What Happened?

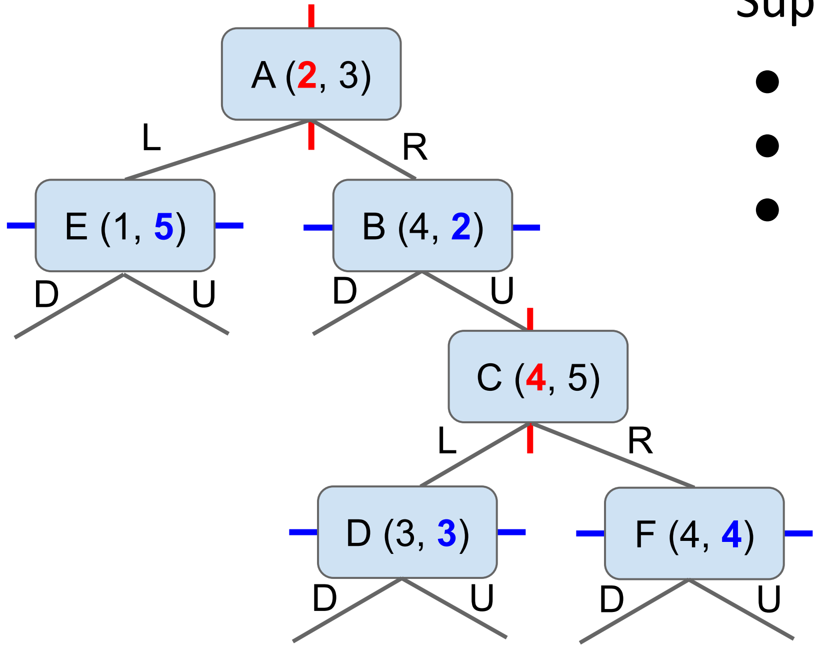
If this looks different than you remember, I recorded a new version of the video for these slides because I made things more complicated than they needed to be!

- You can still use the approach from the old video, but this one is IMO way easier to understand.

In case you're curious, the key change is that now we think of each point as having a "good side" and a "bad side".

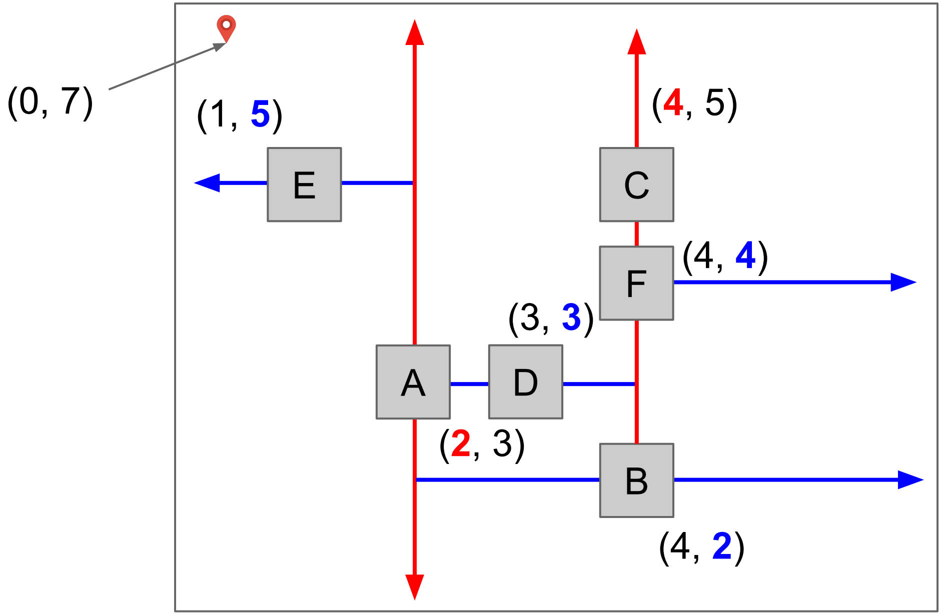
- You always explore the good side [the old version performed an unnecessary check!]
- You only explore the bad side if there's a chance that it could contain something better.

K-d Nearest Demo

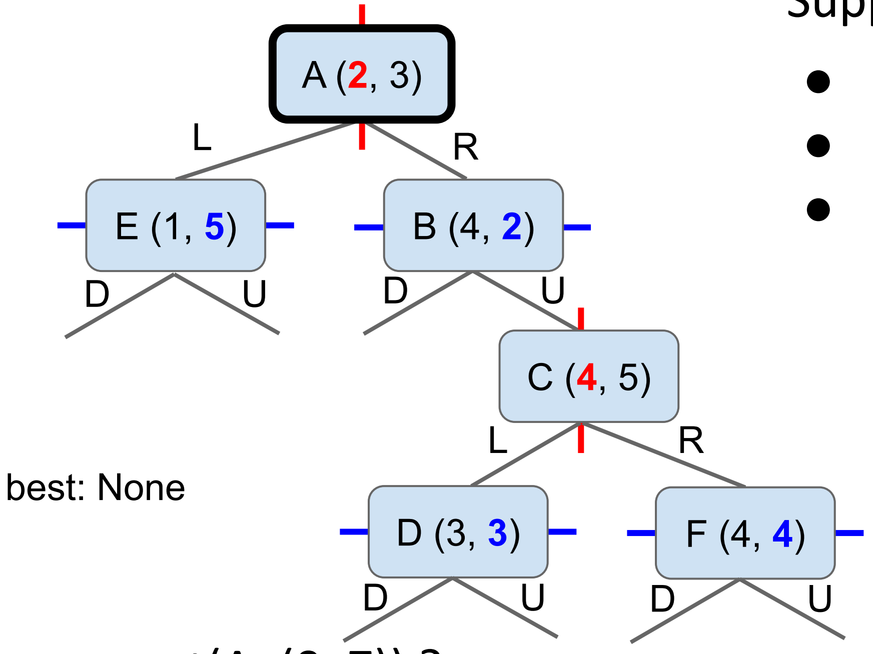


Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



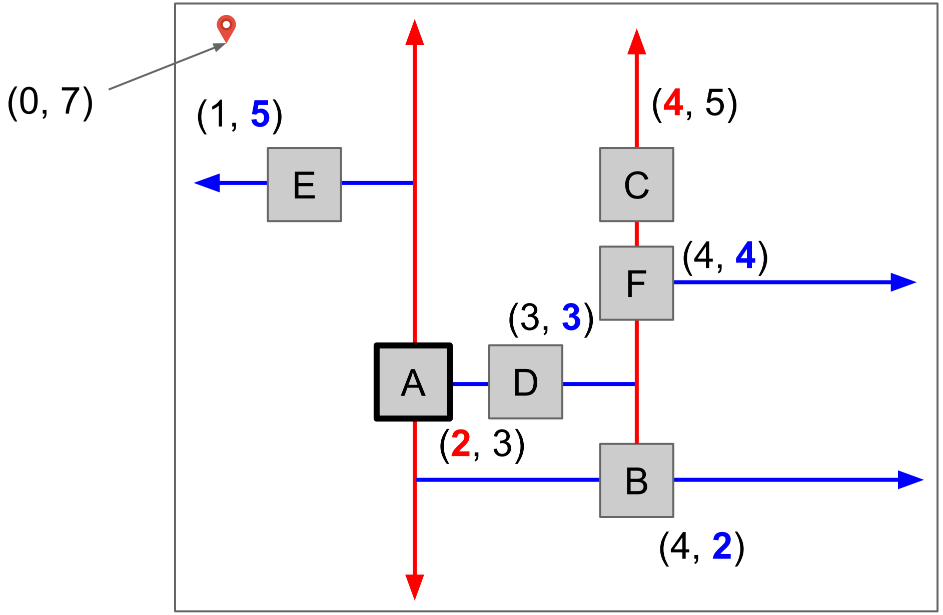
K-d Nearest Demo



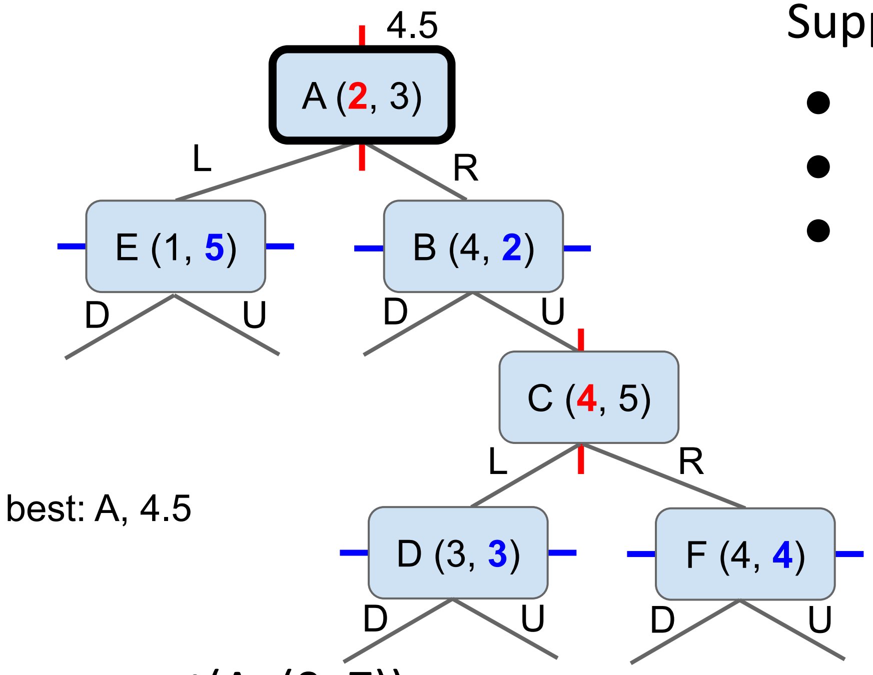
nearest(A, (0, 7)) ?

Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.

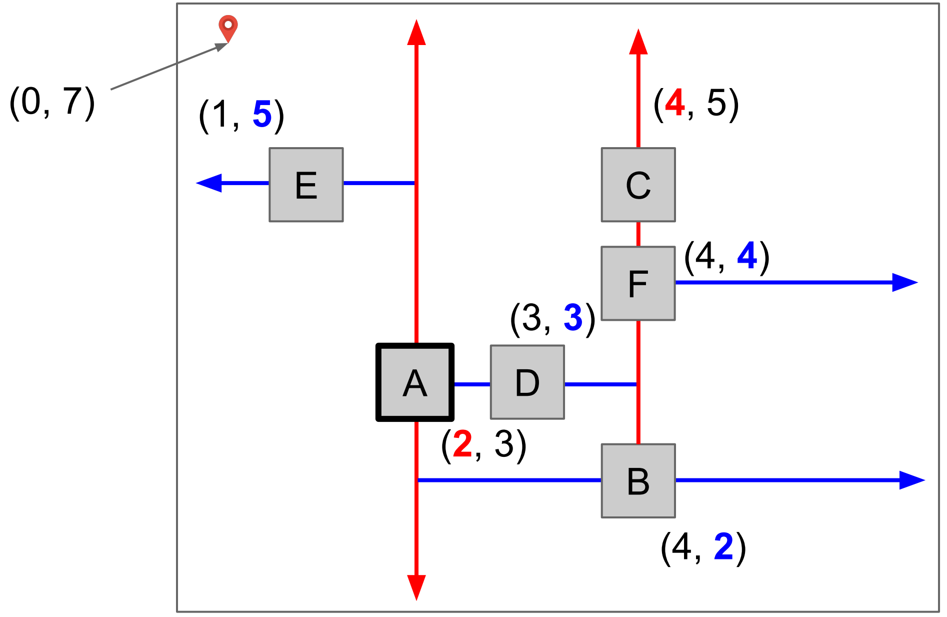


K-d Nearest Demo



Suppose we have the k-d tree shown.

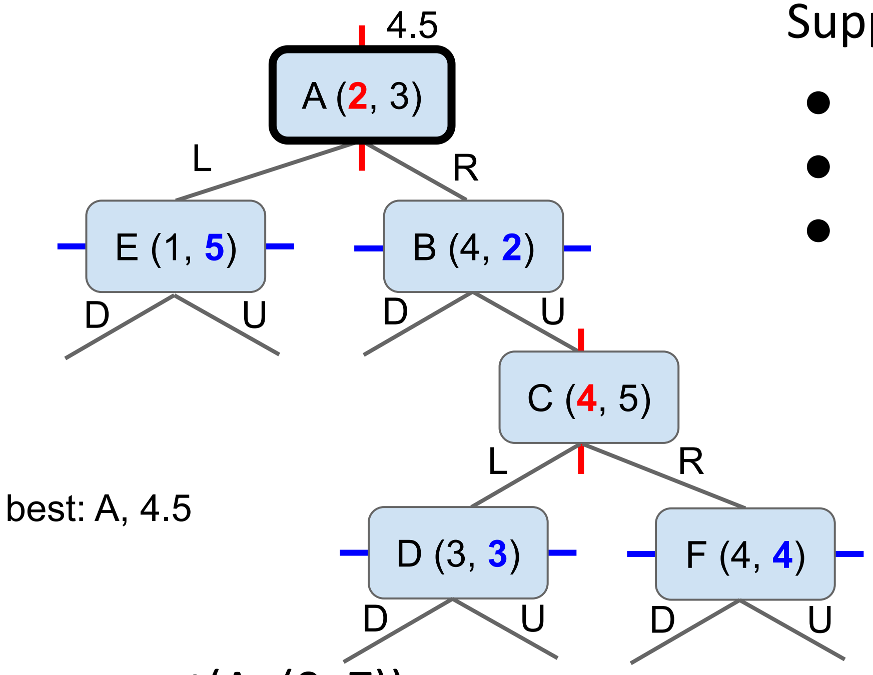
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(A, (0, 7))

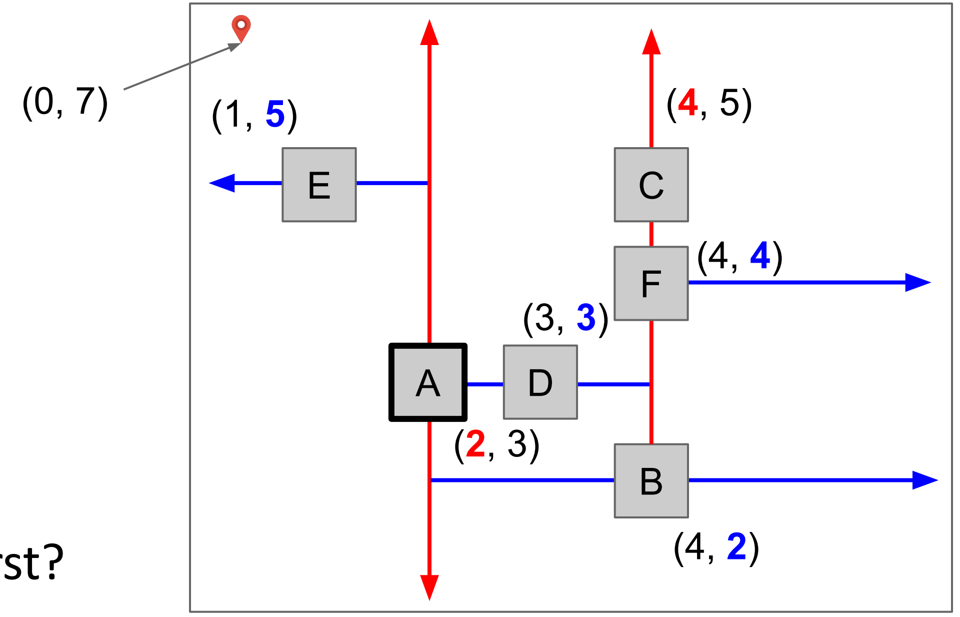
- dist(A) is sqrt(4+16) = 4.5.

K-d Nearest Demo



Suppose we have the k-d tree shown.

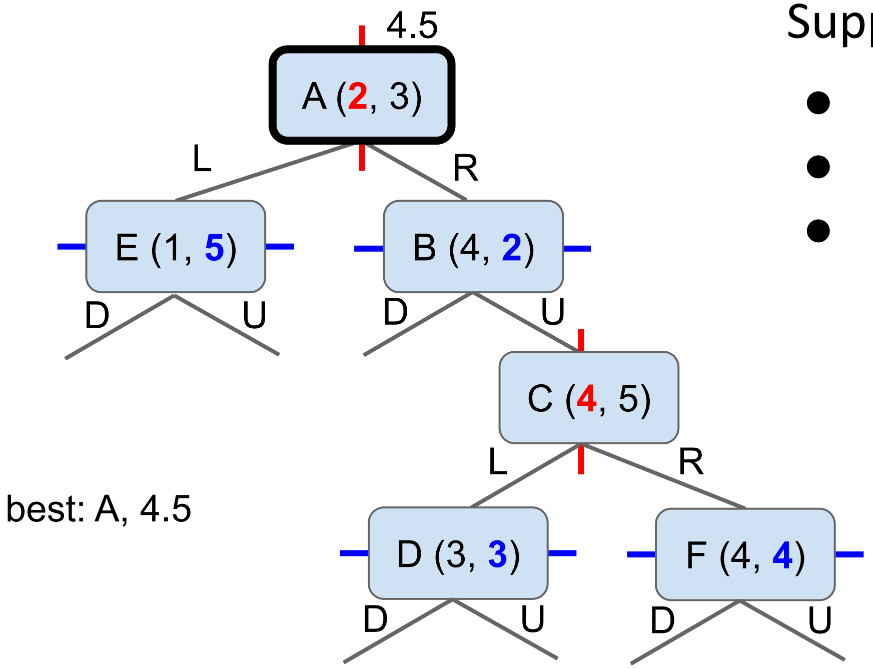
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(A, (0, 7))

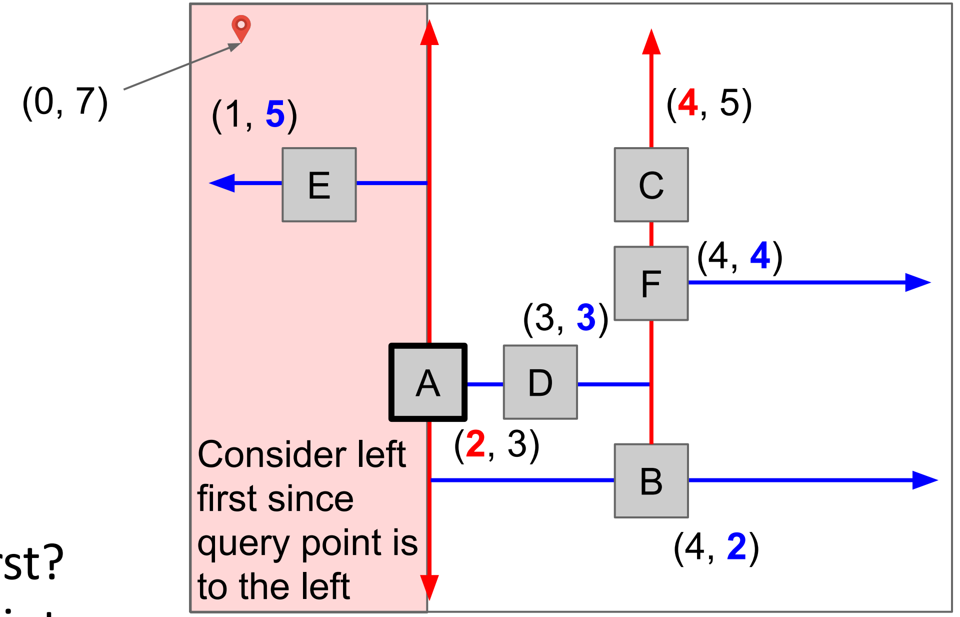
- dist(A) is sqrt(4+16) = 4.5.
- Need to consider children. Which first?

