RANGE TREES

- Balanced binary search tree
- All data stored in the leaf nodes
- Leaf nodes linked in sorted order by a doubly-linked list
- Searching for \([B : E]\)
  1. find node with smallest value \(\geq B\) or largest \(\leq B\)
  2. follow links until reach node with value \(> E\)
- \(O(\log_2 N + F)\) time to search, \(O(N \cdot \log_2 N)\) to build, and \(O(N)\) space for \(N\) points and \(F\) answers
- Ex: sort points in 2-d on their \(x\) coordinate value
2-D RANGE TREES

- Binary tree of binary trees

- Sort all points along one dimension (say $x$) and store them in the leaf nodes of a balanced binary tree such as a range tree (single line)

- Each nonleaf node contains a 1-d range tree of the points in its subtrees sorted along $y$ (double lines)

- Ex:

- Actually, don’t need the 1-d range tree in $y$ at the root and at the sons of the root
SEARCHING 2-D RANGE TREES ([BX:EX],[BY:EY])

1. Search tree T for nodes BX and EX
   - find node LX with a minimum value ≥ BX
   - find node RX with a maximum value ≤ EX

2. Find their nearest common ancestor Q

3. Compute \{L_i\} and \{R_i\}, the sequences of nodes forming the paths from Q to LX and RX, respectively (including LX and RX but excluding Q)
   - LEFT(P) and RIGHT(P) are sons of P
   - MIDRANGE(P) discriminates on x coordinate value
   - RANGE_TREE(P) denotes the 1-d range tree stored at P

4. For each element in the sequences \{L_i\} and \{R_i\} do
SEARCHING 2-D RANGE TREES ([BX:EX],[BY:YE])

1. Search tree $T$ for nodes $BX$ and $EX$
   - find node $L_X$ with a minimum value $\geq BX$
   - find node $R_X$ with a maximum value $\leq EX$

2. Find their nearest common ancestor $Q$

3. Compute $\{L_i\}$ and $\{R_i\}$, the sequences of nodes forming the paths from $Q$ to $L_X$ and $R_X$, respectively (including $L_X$ and $R_X$ but excluding $Q$)
   - $\text{LEFT}(P)$ and $\text{RIGHT}(P)$ are sons of $P$
   - $\text{MIDRANGE}(P)$ discriminates on $x$ coordinate value
   - $\text{RANGE_TREE}(P)$ denotes the 1-d range tree stored at $P$

4. For each element in the sequences $\{L_i\}$ and $\{R_i\}$ do
   - if $P$ and $\text{LEFT}(P)$ are in $\{L_i\}$, then look for $[BY, YE]$ in $\text{RANGE_TREE}(\text{RIGHT}(P))$
SEARCHING 2-D RANGE TREES (\([B_X:E_X],[B_Y:E_Y]\))

1. Search tree \(T\) for nodes \(B_X\) and \(E_X\)
   - find node \(L_X\) with a minimum value \(\geq B_X\)
   - find node \(R_X\) with a maximum value \(\leq E_X\)

2. Find their nearest common ancestor \(Q\)

3. Compute \(\{L_i\}\) and \(\{R_i\}\), the sequences of nodes forming the paths from \(Q\) to \(L_X\) and \(R_X\), respectively (including \(L_X\) and \(R_X\) but excluding \(Q\))
   - \(\text{LEFT}(P)\) and \(\text{RIGHT}(P)\) are sons of \(P\)
   - \(\text{MIDRANGE}(P)\) discriminates on \(x\) coordinate value
   - \(\text{RANGE\_TREE}(P)\) denotes the 1-d range tree stored at \(P\)

4. For each element in the sequences \(\{L_i\}\) and \(\{R_i\}\) do
   - if \(P\) and \(\text{LEFT}(P)\) are in \(\{L_i\}\),
     then look for \([B_Y,E_Y]\) in \(\text{RANGE\_TREE}(\text{RIGHT}(P))\)
   - if \(P\) and \(\text{RIGHT}(P)\) are in \(\{R_i\}\), then look for \([B_Y,E_Y]\) in \(\text{RANGE\_TREE}(\text{LEFT}(P))\)

