KD TREES

CS16: Introduction to Data Structures & Algorithms
KD-Trees

• Have you clicked on a random point on google maps, and used the ‘search nearby’ feature ?
• Have you searched for things near your ‘current location’ on yelp ?
• Have you wondered how these things are searched for ?
1D-Tree Example

2 8 10 16 22 30 40 42
1D-Tree Example

```
22
 /   
10   40
 /     /
8     30
 /       /
2       42
```

2D-Tree Example

• Now let's sort something a little more complicated, points in the Cartesian plane.

• A(10, 12), B(11, 9), C(3,4), D(5,3), E(7,8), F(1, 5)
2D-Tree Example

• Sort list in the 1st dimension (x)
  • F, C, D, E, A, B

• Center element becomes new node.

• Make two new lists, everything in the left list is less than or equal to the center, everything in the right list is greater than the center.

  Left list: F, C, D  Right list: A, B
2D-Tree Example

• We now sort each list in the next dimension (y).
  
  F, C, D -> D, C, F  
  A, B -> B, A

• Again we take the center element to be the new node

• Make our separate lists again.
  
  D  F  B
2D-Tree Example

• Sort our lists; in the first dimension this time
  
  D       F       B

• Make our new nodes

Diagram:

- E
  - C
    - D
  - A
    - F
    - B
2D-Tree Example: Nearest Neighbor

- A(10,12)
- B(11,9)
- C(3,4)
- D(5,3)
- E(7,8)
- F(1,5)
- G(1,3.5)
2D-Tree Example: Nearest Neighbor
2D-Tree Example: Nearest Neighbor

X

Y

A(10, 12)

B(11, 9)

E(7, 8)

C(3,4)

D(5,3)

F(1,5)

G(1, 3.5)

X

E

Y

C

D

F

B

Y

A
NNS: Pseudocode

• Move down the tree recursively starting with the root. Go left or right depending on whether the point is less than or greater than the current node in the current split dimension.
• Once you reach a leaf node, save it as the "current best"
• You have essentially found the closest neighbour in the plane section containing your reference point.
• But are you done yet?
• …..
NNS: Pseudocode

• Not quite. We’ve only checked that portion of the splitting plane that contains the reference point, what if there is another point closer but in some other section of the plane?
• How do we find that?
• Unwind the recursion of the tree, performing the following steps at each node:
  • If the current node is closer than the current best, then it becomes the current best. (Euclidean distance in the plane is used for comparison)
  • Check whether there could be any points on the other side of the splitting plane that are closer to the search point than the current best.
  • How would you check that?
NNS: Pseudocode

- You can draw a circle around the reference point as the centre with the radius as your current minimum distance.
- Then, as you can see, the circle intersects some other splitting planes …
- That means that points in those planes could be potential candidates for the minimum distance.
- Therefore, we move down the relevant branch of the tree from the current node looking for points that might be nearer …
- If the circle did not intersect any other splitting planes, we continue up the tree checking the same until we reach the root.
- Then we’ve found our nearest neighbour.
3D-Tree example