Succinct Data Structures

Auto-completion as our target application

Rossano Venturini
rossano@di.unipi.it

With some of my changes
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 560000 vehicles online. Check Car news, reviews and obtain...
Used cars, Vans, Bikos, 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.
Google Search

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.
Dataset? All the past queries
Dataset?
Searches?

All the past queries
Dataset?
Searches?

All the past queries
Prefix search
Dataset?  All the past queries
Searches?  Prefix search
Data structure?
Dataset? All the past queries
Searches? Prefix search
Data structure? Trie
Dataset?  All the past queries
Searches? Prefix search
Data structure? Trie
How to find top-k efficiently?
Trie
D = { ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2) }
D = \{ \text{ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2)} \}

n = |D|, m \text{ total length of strings in } D
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

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

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

O(n) nodes
O(n \log n + m \log \sigma) 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

\[ P = c \]

- \( \mathcal{O}(n) \) nodes
- \( \mathcal{O}(n \log n + m \log \sigma) \) bits of space

Find all the strings prefixed by any pattern \( P \) in \( \mathcal{O}(|P|) \) time

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

\[ n = |D|, \ m \text{ total length of strings in } D \]
Finding Top-1

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

\[ n = |D|, m \text{ total length of strings in } D \]
Finding Top-1

\[ P = c \]

How to find Top-1?

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

\[ n = |D|, \text{ m total length of strings in } D \]
Finding Top-1

How to find Top-1?

\[ P = c \]

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

\[ n = |D|, \text{ m total length of strings in } D \]
Finding Top-1

$P = c$

How to find Top-1?

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

$n = |D|$, $m$ total length of strings in $D$
Finding Top-1

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

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

Scan to find the maximum!

O(n) query time :-(

\[ \text{ab} \rightarrow \text{b} \rightarrow \text{c} \]

\[ \text{ab} \rightarrow \text{ca} \]

\[ \text{a} \rightarrow \text{b} \rightarrow \text{c} \rightarrow \text{ac} \rightarrow \text{ba} \]

\[ \text{2} \rightarrow \text{1} \]
Finding Top-1

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

How to find Top-1?

$$P = c$$

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

$$n = |D|, m \text{ total length of strings in } D$$
Finding Top-1

P = c

How to find Top-1?

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

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

\[ n = |D|, \ m \text{ total length of strings in } D \]
Finding Top-1

How to find Top-1?

P = c

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

How to find Top-1?

P = c

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

How to find Top-1?

P = c

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

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

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

\[ n = |D|, \ m \text{ total length of strings in } D \]
Finding Top-1

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

Augment each node with the max (and string id) within its subtree!
Finding Top-1

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 = \{ 7 (1), ab (1), a (1), b (1), c (1), cab (4), cac (1), cbac (6), cbba (2) \}

P = c
Finding 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: $O(1)$

$P = c$
How to find Top-1?

Solving Top-$k$?
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: $O(k^3 \cdot n \cdot \log n)$ bits
- You must know $k$ at building time!
Finding Top-1

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

$n = |D|$, $m$ 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

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

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

\[ n = |D|, \text{ m total length of strings in } D \]
Finding Top-1

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 $S[i,j]$

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

$n = |D|$, $m$ 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

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

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

\[ n = |D|, m \text{ total length of strings in } D \]
Finding 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 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

\[ \text{RMQ}(i,j) = \text{position of the range } S[i,j] \]

Can you solve Top-2?

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

\[ n = |D|, m \text{ total length of strings in } D \]
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

\[ \text{RMQ}(i,j) = \text{position of the range } S[i,j] \]

Can you solve Top-2?

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

\[ n = |D|, m \text{ total length of strings in } D \]
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: $O(1)$

$M[i,j] = \text{RMQ}(i,j)$

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: $O(1)$

$M[i,j] = \text{RMQ}(i,j)$

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!
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

<table>
<thead>
<tr>
<th>S</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
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 $i$. 

<table>
<thead>
<tr>
<th>S</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
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 $i$.

\[
M[i,j] = \text{RMQ}(i,i+2^j)
\]
Range Maximum Query (2)

Space: $\Omega(n \log^2 n)$ 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 $i$.

\[
M[i,j] = \text{RMQ}(i, i+2^j)
\]
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 $i$.