Copyright © 1998 by Hanan Samet
SEARCHING 2-D RANGE TREES ([BX:EX],[BY:EY])

1. Search tree $T$ for nodes $B_X$ and $E_X$
   - find node $L_X$ with a minimum value $\geq B_X$
   - find node $R_X$ with a maximum value $\leq E_X$

2. Find their nearest common ancestor $Q$

3. Compute $\{L_i\}$ and $\{R_i\}$, the sequences of nodes forming the paths from $Q$ to $L_X$ and $R_X$, respectively (including $L_X$ and $R_X$ but excluding $Q$)
   - $\text{LEFT}(P)$ and $\text{RIGHT}(P)$ are sons of $P$
   - $\text{MIDRANGE}(P)$ discriminates on $x$ coordinate value
   - $\text{RANGE}_{\text{TREE}}(P)$ denotes the 1-d range tree stored at $P$

4. For each element in the sequences $\{L_i\}$ and $\{R_i\}$ do
   - if $P$ and $\text{LEFT}(P)$ are in $\{L_i\}$, then look for $[B_Y,E_Y]$ in $\text{RANGE}_{\text{TREE}}(\text{RIGHT}(P))$
   - if $P$ and $\text{RIGHT}(P)$ are in $\{R_i\}$, then look for $[B_Y,E_Y]$ in $\text{RANGE}_{\text{TREE}}(\text{LEFT}(P))$

5. Check if $L_X$ and $R_X$ are in $([B_X:E_X],[B_Y:E_Y])$
SEARCHING 2-D RANGE TREES ([BX:EX],[BY:EY])

1. Search tree $T$ for nodes $BX$ and $EX$
   - find node $LX$ with a minimum value $\geq BX$
   - find node $RX$ with a maximum value $\leq EX$

2. Find their nearest common ancestor $Q$

3. Compute $\{L_i\}$ and $\{R_i\}$, the sequences of nodes forming
   the paths from $Q$ to $LX$ and $RX$, respectively (including $LX$
   and $RX$ but excluding $Q$)
   - LEFT($P$) and RIGHT($P$) are sons of $P$
   - MIDRANGE($P$) discriminates on $x$ coordinate value
   - RANGE_TREE($P$) denotes the 1-d range tree stored at $P$

4. For each element in the sequences
   $\{L_i\}$ and $\{R_i\}$ do
   - if $P$ and LEFT($P$) are in $\{L_i\}$,
     then look for $[BY, EY]$ in
     RANGE_TREE(RIGHT($P$))
   - if $P$ and RIGHT($P$) are in
     $\{R_i\}$, then look for $[BY, EY]$ in
     RANGE_TREE(LEFT($P$))

5. Check if $LX$ and $RX$ are in
   $([BX:EX],[BY:EY])$
   - Total $O(\log_2^2 N + F)$
     time to search and
     $O(N \cdot \log_2 N)$ space
     and time to build for
     $N$ points and $F$
     answers

Copyright © 1998 by Hanan Samet
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

- Find all points in $([25:85],[8:16])$
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

• Find all points in ([25:85],[8:16])

1. Find nearest common ancestor — i.e., A

Copyright © 1998 by Hanan Samet
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

• Find all points in \([25:85],[8:16]\))

1. Find nearest common ancestor — i.e., A

2. Find paths to \(L_X=25\) and \(R_X=85\)
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

• Find all points in ([25:85],[8:16])

1. Find nearest common ancestor — i.e., A

2. Find paths to \( L_X = 25 \) and \( R_X = 85 \)

3. Look in subtrees
   • B and B’s left son D are in path, so search range tree of B’s right son E and report (52,10)
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

- Find all points in ([25:85],[8:16])

1. Find nearest common ancestor — i.e., A
2. Find paths to LX=25 and RX=85
3. Look in subtrees
   - B and B’s left son D are in path, so search range tree of B’s right son E and report (52,10)
   - C and C’s right son G are in path, so search range tree of C’s left son F and report none

