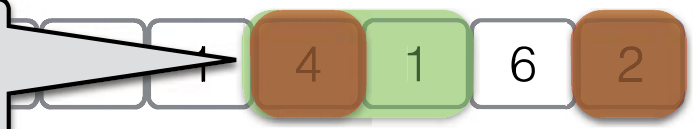
How to find Top-1?



Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of th
range S[i,j]

Can you solve Top-2?



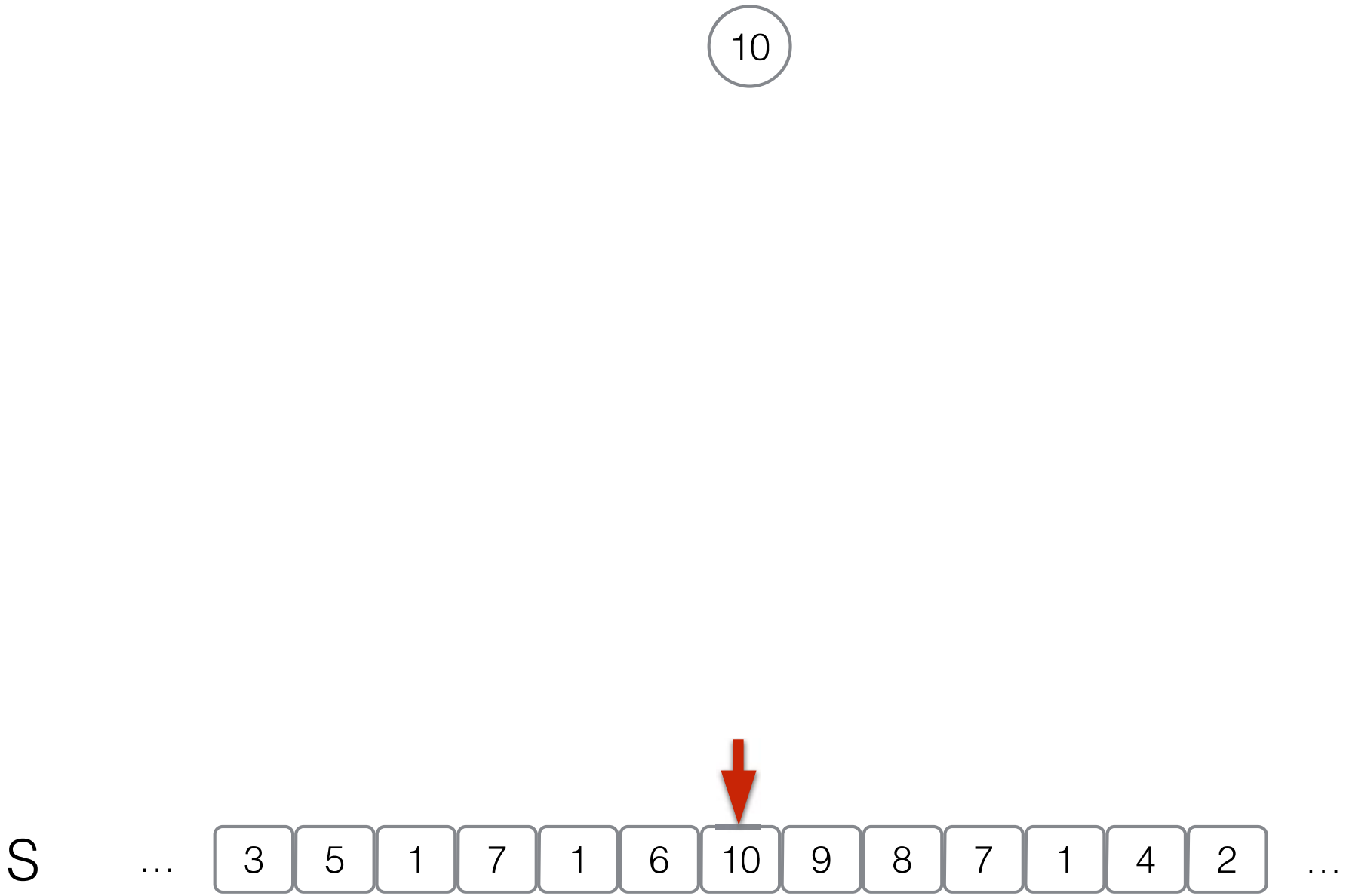
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

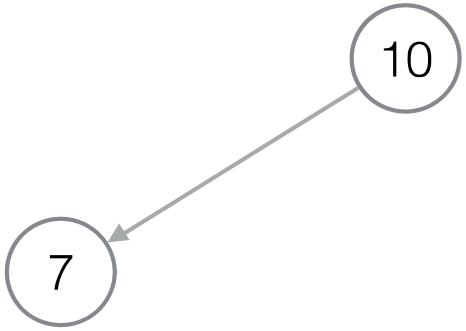
Finding Top-k

Finding Top-k

Finding Top-k

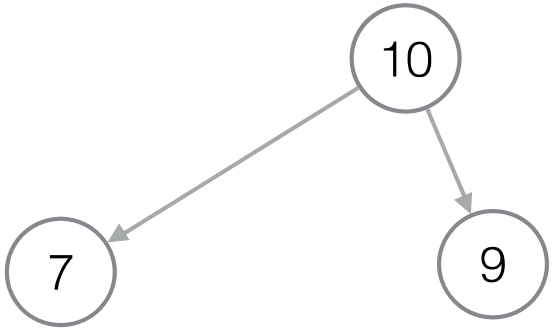


Finding Top-k



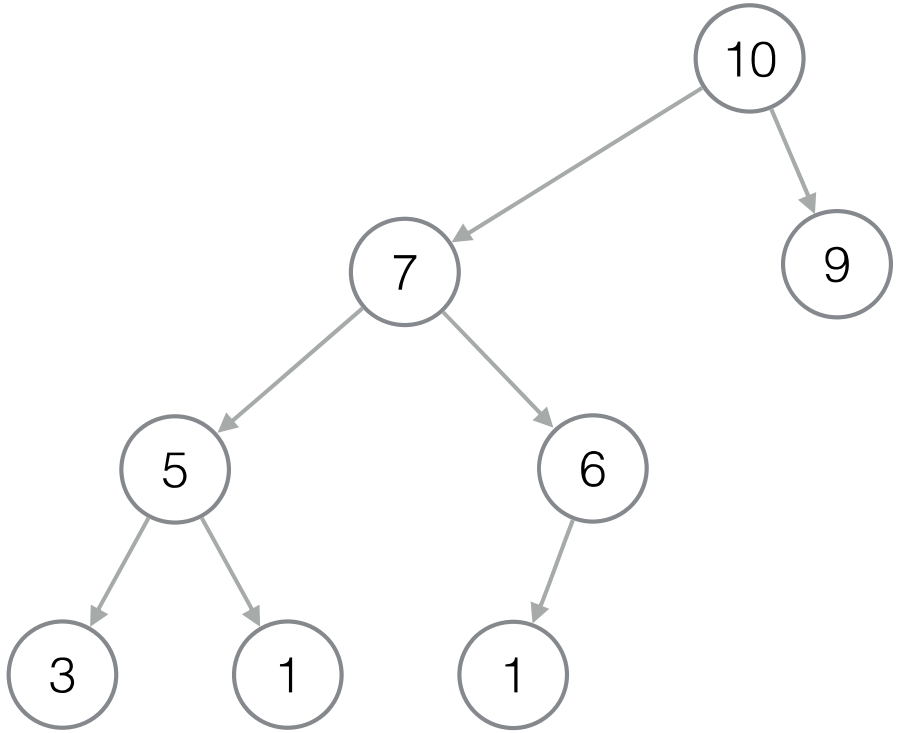
S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k



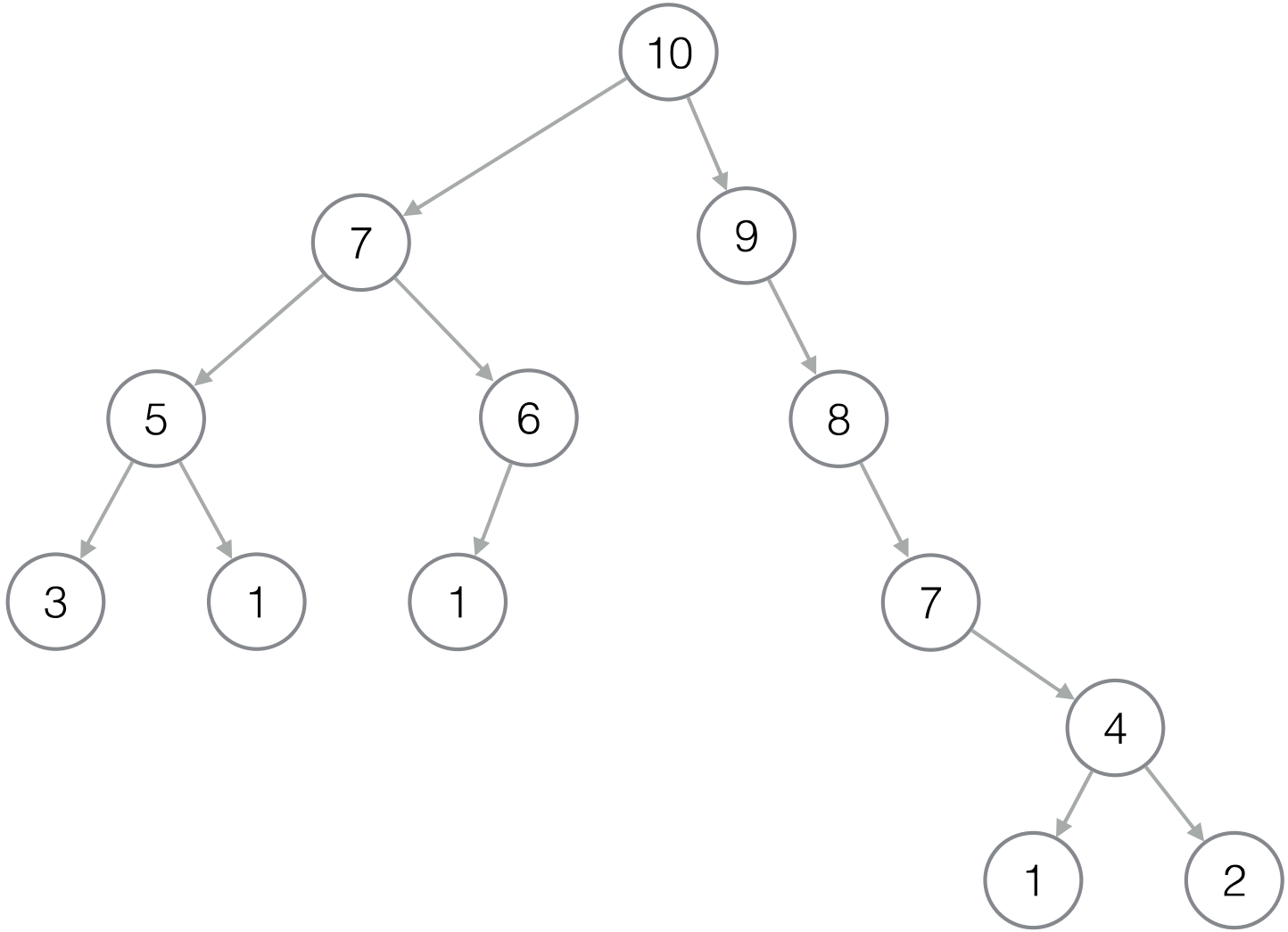
S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

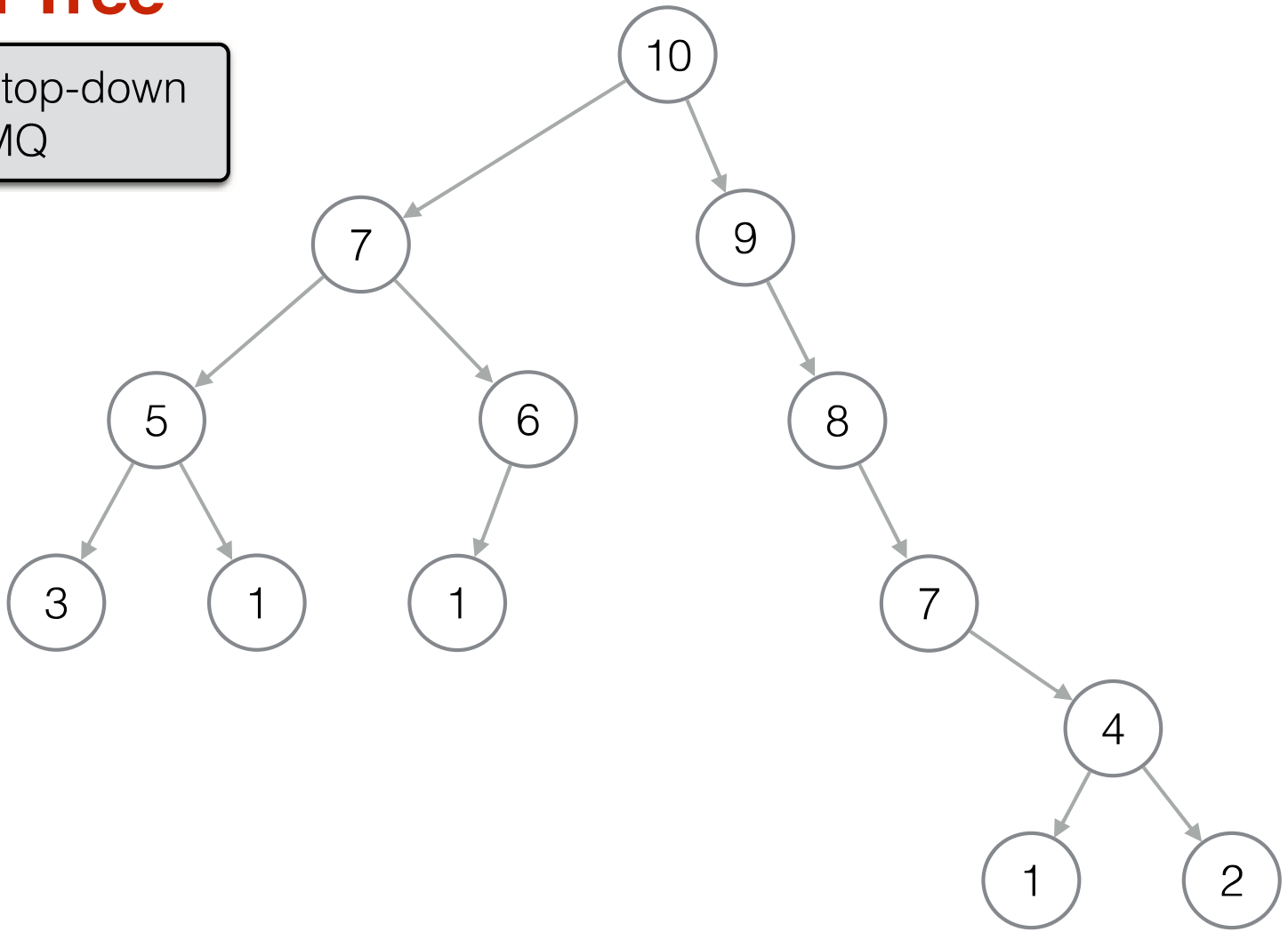


S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree

It can be built top-down with RMQ

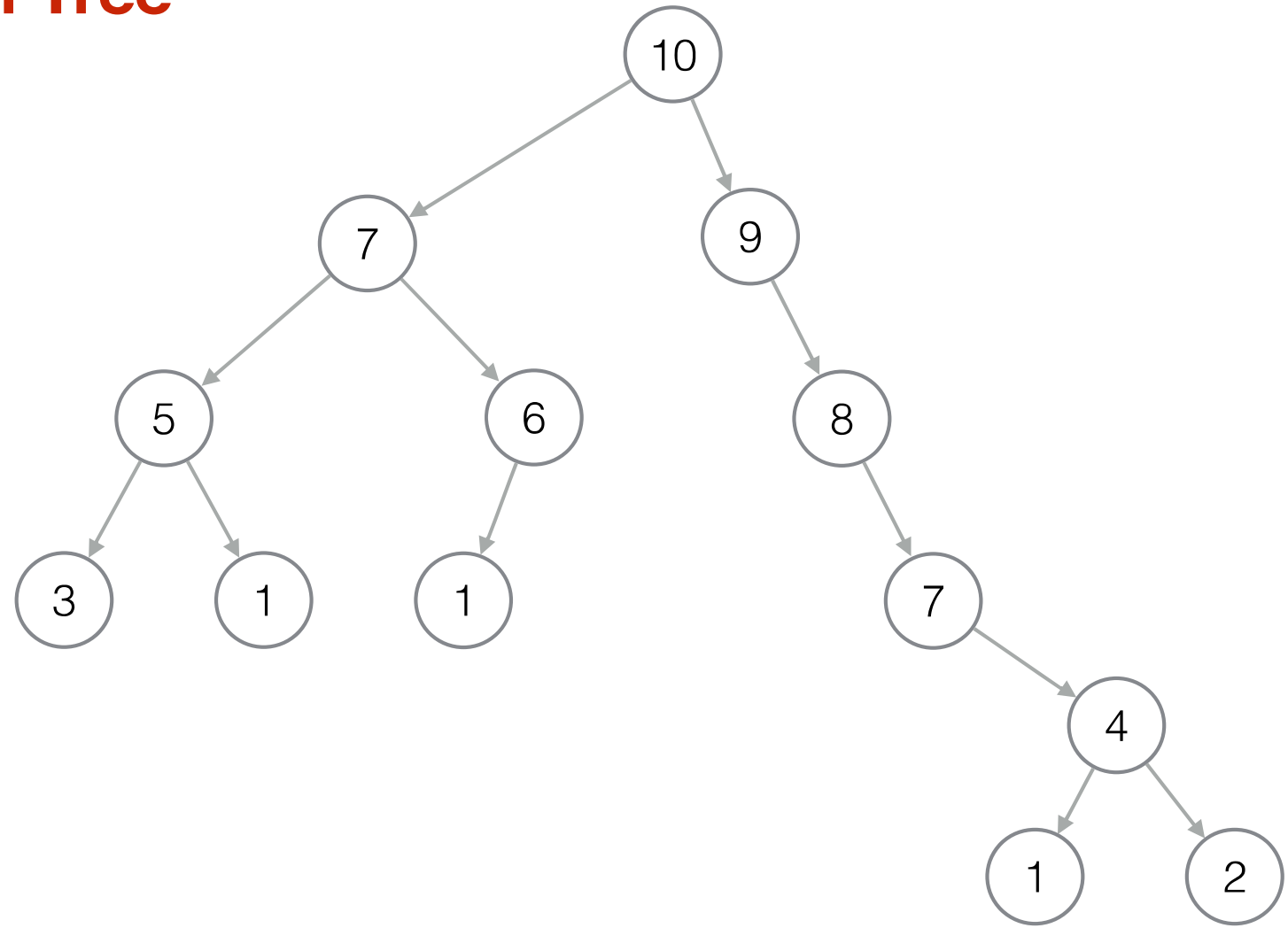


S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

How to find Top-k?

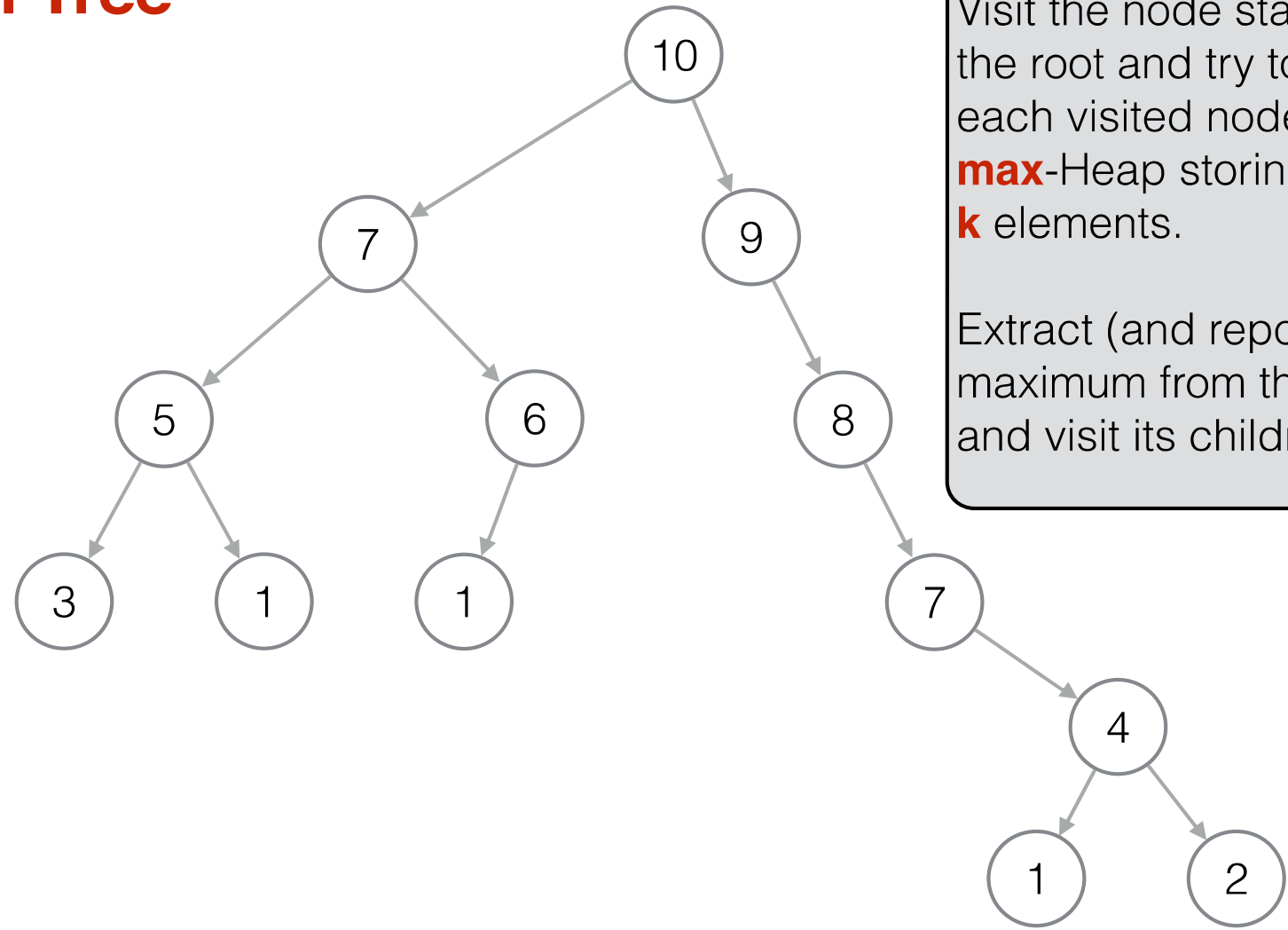
Cartesian Tree



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



How to find Top-k?