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

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

$9 = 1 + 2^3$
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 $i$. 

$M[i,j] = \text{RMQ}(i, i+2^j)$
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 $i$.

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

![Matrix M with values 0 to 11]

$S$:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
Range Maximum Query (2)

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

Maximum of an 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) =

<table>
<thead>
<tr>
<th>M</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>
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) = $\arg\max(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$
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.

$RMQ(1,7) = \arg\max(S[M[1,1+2^2]], S[M[7-2^2,7]]) = 6$
Range Maximum Query (2)

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

Maximum of an 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 $i$.

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

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

Maximum of an 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 $i$.

$$RMQ(1, 7) = \arg\max(S[M[1,1+2^2]], S[M[7-2^2, 7]]) = 6$$
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 $i$.

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

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

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

\[\begin{array}{cccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
3 & 5 & 1 & 7 & 1 & 6 & 10 & 9 & 8 & 7 & 1 & 4 \\
\end{array}\]
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: $O(\log n)$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>
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: $O(1)$

S

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

R

<table>
<thead>
<tr>
<th>log n</th>
<th>5</th>
<th>7</th>
<th>10</th>
<th>7</th>
</tr>
</thead>
</table>
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) = \_?

$S$

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

$3 \ 5 \ 1 \ 7 \ 1 \ 6 \ 10 \ 9 \ 8 \ 7 \ 1 \ 4$

$R$

$log n$

$5 \ 7 \ 10 \ 7$
Range Maximum Query (3)

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

RMQ(1,10) = ?

Use the previous solution on R!

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

R

\begin{align*}
    &5 & 7 & 10 & 7 \\
\text{log n}
\end{align*}

S

\begin{align*}
    &3 & 5 & 1 & 7 & 1 & 6 & 10 & 9 & 8 & 7 & 1 & 4 \\
\end{align*}
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: $O(1)$

$RMQ(1, 10) = ?$

R

5 7 10 7

log n

S

0 1 2 3 4 5 6 7 8 9 10 11
3 5 7 1 6 10 9 8 7 1 4
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: $O(1)$

$RMQ(1, 10) = ?$

<table>
<thead>
<tr>
<th>S</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>S</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>R</td>
<td>5</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$log n$
Range Maximum Query (3)

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

$RMQ(1, 10) = ?$

Use the previous solution on $R$!

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

$O(1)$ time

$S$

$R$

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

$3$ $5$ $1$ $7$ $1$ $6$ $10$ $9$ $8$ $7$ $1$ $4$
Range Maximum Query (3)

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

$RMQ(1,10) = ?$

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: $O(\log n)$

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

$\text{RMQ}(1, 10) = ?$

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: $O(\log n)$

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: $O(1)$

RMQ(1,10) = ?

Use the previous solution on R!

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

O(1) time

O(1) time

S

0 1 2 3 4 5 6 7 8 9 10 11
\[ D = \{ \text{ab (7), bab (2), bca (1), cab (4), cac (1), cbac (6), cbba (2)} \} \]

\[ n = |D|, \ m \text{ total length of strings in } D \]
Find the node “prefixed” by \( P \)

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

\[ n = |D|, \text{ m total length of strings in } D \]
Summary

Find the node “prefixed” by $P$ in $O(|P|)$ time

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

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

\[ \text{Find the node “prefixed” by } P \quad \text{O(|P|) time} \quad \text{O(m log } \sigma + n \log m \text{) bits} \]

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

\[ n = |D|, \text{ m total length of strings in } D \]
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

Compute the top-$k$ strings : $O(k \log k)$ time

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

cbac (6), cbba (2)
Summary

Find the node “prefixed” by $P$

$O(|P|)$ time

$O(m \log \sigma + n \log m)$ 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 \sigma + 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

Find the node “prefixed” by P: O(|P|) time
Compute the top-k strings: O(k log k) time

O(m log σ + n log m) bits
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
D = \{ ab, bab, bca, cab, cac, cbac, cbba \}

n = |D|, m total length of strings in D
D = { ab, bab, bca, cab, cac, cbac, cbba } 

n = |D|, m total length of strings in D
$D = \{\text{ab, bab, bca, cab, cac, cbac, cbba}\}$

