**Range Counting**

Count (or enumerate) objects in a given range (many times)

1D:

```
[..........................]
```

2D:

```
[..........................]
```
USE ARRAY: $O(\log n)$ to place $L, R \rightarrow$ to count.
$O(k + \log n)$ to enumerate/report.
USE ARRAY: $O(\log n)$ to place $L, R \rightarrow$ to count.

$O(k + \log n)$ to enumerate/report.

2 problems:
- doesn't generalize to 2D (no array)
- not dynamic ... insert, delete data: $O(n)$
store size of subtree in each node
1D:

\[
\begin{array}{c}
\text{k = 6} \\
\end{array}
\]

\[
\begin{array}{c}
\text{L} \\
\text{R}
\end{array}
\]

\[\square \rightarrow \text{count } 1\]
1D:

\[ L \rightarrow \text{count } 1 \rightarrow 1 \rightarrow 6 \rightarrow R \rightarrow \text{count subtree} \]

\[ k = 6 \]
$O(\log n)$ nodes visited

- 2 paths root→leaf
- 1 neighbor off path per node

- : always "inside"
- X: always "outside"
Intuitive idea for 2D range counting: search X, then Y
Intuitive idea for 2D range counting: search X, then Y

Expensive to Y-sort all X-ranges

\[ O(\log n) \cdot O(n^2) \]

by dealing with ranges in careful order
Intuitive idea for 2D range counting: search X, then Y

Expensive to Y-sort all X-ranges

$O(\log n) \cdot O(n^2)$

by dealing with ranges in careful order

Also expensive to store all X-ranges by brute-force
Every X-range is represented by $O(\log n)$ nodes.
Every $X$-range is represented by $O(\log n)$ nodes.

For each node, create a new (aux.) tree containing all nodes of subtree, sorted by $Y$. 

Size?
Every X-range is represented by $O(\log n)$ nodes.

For each node, create a new (aux.) tree containing all nodes of subtree, sorted by $Y$.

$\sum_{i=1}^{n} \frac{n}{2^i} = \Theta(n\log n)$
Every X-range is represented by $O(\log n)$ nodes.

For each node, create a new (aux.) tree containing all nodes of subtree, sorted by Y.

Size of aux. trees:

$$1 \cdot n + 2 \cdot \frac{n}{2} + 4 \cdot \frac{n}{4} + 8 \cdot \frac{n}{8} + \ldots + n \cdot 1$$

Number of aux. trees:

$$= \Theta(n \log n \text{ space})$$

Every node is represented in $O(\log n)$ aux. subtrees.
Build primary tree: $\Theta(n \log n)$ time
Build primary tree: $\Theta(n \log n)$ time
Build all aux. trees, bottom-up (merge)
Build primary tree: $\Theta(n\log n)$ time
Build all aux. trees, bottom-up (merge): $\Theta(n\log n)$
Build primary tree: $\Theta(n \log n)$ time
Build all aux. trees, bottom-up (merge): $\Theta(n \log n)$

\[
\text{if built independently: } T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n \log n) \\
= n \log n + 2\left(\frac{n}{2} \log \frac{n}{2}\right) + 4\left(\frac{n}{4} \log \frac{n}{4}\right) + \cdots = \Theta(n \log^2 n)
\]
(X, Y) Query → Search X: identify O(log n) nodes □ & ○
(X, Y)-Query $\rightarrow$ Search X: identify $O(\log n)$ nodes $\square$ \\  
For each $\square$ check if y-range is ok. $O(1)$
(X, Y)-Query \rightarrow \text{Search X: identify } \mathcal{O}(\log n) \text{ nodes } \square \& \circ \text{.}

For each \square \text{ check if y-range is ok. } \mathcal{O}(1)\text{.}

For each \circ \text{ check y-range of aux. tree } \mathcal{O}(\log n)\text{.}

\text{TOTAL WORK: } \mathcal{O}(\log n) \text{.}
(x, y) - Query → Search X: identify O(log n) nodes ⊗ & ⊘
  For each ⊗ check if y-range is ok. O(1)
  For each ⊘ check y-range of aux. tree O(log n)
Union results

