Augmenting data structures (BSTs)
Finding the rank of an element in a set

Use array:

\[
\begin{array}{cccccccc}
P & F & C & H & Q & A & N & D & M \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9
\end{array}
\]

\[\text{rank}(F) = ?\]
Finding the rank of an element in a set

Use array:

```
   P  F  C  H  Q  A  N  D  M
```

```
   C  A  D  F  P  H  Q  N  M
```

\[ \text{rank}(F) = ? \]

\[ \Theta(?) \]
Finding the rank of an element in a set

Use array:

<table>
<thead>
<tr>
<th>P</th>
<th>F</th>
<th>C</th>
<th>H</th>
<th>Q</th>
<th>A</th>
<th>N</th>
<th>D</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>D</th>
<th>F</th>
<th>P</th>
<th>H</th>
<th>Q</th>
<th>N</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

\[ \text{rank}(F) = ? \]

\[ \Theta(n) \]

OK if done once.

Not for multiple queries

Suggestions?
Finding the rank of an element in a set

Use array:

\[
\begin{array}{cccccccc}
P & F & C & H & Q & A & N & D & M \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
C & A & D & F & P & H & Q & N & M \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

rank(F) = ?

\{ 
\text{\(\Theta(n)\)}
\text{\ OK if done once.}
\text{\ Not for multiple queries}
\}

Preprocess (sort)

\[
\begin{array}{cccccccc}
A & C & D & F & H & M & N & P & Q \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\text{O(nlogn)}

Now all queries: \text{O(1)}
Finding the rank of an element in a set

Use array:

\[
\begin{array}{cccccccccc}
P & F & C & H & Q & A & N & D & M \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
C & A & D & F & P & H & Q & N & M \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
A & C & D & F & H & M & N & P & Q \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]

\[
\text{rank}(F) = ?
\]

\[
\begin{array}{c}
\Theta(n) \\
\text{OK if done once.}
\end{array}
\]

\[
\begin{array}{c}
\text{Not for multiple queries}
\end{array}
\]

\[
\begin{array}{c}
O(n \log n) \\
\text{Now all queries: } O(1)
\end{array}
\]

What if we want to insert/delete?
Finding the rank of an element in a set

Use array:

<table>
<thead>
<tr>
<th>P</th>
<th>F</th>
<th>C</th>
<th>H</th>
<th>Q</th>
<th>A</th>
<th>N</th>
<th>D</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>D</th>
<th>F</th>
<th>P</th>
<th>H</th>
<th>Q</th>
<th>N</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

\[ \text{rank}(F) = ? \]

\( \Theta(n) \)
- OK if done once.
- Not for multiple queries

Preprocess (sort):

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
<th>F</th>
<th>H</th>
<th>M</th>
<th>N</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
</tbody>
</table>

\( O(n \log n) \)
- Now all queries: \( O(1) \)

What if we want to insert/delete? \( \rightarrow \) bad \( O(n) \)
Finding the rank of an element in a DYNAMIC SET with pre-processing

Allow insertions & deletions "quickly"
Finding the rank of an element in a dynamic set with preprocessing

RB-tree contains sorted letters
Finding the rank of an element in a DYNAMIC SET with preprocessing

RB-tree contains sorted letters
Now we can quickly restore sorted order
(still need to get ranks)
Finding the rank of an element in a dynamic set with preprocessing

RB-tree contains sorted letters
Now we can quickly restore sorted order
Store ranks
Finding the rank of an element in a dynamic set with preprocessing

Dynamic X

RB-tree contains sorted letters
Now we can quickly restore sorted order
Store ranks... → bad
(too many ranks change w/ insert)
Finding the rank of an element in a dynamic set with preprocessing

- RB-tree contains sorted letters
- Now we can quickly restore sorted order
  - Store ranks... → bad
    - (too many ranks change with insert)

Dynamic?