K-d Nearest Demo



Suppose we have the k-d tree shown.

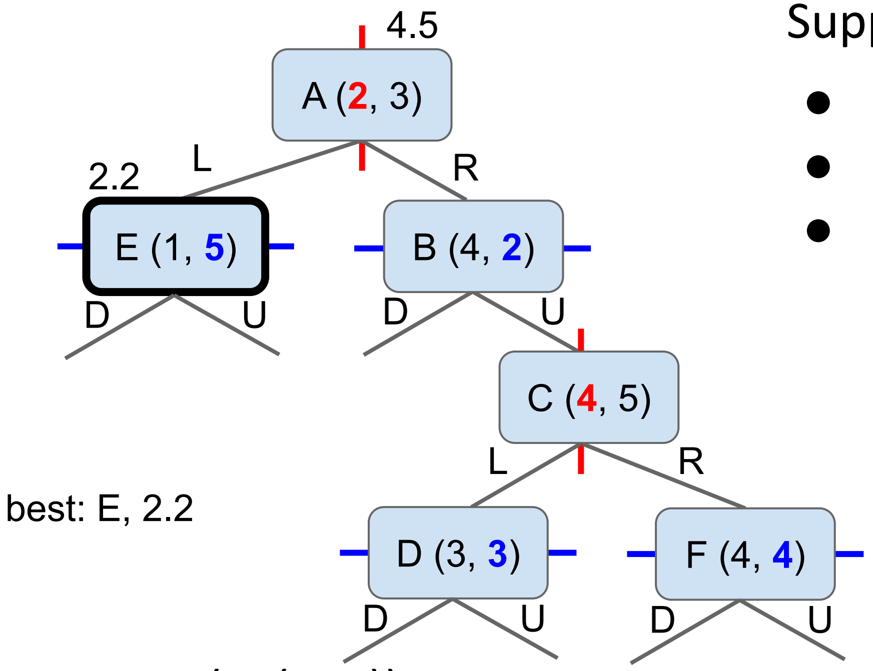
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(A, (0, 7))

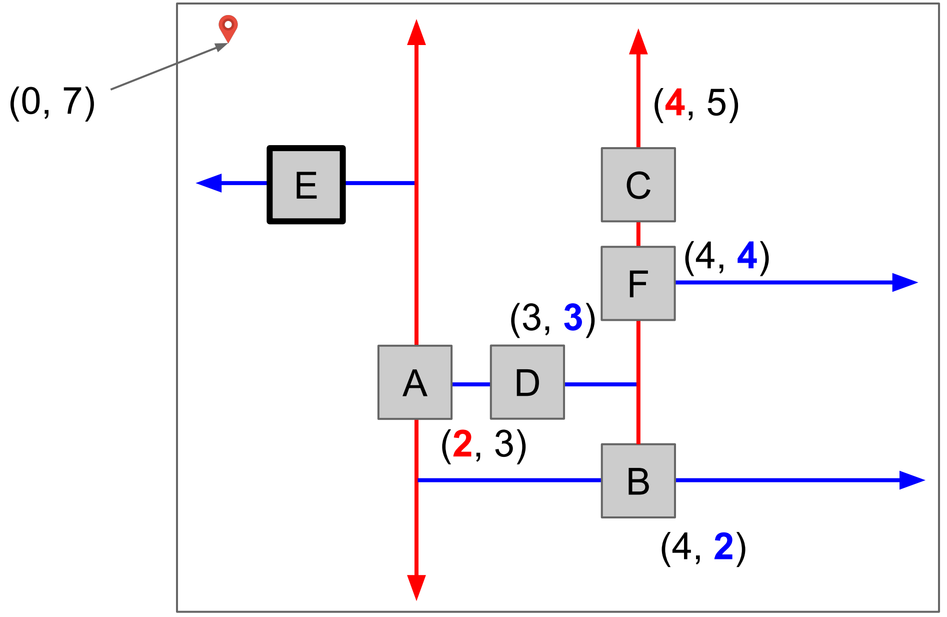
- dist(A) is sqrt(4+16) = 4.5.
- Need to consider children. Which first?
    - Whichever is closer to query point.

K-d Nearest Demo



Suppose we have the k-d tree shown.

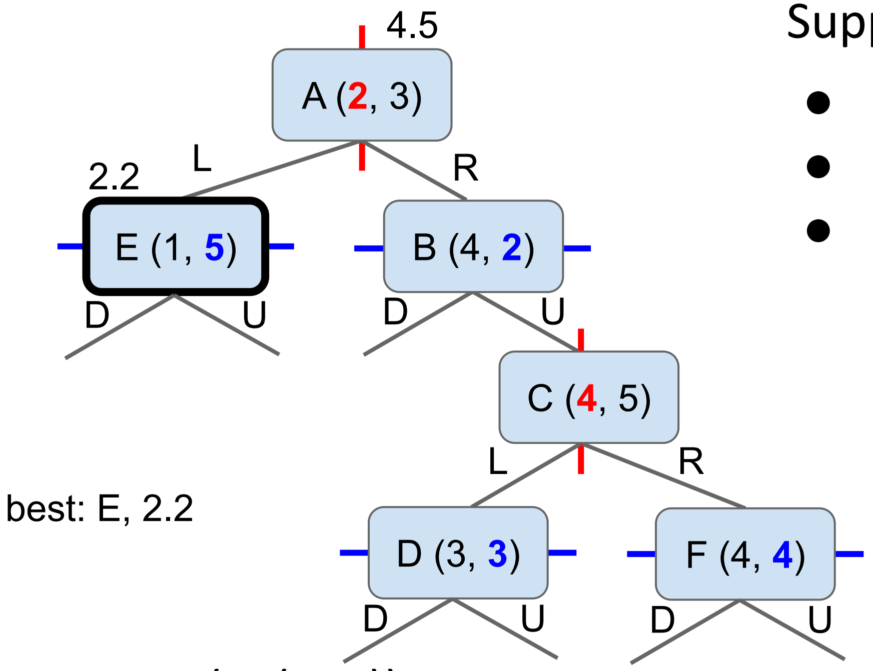
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(E, (0, 7))

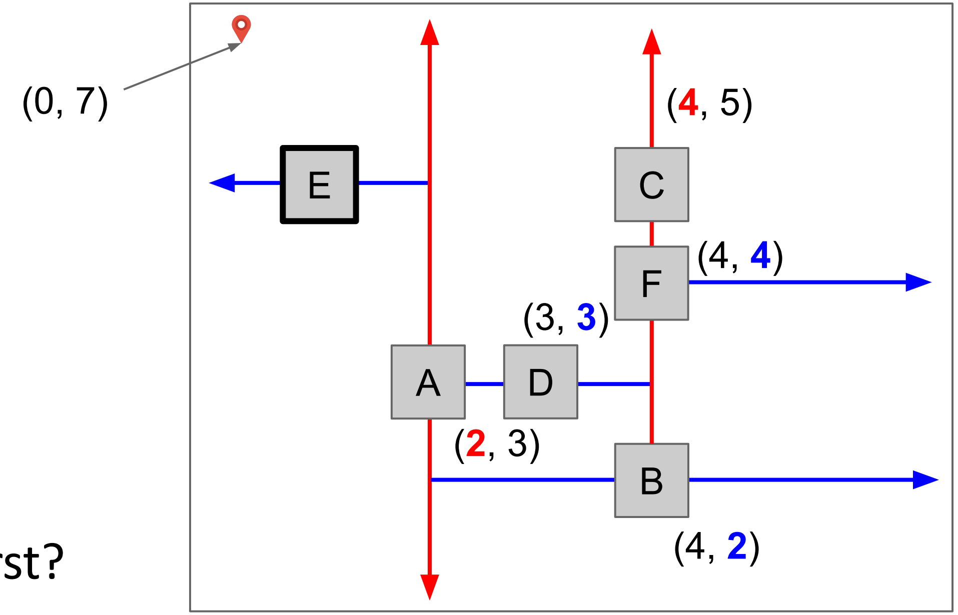
- dist(E) is sqrt(1+4) = 2.2. New best.

K-d Nearest Demo



Suppose we have the k-d tree shown.

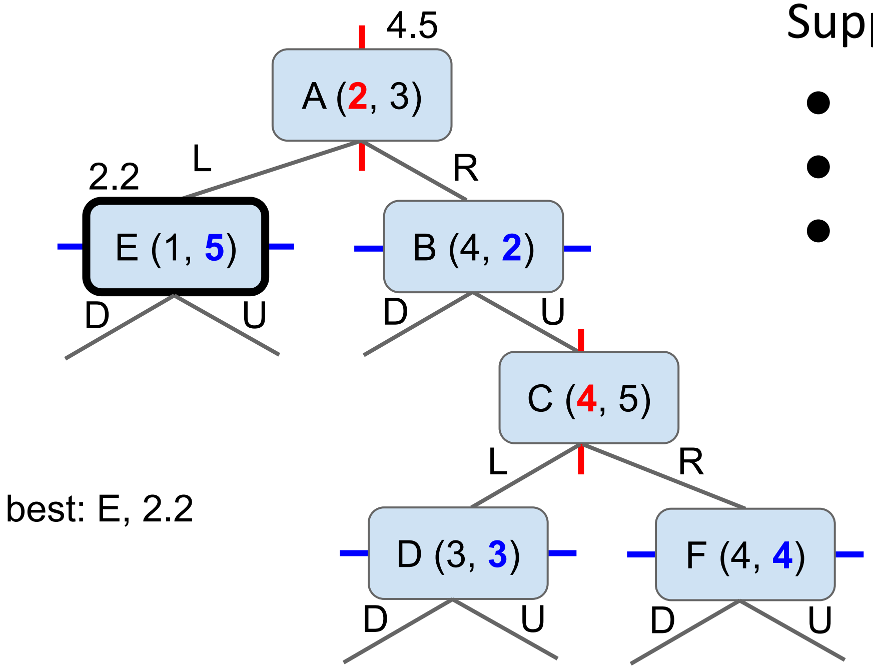
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(E, (0, 7))

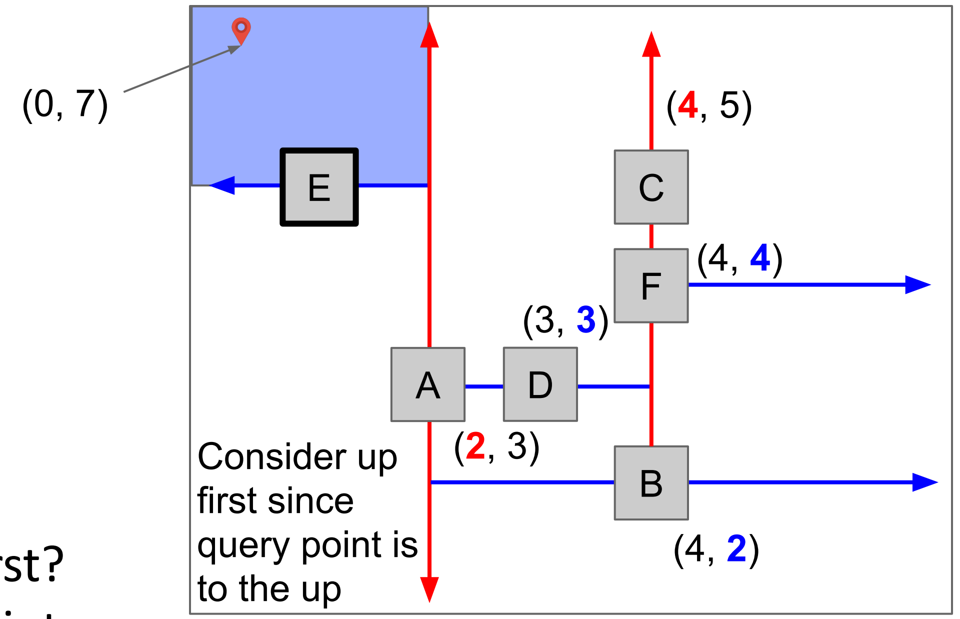
- dist(E) is sqrt(1+4) = 2.2. New best.
- Need to consider children. Which first?

K-d Nearest Demo



Suppose we have the k-d tree shown.

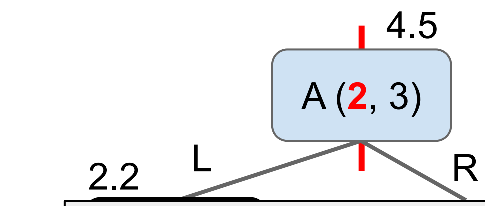
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(E, (0, 7))

- dist(E) is sqrt(1+4) = 2.2. New best.
- Need to consider children. Which first?
  - Whichever is closer to query point.

K-d Nearest Demo



Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- proper k-d tree traversal.

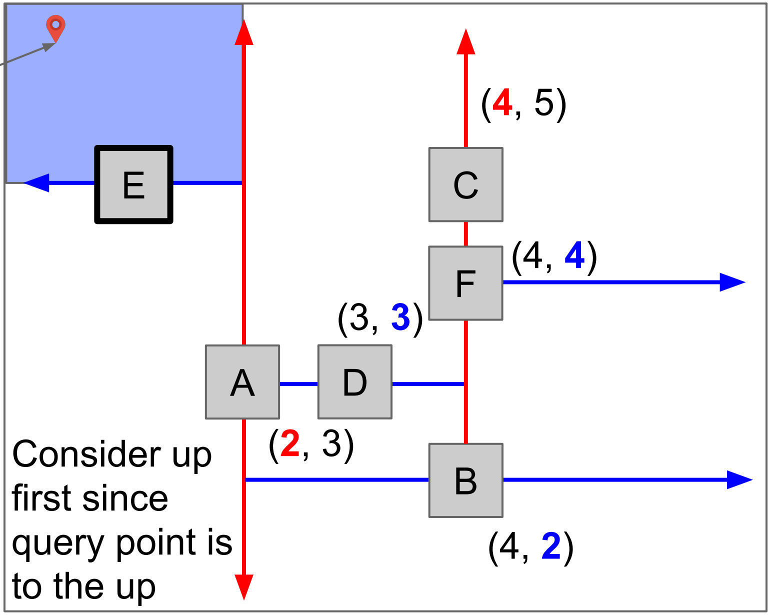
Project note: It is incredibly important that you consider the correct child first! Your code will be much slower if you always use the “left” link before the “right” one.

Intuition is that we want to search more promising parts of the space first, so we can prune less promising parts later.

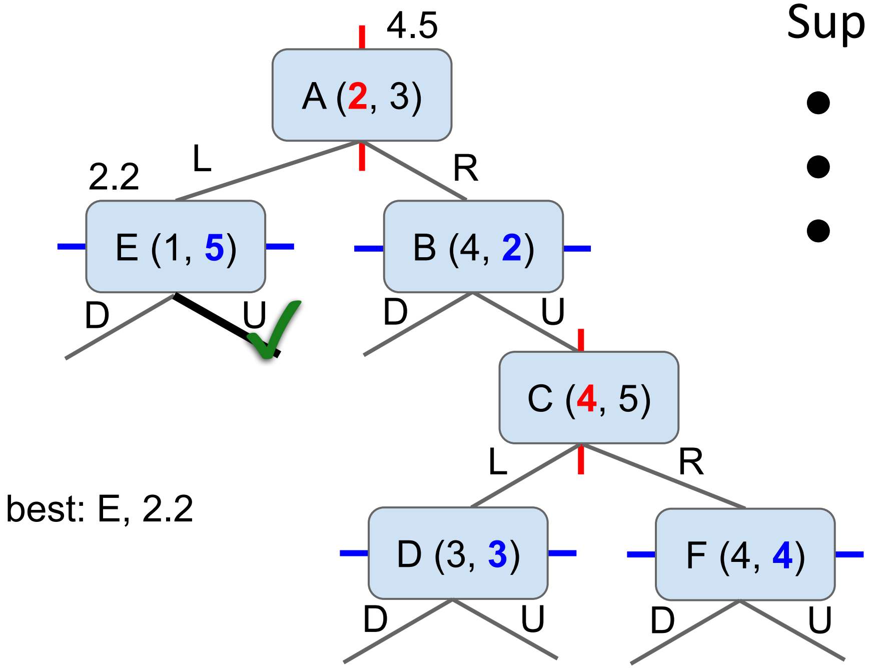
bes

n

- Need to consider children. Which first?
    - Whichever is closer to query point.