Copyright © 1998 by Hanan Samet
EXAMPLE OF SEARCH IN A 2-D RANGE TREE

• Find all points in ([25:85],[8:16])

1. Find nearest common ancestor — i.e., A

2. Find paths to $L_x=25$ and $R_x=85$

3. Look in subtrees
   • B and B’s left son D are in path, so search range tree of
     B’s right son E and report (52,10)
   • C and C’s right son G are in path, so search range tree
     of C’s left son F and report none

4. Check boundaries of x range (i.e., (27,35) and (85,15))
   and report (85,15)
PRIORITY SEARCH TREES

- Sort all points by their \( x \) coordinate value and store them in the leaf nodes of a balanced binary tree (i.e., a range tree)

- Starting at the root, each node contains the point in its subtree with the maximum value for its \( y \) coordinate that has not been stored at a shallower depth in the tree; if no such node exists, then node is empty

- \( O(N) \) space and \( O(N \cdot \log_2 N) \) time to build for \( N \) points

- Result: range tree in \( x \) and heap (i.e., priority queue) in \( y \)

- Ex:

Good for semi-infinite ranges — i.e., \([\text{BX:EX}],[\text{BY:}\infty]\)

Can only perform a 2-d range query if find \([\text{BX:EX}],[\text{BY:}\infty]\) and discard all points \((x,y)\) such that \( y > EY \)

No need to link leaf nodes unless search for all points in range of \( x \) coordinate values

Copyright © 1998 by Hanan Samet
SEMI-INFINITE RANGE QUERY ON A PRIORITY SEARCH TREE ([BX:EX],[BY:∞])

• Procedure
  1. Descend tree looking for the nearest common ancestor of BX and EX — i.e., Q
     • associated with each examined node T is a point P
     • exit if P does not exist as all points in the subtrees have been examined and/or reported
     • exit if P_y < BY as P is point with maximum y coordinate value in T
     • otherwise, output P if P_x is in [BX:EX]
  2. Once Q has been found, process left and right subtrees applying the tests above to their root nodes T
     • T in left (right) subtree of Q:
       a. check if BX (EX) in LEFT(T) (RIGHT(T))
       b. yes: all points in RIGHT(T) (LEFT(T)) are in x range
          • check if in y range
          • recursively apply to LEFT(T) (RIGHT(T))
       c. no: recursively apply to RIGHT(T) (LEFT(T))

• O(\log_2 N + F) time to search for N points and F answers

• Ex:
SEMI-INFINITE RANGE QUERY ON A PRIORITY SEARCH TREE ([BX:EX],[BY:∞])

• Procedure
  1. Descend tree looking for the nearest common ancestor of BX and EX — i.e., Q
     • associated with each examined node T is a point P
     • exit if P does not exist as all points in the subtrees have been examined and/or reported
     • exit if P_y < BY as P is point with maximum y coordinate value in T
     • otherwise, output P if P_x is in [BX:EX]
  2. Once Q has been found, process left and right subtrees applying the tests above to their root nodes T
     • T in left (right) subtree of Q:
       a. check if BX (EX) in LEFT(T) (RIGHT(T))
       b. yes: all points in RIGHT(T) (LEFT(T)) are in x range
          • check if in y range
          • recursively apply to LEFT(T) (RIGHT(T))
       c. no: recursively apply to RIGHT(T) (LEFT(T))

• O(log_2 N + F) time to search for N points and F answers

• Ex: Find all points in ([35:80],[50:∞])
EXAMPLE OF A SEARCH IN A PRIORITY SEARCH TREE

- Find all points in \([35:83],[50:\infty]\)
EXAMPLE OF A SEARCH IN A PRIORITY SEARCH TREE

- Find all points in ([35:83],[50:∞])

1. Find nearest common ancestor — i.e., A
   - output Toronto (62,77) since 62 is in [35:80] and 77 ≥ 50
EXAMPLE OF A SEARCH IN A PRIORITY SEARCH TREE