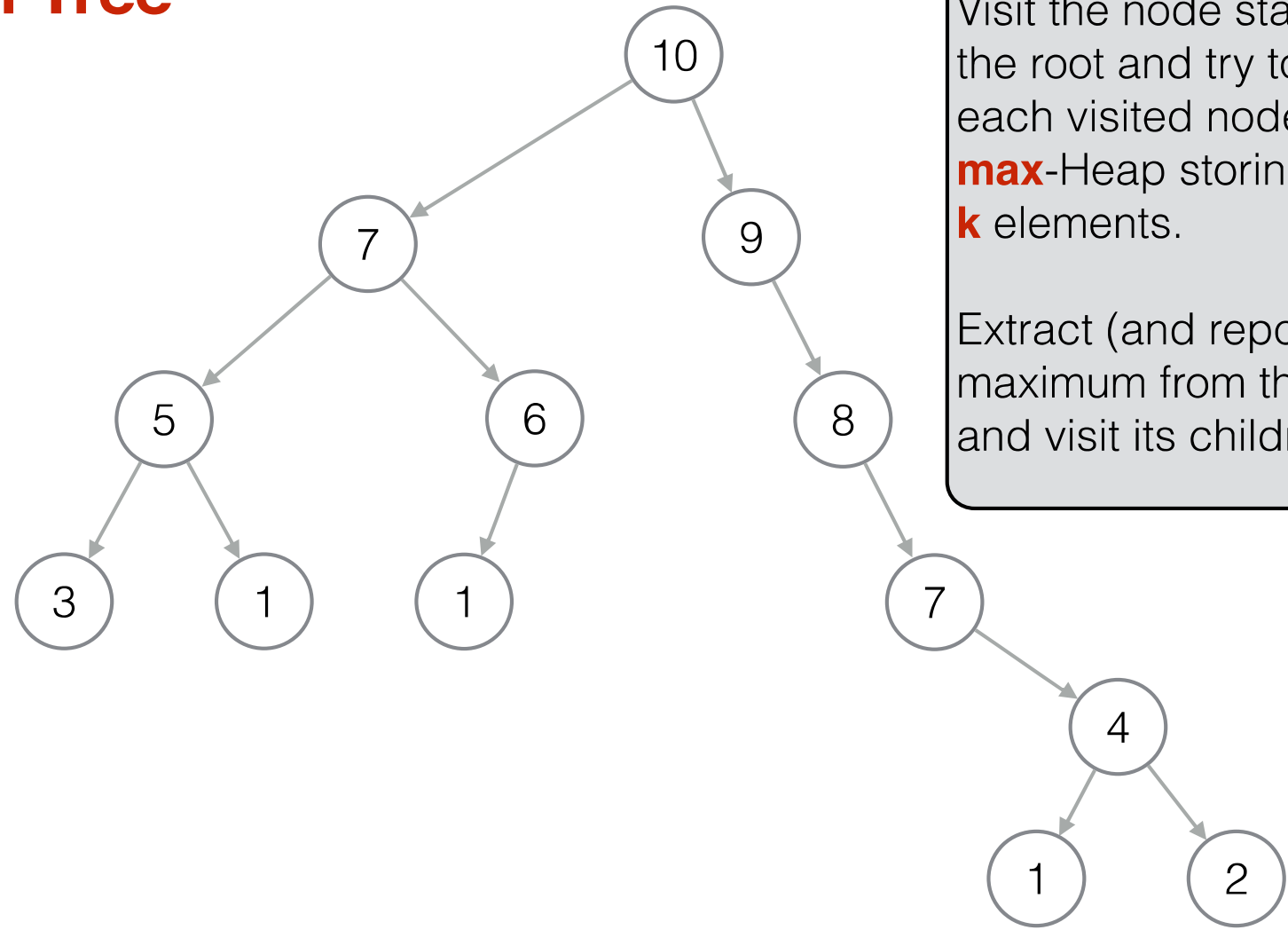
Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



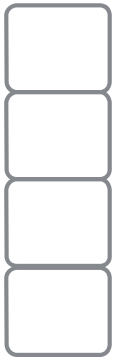
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

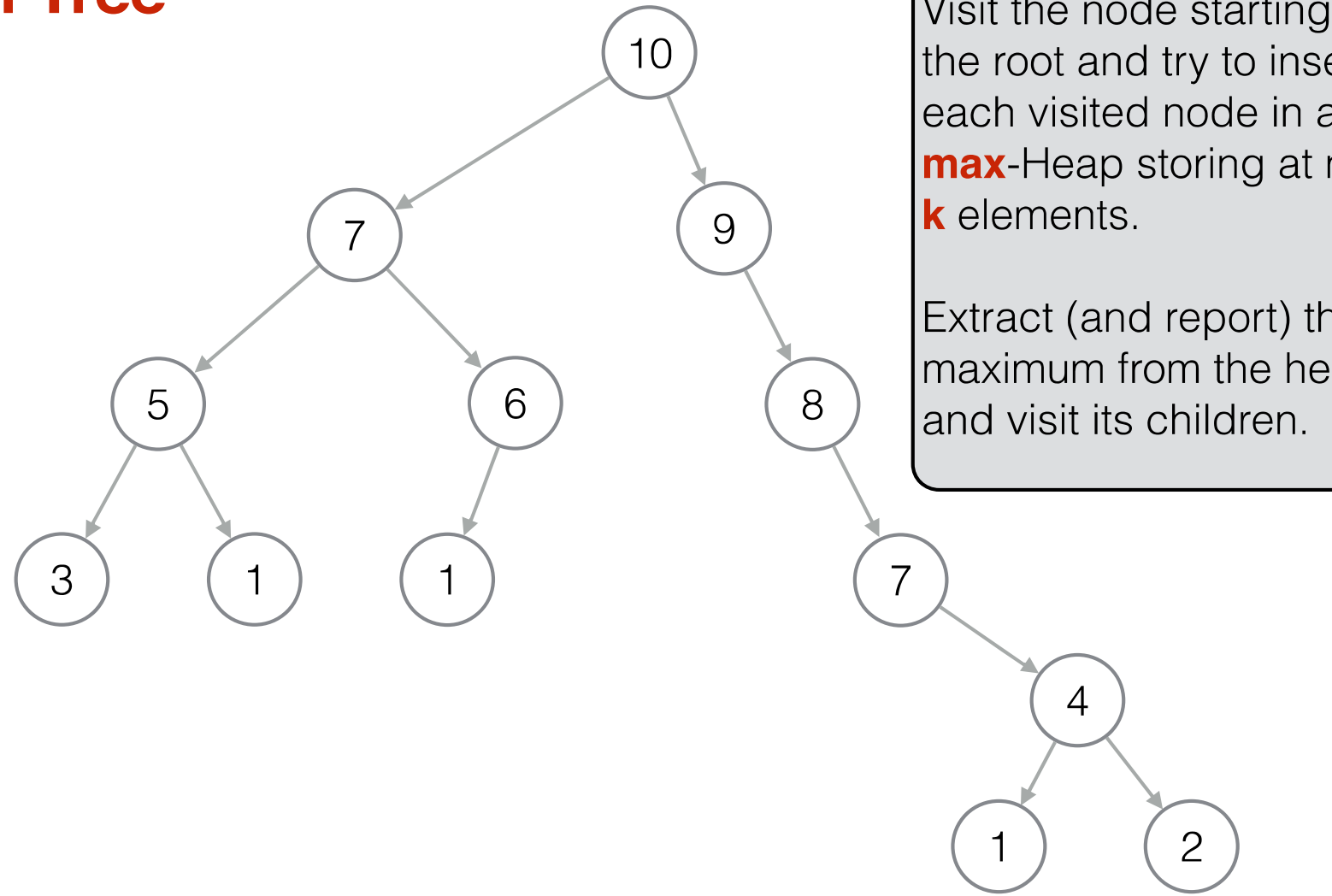
max-Heap



S

Finding Top-k

Cartesian Tree



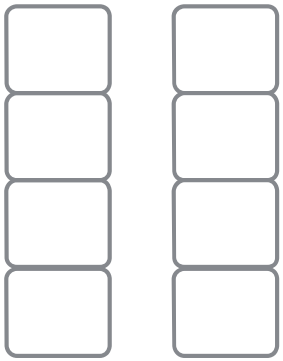
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

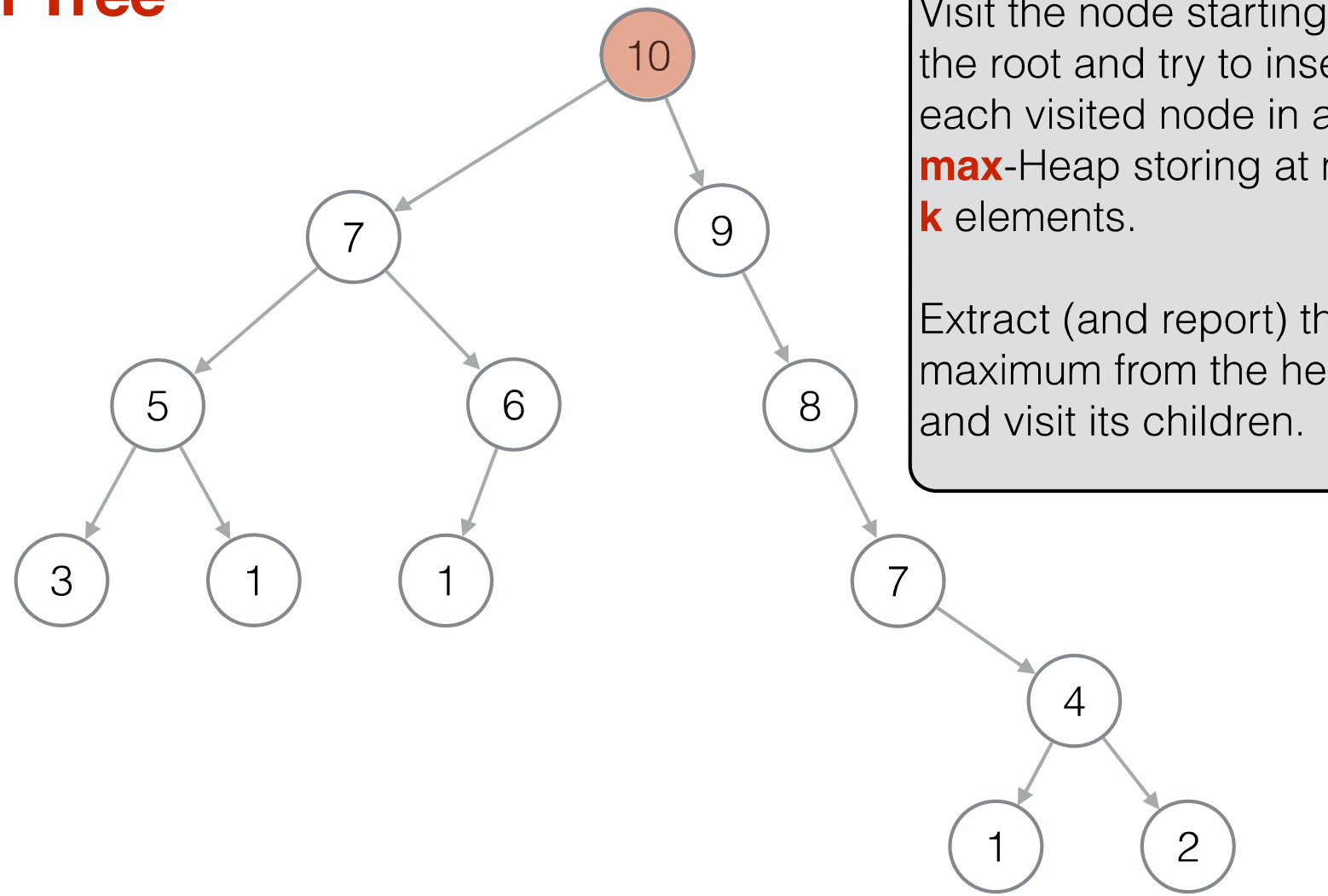
max-Heap Results



S

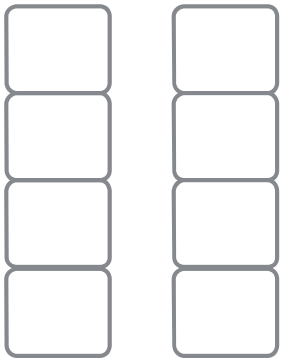
Finding Top-k

Cartesian Tree



k=4

max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

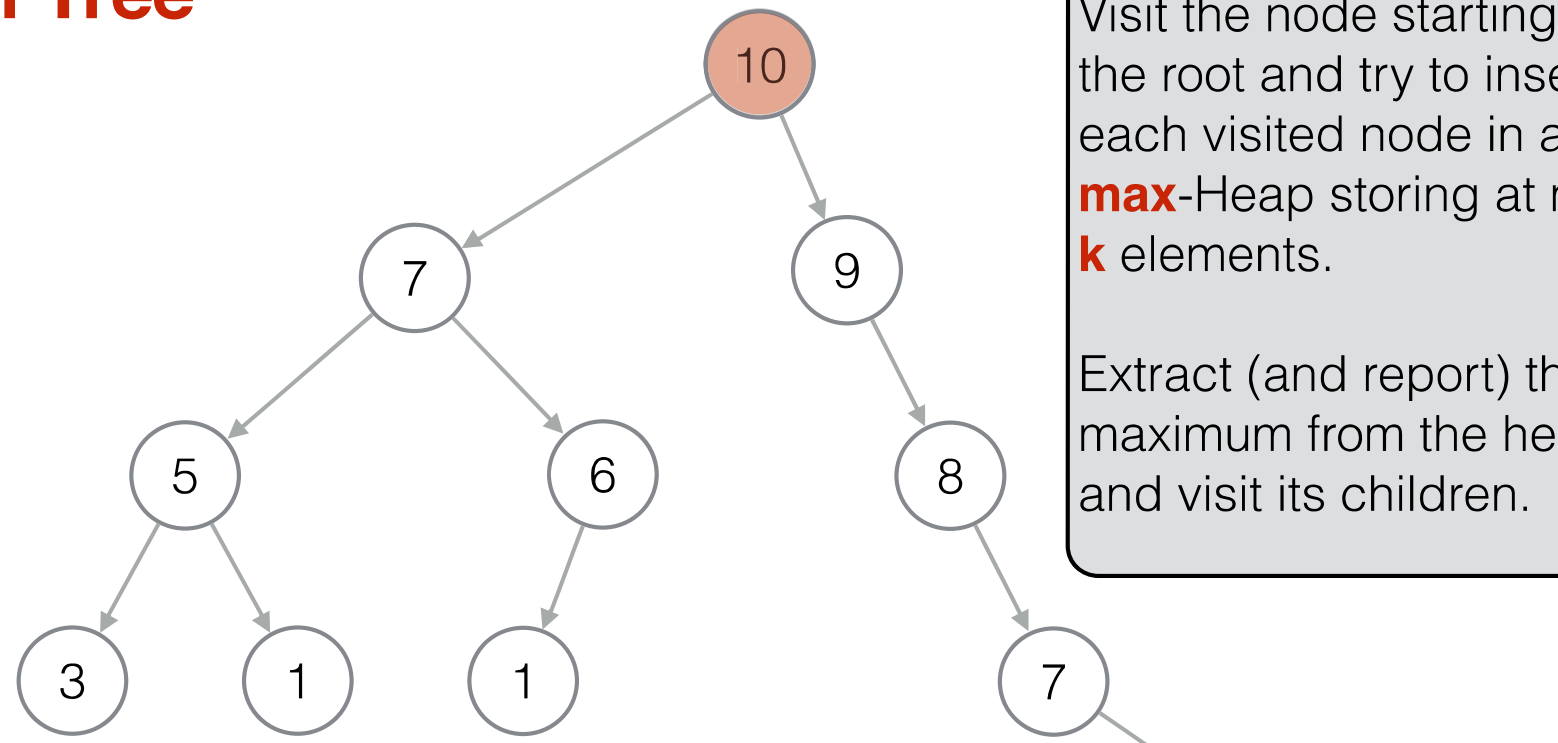
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

Finding Top-k

Cartesian Tree



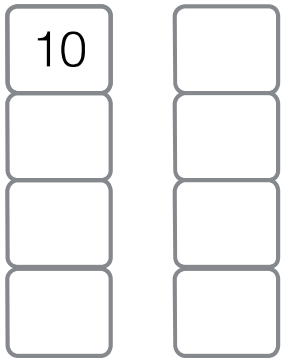
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

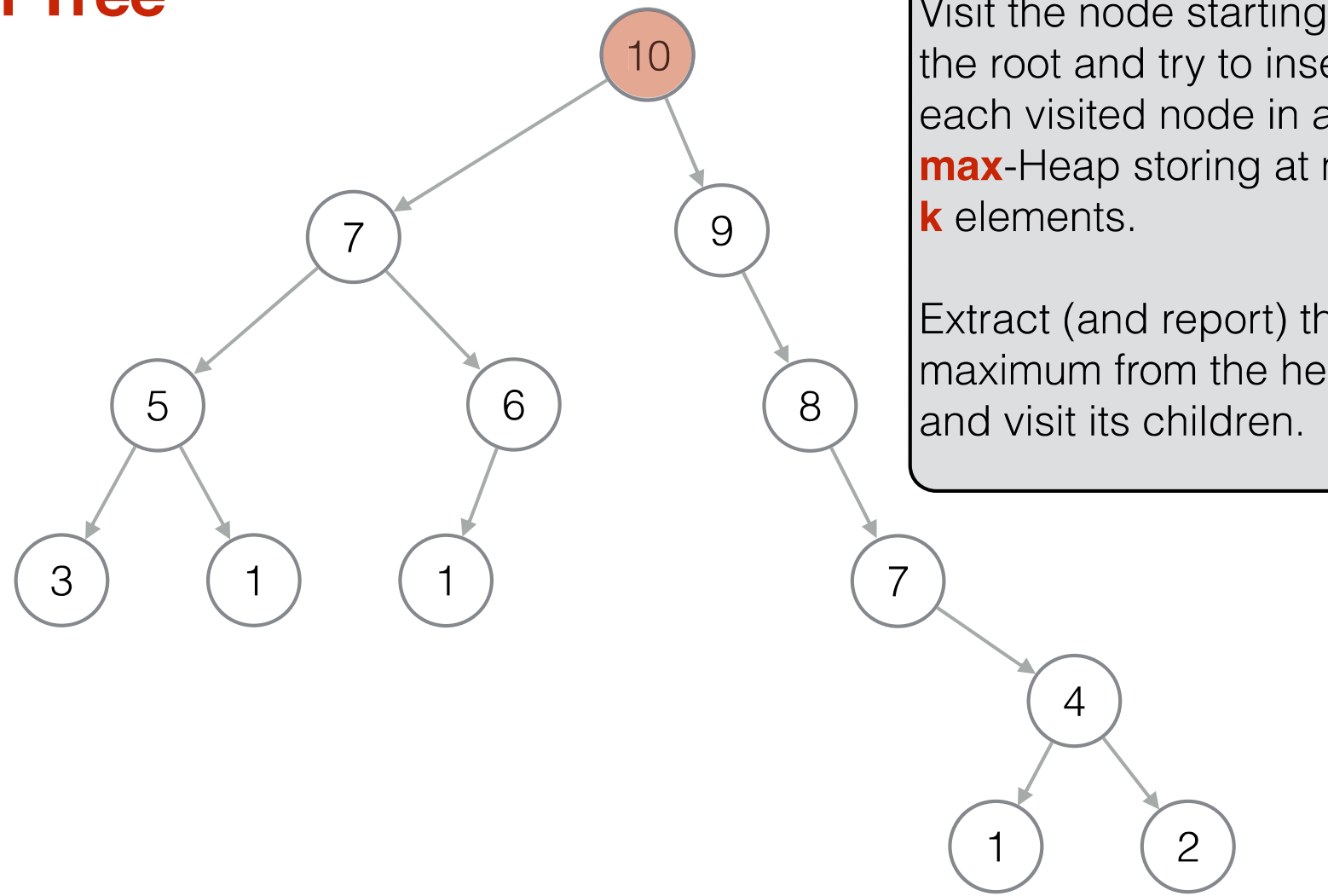
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

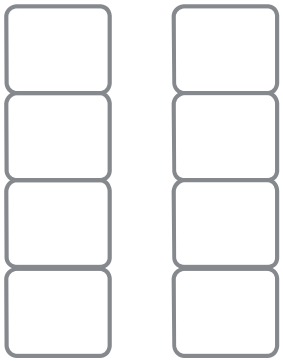
Finding Top-k

Cartesian Tree



k=4

max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

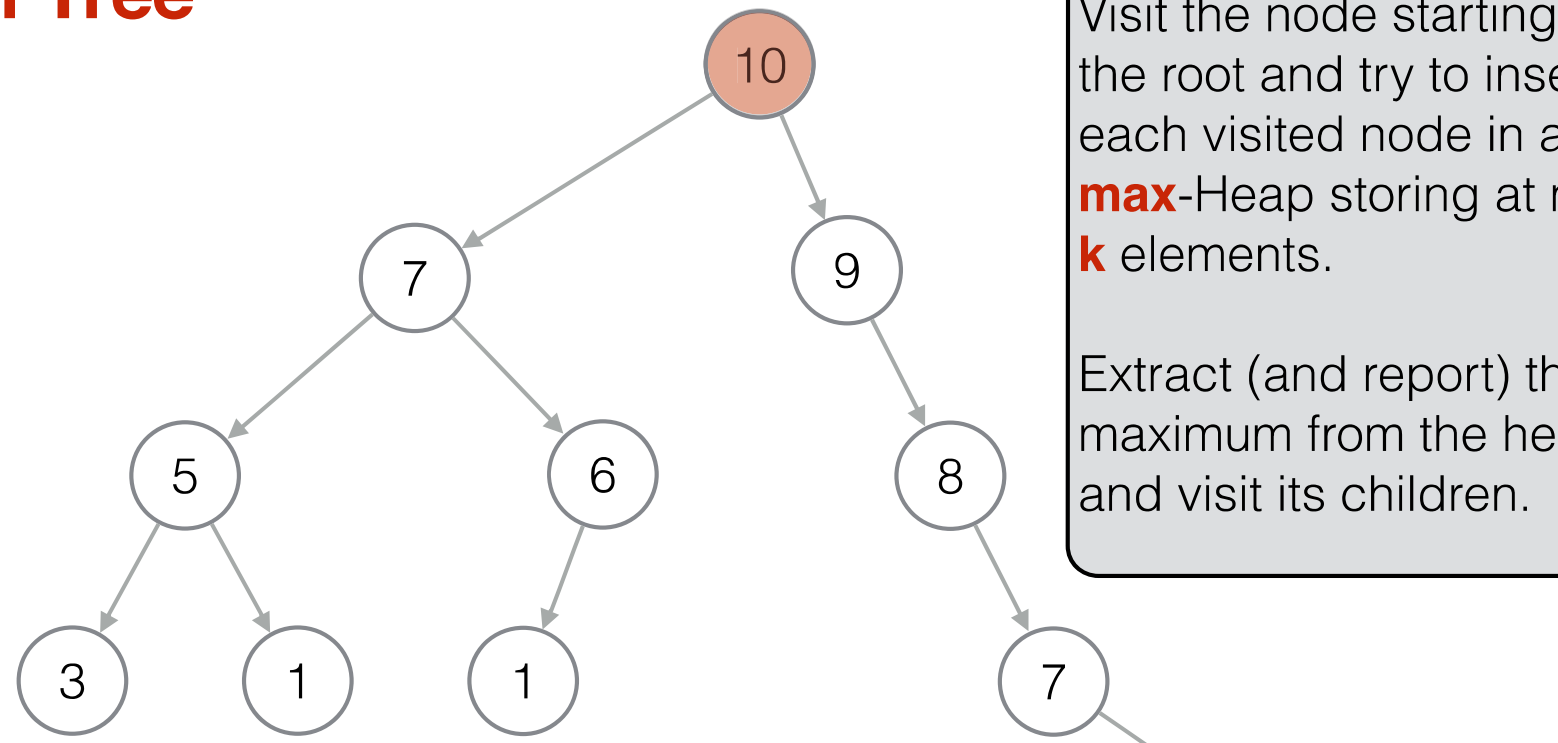
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

Finding Top-k

Cartesian Tree



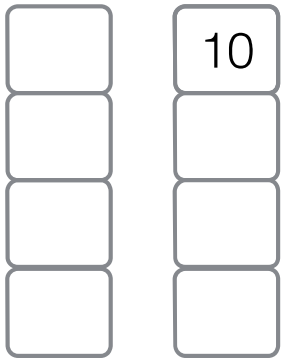
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

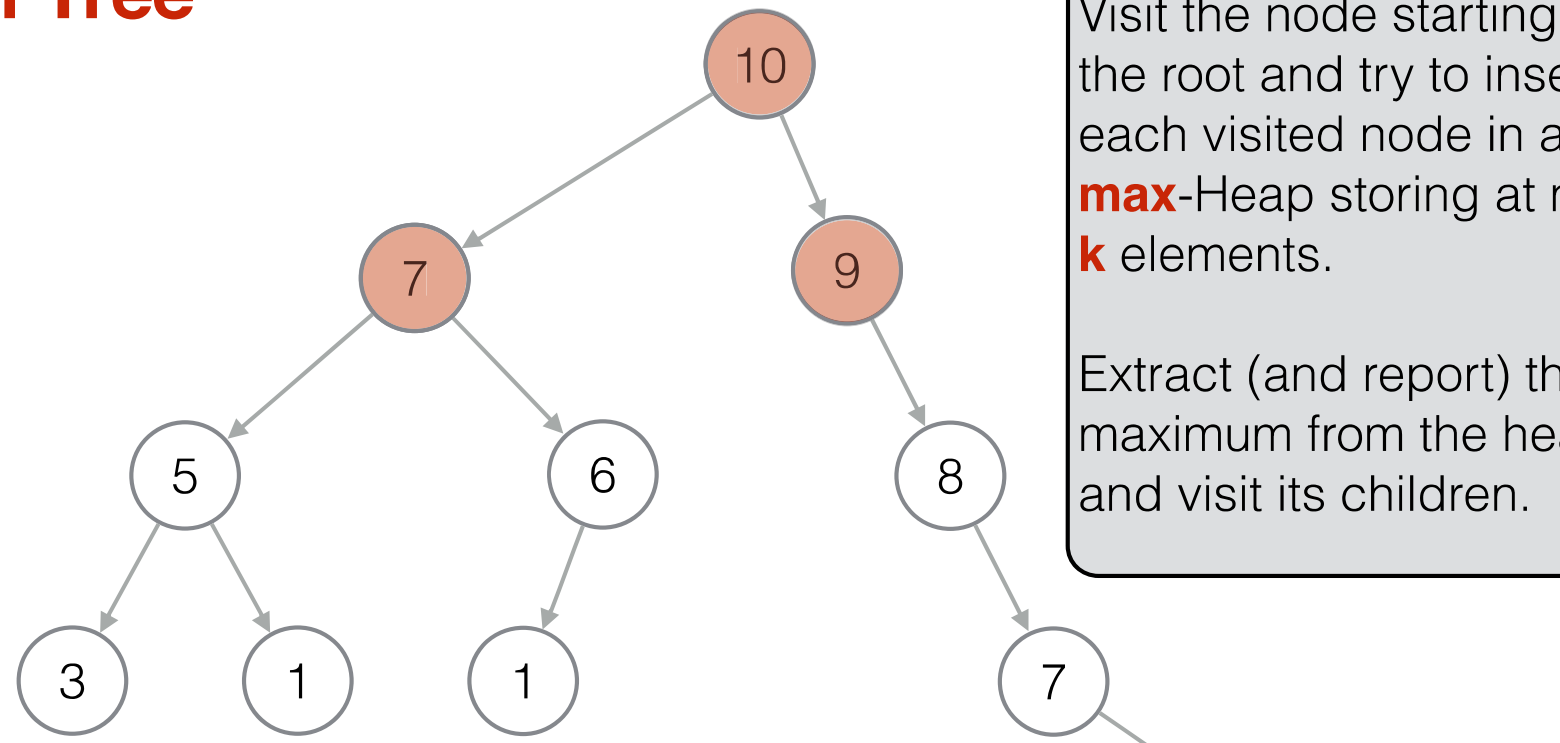
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



