Binary Search Tree Review

• Invariants:
  – Left nodes are always < current node.
  – Right nodes are always > current node value.

• Therefore:
  – At each node, we can split the search space in half, enabling \(O(\log n)\) search times.

Range Search

• Regular Search:

```c
node * query(node * root, int target_val){
    if (target_val == root->value)
        return root;
    if (target_val < root->value)
        return query(root->left, target_val);
    if (target_val > root->value)
        return query(root->right, target_val);
}
```
Range Search

• In order to be able to query a range, what should we do?
  – We should return a list of matches
  – Recursive or Iterative?

Range Search

• In order to be able to query a range, what should we do?
  – What should we return?

Range Search

• In order to be able to query a range, what should we do?

Range Search

• In order to be able to query a range, what should we do?
  – We should return a list of matches
  – Recursive
Range Search – Recursive Definition

• Take 5 minutes to develop a recursive definition.
• On a node of value x:
  – 1. What if min < max < x?
  – 2. What if min <= x <= max
  – 3. What if x < min < max?

K-D Trees Review

• Like Binary Search Trees except:
  – Each node has multiple keys.
  – At each branch, you consider a different key to split on.

  • Due to similarity, you can easily adapt the algorithms for querying BSTs for use in K-D trees.
Multi-dimensional Search

- BST version:
  - 1. if goal < current node value, return left
  - 2. if goal == current node value, return the node
  - 3. if goal > current node value, return right

- How to modify this for K-D trees?

Multi-dimensional Search

- K-D Tree version:
  - 1. if \[ \text{goal}[\text{dim}] < \text{current node value}[\text{dim}] \], \text{dim} = (\text{dim}+1) \text{ mod } \text{total_dim}, return left
  - 2. if \[ \text{goal}[\text{dim}] == \text{current node value}[\text{dim}] \], return the node
  - 3. if \[ \text{goal}[\text{dim}] > \text{current node value}[\text{dim}] \], \text{dim} = (\text{dim}+1) \text{ mod } \text{total_dim}, return right

- How to modify this for K-D trees?
  - Keep track of which dimension you are searching.
  - As you keep searching down, be sure to keep track of which dimension the nodes are being split on.

Nearest Neighbor Search

- Goal:
  - Given a point in a K-dimensional space, find the closest point stored in a data structure.

Nearest Neighbor Search

- Goal:
  - Given a point in a K-dimensional space, find the closest point stored in a data structure.

- Use a K-D Tree!
Nearest Neighbor Search

- **Steps:**
  - 1. Start with root node and use depth-first search to find where you would insert the node if you were inserting it. Save this as current best.
  - 2. Go up one node. If it’s better than closest best, it becomes closest best.
  - 3. Check whether there could be any points on the other side of the splitting plane by checking the distance between the target node and the splitting plane.
  - 4. Repeat steps 2-3 until you are at the root node.

Nearest Neighbor Search

- **Given a K-D Tree with the points:**
  - \((1, 1) (1, 2) (5, 1) (4, 3) (2, 4)\)
  - Find the point closest to \((3, 2)\) (manhatten space)
### Nearest Neighbor Search

- 1. Depth-first Search to E.
- 2. E is the best distance = 3.
- 3. Go back to C:
  - C is not closer (3 away). No nodes below C.
- 4. Go back to A.
  - A is not closer. Check if other side of A could be closer. \( \text{Abs(Goal.x - A.x)} = 1 < 3 \). Check other side.
- 5. DFS to D. D is closer than E (2 away)

---

### Nearest Neighbor Search

- 1. Depth-first Search to E.
- 2. E is the best distance = 3.
- 3. Go back to C:
  - C is not closer (3 away). No nodes below C.
- 4. Go back to A.
  - A is not closer. Check if other side of A could be closer. \( \text{Abs(Goal.x - A.x)} = 1 < 3 \). Check other side.
- 5. DFS to D. D is closer than E (2 away)
- 6. Go back to B.
  - B is not closer. No nodes below B.
Nearest Neighbor Search

• 4. Go back to A.
  – A is not closer. Check if other side of A could be closer. \( \text{Abs}(\text{Goal}.x - A.x) = 1 < 3 \). Check other side.
• 5. DFS to D. D is closer than E (2 away)
• 6. Go back to B.
  – B is not closer. No nodes below B.
• 7. Go back to the A. Checked both sub-trees, therefore done.