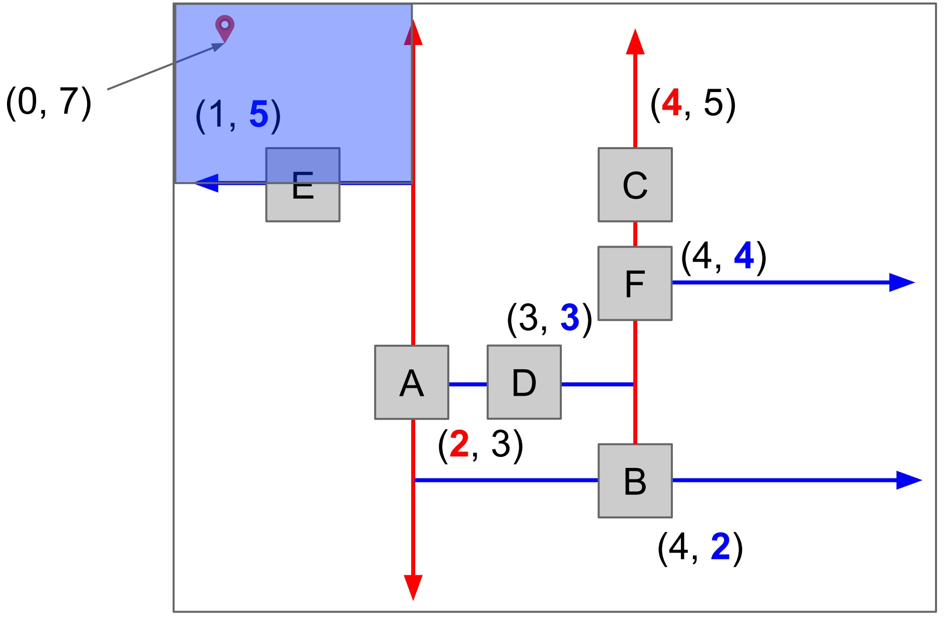


K-d Nearest Demo



Suppose we have the k-d tree shown.

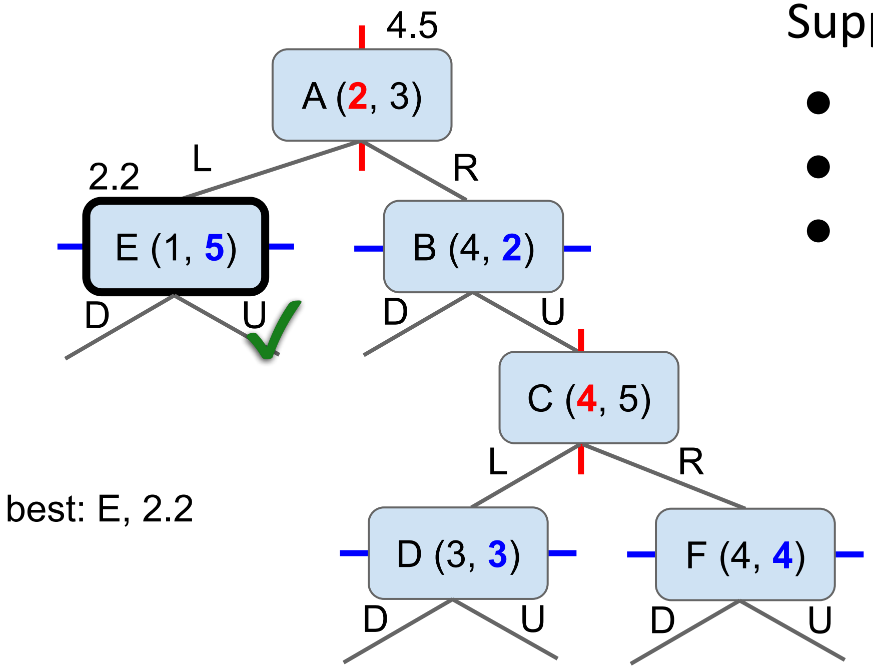
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(null, (0, 7))

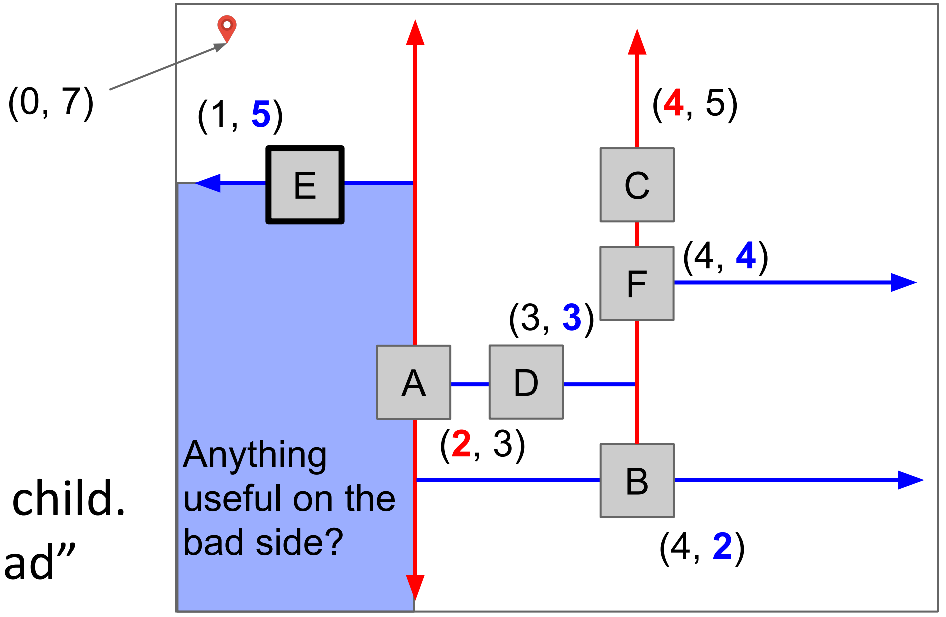
- This node is null. Return.

K-d Nearest Demo



Suppose we have the k-d tree shown.

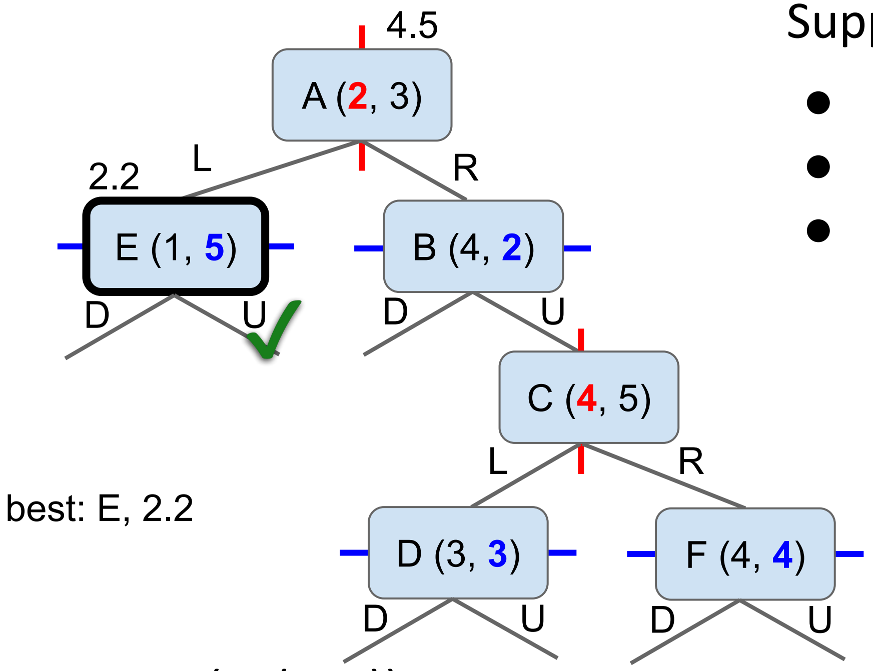
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(E, (0, 7))

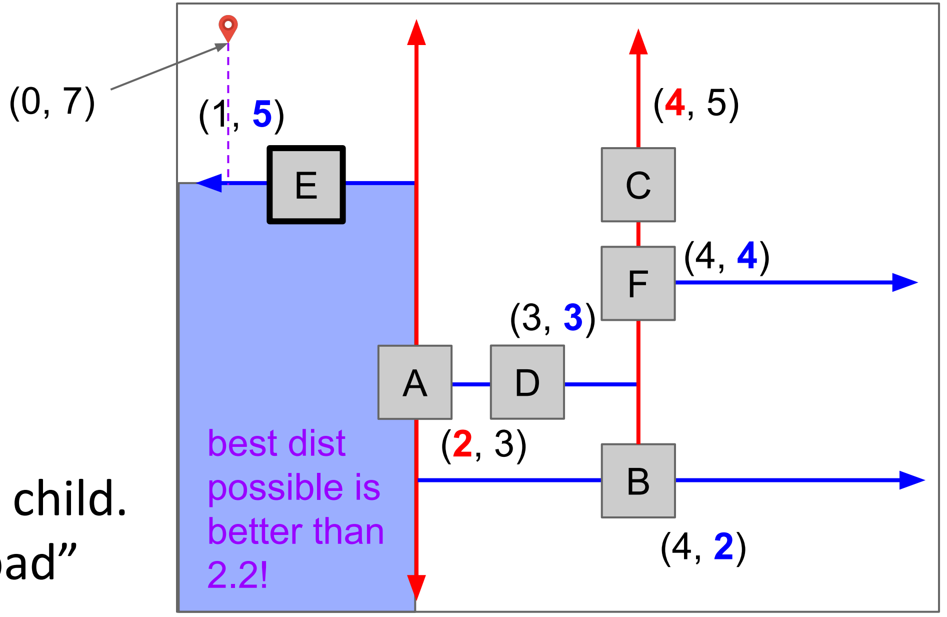
- … just finished exploring the "good" child.
- Could something better be on the "bad" side of the line, i.e. in E.down?

K-d Nearest Demo



Suppose we have the k-d tree shown.

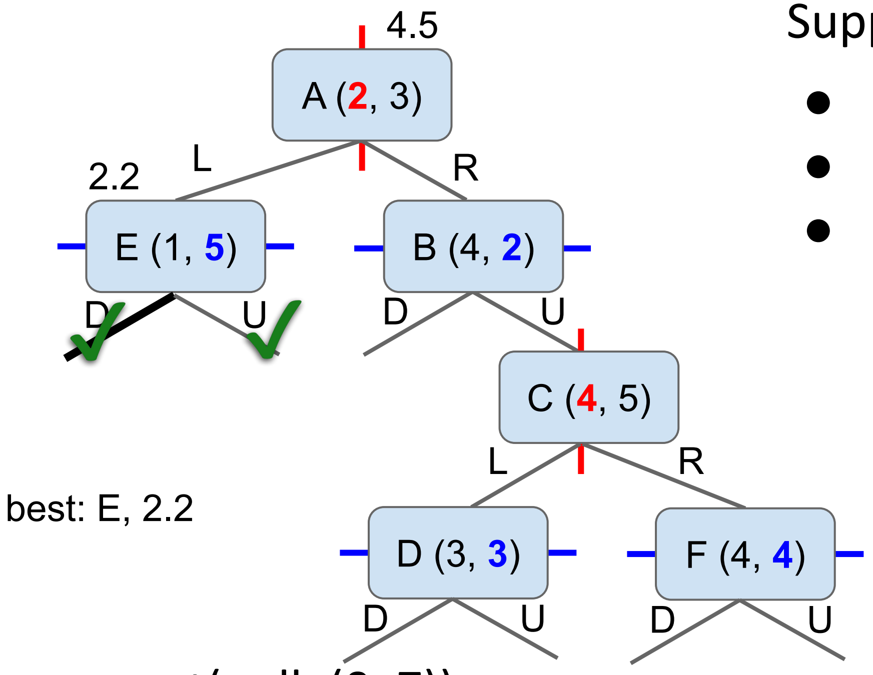
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(E, (0, 7))

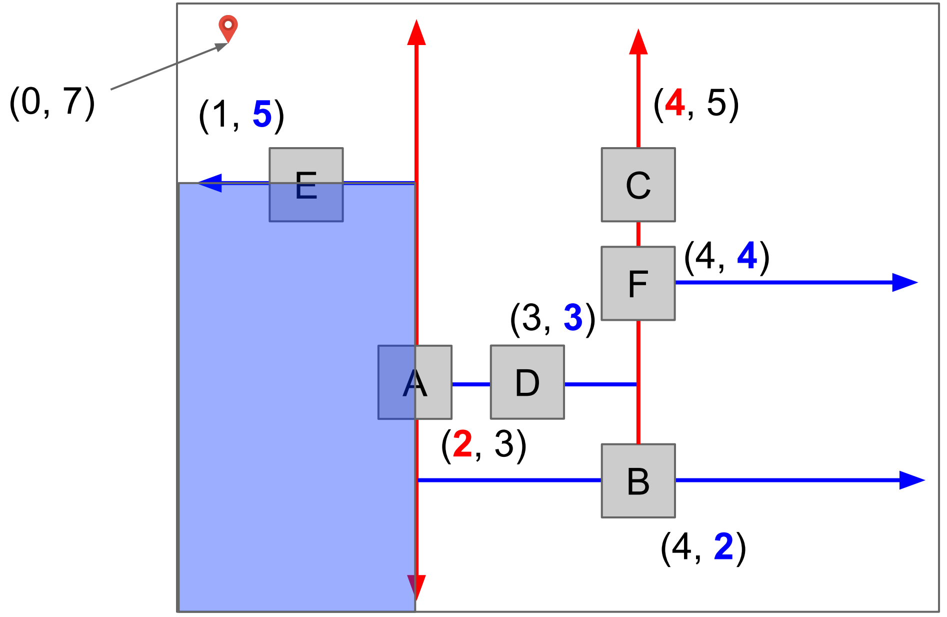
- … just finished exploring the “good” child.
- Could something better be on the “bad” side of the line, i.e. in E.down? Yes!

K-d Nearest Demo



Suppose we have the k-d tree shown.

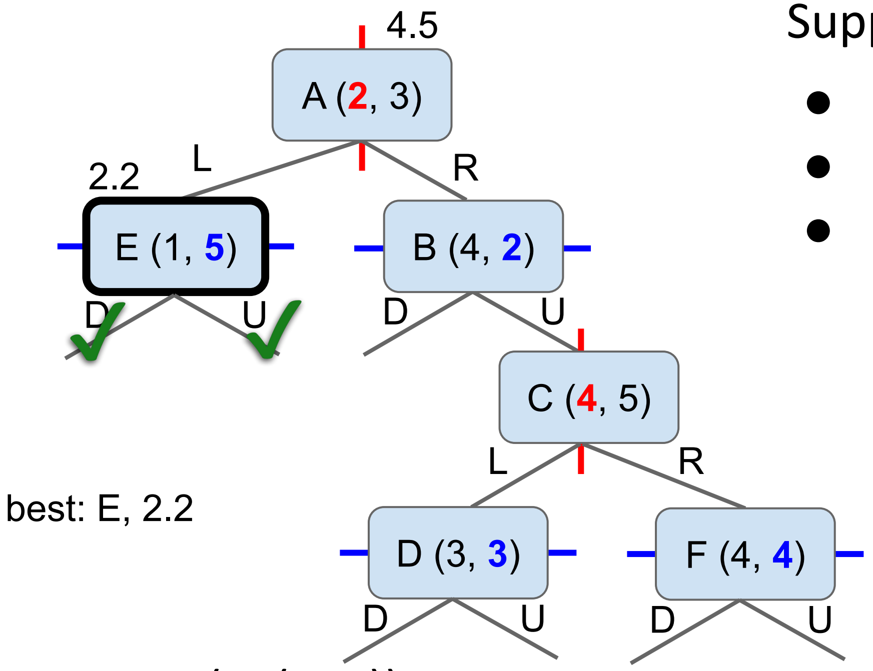
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(null, (0, 7))

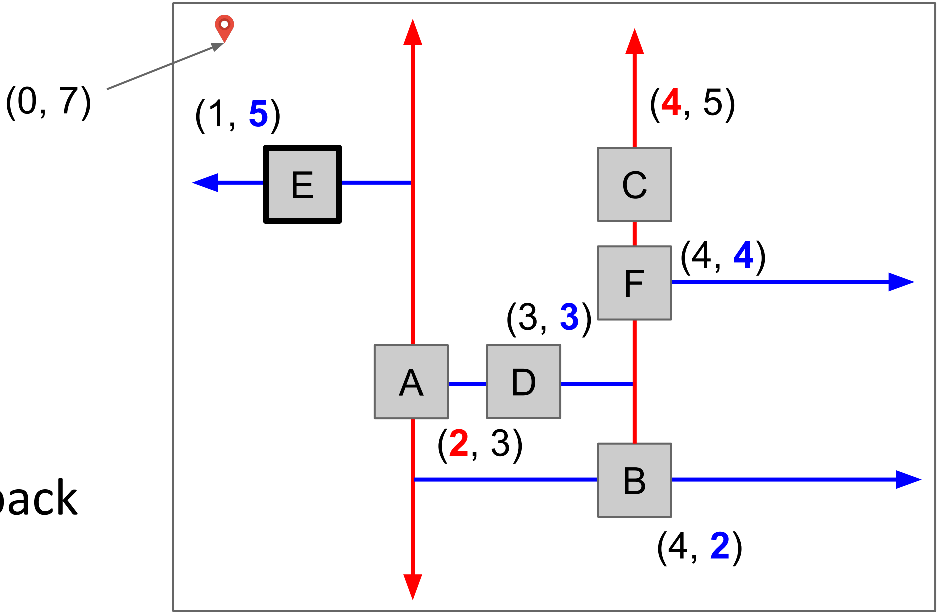
- Down link of E is null, so return.

K-d Nearest Demo



Suppose we have the k-d tree shown.

