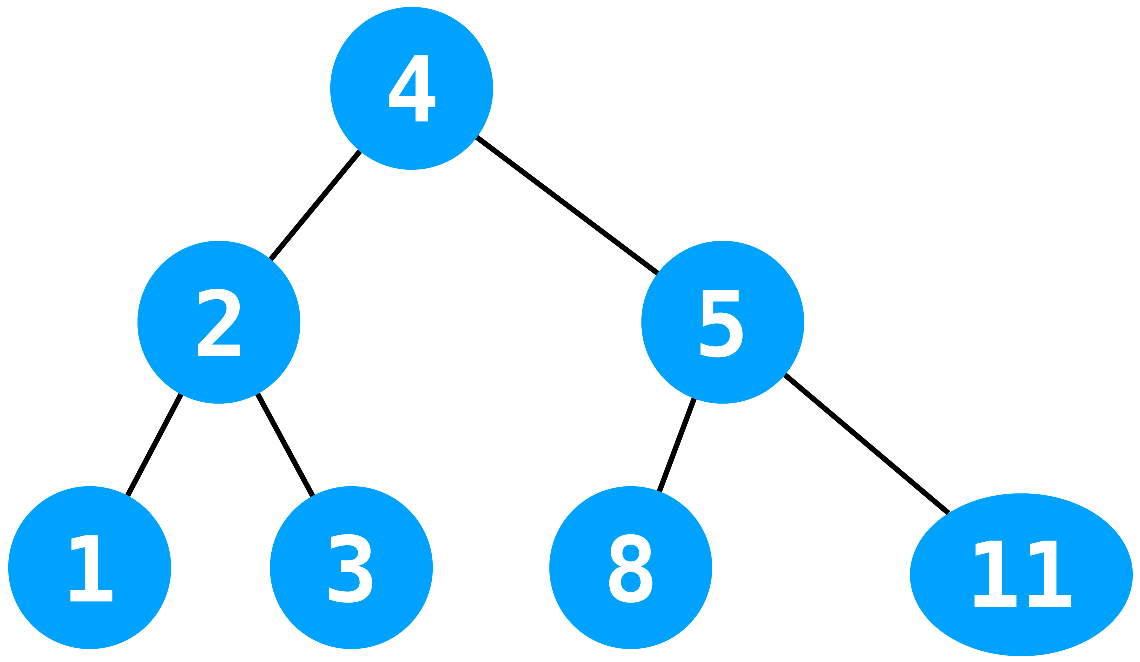
CS 61BL Lab 17

Ryan Purpura

k-d trees

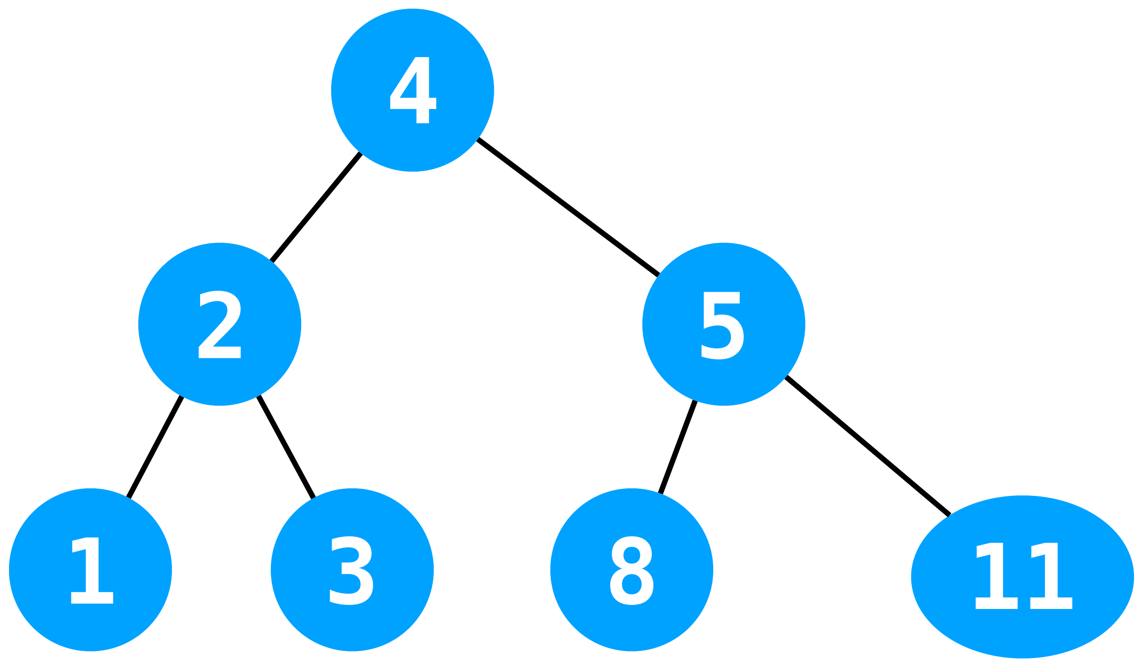
- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

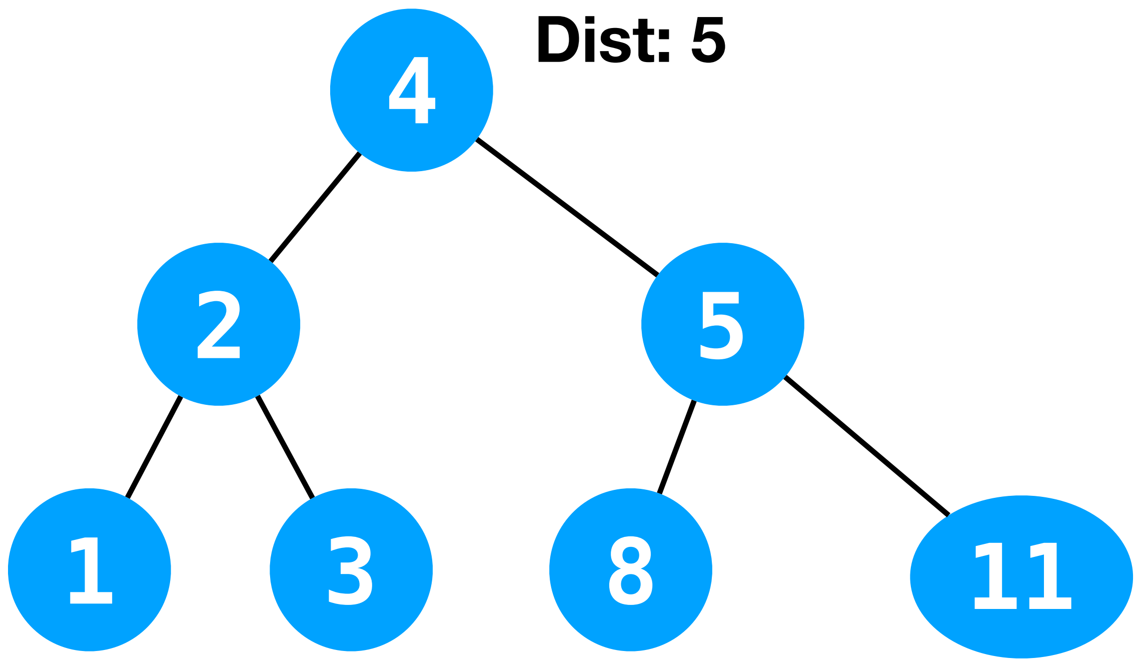
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

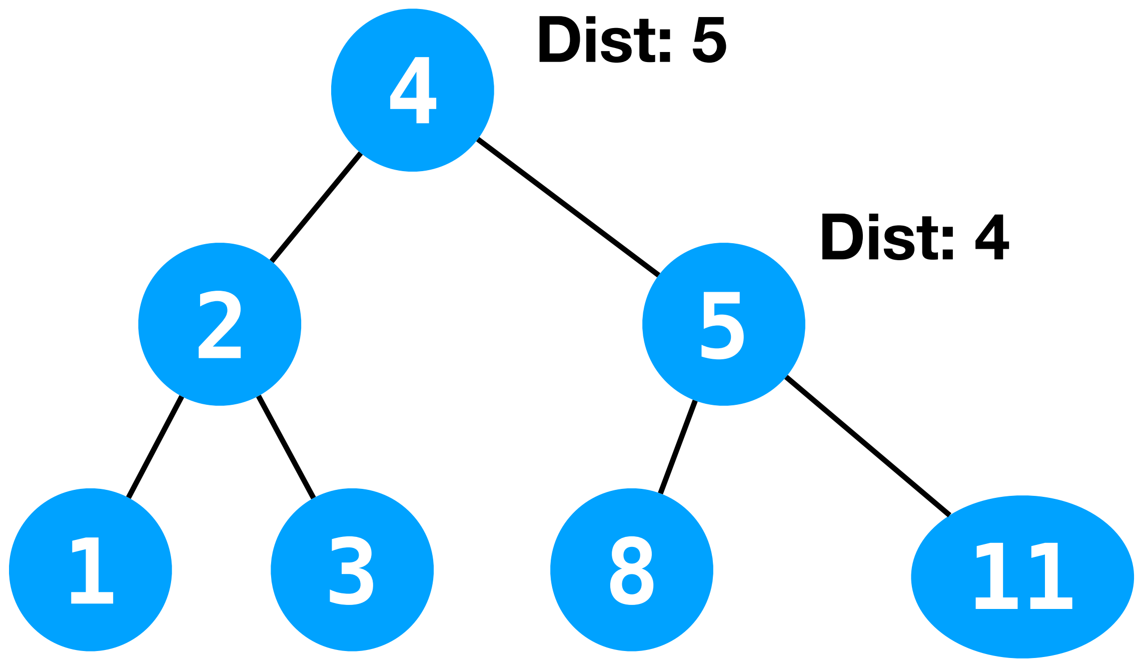
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

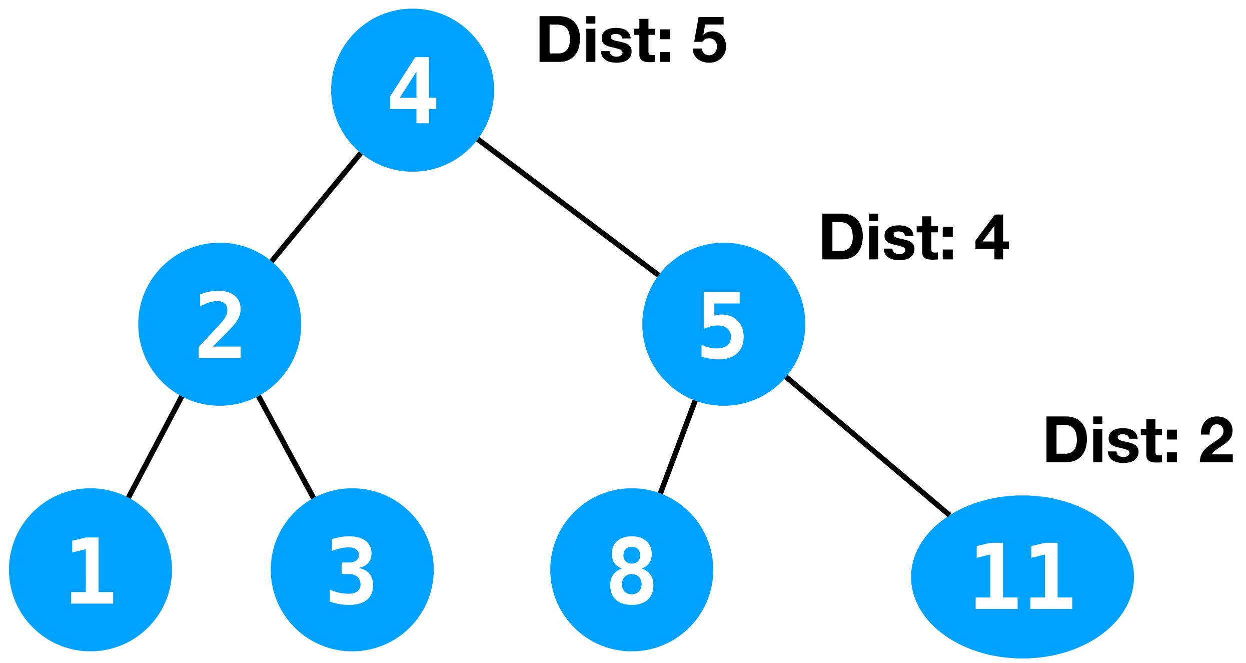
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

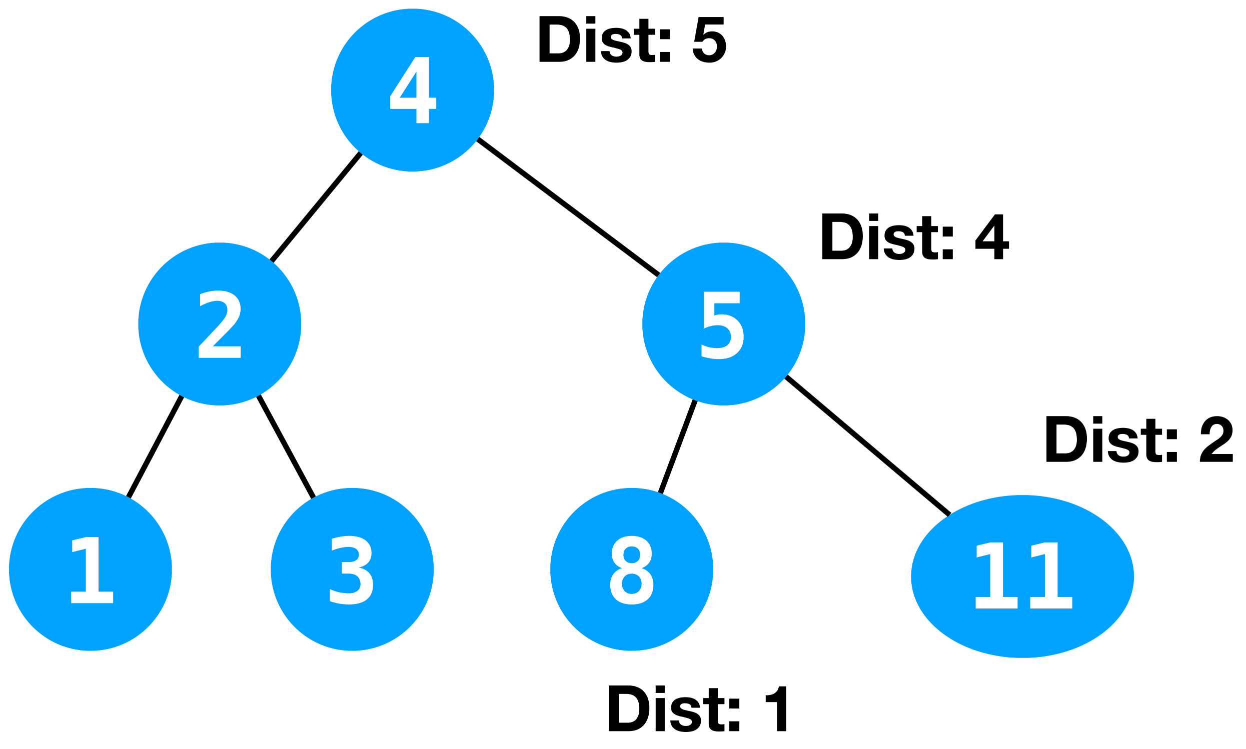
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

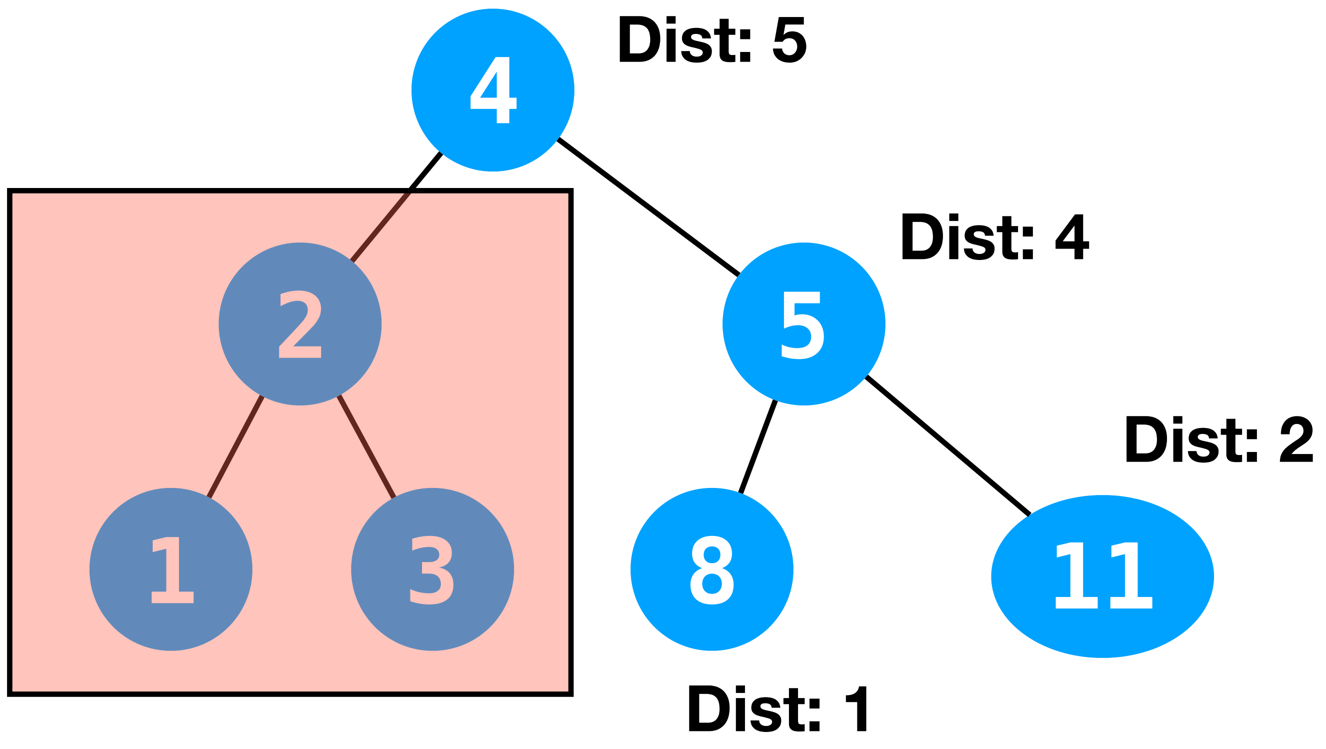
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

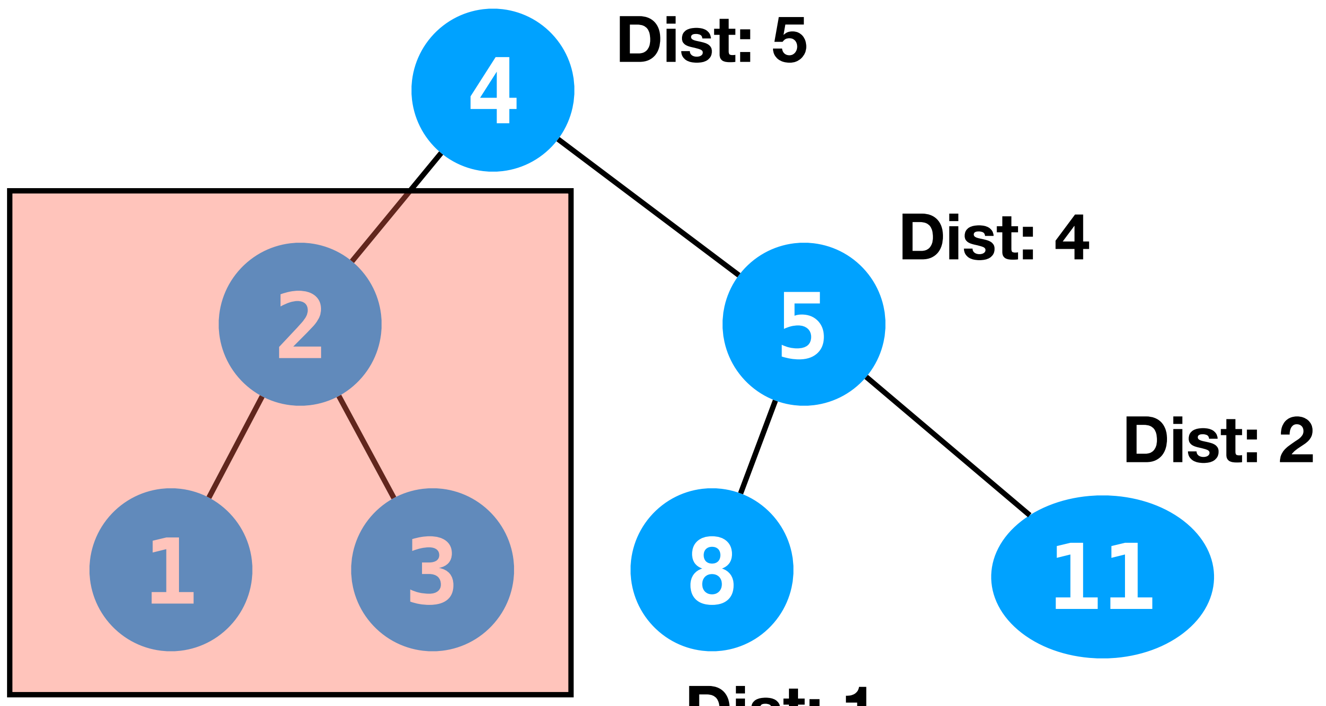
How would you find the node with value closest to 9?



k-d trees

- Previously, we've been using binary search trees to organize sortable data.
- Binary search trees are excellent for range operations (e.g. get all nodes greater than 50 and less than 100) and fuzzy lookups (e.g. give me the closest node with value closest to 72).

