Ph.d. Defense
Allan Grønlund Jørgensen
April 29th - 2010
Outline

- My Work
- Range Queries
- Range Mode
- Range Selection
My Work

A Linear Time Algorithm for the $k$ Maximal Sums Problem (MFCS’07)
Selecting Sums in Arrays (ISAAC’08)

Data Structures for Range Median Queries (ISAAC’09)
Cell Probe Lower Bounds and Approximations
for Range Mode (ICALP’10)

Priority Queues Resilient to Memory Faults (WADS’07)
Optimal Resilient Dictionaries (ESA’07)
Fault Tolerant External Memory Algorithms (WADS’09)
Counting in the Presence of Memory Faults (ISAAC’09)
# My Work

<table>
<thead>
<tr>
<th>Title</th>
<th>Conference/Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Linear Time Algorithm for the $k$: Maximal Sums Problem</td>
<td>MFCS’07</td>
</tr>
<tr>
<td>Selecting Sums in Arrays</td>
<td>ISAAC’08</td>
</tr>
<tr>
<td>Data Structures for Range Median Queries</td>
<td>ISAAC’09</td>
</tr>
<tr>
<td>Cell Probe Lower Bounds and Approximations for Range Mode</td>
<td>ICALP’10</td>
</tr>
<tr>
<td>Range Median and Selection: Cell Probe Lower Bounds and Adaptive Data Structures</td>
<td>In submission</td>
</tr>
<tr>
<td>Priority Queues Resilient to Memory Faults</td>
<td>WADS’07</td>
</tr>
<tr>
<td>Optimal Resillient Dictionaries</td>
<td>ESA’07</td>
</tr>
<tr>
<td>Fault Tolerant External Memory Algorithms</td>
<td>WADS’09</td>
</tr>
<tr>
<td>Counting in the Presence of Memory Faults</td>
<td>ISAAC’09</td>
</tr>
<tr>
<td>Geometric Computations on Indecisive Points</td>
<td>Submitted soon</td>
</tr>
</tbody>
</table>

Allan G. Jørgensen
This Talk

Data Structures for Range Median Queries
with Gerth S. Brodal (ISAAC’09)

Cell Probe Lower Bounds and Approximations for Range Mode
With Mark Greve, Kasper D. Larsen and Jakob Truelsen (ICALP’10)

Range Median and Selection: Cell Probe Lower Bounds
and Adaptive Data Structures
with Kasper D. Larsen (In submission)
RAM model

- memory of $w$ bit cells
- each memory access takes constant time
- word operations in constant time (arithmetic, shifts etc.)
- $\log n = \Theta(w)$
Range Queries

\[ A = \begin{array}{cccc}
a_1 & a_2 & \cdots & a_i \\
& \cdots & & \cdots \\
a_j & \cdots & & a_n
\end{array} \]
Range Queries

\[ A = \begin{array}{cccc} a_1 & a_2 & \cdots & a_i \end{array} \quad \begin{array}{cccc} \cdots & a_j & \cdots & a_n \end{array} \]

Data Structure
Given indices \( i, j \) compute \( f(A[i..j]) \)
Range Queries

\[ A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n
\end{array} \]

Data Structure
Given indices \( i, j \) compute \( f(A[i..j]) \)
Range Queries

\[ A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

Data Structure
Given indices \( i, j \) compute \( f(A[i..j]) \)

- Query Time
- Space
- Preprocessing Time
Range Queries

\[ A = \begin{array}{cccccccc}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

Data Structure
Given indices \( i, j \) compute \( f(A[i..j]) \)

- Query Time
- Space
- Preprocessing Time

Dynamic - Not discussed today
\[ A = \begin{array}{cccccc}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]
Functions

\[ A = \begin{array}{ccccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

- Average
Functions

\[ A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix} \]

- Average
- Minimum/Maximum
Functions

\[ A = \begin{array}{ccccccc}
     a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

- Average
- Minimum/Maximum
- Median (this talk)
Functions

\[ A = \begin{bmatrix}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n
\end{bmatrix} \]

- Average
- Minimum/Maximum
- Median (this talk)
- \( k \)'th smallest (this talk)
Functions

\[
A = \begin{pmatrix}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n
\end{pmatrix}
\]

- Average
- Minimum/Maximum
- Median (this talk)
- \(k\)'th smallest (this talk)
- Rank of \(e\)
Functions

\[ A = \begin{array}{ccccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

- Average
- Minimum/Maximum
- Median (this talk)
- \( k \)'th smallest (this talk)
- Rank of \( e \)
- Mode (this talk)
Example

\[-1, -2, 0, 0, 0, 1, 0, -5, -1, -2, -1, 0, 4, 0, -2, -1, -1, -3\]
Example

\[
\begin{bmatrix}
-1, -2, 0, 0, 0, 1, 0, -5, -1, -2, -1, 0, 4, 0, -2, -1, -1, -3 \\
0, 0, 1, 1, 1, 3, 1, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0
\end{bmatrix}
\]
Example

Goal Difference in AGF games

\[
\begin{array}{ccccc}
\text{Nov.} & \text{Dec.} & \text{March} & \text{April} \\
[-1, -2, 0, 0, 0, 1, 0, -5, -1, -2, -1, 0, 4, 0, -2, -1, -1, -3] \\
[0, 0, 1, 1, 1, 3, 1, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0] \\
\end{array}
\]
Example

Goal Difference in AGF games

<table>
<thead>
<tr>
<th>Nov.</th>
<th>Dec.</th>
<th>March</th>
<th>April</th>
</tr>
</thead>
<tbody>
<tr>
<td>−1</td>
<td>−2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>−5</td>
<td>−1</td>
<td>−2</td>
<td>−1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>−2</td>
<td>−1</td>
<td>−1</td>
<td>−3</td>
</tr>
</tbody>
</table>

[−1, −2, 0, 0, 1, 0, −5, −1, −2, −1, 0, 4, 0, −2, −1, −1, −3]

[0, 0, 1, 1, 1, 3, 1, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0]

March:
Average: −5/6
Min: −5
Max: 4
Median: −1
Rank of −1: 4
Mode: −1
There are only $O(n^2)$ ranges
There are only $O(n^2)$ ranges.