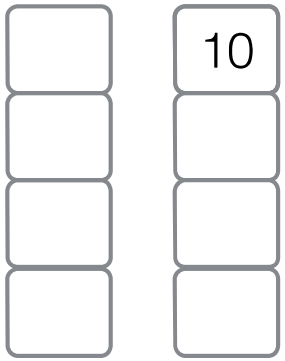
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

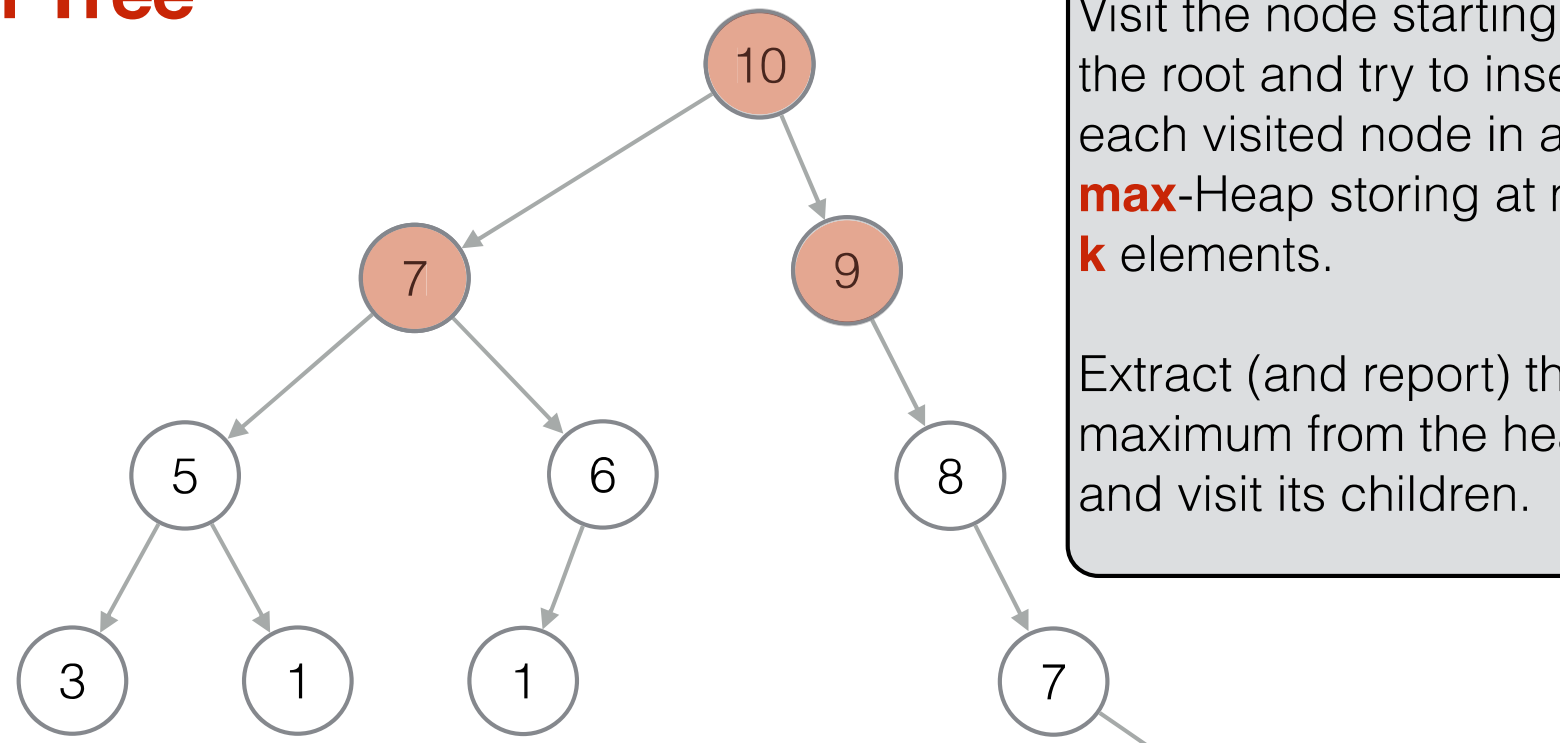
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



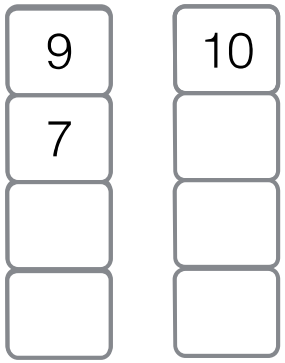
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

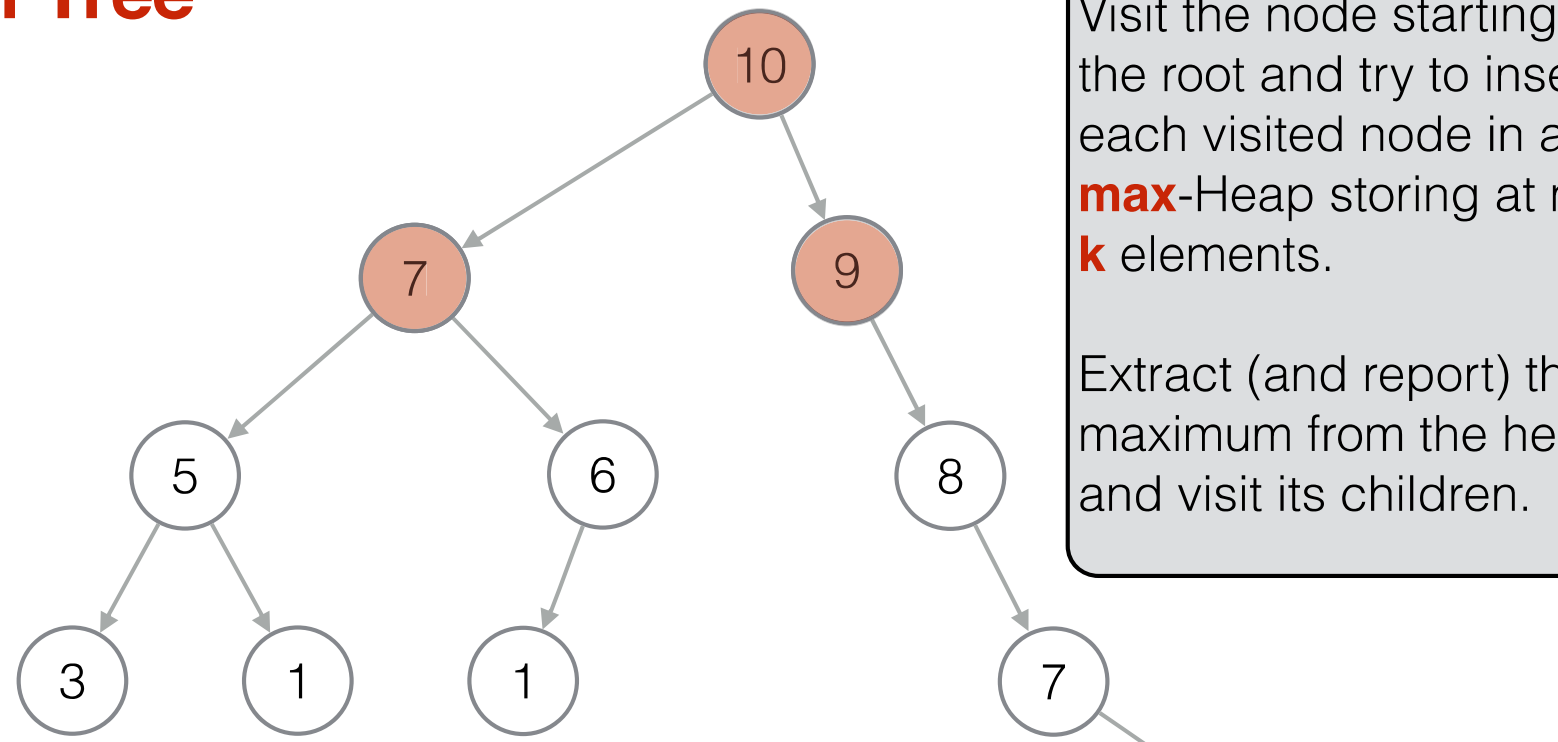
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



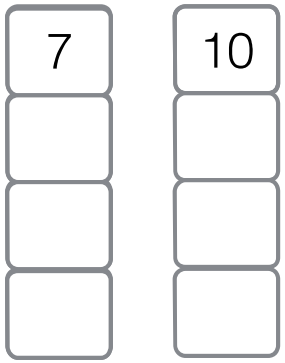
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

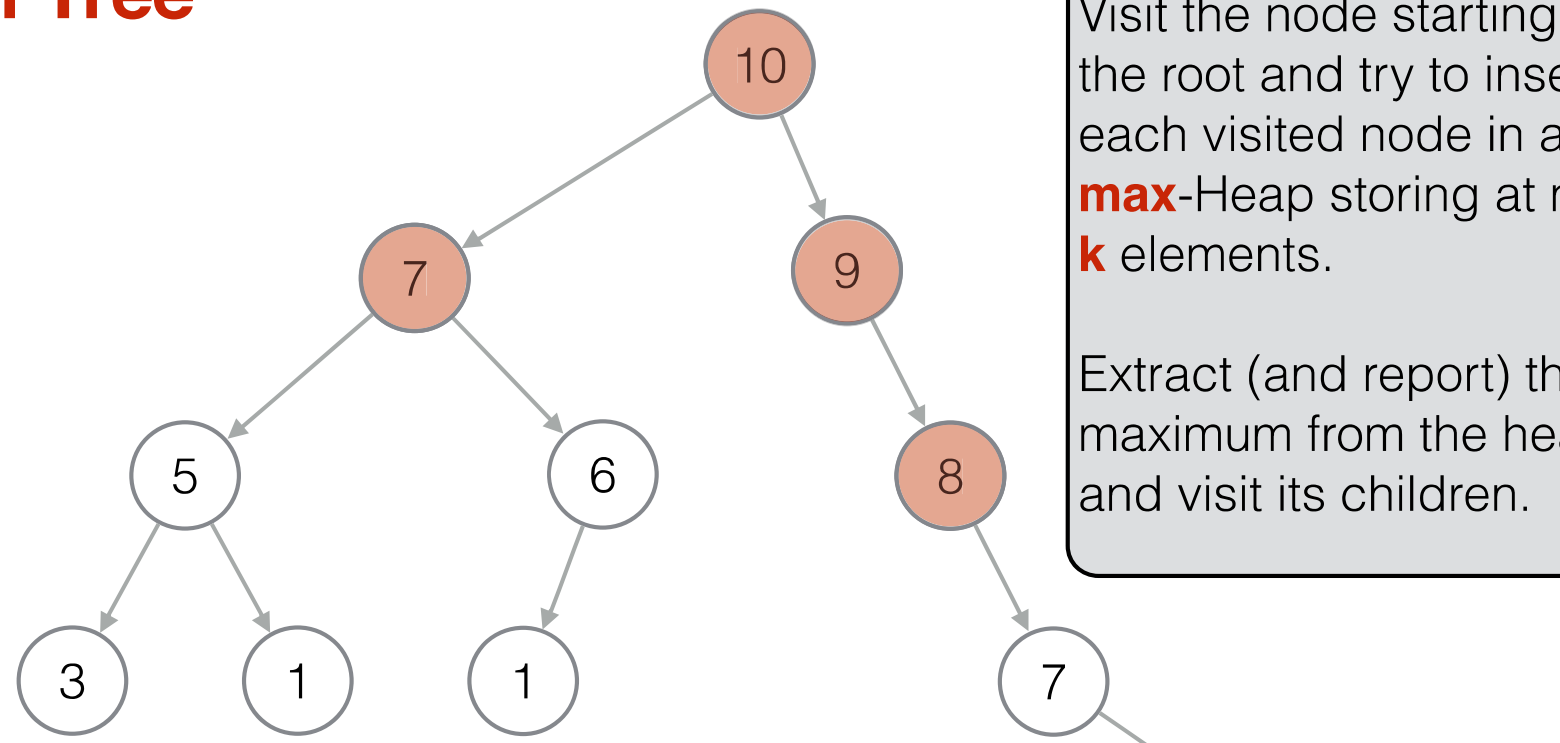
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



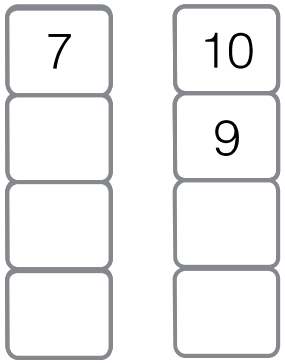
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

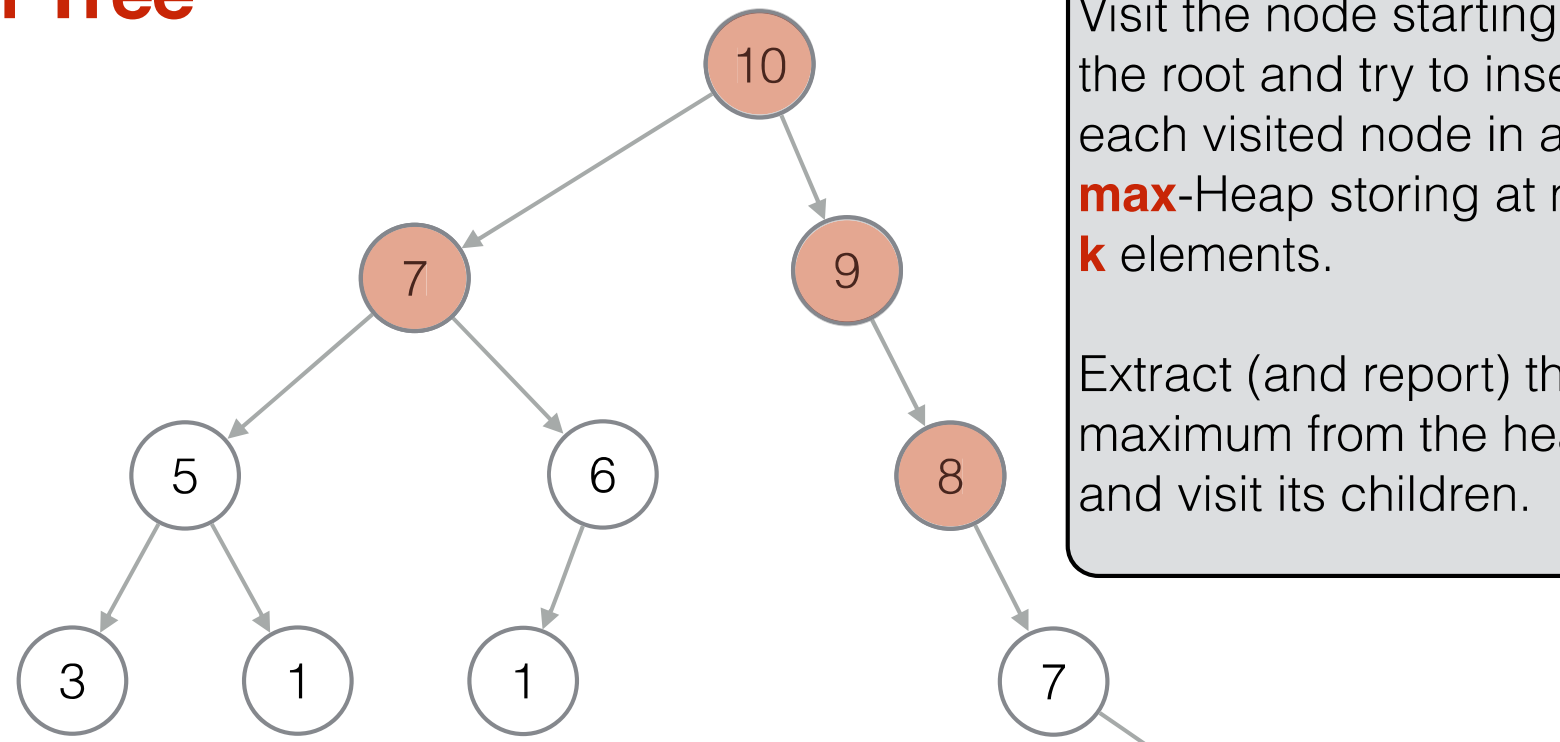
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

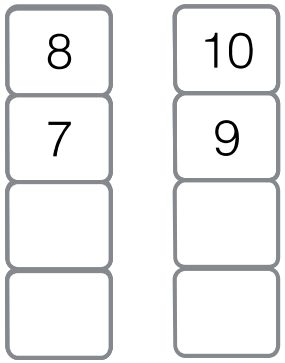
Finding Top-k

Cartesian Tree



k=4

max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

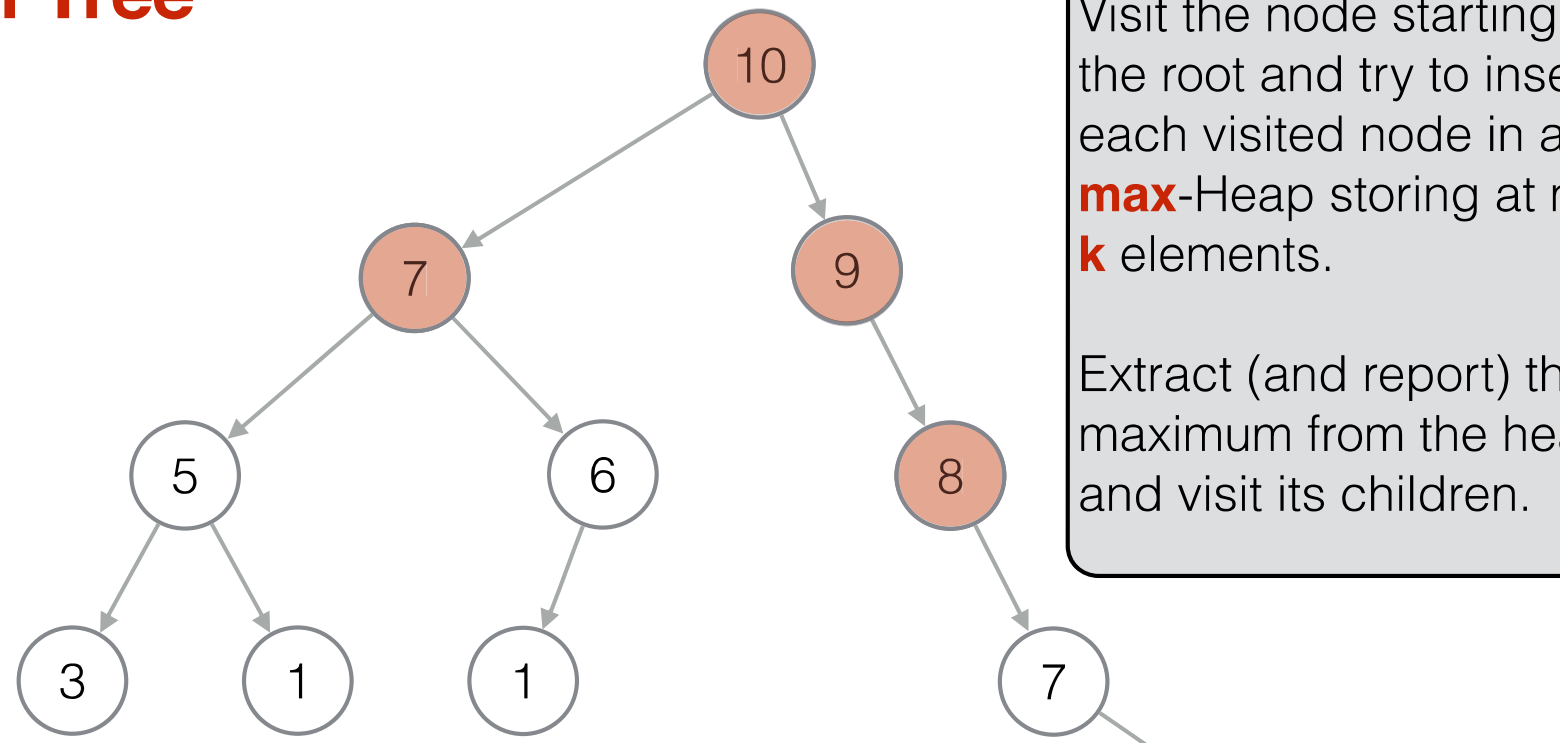
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

Finding Top-k

Cartesian Tree



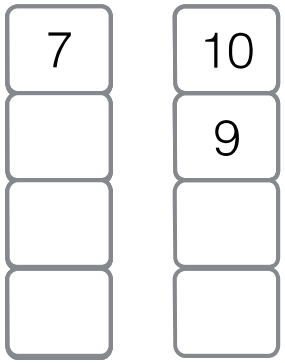
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

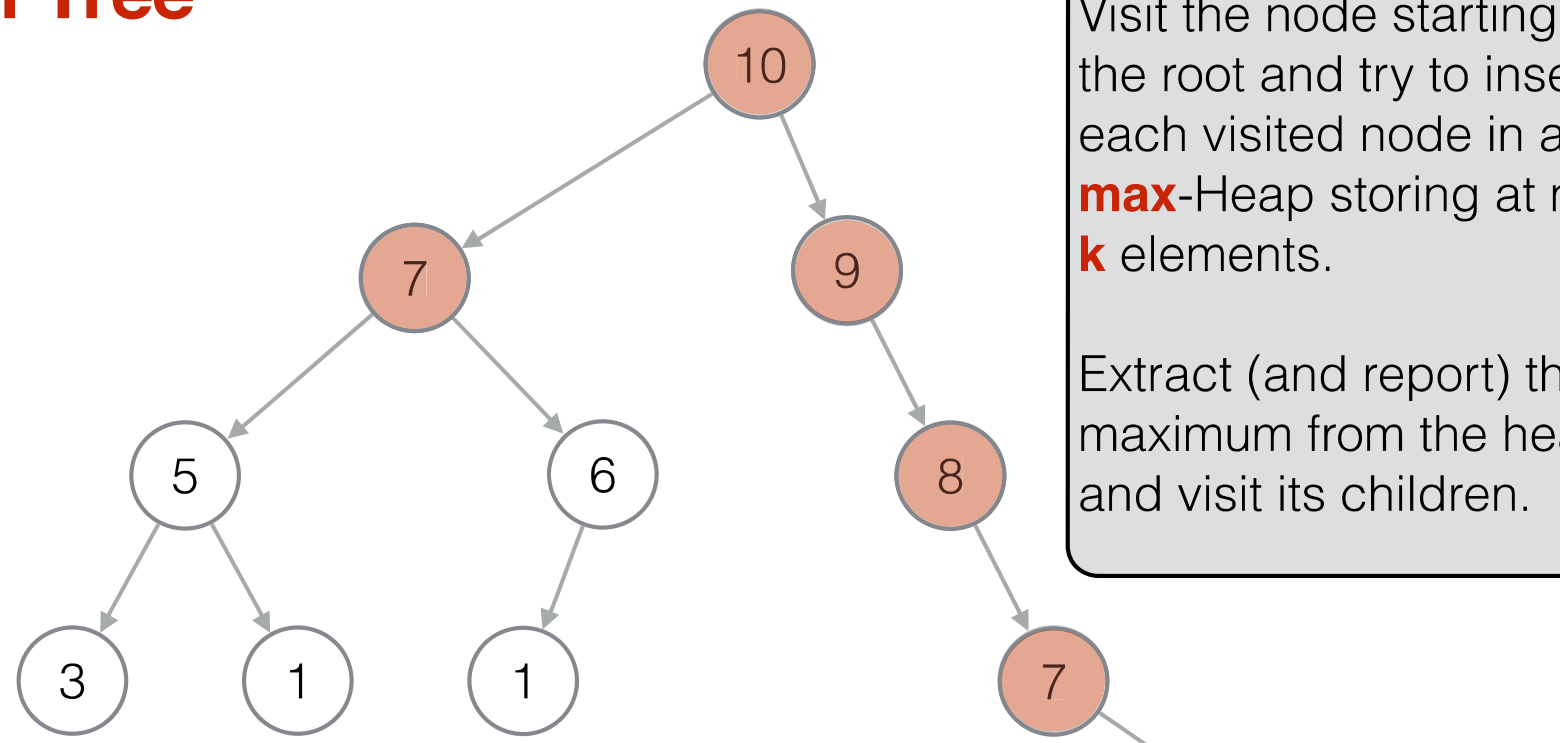
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