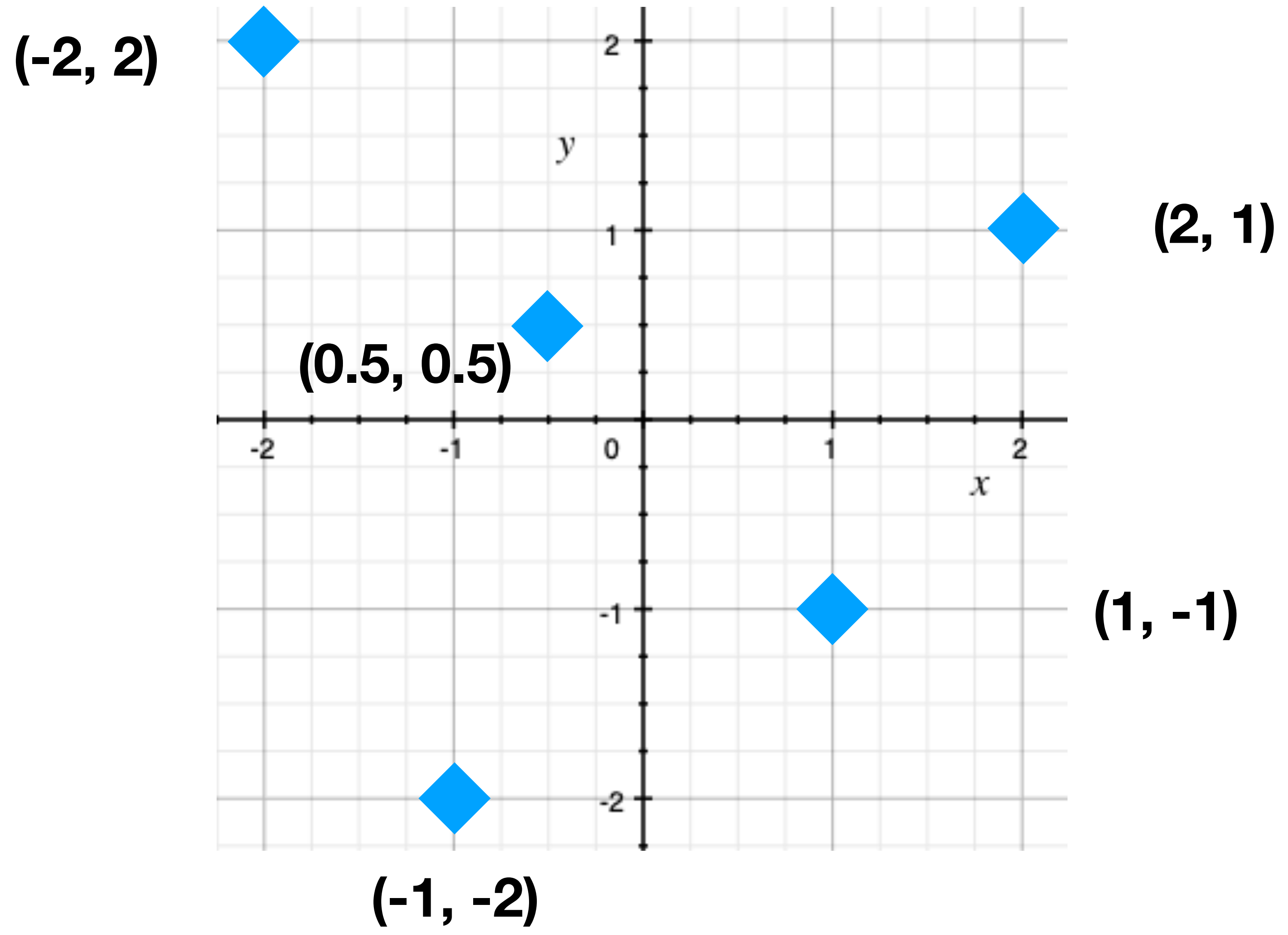
How would you find the node with value closest to 9?



This side of the tree is pruned from our search!

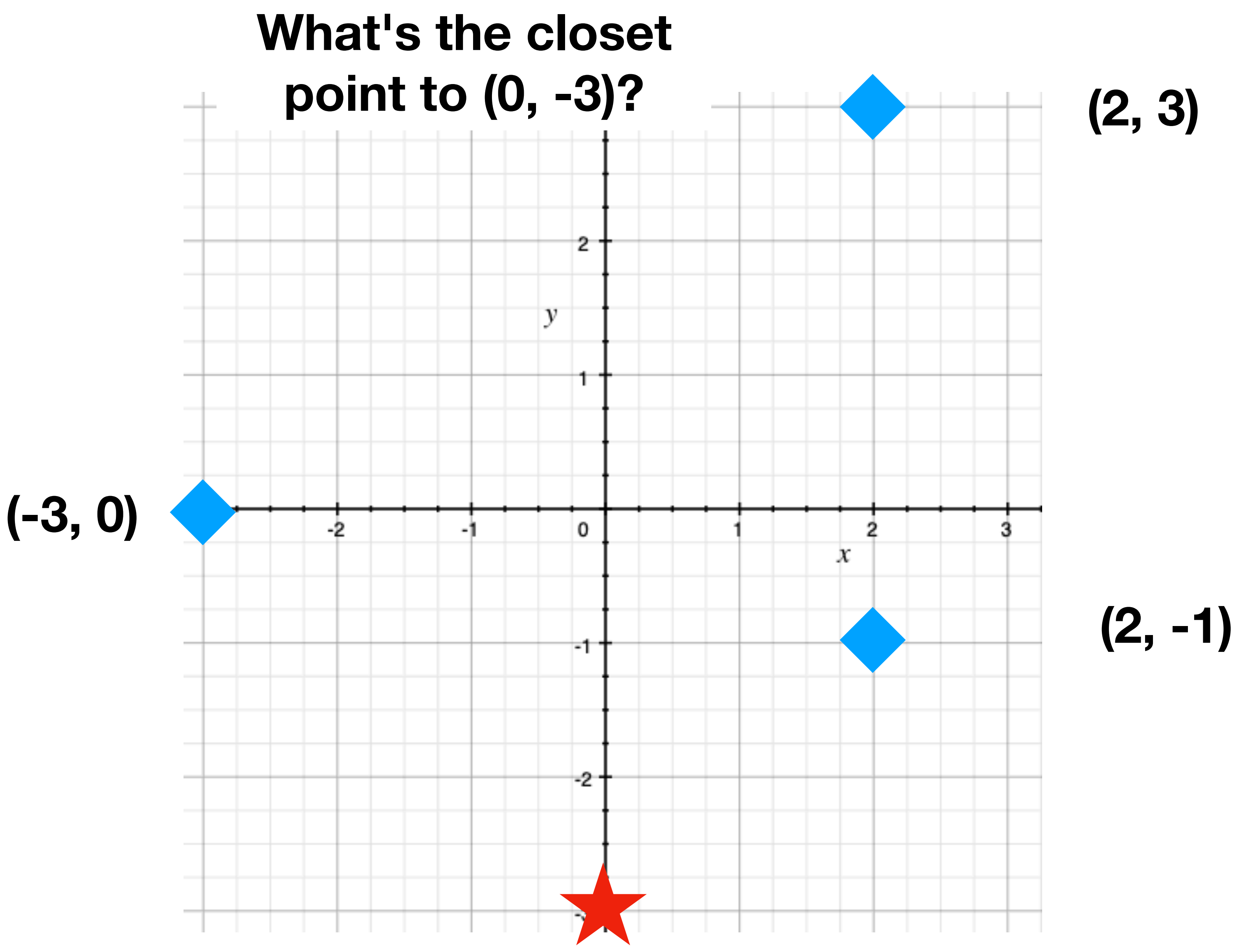
The Problem

- Not all data is one-dimensional.



Two-Dimensional Trickery

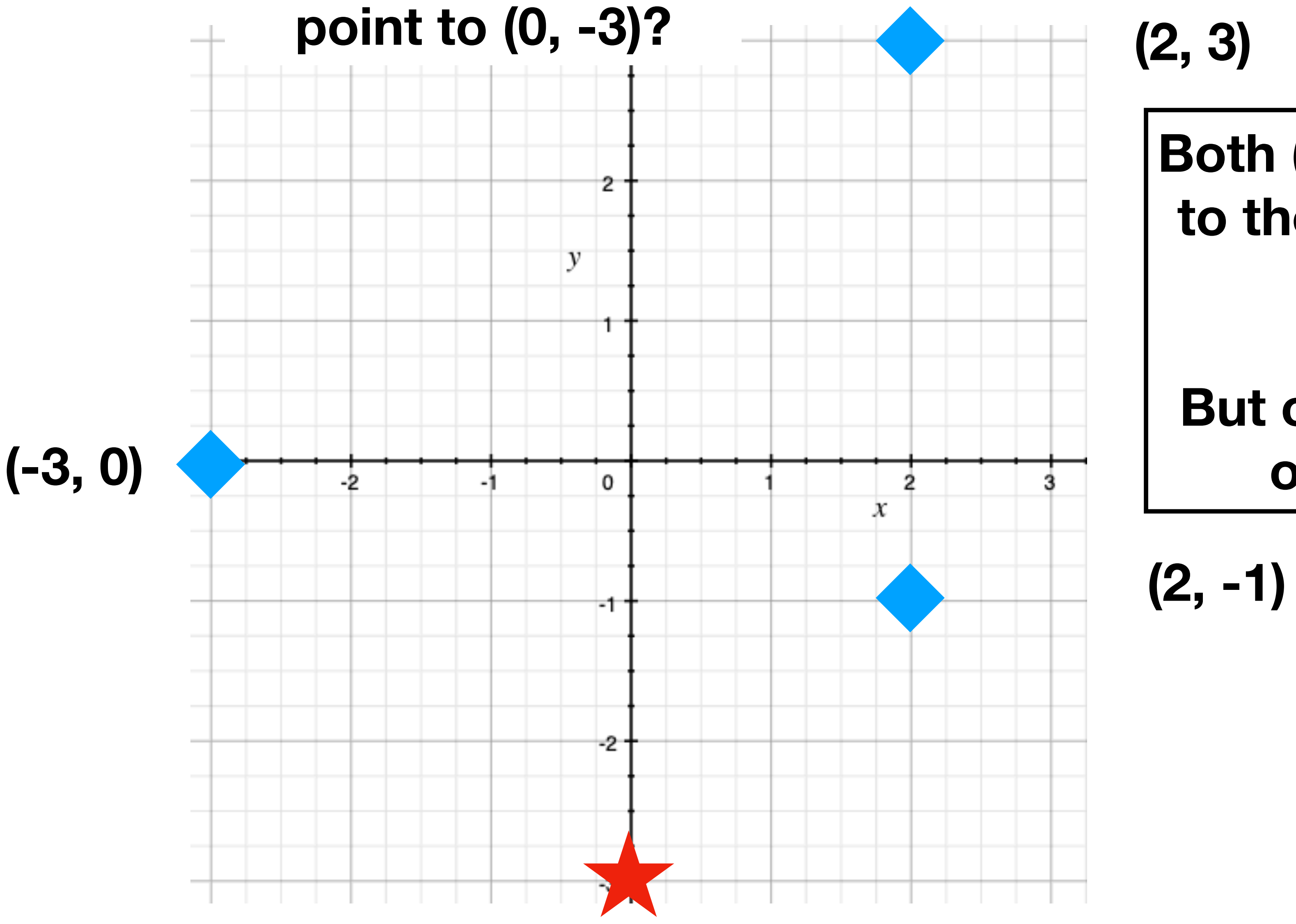
- Going closer in one dimension does not necessarily you get closer to your target, since you might go even farther in another dimension.



Two-Dimensional Trickery

- Going closer in one dimension does not necessarily you get closer to your target, since you might go even farther in another dimension.

What's the closet point to (0, -3)?

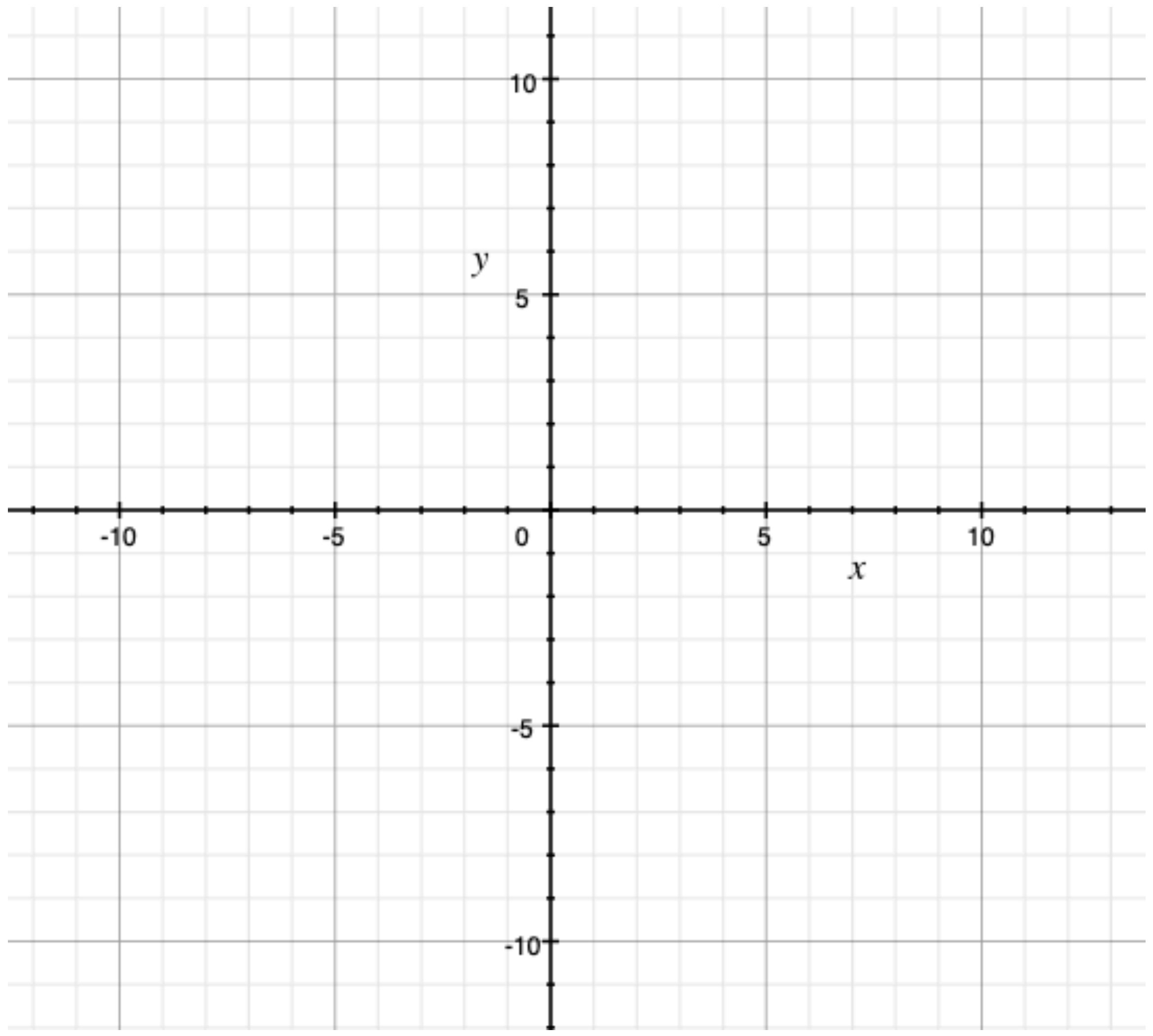


Both (2, 3) and (2, -1) are closer to the target in the x direction than (-3, 0).

But one of them is closer and one of them is farther!

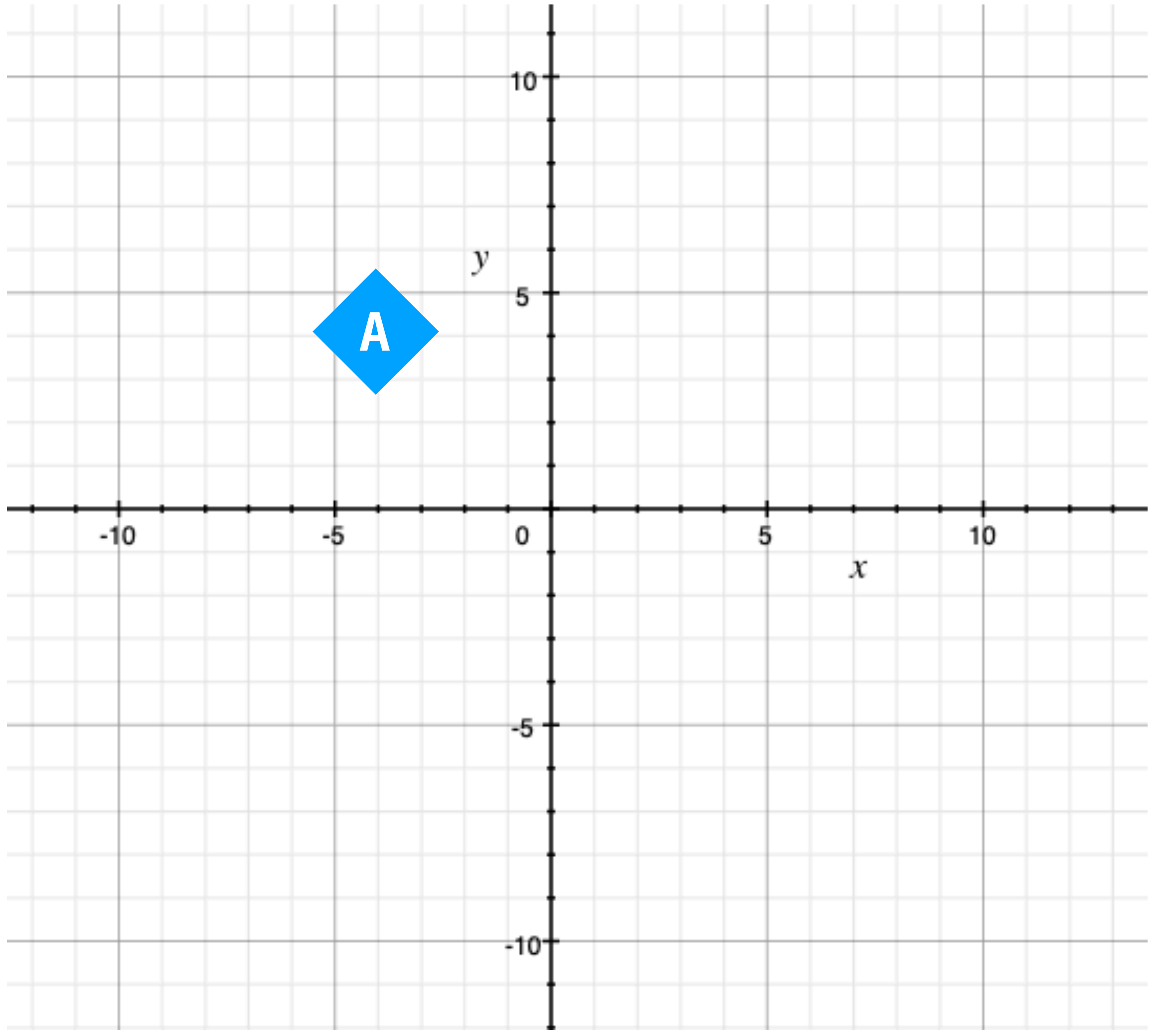
The Solution: k-d trees

- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.



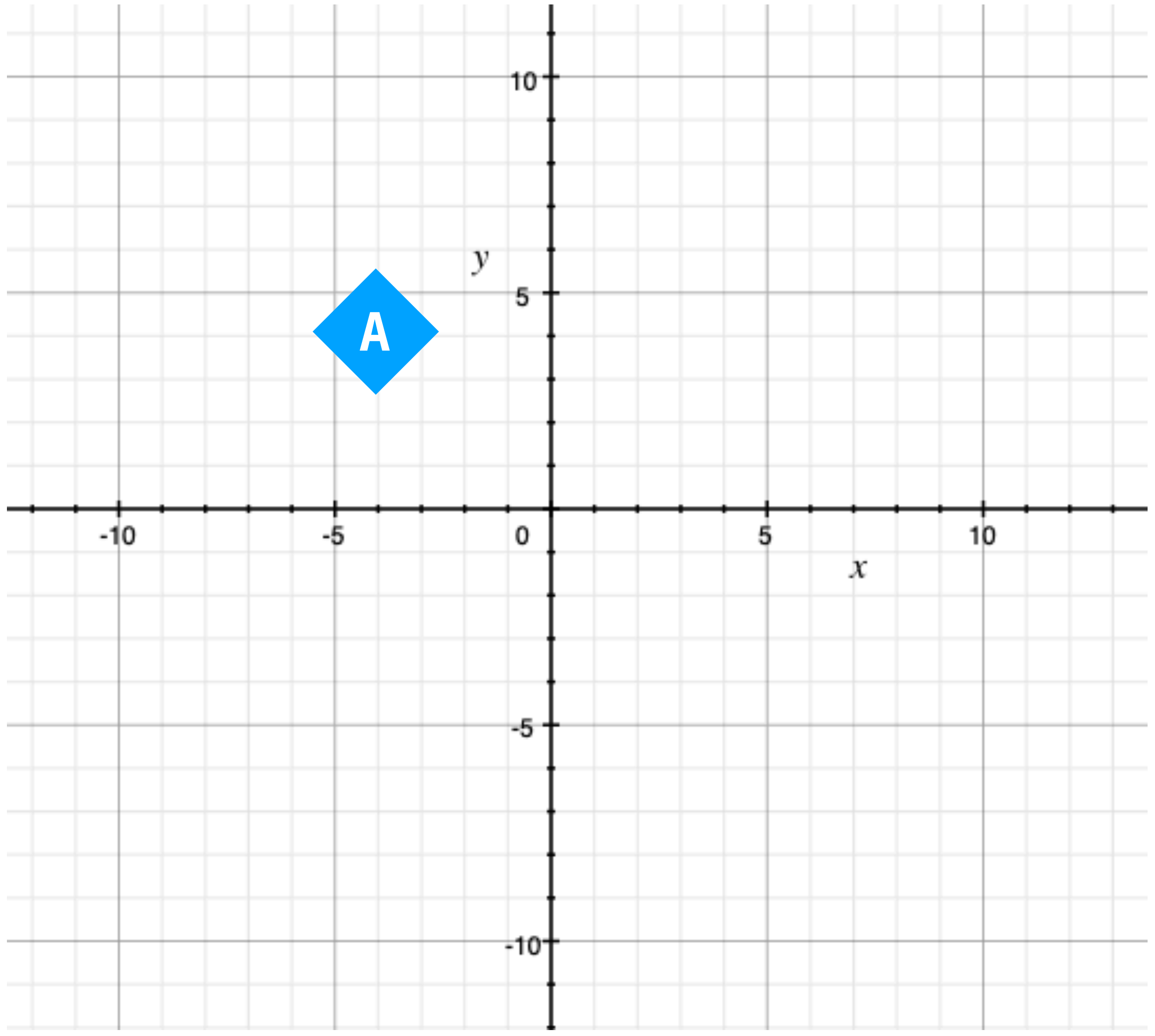
The Solution: k-d trees

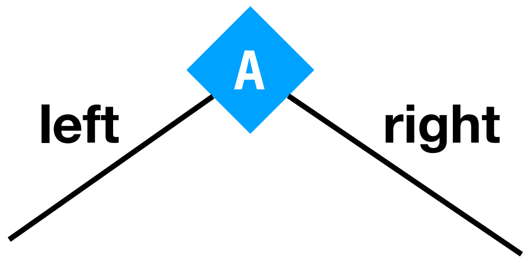
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.