Store subtree sizes
Using an Augmented R-B Tree to Find Ranks (with subtree sizes)
USING AN AUGMENTED R-B TREE TO FIND RANKS
(with subtree sizes)

```
Rank(J) : Walk up,
   ✓ count smaller ancestors
```

= 2
Using an augmented R-B tree to find ranks
(with subtree sizes)

\[
\text{Rank}(J) : \text{ Walk up,} \\
\quad \checkmark \text{ count smaller ancestors,} \\
\quad \text{ but also} \\
\quad + \text{ add sizes of subtrees containing smaller numbers.}
\]

\[
\begin{align*}
\text{J} & \quad \text{I} \\
2 \quad & \quad \text{G} \\
\quad & \quad \text{H} \\
\quad & \quad = 4
\end{align*}
\]
Using an augmented R-B tree to find ranks
(with subtree sizes)

Rank(J): Walk up,
✓ count smaller ancestors,
but also
+ add sizes of subtrees containing smaller numbers.

\[ \text{Rank}(J) = 5 \]
Using an augmented R-B tree to find ranks
(with subtree sizes)

Rank(J): Walk up,
✓ count smaller ancestors,
but also
+ add sizes of subtrees containing smaller numbers.

\[
\text{Rank}(J) = \text{Walk up, count smaller ancestors, add sizes of subtrees containing smaller numbers.}
\]
USING AN AUGMENTED R-B TREE TO FIND RANKS
(with subtree sizes)

\[ \text{Rank}(J) : \text{Walk up,} \]
\[ \checkmark \text{count smaller ancestors,} \]
\[ \text{but also} \]
\[ + \text{add sizes of subtrees containing smaller numbers.} \]

\[ J \quad I \quad 2 \quad G \quad F \quad D = 10 \]
Don't forget to walk all the way up.
$O(\log n)$ time
The balanced BST can be built in $\Theta(n \log n)$ time.
The balanced BST can be built in $\Theta(n \log n)$ time.

Compute subtree sizes after building by postorder walk.
The balanced BST can be built in $\Theta(n \log n)$ time

Compute subtree sizes after building by postorder walk
The balanced BST can be built in $\Theta(n\log n)$ time

Compute subtree sizes after building by postorder walk
The balanced BST can be built in $\Theta(n\log n)$ time

Compute subtree sizes after building by postorder walk
The balanced BST can be built in $\Theta(n\log n)$ time

Compute subtree sizes after building by postorder walk
The balanced BST can be built in $\Theta(n\log n)$ time

Compute subtree sizes after building by postorder walk...

...or update path $\{}$ when inserting $\}$
The balanced BST can be built in $\Theta(n \log n)$ time

Compute subtree sizes after building by postorder walk...

... or update path when inserting

BUT...
The balanced BST can be built in $\Theta(n \log n)$ time.

Compute subtree sizes after building by postorder walk...

... or update path $\tau$ when inserting $\tau$

BUT...

we will need to rebalance
Can we update subtree sizes when inserting/deleting data?
Can we update subtree sizes when inserting/deleting data?

Use a RB tree

when are subtree sizes affected?
Can we update subtree sizes when inserting/deleting data?

Use a RB tree

when are subtree sizes affected? Rotations
AUGMENTED TREE TO FIND RANKS

- easy to find rank:
  - look at ancestor path & some adjacent subtree sizes

- subtree sizes can be updated when inserting and rebalancing

$O(\log n)$ per search/insertion/deletion
DYNAMIC SELECTION
find the i-th smallest element in a set

Static: $\Theta(n)$

Dynamic: $O(n\log n)$ preprocessing $\rightarrow$ balanced BST w/ subtree sizes $O(\log n)$ after that
Select($x, i$) $\triangleright$ get $i$-th element in subtree rooted at $x$.