There are only queries $O(n^2)$ ($O(n^3)$ for selection and rank).
There are only $O(n^2)$ ranges

There are only queries $O(n^2)$ ($O(n^3)$ for selection and rank)

Table with $O(n^2)$ entries $O(1)$ query time
Words on Space

\[ A = \begin{array}{cccc}
  a_1 & a_2 & \cdots & a_i \\
  \vdots & \vdots & \ddots & \vdots \\
  \cdots & \cdots & \cdots & a_j \\
  \cdots & \cdots & \cdots & a_n
\end{array} \]

There are only \( O(n^2) \) ranges
There are only queries \( O(n^2) \) \( (O(n^3) \) for selection and rank) Table with \( O(n^2) \) entries \( O(1) \) query time \( O(n^2) \) space is BAD!!!!!
Words on Space

\[ A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix} \]

There are only \( O(n^2) \) ranges

There are only queries \( O(n^2) \) (\( O(n^3) \) for selection and rank)

Table with \( O(n^2) \) entries \( O(1) \) query time

\( O(n^2) \) space is BAD!!!!!

Near-Linear Space: \( n \log^{O(1)} n \)

Polynomial Space: \( n^{1+O(1)} \)
Range Mode

\[
A = \begin{bmatrix}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{bmatrix}
\]

Given \(i, j\) return a most frequently occurring element in \(A[i, j]\)
Range Mode

\[ A = \begin{array}{cccccc}
    a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

Given \( i, j \) return a most frequently occurring element in \( A[i, j] \)

Example:

\[ A = [a \ b \ b \ a \ a \ b \ a \ b \ b \ a \ a \ b] \]

1 2 3 4 5 6 7 8 9 \ldots
Range Mode

Given \( i, j \) return a most frequently occurring element in \( A[i, j] \)

Example:

\[
A = [a \ b \ b \ a \ a \ b \ a \ b \ b \ a \ a \ b]
\]

\[
1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ ...
\]

\[
\text{mode}(A[1..5]) = a
\]
Range Mode

\[ A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix} \]

Given \( i, j \) return a most frequently occurring element in \( A[i, j] \)

Example:

\[ A = [a \ b \ b \ a \ a \ b \ a \ b \ b \ a \ a \ b] \]

\[ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ \ldots \]

\[ \text{mode}(A[1..5]) = a \]

\[ \text{mode}(A[2..6]) = b \]
## Range Mode - Upper Bounds

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Store Input</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Complete tabulation</td>
<td>$O(1)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Bit tricks</td>
<td>$O(1)$</td>
<td>$O\left(\frac{n^2 \log\log n}{\log^2 n}\right)$</td>
</tr>
<tr>
<td>Tradeoff</td>
<td>$O(n^\varepsilon)$</td>
<td>$O(n^{2-2\varepsilon})$</td>
</tr>
</tbody>
</table>

[Petersen 08, Petersen, Grabowski 08]
Range Mode

\[ A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array} \]

Ideally \( n \log^{O(1)} n \) space and \( \log^{O(1)} n \) time.
### Range Mode

\[
A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array}
\]

Ideally \( n \log^{O(1)} n \) space and \( \log^{O(1)} n \) time.

We have tried and failed :o(}
Range Mode

\[ A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix} \]

Ideally \( n \log^{O(1)} n \) space and \( \log^{O(1)} n \) time.

We have tried and failed :o(

We were able to approximate efficiently

<table>
<thead>
<tr>
<th>Approximation</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-approximation</td>
<td>( O(1) )</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>((1 + \varepsilon))-approximation</td>
<td>( O(\log \frac{1}{\varepsilon}) )</td>
<td>( O(\frac{n}{\varepsilon}) )</td>
</tr>
</tbody>
</table>
What About Lower Bounds?
Cell Probe Model

- Memory of $w$ bit cells
- Complexity is the number of memory probes
- All other computation is free
- Strong enough to include RAM model
Cell Probe Model

- Memory of $w$ bit cells
- Complexity is the number of memory probes
- All other computation is free
- Strong enough to include RAM model

Use strategy initiated by [Miltersen et. al. 93]
refined by Pătraşcu and Thorup
Lower Bound

Range Mode:

Data structure of size $S$ needs $\Omega\left(\frac{\log n}{\log(Sw/n)}\right)$ time for a query
Lower Bound

Range Mode:
Data structure of size $S$ needs $\Omega\left(\frac{\log n}{\log(SW/n)}\right)$ time for a query

Near-Linear space data structures need $\Omega(\log n / \log \log n)$ time
Lower Bound

Range Mode:

Data structure of size $S$ needs $\Omega\left(\frac{\log n}{\log(Sw/n)}\right)$ time for a query

Near-Linear space data structures need $\Omega\left(\log n / \log \log n\right)$ time

Constant query time data structures need $n^{1+\Omega(1)}$ space
Communication Complexity

Alice

Bob
Communication Complexity

Predetermined function $f$

Input $X$

Alice

Input $Y$

Bob
Communication Complexity

Predetermined function $f$

Input $X$

Compute $f(X, Y)$

Minimize communication

Input $Y$

Alice

Bob
Simulating Data Structures

Reduce $f$ to $N/k$ queries on data structure.
Simulating Data Structures

Reduce $f$ to $N/k$ queries on data structure.
Map Alice’s input to queries and Bobs to data structure input
Simulating Data Structures

Reduce $f$ to $N/k$ queries on data structure.
Map Alice’s input to queries and Bobs to data structure input
Data Structure with query time $t$ and space $S$ gives protocol:
Simulating Data Structures

Reduce $f$ to $N/k$ queries on data structure.