The Solution: k-d trees

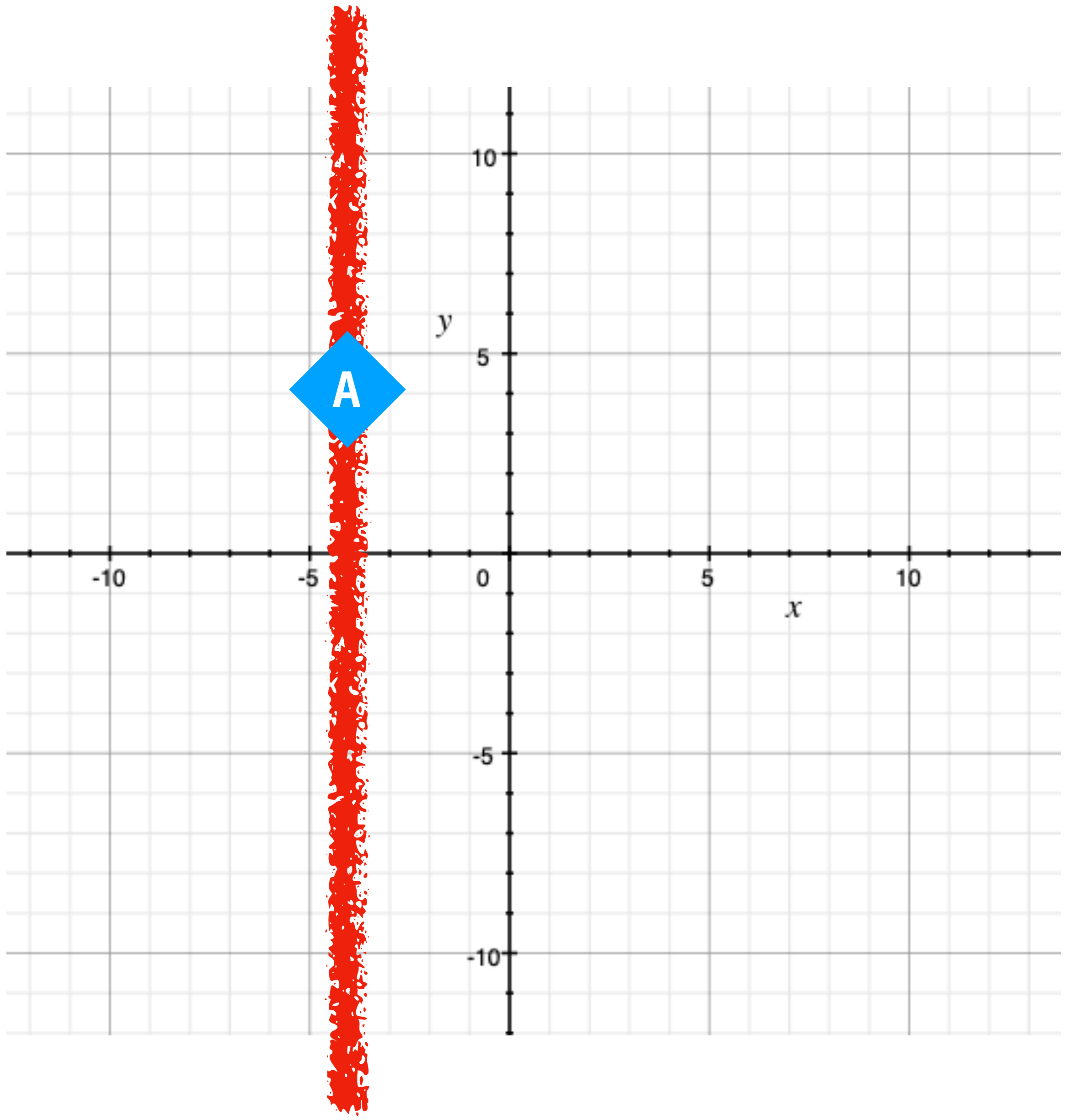
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

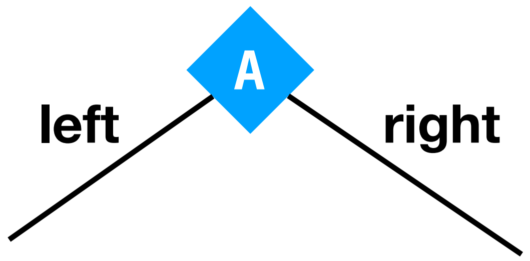




The Solution: k-d trees

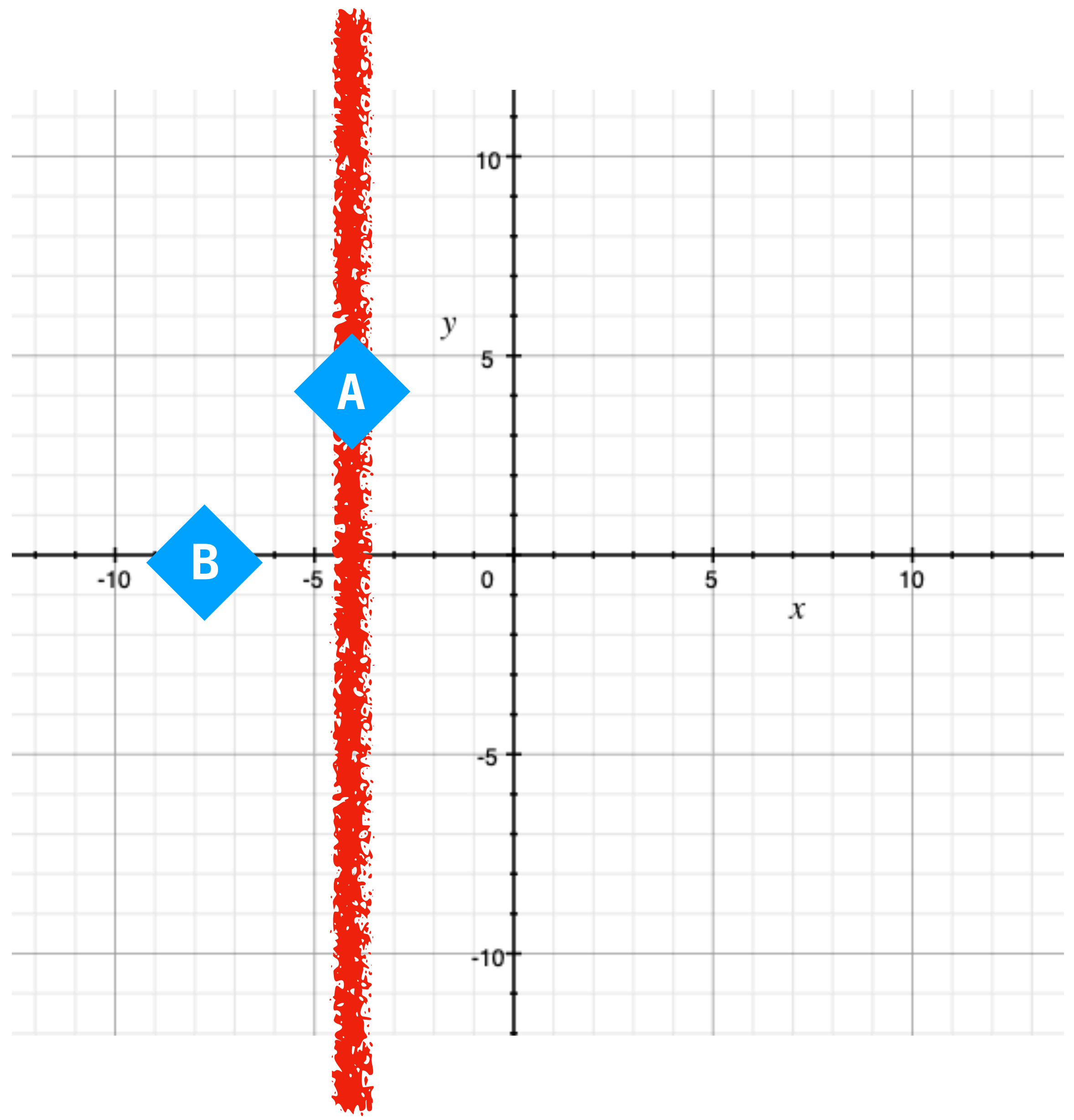
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

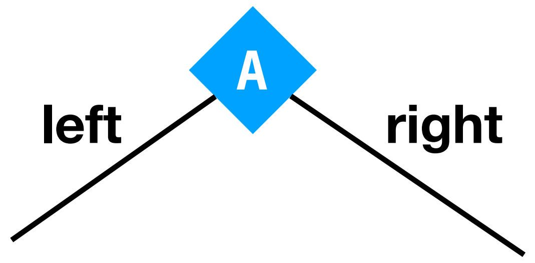




The Solution: k-d trees

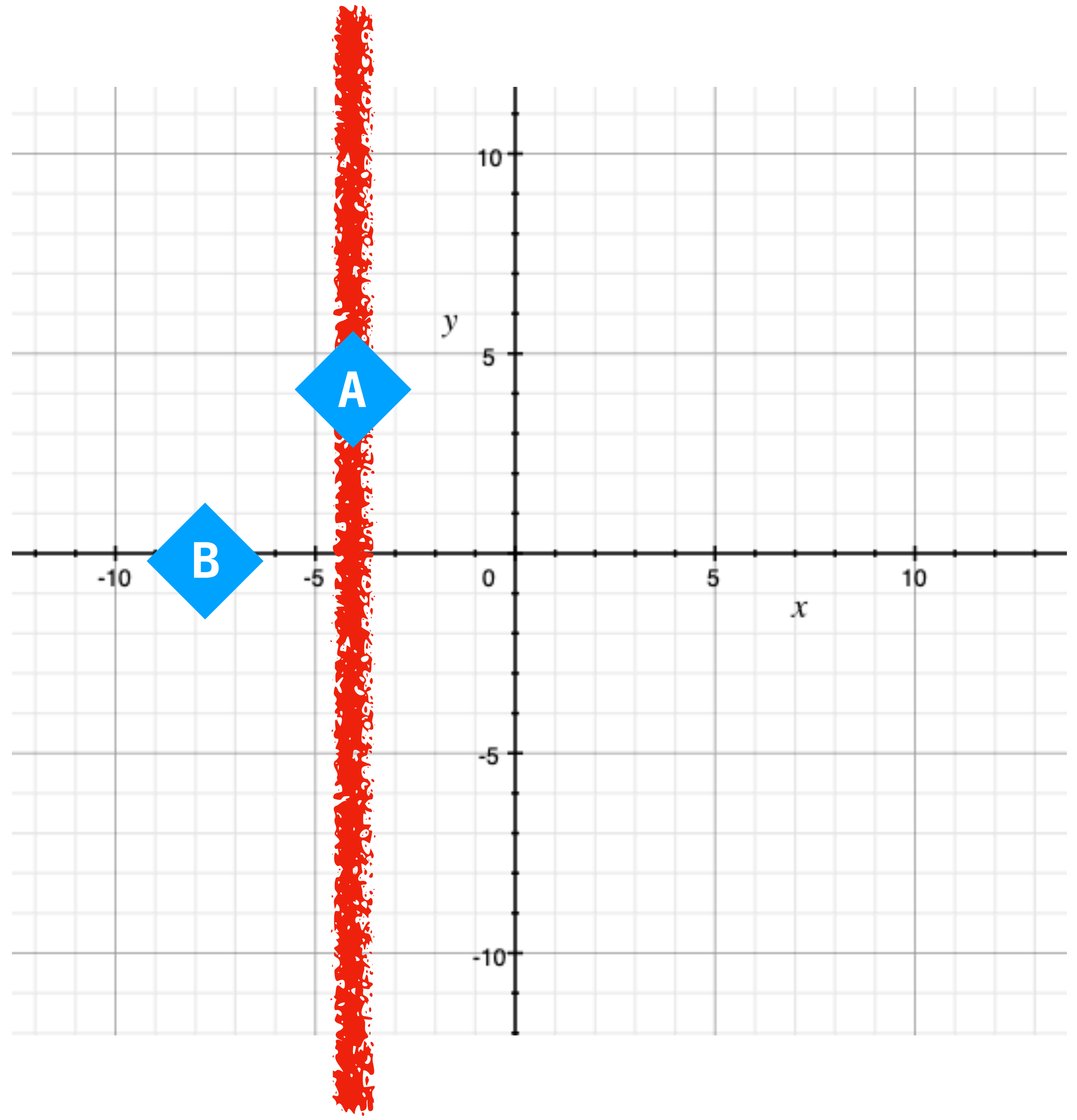
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

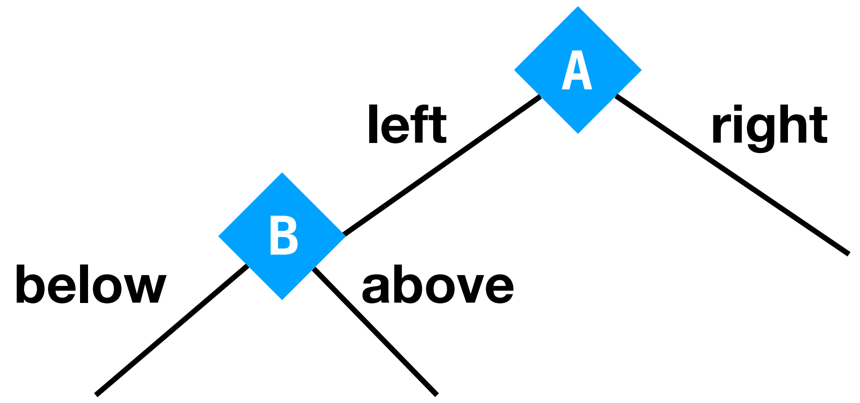




The Solution: k-d trees

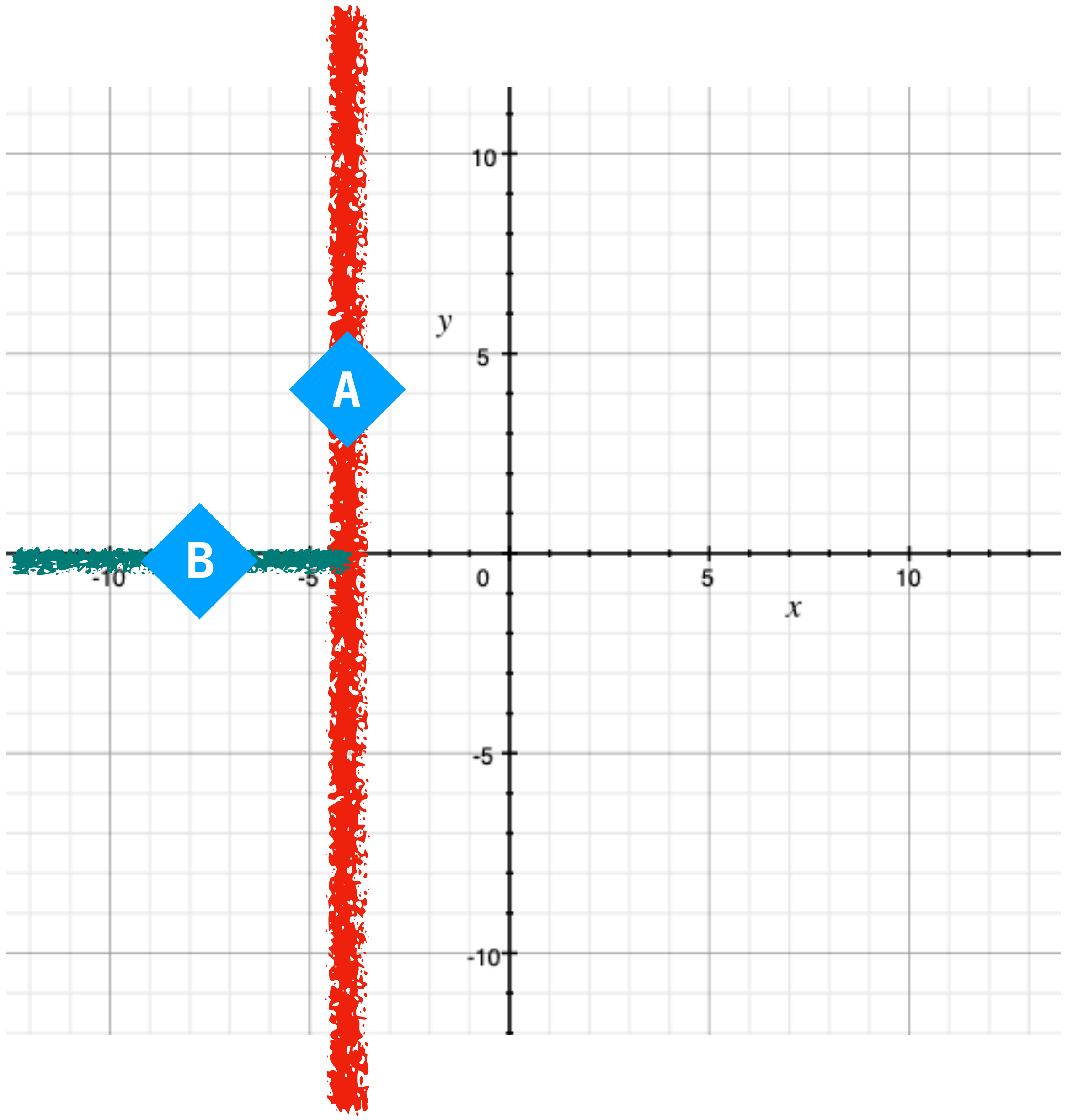
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

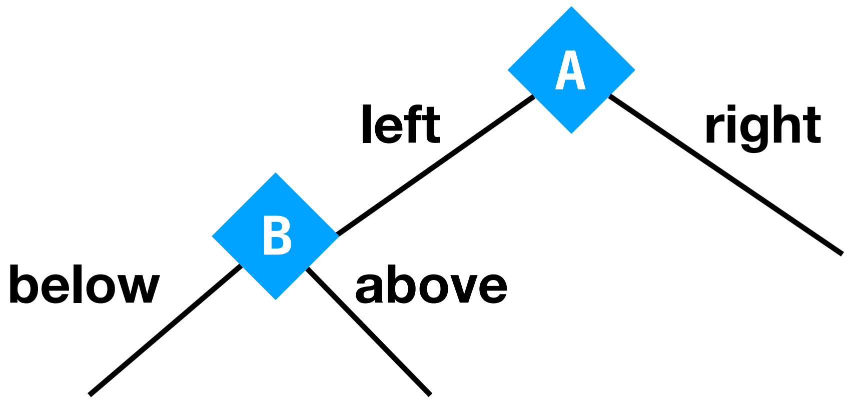




The Solution: k-d trees

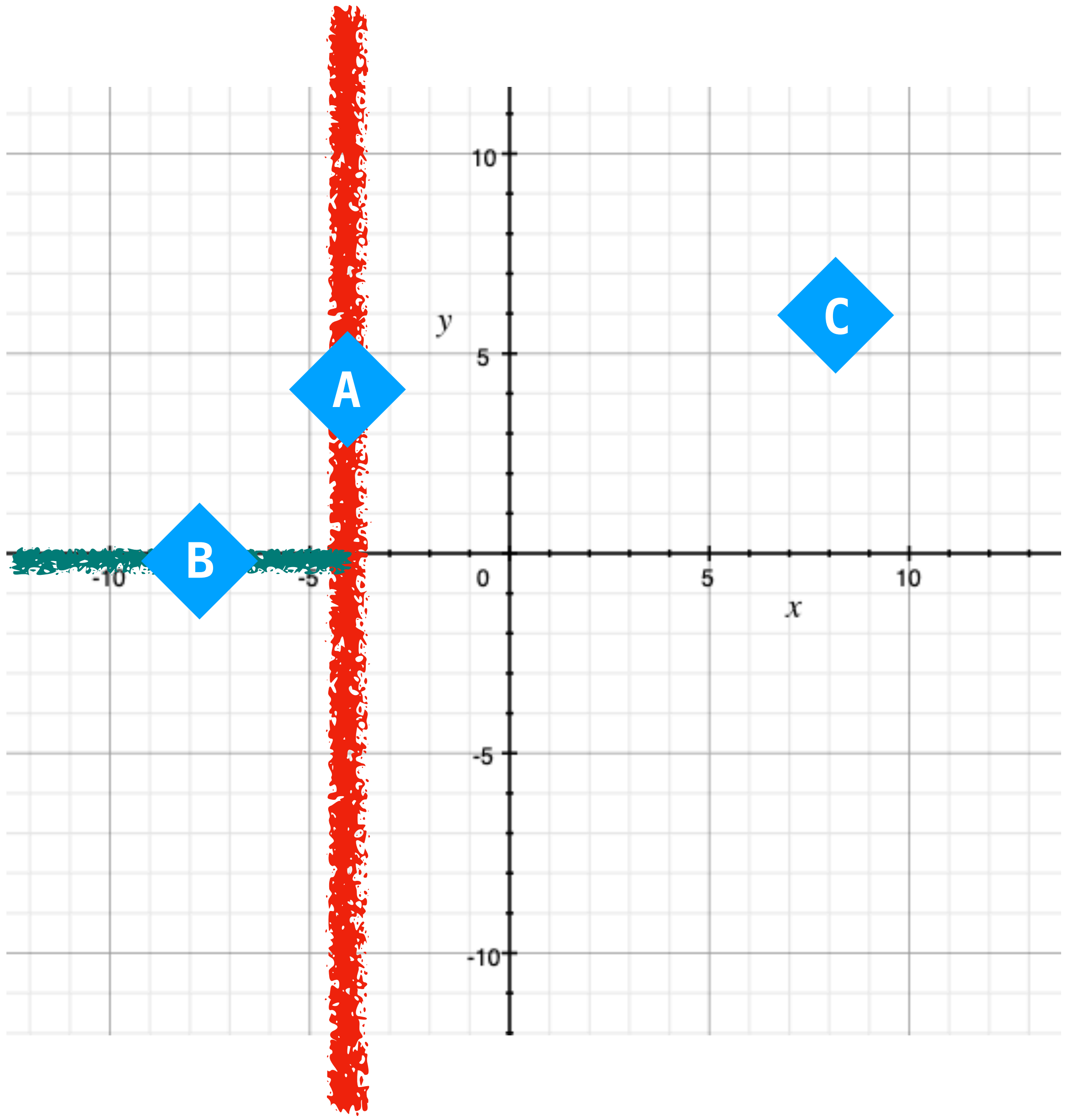
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