$k \leftarrow 1 + \text{size}(l_x) \quad \triangleright \ l_x: \text{left child of } x$

if $i = k$, return $x$. 

\[ l_x \rightarrow \]
\textbf{Select}(x, i) \quad \text{\textcopyright get $i$-th element in subtree rooted at $x$.}

\begin{align*}
k & \leftarrow 1 + \text{size}(l_x) \quad \text{\textcopyright $l_x$: left child of $x$} \\
\text{if } i &= k, \text{ return } x.
\end{align*}

\textbf{example: } i = 5

\[ k = 6 \]

\[ k \leftarrow 1 + 5 \]

\[ i < k \]

Now what?
Select($x, i$)  \hspace{1em} \text{get $i$-th element in subtree rooted at $x$.}

\[
k \leftarrow 1 + \text{size}(l_x) \quad \text{\textbackslash l_x: left child of $x$}
\]

if $i = k$, return $x$.
else if $i < k$, return Select($l_x, i$)
Select($x, i$)  \hspace{1cm} \text{get } i\text{-th element in subtree rooted at } x.

$k \leftarrow 1 + \text{size}(l_x)$  \hspace{1cm} \text{$l_x$: left child of } x

if $i = k$, return $x$.
else if $i < k$, return $\text{Select}(l_x, i)$

example: $i = 5$

$k = 6$

$k \leftarrow 1 + 5$

$i < k \Rightarrow \text{Select}(c, 5)$
Select($x, i$) \hspace{1em} \text{get } i\text{-th element in subtree rooted at } x.

\begin{align*}
k &\leftarrow 1 + \text{size}(l_x) \hspace{1em} \text{ } l_x: \text{left child of } x \\
\text{if } i = k, &\text{ return } x. \\
\text{else if } i < k, &\text{ return } \text{Select}(l_x, i)
\end{align*}

\text{example: } i = 5

\begin{align*}
\text{Select(root, 5)} \\

k &\leftarrow 1 + 5 \\
\text{i<k } \Rightarrow \text{ Select(c, 5)} \\

k &\leftarrow 1 + 1 \\
\text{i>k }
\end{align*}

...next?
Select(x, i) \ get i-th element in subtree rooted at x.

k ← 1 + size(l_x) \ l_x: left child of x

if i = k, return x.
else if i < k, return Select(l_x, i)
else (i > k) return Select(r_x, i-k)

element: i=5

Select(root, 5)

k ← 1 + 5

i < k ⇒ Select(c, 5)
k = 1 + 1
i > k ⇒ Select(F, 3)
Select(x, i) \ get i-th element in subtree rooted at x.

\[ k \leftarrow 1 + \text{size}(l_x) \] \ l_x: left child of x

if \ i = k \, \text{, return } x.
else if \ i < k \, \text{, return } \text{Select}(l_x, i)
else \ (i > k) \, \text{ return } \text{Select}(r_x, i - k)

example: \ i = 5

Select(root, 5)

\[ k \leftarrow 1 + 5 \]

\[ i < k \Rightarrow \text{Select}(C, 5) \]

\[ k = 1 + 1 \]

\[ i > k \Rightarrow \text{Select}(F, 3) \]

\[ k = 1 + 1 \]

\[ i > k \Rightarrow \text{Select}(H, 1) \]
Select\( (x, i) \) \hspace{1em} \text{get } i\text{-th element in subtree rooted at } x.

\[
k \leftarrow 1 + \text{size}(lx) \hspace{1em} \text{\(l_x\): left child of } x
\]

\text{if } i = k \text{, return } x.
\text{else if } i < k \text{, return } \text{Select}(lx, i)
\text{else (}\ i > k \text{)} \text{, return } \text{Select}(rx, i-k)

\text{example: } i = 5

\text{Select}\( (\text{root}, 5) \)

\[
k = 6
\]

\text{i < } k \Rightarrow \text{Select}(c, 5)

\[
k = 1 + 1
\]

\text{i > } k \Rightarrow \text{Select}(F, 3)

\[
k = 1 + 1
\]

\text{i > } k \Rightarrow \text{Select}(H, 1)

\[
k = 1 + 0
\]

\text{i = } k \Rightarrow \text{return } H
DYNAMIC SELECTION

find the i-th smallest element in a set

Static: $\Theta(n)$

Dynamic: $O(n \log n)$ preprocessing $\rightarrow$ balanced BST w/ subtree sizes

$O(\log n)$ query / insert / delete