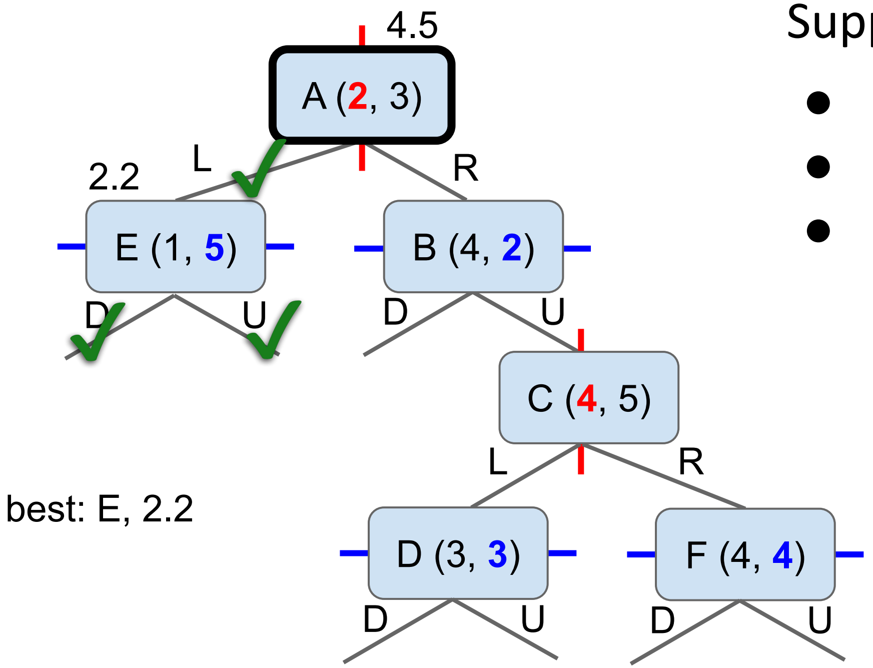
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(E, (0, 7))

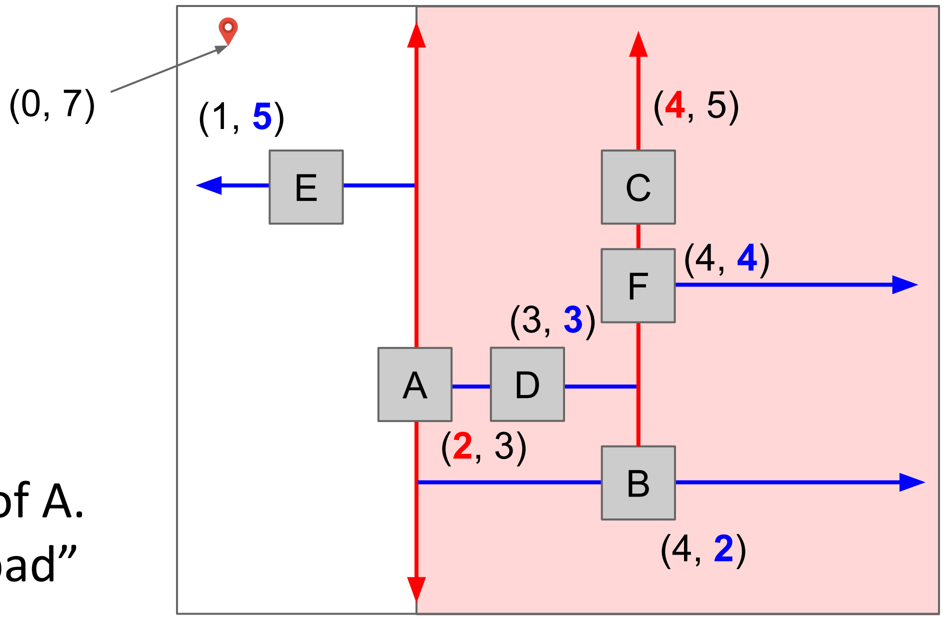
- Explored both sides of E, so let’s go back up.

K-d Nearest Demo



Suppose we have the k-d tree shown.

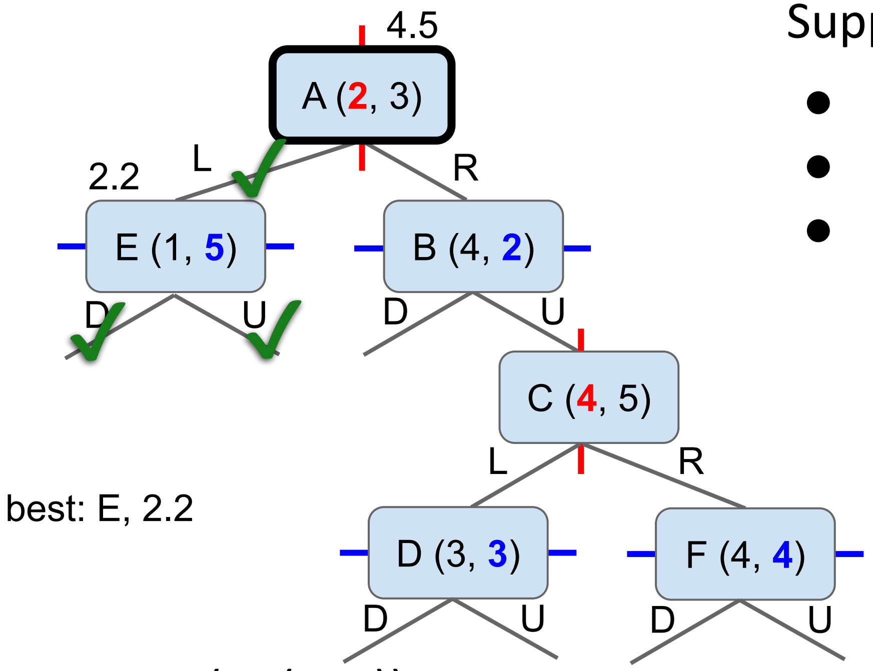
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(A, (0, 7))

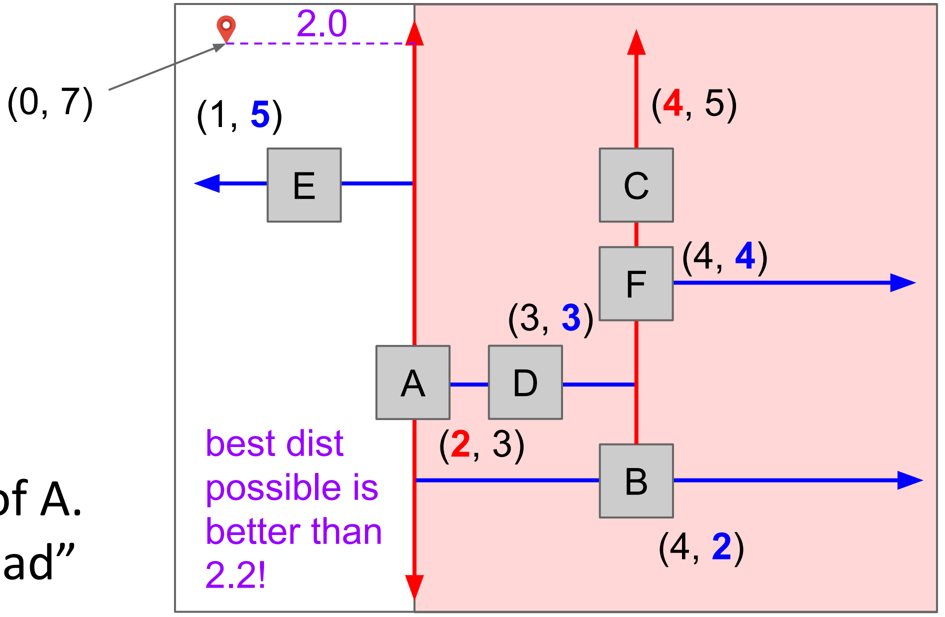
- … just finished exploring good side of A.
- Could something better be on the "bad" side of the line, i.e. in A.right?

K-d Nearest Demo



Suppose we have the k-d tree shown.

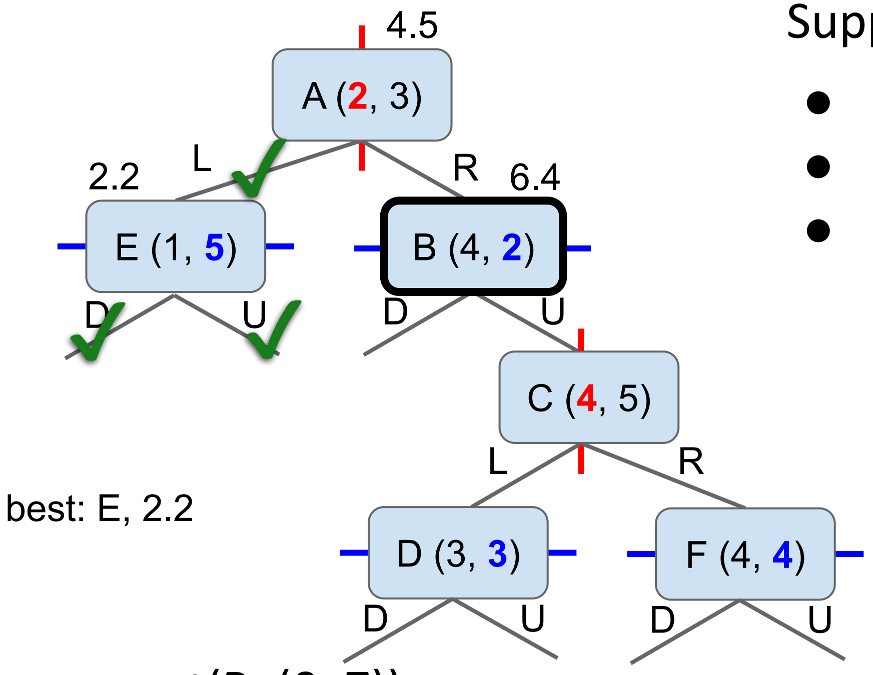
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(A, (0, 7))

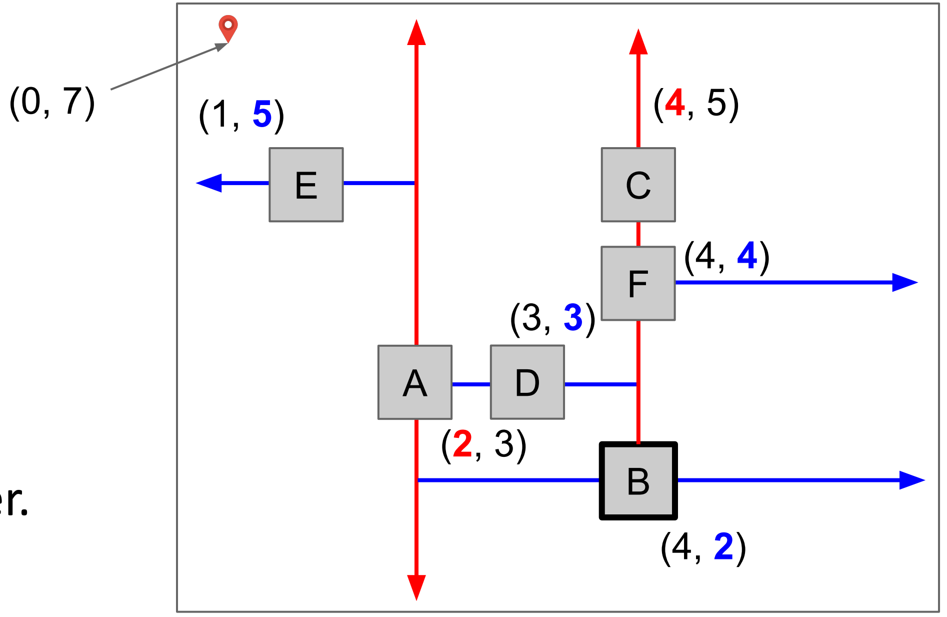
- … just finished exploring good side of A.
- Could something better be on the "bad" side of the line, i.e. in A.right? Yes!

K-d Nearest Demo



Suppose we have the k-d tree shown.

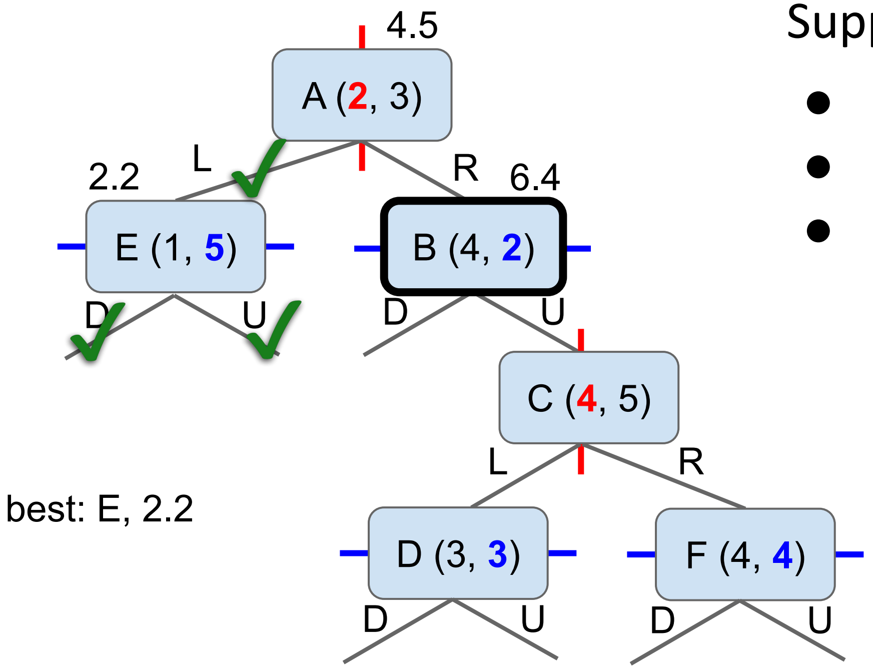
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(B, (0, 7))

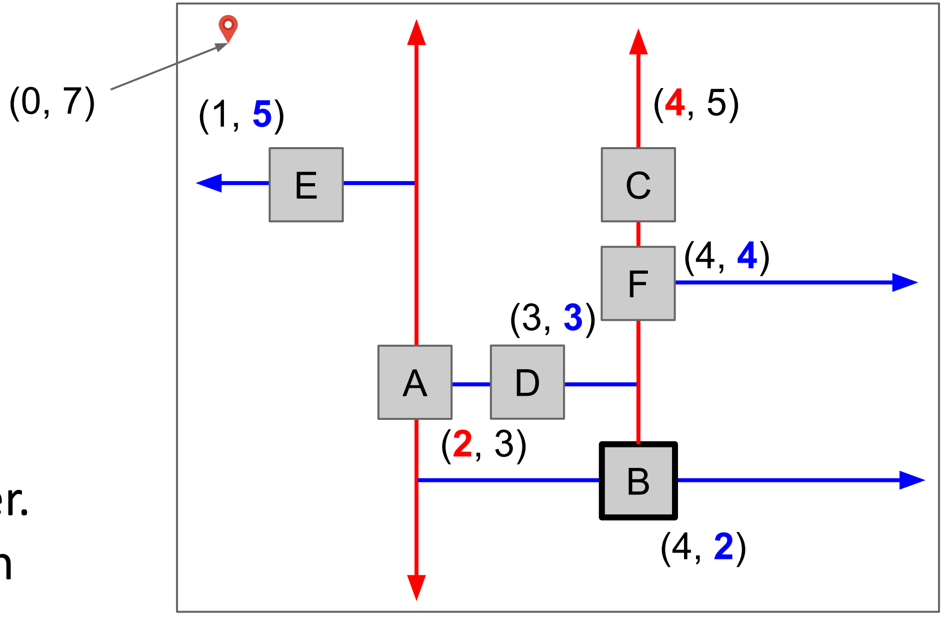
- dist(B) is sqrt(16+25) = 6.4, not better.

K-d Nearest Demo



Suppose we have the k-d tree shown.

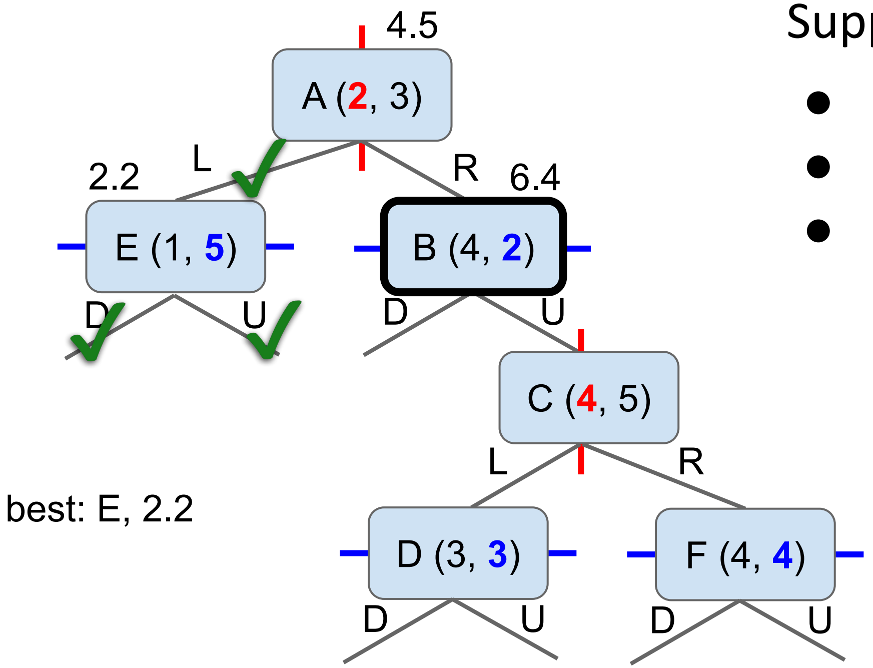
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(B, (0, 7))

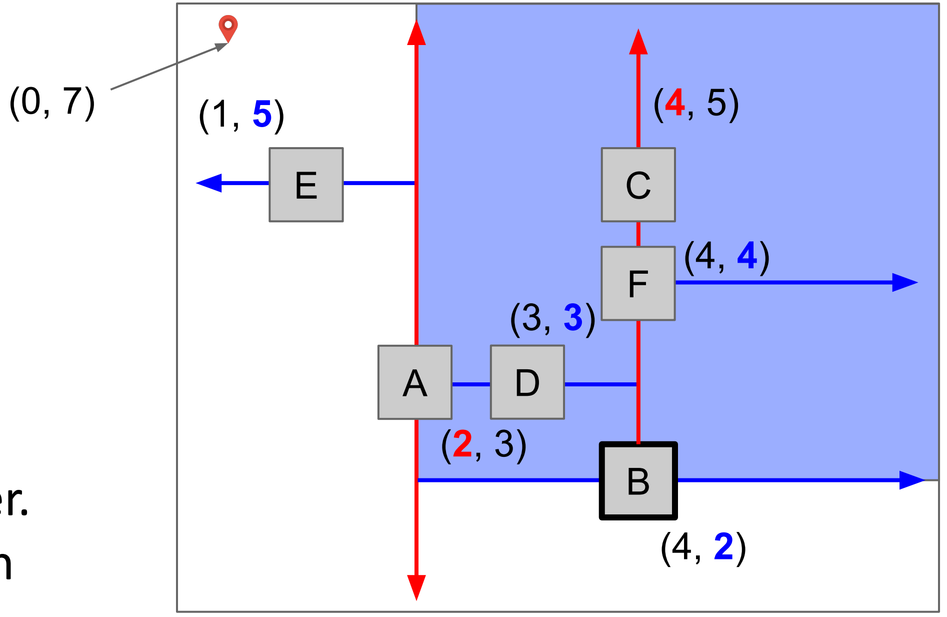
- dist(B) is sqrt(16+25) = 6.4, not better.
- Now need to explore children. Which side is the "good" side?

K-d Nearest Demo



Suppose we have the k-d tree shown.