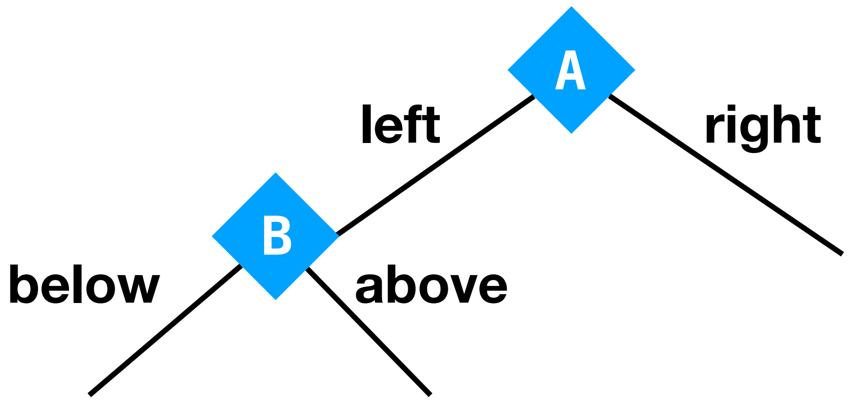




The Solution: k-d trees

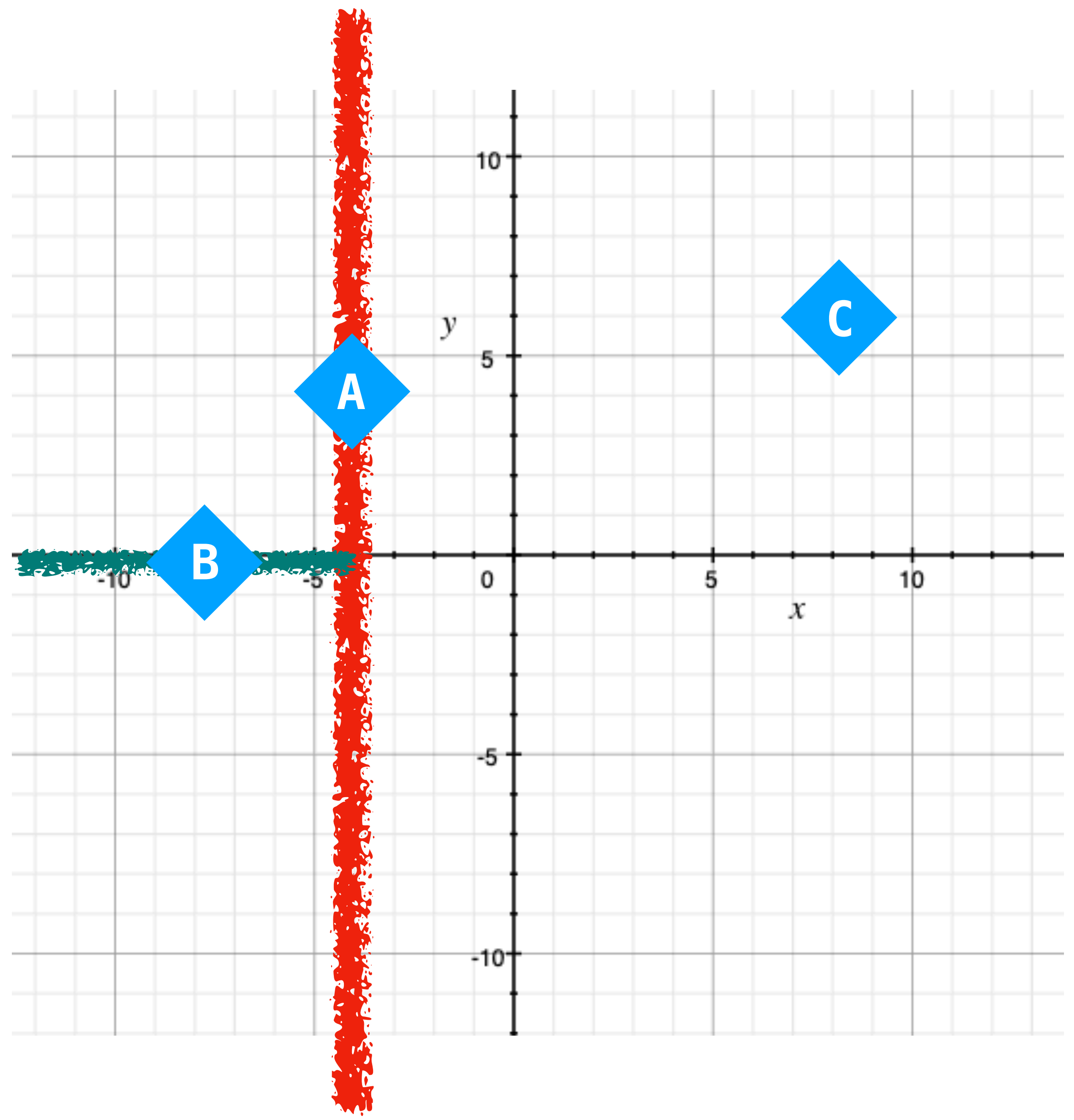
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

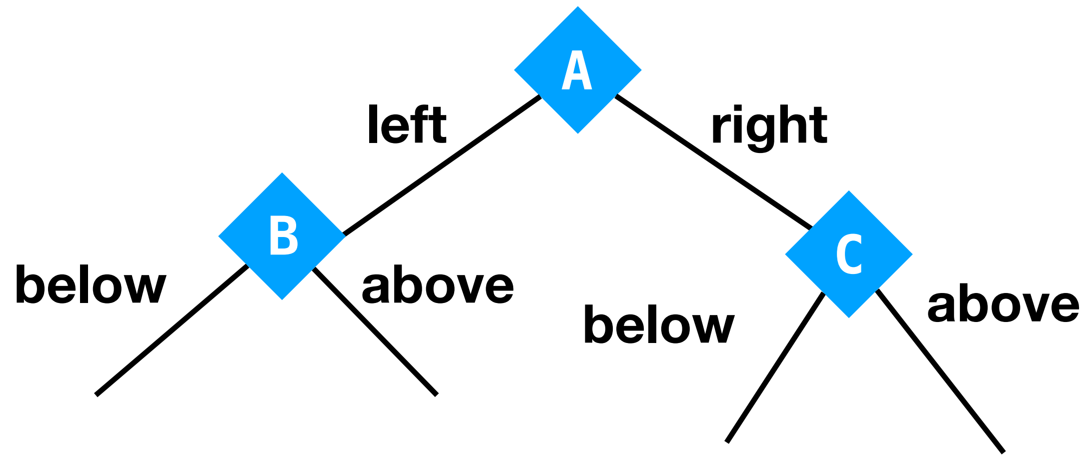




The Solution: k-d trees

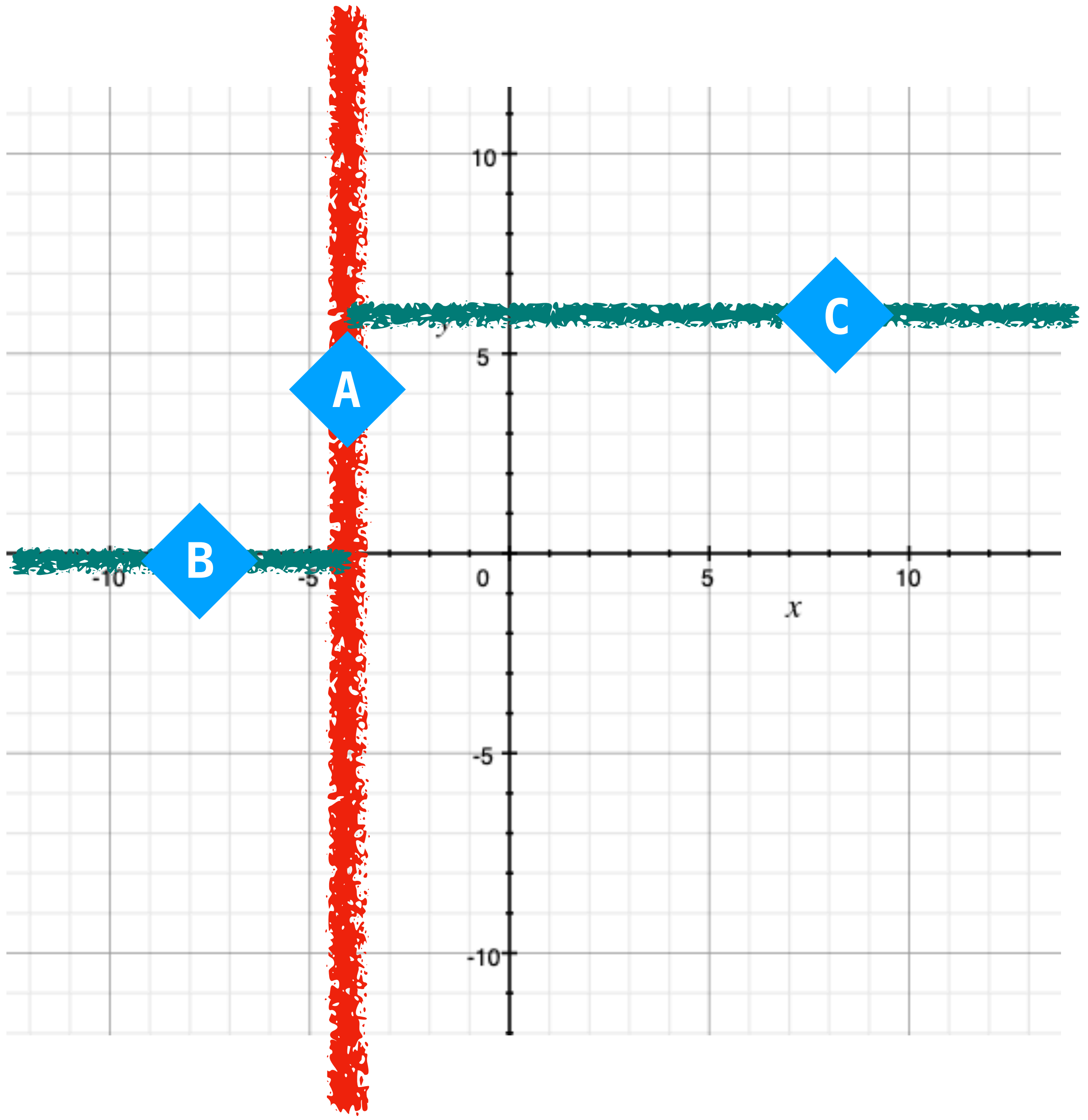
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

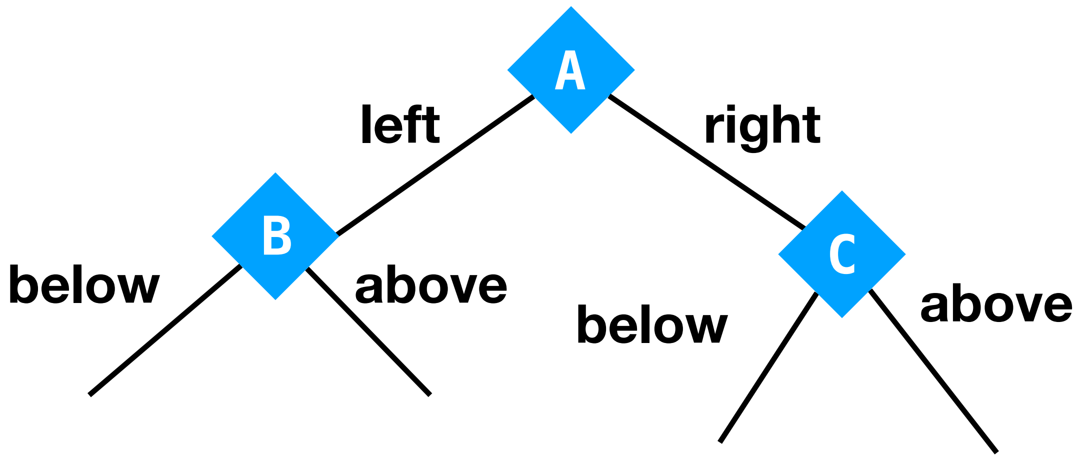




The Solution: k-d trees

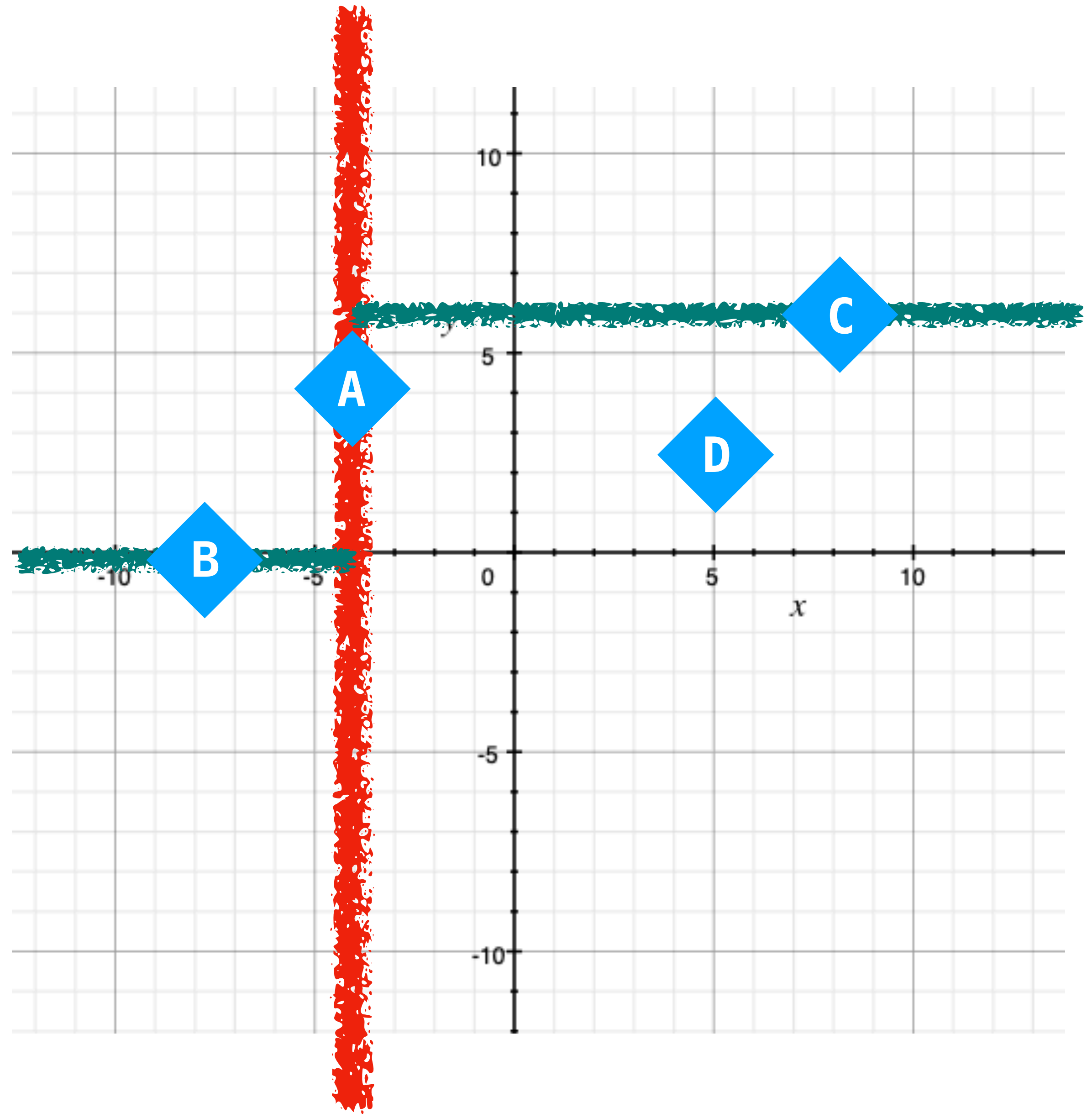
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

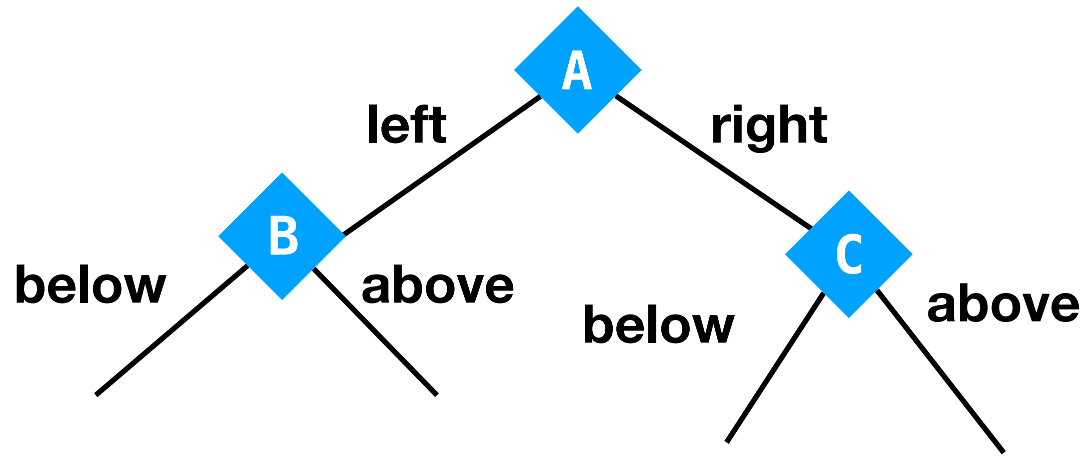




The Solution: k-d trees

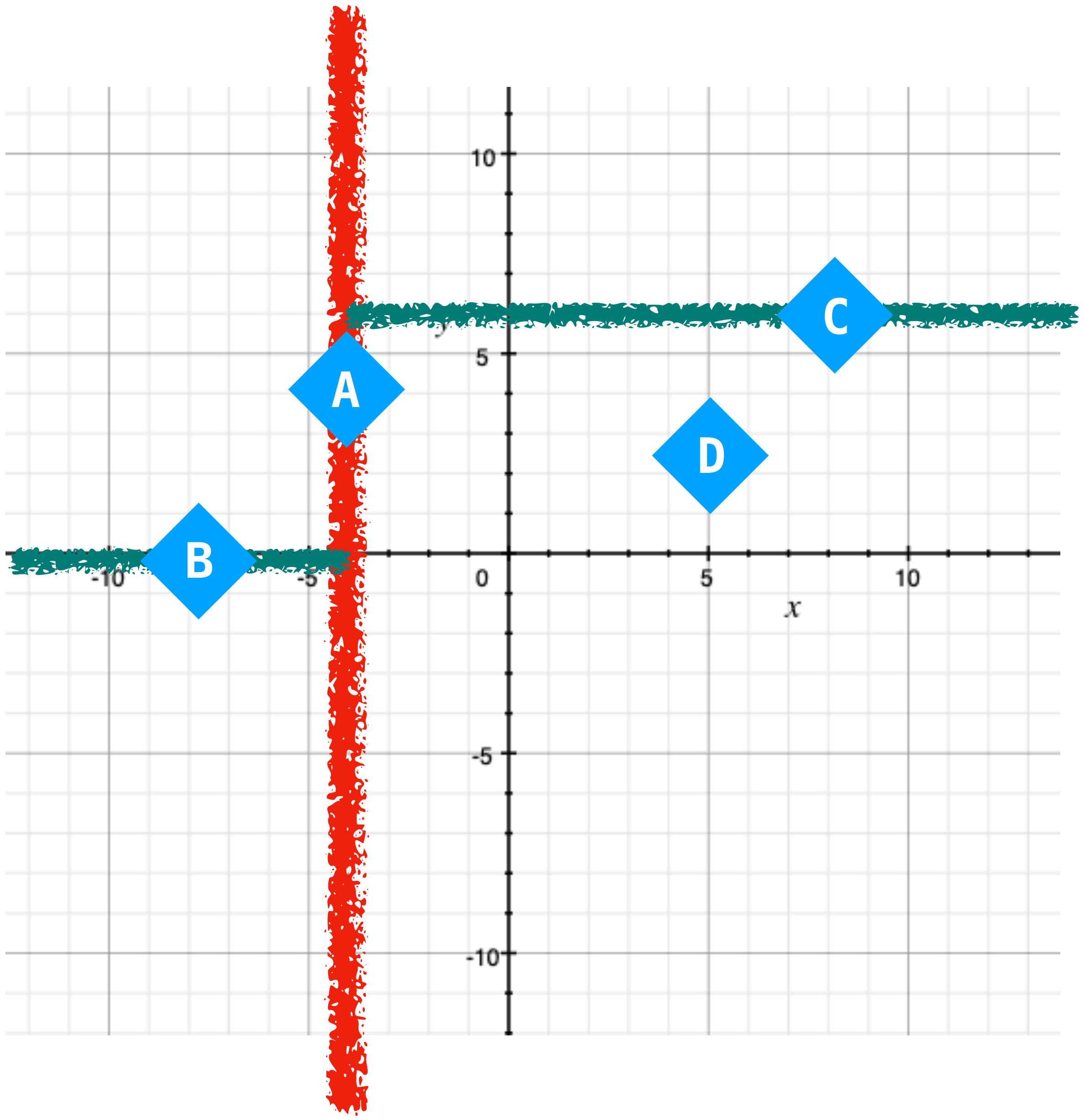
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

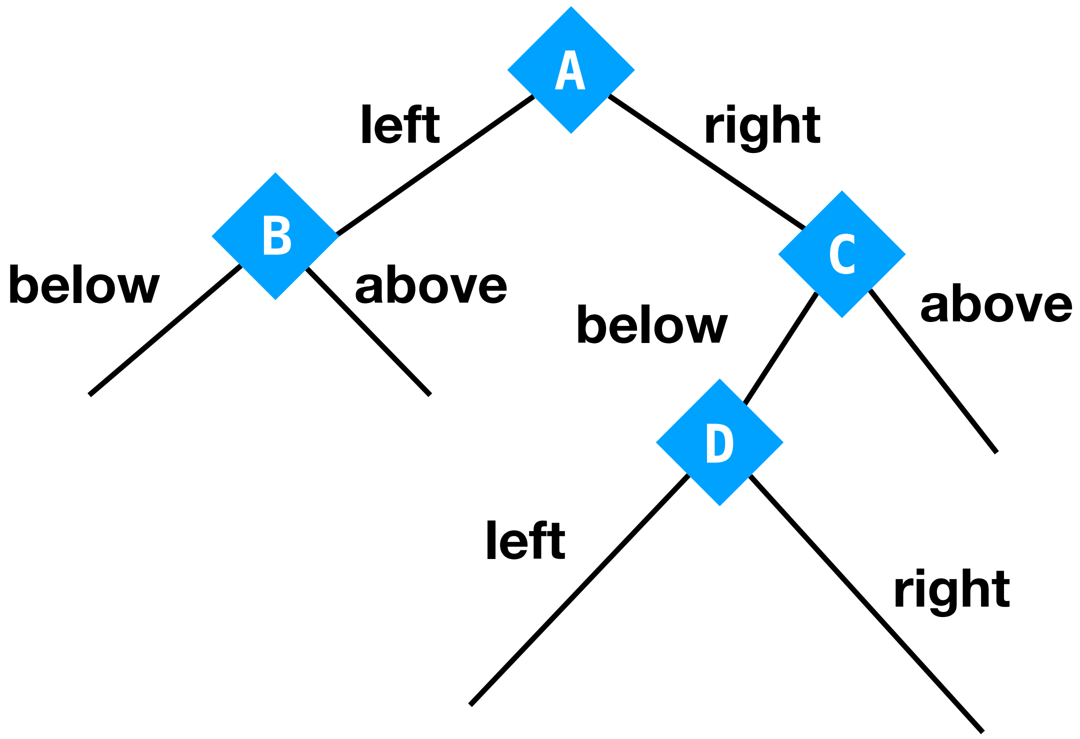




The Solution: k-d trees

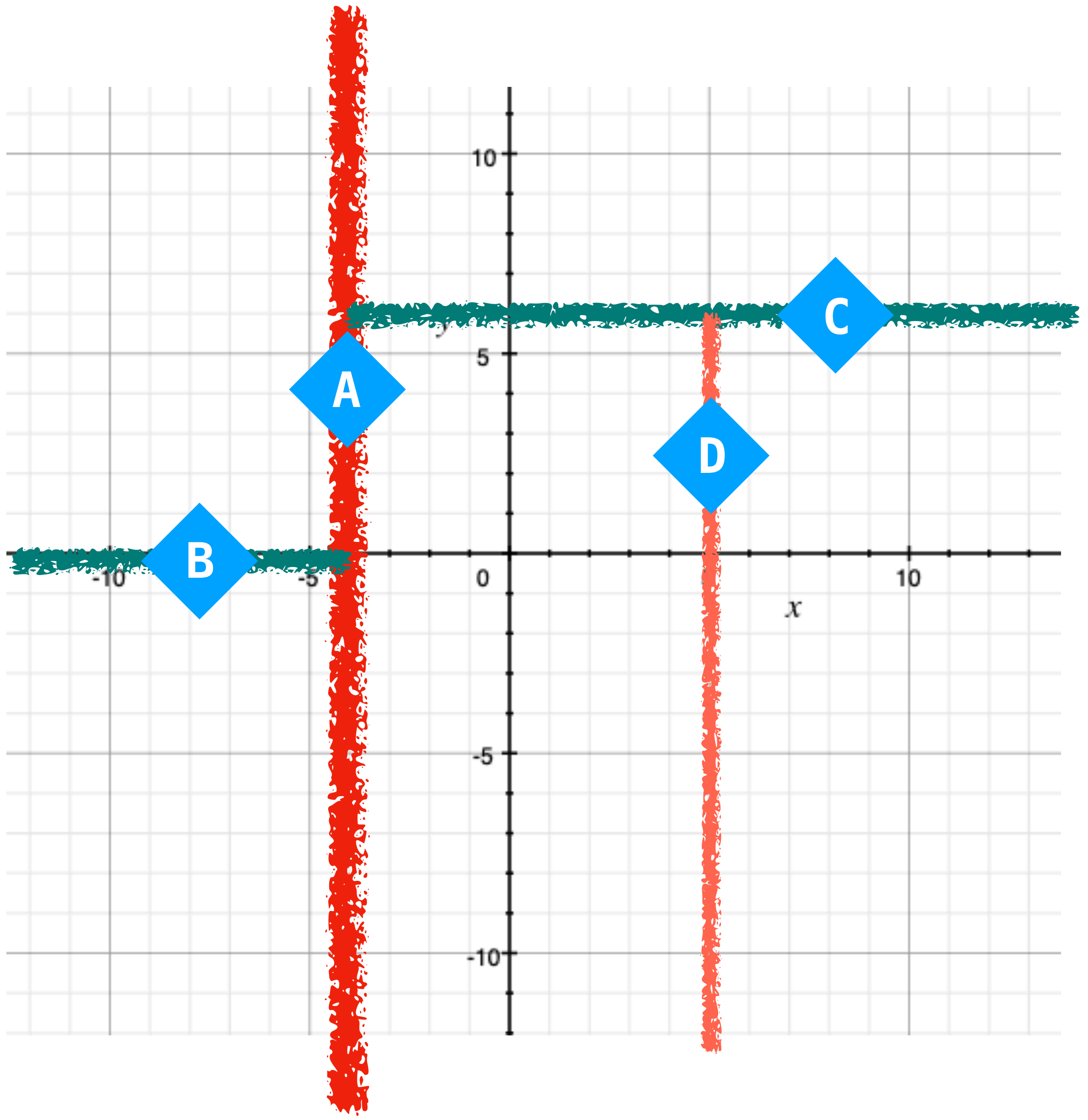
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