TOTAL WORK
O(log n)
O(log^2 n)
O(log n)
2D: tree of trees
3D: tree of trees of trees
3D: tree of trees of trees

Space:
- root links to 2D structure: \( n \log n \)
- 2nd level: \( 2 \cdot \frac{n}{2} \log \frac{n}{2} \)
3D: tree of trees of trees

Space:
- root links to 2D structure: $n \log n$
- 2nd level: $2 \cdot \frac{n}{2} \log \frac{n}{2}$
- $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n \log n)$

$\Theta(n \log^2 n)$
3D: tree of trees of trees

Space:
- root links to 2D structure: nlogn
- 2nd level: $2 \cdot \frac{n}{2} \log \frac{n}{2}$
- $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n\log n)$
  \[ \Theta(n\log^2 n) \]

Query:
- $O(\log n) \cdot T(2D)$
  \[ \Theta(n\log^3 n) \]
<table>
<thead>
<tr>
<th>Range Counting</th>
<th>Size of Structure</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add $\Theta(R)$ to report $R$ items</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1D Tree</td>
<td>$\Theta(n)$</td>
<td>$\Theta(\log n)$</td>
</tr>
<tr>
<td>2D Tree of trees</td>
<td>$\Theta(n\log n)$</td>
<td>$O(\log^2 n)$</td>
</tr>
<tr>
<td>$k \geq 3$, tree (of trees)</td>
<td>$O(n\log^{k-1} n)$</td>
<td>$O(\log^k n)$</td>
</tr>
</tbody>
</table>
Back to 2D: tree of trees (just static)
Back to 2D: tree of trees (just static)

Replace aux. trees with arrays $\rightarrow$ doesn't change search idea (or time)
Back to 2D: tree of trees (just static)
Replace aux. trees with arrays → doesn’t change search idea (or time)

For every array,
add a pointer from every key
to the position it would be at
in each child array
Back to 2D: tree of trees (just static)
Replace aux. trees with arrays → doesn’t change search idea (or time)

For every array,
add a pointer from every key to the position it would be at in each child array

Same space complexity
New \((X,Y)\)-search:

1) binary search by \(Y\) at root (array)
   \[\Rightarrow\text{mark } Y_{\text{max}} \& Y_{\text{min}}\]
New \((X,Y)\)-search:

1) binary search by \(Y\) at root (array)
   \[ \text{mark } Y_{\text{max}} & Y_{\text{min}} \]

2) do regular search by \(X\) ...
New \((X,Y)\)-search:

1) binary search by \(Y\) at root (array)
   - mark \(Y_{\text{max}}\) & \(Y_{\text{min}}\)

2) do regular search by \(X\) but also:
   - for every node visited, follow pointers to child array
     - tells us #points in \(Y\) range
New \((X,Y)\)-search:

1) binary search by \(Y\) at root (array)
   \(\Rightarrow\) mark \(Y_{\text{max}}\) & \(Y_{\text{min}}\)

2) do regular search by \(X\) but also:
   - for every node visited,
     follow pointers to child array
     \(\Rightarrow\) tells us \#points in \(Y\) range
   - when we have a subtree entirely in \(X\)-range
     \(\Rightarrow\) we get \#pts also in \(Y\)-range
     in \(O(1)\) time
     instead of \(O(\log n)\)
New \((X,Y)\)-search:

1) binary search by \(Y\) at root (array)
   \[\text{mark} \ Y_{\text{max}} \& \ Y_{\text{min}}\]