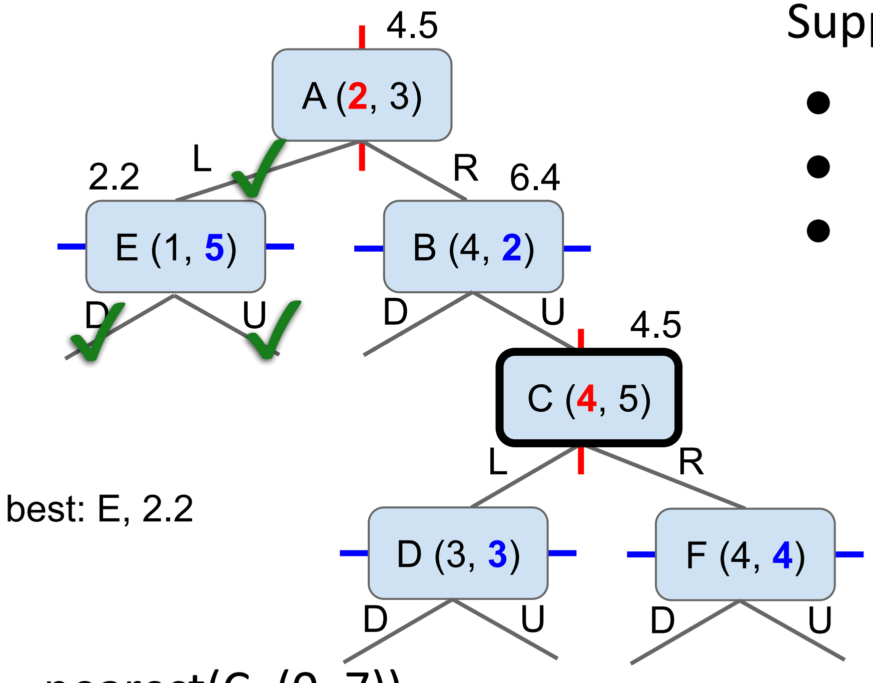
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(B, (0, 7))

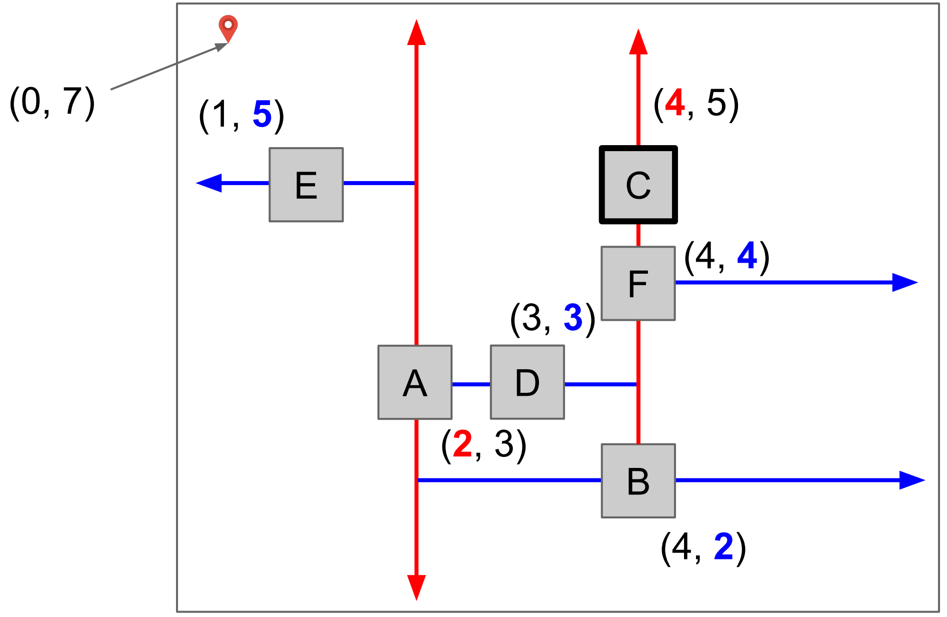
- dist(B) is sqrt(16+25) = 6.4, not better.
- Now need to explore children. Which side is the "good" side? B.up!

K-d Nearest Demo



Suppose we have the k-d tree shown.

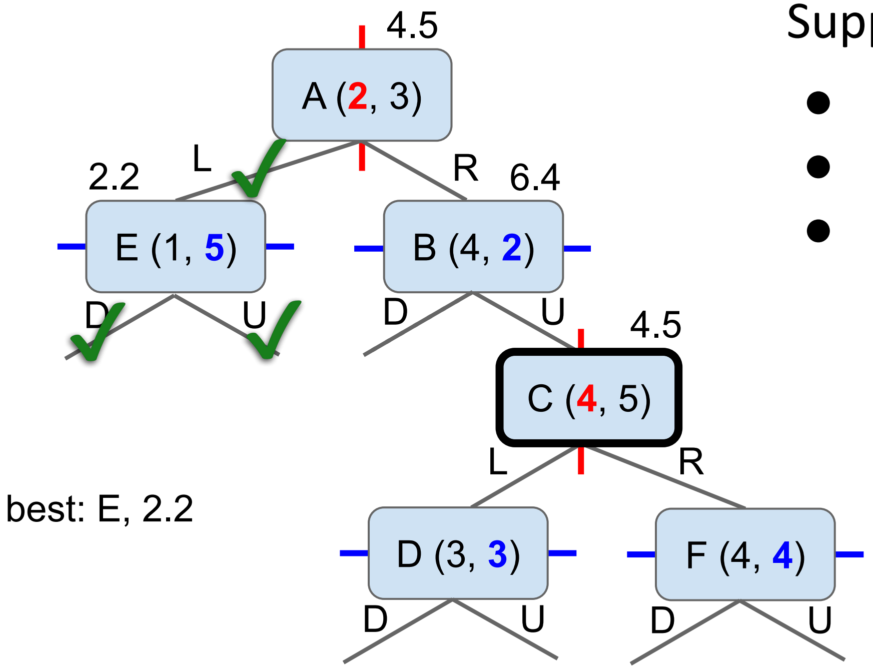
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(C, (0, 7))

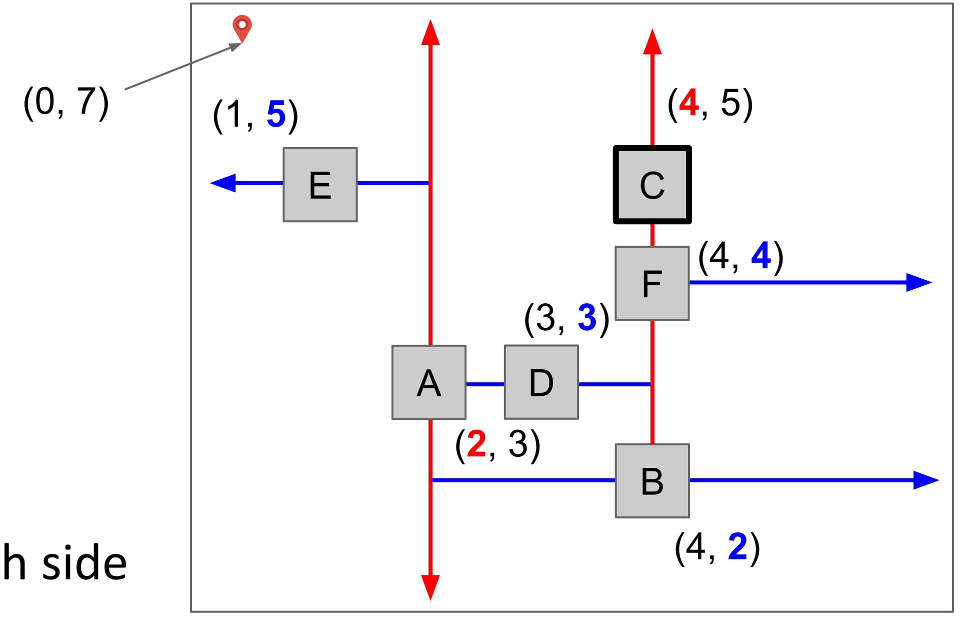
- dist(C) is sqrt(16+4) = 4.5

K-d Nearest Demo



Suppose we have the k-d tree shown.

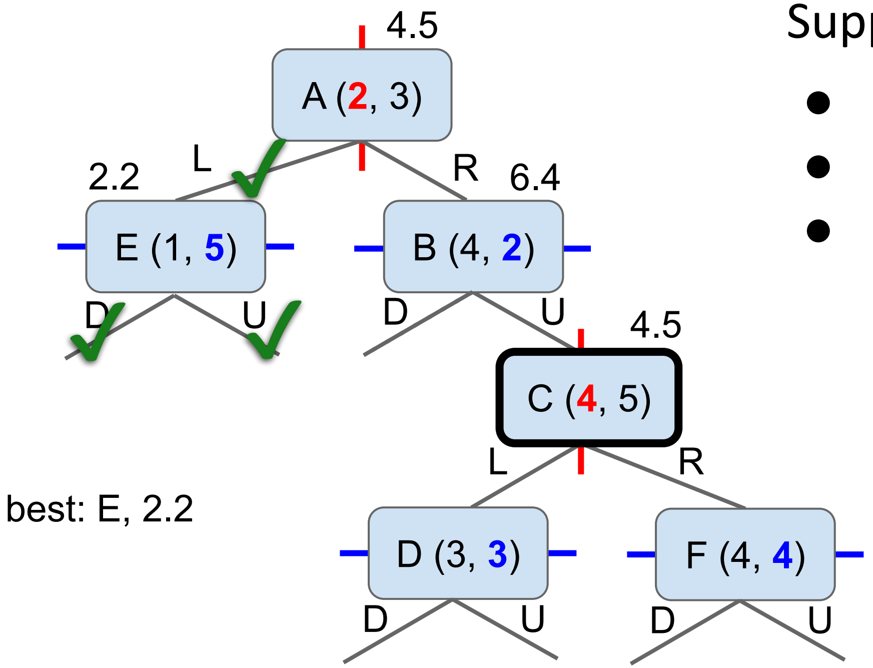
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(C, (0, 7))

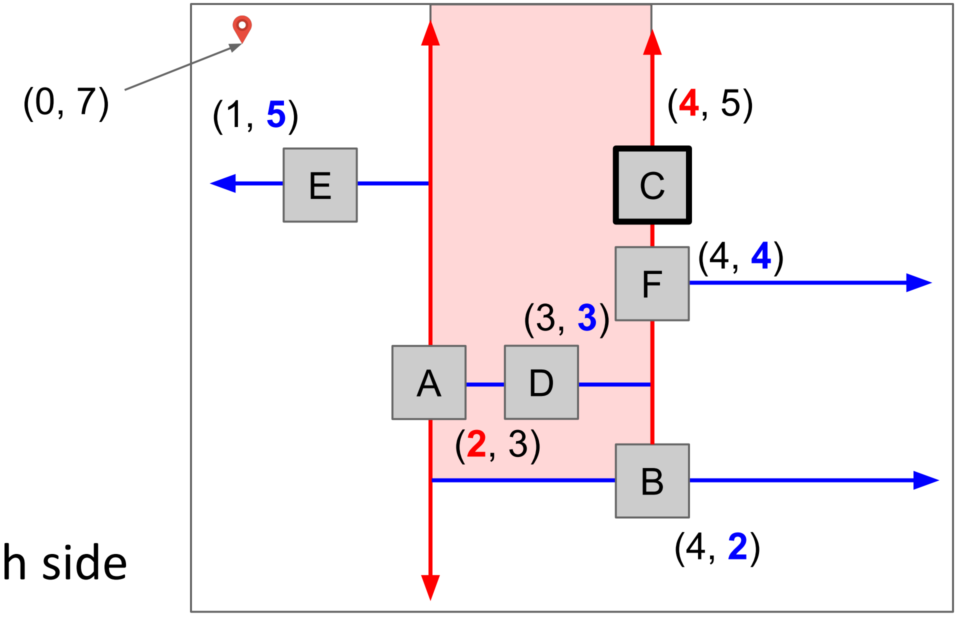
- dist(C) is sqrt(16+4) = 4.5
- Now need to explore children. Which side is the "good" side?

K-d Nearest Demo



Suppose we have the k-d tree shown.

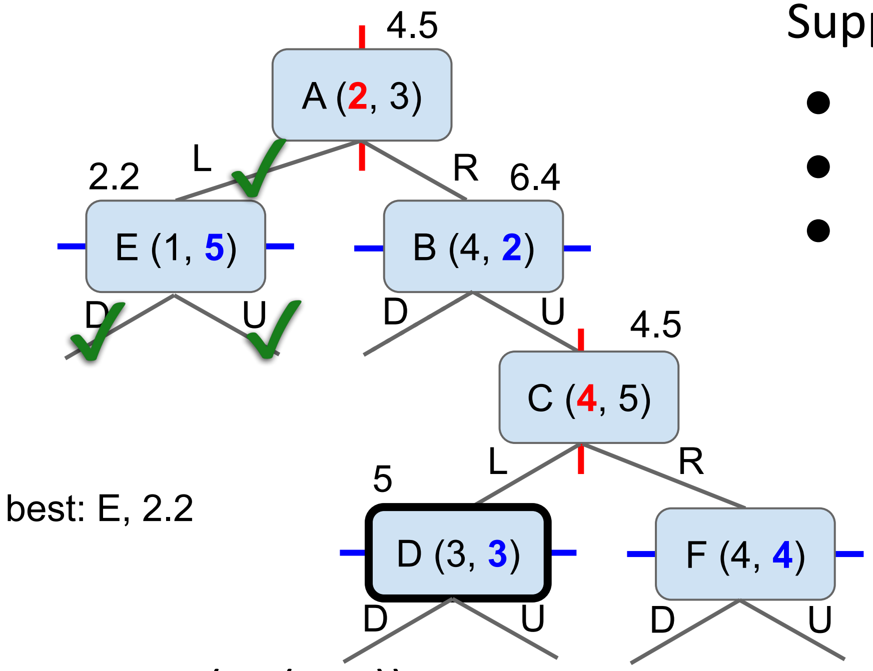
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(C, (0, 7))

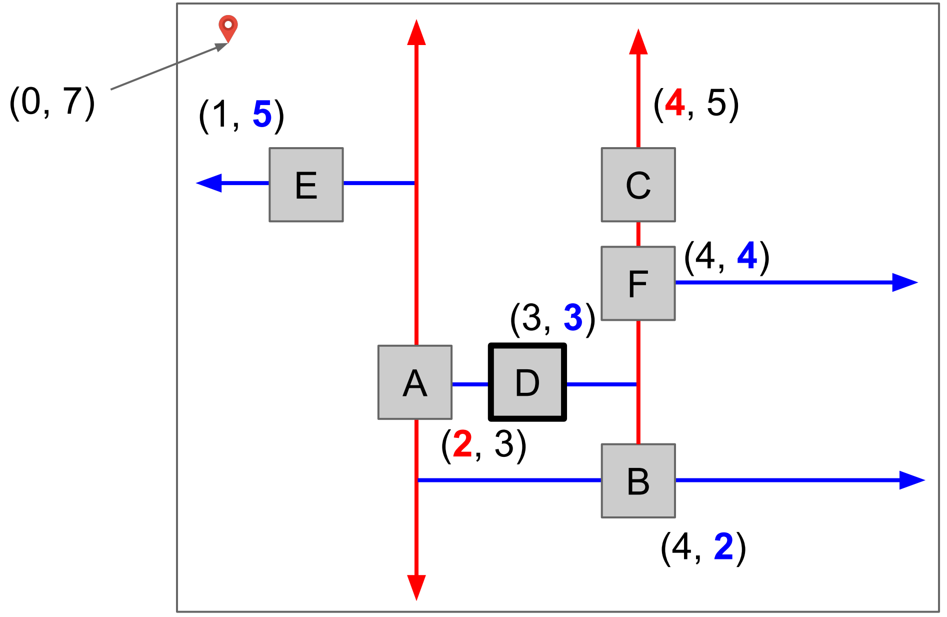
- dist(C) is sqrt(16+4) = 4.5
- Now need to explore children. Which side is the "good" side? C.left!

K-d Nearest Demo



Suppose we have the k-d tree shown.

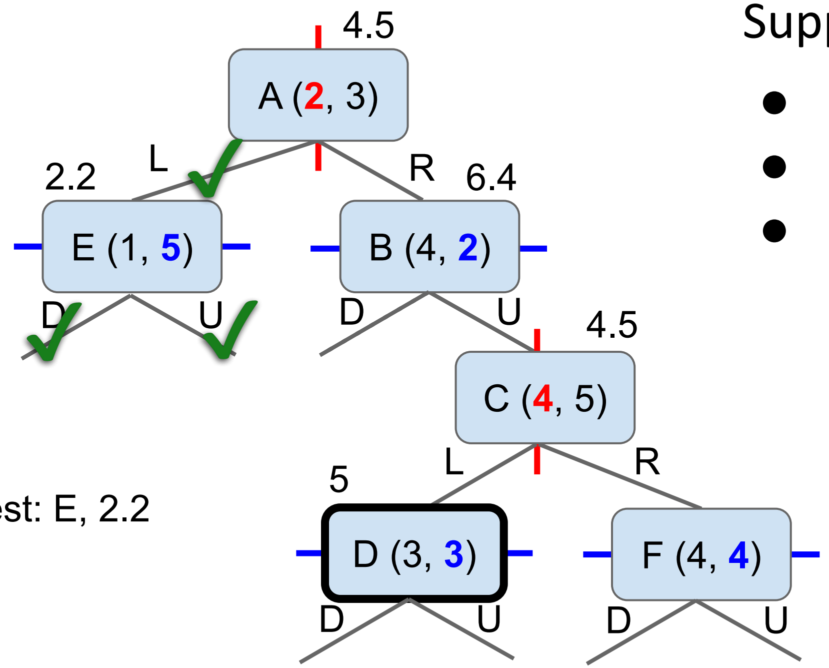
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(D, (0, 7))

- dist(D) is sqrt(9+16) = 5, not better.