Map Alice’s input to queries and Bobs to data structure input

Data Structure with query time $t$ and space $S$ gives protocol:

Bob builds data structure

Alice simulates query algorithm for all queries in parallel

Adress(es) $\rightarrow$ Contents
Simulating Data Structures

Reduce \( f \) to \( N/k \) queries on data structure.

Map Alice’s input to queries and Bobs to data structure input.

Data Structure with query time \( t \) and space \( S' \) gives protocol:

Bob builds data structure
Alice simulates query algorithm for all queries in parallel

Alice sends \( t \log \left( \frac{S}{N/k} \right) = O(tk \log(Sk/N)) \) bits
Bob sends \( twN/k \) bits

Adress(es) \( \rightarrow \) Contents
Blocked Lopsided Set Disjointness

Universe $U = [1, 2, \ldots, n] \times [1, 2, \ldots, B]$
Blocke  Lopsided Set Disjointness

Universe $U = [1, 2, \ldots, n] \times [1, 2, \ldots, B]$

$X = \{(1, B_1), \ldots, (N, B_N)\}$

$Y \subseteq U$
Blocked Lopsided Set Disjointness

Universe $U = [1, 2, \ldots, n] \times [1, 2, \ldots, B]$

$X = \{(1, B_1), \ldots, (N, B_N)\}$

$Y \subseteq U$

Determine if $X \cap Y \neq \emptyset$
Blocked Lopsided Set Disjointness

Universe $U = [1, 2, \ldots, n] \times [1, 2, \ldots, B]$ 

$X = \{(1, B_1), \ldots, (N, B_N)\}$ 

$Y \subseteq U$

Determine if $X \cap Y \neq \emptyset$

Either Alice sends $\Omega(N \log B)$

or Bob sends $\Omega(NB^{1-\delta})$ bits

[Miltersen et al. 93, Pătraşcu 08]
Lower Bound for Range Mode

Lower bound on determining the frequency of the mode
Lower bound on determining the frequency of the mode

Reduce Blocked LSD to $N/k$ queries
Lower Bound for Range Mode

Lower bound on determining the frequency of the mode
Reduce Blocked LSD to $\frac{N}{k}$ queries

Map $X$ to $\frac{N}{k}$ queries

Map $Y$ to input, $n = O(NB)$
Lower Bound for Range Mode

Lower bound on determining the frequency of the mode

Reduce Blocked LSD to $N/k$ queries

Map $X$ to $N/k$ queries  Map $Y$ to input, $n = O(NB)$

The answer to the $N/k$ queries determine whether $X \cap Y \neq \emptyset$
Lower Bound - Intuition

\[ A \quad B \]
Lower Bound - Intuition

\[ A \quad B \]

Query
Lower Bound - Intuition

Solve set disjointness on $A$ and $B$!!!
Each block is a permutation of $[1, 2, \ldots, B]$
Lower Bound - Intuition cont.

Each block is a permutation of \([1, 2, \ldots, B]\)

Query answers set disjointness on subsets of two blocks
Each block is a permutation of \([1, 2, \ldots, B]\)

Query answers set disjointness on subsets of two blocks

First block is subset of Alice’s input

Second block is subset of Bob’s input
Lower Bound Construction

\[ U = [N] \times [B] \]

Split \([N]\) into subsets of \(k\) elm.

\[
X = \{(i, B_i)\}_{i=1,\ldots,N}
\]

\[
X_1 = \{(i, B_i)\}_{i=1,\ldots,k}
\]

\[
X_2 = \{(i, B_i)\}_{i=k+1,\ldots,2k}
\]

\[
\vdots
\]

\[
X_{N/k} = \{(i, B_i)\}_{i=N/k+1,\ldots,N}
\]

\[
Y = \{(i, b)\}
\]

\[
Y_1 = Y \cap \{1,\ldots,k\} \times [B]
\]

\[
Y_2 = Y \cap [k+1,\ldots,2k] \times [B]
\]

\[
\vdots
\]

\[
Y_{N/k} = Y \cap [N/k+1,\ldots,N] \times [B]
\]
Lower Bound cont.

Make an array (range mode input) of two parts
Part 1: $B^k$ options Alice blocks are made of $P_1, \ldots, P_{B^k}$
Part 2: Bobs input
Lower Bound cont.

Make an array (range mode input) of two parts
Part 1: $B^k$ options Alice blocks are made of $P_1, \ldots, P_{B^k}$
Part 2: Bobs input

$$m : [k] \times [B] \rightarrow [kb] \quad \text{(for } [ik + 1, (i + 1)k] \times B \text{ modulo with } k)$$
Make an array (range mode input) of two parts
Part 1: $B^k$ options Alice blocks are made of $P_1, \ldots, P_{B^k}$
Part 2: Bobs input

$$m : [k] \times [B] \rightarrow [kb] \text{ (for } [ik + 1, (i + 1)k] \times B \text{ modulo with } k)$$

Part 2: $[m(Y_i), [kb \setminus m(Y_i)]_{i=1,\ldots,N/k}$

$[kb] = [1, 8]$ and $m(Y_1) = \{1, 4, 6, 7\}$ then $\{1, 4, 6, 7, 2, 3, 5, 8\}$
Lower Bound cont.