$n = |D|, m \text{ total length of strings in } D$
Patricia trie

\[ D = \{ \text{ab, bab, bca, cab, cac, cbac, cbba} \} \]

\[ n = |D|, m \text{ total length of strings in } D \]
D = \{ ab, bab, bca, cab, cac, cbac, cbba \}

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

P = cba
$D = \{ \text{ab, bab, bca, cab, cac, cbac, cbba} \}$

$n = |D|$, $m$ total length of strings in $D$
\[ D = \{ \text{ab, bab, bca, cab, cac, cbac, cbba} \} \]

\[ n = |D|, \ m \text{ total length of strings in } D \]
Patricia trie

D = \{ ab, bab, bca, cab, cac, cbac, cbba \}

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

$D = \{ \text{ab, bab, bca, cab, cac, cbac, cbba} \}$

$n = |D|$, $m$ total length of strings in $D$
D = \{ ab, bab, bca, cab, cac, cbac, cbba \}

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

Blind search.
We can skip symbols and check at the end.

O(|P|) time

O(n \log m + m \log \sigma) bits

n = |D|, m total length of strings in D
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: \( O(n \log n) \) bits
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits
Succinct representation of trees (1)
[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

0 0 0 2 2 2 2 0 0 0 0
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

D 3
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order:

**D** 3 0 2 2
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

D 3 0 2 2 0 0 2 2
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

$D = 3 \ 0 \ 2 \ 2 \ 0 \ 0 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0 \ 0$
Succinct representation of trees (1)
[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order
\[D = 3\ 0\ 2\ 2\ 0\ 0\ 2\ 2\ 0\ 0\ 0\ 0\ 0\]

A tree is uniquely determined by the degree sequence
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

$D: 3 \ 0 \ 2 \ 2 \ 0 \ 0 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0$

A tree is uniquely determined by the degree sequence

How reconstruct the tree?
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

It still requires $O(n \log n)$ bits :-(

Write the degree sequence in level order:

D: 3 0 2 2 0 0 2 2 0 0 0 0 0
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

It still requires $O(n \log n)$ bits :-(

Write the degree sequence in level order

$D \ 3 \ 0 \ 2 \ 2 \ 0 \ 0 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0 \ 0$

Solution: write them in unary
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: \(O(n \log n)\) bits

Best: \(2n\) bits

Write the degree sequence in level order:

\[ D \: 3 \: 0 \: 2 \: 2 \: 0 \: 0 \: 2 \: 2 \: 0 \: 0 \: 0 \: 0 \: 0 \: 0 \]

Solution: write them in unary

B
Succinct representation of trees (1)

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

Write the degree sequence in level order

D: 3 0 2 2 0 0 2 2 0 0 0 0

Solution: write them in unary

B: 1110

It still requires $O(n \log n)$ bits :-(
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

It still requires $O(n \log n)$ bits :-(

Write the degree sequence in level order

D 3 0 2 2 0 0 2 2 0 0 0 0 0

Solution: write them in unary

B 1110 0 110 110 0 0 110 110 0 0 0 0
Succinct representation of trees (1)

Trivial: $O(n \log n)$ bits

Best: $2n$ bits

It still requires $O(n \log n)$ bits :-(

Write the degree sequence in level order

D  3 0 2 2 0 0 2 2 0 0 0 0 0

Solution: write them in unary

B  1110 0 110 110 0 0 110 110 0 0 0 0

B takes $2n - 1$ bits! For each node we have a 0 and a 1 (but the root)
Succinct representation of trees (1)

Trivial: \( O(n \log n) \) bits

Best: \( 2n \) bits

It still requires \( O(n \log n) \) bits :-(

Can we navigate the tree?