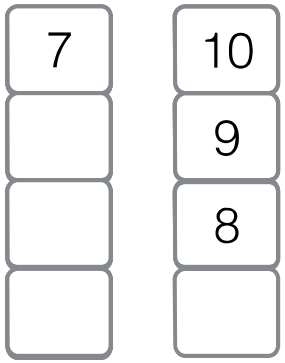
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

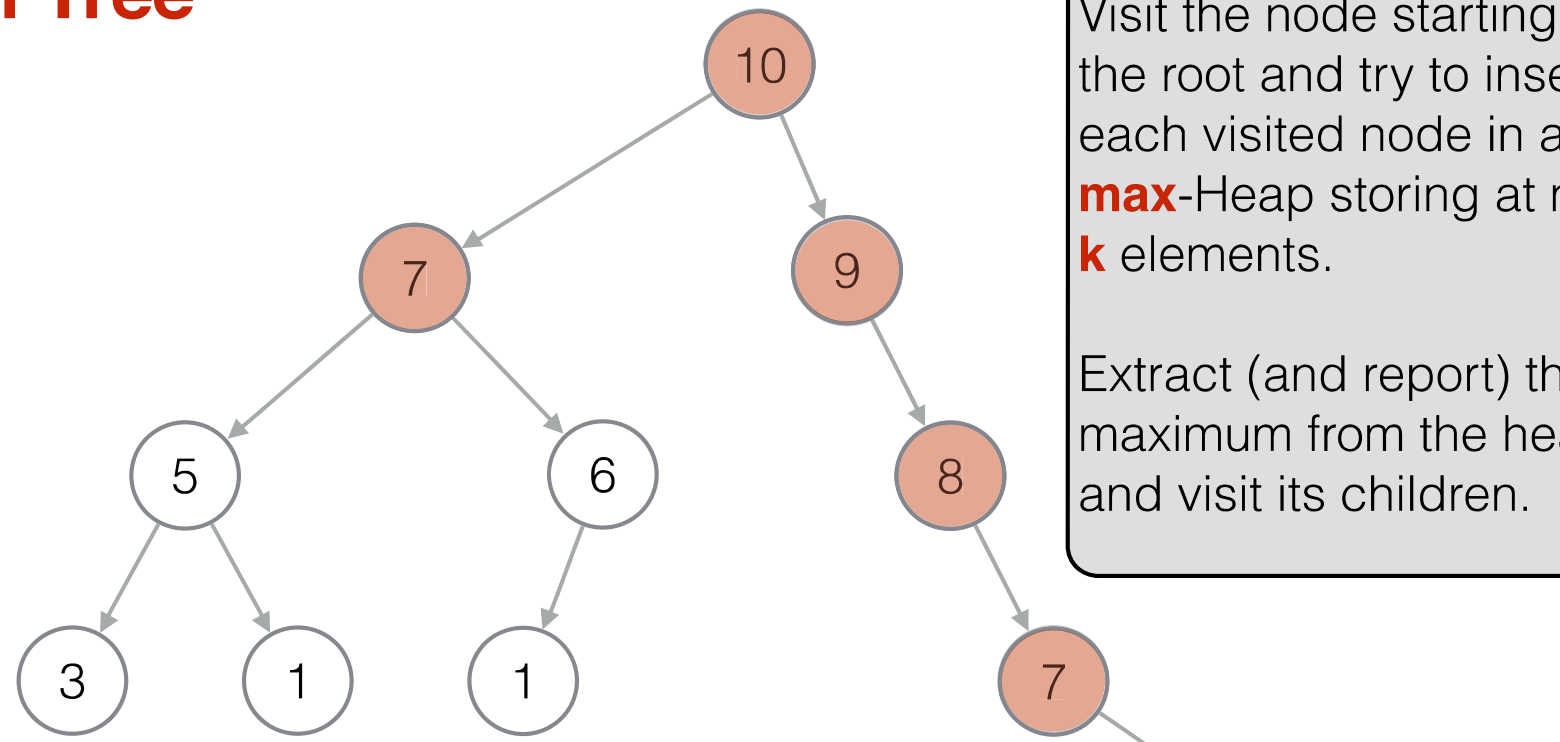
max-Heap Results



S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

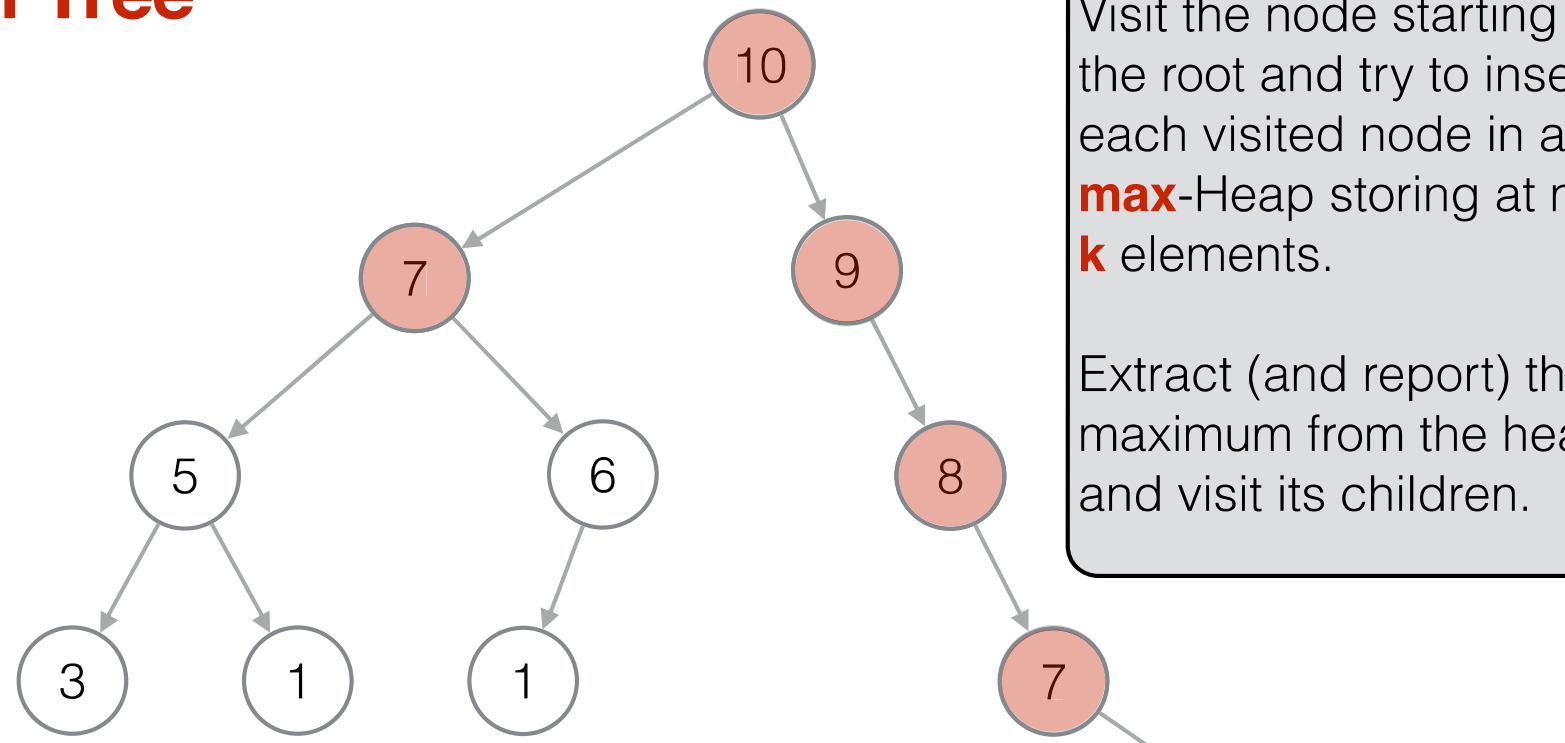
max-Heap Results

7	10
7	9
	8

S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

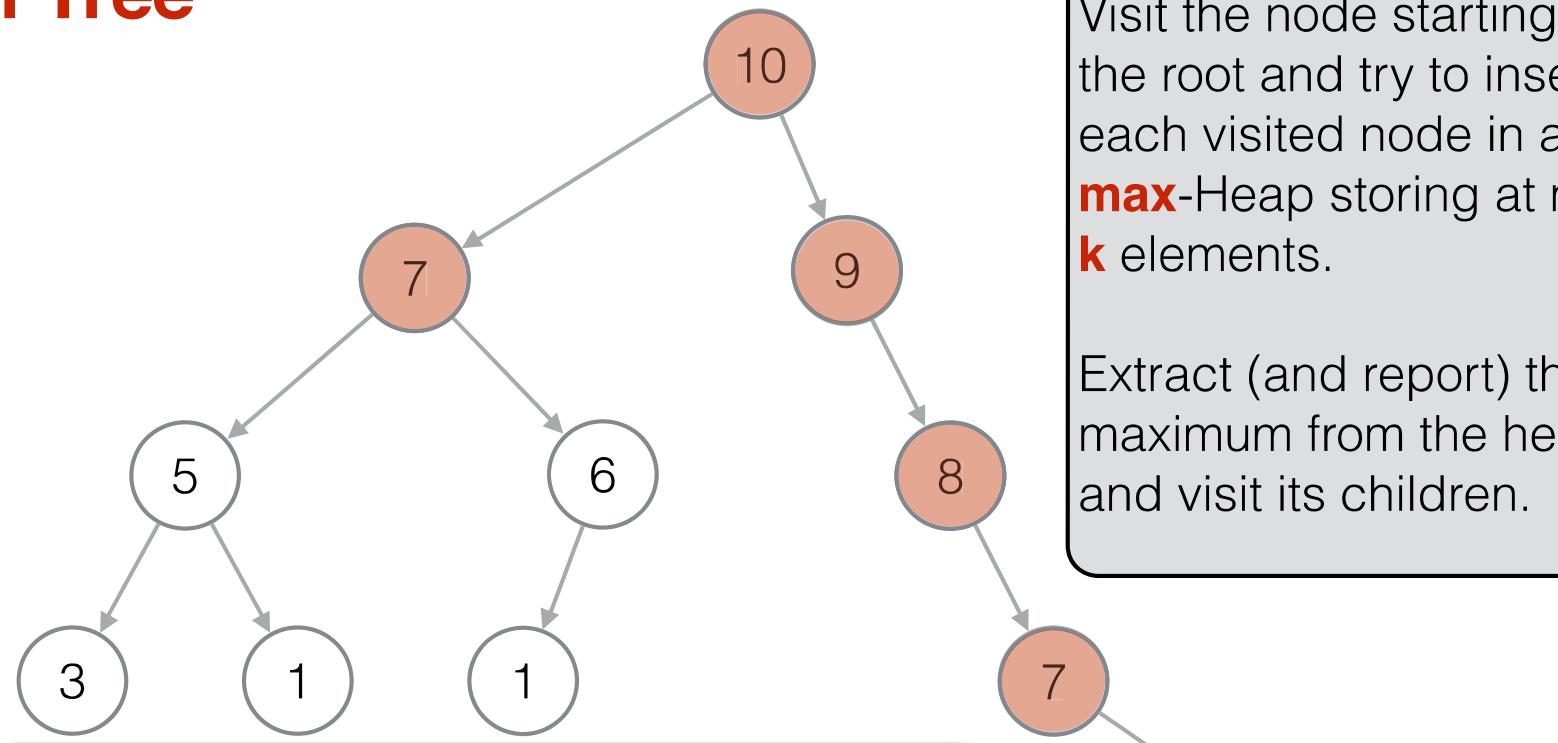
max-Heap Results

7	10
	9
	8
	7

S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



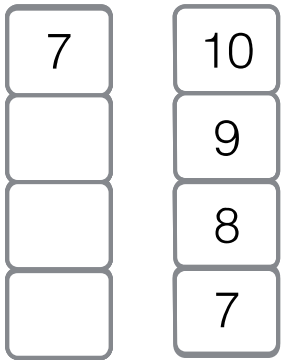
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

max-Heap Results

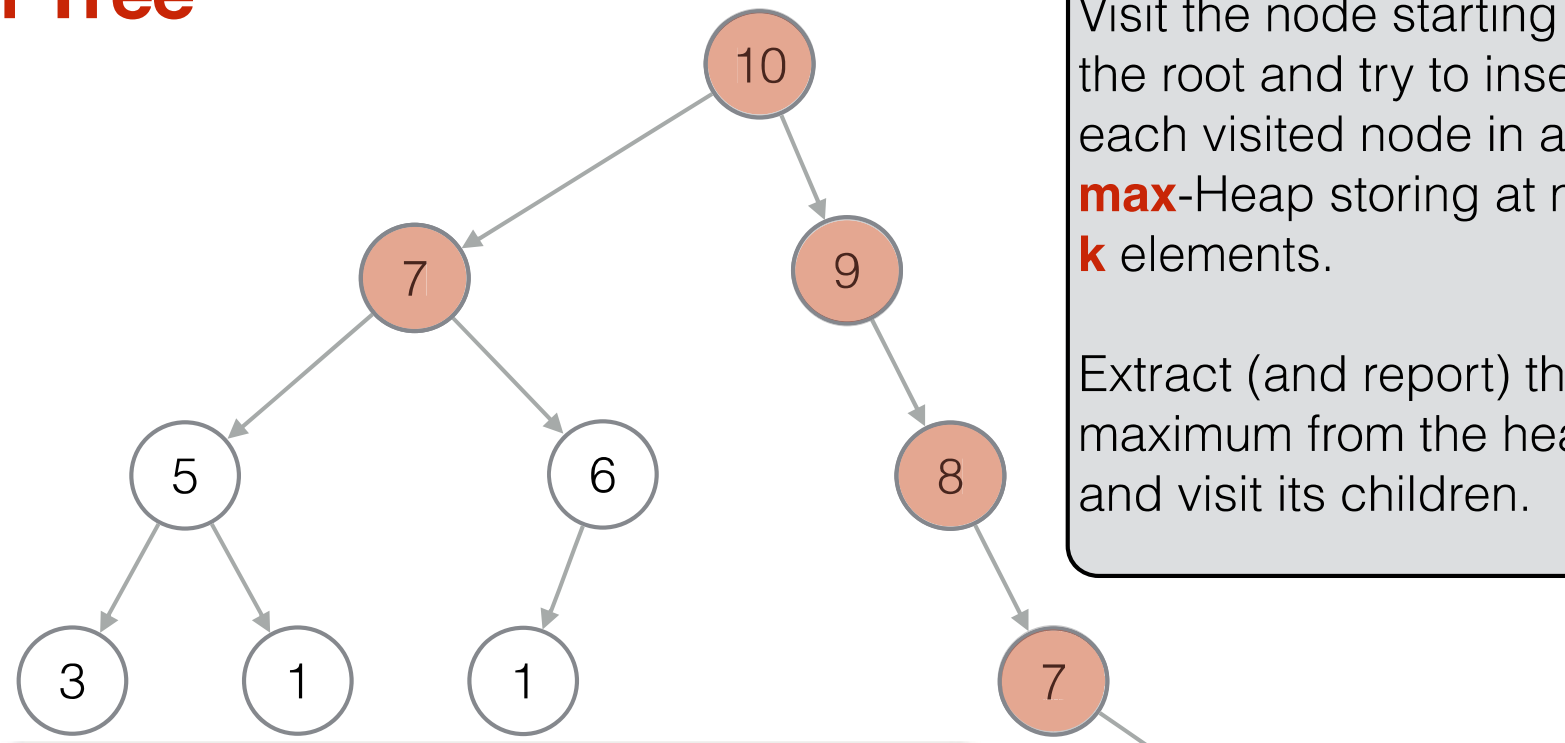


Claim: we “touch” at most 2k nodes.
⇒ Query time O(k log k)

S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Finding Top-k

Cartesian Tree



How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

Extract (and report) the maximum from the heap and visit its children.

k=4

max-Heap Results

7	10
	9
	8
	7

Claim: we "touch" at most 2k nodes.
⇒ Query time O(k log k)

Important: the cartesian tree is not built!

S … | 3 | 5 | 1 | 7 | 1 | 6 | 10 | 9 | 8 | 7 | 1 | 4 | 2 | …

Finding Top-k

Cartesian Tree

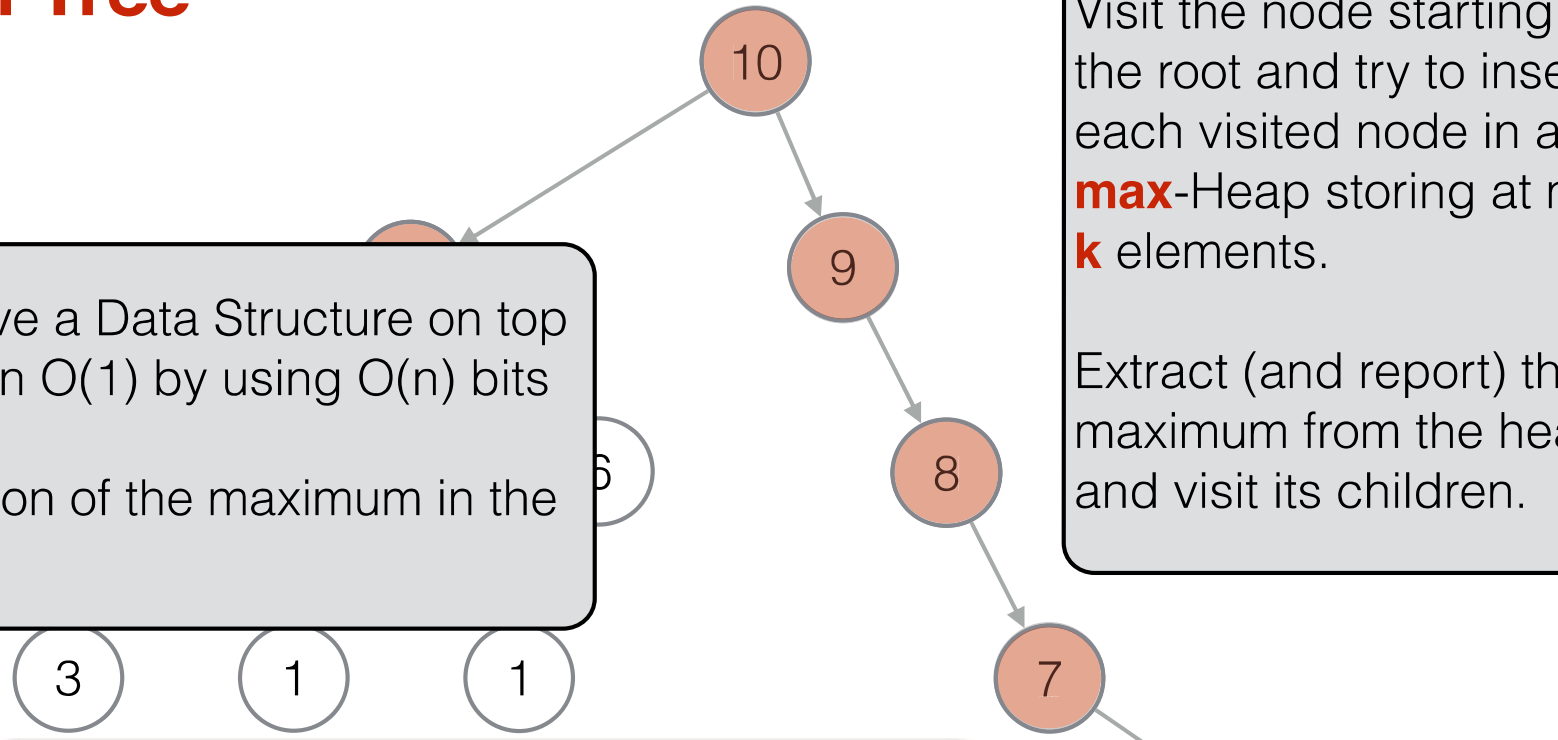
How to find Top-k?

Visit the node starting from the root and try to insert each visited node in a **max**-Heap storing at most **k** elements.

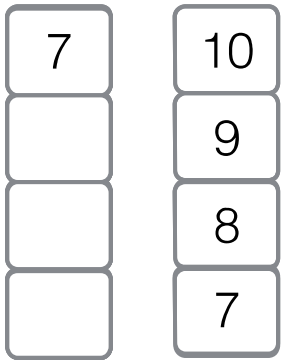
Extract (and report) the maximum from the heap and visit its children.

Assume you have a Data Structure on top of S answering in O(1) by using O(n) bits

RMQ(i,j) = position of the maximum in the range S[i,j]



max-Heap Results

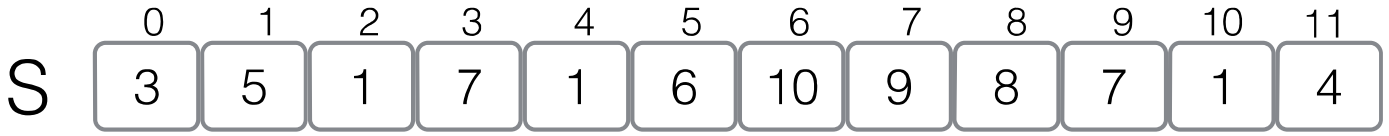


Claim: we “touch” at most 2k nodes.
⇒ Query time O(k log k)

Important: the cartesian tree is not built!

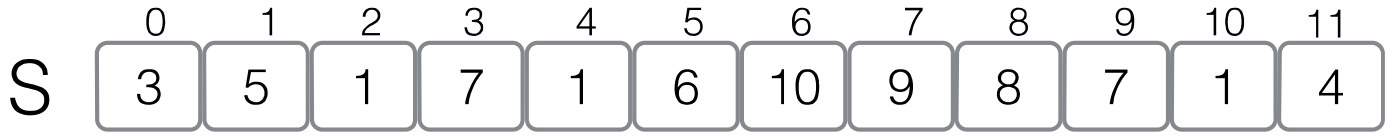
S … 3 5 1 7 1 6 10 9 8 7 1 4 2 …

Range Maximum Query (1)



Range Maximum Query (1)

Space: $O(n^2 \log n)$ bits
Query time: $O(1)$

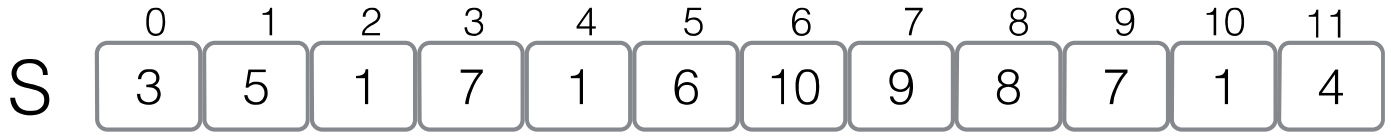


Range Maximum Query (1)

Space: $O(n^2 \log n)$ bits
Query time: $O(1)$

Precompute the answer to any possible query.

There are $O(n^2)$ distinct queries!



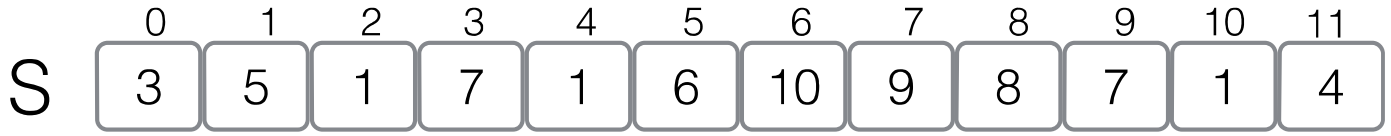
Range Maximum Query (1)

Space: $O(n^2 \log n)$ bits
Query time: $O(1)$

$M[i,j] = RMQ(i,j)$

Precompute the answer to any possible query.

There are $O(n^2)$ distinct queries!