Make an array (range mode input) of two parts
Part 1: $B^k$ options Alice blocks are made of $P_1, \ldots, P_{B^k}$
Part 2: Bobs input

$m : [k] \times [B] \rightarrow [kb]$ (for $[ik + 1, (i + 1)k] \times B$ modulo with $k$)

Part 2: $[m(Y_i), [kb] \setminus m(Y_i)]_{i=1,\ldots,N/k}$

$[kb] = [1, 8]$ and $m(Y_1) = \{1, 4, 6, 7\}$ then $\{1, 4, 6, 7, 2, 3, 5, 8\}$

Part 1: $[[kb] \setminus m(P_i), m(P_i)]_{i=1,\ldots,B^k}$
Make an array (range mode input) of two parts

Part 1: $B^k$ options Alice blocks are made of $P_1, \ldots, P_{B^k}$

Part 2: Bobs input

$$m: [k] \times [B] \rightarrow [kb] \text{ (for } [ik + 1, (i + 1)k] \times B \text{ modulo with } k)$$

Part 2: $[m(Y_i), [kb] \setminus m(Y_i)]_{i=1, \ldots, N/k}$

$[kb] = [1, 8]$ and $m(Y_1) = \{1, 4, 6, 7\}$ then $\{1, 4, 6, 7, 2, 3, 5, 8\}$

Part 1: $[[kb] \setminus m(P_i), m(P_i)]_{i=1, \ldots, B^k}$

Range Mode Input: Part 1 followed by Part 2

Notice that Alice knows Part 1
Bob sends $|Y_i|$ for $i = 1, \ldots, N/k$.
Bob sends $|Y_i|$ for $i = 1, \ldots, N/k$

Alice maps each $X_i$ to query

Compute $j$ such that $P_j = X_i$
Lower Bound cont.

Bob sends $|Y_i|$ for $i = 1, \ldots, N/k$

Alice maps each $X_i$ to query

Compute $j$ such that $P_j = X_i$

\[
\begin{array}{ccc}
  m(P_j) & \cdots & m(Y_i)
\end{array}
\]

Query $i$
Bob sends $|Y_i|$ for $i = 1, \ldots, N/k$

Alice maps each $X_i$ to query

Compute $j$ such that $P_j = X_i$

| $m(P_j)$ | $\cdots$ | $m(Y_i)$ |
---|---|---|

Query $i$

Alice knows that number of blocks here
Bob sends $|Y_i|$ for $i = 1, \ldots, N/k$

Alice maps each $X_i$ to query

Compute $j$ such that $P_j = X_i$

Alice knows that number of blocks here

$+2$ iff $X_i \cap Y_i \neq \emptyset$ and $+1$ otherwise
Lower Bound

Alice sends $t \log \left( \binom{S}{N/k} \right) = O(tk \log(Sk/N)) = \Omega(N \log B)$ bits

Or

Bob sends $twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2})$ bits
Lower Bound

Alice sends $t \log \left( \frac{S}{N/k} \right) = O(tk \log(Sk/N)) = \Omega(N \log B)$ bits

Or

Bob sends $twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2})$ bits

$N/k \log(Bk) = o(NB^{1/2})$ so $twN/k = \Omega(NB^{1/2})$
Lower Bound

Alice sends $t \log \left( \binom{S}{N/k} \right) = O(tk \log(Sk/N)) = \Omega(N \log B)$ bits

Or

Bob sends $twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2})$ bits

$N/k \log(Bk) = o(NB^{1/2})$ so $twN/k = \Omega(NB^{1/2})$

Constrain $B > w^2$ and $\log B \geq 1/2 \log(Sk/N)$ then $t = \Omega(k)$
Lower Bound

Alice sends \( t \log \left( \binom{S}{N/k} \right) = O(tk \log(Sk/N)) = \Omega(N \log B) \) bits

Or

Bob sends \( twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2}) \) bits

\( N/k \log(Bk) = o(NB^{1/2}) \) so \( twN/k = \Omega(NB^{1/2}) \)

Constrain \( B > w^2 \) and \( \log B \geq 1/2 \log(Sk/N) \) then \( t = \Omega(k) \)

Bobs array is of size \( NB + B^k kB \)

Make it \( O(n) \) by setting \( k = O(\log_B n) \)
Lower Bound

Alice sends \( t \log \left( \frac{S}{N/k} \right) = O(tk \log(Sk/N)) = \Omega(N \log B) \) bits

Or

Bob sends \( twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2}) \) bits

\( N/k \log(Bk) = o(NB^{1/2}) \) so \( twN/k = \Omega(NB^{1/2}) \)

Constrain \( B > w^2 \) and \( \log B \geq 1/2 \log(Sk/N) \) then \( t = \Omega(k) \)

Bobs array is of size \( NB + B^k kB \)

Make it \( O(n) \) by setting \( k = O(\log_B n) \)

Maximize \( k \) and set \( B = \max\{w^2, Sk/n\} \)
Lower Bound

Alice sends $t \log \binom{S}{N/k} = O(tk \log(Sk/N)) = \Omega(N \log B)$ bits

Or

Bob sends $twN/k + O(N/k \log(Bk)) = \Omega(NB^{1/2})$ bits

$N/k \log(Bk) = o(NB^{1/2})$ so $twN/k = \Omega(NB^{1/2})$

Constrain $B > w^2$ and $\log B \geq 1/2 \log(Sk/N)$ then $t = \Omega(k)$

Bobs array is of size $NB + B^k kB$

Make it $O(n)$ by setting $k = O(\log_B n)$

Maximize $k$ and set $B = \max\{w^2, Sk/n\}$

Obtain $t = \Omega(k) = \Omega(\log n / \log(Sw/n)$
Decision Version of Range Mode