B takes \( 2n - 1 \) bits!
For each node we have a 0 and a 1 (but the root)
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \]

```
B  1 0 1 1 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0
  1 2 3 4 5 6 7 8 9 10 11 12
```

\[ \text{pos}(5) \]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
\[ y = \text{Select}_0(x) + 1 \]

// start of x's children in B
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
\[ y = \text{Select}_0(x) + 1 \]

// start of x's children in B

firstChild(3)
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[
p_{\text{pos}}(x) = \text{Select}_1(x) \\
\text{firstChild}(x) = ? \\
y = \text{Select}_0(x) + 1
\]

// start of x's children in B

firstChild(3)

\[
y = \text{Select}_0(3) + 1 = 8
\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[\text{pos}(x) = \text{Select}_1(x)\]
\[\text{firstChild}(x) = ?\]
\[y = \text{Select}_0(x) + 1\]

// start of x's children in B
if B[y] == 0
    return -1  // is a leaf

firstChild(3)

\[y = \text{Select}_0(3) + 1 = 8\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]

\[ \text{firstChild}(x) = ? \]

\[ y = \text{Select}_0(x) + 1 \]

// start of x's children in B

if \( B[y] \) == 0

\[
\begin{align*}
\text{return } -1 & \quad // \text{is a leaf} \\
\text{else} & \\
\text{return } y-x & \quad // \text{Rank}_1(y)
\end{align*}
\]

firstChild(3)

\[ y = \text{Select}_0(3) + 1 = 8 \]

\[
\begin{array}{cccccccccccc}
B & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

pos(x) = Select_1(x)
firstChild(x) = ?
y = Select_0(x)+1
// start of x's children in B
if B[y] == 0
    return -1  // is a leaf
else
    return y-x  // Rank_1(y)

firstChild(3) = 8-3 = 5

y = Select_0(3)+1=8
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
\[ y = \text{Select}_0(x) + 1 \]
\[ // \text{start of } x\text{'s children in } B \]
\[ \text{if } B[y] == 0 \]
\[ \text{return } -1 // \text{is a leaf} \]
\[ \text{else} \]
\[ \text{return } y-x // \text{Rank}_1(y) \]
\[ \text{degree}(x) = ? \]

\[ y = \text{Select}_0(3) + 1 = 8 \]

\[ B \]
\[ 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \]

1 2 3 4 5 6 7 8 9 10 11 12
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[
\text{pos}(x) = \text{Select}_1(x)
\]

\[
\text{firstChild}(x) = \text{?}
\]

\[
y = \text{Select}_0(x) + 1
\]

// start of x's children in B

\[
\text{if } B[y] == 0
\]

\[
\text{return } -1 \quad // \text{is a leaf}
\]

\[
\text{else}
\]

\[
\text{return } y-x \quad // \text{Rank}_1(y)
\]

degree(x) = ?

\[
\text{Select}_0(x+1) - (\text{Select}_0(x) + 1)
\]

\[
y = \text{Select}_0(3) + 1 = 8
\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
\[ y = \text{Select}_0(x) + 1 \]
\[ \text{if } B[y] == 0 \]
\[ \text{return } -1 \quad \text{// is a leaf} \]
\[ \text{else} \]
\[ \text{return } y-x \quad \text{// Rank}_1(y) \]

\[ \text{degree}(x) = ? \]
\[ \text{Select}_0(x+1) - (\text{Select}_0(x) + 1) \]

\[ y = \text{Select}_0(3) + 1 = 8 \]

\[ B = 1011100110110001101100000000 \]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]

\[ \text{firstChild}(x) = ? \]

\[ y = \text{Select}_0(x) + 1 \]

// start of x's children in B

if \( \text{B}[y] == 0 \)

return -1 // is a leaf

else

return \( y-x \) // \( \text{Rank}_1(y) \)

\[ \text{degree}(x) = ? \]

\[ \text{Select}_0(x+1) - (\text{Select}_0(x) + 1) \]

degree(3) = Select_0(4) - (Select_0(3) + 1)

\[ y = \text{Select}_0(3) + 1 = 8 \]

\[ \text{B} = \begin{array}{*{12}c} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array} \]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[
\text{pos}(x) = \text{Select}_1(x) \\
\text{firstChild}(x) = ? \\
\]

\[
y = \text{Select}_0(x) + 1 \\
// \text{start of } x's \text{ children in } B
\]

\[
\text{if } B[y] == 0 \\
\text{return } -1 // \text{is a leaf}
\]

\[
\text{else} \\
\text{return } y - x // \text{Rank}_1(y)
\]

\[
\text{degree}(x) = ? \\
\text{Select}_0(x + 1) - (\text{Select}_0(x) + 1)
\]

degree(3) = \text{Select}_0(4) - (\text{Select}_0(3) + 1)

\[
y = \text{Select}_0(3) + 1 = 8 \quad \text{Select}_0(4) = 10
\]