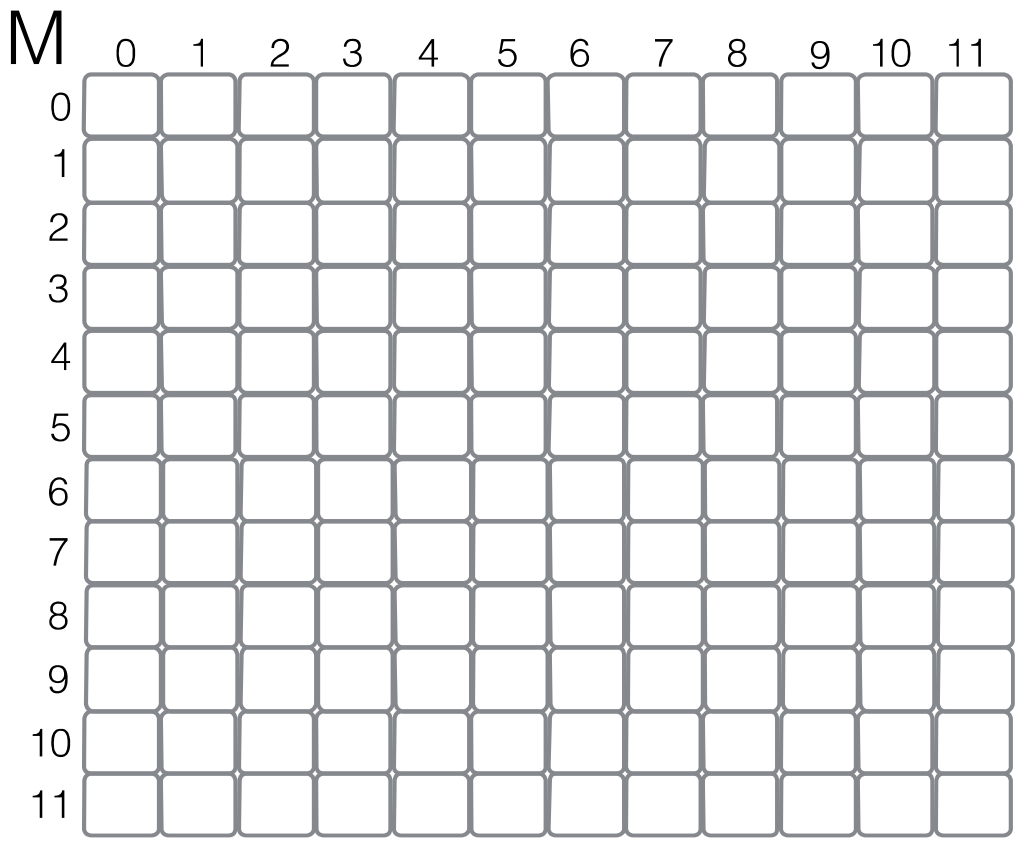
Range Maximum Query (1)

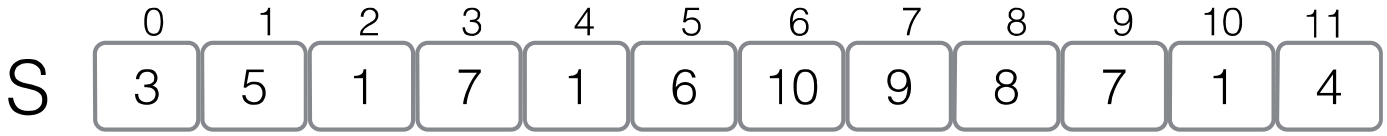
Space: $O(n^2 \log n)$ bits
Query time: O(1)

Precompute the answer to any possible query.

There are $O(n^2)$ distinct queries!

M[i,j] = RMQ(i,j)





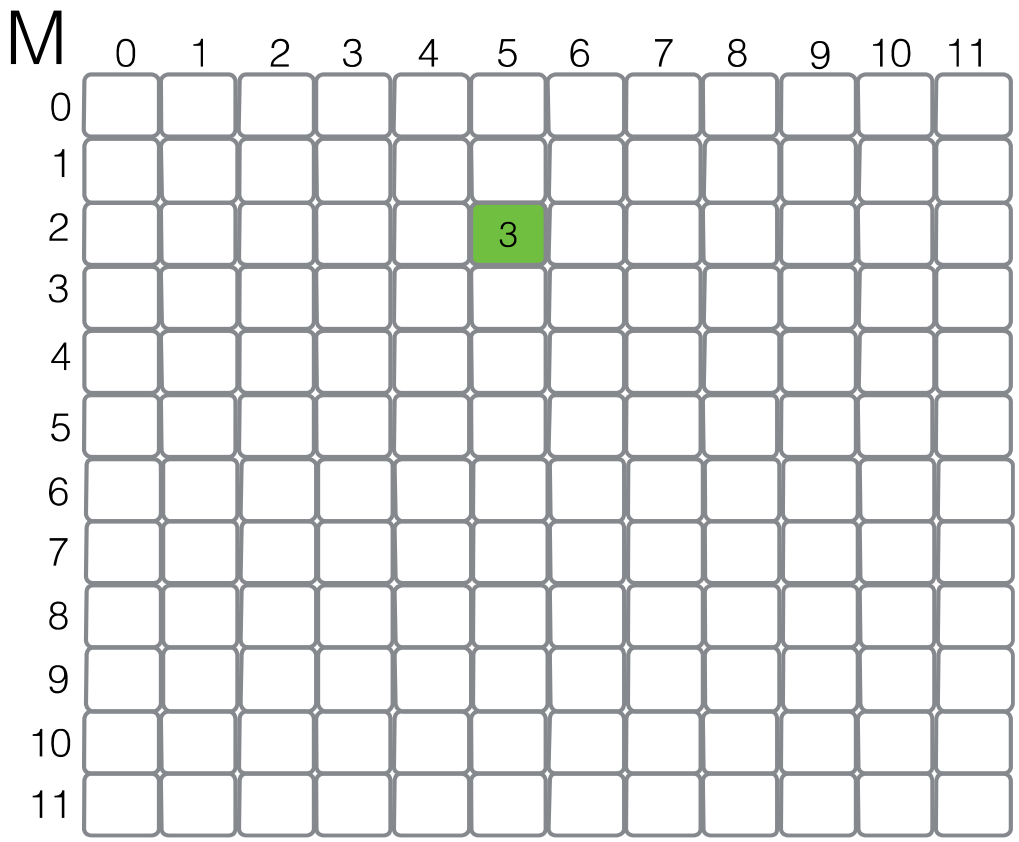
Range Maximum Query (1)

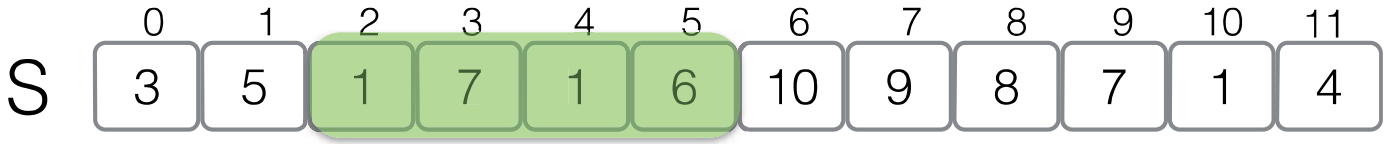
Space: $O(n^2 \log n)$ bits
Query time: O(1)

Precompute the answer to any possible query.

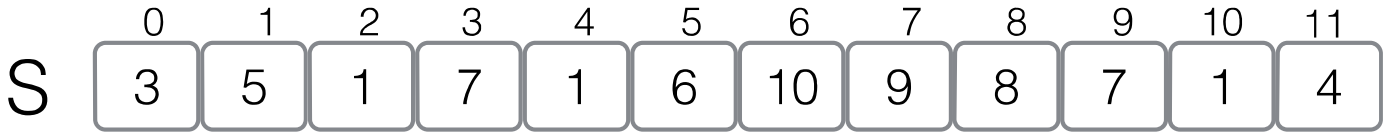
There are $O(n^2)$ distinct queries!

M[i,j] = RMQ(i,j)



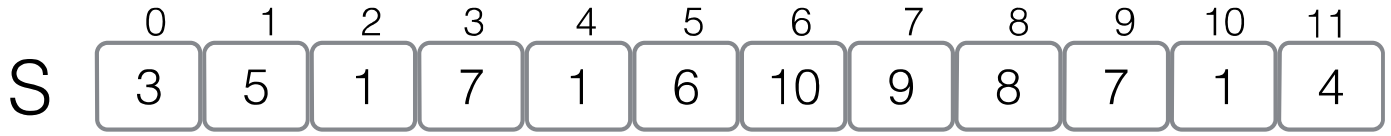


Range Maximum Query (2)



Range Maximum Query (2)

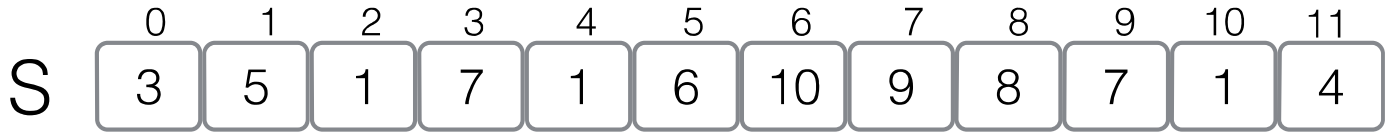
Space: $O(n \log^2 n)$ bits
Query time: $O(1)$



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

Maximum in a interval is the max between the maxima of any its subintervals



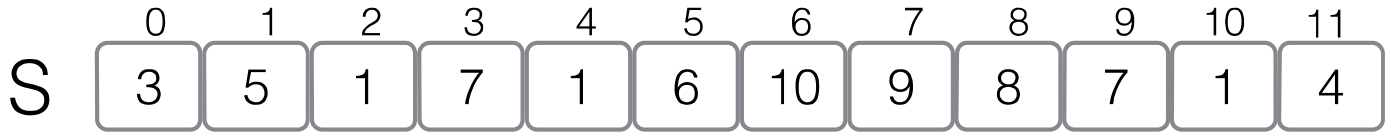
Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum in a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum in a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.

$$M[i,j] = RMQ(i,i+2^j)$$

M

	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

S

0	1	2	3	4	5	6	7	8	9	10	11
3	5	1	7	1	6	10	9	8	7	1	4

Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum in a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.

$$M[i,j] = RMQ(i,i+2^j)$$

M

	0	1	2	3	4
0					
1				?	
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

S

0	1	2	3	4	5	6	7	8	9	10	11
3	5	1	7	1	6	10	9	8	7	1	4

Range Maximum Query (2)

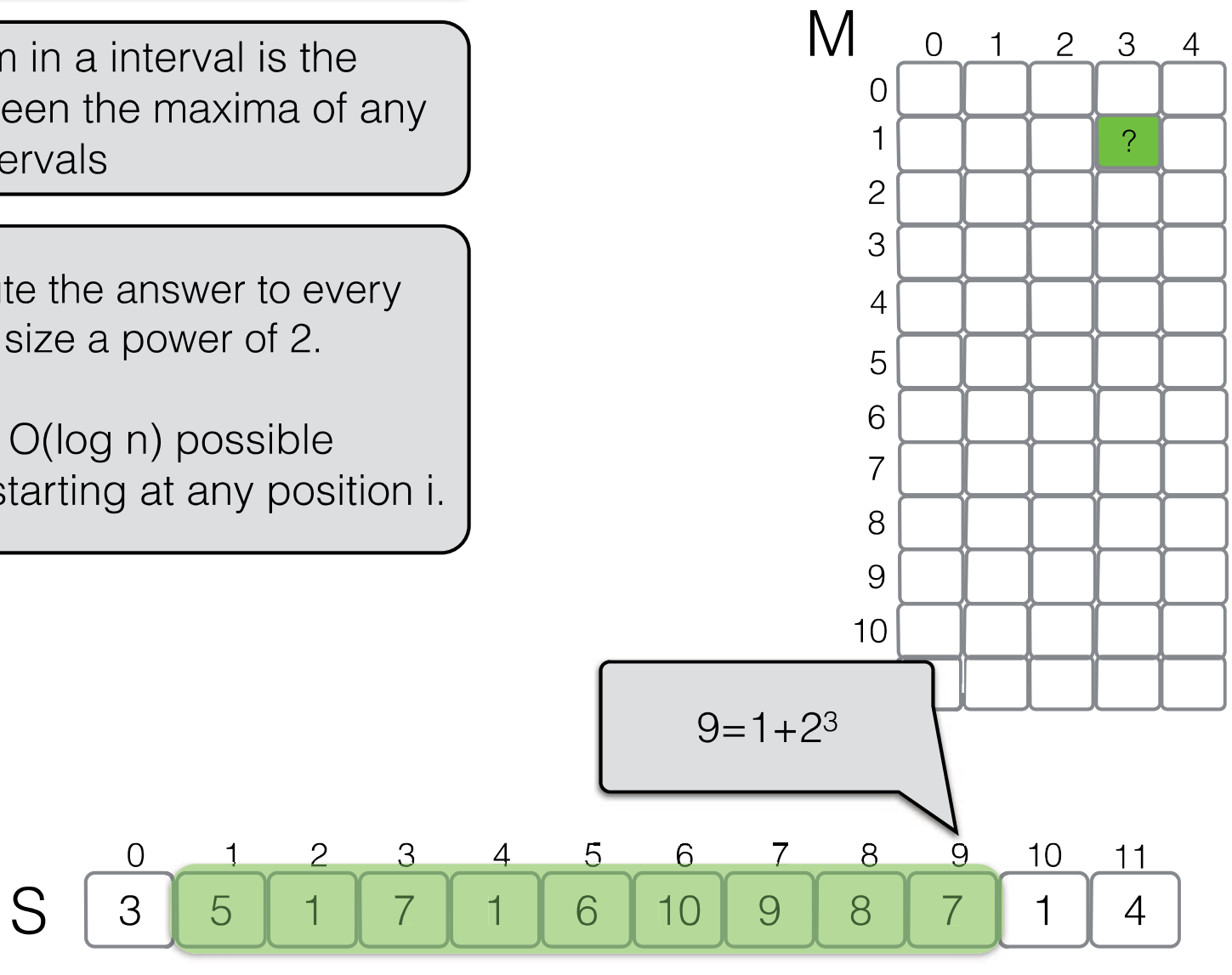
Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

Maximum in a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

$$M[i,j] = RMQ(i, i+2^j)$$



Range Maximum Query (2)

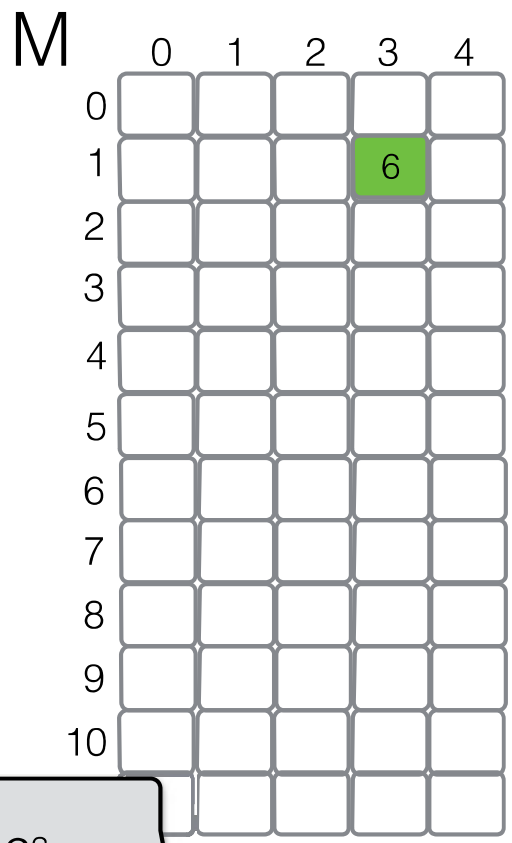
Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

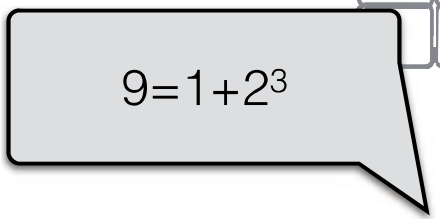
Maximum in a interval is the max between the maxima of any its subintervals

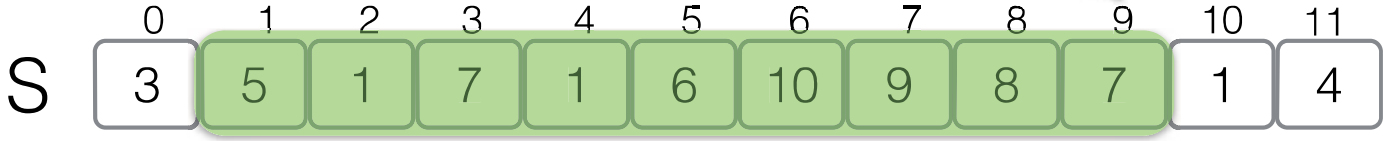
Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

$$M[i,j] = RMQ(i,i+2^j)$$







Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.

$$M[i,j] = RMQ(i,i+2^j)$$

M

	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

S

0	1	2	3	4	5	6	7	8	9	10	11
3	5	1	7	1	6	10	9	8	7	1	4

Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.

RMQ(1,7) =

$M[i,j] = RMQ(i,i+2^j)$

M

	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

S

0	1	2	3	4	5	6	7	8	9	10	11
3	5	1	7	1	6	10	9	8	7	1	4

Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: O(1)

Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are O(log n) possible intervals starting at any position i.

$M[i,j] = RMQ(i,i+2^j)$

M

	0	1	2	3	4
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

S

0	1	2	3	4	5	6	7	8	9	10	11
3	5	1	7	1	6	10	9	8	7	1	4

Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

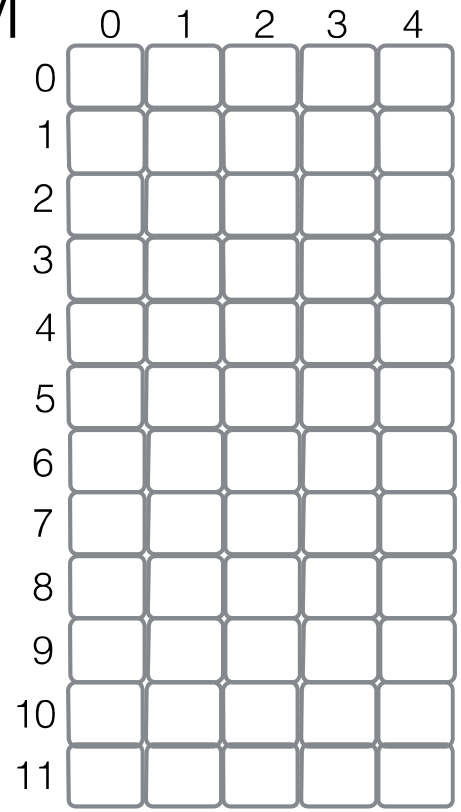
Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

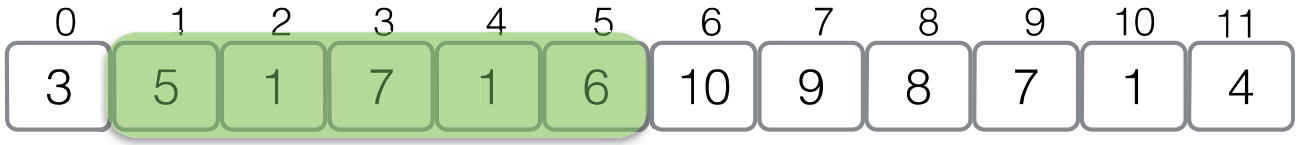
$M[i,j] = RMQ(i,i+2^j)$

M



$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

S



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

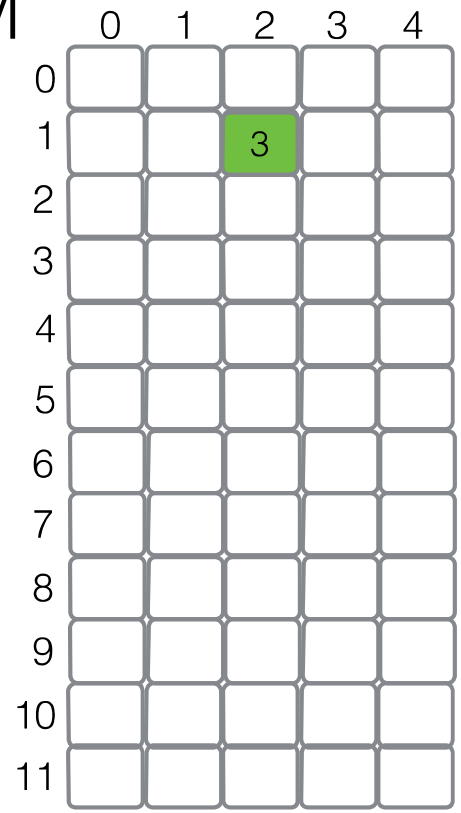
Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

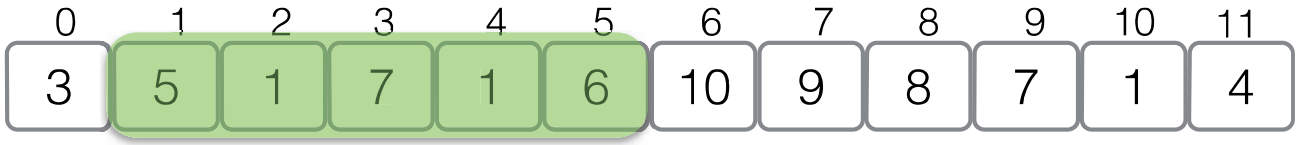
$$M[i,j] = RMQ(i,i+2^j)$$

M



$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

S



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

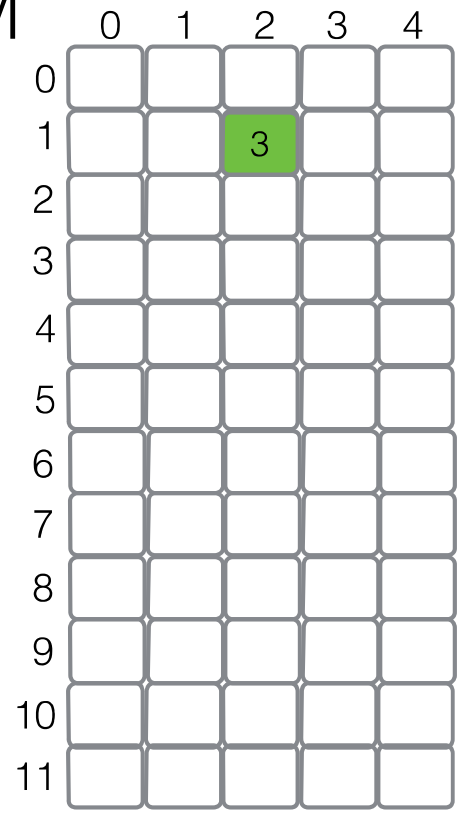
Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

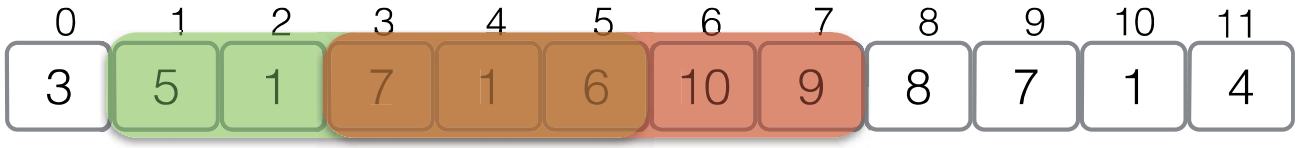
$M[i,j] = RMQ(i,i+2^j)$

M



$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

S



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

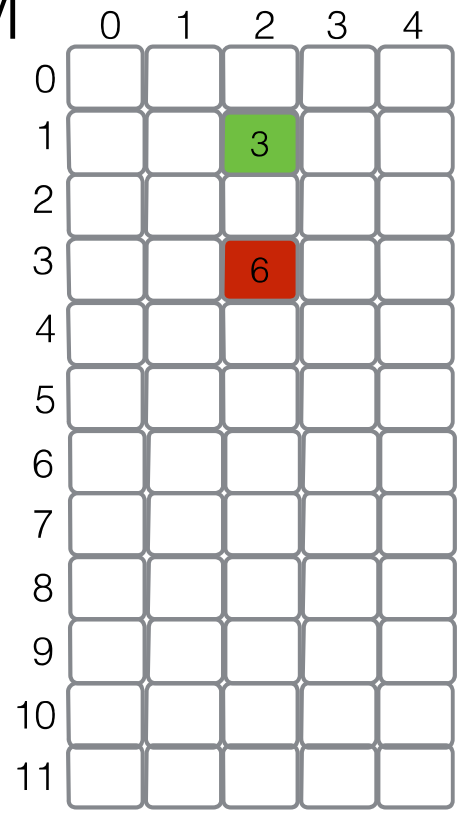
Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

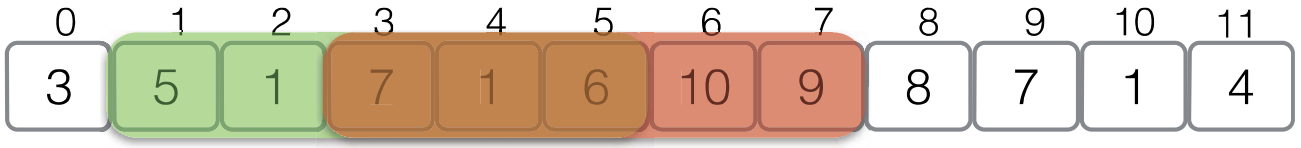
$$M[i,j] = RMQ(i,i+2^j)$$

M



$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

S



Range Maximum Query (2)

Space: $O(n \log^2 n)$ bits
Query time: $O(1)$

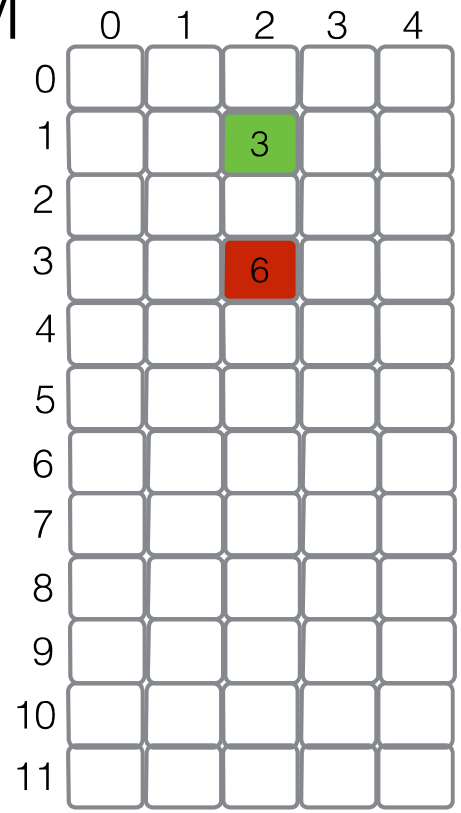
Maximum of a interval is the max between the maxima of any its subintervals

Precompute the answer to every interval of size a power of 2.