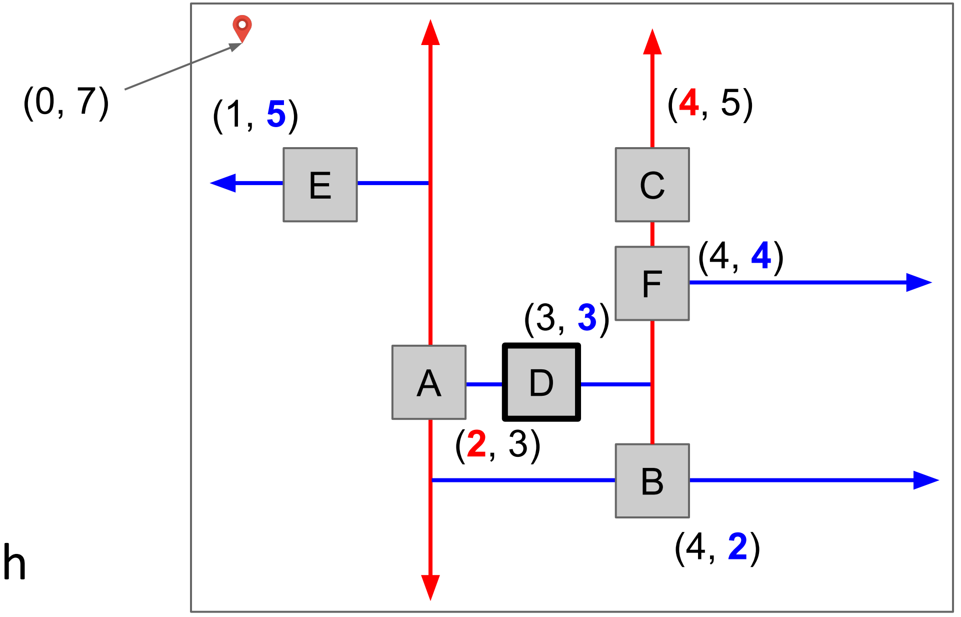
K-d Nearest Demo



best: E, 2.2

Suppose we have the k-d tree shown.

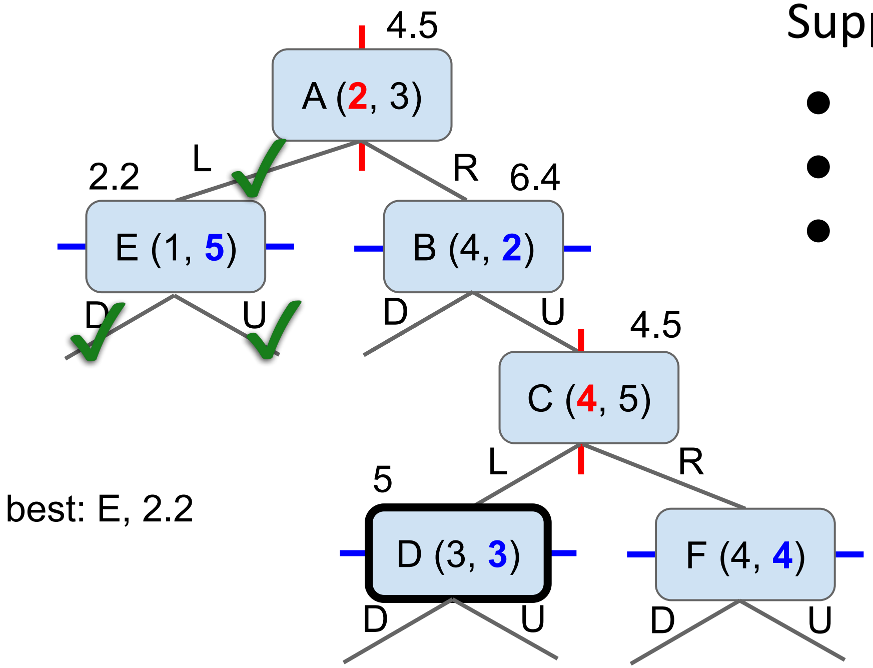
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(D, (0, 7))

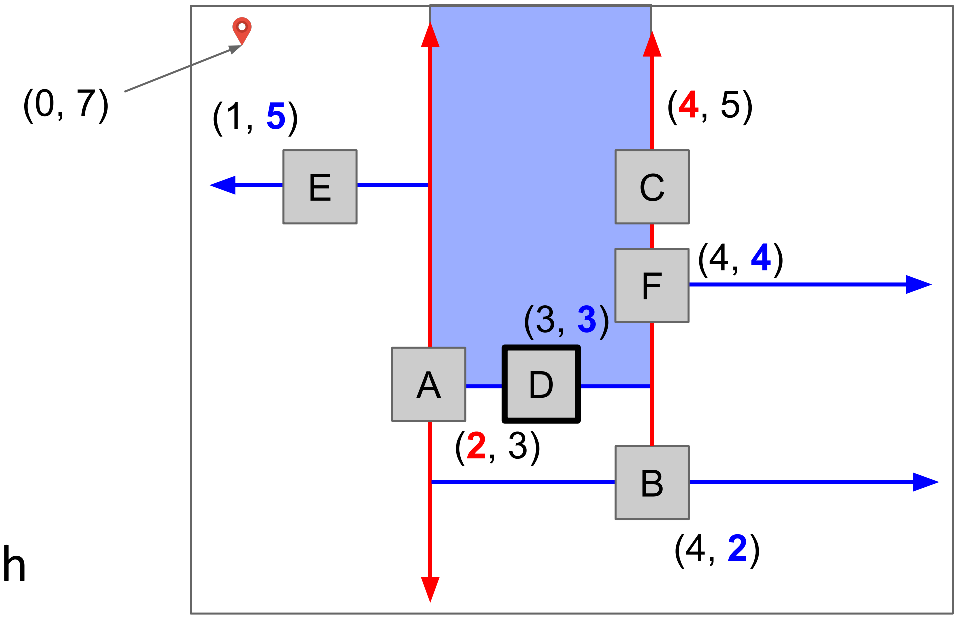
- dist(D) is sqrt(9+16) = 5, not better.
- Now need to explore children. Which side is the "good" side?

K-d Nearest Demo



Suppose we have the k-d tree shown.

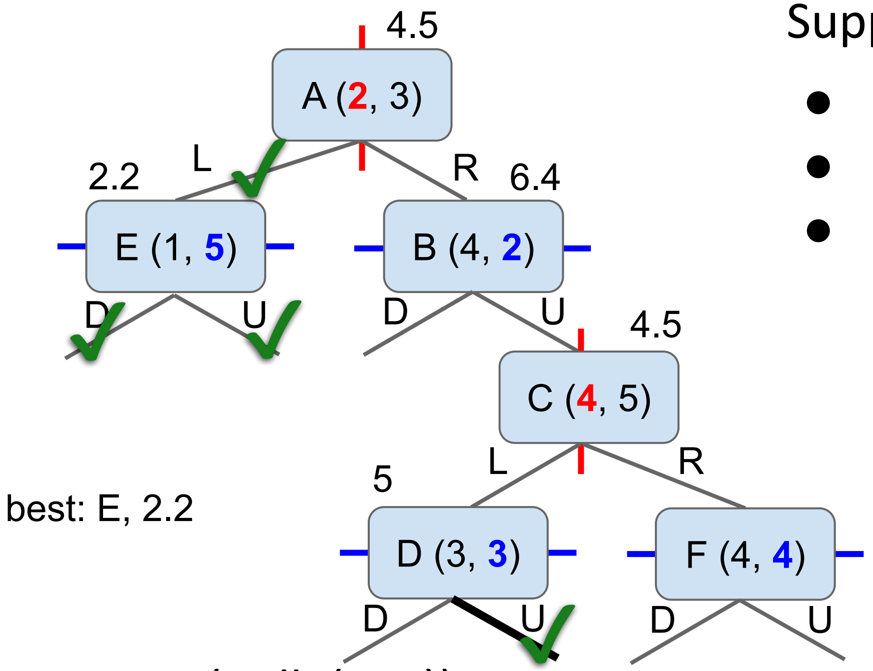
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(D, (0, 7))

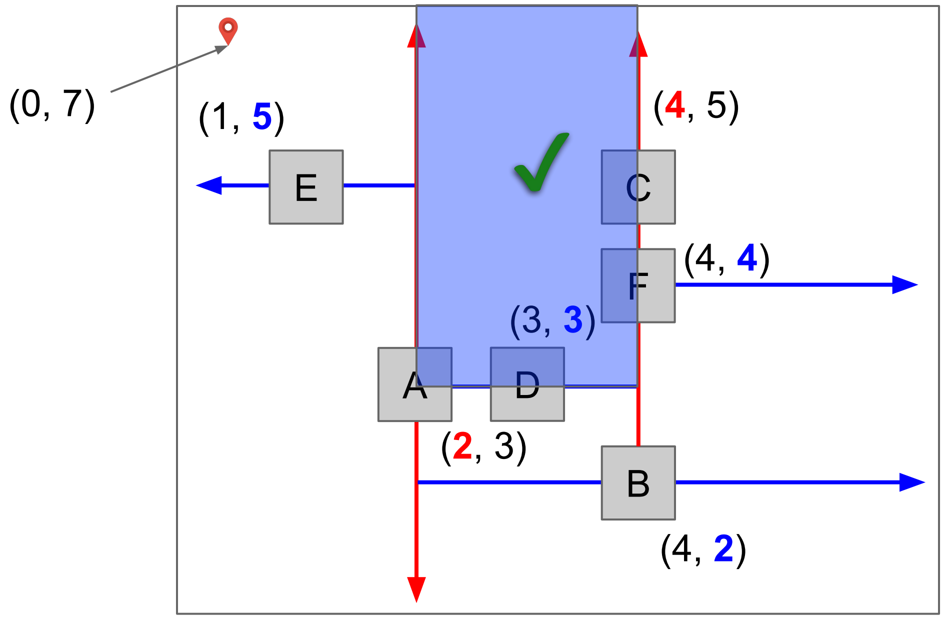
- dist(D) is sqrt(9+16) = 5, not better.
- Now need to explore children. Which side is the “good” side? D.up!

K-d Nearest Demo



Suppose we have the k-d tree shown.

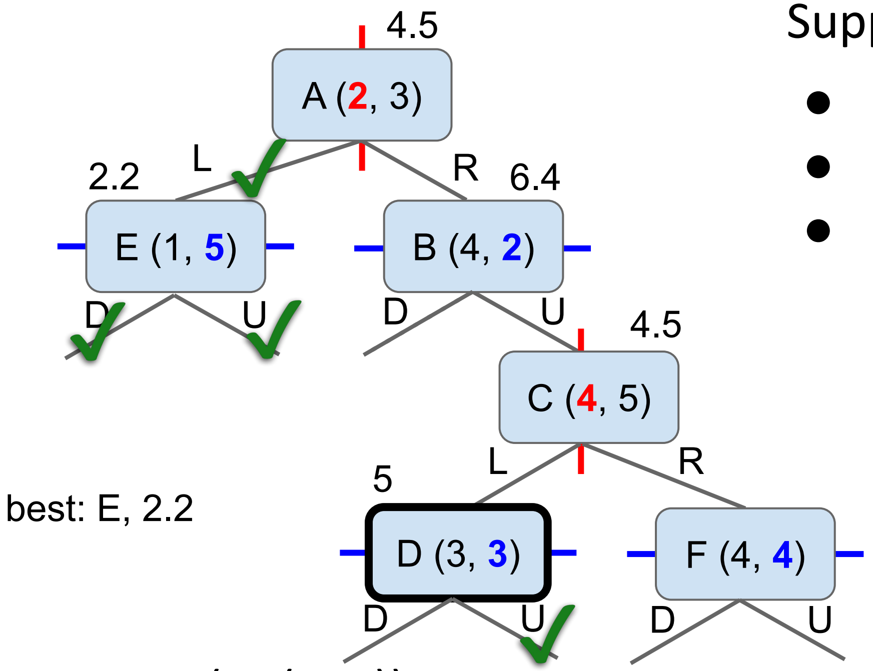
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(null, (0, 7))

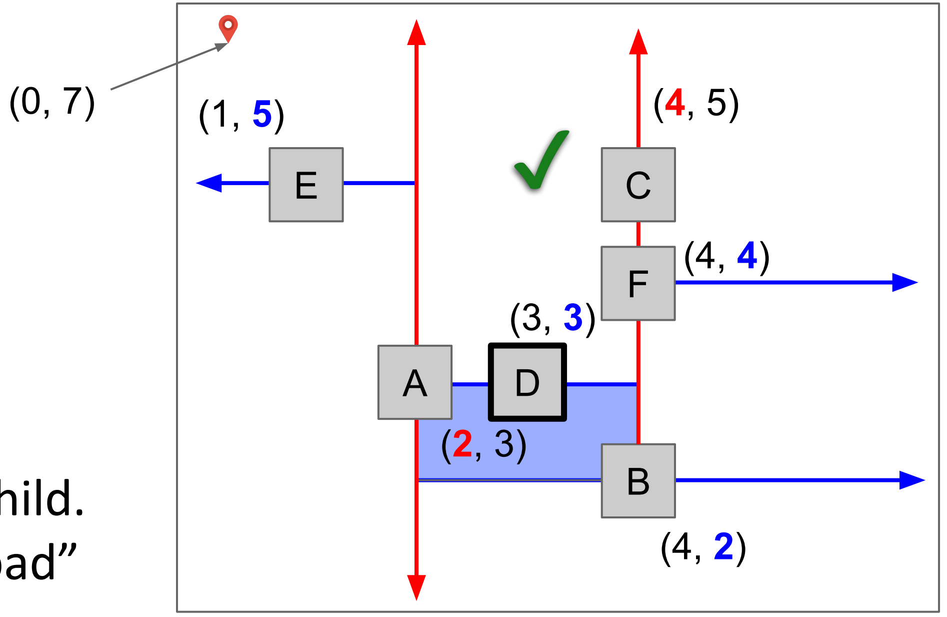
- This node is null. Return.

K-d Nearest Demo



Suppose we have the k-d tree shown.

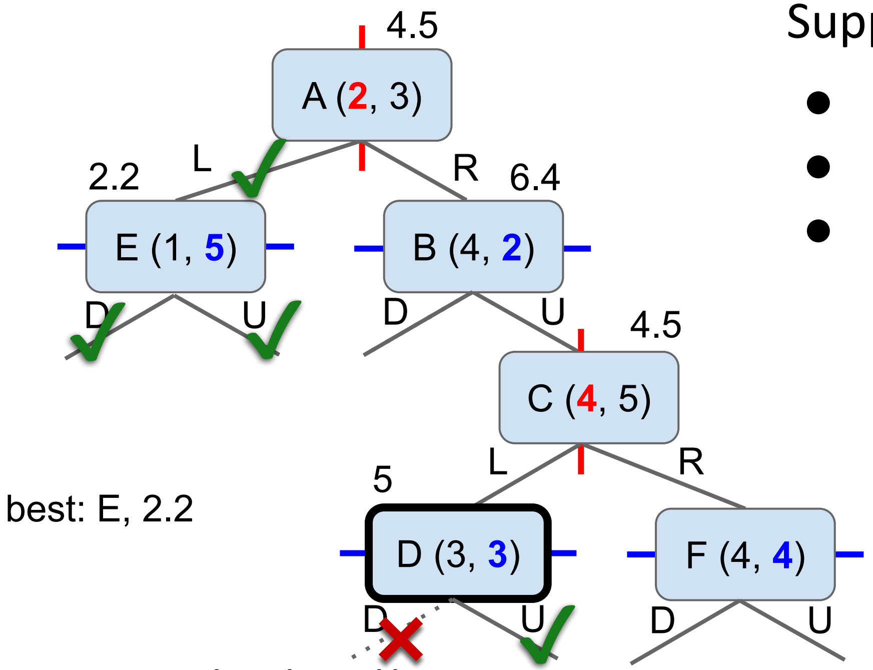
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(D, (0, 7))

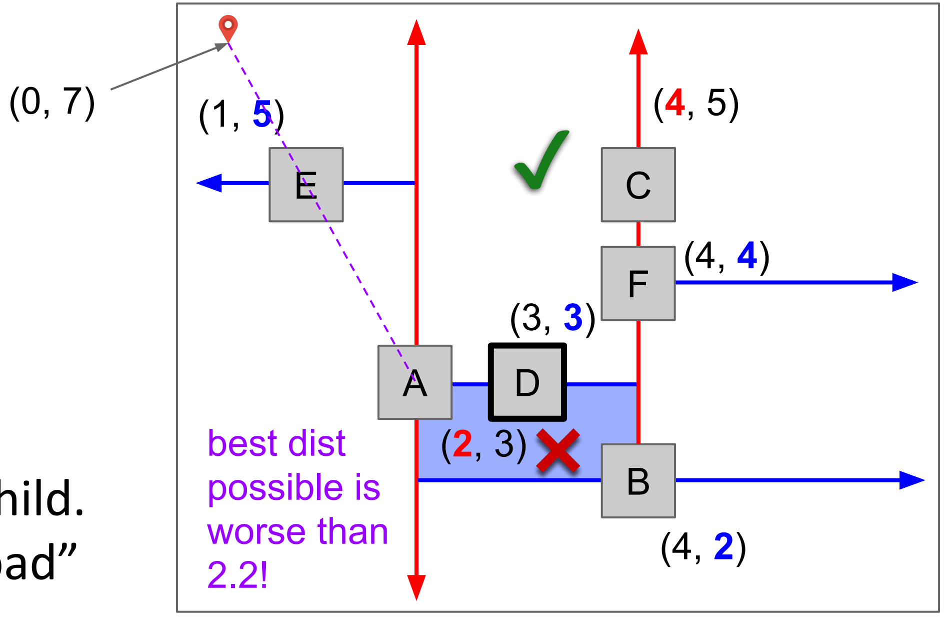
- … just finished exploring the good child.
- Could something better be on the "bad" side of the line, i.e. D.down?

K-d Nearest Demo



Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(D, (0, 7))

- … just finished exploring the good child.
- Could something better be on the "bad" side of the line, i.e. D.down? No!

K-d Nearest Demo

Project note: You can simplify your code by only measuring the length of the **green** dashed vertical line rather than the purple diagonal hypotenuse. So here, we'd compute goal.y - d.y (which is pretty easy!)

- Warning, the Point class in proj 2b actually uses the squared distance, so you'll need to compare with (goal.y - d.y)^2.
- Or even better, you can create the best hypothetical point and use Point.distance.

Effectively the green pruning rule is less aggressive than the purple one, so we might sometimes look at a "bad side" that has no possible better points. However, the resulting answer will still be correct.