Range $k$-frequency

$$A = \begin{array}{cccccccc}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \\
\end{array}$$
Decision Version of Range Mode

Range $k$-frequency

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix}$$

Given $i, j$ return whether there is an element occurring exactly $k$ times in $A[i, j]$
Decision Version of Range Mode

Range $k$-frequency

\[ A = \begin{array}{ccccccc} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{array} \]

Given $i, j$ return whether there is an element occurring exactly $k$ times in $A[i, j]$.

<table>
<thead>
<tr>
<th>Space</th>
<th>Query</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$\Omega(\log n / \log (Sw/n))$</td>
<td>$2 \leq k = O(1)$</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>$O(\log n / \log \log n)$</td>
<td>any</td>
</tr>
<tr>
<td>$O(n \log n)$</td>
<td>$O(\log^2 \log n)$</td>
<td>$k = 1$</td>
</tr>
</tbody>
</table>
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[ \cdots x \cdots x \cdots x \cdots x \cdots x \cdots x \cdots x \cdots x \cdots x \cdots \]
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[
\begin{bmatrix}
[x_1, x_2] & [x_3, x_4] & [x_5, x_6] & [x_7, x_8] & [x_9, x_{10}] & [x_{11}, x_{12}] \\
\vdots & x & \vdots & x & \vdots & x & \vdots & x & \vdots & x & \vdots
\end{bmatrix}
\]
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[ [x_1, x_2] \ [x_3, x_4] \ [x_5, x_6] \ [x_7, x_8] \ [x_9, x_{10}] [x_{11}, x_{12}] \]

\[ \ldots x \ldots x \ldots x \ldots x \ldots x \ldots x \ldots x \ldots \]

\[ [x_1, x_2] \times [x_7, x_8] \]
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[
\begin{align*}
[x_1, x_2] & \quad [x_3, x_4] & \quad [x_5, x_6] & \quad [x_7, x_8] & \quad [x_9, x_{10}] & \quad [x_{11}, x_{12}] \\
\cdots & \quad x & \quad \cdots & \quad x & \quad \cdots & \quad x & \quad \cdots & \quad x & \quad \cdots \\
\end{align*}
\]

\[
[x_1, x_2] \times [x_7, x_8] \quad [x_3, x_4] \times [x_9, x_{10}]
\]
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[
\begin{array}{cccccccc}
[x_1, x_2] & [x_3, x_4] & [x_5, x_6] & [x_7, x_8] & [x_9, x_{10}] & [x_{11}, x_{12}] \\
\cdots & x & \cdots & x & \cdots & x & \cdots & x & \cdots
\end{array}
\]

\[
[x_1, x_2] \times [x_7, x_8] \quad [x_3, x_4] \times [x_9, x_{10}] \quad [x_5, x_6] \times [x_{11}, x_{12}]
\]
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[
\begin{array}{ccccccccc}
[x_1, x_2] & [x_3, x_4] & [x_5, x_6] & [x_7, x_8] & [x_9, x_{10}] & [x_{11}, x_{12}] \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{array}
\]

\[
\begin{array}{ccccccccc}
x & \cdots & x & \cdots & x & \cdots & x & \cdots & x & \cdots \\
\end{array}
\]

\[
\begin{array}{ccccccccc}
[x_1, x_2] \times [x_7, x_8] & [x_3, x_4] \times [x_9, x_{10}] & [x_5, x_6] \times [x_{11}, x_{12}] \\
\end{array}
\]

Input gives \( O(n) \) rectangles
Reduction to Rectangle Stabbing

\[ k = 3 \]

\[
\begin{align*}
[x_1, x_2] & \hspace{1em} [x_3, x_4] & \hspace{1em} [x_5, x_6] & \hspace{1em} [x_7, x_8] & \hspace{1em} [x_9, x_{10}] & \hspace{1em} [x_{11}, x_{12}] \\
\cdots & \hspace{1em} x & \hspace{1em} \cdots & \hspace{1em} x & \hspace{1em} \cdots & \hspace{1em} x & \hspace{1em} \cdots & \hspace{1em} x & \hspace{1em} \cdots
\end{align*}
\]

\[
\begin{align*}
[x_1, x_2] \times [x_7, x_8] & \hspace{1em} [x_3, x_4] \times [x_9, x_{10}] & \hspace{1em} [x_5, x_6] \times [x_{11}, x_{12}]
\end{align*}
\]

Input gives \(O(n)\) rectangles

\(O(n)\) space \(O(\log n / \log \log n)\) query time

[JáJá, Mortensen, Shi 04]
Reduction from Rectangle Stabbing
Reduction from Rectangle Stabbing
Reduction from Rectangle Stabbing
Reduction from Rectangle Stabbing

A A B B A
Reduction from Rectangle Stabbing

A A B B A C
Reduction from Rectangle Stabbing

A A B B A C C C C
Reduction from Rectangle Stabbing

A A B B A C C C C B
Reduction from Rectangle Stabbing

A A B B A C C C B  C A B A A B B C C
Reduction from Rectangle Stabbing

A A B B A C C C B C A B A A B B C C
Reduction from Rectangle Stabbing

\[\begin{array}{cccccccc}
B & B & A & C & C & C & B & \times \\
A & \times & B & \times & C & A & B & A
\end{array}\]
Reduction from Rectangle Stabbing