There are $O(\log n)$ possible intervals starting at any position i.

$$M[i,j] = RMQ(i,i+2^j)$$

M

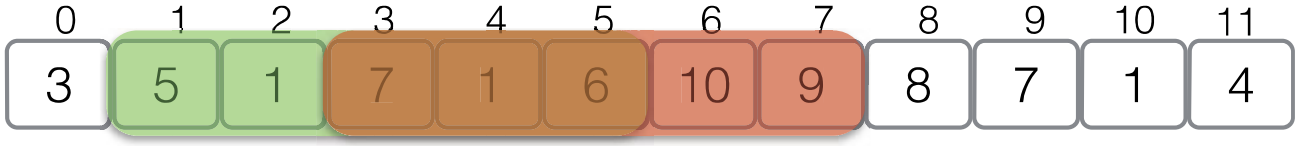


$RMQ(1,7) = \text{argmax}(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$

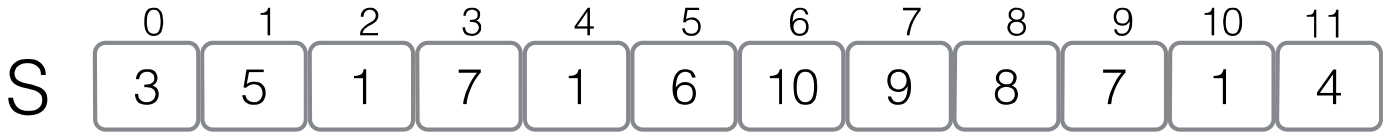
$RMQ(i,j) = \text{argmax}(S[M[i,i+2^{len}]], S[M[j-2^{len},j]])$

where $len = \lfloor \log (j-i+1) \rfloor$

S

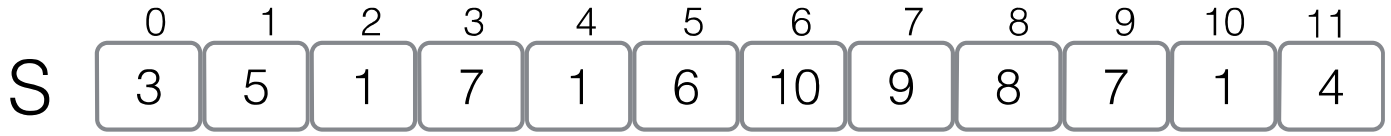


Range Maximum Query (3)



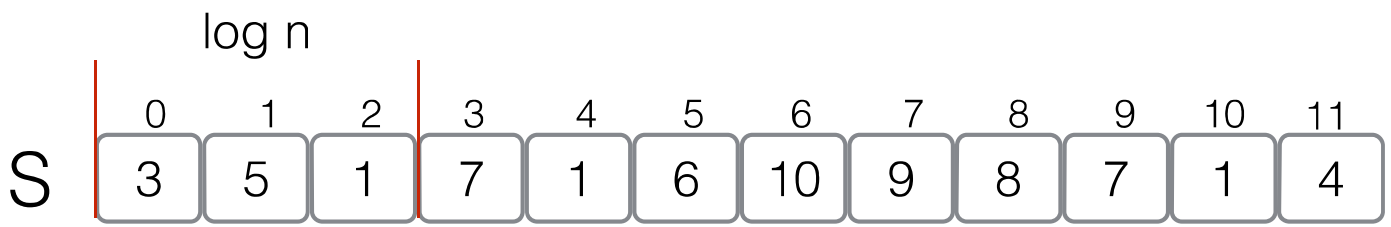
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)



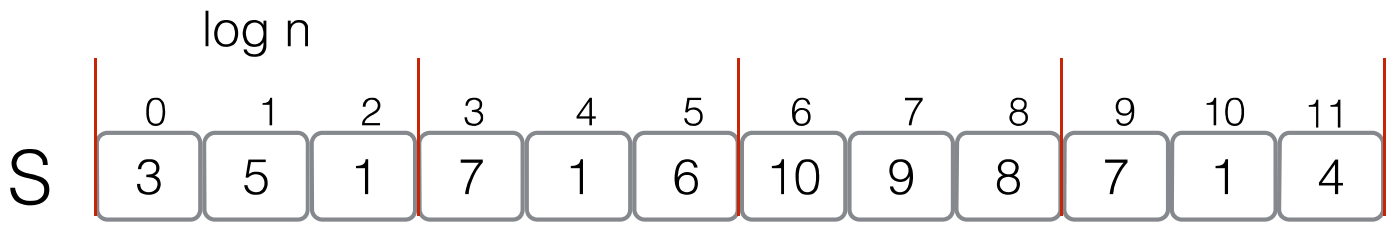
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)



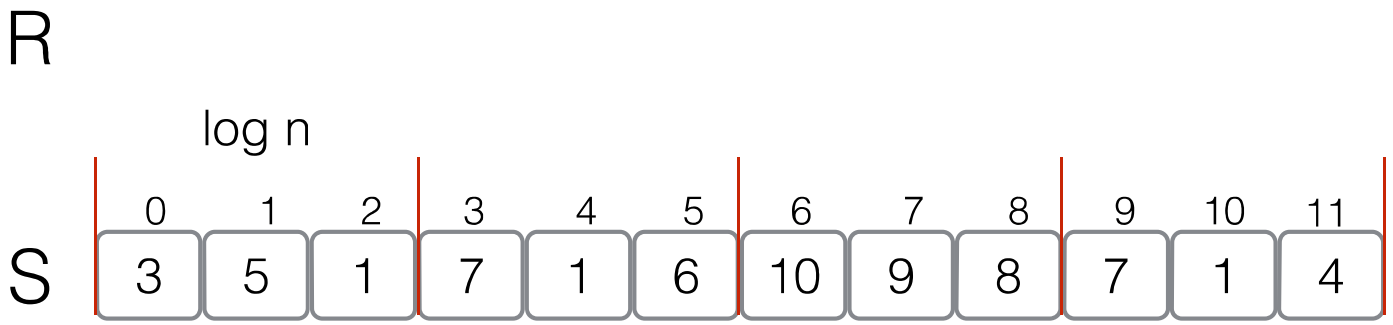
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)



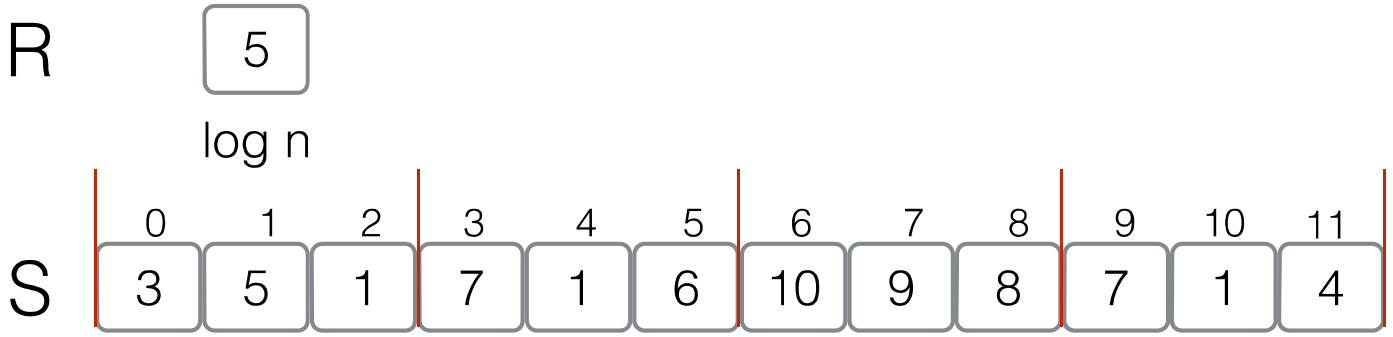
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)



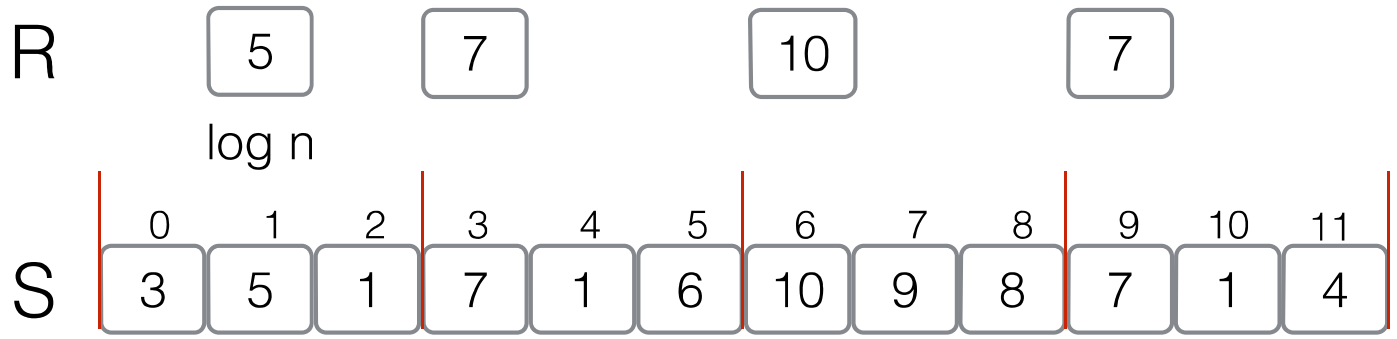
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)



Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

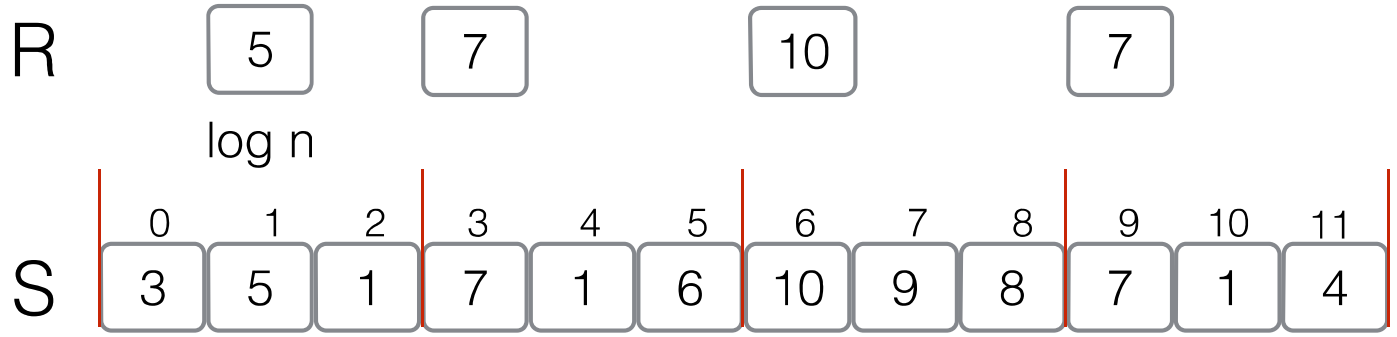


Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: ? bits
Query time: O(1)

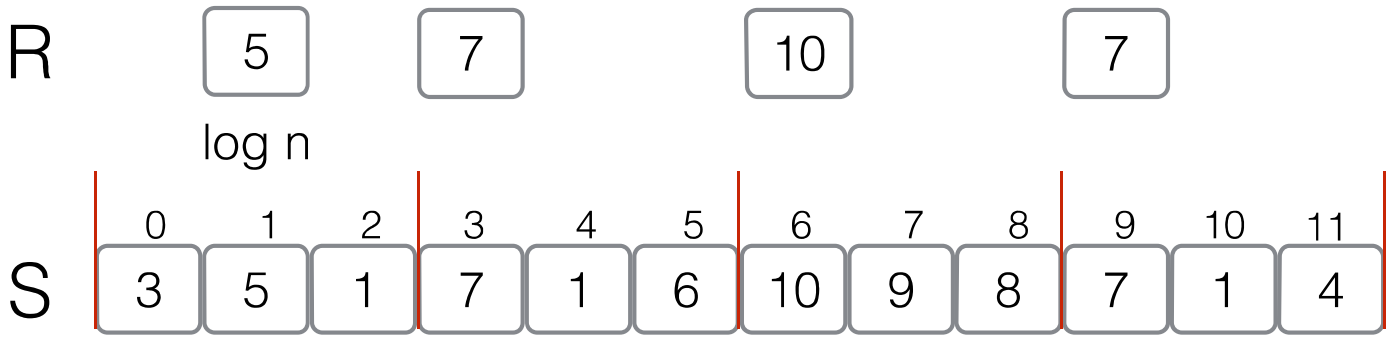


Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)



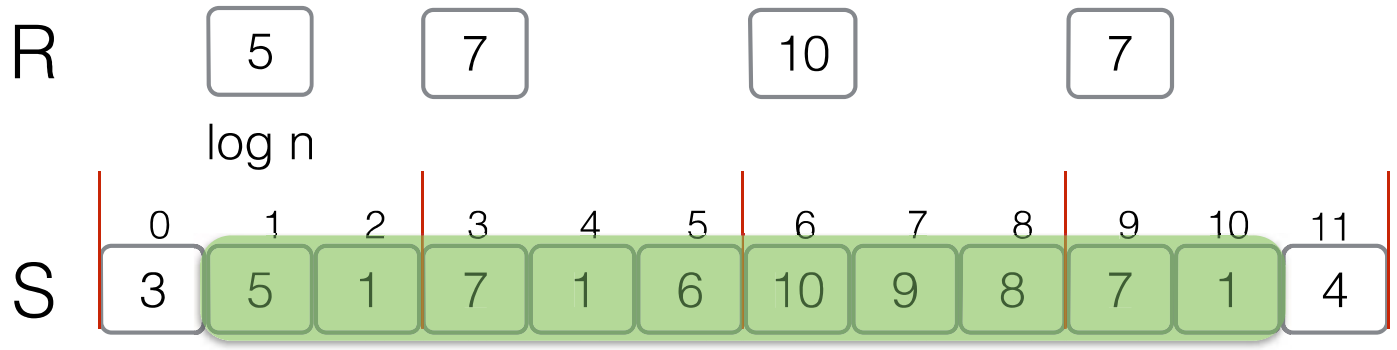
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



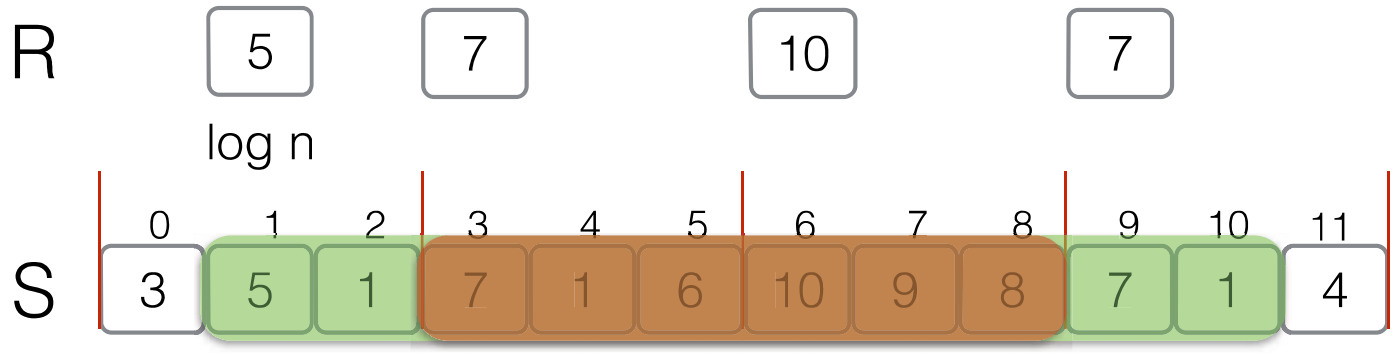
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



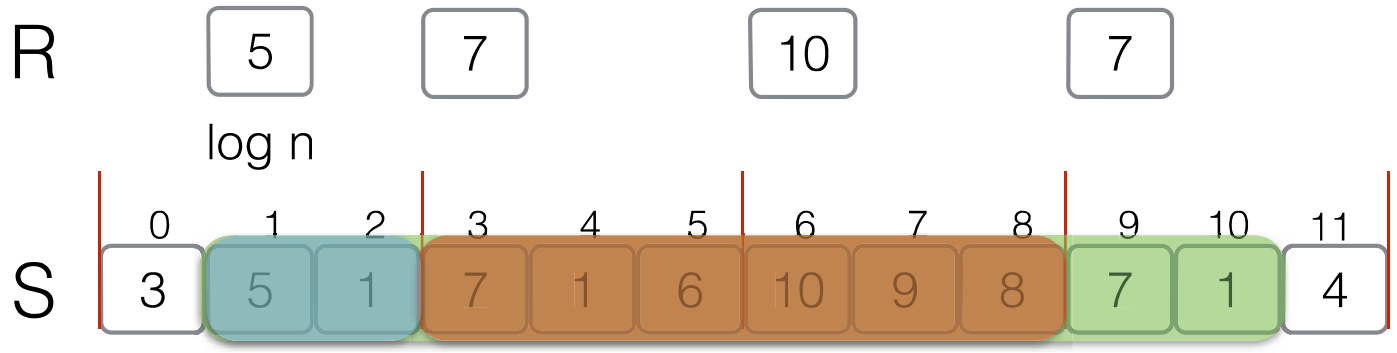
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



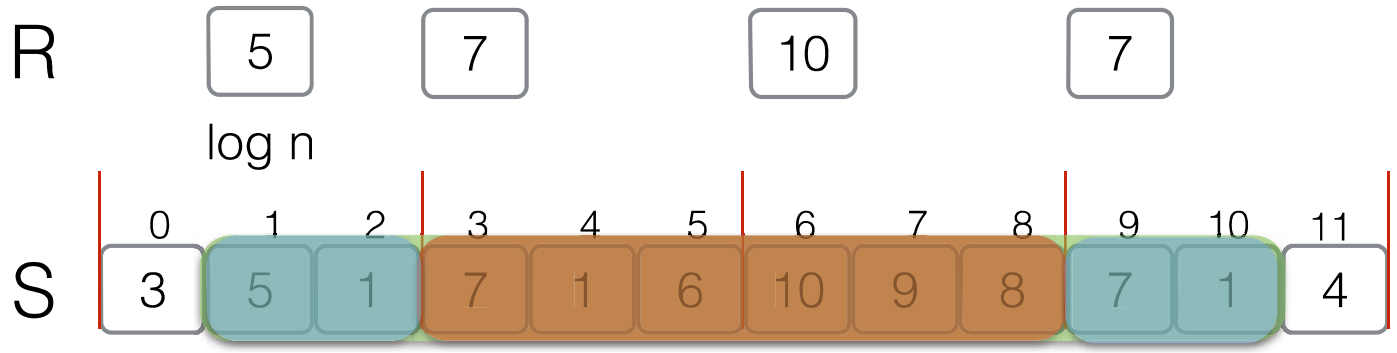
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



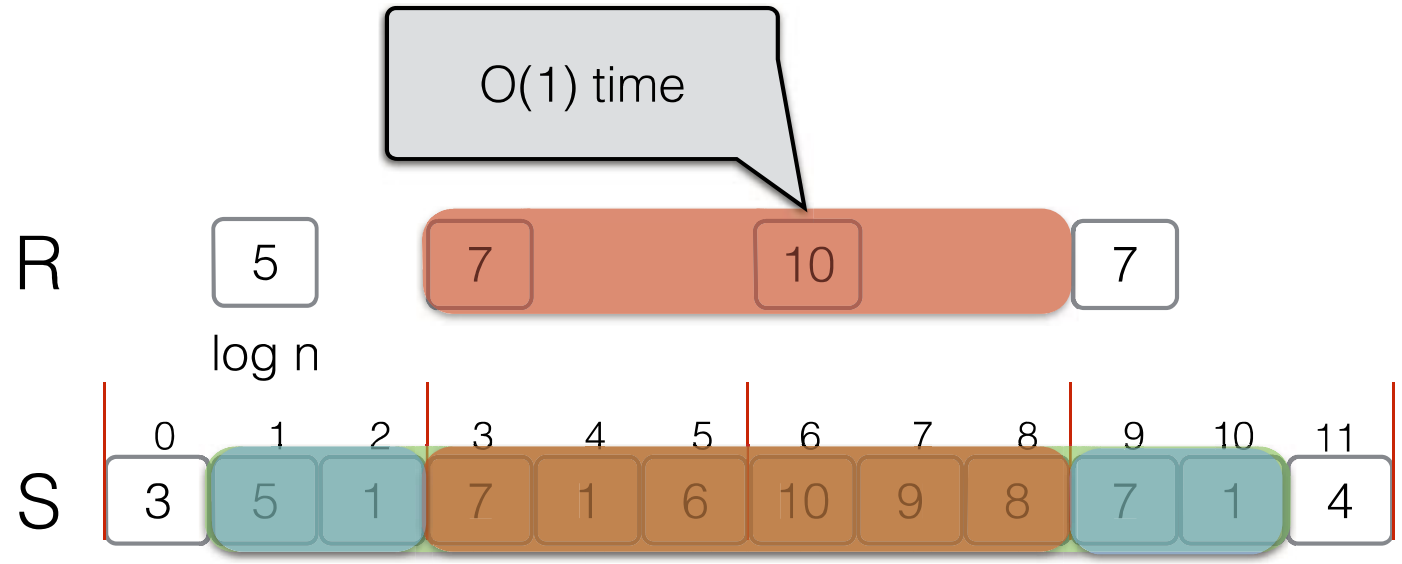
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



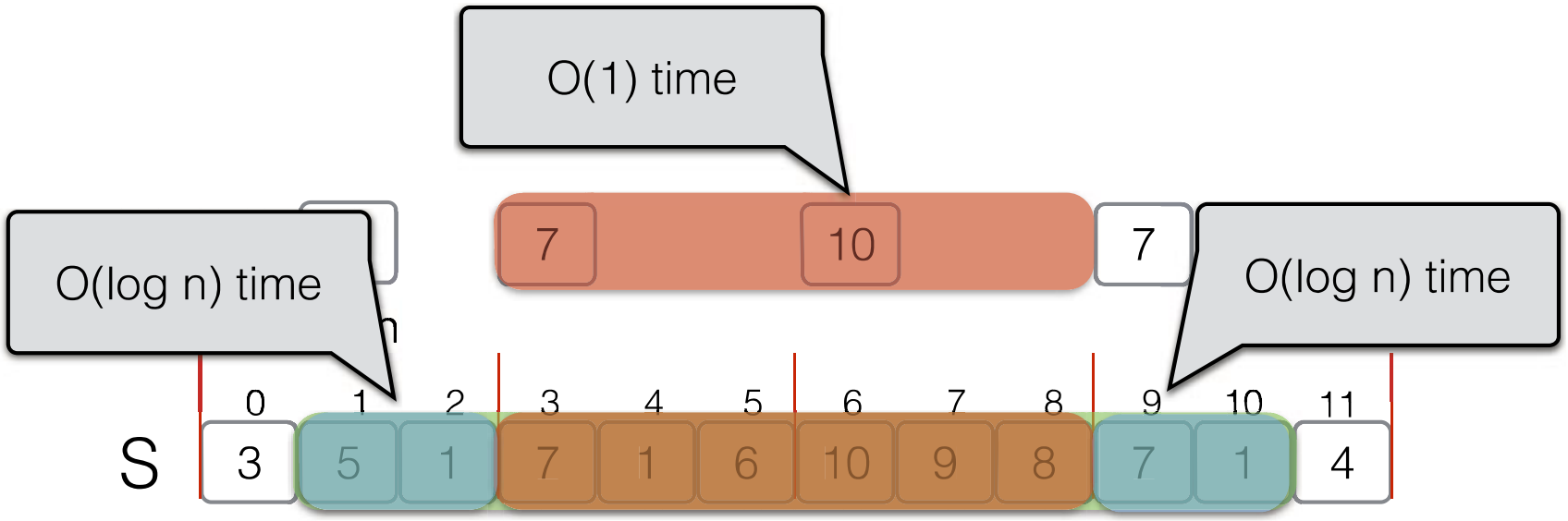
Range Maximum Query (3)

Space: O(n log n) bits
Query time: O(log n)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



Range Maximum Query (3)

