Spatial Data

- info about objects that includes location

queries:
- find all objects with locations in a given region within radius r of pt p
- find the object closest to a location
1 Dimension

Find in range $[a, b]$

Find closest to $p \Rightarrow D$

Search for $p$

Return closest of

1) Inorder predecessor
2) Inorder successor

Next highest value in tree

Leftmost in right subtree if there is a right subtree otherwise, most recent ancestor you went left from $O(\log n)$

Output-sensitive running time $O(\log n + k)$

$k$ of things in output
kd-Tree

k=2 for (x,y) coords

kd tree: nodes at level l obey BST order property for

dimension 0 if l is a multiple of k

dimension 1 if l is one more than multiple of k

dimension 2 if l is 2 more than ...

: x at even levels

: y at odd levels
Adding to kd-Tree

Adding to a kd-tree:

1. Start at the root of the tree.
2. For each dimension $d$ from 0 to $k-1$:
   a. If the current data point $p$ is smaller than the current node $curr$'s data point in dimension $d$, move to the left child.
   b. Otherwise, move to the right child.
3. If the current node is NULL, add a new node as appropriate.

Time complexity: $O(\log n)$ if balanced.
Removing From kd-Tree

```plaintext
find_min(n, d)
  if n == NULL return NULL
  if n's cut is dum d
    if n -> left != NULL
      return find_min(n -> left, d)
    else return n
  else return min(n, find_min(n -> left, d), find_min(n -> right))
```

The time complexity is \(O(\sqrt{n})\).
Range Query in kd-Tree

\[ \text{range}(n, r, \emptyset, d) \]
\[
\text{if } n = \text{null or } r \cap b = \emptyset \\
\text{return}
\]
\[
\text{if } n \rightarrow \text{data in } r, \text{ process } n \rightarrow \text{data[\text{dim}]}
\]
\[
\text{range } (n \rightarrow \text{left}, r, b \cap \text{dim} \leq (d+1)/2k)
\]
\[
\text{range } (n \rightarrow \text{right}, r, b \cap \text{dim} \geq n \rightarrow \text{data[\text{dim}]}, (d+1)/2k)
\]
\[
O(\sqrt{n} + k), \text{ if balanced}
\]