A A B B A C C C B

C A B A A B B C C

A A B B A C C C B

C A B A A B B C C
Reduction from Rectangle Stabbing

\[
\begin{array}{cccccc}
\text{A} & \text{A} & \text{B} & \text{B} & \text{A} & \text{C} \\
\text{C} & \text{C} & \text{C} & \text{C} & \text{B} & \text{C} \\
\end{array}
\]

Allan G. Jørgensen

madalgo
Reduction from Rectangle Stabbing

Space $S$ needs $\Omega(\log n / \log(Sw/n))$ time for a query. Pătrașcu 08]

Allan G. Jørgensen
Range Mode Open Problem

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range Mode</td>
<td>$O(n^{2-2\varepsilon})$</td>
<td>$O(n^\varepsilon)$</td>
</tr>
<tr>
<td>Range Mode</td>
<td>$S$</td>
<td>$\Omega(\log n/ \log (Sw/n))$</td>
</tr>
<tr>
<td>Range $k$ freq.</td>
<td>$O(n)$</td>
<td>$O(\log n/ \log \log n)$</td>
</tr>
<tr>
<td>Range $2 \leq k$ freq.</td>
<td>$S$</td>
<td>$\Omega(\log n/ \log (Sw/n))$</td>
</tr>
<tr>
<td>Range 1 freq.</td>
<td>$O(n \log n)$</td>
<td>$O(\log^2 \log n)$</td>
</tr>
</tbody>
</table>

Close the gap between upper bound and lower bound
Range Selection

Given $i, j, k$ return the $k$’th smallest element in $A[i, j]$
Range Selection

\[
A = \begin{bmatrix}
a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n
\end{bmatrix}
\]

Given \(i, j, k\) return the \(k\)'th smallest element in \(A[i, j]\)

- Prefix Selection \((i = 1)\)
- Range Median \((k = \lceil (j - i)/2 \rceil)\)
- Bounded Rank Prefix Selection \((k \leq \kappa)\)
- Fixed Rank Range Selection \((k\) fixed for all queries)
Relation to Range Counting
Relation to Range Counting

Dominance Counting \((x, y)\)
4-Sided Range Counting \((x_1, x_2, y_1, y_2)\)
Relation to Range Counting

Prefix Selection \((x, k)\)
Relation to Range Counting

Prefix Selection \((x, k)\)
Relation to Range Counting

Prefix Selection \((x, k)\)

Until \(k\) elm.
Tight Bounds for Dominance Counting

\(O(n)\) space \(O(\log n / \log \log n)\) query time

[JáJá, Mortensen, Shi 04]

Space \(S\) needs \(\Omega(\log n / \log(Sw/n))\) time for a query.

[Pătrașcu 07,08]
Range Selection Results

Static Data Structure:
\[ O(n) \text{ space and } O(\log n / \log \log n) \text{ query time.} \]
Static Data Structure:

\[ O(n) \text{ space and } O(\log n / \log \log n) \text{ query time.} \]

So it is not harder than range counting
Static Data Structure:
\[ O(n) \] space and \[ O(\log n / \log \log n) \] query time.

So it is not harder than range counting

Dynamic Data Structure:
\[ O(n \frac{\log n}{\log \log n}) \] space and \[ O((\log n / \log \log n)^2) \] query time
Range Selection Data Structure

\[ A = \begin{bmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_j & \cdots & a_n \end{bmatrix} \]

Leaves contain input in sorted order

Fanout = \( \log^\varepsilon n \), \( 0 < \varepsilon < 1 \)
Guiding A Query

Query \((i, j, k)\)
Guiding A Query

Query \((i, j, k)\)

For each subtree \(T_a\) compute \(|A[i, j] \cap T_a|\)

Example: 5, 14, 7, 17, 10

Store as prefix sums: 5, 19, 26, 43, 53
Guiding A Query

Query \((i, j, k)\)

For each subtree \(T_a\) compute \(|A[i, j] \cap T_a|\)

Example: 5, 14, 7, 17, 10
Store as prefix sums: 5, 19, 26, 43, 53

Next node is determined by successor search for \(k\)
Guiding A Query

Query \((i, j, k)\)

For each subtree \(T_a\) compute \(|A[i, j] \cap T_a|\)

Example: 5, 14, 7, 17, 10

Store as prefix sums: 5, 19, 26, 43, 53

Next node is determined by successor search for \(k\)

For constant time only use \(\log^{1-\varepsilon}\) bits of each - fit in one word
Guiding A Query

Query \((i, j, k)\)

For each subtree \(T_a\) compute \(|A[i, j] \cap T_a|\)

Example: 5, 14, 7, 17, 10

Store as prefix sums: 5, 19, 26, 43, 53

Next node is determined by successor search for \(k\)

For constant time only use \(\log^{1-\varepsilon}\) bits of each - fit in one word

<table>
<thead>
<tr>
<th>Good</th>
<th>&quot;Bad&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>05..</td>
<td>14..</td>
</tr>
<tr>
<td>12..</td>
<td>42..</td>
</tr>
<tr>
<td>17..</td>
<td>42..</td>
</tr>
<tr>
<td>32..</td>
<td>42..</td>
</tr>
<tr>
<td>45..</td>
<td>42..</td>
</tr>
<tr>
<td>76..</td>
<td>47..</td>
</tr>
</tbody>
</table>

\(k = 4215\)
Guiding A Query

Query \((i, j, k)\)

For each subtree \(T_a\) compute \(|A[i, j] \cap T_a|\)

