Range Search: The Scenario

- Input $Q$ is an orthogonal range (for the time being)
  - Output the points in $Q$
  - Output the number of points
- Can be used (how?) to help solve the problem “Find the nearest McDonald within a 20 mile radius of a query $p$”

- Straightforward solution: Check if each of $n$ points are inside $Q$
- Can we preprocess the points so that the query can be answered more efficiently?
An Idea That Does Not Help

Assume points stored in a kd-tree

- Locating the lower left corner of $q$ in kd-tree does not help too much. We fall off $k$ and the path taken ($a, h, i, j, k$) contains some relevant nodes and some irrelevant nodes.
Alternate Idea

• Instead, let $C$ is the region associated with a node.

• If the query $Q$ rectangle
  – Is disjoint from $C$
  – Completely contains $C$

• We do not need to process the possibly many points within $C$

• In the example, this situation arises for the region associated with node $b$

• What should we do if the query $Q$ does not satisfy the two conditions?
Smaller Easier Queries

- \( \text{Q.contains (Point } p \text{)} \)  \((p: \text{point associated with a kdtree node})\)
- \( \text{Q.contains (Rectangle } C \text{)} \)  \((C: \text{rectangle associated with a kdtree node})\)
- \( \text{Q.isDisjoint (Rectangle } C \text{)} \)

- Also need to be able to split the rectangle \( C \)
Range Search: Code

Set searchKDTree(KDNode n, Query Q, int cd) {
    if (n = null) return {};
    else if (Q.isDisjoint(n)) return {};
    else if (Q.contains(n)) return AllPoints(n);
    else {
        Set middle = Q.contains(n.data);
        Set left = searchKDTree(n.left, Q, cd+1);
        Set right = searchKDTree(n.right, Q, cd+1);
        return union(union(left, right), middle)
    }
}
Example

- Search takes $O(\sqrt{n})$ time if tree is balanced.
- A range \textit{stabs} a node if it overlaps the rectangle associated with the node without being contained in the cell.
Range Search: Proof

- Running time is a function of how many nodes get stabbed
- Any vertical or horizontal line stabs at most $O(\sqrt{n})$ nodes
  - Consider a vertical line $x = x_0$.
    - If the cutting dimension is $x$, then the line stabs either the left child, or the right child but not both.
    - If it fails to stab a child, it fails to stab the descendents of that child.
  - Otherwise it stabs both children.
  - After descending to the grandchildren level, at most four more nodes are stabbed.
  - In the worst case, root is stabbed, 4 nodes are stabbed at the grandchildren level, 8 nodes are stabbed at level 6
  - At most $2^i$ nodes at level $2i$