- Find all points in ([35:83],[50:∞])

1. Find nearest common ancestor — i.e., A
   - output Toronto (62,77) since 62 is in [35:80] and 77 ≥ 50

2. Process left subtree of A (i.e., B)
   - cease processing as 45 < 50
EXAMPLE OF A SEARCH IN A PRIORITY SEARCH TREE

- Find all points in ([35:80],[50:∞])

1. Find nearest common ancestor — i.e., A
   • output Toronto (62,77) since 62 is in [35:80] and 77 ≥ 50

2. Process left subtree of A (i.e., B)
   • cease processing as 45 < 50

3. Process right subtree of A (i.e., C)
   • output (82,65) as 65 ≥ 50 and 82 is in [35:83]
EXAMPLE OF A SEARCH IN A PRIORITY SEARCH TREE

• Find all points in ([35:83],[50:∞])

1. Find nearest common ancestor — i.e., A
   • output Toronto (62,77) since 62 is in [35:80] and 77 ≥ 50

2. Process left subtree of A (i.e., B)
   • cease processing as 45 < 50

3. Process right subtree of A (i.e., C)
   • output (82,65) as 65 ≥ 50 and 82 is in [35:83]

4. Examine midrange value of C which is 84 and descend left subtree of C (i.e., F)
   • cease processing since no point is associated with F meaning all nodes in the subtree have been examined
RANGE PRIORITY TREES

- Variation on priority search tree
- Inverse priority search tree: heap node stores point with minimum y coordinate value that has not been stored in a shallower depth in the tree (instead of maximum)

- Structure
  1. sort all points by their y coordinate value and store in leaf of a balanced binary tree such as range tree (single lines)
     - no need to link leaf nodes unless search for all points in range of x coordinate values
  2. nonleaf node left sons of their father contains a priority search tree of points in subtree (double lines)
  3. nonleaf node right sons of their father contains an inverse priority search tree of points in subtree (double lines)

- $O(N \cdot \log_2 N)$ space and time to build for $N$ points

- Ex:
SEARCHING A RANGE PRIORITY TREE ([BX:EX],[BY:Ey])

• Procedure
  1. find nearest common ancestor of BY and EY — i.e., Q
SEARCHING A RANGE PRIORITY TREE \( ([BX:EX],[BY:YE]) \)

- **Procedure**
  
  1. find nearest common ancestor of BY and EY — i.e., Q
  
  2. all points in \( \text{LEFT}(Q) \) have \( y \) coordinate values \(<EY\)
     - want to retrieve just the ones \( \geq BY\)
     - find them with \( ([BX:EX],[BY:\infty]) \) on priority tree of \( \text{LEFT}(Q) \)
     - priority tree is good for retrieving all points with a specific lower bound as it stores an upper bound and hence irrelevant values can be easily pruned

```
Copyright © 1998 by Hanan Samet
```
SEARCHING A RANGE PRIORITY TREE ([BX:EX],[BY:YE])

- **Procedure**
  1. find nearest common ancestor of BY and EY — i.e., Q
  2. all points in \( \text{LEFT}(Q) \) have y coordinate values \( \leq EY \)
     - want to retrieve just the ones \( \geq BY \)
     - find them with \([BX:EX],[BY:\infty]\) on priority tree of \( \text{LEFT}(Q) \)
     - priority tree is good for retrieving all points with a specific lower bound as it stores an upper bound and hence irrelevant values can be easily pruned
  3. all points in \( \text{RIGHT}(Q) \) have y coordinate values \( \geq BY \)
     - want to retrieve just the ones \( \leq EY \)
     - find them with \([BX:EX],[-\infty:YE]\) on the inverse priority tree of \( \text{RIGHT}(Q) \)
     - inverse priority tree is good for retrieving all points with a specific upper bound as it stores a lower bound and hence irrelevant values can be easily pruned
SEARCHING A RANGE PRIORITY TREE ([BX:EX], [BY:YE])