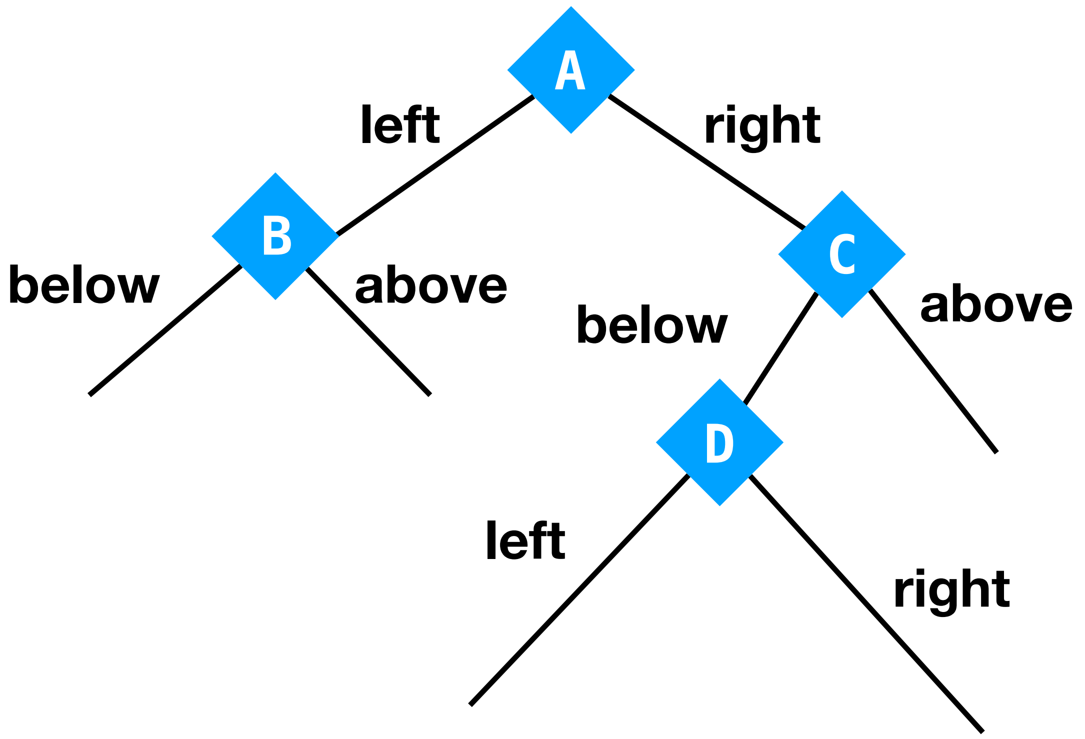




The Solution: k-d trees

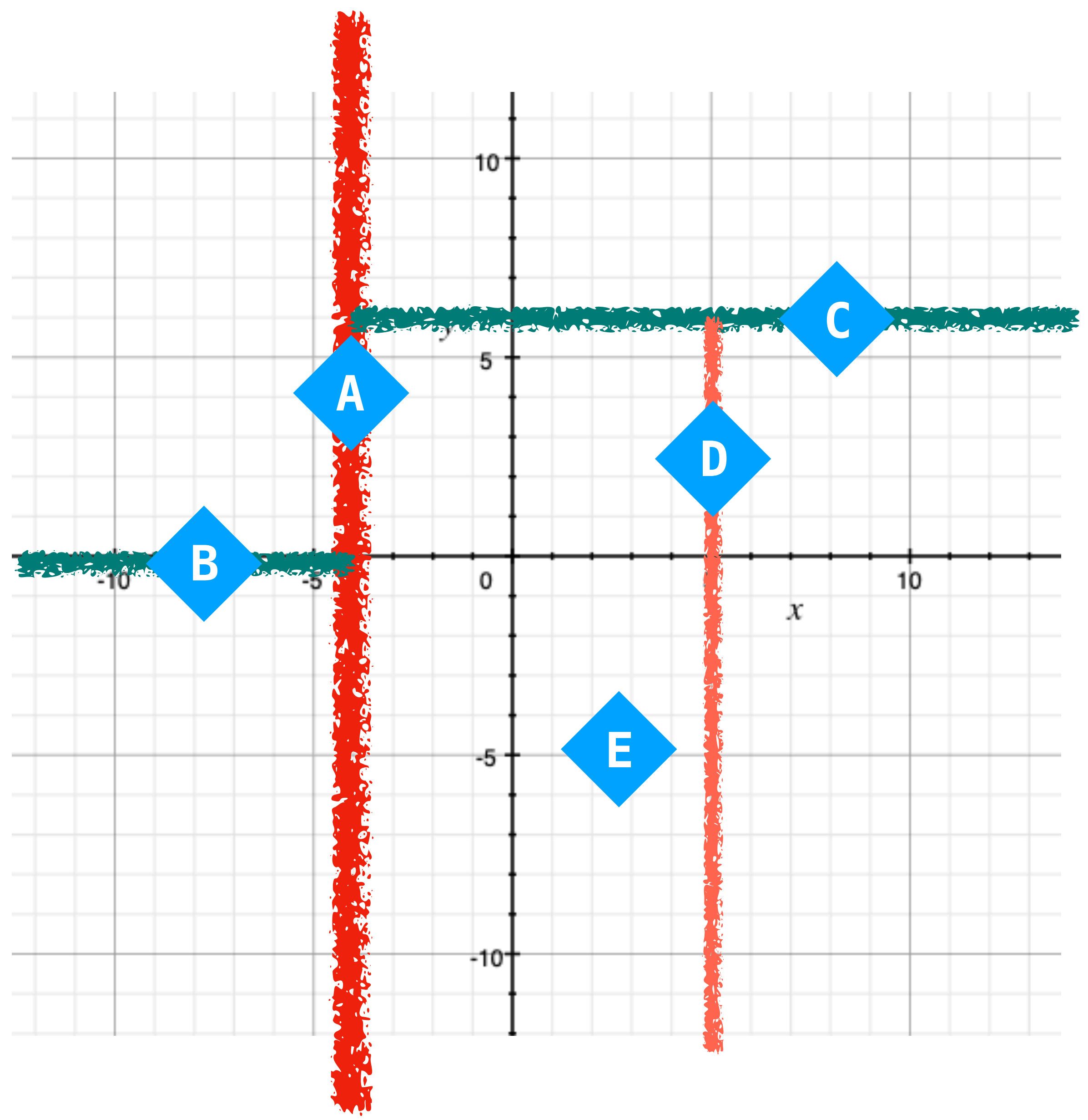
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

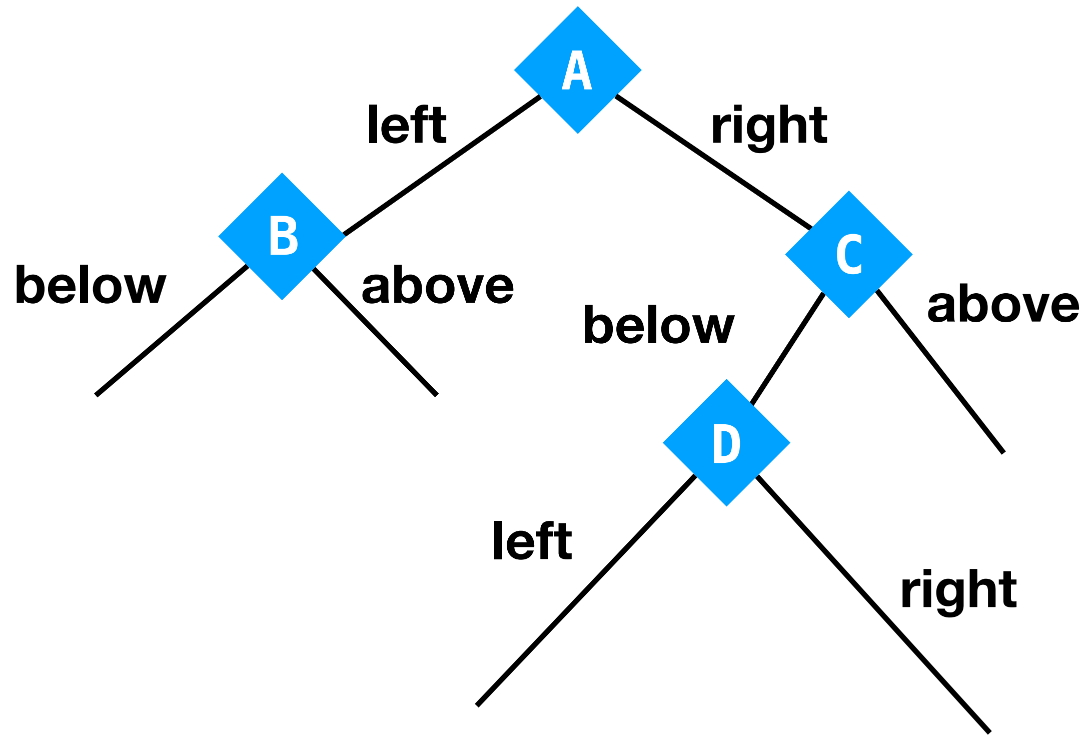




The Solution: k-d trees

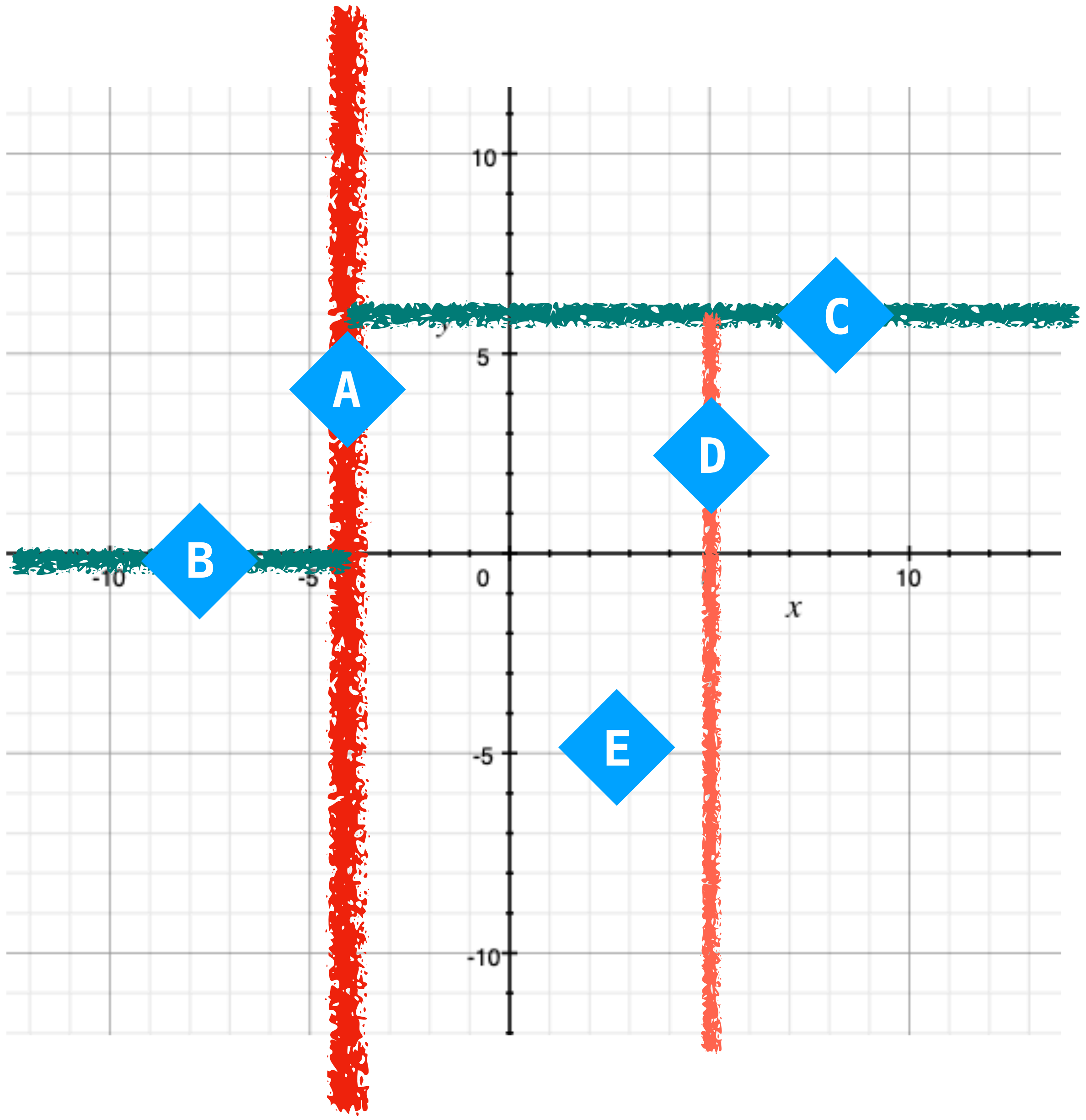
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

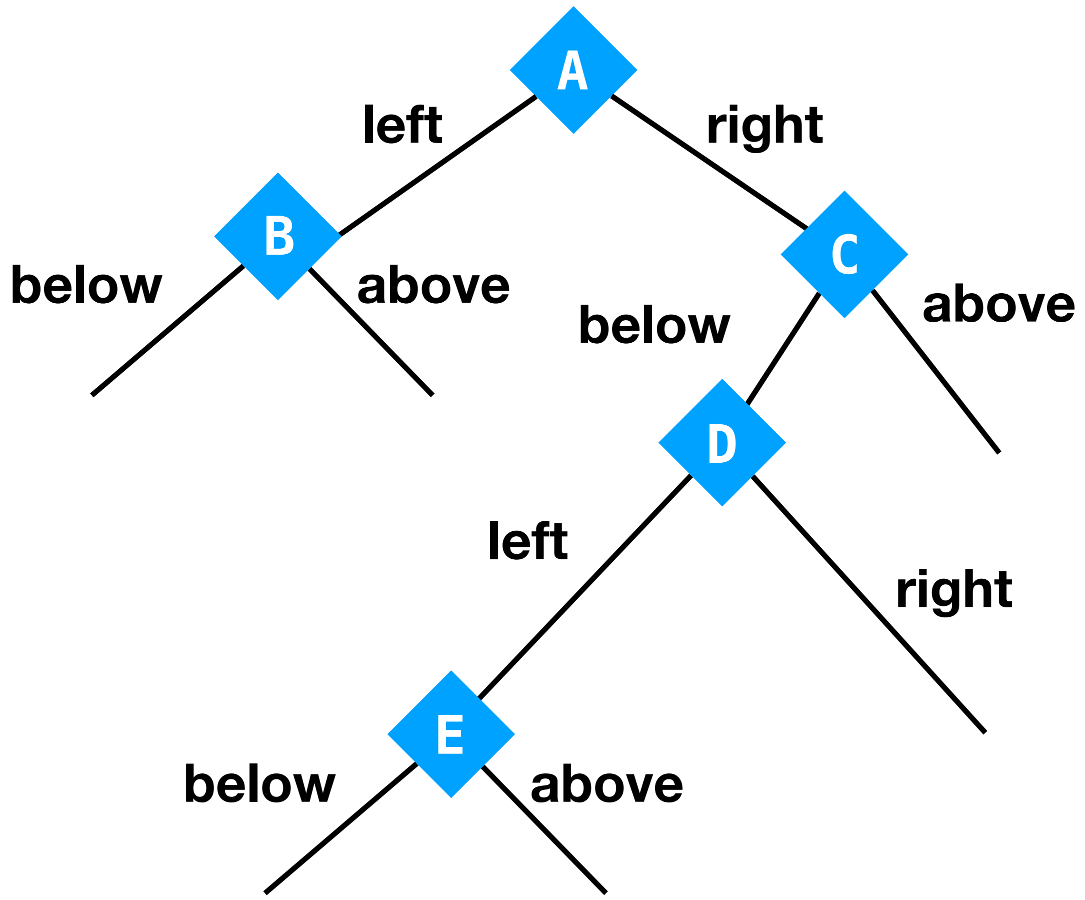




The Solution: k-d trees

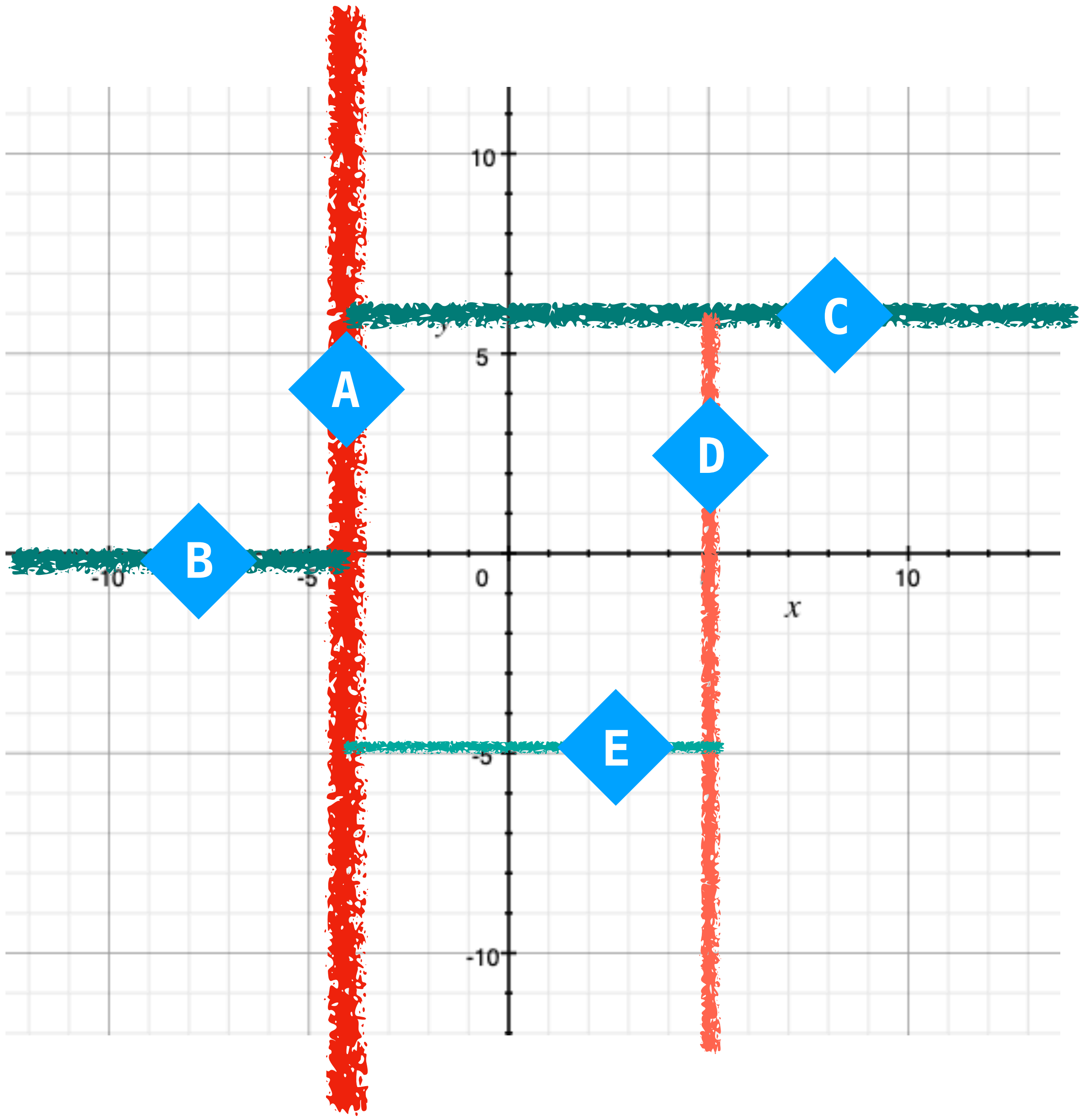
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