2) do regular search by \(X\) but also:
   - for every node visited,
     follow pointers to child array
     \[\text{tells us #points in Y range}\]
   - when we have a subtree entirely in \(X\)-range
     \[\text{we get #pts also in Y-range in } O(1) \text{ time}\]
     \[\text{instead of } O(\log n)\]

\[\text{Time: } O(\log n)\]
Works only for last level: time = $O(\log^{d-1} n)$

Can be made dynamic

Can be improved → projects
RANGE COUNTING
add $\Theta(R)$ to report $R$ items

<table>
<thead>
<tr>
<th>1D</th>
<th>size of structure</th>
<th>query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>$\Theta(n)$</td>
<td>$\Theta(\log n)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D</th>
<th>size of structure</th>
<th>query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree of trees</td>
<td>$\Theta(n\log n)$</td>
<td>$O(\log^2 n) \rightarrow \Theta(\log n)$</td>
</tr>
<tr>
<td>Structure</td>
<td>Size of Structure</td>
<td>Query Time</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>1D Tree</td>
<td>$\Theta(n)$</td>
<td>$\Theta(\log n)$</td>
</tr>
<tr>
<td>2D Tree of trees</td>
<td>$\Theta(n\log n)$</td>
<td>$O(\log^2 n) \rightarrow \Theta(\log n)$</td>
</tr>
<tr>
<td>$k \geq 3$ tree of trees</td>
<td>$O(n\log^{k-1} n)$</td>
<td>$O(\log^k n) \rightarrow O(\log^{k-1} n)$</td>
</tr>
<tr>
<td>Add $\Theta(R)$ to report $R$ items</td>
<td>Size of structure</td>
<td>Query time</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>1D Tree</td>
<td>$\Theta(n)$</td>
<td>$\Theta(\log n)$</td>
</tr>
<tr>
<td>2D Tree of trees</td>
<td>$\Theta(n\log n)$</td>
<td>$O(\log^2 n) \rightarrow \Theta(\log n)$</td>
</tr>
<tr>
<td>$k \geq 3$ tree (of trees)$^{k-1}$</td>
<td>$O(n\log^{k-1} n) \rightarrow O\left(n\frac{\log^{k-1} n}{\log \log n}\right)$</td>
<td>$O(\log^k n) \rightarrow O(\log^{k-1} n)$</td>
</tr>
</tbody>
</table>
**Range Counting**

Add $\Theta(R)$ to report $R$ items.

<table>
<thead>
<tr>
<th>1D Tree</th>
<th>Size of Structure</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(n)$</td>
<td>$\Theta(\log n)$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D Tree of trees</th>
<th>Size of Structure</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Theta(n \log n)$</td>
<td>$O(\log^2 n) \rightarrow \Theta(\log n)$</td>
<td></td>
</tr>
</tbody>
</table>

$k \geq 3$

Tree (of trees)$^{k-1}$

Size of Structure:

- For $k = 3$, $O(n \log^{k-1} n) \rightarrow O(n \cdot \frac{\log^{k-1} n}{\log \log n})$
- For $k = 4$, $O(\log^4 n) \rightarrow O(\log^{k-1} n)$

Query Time:

- $O(\log^k n) \rightarrow O(\log^{k-2} n)$

Paper by B. Chazelle:

$\Theta(n \log^k n) \rightarrow O(\log^{k-2} n)$
<table>
<thead>
<tr>
<th>Range Counting</th>
<th>Size of Structure</th>
<th>Query Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>k=2 k-d tree</td>
<td>$\Theta(n)$</td>
<td>$O(\sqrt{n})$</td>
</tr>
<tr>
<td>(see separate pdf)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tree of trees</td>
<td>$\Theta(n \log n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>Dimension $k &gt; 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k-d tree</td>
<td>$\Theta(kn)$</td>
<td>$O(k \cdot n^{1-\frac{1}{k}})$</td>
</tr>
<tr>
<td>(tree of) $^{k-1}$ trees</td>
<td>$o(n \log^{k-1} n)$</td>
<td>$o(\log^{k-1} n)$</td>
</tr>
</tbody>
</table>

Add $O(R)$ to report $R$ items.