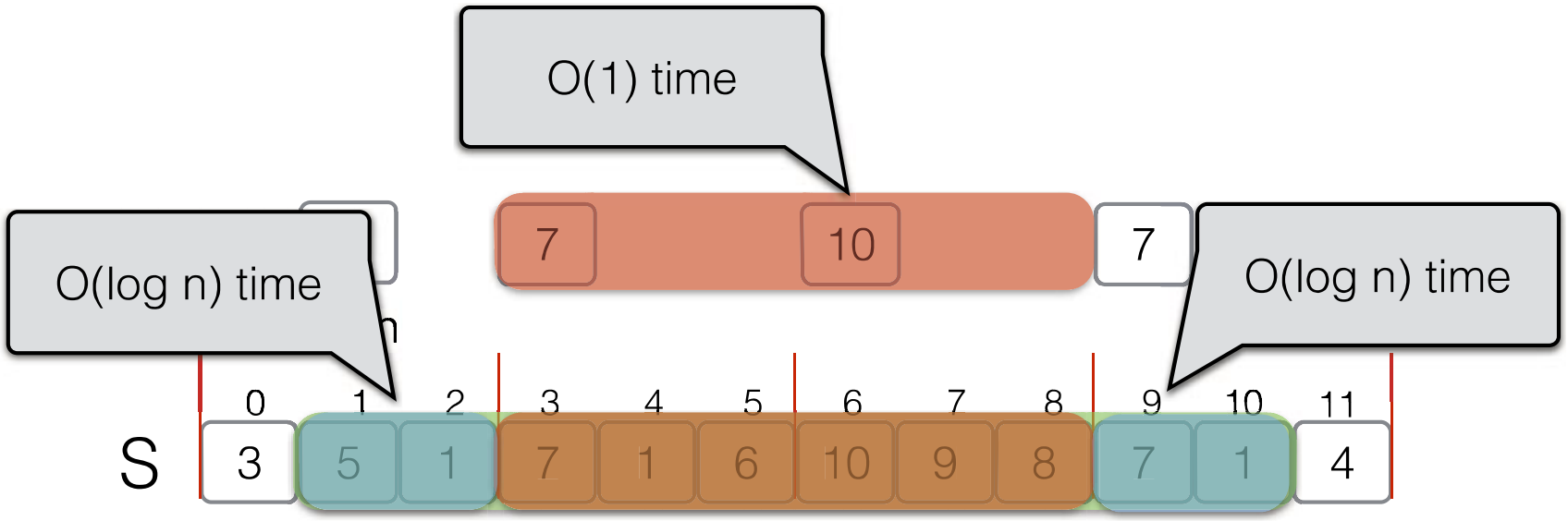
Space: O(n log n) bits
Query time: O(log n)

Space: O(n log n) bits
Query time: O(1)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



Range Maximum Query (3)

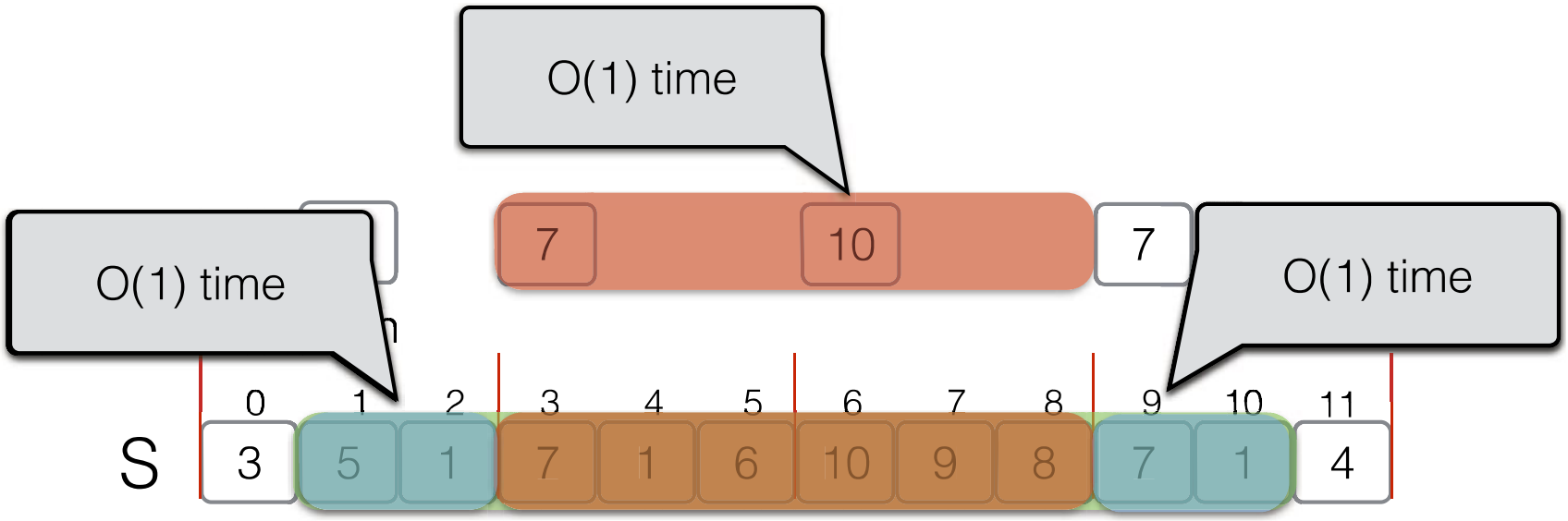
Space: O(n log n) bits
Query time: O(log n)

Space: O(n log n) bits
Query time: O(1)

Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



Range Maximum Query (3)

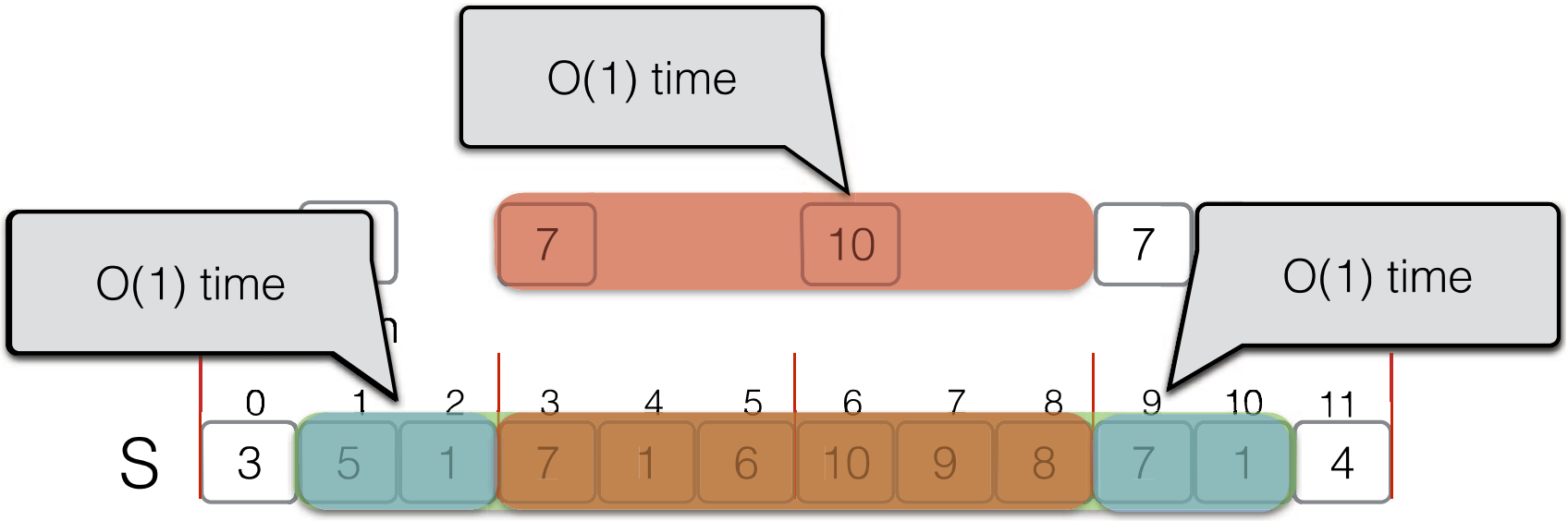
Space: O(n log n) bits
Query time: O(log n)

Space: O(n log n) bits
Query time: O(1)

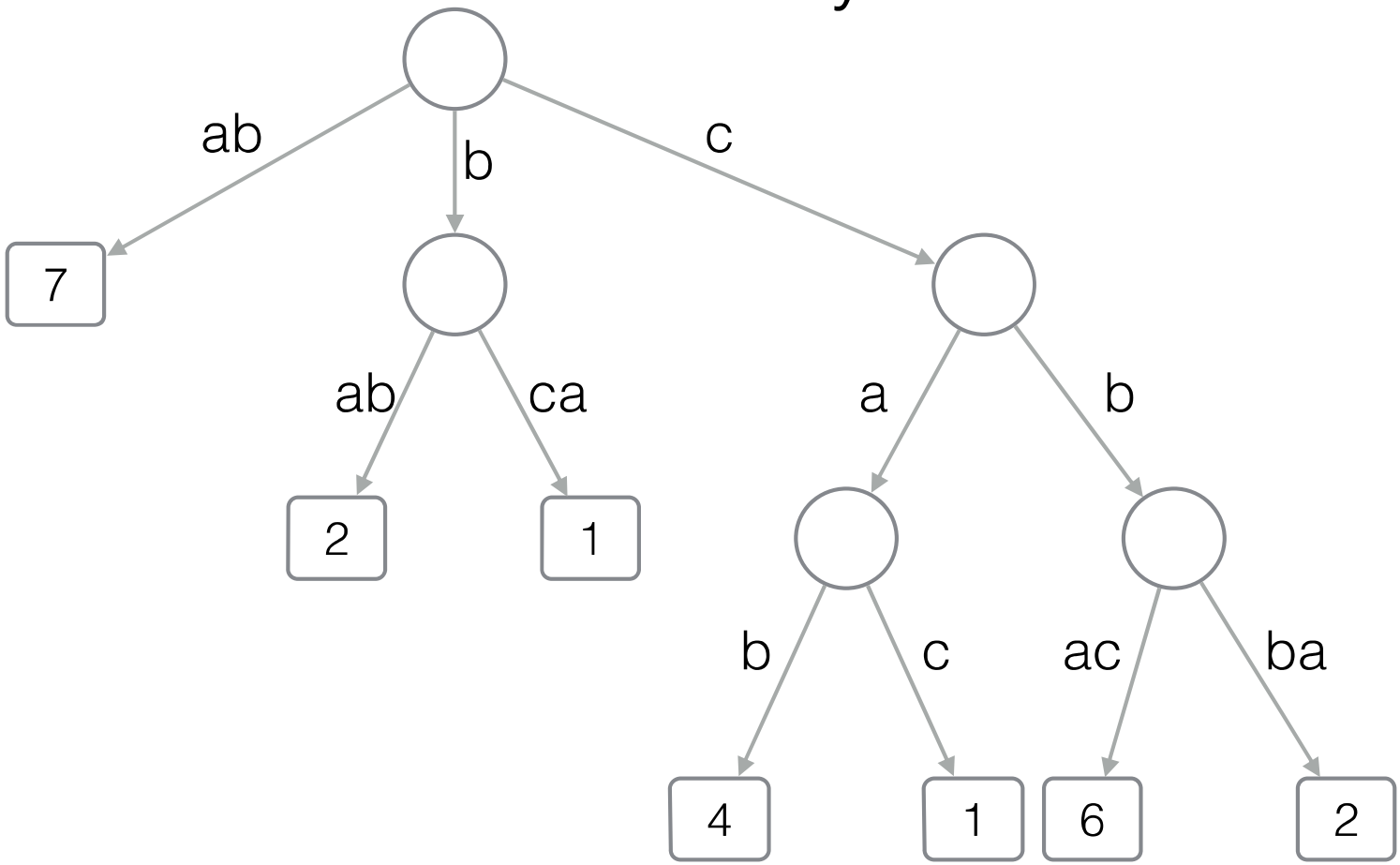
Use the previous solution on R!

Space: O(n log n) bits
Query time: O(1)

RMQ(1,10) = ?