Example: 5, 14, 7, 17, 10

Store as prefix sums: 5, 19, 26, 43, 53

Next node is determined by successor search for \(k\)

For constant time only use \(\log^{1-\varepsilon}\) bits of each - fit in one word

<table>
<thead>
<tr>
<th>Good</th>
<th>&quot;Bad&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>05..</td>
<td>14..</td>
</tr>
<tr>
<td>12..</td>
<td>42..</td>
</tr>
<tr>
<td>17..</td>
<td>42..</td>
</tr>
<tr>
<td>32..</td>
<td>42..</td>
</tr>
<tr>
<td>45..</td>
<td>42..</td>
</tr>
<tr>
<td>76..</td>
<td>47..</td>
</tr>
</tbody>
</table>

\[ k = 4215 \]

Following

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>00..</td>
<td>00..</td>
</tr>
<tr>
<td>00..</td>
<td>00..</td>
</tr>
<tr>
<td>00..</td>
<td>00..</td>
</tr>
<tr>
<td>00..</td>
<td>00..</td>
</tr>
<tr>
<td>00..</td>
<td>00..</td>
</tr>
</tbody>
</table>

Allan G. Jørgensen
Range Selection NEW results

Prefix Selection:

\[ A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]
Range Selection NEW results

Prefix Selection:

\[ A = \begin{array}{ccccccc}
a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

Lower bound:
Space \( S \) needs \( \Omega(\log n / \log(Sw/n)) \) time for a query.
Prefix Selection:

\[ A = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n
\end{array} \]

Lower bound:
Space \( S \) needs \( \Omega(\log n / \log(Sw/n)) \) time for a query.

Our data structure is optimal

Bounds ”Equivalent” to range counting
Range Median

Prefix Selection reduces to Range Median
Range Median

Prefix Selection reduces to Range Median

\[ PS = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_n \\
\end{array} \]
Range Median

Prefix Selection reduces to Range Median

\[ PS = \begin{array}{ccccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

\[ RM = \begin{array}{ccccccc}
  \infty, \ldots, \infty & -\infty, \ldots, -\infty & a_1 & a_2 & \cdots & a_x & \cdots & a_n \\
\end{array} \]

2n times \quad n \text{ times}
Prefix Selection reduces to Range Median

\[ PS = \begin{array}{cccccc} a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \end{array} \]

query \((x, k)\)

\[ RM = \begin{array}{cccccccc} \infty, \ldots, \infty & -\infty, \ldots, -\infty & a_1 & a_2 & \cdots & a_x & \cdots & a_n \end{array} \]

2n times \hspace{1cm} n \times 2 \hspace{1cm} n \times 2
Prefix Selection reduces to Range Median

\[ PS = \begin{array}{cccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

query \((x, k)\)

\[ RM = \begin{array}{cccccc}
  \infty & \ldots & \infty & -\infty & \ldots & -\infty & a_1 & a_2 & \cdots & a_x & \cdots & a_n \\
\end{array} \]

\[ \text{query } (3n - 2(\lfloor x/2 \rfloor - k), x + 3n) \]
Prefix Selection reduces to Range Median

\[
\begin{align*}
PS &= \begin{array}{cccccc}
    a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \\
\text{query} \ (x,k)
\end{align*}
\text{2n times}

RM &= \begin{array}{cccccc}
    \infty, \ldots, \infty & -\infty, \ldots, -\infty & a_1 & a_2 & \cdots & a_x & \cdots & a_n \\
\end{array} \\
\text{query} \ (3n - 2(\lfloor x/2 \rfloor - k), x + 3n)
\end{align*}

\[ [b_1, \ldots, b_k, \ldots, b_{x/2}, \ldots, b_x] \]
Prefix Selection reduces to Range Median

\[ PS = \begin{pmatrix} a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \end{pmatrix} \]

query \((x, k)\)

\[ RM = \begin{pmatrix} \infty, \ldots, \infty & -\infty, \ldots, -\infty & a_1 & a_2 & \cdots & a_x & \cdots & a_n \end{pmatrix} \]

query \((3n - 2(\lfloor x/2 \rfloor - k), x + 3n)\)

\([-\infty, -\infty, b_1, \ldots, b_k, \ldots, b_{x/2}, \ldots, b_x]\)
Prefix Selection, Range Selection, Range Median:

<table>
<thead>
<tr>
<th>Space</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>$O(\log n / \log \log n)$</td>
</tr>
<tr>
<td>$S'$</td>
<td>$\Omega(\log n / \log(Sw/n))$</td>
</tr>
</tbody>
</table>
Range Selection Summary

Prefix Selection, Range Selection, Range Median:

<table>
<thead>
<tr>
<th>Space</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(n)$</td>
<td>$O(\log n/ \log \log n)$</td>
</tr>
<tr>
<td>$S'$</td>
<td>$\Omega(\log n/ \log(Sw/n))$</td>
</tr>
</tbody>
</table>

Is this the whole story?
Select depends on $k$

Range Min ($k = 1$):
Select depends on $k$

Range Min ($k = 1$):

$O(n)$ space $O(1)$ query time

[Harel, Tarjan 84]
Select depends on $k$

Range Min ($k = 1$):

$O(n)$ space $O(1)$ query time

[Harel, Tarjan 84]

Extendable to $O(n)$ space and $O(k)$ query time
Select depends on $k$

Range Min ($k = 1$):

$O(n)$ space $O(1)$ query time

[Harel, Tarjan 84]

Extendable to $O(n)$ space and $O(k)$ query time

Hardness depends on $k$?
Select depends on $k$.

