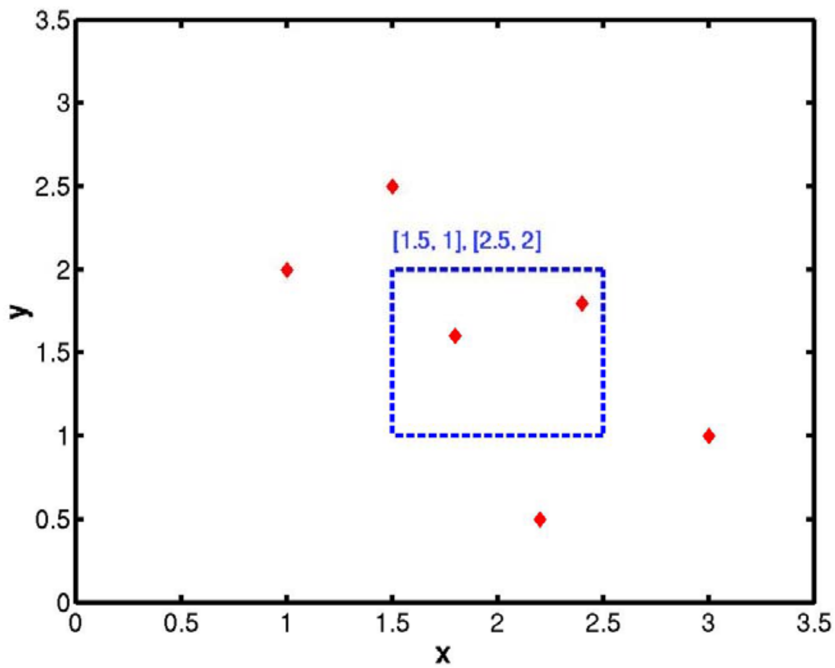
COP 5405: Advanced Algorithms

Fall 2006

Lecture 18

Orthogonal Range Search using Range trees

Data: A set P of n points in d-dimensional space. Assume that no two points have identical values for the same coordinate.
Query: $[x_1, x_2, \ldots, x_d], [x'_1, x'_2, \ldots, x'_d]$.
Output: All points in P that fall within the d-dimensional rectangle $[x_1, x_2, \ldots, x_d] \times [x'_1, x'_2, \ldots, x'_d]$.



1-Dimensional range search

- Organize the data as a balanced binary search tree with all points at the leaves.
- The internal nodes contain the value of the largest element in the left subtree.
- This can be done in $O(n \log n)$ time.

<table>
<tr><th>FindSplitNode(T, x, x'), $x \le x'$</th><th>1-D RangeQuery(T, x, x')</th></tr>
<tr><td>$v \leftarrow$ root(T)
while v is not a leaf and ($x' \le x_v$ or $x > x_v$)
   if $x' \le x_v$ then
      $v \leftarrow$ leftchild(v)
   else
      $v \leftarrow$ rightchild(v)
return v</td><td>$v_{split} \leftarrow$ FindSplitNode(T, x, x')
if v_{split} is a leaf
   check if v_{split} is in the range, and report it if it is
else
   $v \leftarrow$ leftchild(v_{split}) /* search left subtree */
   while v is not a leaf
   if $x \le x_v$ then
      ReportSubtree(rightchild(v))
      $v \leftarrow$ leftchild(v)
   else
      $v \leftarrow$ rightchild(v)
   Report v if it is in the range
   /* Similar search in the right subtree */</td></tr>
</table>

Lemma 5.1: The above algorithm reports exactly those points that lie in the query region.

Lemma 5.2: P can be stored in a balanced BST taking $O(n)$ storage and $O(n \log n)$ time to construct, such that the points in the query range can be reported in $O(\log n + k)$ time, where k is the number of points in the range. Note that the time complexity is output sensitive.

Range trees in 2-D

- The main tree is a balanced BST on the x coordinates, built as in the 1-D case. The points are stored in the leaves.
- The interior nodes contain an additional pointer to another balanced BST, which contains $P(v)$ in its leaves, where $P(v)$ is the set of points in the leaves of the subtree rooted at v. This associated structure of v is built on the y coordinates of $P(v)$.
- This can be constructed in $O(n \log n)$ time and uses $O(n \log n)$ storage.

<table>
<tr><th>2-D RangeQuery(T, x, x', y, y')</th></tr>
<tr><td>$v_{split} \leftarrow$ FindSplitNode(T, x, x')
if v_{split} is a leaf
   check if v_{split} is in the range, and report it if it is
else
   $v \leftarrow$ leftchild(v_{split}) /* search left subtree */
   while v is not a leaf
   if $x \le x_v$ then
      1-D RangeQuery($T_{associated}$(rightchild(v)), y, y'))
      $v \leftarrow$ leftchild(v)
   else
      $v \leftarrow$ rightchild(v)
   Report v if it is in the range
   /* Similar search in the right subtree */</td></tr>
</table>

Lemma 5.7: Time complexity of the search is $O(\log^2 n + k)$ time to report k points.
Proof: Time spent in each 1-D RangeQuery call is $O(\log n + k_v)$, if there are k_v points reported under v. Total time = $\Sigma_{v \text{ in 1-D calls}} O(\log n + k_v)$. Since there are $O(\log n)$ vs, with points in them being distinct, the total time is $O(\log^2 n + k)$.