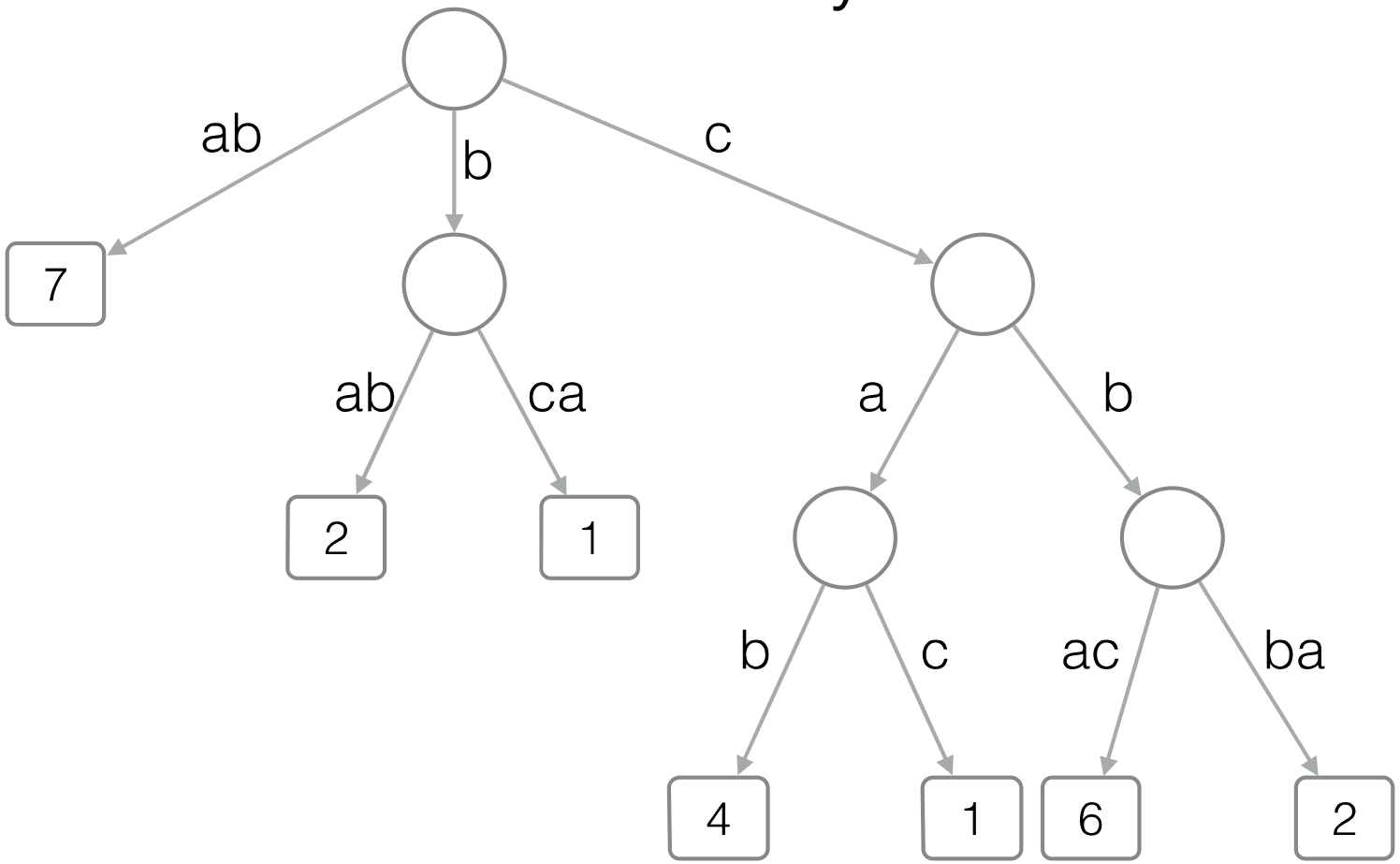
Summary



D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Summary

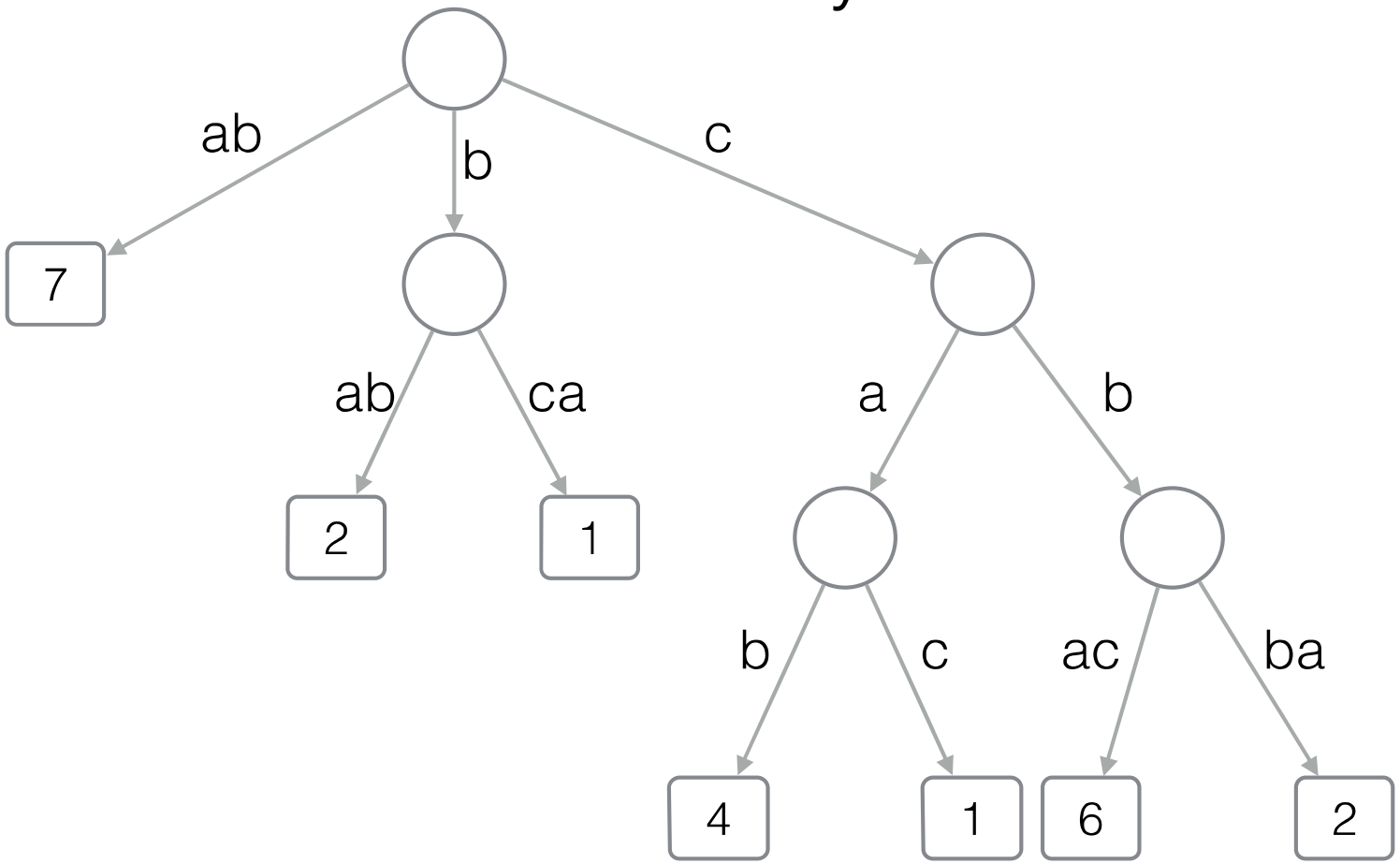


Find the node “prefixed” by P

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Summary



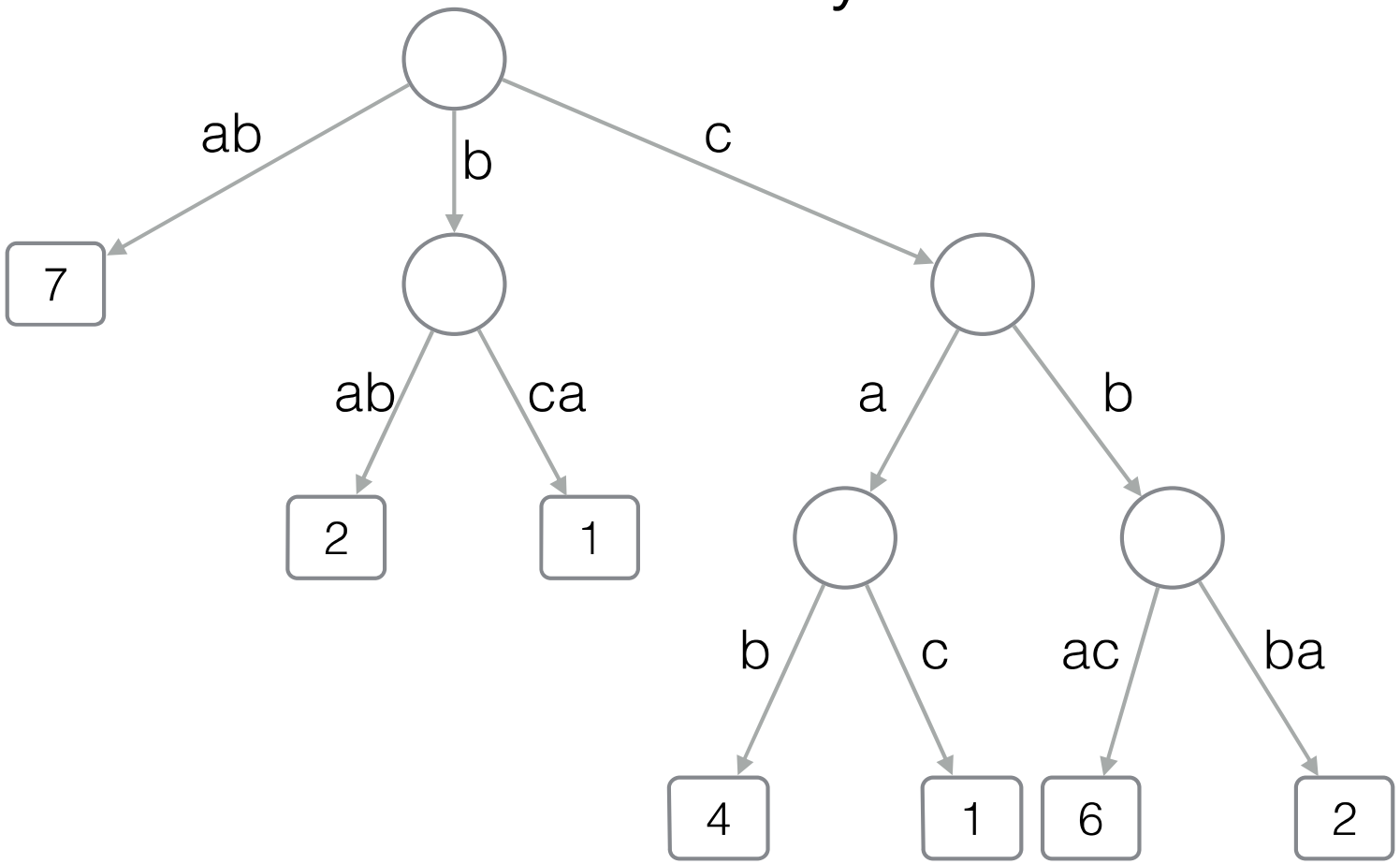
Find the node “prefixed” by P

O(|P|) time

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Summary

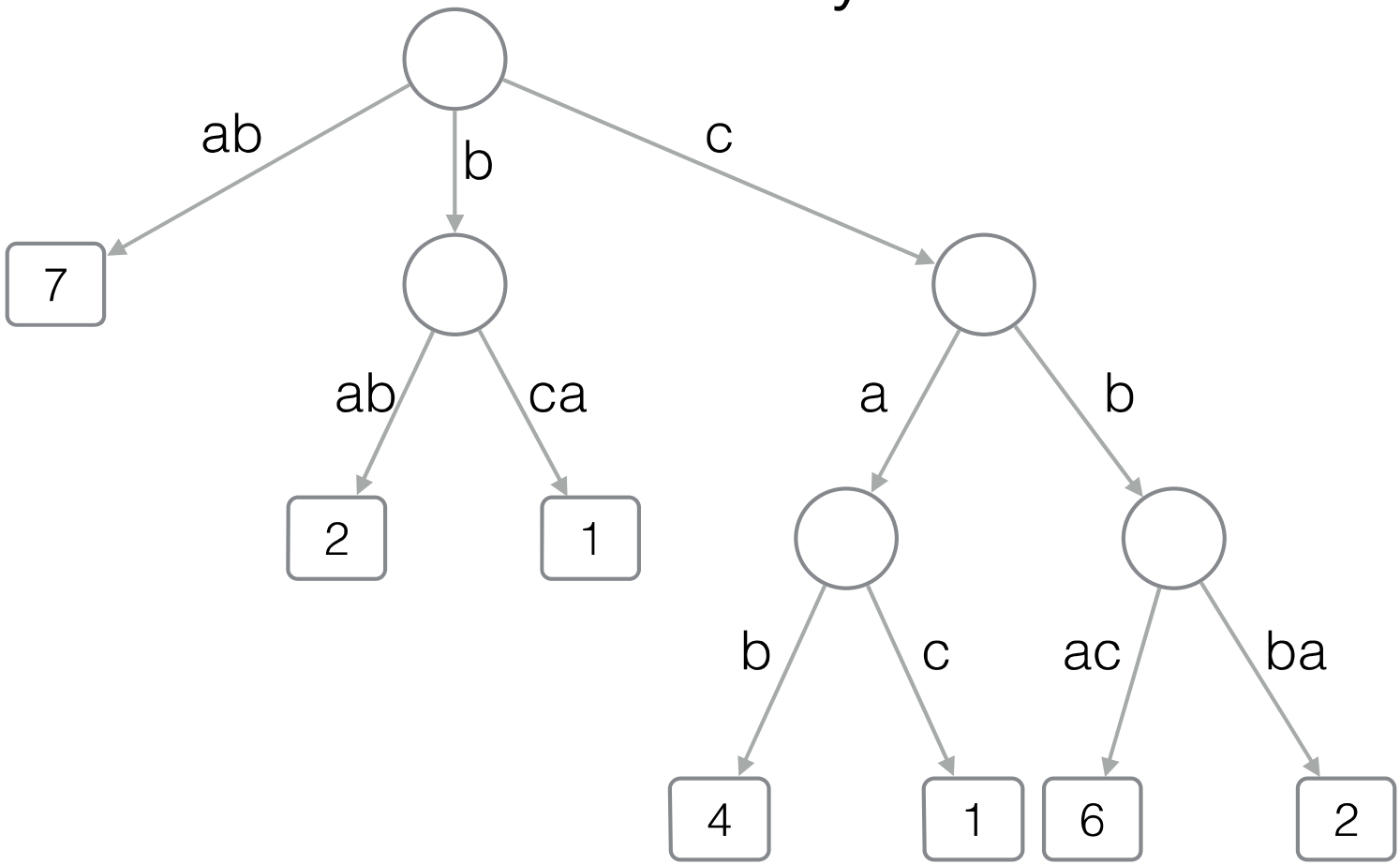


Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits

D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Summary



Find the node “prefixed” by P

O(|P|) time

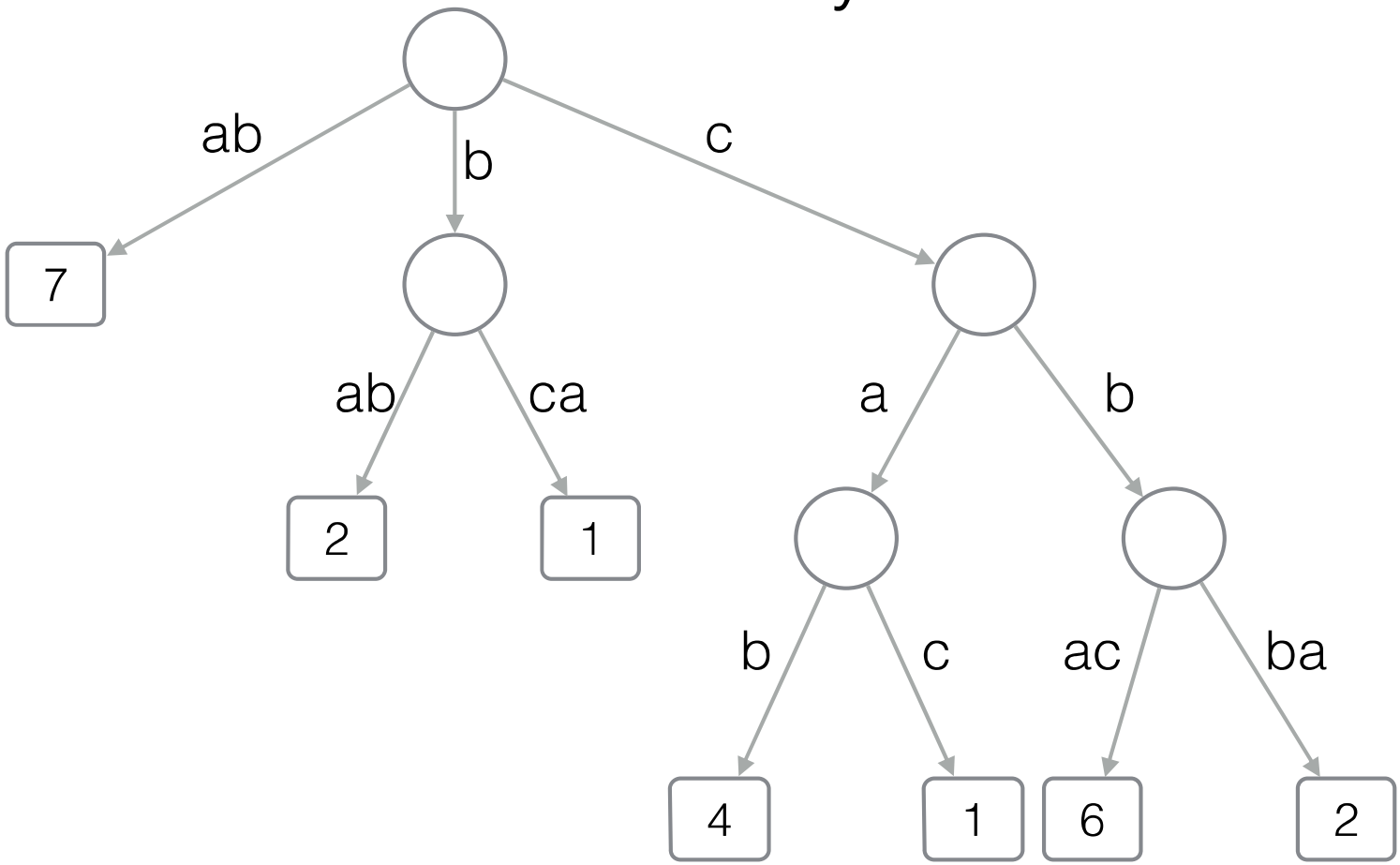
O(m log σ + n log m) bits

Compute the top-k strings

a (1), cab (4), cac (1), cbac (6), cbba (2) }

n = |D|, m total length of strings in D

Summary

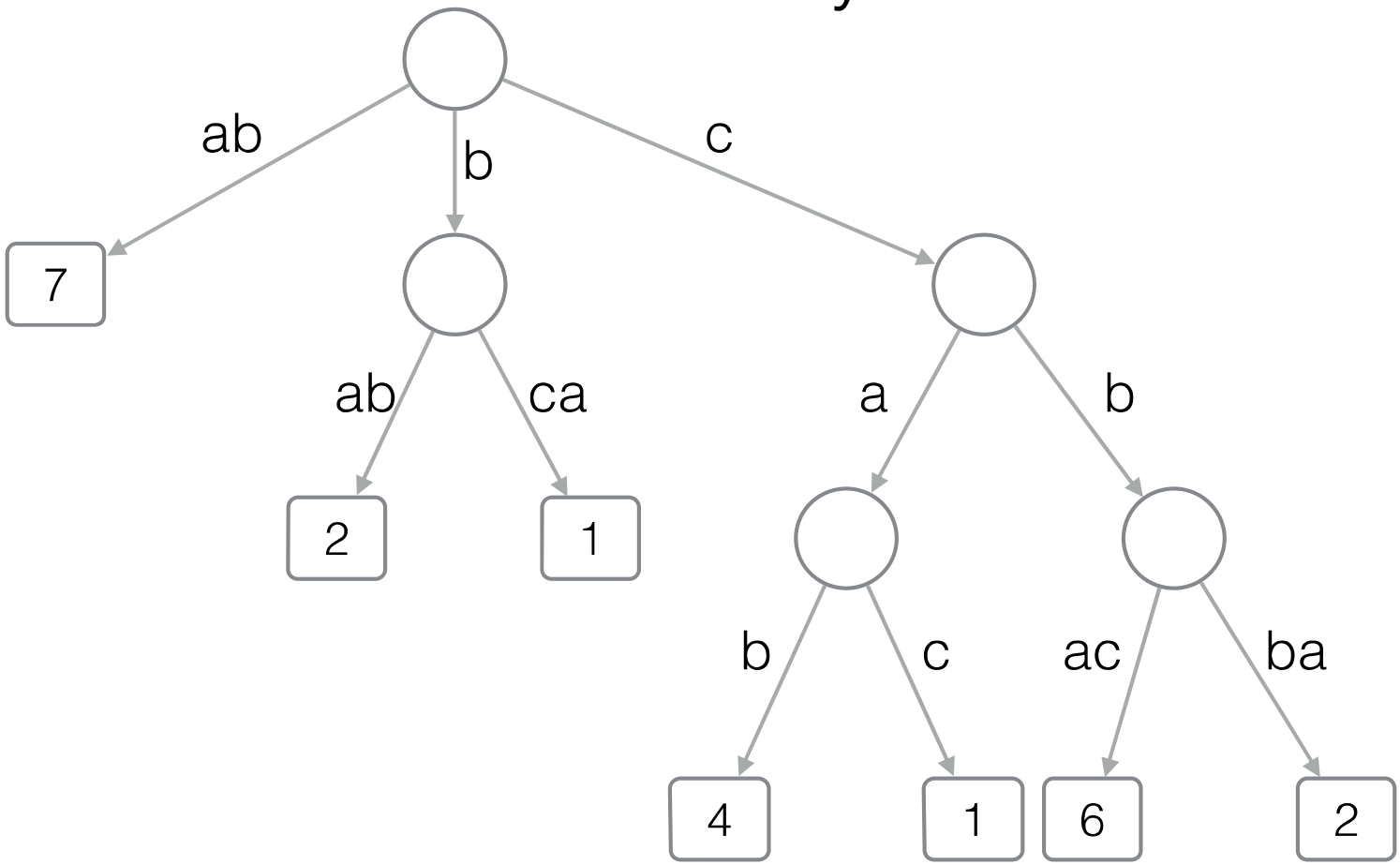


Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	

cbac (6), cbba (2) }

n = |D|, m total length of strings in D

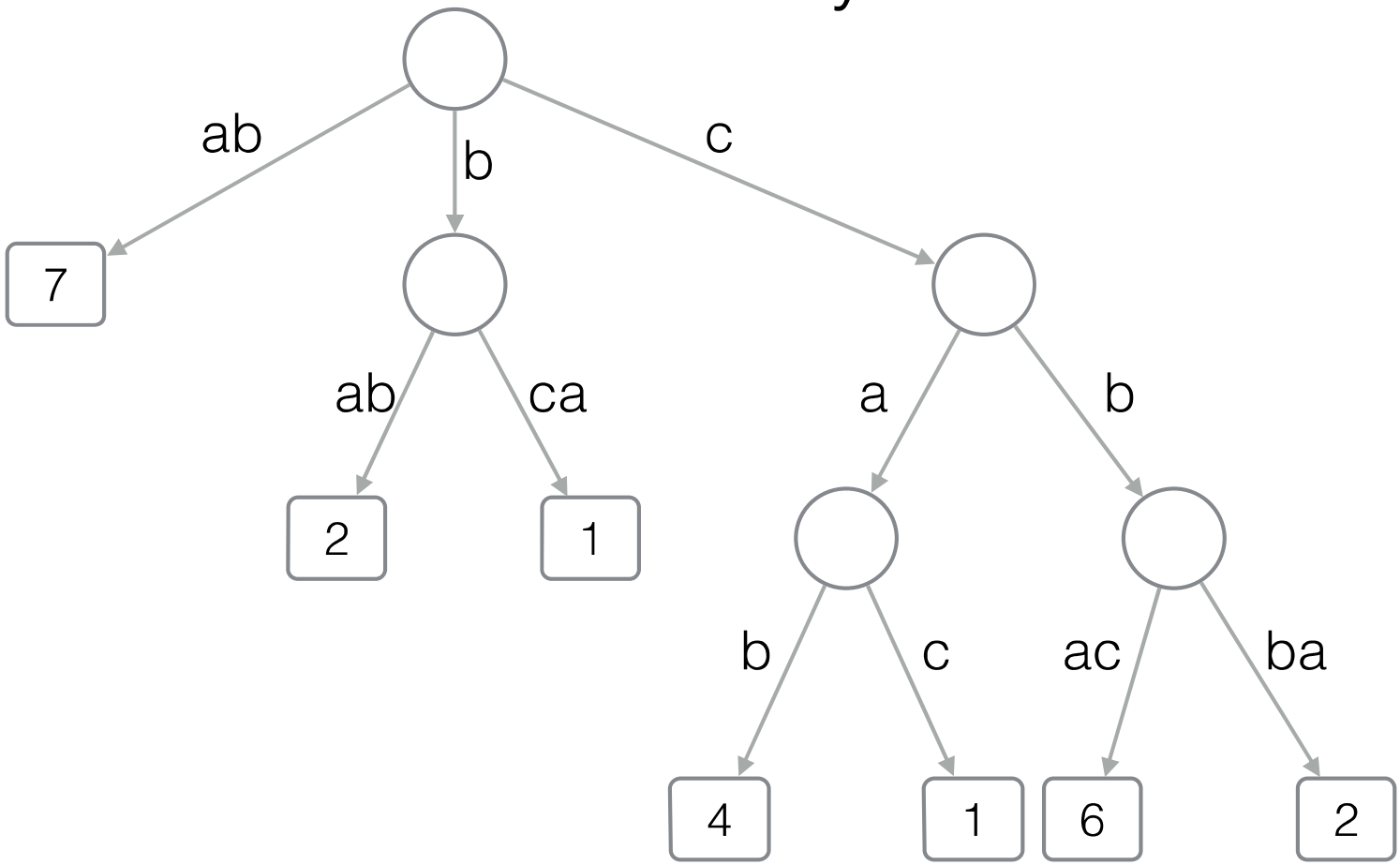
Summary



Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	O(n) bits

n = |D|, m total length of strings in D

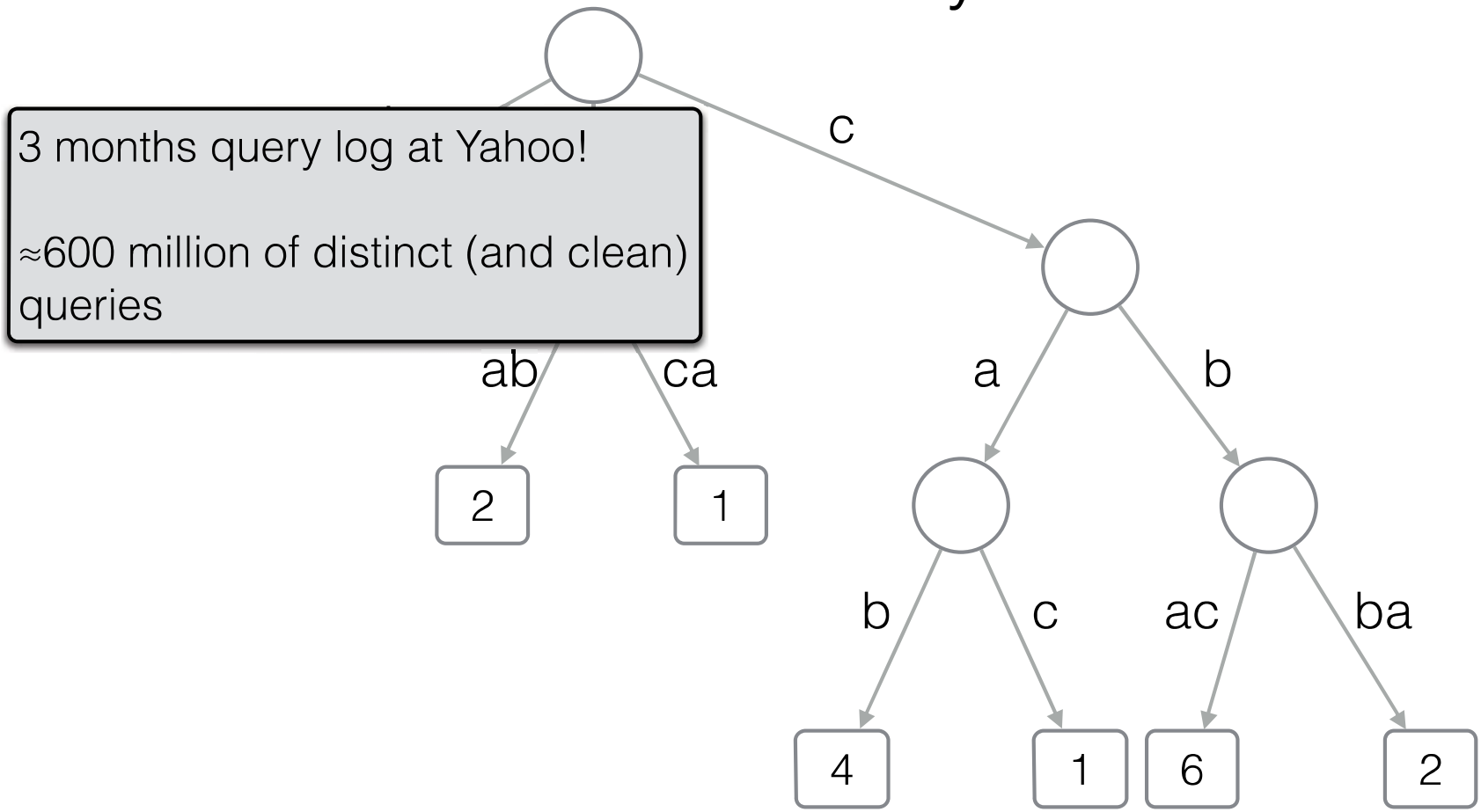
Summary



Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	O(n) bits

n = |D|, m total length of strings in D

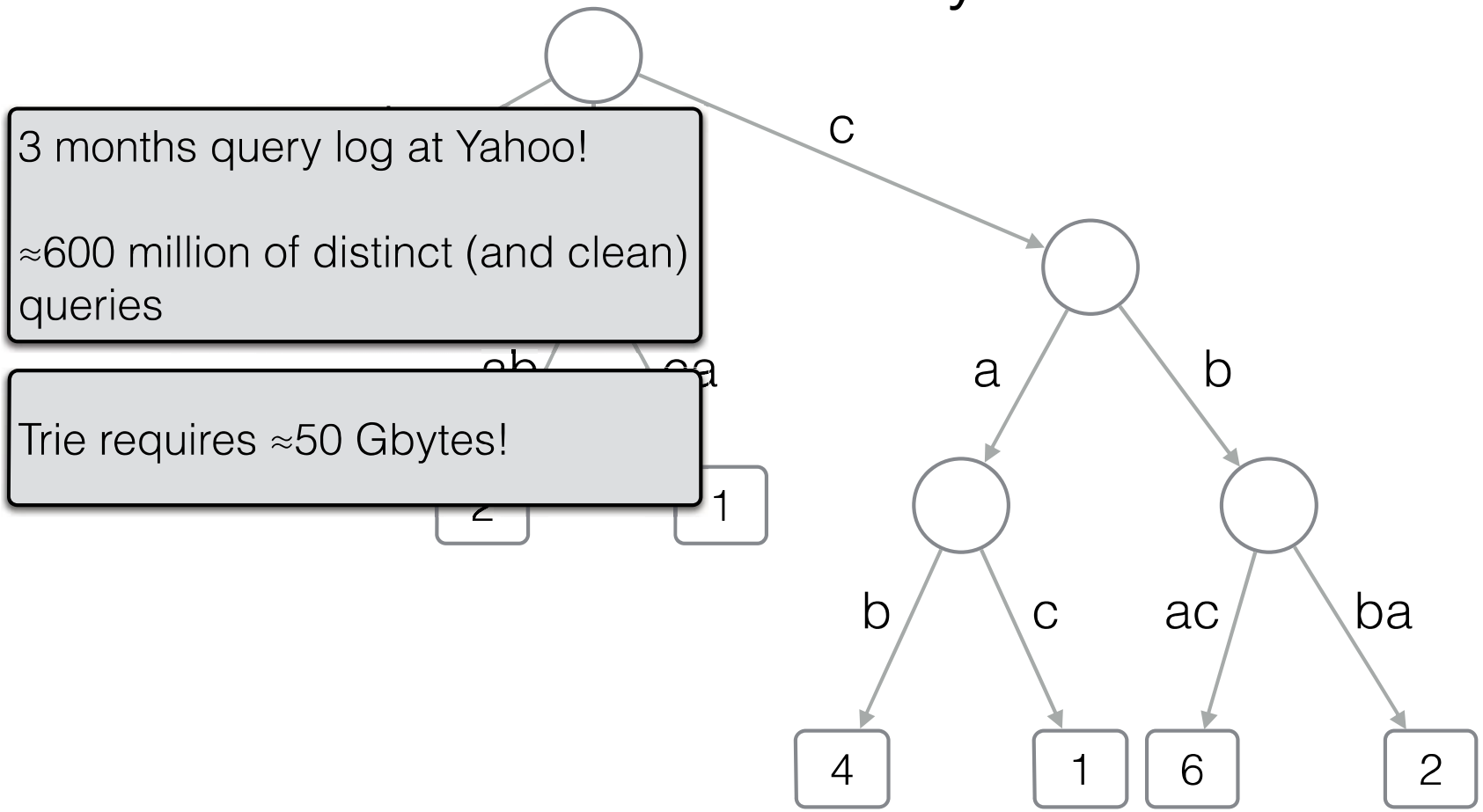
Summary



Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	O(n) bits

n = |D|, m total length of strings in D

Summary



3 months query log at Yahoo!

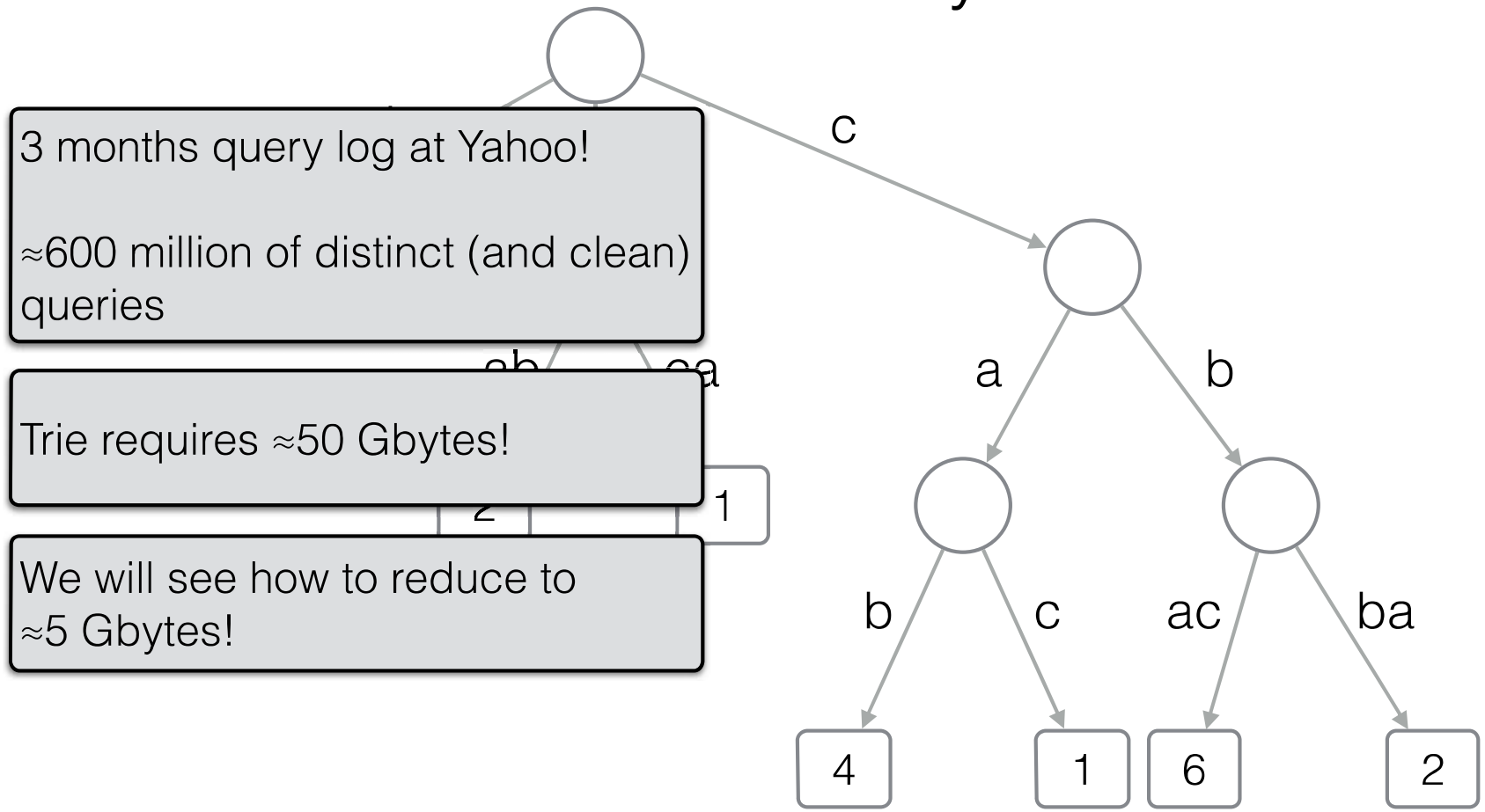
≈600 million of distinct (and clean) queries

Trie requires ≈50 Gbytes!

Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	O(n) bits

n = |D|, m total length of strings in D

Summary



3 months query log at Yahoo!

≈600 million of distinct (and clean) queries

Trie requires ≈50 Gbytes!

We will see how to reduce to ≈5 Gbytes!

Find the node “prefixed” by P	O(|P|) time	O(m log σ + n log m) bits
Compute the top-k strings	O(k log k) time	O(n) bits

n = |D|, m total length of strings in D