B 1 0 1 1 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 0 0
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

pos(x) = Select₁(x)
firstChild(x) = ?

\[ y = \text{Select}_0(x) + 1 \]
// start of x's children in B

if B[y] == 0
    return -1  // is a leaf
else
    return y-x // Rank₁(y)

degree(x) = ?
Select₀(x+1) - (Select₀(x) + 1)

parent(x) =
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[\text{pos}(x) = \text{Select}_1(x)\]
\[\text{firstChild}(x) = ?\]
\[y = \text{Select}_0(x)+1\]
// start of x's children in B
\[\text{if } B[y] == 0 \]
\[\quad \text{return } -1 \quad \text{// is a leaf}\]
\[\text{else}\]
\[\quad \text{return } y-x \quad \text{// Rank}_1(y)\]
\[\text{degree}(x) = ?\]
\[\text{Select}_0(x+1) - (\text{Select}_0(x) + 1)\]
\[\text{parent}(x) = \text{Rank}_0(\text{pos}(x))\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[
\begin{align*}
pos(x) &= \text{Select}_1(x) \\
\text{firstChild}(x) &= ? \\
y &= \text{Select}_0(x) + 1 \\
&\quad \text{// start of } x\text{'s children in } B \\
\text{if } B[y] &= 0 \\
&\quad \text{return } -1 \quad \text{// is a leaf} \\
\text{else} \\
&\quad \text{return } y - x \quad \text{// } \text{Rank}_1(y) \\
\text{degree}(x) &= ? \\
\text{Select}_0(x + 1) - (\text{Select}_0(x) + 1) \\
\text{parent}(x) &= \text{Rank}_0(\text{pos}(x))
\end{align*}
\]

All these operations in \(O(1)\) time!
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[
\begin{align*}
\text{pos}(x) &= \text{Select}_1(x) \\
\text{firstChild}(x) &= \text{?} \\
y &= \text{Select}_0(x)+1 \\
&\text{// start of x's children in B} \\
\text{if } B[y] &= 0 \\
\text{return } -1 &\text{ // is a leaf} \\
\text{else} \\
\text{return } y-x &\text{ // Rank}_1(y) \\
\text{degree}(x) &= \text{?} \\
\text{Select}_0(x+1) - (\text{Select}_0(x) + 1) \\
\text{parent}(x) &= \text{Rank}_0(\text{pos}(x)) \\
\text{subtreeSize}(x) &= \text{?}
\end{align*}
\]
Succinct representation of trees (1)

[LOUDS - Level-order unary degree sequence]

\[ \text{pos}(x) = \text{Select}_1(x) \]
\[ \text{firstChild}(x) = ? \]
\[ y = \text{Select}_0(x) + 1 \]
// start of x's children in B
\[ \text{if } B[y] == 0 \]
\[ \text{return } -1 \] // is a leaf
\[ \text{else} \]
\[ \text{return } y-x \] // Rank_1(y)
\[ \text{degree}(x) = ? \]
\[ \text{Select}_0(x+1) - (\text{Select}_0(x) + 1) \]
\[ \text{parent}(x) = \text{Rank}_0(\text{pos}(x)) \]
\[ \text{subtreeSize}(x) = ? \]

Not efficient!
Nodes of the subtree are spread in B
We store: The bit vector $B$ (2n+1 bits),
its Rank/Select data structure (o(n) bits),
the n char labels (O(n log |Alphabet|) bits)
the n int-labels spread over [1,m], using Elias-Fano in n log m/n + 2n bits

Total space is (m+n) log |Alphabet| + n log m/n + O(n) bits
THUS
avoiding the term n log m bits

$D = \{ \text{ab, bab, bca, cab, cac, cbac, cbba} \}$

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