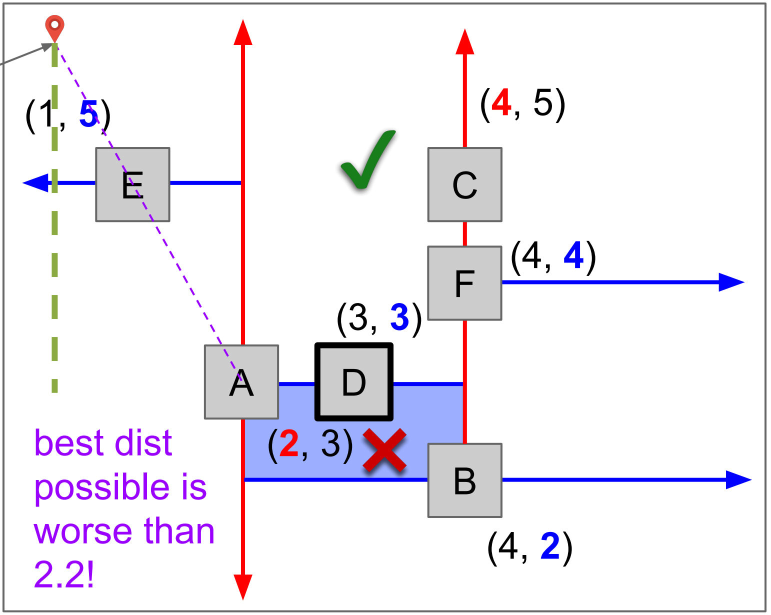
ave the k-d tree shown.

to find nearest((0, 7)).

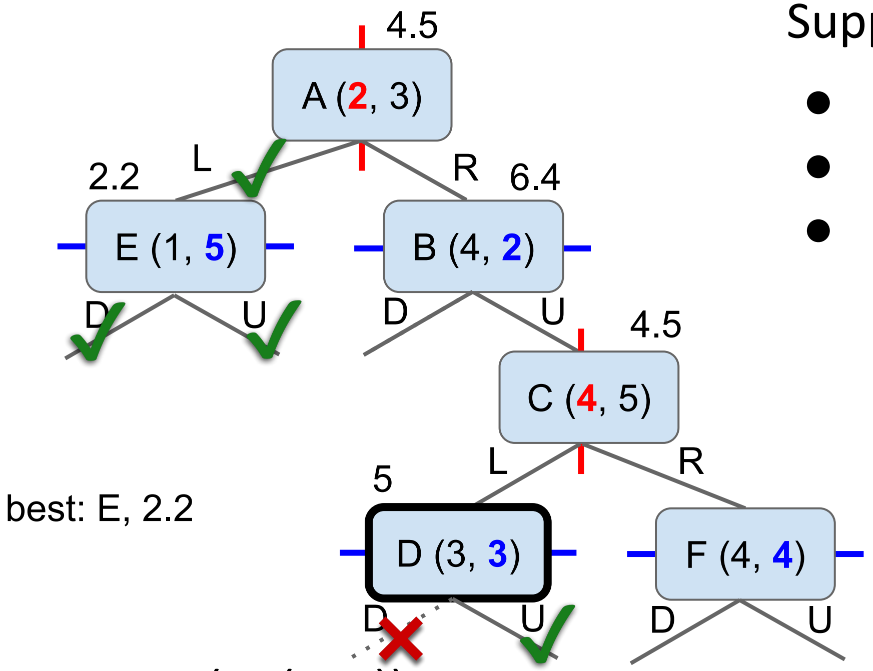
lly see the answer is (1, 5).

proper k-d tree traversal.

be



K-d Nearest Demo

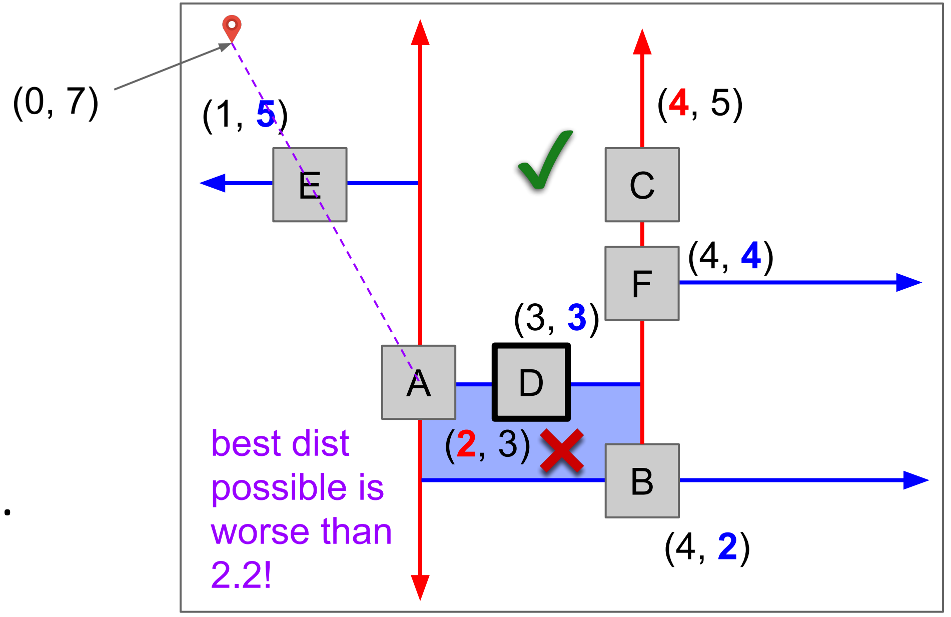


nearest(D, (0, 7))

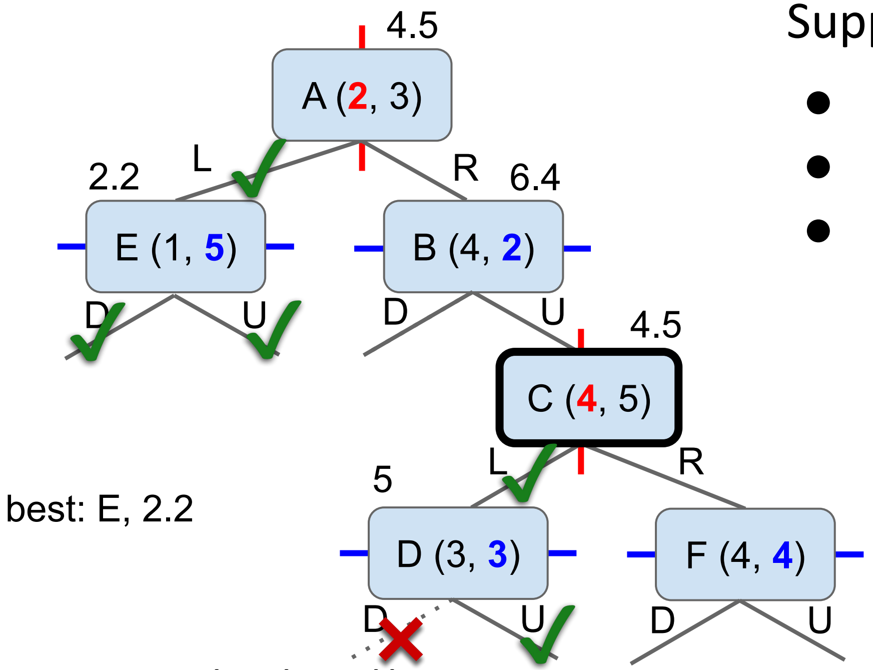
- Explored good side, pruned bad side.
- All done with D, so let's go back up.

Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.

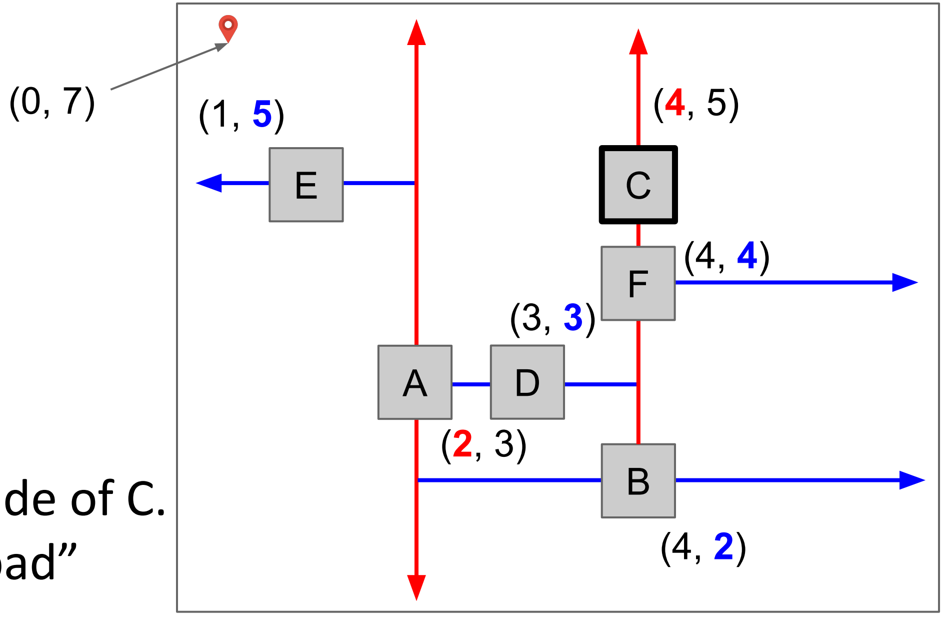


K-d Nearest Demo



Suppose we have the k-d tree shown.

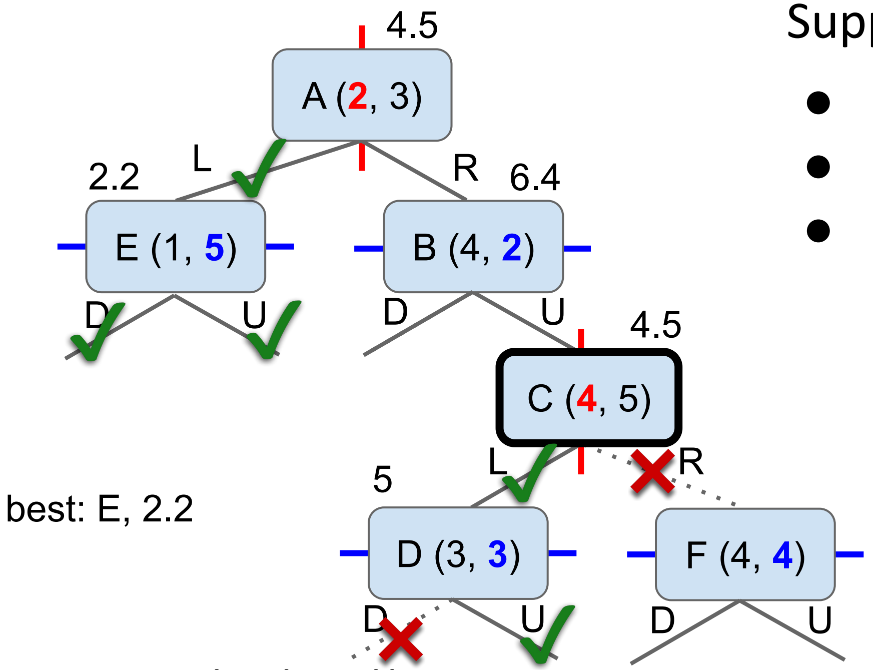
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(C, (0, 7))

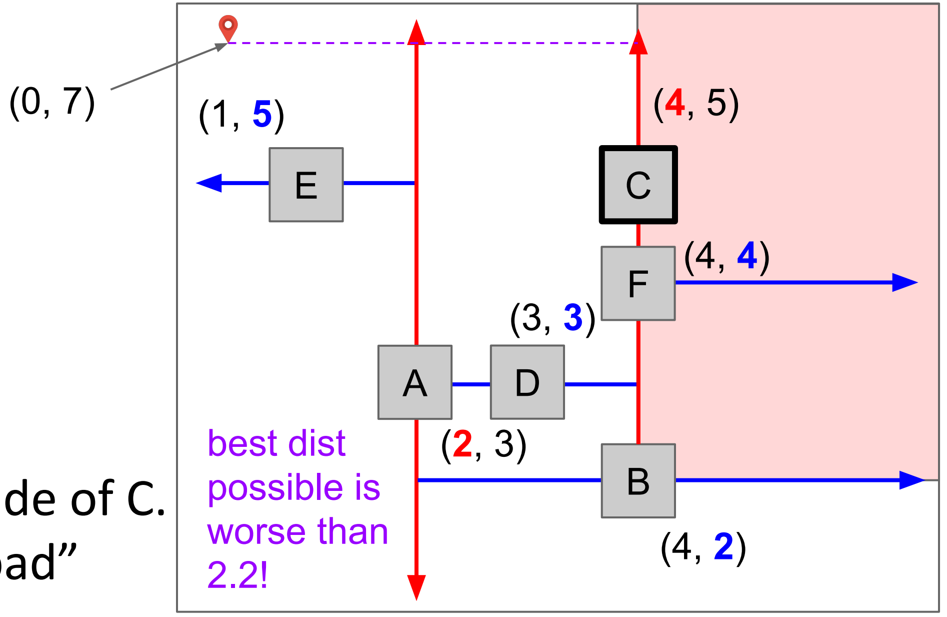
- … just finished exploring the good side of C.
- Could something better be on the "bad" side of the line, i.e. C.right?

K-d Nearest Demo



Suppose we have the k-d tree shown.

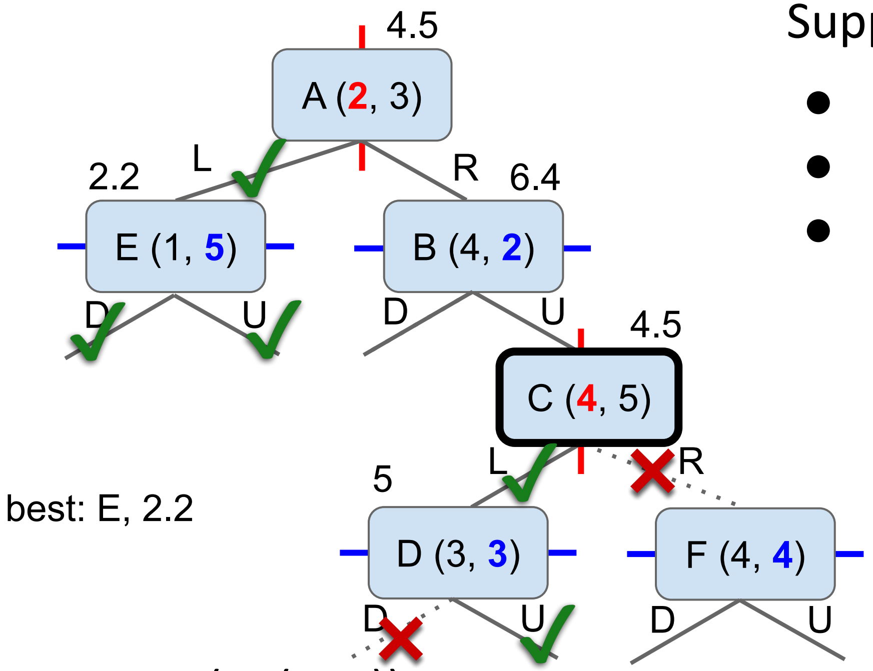
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



nearest(C, (0, 7))

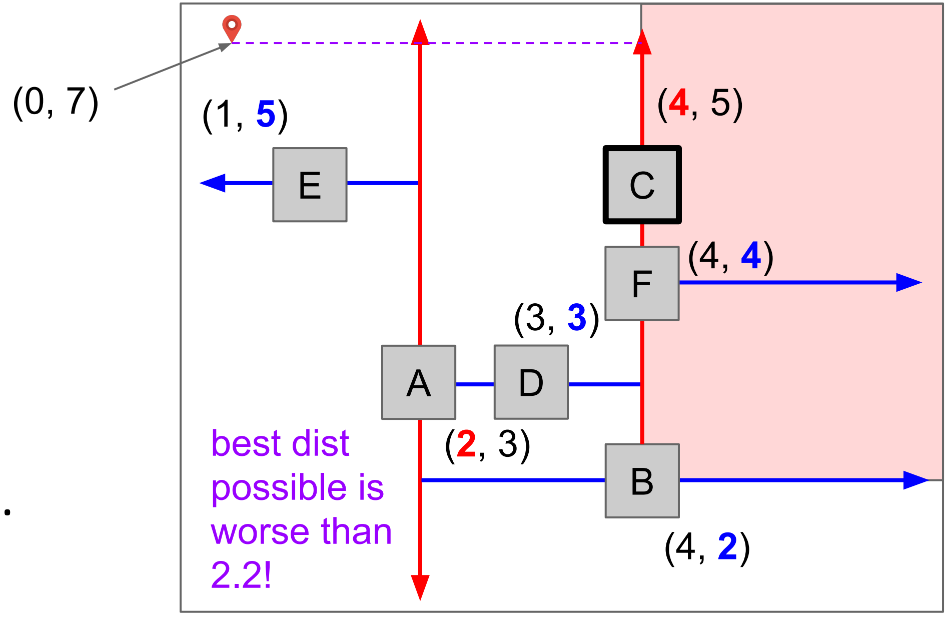
- … just finished exploring the good side of C.
- Could something better be on the “bad” side of the line, i.e. C.right? No!

K-d Nearest Demo



Suppose we have the k-d tree shown.