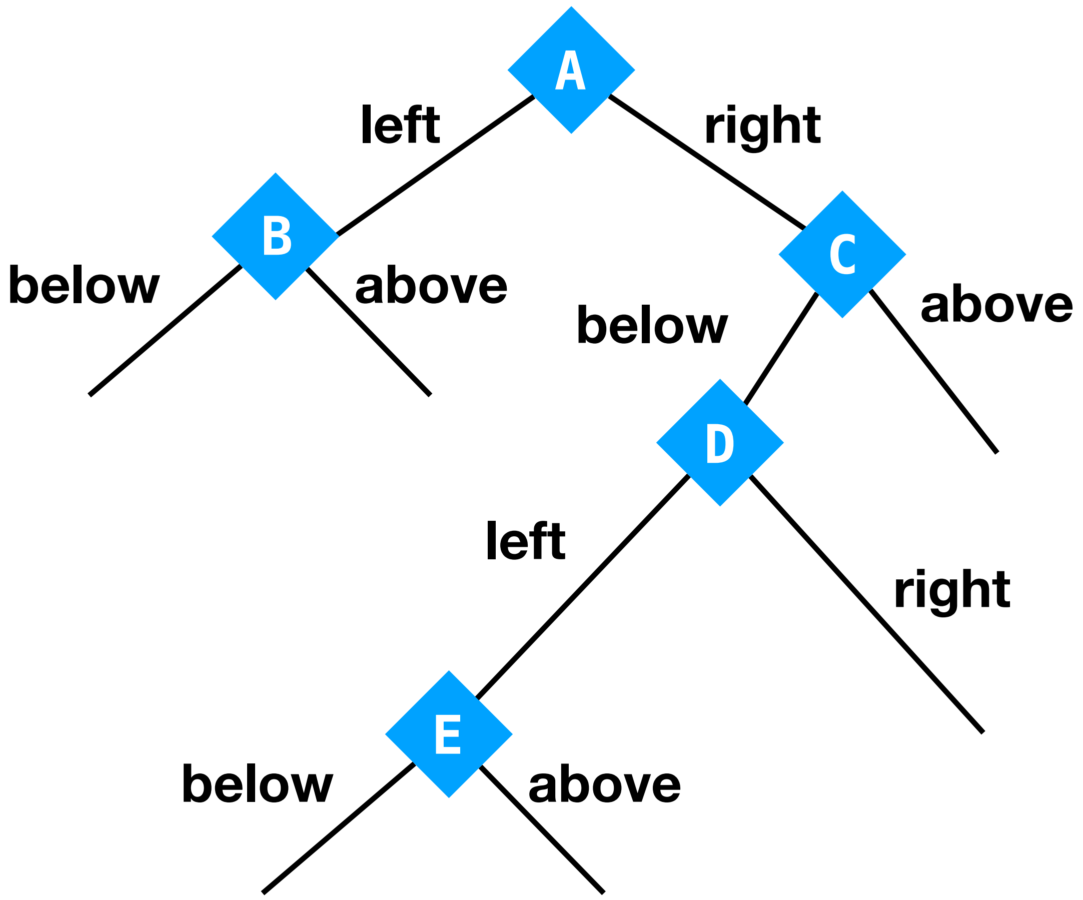




The Solution: k-d trees

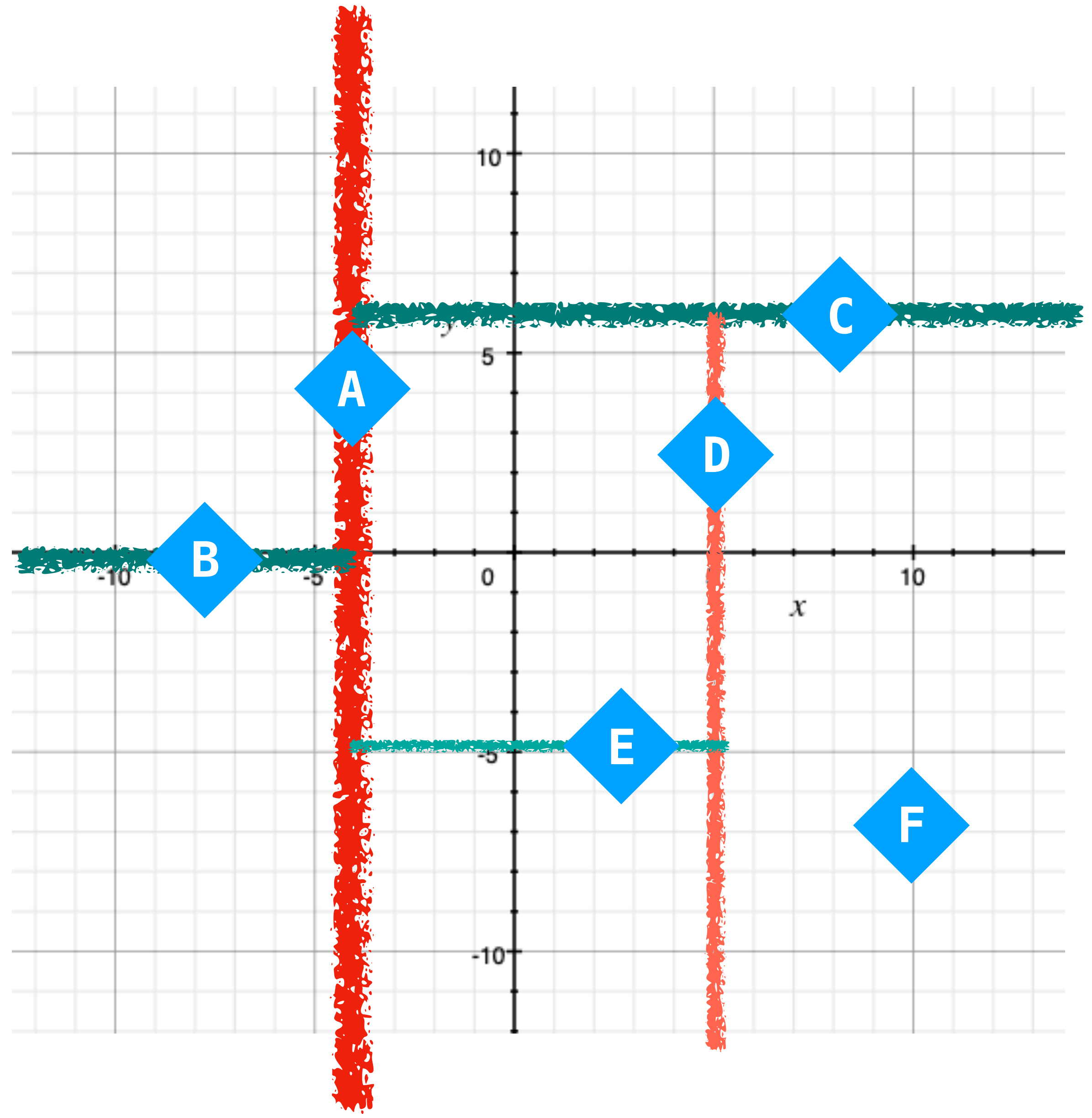
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

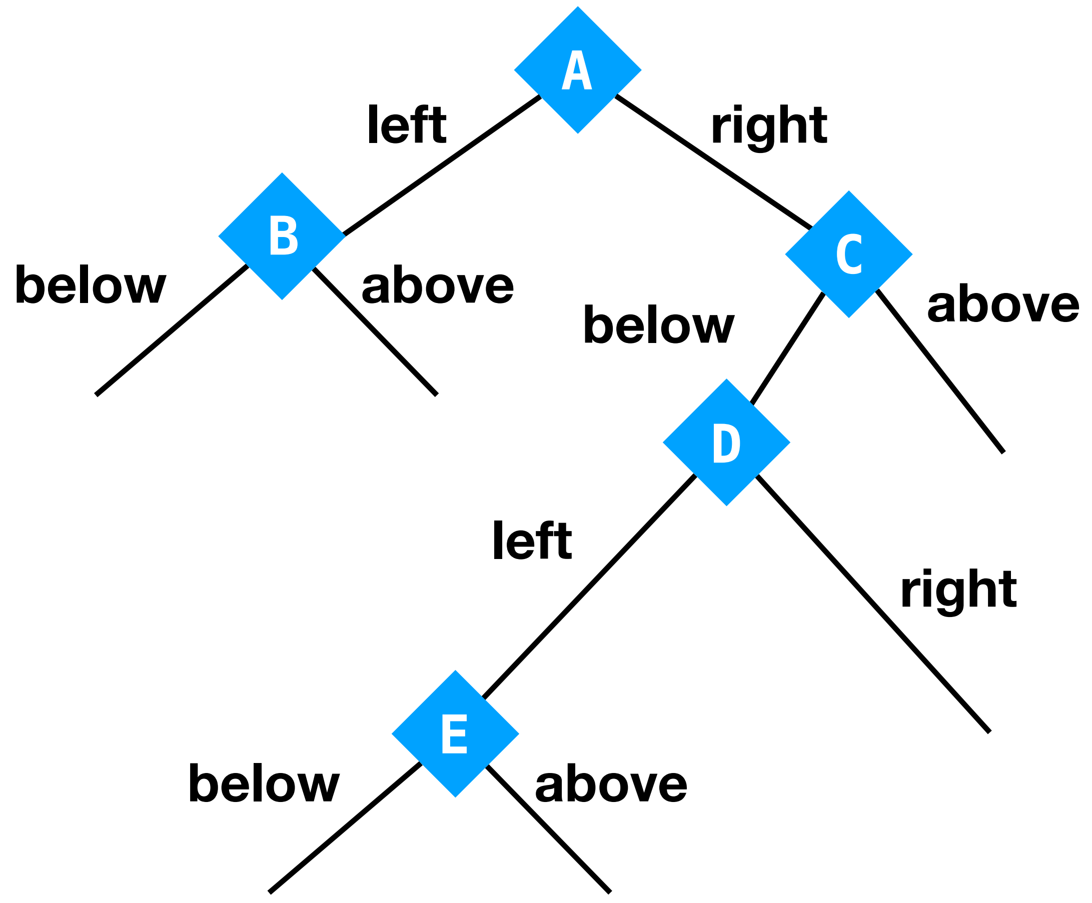




The Solution: k-d trees

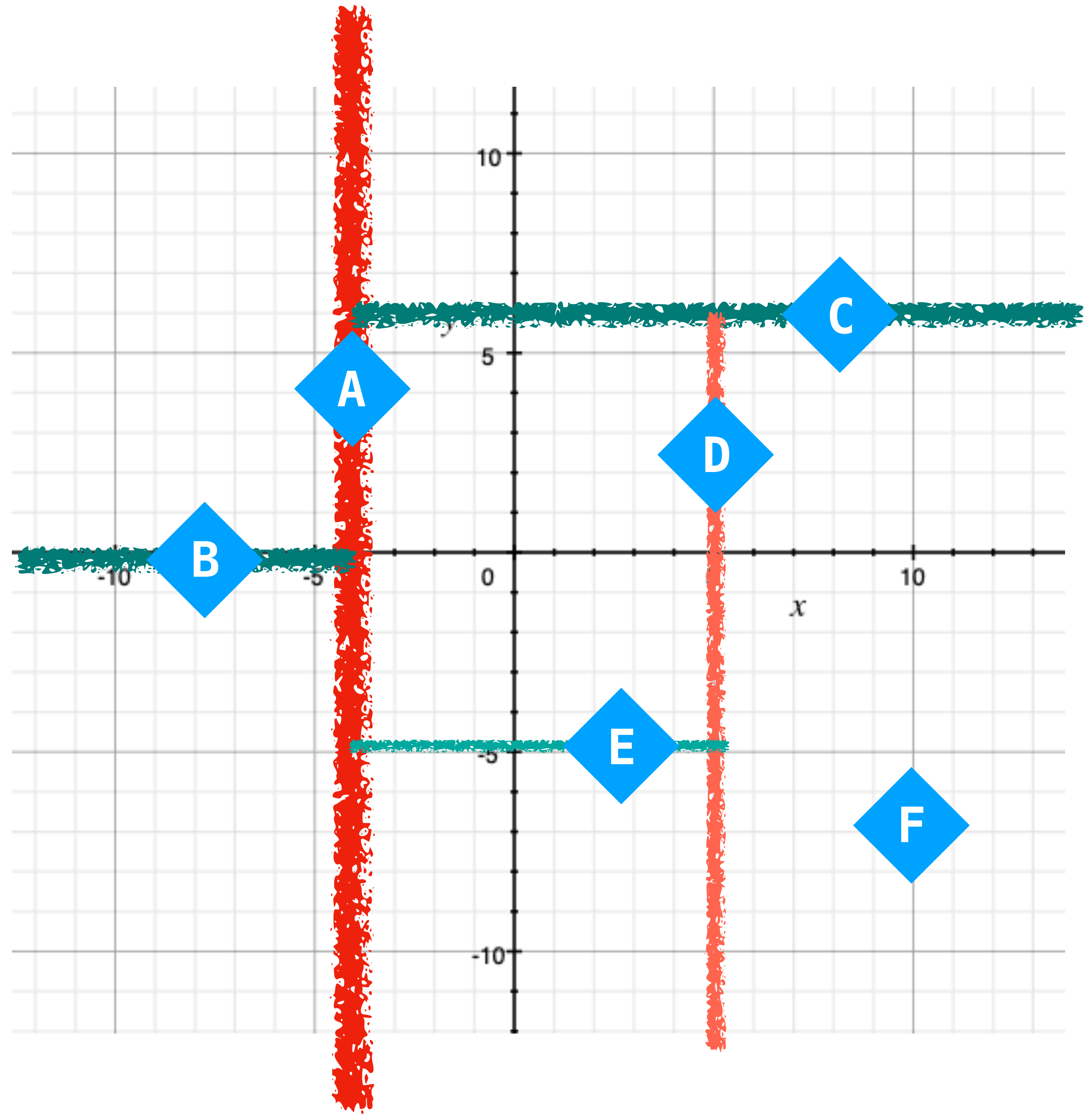
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

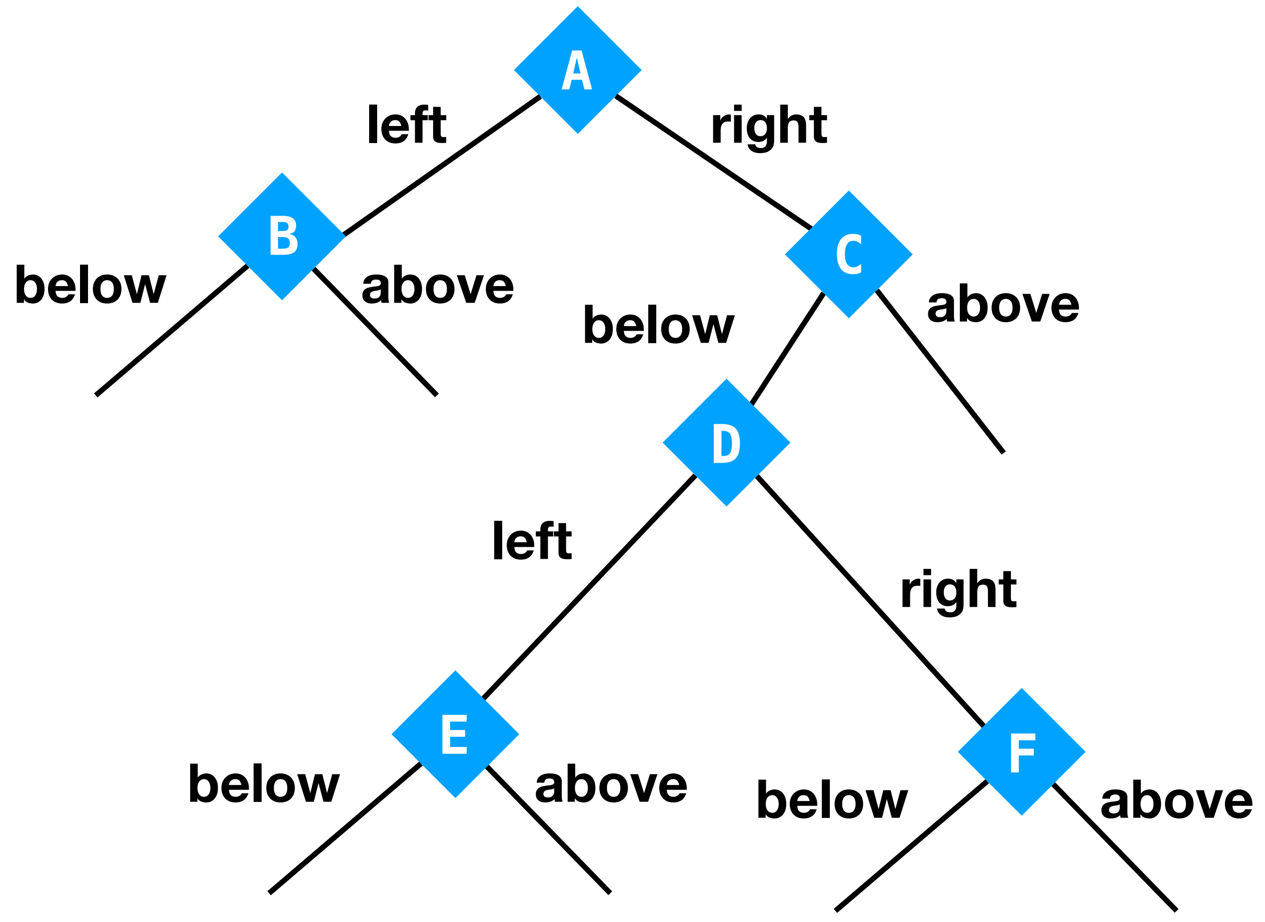




The Solution: k-d trees

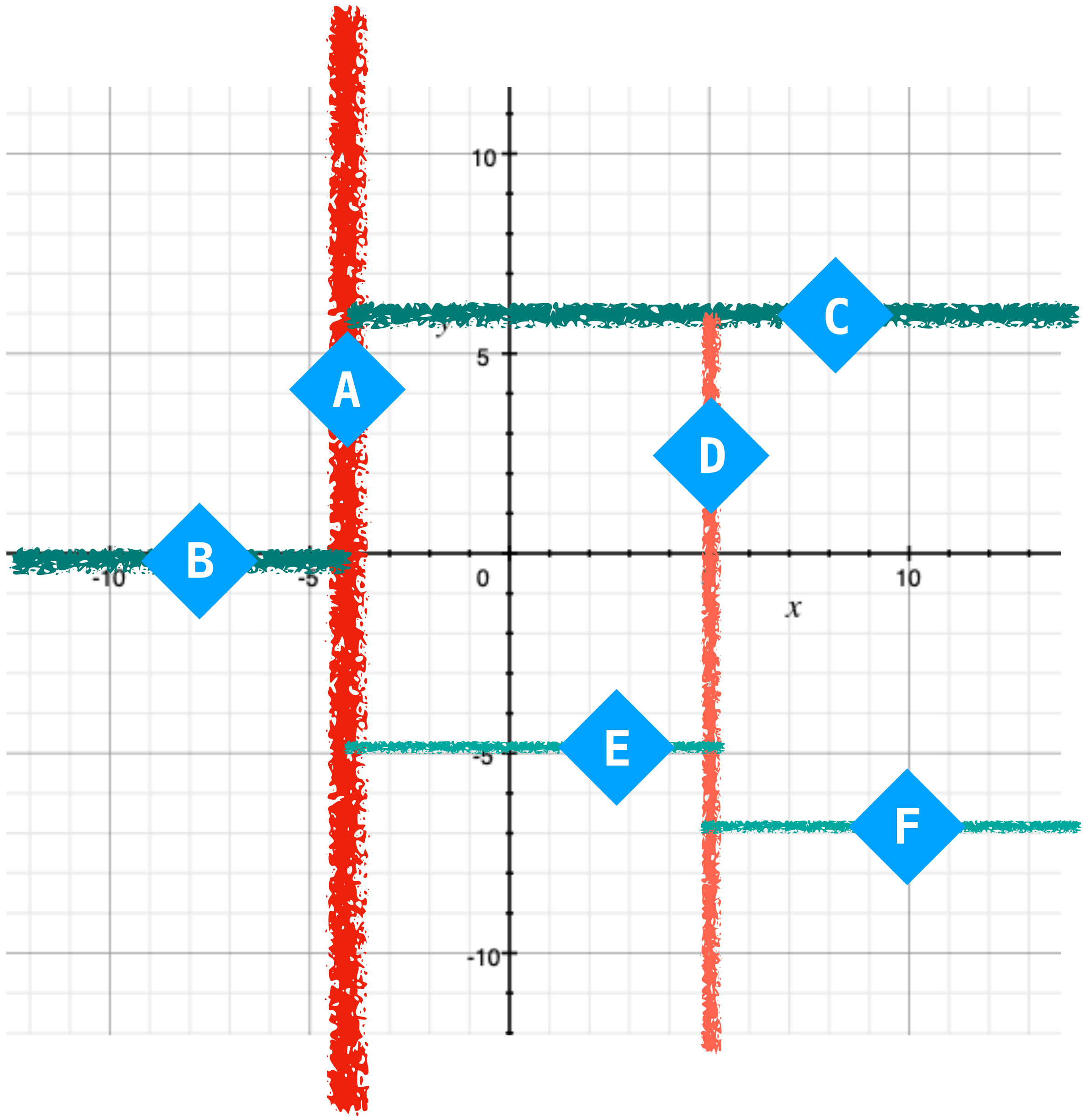
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