Range Min ($k = 1$):
$O(n)$ space $O(1)$ query time
[Harel, Tarjan 84]

Extendable to $O(n)$ space and $O(k)$ query time

Hardness depends on $k$?

Is it easy if $k$ is fixed for all queries?
Adaptive Data Structures

\[ A = \begin{array}{ccccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]
Adaptive Data Structures

\[ A = \begin{array}{cccccc}
    a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

Prefix Selection:
Adaptive Data Structures

\[
A = \begin{bmatrix}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n
\end{bmatrix}
\]

Prefix Selection:
Adaptive Data Structures

Prefix Selection:

Data Structure:

\[ A = \begin{array}{cccccc}
    a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

\[ A = \text{Data Structure: } O(n) \text{ space and } O(\log k \slash \log \log n + \log \log n) \text{ query time.} \]
Prefix Selection:
Data Structure:
\(O(n)\) space and \(O(\log k / \log \log n + \log \log n)\) query time.

Range Selection:
Adaptive Data Structures

\[ A = \begin{array}{cccccc}
    \text{a}_1 & \text{a}_2 & \cdots & \text{a}_i & \cdots & \text{a}_x & \cdots & \text{a}_n \\
\end{array} \]

Prefix Selection:

Data Structure:

\[ O(n) \text{ space and } O(\log k / \log \log n + \log \log n) \text{ query time.} \]

Range Selection:
Adaptive Data Structures

\[ A = \begin{array}{cccccccc}
  a_1 & a_2 & \cdots & a_i & \cdots & a_x & \cdots & a_n \\
\end{array} \]

Prefix Selection:
Data Structure:
\[ O(n) \text{ space and } O(\log k / \log \log n + \log \log n) \text{ query time.} \]

Range Selection:
Data Structure:
\[ O(n) \text{ space and } O(\log k / \log \log n + \log^2 \log n) \text{ query time.} \]
Adaptive Data Structures

Let \( A = [a_1, a_2, \ldots, a_i, \ldots, a_x, \ldots, a_n] \)

Prefix Selection:
Data Structure:
\( O(n) \) space and \( O(\log k / \log \log n + \log \log n) \) query time.

Range Selection:
Data Structure:
\( O(n) \) space and \( O(\log k / \log \log n + \log^2 \log n) \) query time.

Are these optimal?
Bounded Rank Prefix Selection:
Input array and integer $\kappa$. All queries has $k \leq \kappa$. 
Range Selection NEW results

Bounded Rank Prefix Selection:
Input array and integer $\kappa$. All queries has $k \leq \kappa$.

Space $S$ needs $\Omega(\log \kappa / \log(Sw/n))$ time for a query.
Range Selection NEW results

Bounded Rank Prefix Selection:
Input array and integer $\kappa$. All queries has $k \leq \kappa$.

Space $S$ needs $\Omega(\log \kappa / \log(Sw/n))$ time for a query.

Fixed Rank Range Selection.
Array and integer $k$. All queries selects the $k$’th smallest elm.
Range Selection NEW results

Bounded Rank Prefix Selection:
Input array and integer $\kappa$. All queries has $k \leq \kappa$.

Space $S$ needs $\Omega(\log \kappa / \log(Sw/n))$ time for a query.

Fixed Rank Range Selection.
Array and integer $k$. All queries selects the $k$’th smallest elm.

Similar to the reduction for range median we get:
$S$ space needs $\Omega(\log k / \log(Sw/n))$ query time
Range Selection NEW results

Bounded Rank Prefix Selection:
Input array and integer \( \kappa \). All queries has \( k \leq \kappa \).

Space \( S \) needs \( \Omega(\log \kappa / \log(Sw/n)) \) time for a query.

Fixed Rank Range Selection.
Array and integer \( k \). All queries selects the \( k \)'th smallest elm.

Similar to the reduction for range median we get:
\( S \) space needs \( \Omega(\log k / \log(Sw/n)) \) query time

Our adaptive data structures are almost optimal
Range Selection NEW results

Bounded Rank Prefix Selection:
Input array and integer \( \kappa \). All queries has \( k \leq \kappa \).

Space \( S \) needs \( \Omega\left(\log \kappa / \log \left(\frac{Sw}{n}\right)\right) \) time for a query.

Fixed Rank Range Selection.
Array and integer \( k \). All queries selects the \( k \)'th smallest elm.

Similar to the reduction for range median we get:
\( S \) space needs \( \Omega\left(\log k / \log \left(\frac{Sw}{n}\right)\right) \) query time

Our adaptive data structures are almost optimal

Range Min Data Structures does not generalize!!!
## Range Selection Summary

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range Selection</strong></td>
<td>$O(n)$</td>
<td>$O(\log n / \log \log n)$</td>
</tr>
<tr>
<td><strong>Prefix Selection</strong></td>
<td>$O(n)$</td>
<td>$O(\log k / \log \log n + \log \log n)$</td>
</tr>
<tr>
<td><strong>Range Selection</strong></td>
<td>$O(n)$</td>
<td>$O(\log k / \log \log n + \log^2 \log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range Selection</strong></td>
<td>$S$</td>
<td>$\Omega(\log n / \log (Sw/n))$</td>
</tr>
<tr>
<td><strong>Bounded Rank Prefix Sel.</strong></td>
<td>$S$</td>
<td>$\Omega(\log \kappa / \log (Sw/n))$</td>
</tr>
<tr>
<td><strong>Fixed Rank Range Selection</strong></td>
<td>$S$</td>
<td>$\Omega(\log k / \log (Sw/n))$</td>
</tr>
</tbody>
</table>
Goto Next Slide
Thank You