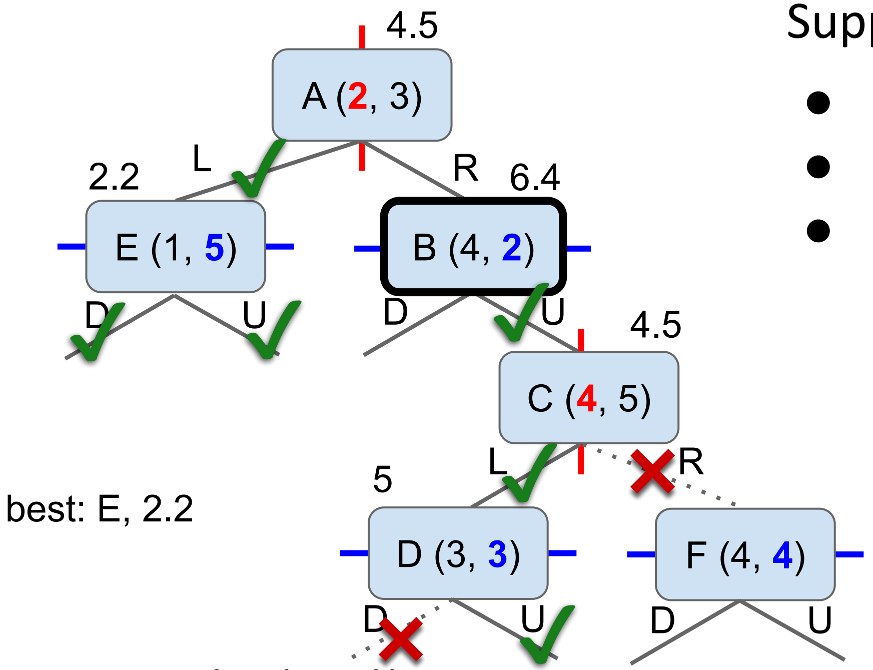
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(C, (0, 7))

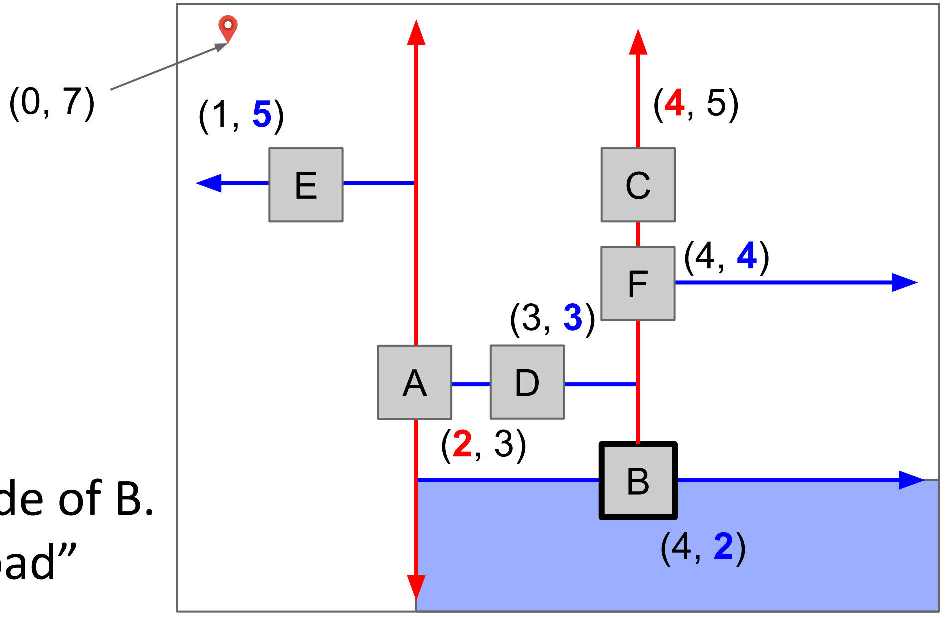
- Explored good side, pruned bad side.
- All done with C, so let's go back up.

K-d Nearest Demo



Suppose we have the k-d tree shown.

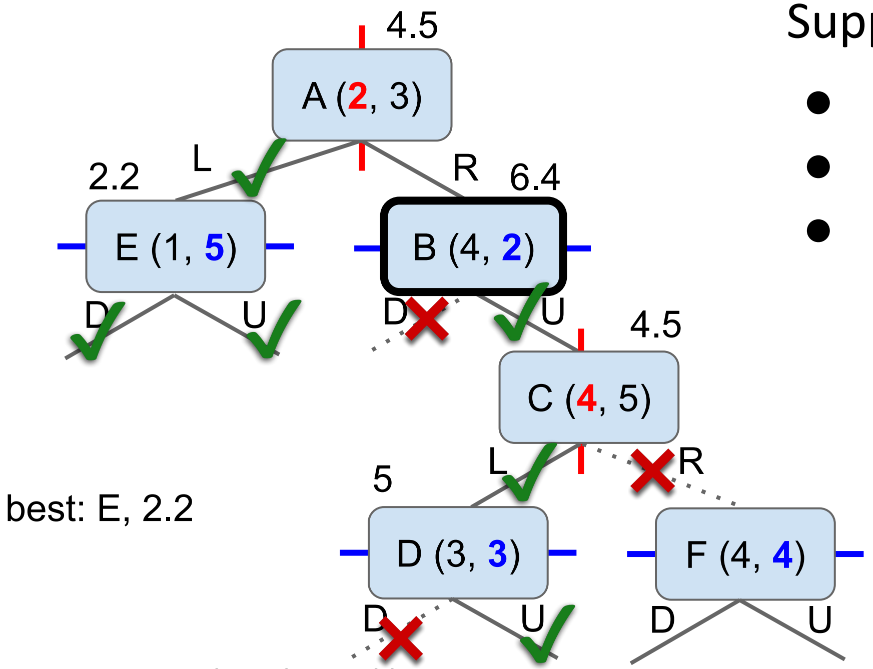
- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.



nearest(B, (0, 7))

- …just finished exploring the good side of B.
- Could something better be on the "bad" side of the line, i.e. B.down?

K-d Nearest Demo

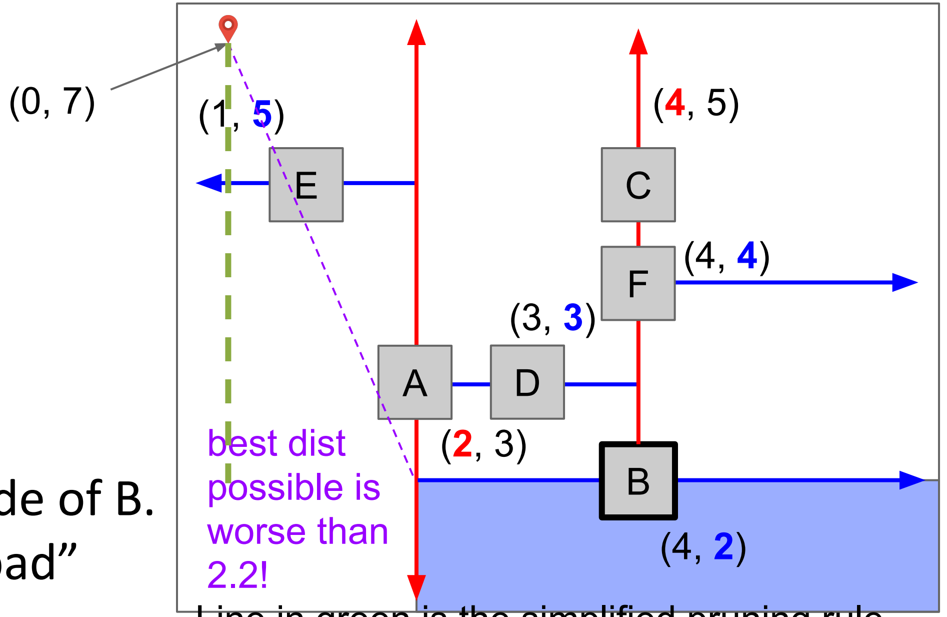


nearest(B, (0, 7))

- …just finished exploring the good side of B.
- Could something better be on the "bad" side of the line, i.e. B.down? No!

Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.

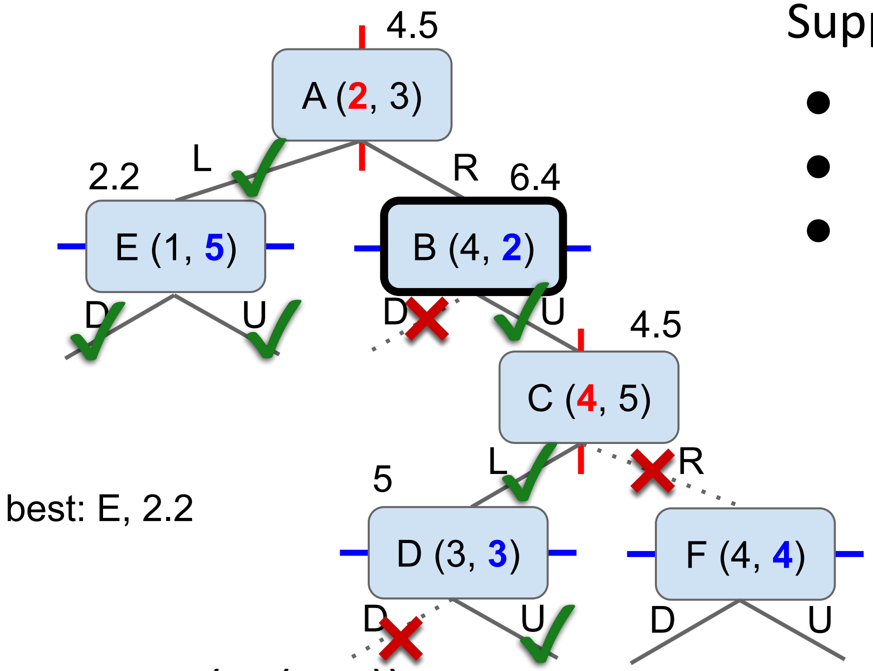


Line in green is the simplified pruning rule.

K-d Nearest Demo

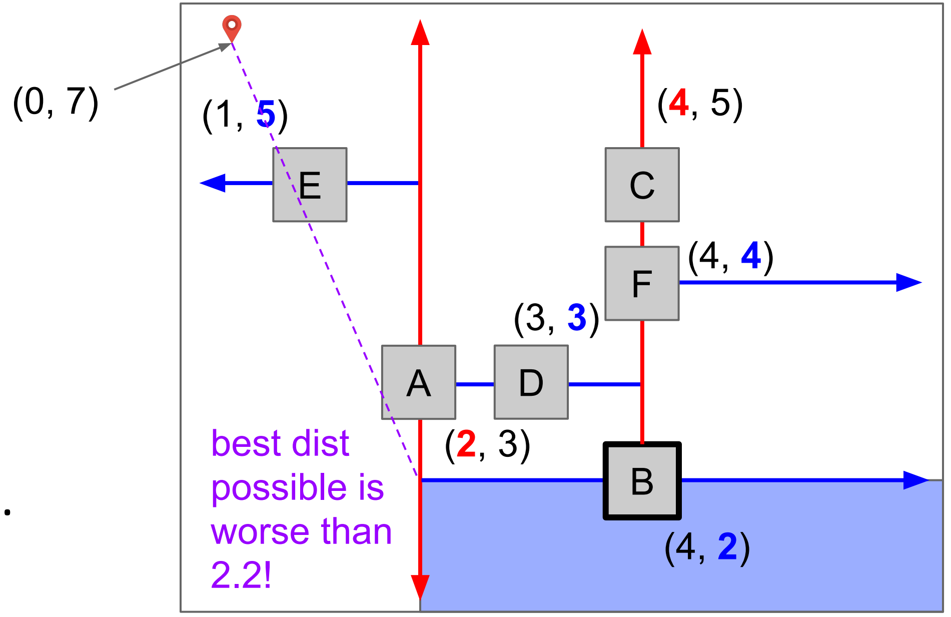
Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.

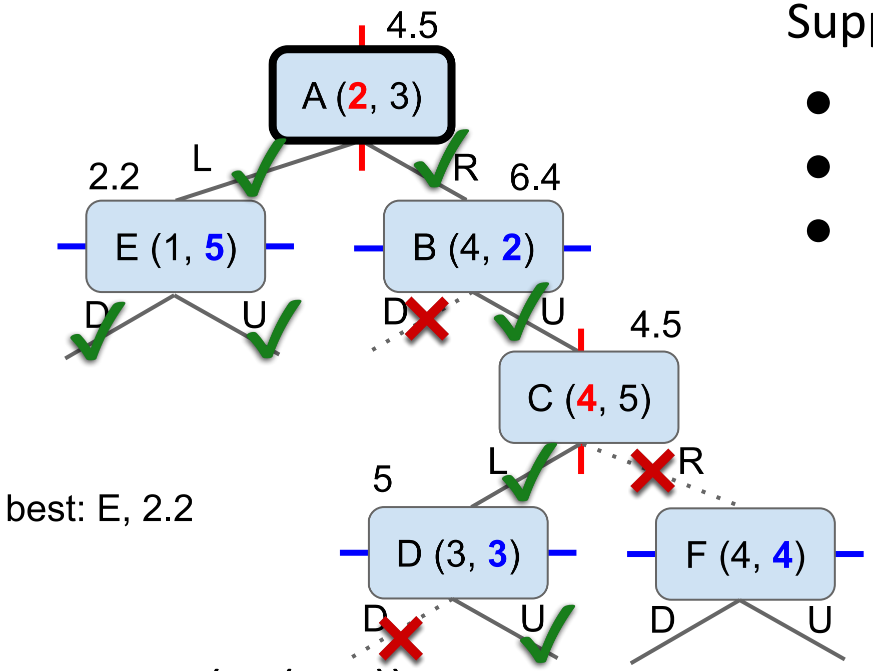


nearest(B, (0, 7))

- Explored good side, pruned bad side.
- All done with B, so let's go back up.



K-d Nearest Demo

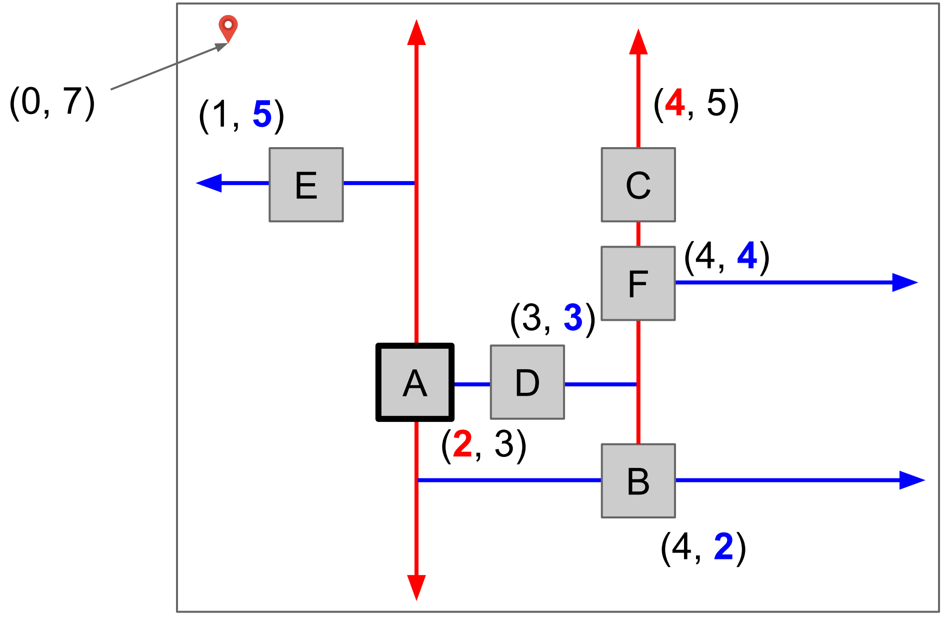


nearest(A, (0, 7))

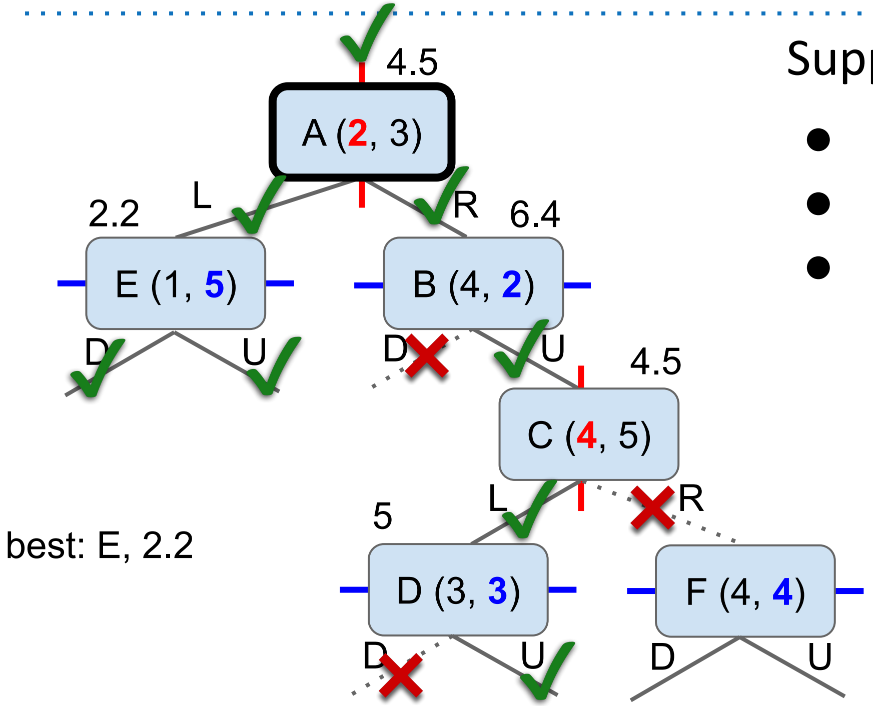
- Explored good side AND bad side.
- All done, so let's go back up.

Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let's do a proper k-d tree traversal.

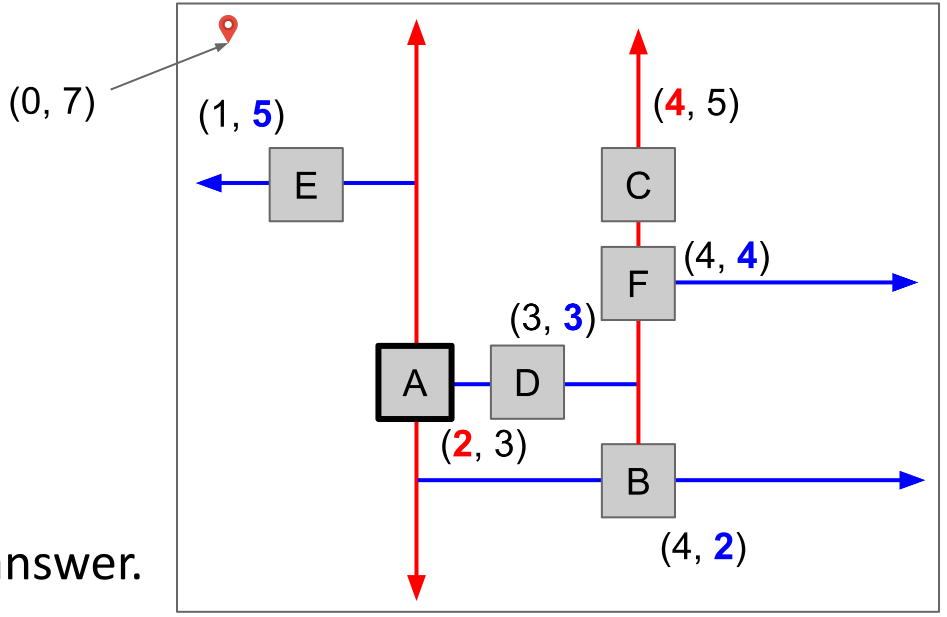


K-d Nearest Demo



Suppose we have the k-d tree shown.

- We want to find nearest((0, 7)).
- Can visually see the answer is (1, 5).
- Let’s do a proper k-d tree traversal.



All done, and best we found was E.

- Dashed lines are unexplored links.
- Guaranteed not to contain a better answer.