Binary tree traversal

**Preorder traversal of T:** root, left subtree, right subtree.

**Postorder traversal of T:** left subtree, right subtree, root.

**Inorder traversal of T:** left subtree, root, right subtree.

![Binary tree traversal](image)
Orthogonal Range Searching
1D Orthogonal Range Searching

\textbf{Build\_1D\_RT}(A, i, j)

\textbf{Input:} A sorted array \( A = A[0..n - 1] \).

\textbf{if} \( (i = j) \)

create a node with value \( A[i] \)
return a pointer to this node

\textbf{else}

\( k = \lfloor \frac{i + j}{2} \rfloor \)
create a node \( v \) with value \( A[k] \)
left_child\((v) \leftarrow \text{Build\_1D\_RT}(A, i, k)\)
right_child\((v) \leftarrow \text{Build\_1D\_RT}(A, k + 1, j)\)
return a pointer to \( v \)
1D Orthogonal Range Searching

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>19</td>
<td>23</td>
<td>30</td>
<td>37</td>
<td>49</td>
<td>59</td>
<td>62</td>
<td>70</td>
<td>80</td>
<td>89</td>
<td>100</td>
</tr>
</tbody>
</table>
Searching with the interval [18:77]
**FindSplitNode**\((\mathcal{T}, x, x')\)

**Input:** A 1D range tree \(\mathcal{T}\) and two values \(x \leq x'\).

**Output:** The node \(v\) where the paths to \(x\) and \(x'\) split, or the leaf where both paths end.

\[
v \leftarrow \text{root}(\mathcal{T})
\]

**while** \(v\) is not a leaf and \((x' \leq x_v \text{ or } x > x_v)\)

**if** \(x' \leq x_v\)

\[
v \leftarrow \text{left\_child}(v)
\]

**else**

\[
v \leftarrow \text{right\_child}(v)
\]
Searching with the interval $[18:77]$
1DRangeQuery($T, [x, x']$)

**Input:** A range tree $T$ and a range $[x, x']$.

**Output:** All points that lie in the range.

$v_{split} \leftarrow \text{FindSplitNode}(T, x, x')$

If $v_{split}$ is a leaf, then report the point stored in $v_{split}$ (if in the range); return.

**comment:** Follow the path to $\mu$ and report the points in the subtrees to the right of the path.

$v \leftarrow \text{left_child}(v_{split})$

**while** $v$ is not a leaf

*if* $x \leq x_v$

ReportSubtree(right_child($v$))

$v \leftarrow \text{left_child}(v)$

*else* $v \leftarrow \text{right_child}(v)$

Report the point stored in $\mu$ if in the range.

**comment:** Follow the path to $\mu'$ and report the points in the subtrees to the left of the path.

Report the point stored in $\mu'$ if in the range.
RT on $x$-coords $\mathcal{T}$

assoc tree of $v$

RT on $y$-coords
**Build_2D_RT**(*A, i, j*)

**Input:** An array \(A = A[0..n-1]\) of points sorted by their \(x\)-coordinate.

**Output:** The root of a 2-dimensional RT.

**if** \((i = j)\)

- create the associated structure of \(A[i]\)
- create a node with value \(A[i].x\) and
  let it point to its associated structure
- return a pointer to this node

**else**

- create the associated structure of \(A[i..j]\)
- \(k = \lfloor \frac{i+j}{2} \rfloor\)
- create a node \(v\) with value \(A[k].x\) and
  let it point to its associated structure
- left_child\((v) \leftarrow \text{Build}_2\text{D}_\text{RT}(A, i, k)\)
- right_child\((v) \leftarrow \text{Build}_2\text{D}_\text{RT}(A, k + 1, j)\)
- return a pointer to \(v\)
2DRangeQuery($\mathcal{T}, [x, x'] \times [y, y'])$

**Input:** A 2D RT $\mathcal{T}$ and a range $[x, x'] \times [y, y'].$

**Output:** All points that lie in the range.

$v_{split} \leftarrow \text{FindSplitNode}(\mathcal{T}, x, x')$

If $v_{split}$ is a leaf, then report the point stored in $v_{split}$ (if in the range); return.

**comment:** Follow the path to $\mu$ and call 1DRangeQuery for each of the subtrees to the right of the path.

$v \leftarrow \text{left_child}(v_{split})$

**while** $v$ is not a leaf

- **if** $x \leq x_v$
  
  1DRangeQuery($\mathcal{T}_{assoc}(\text{rc}(v)), [y, y'])$
  
  $v \leftarrow \text{left_child}(v)$

- **else** $v \leftarrow \text{right_child}(v)$

Report the point stored in $\mu$ if in the range.

**comment:** Follow the path to $\mu'$ and call 1DRangeQuery for each of the subtrees to the left of the path.

Report the point stored in $\mu'$ if in the range.