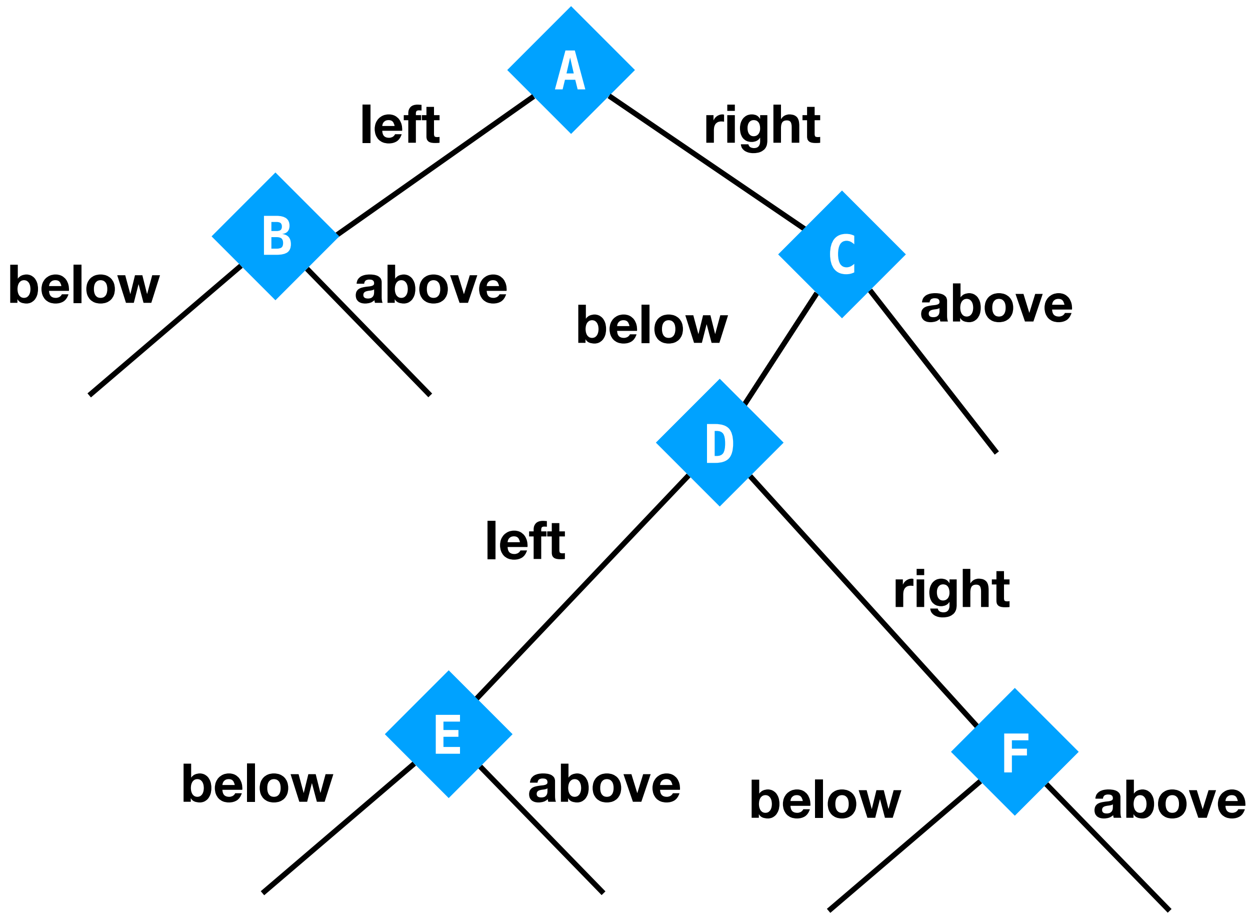




The Solution: k-d trees

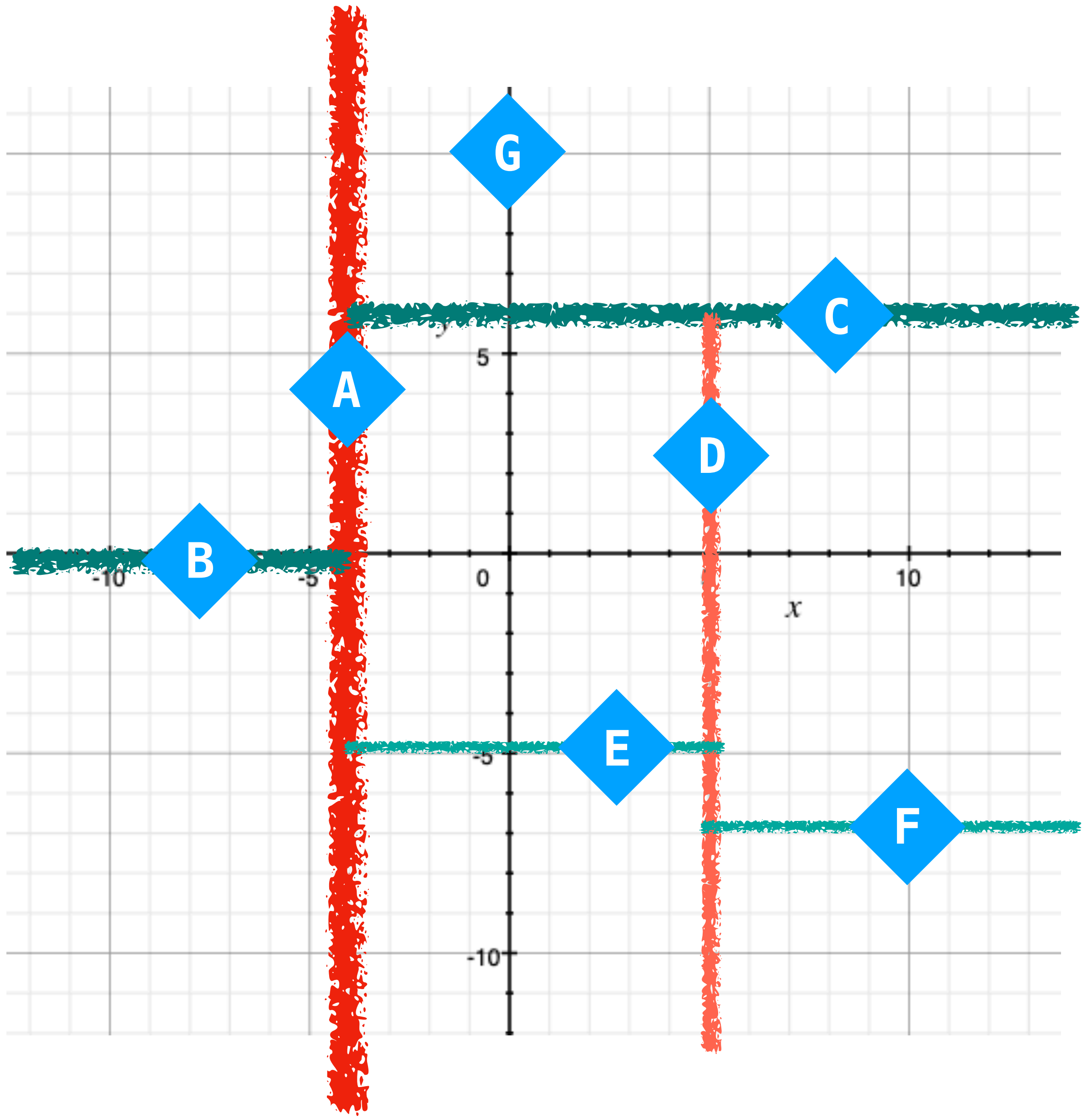
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

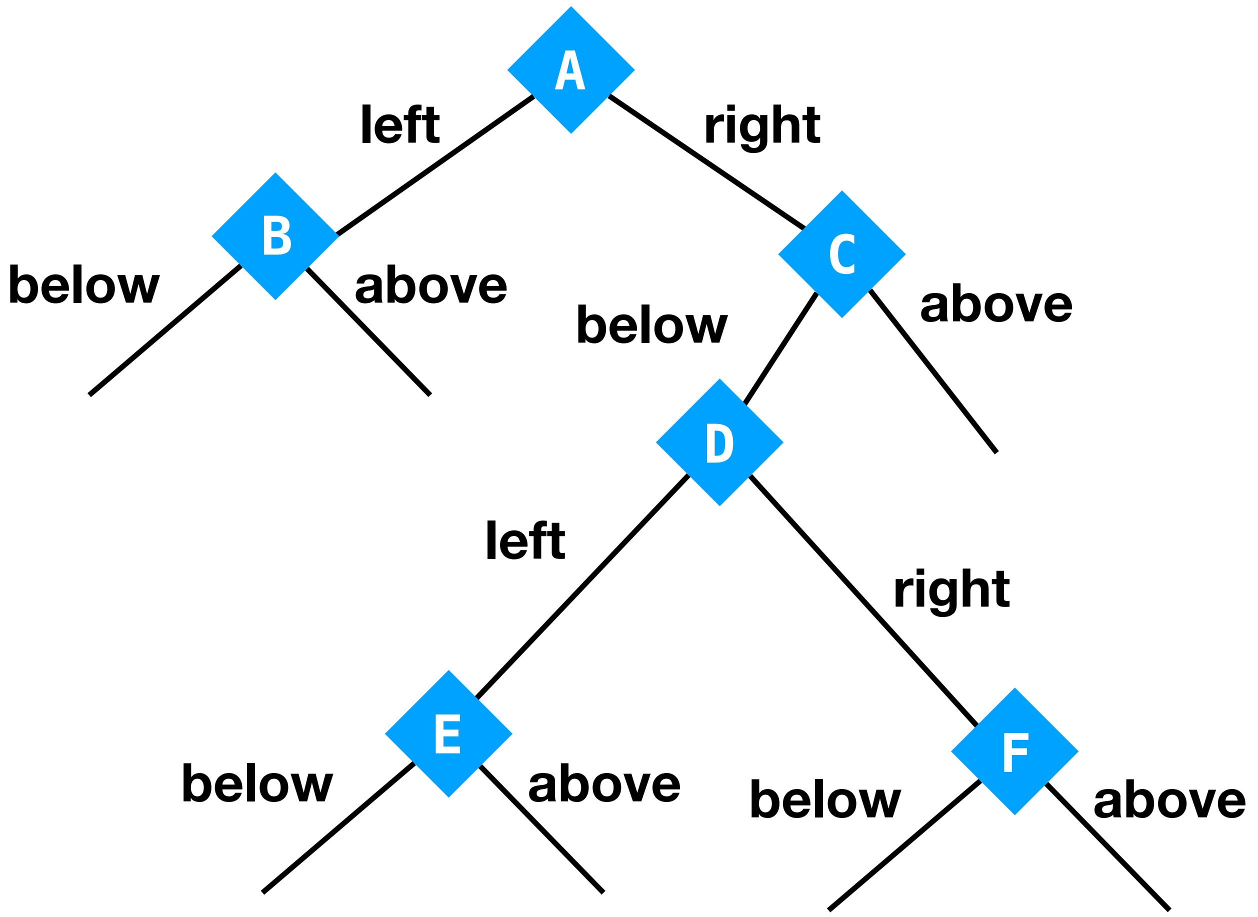




The Solution: k-d trees

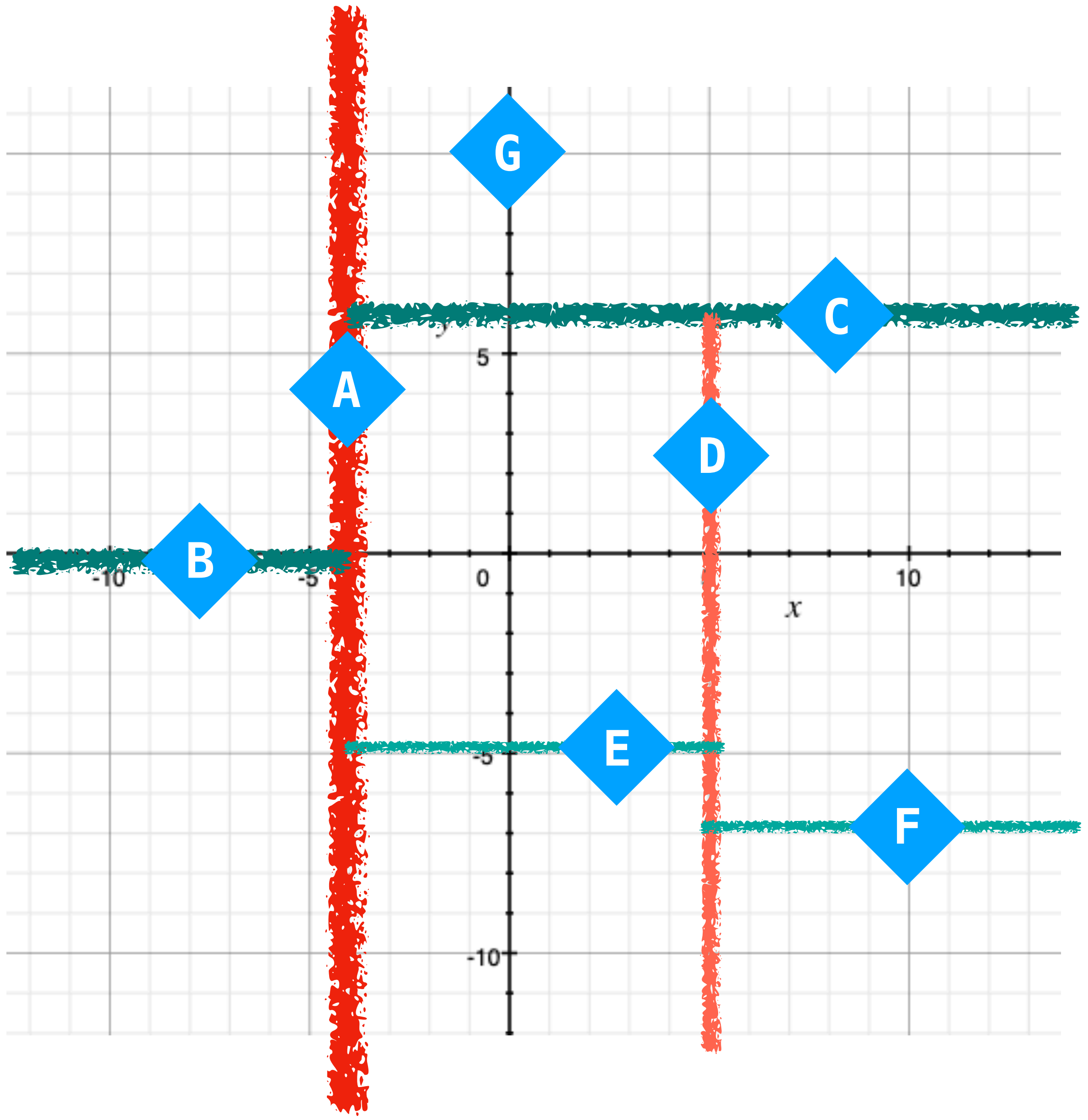
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

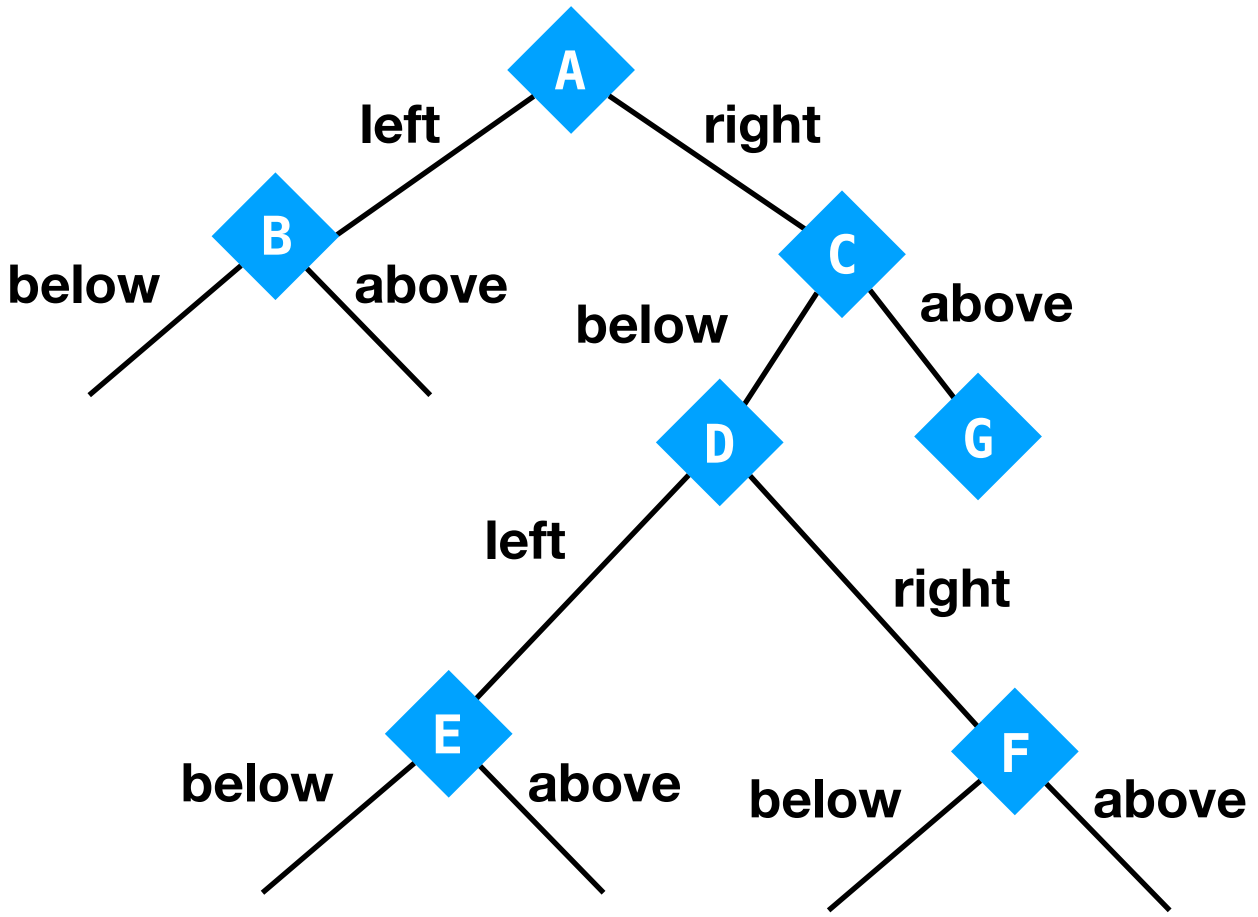




The Solution: k-d trees

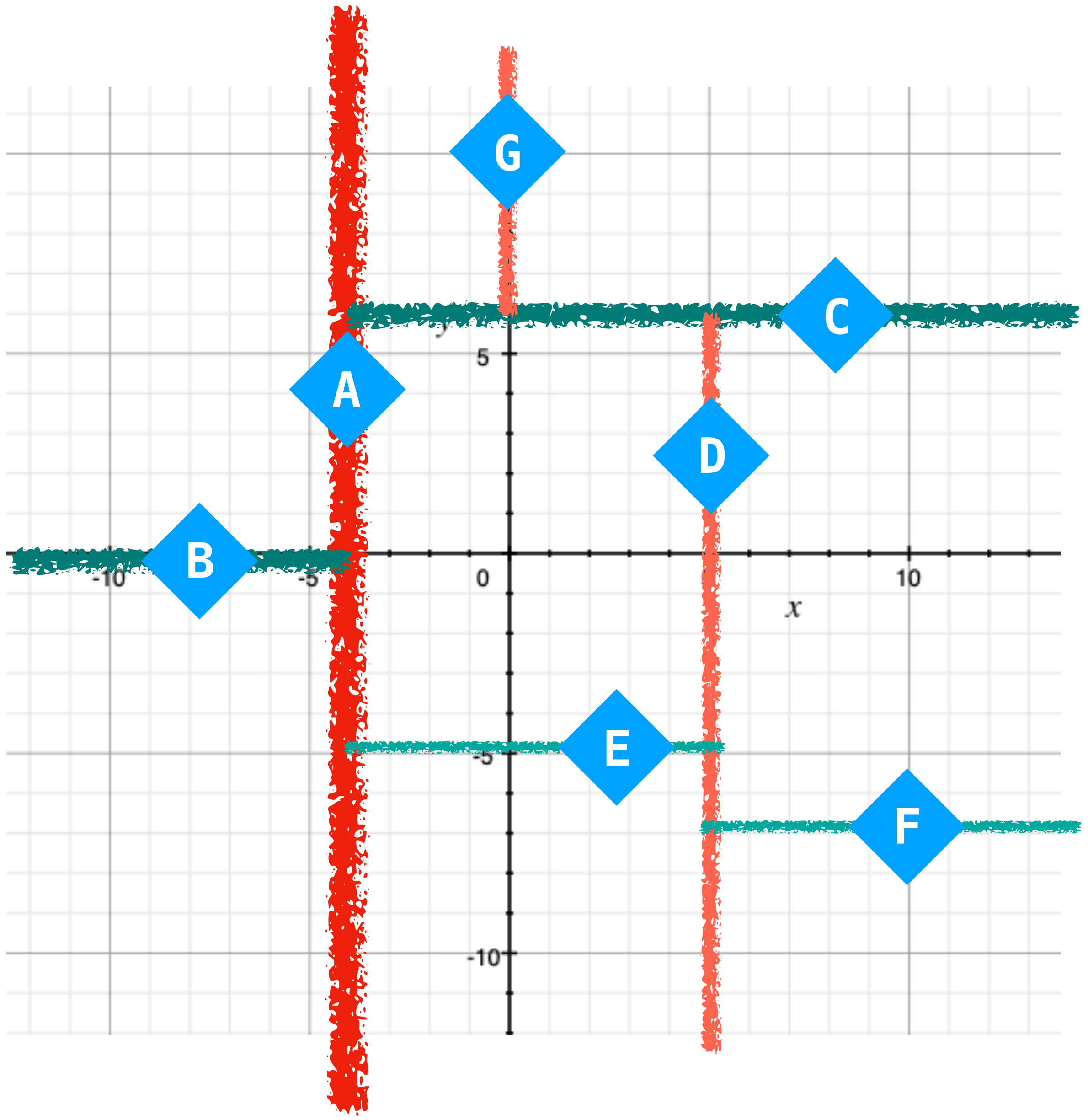
- Make a binary search tree, with a modification:
    - Each level of the tree will compare a different dimension.