• Procedure

1. find nearest common ancestor of BY and YE — i.e., Q

2. all points in LEFT(Q) have y coordinate values < YE
   • want to retrieve just the ones ≥ BY
   • find them with ([BX:EX], [BY:∞]) on priority tree of LEFT(Q)
   • priority tree is good for retrieving all points with a specific lower bound as it stores an upper bound and hence irrelevant values can be easily pruned

3. all points in RIGHT(Q) have y coordinate values > BY
   • want to retrieve just the ones ≤ YE
   • find them with ([BX:EX], [–∞:YE]) on the inverse priority tree of RIGHT(Q)
   • inverse priority tree is good for retrieving all points with a specific upper bound as it stores a lower bound and hence irrelevant values can be easily pruned

• \( O(\log_2 N + F) \) time to search for \( N \) points and \( F \) answers
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

- Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A

2. Search for ([25:60],[15:∞]) in priority tree hanging from left son of A — i.e., B (all with \( y \leq 45 \) since a range tree in \( y \) and in left subtree of a node with \( y \) midrange value of 39)
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A
2. Search for ([25:60],[15:\infty]) in priority tree hanging from left son of A — i.e., B (all with $y \leq 45$ since a range tree in $y$ and in left subtree of a node with $y$ midrange value of 39)
   • output (27,35) as in range

Copyright © 1998 by Hanan Samet
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A
2. Search for ([25:60],[15:¥]) in priority tree hanging from left son of A — i.e., B (all with $y \leq 45$ since a range tree in $y$ and in left subtree of a node with $y$ midrange value of 39)
   • output (27,35) as in range
   • reject left subtree as $10 < $ lower limit of $y$ range
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

- Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A

2. Search for ([25:60],[15:∞]) in priority tree hanging from left son of A — i.e., B (all with y≤45 since a range tree in y and in left subtree of a node with y midrange value of 39)
   - output (27,35) as in range
   - reject left subtree as 10 < lower limit of y range
   - reject items in right subtree as out of x range
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in \([25:60],[15:45]\)

1. Find nearest common ancestor of 15 and 45 — i.e., A

2. Search for \([25:60],[15:∞]\) in priority tree hanging from left son of A — i.e., B (all with \(y ≤ 45\) since a range tree in \(y\) and in left subtree of a node with \(y\) midrange value of 39)
   • output (27,35) as in range
   • reject left subtree as 10 < lower limit of \(y\) range
   • reject items in right subtree as out of \(x\) range

3. Search for \([25:60],[–∞:45]\) in inverse priority tree hanging from right son of A — i.e., C (all with \(y ≥ 15\) since in right subtree of a node with \(y\) midrange value of 39)
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A

2. Search for ([25:60],[15:\infty]) in priority tree hanging from left son of A — i.e., B (all with $y\leq 45$ since a range tree in $y$ and in left subtree of a node with $y$ midrange value of 39)
   - output (27,35) as in range
   - reject left subtree as $10 < $ lower limit of $y$ range
   - reject items in right subtree as out of $x$ range

3. Search for ([25:60],[-\infty:45]) in inverse priority tree hanging from right son of A — i.e., C (all with $y\geq 15$ since in right subtree of a node with $y$ midrange value of 39)
   - output (35,42) as in range
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in ([25:60],[15:45])

1. Find nearest common ancestor of 15 and 45 — i.e., A

2. Search for ([25:60],[15:\infty]) in priority tree hanging from left son of A — i.e., B (all with $y \leq 45$ since a range tree in $y$ and in left subtree of a node with $y$ midrange value of 39)
   • output (27,35) as in range
   • reject left subtree as 10 < lower limit of $y$ range
   • reject items in right subtree as out of $x$ range

3. Search for ([25:60],[\infty:45]) in inverse priority tree hanging from right son of A — i.e., C (all with $y \geq 15$ since in right subtree of a node with $y$ midrange value of 39)
   • output (35,42) as in range
   • reject unreported items in left subtree as out of $x$ range
EXAMPLE OF A SEARCH IN A RANGE PRIORITY TREE

• Find all points in \([25:60],[15:45]\)