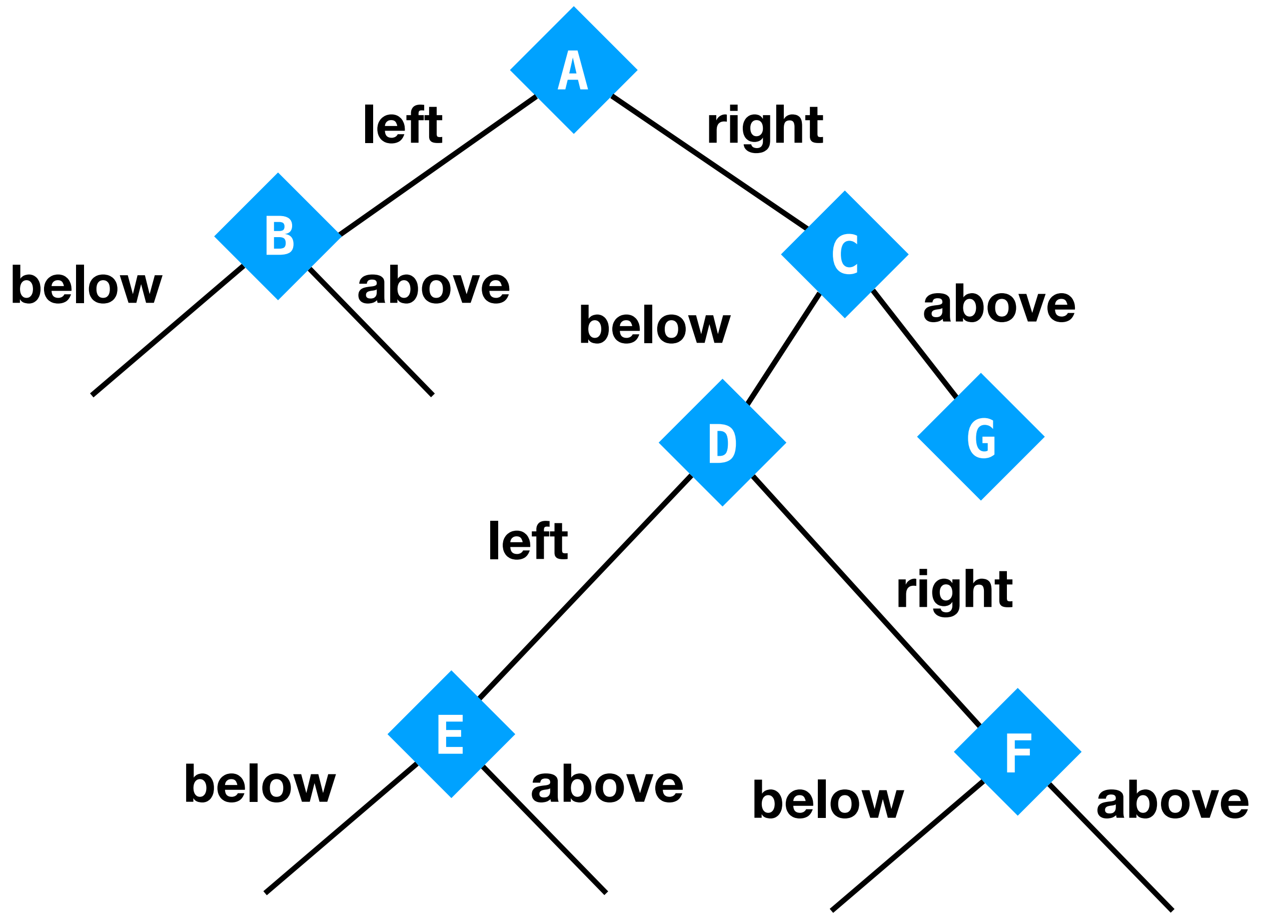




The Solution: k-d trees

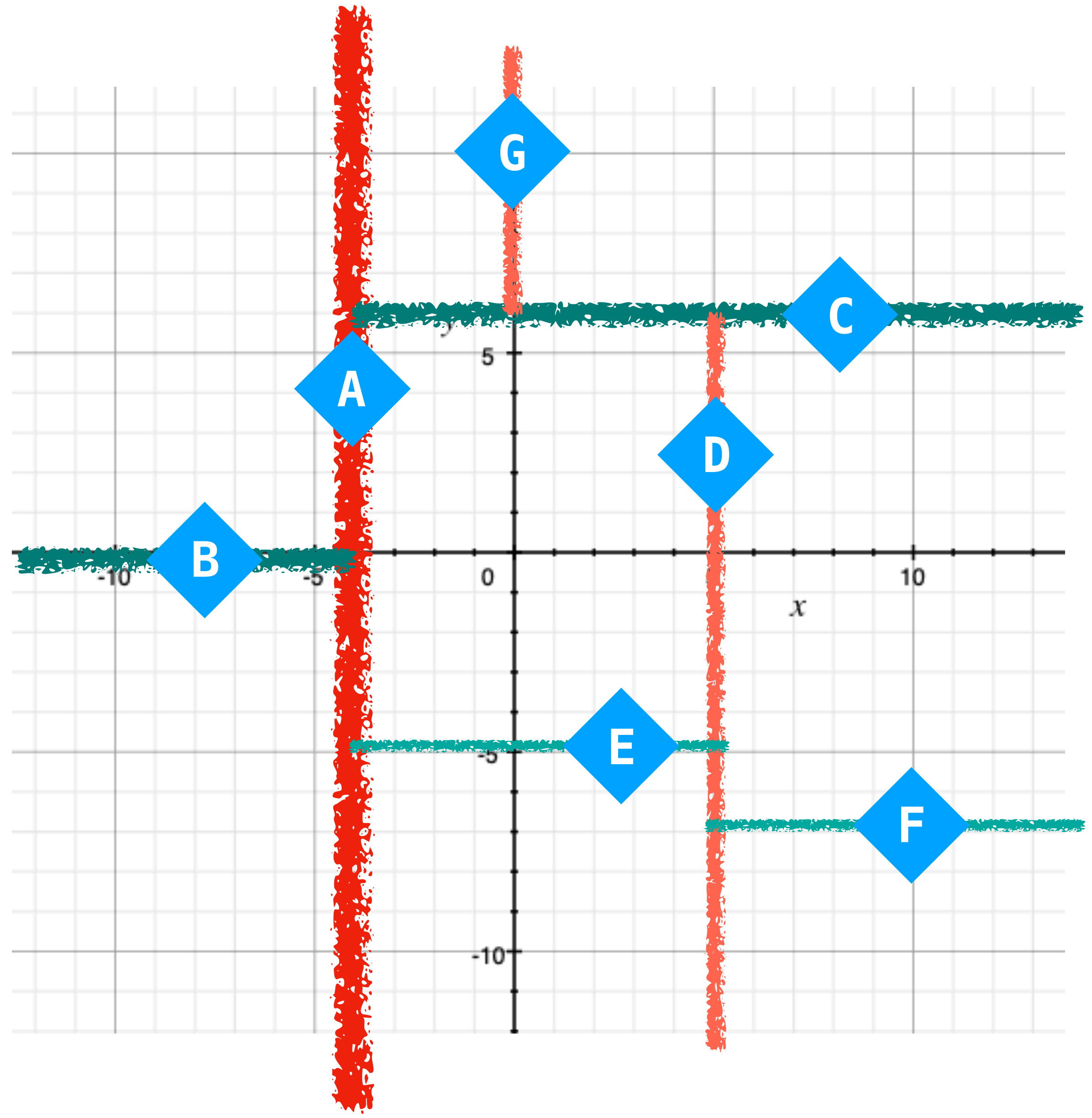
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

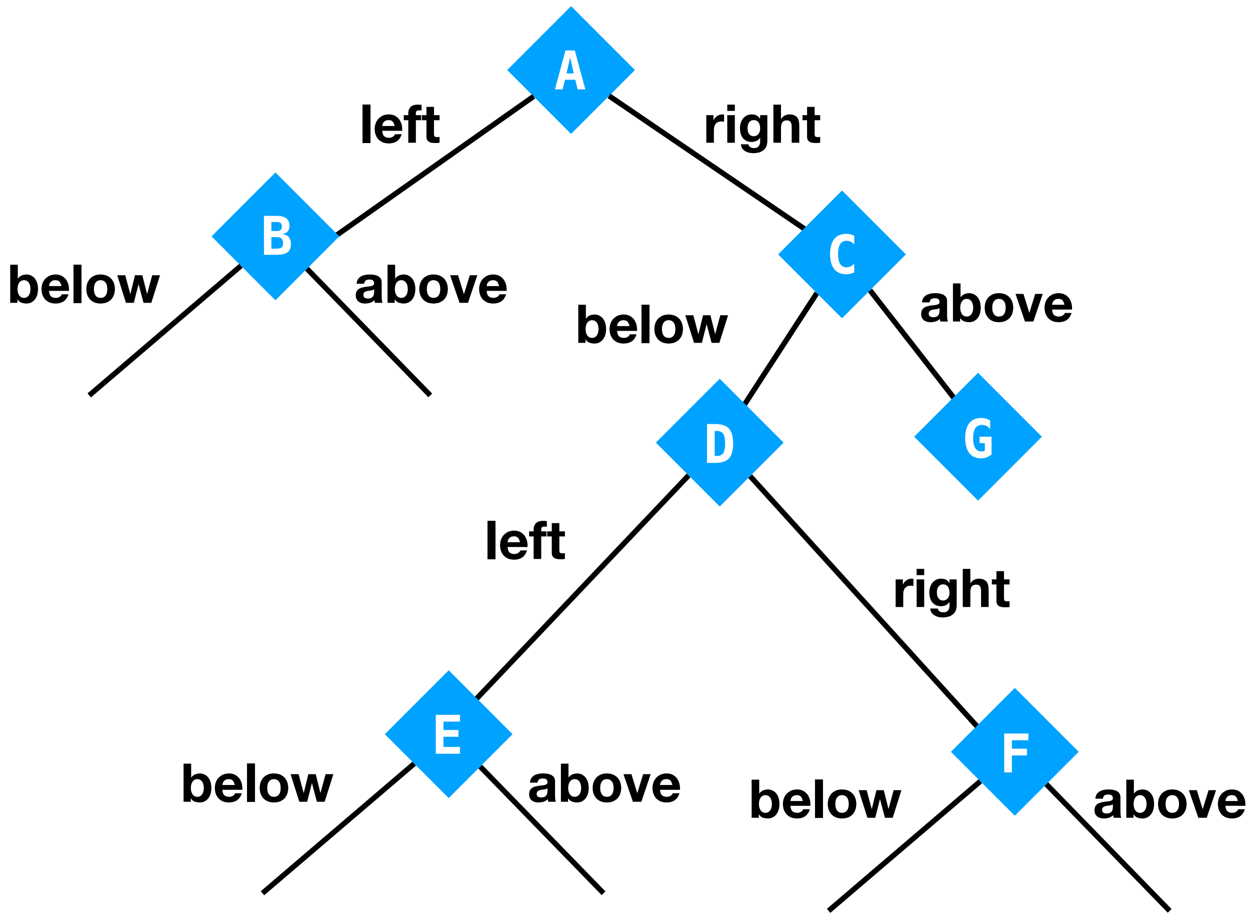




The Solution: k-d trees

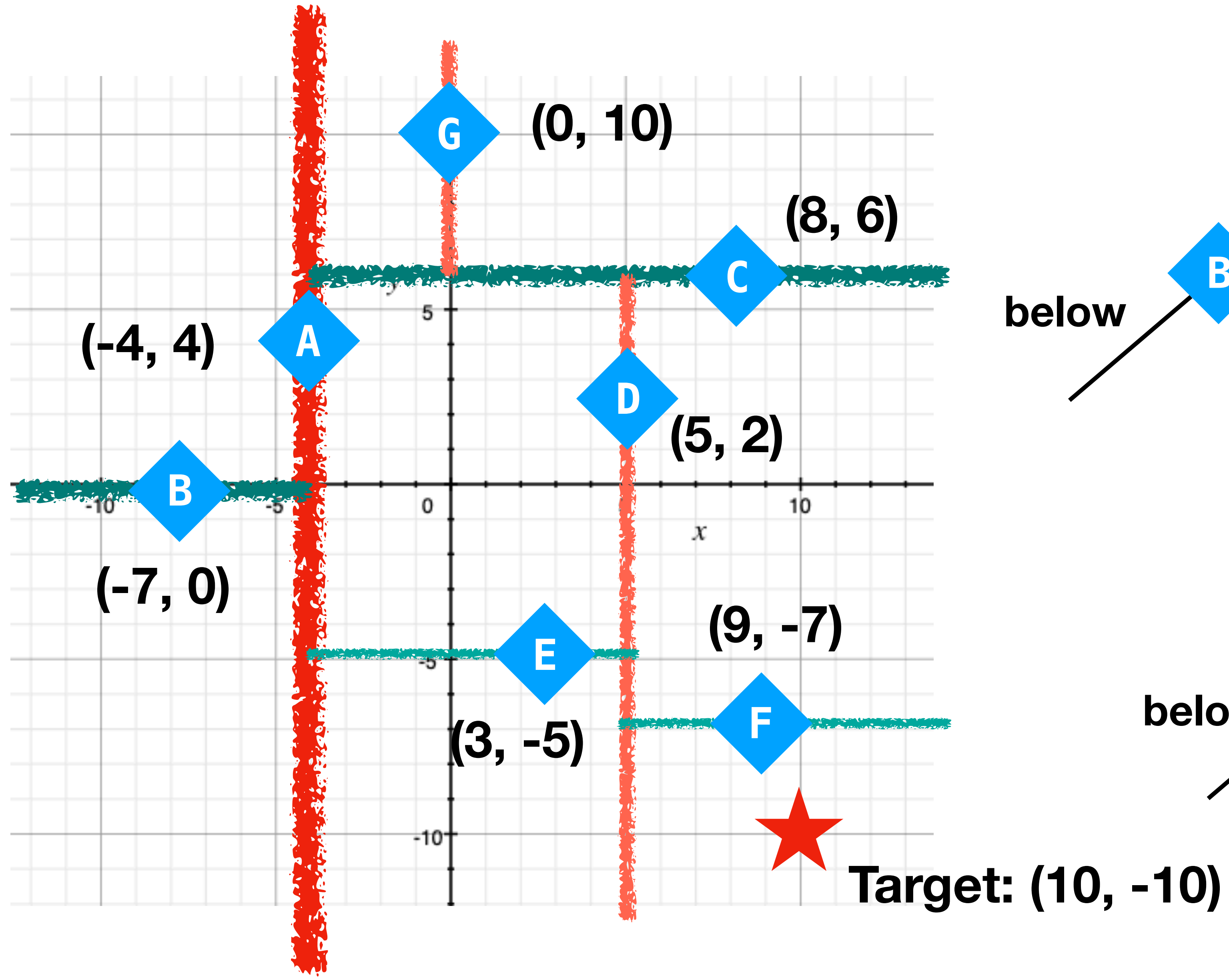
- Make a binary search tree, with a modification:
  - Each level of the tree will compare a different dimension.

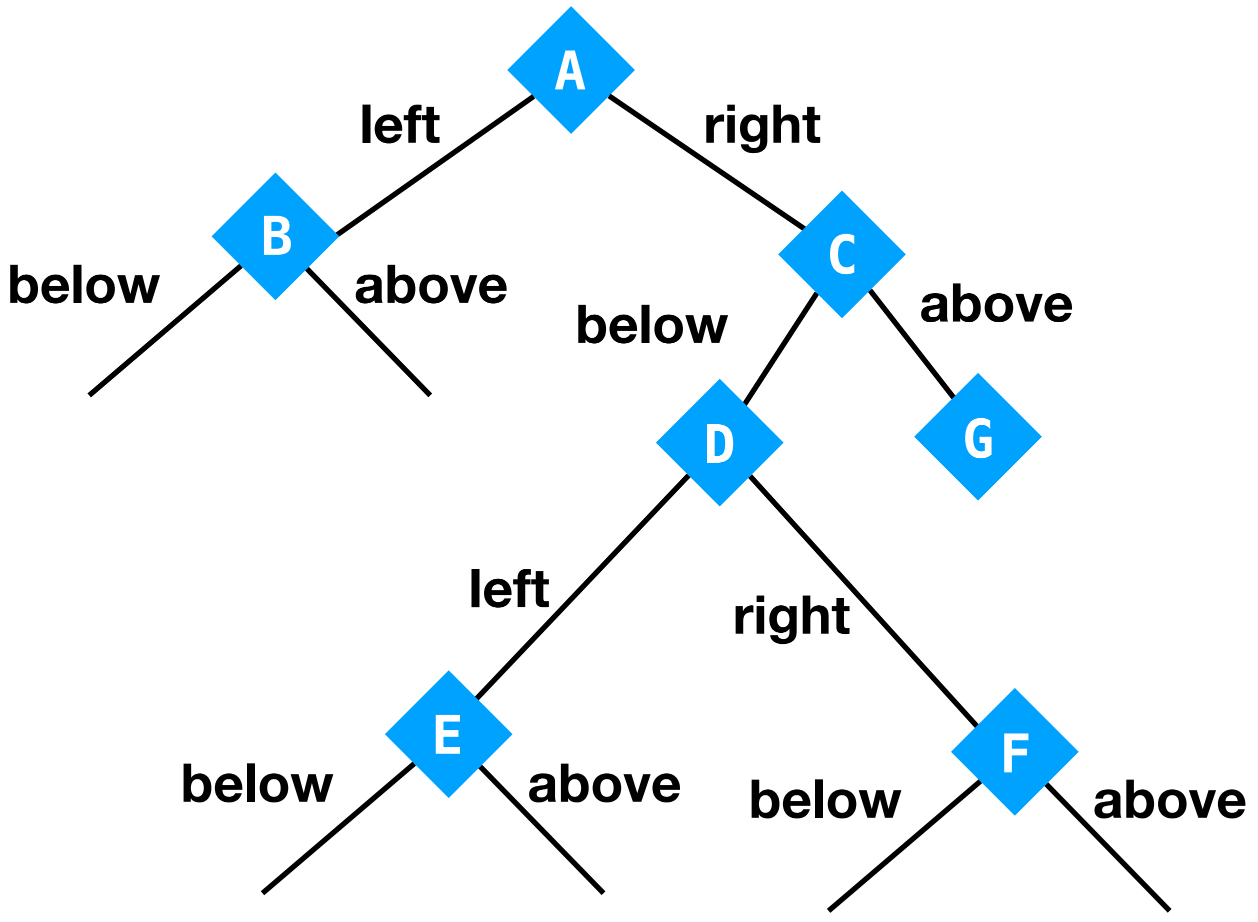




Nearest

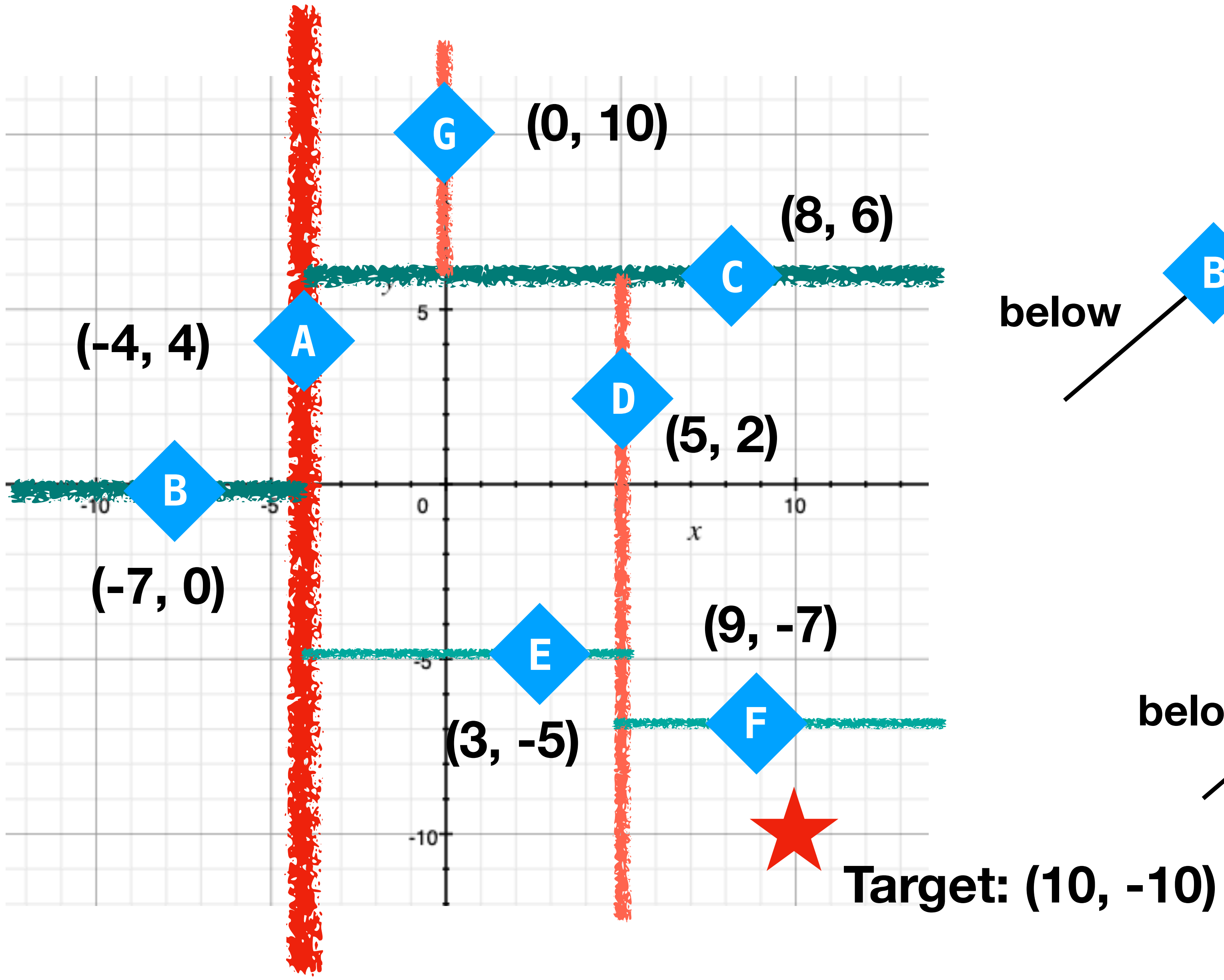
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

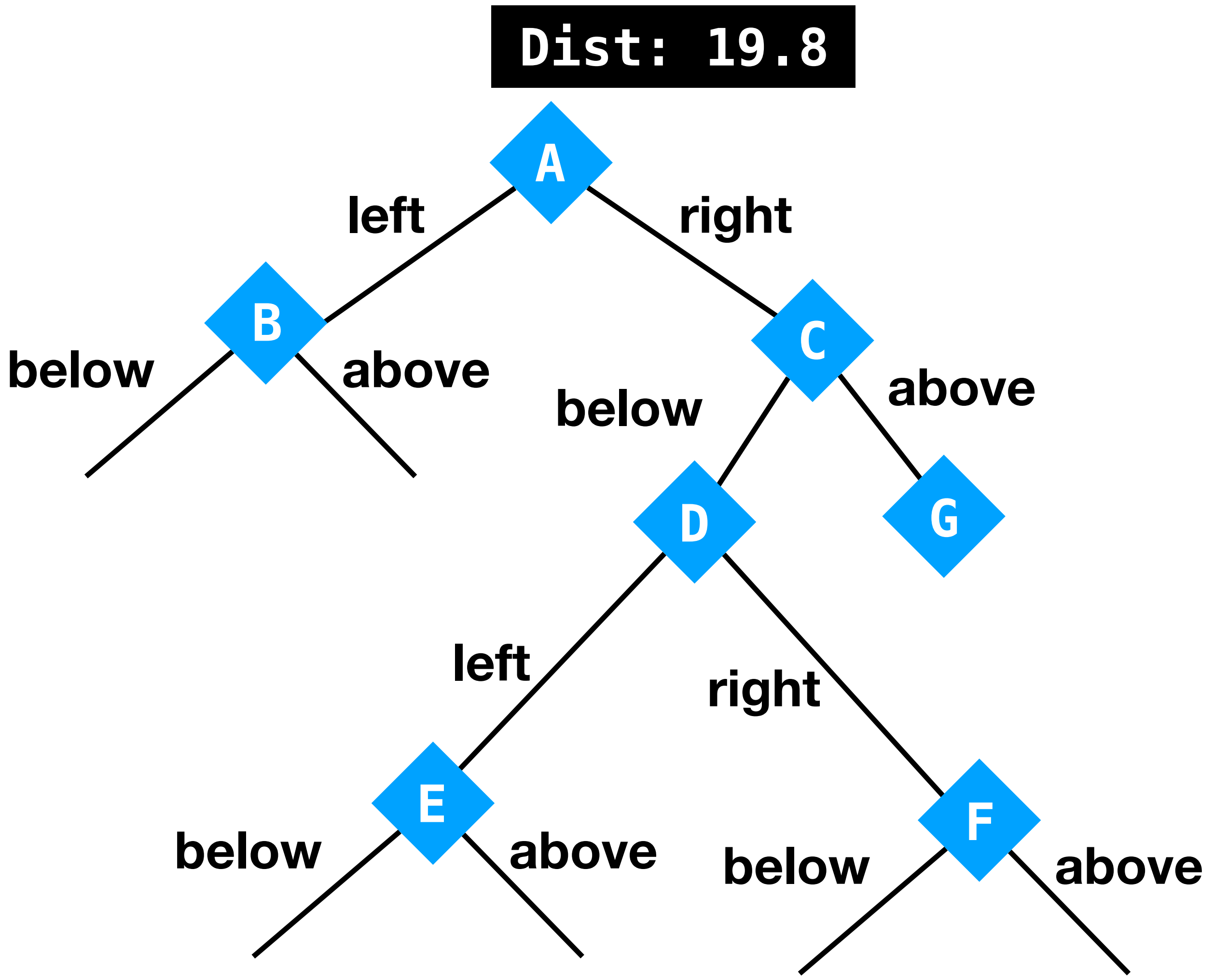




Nearest

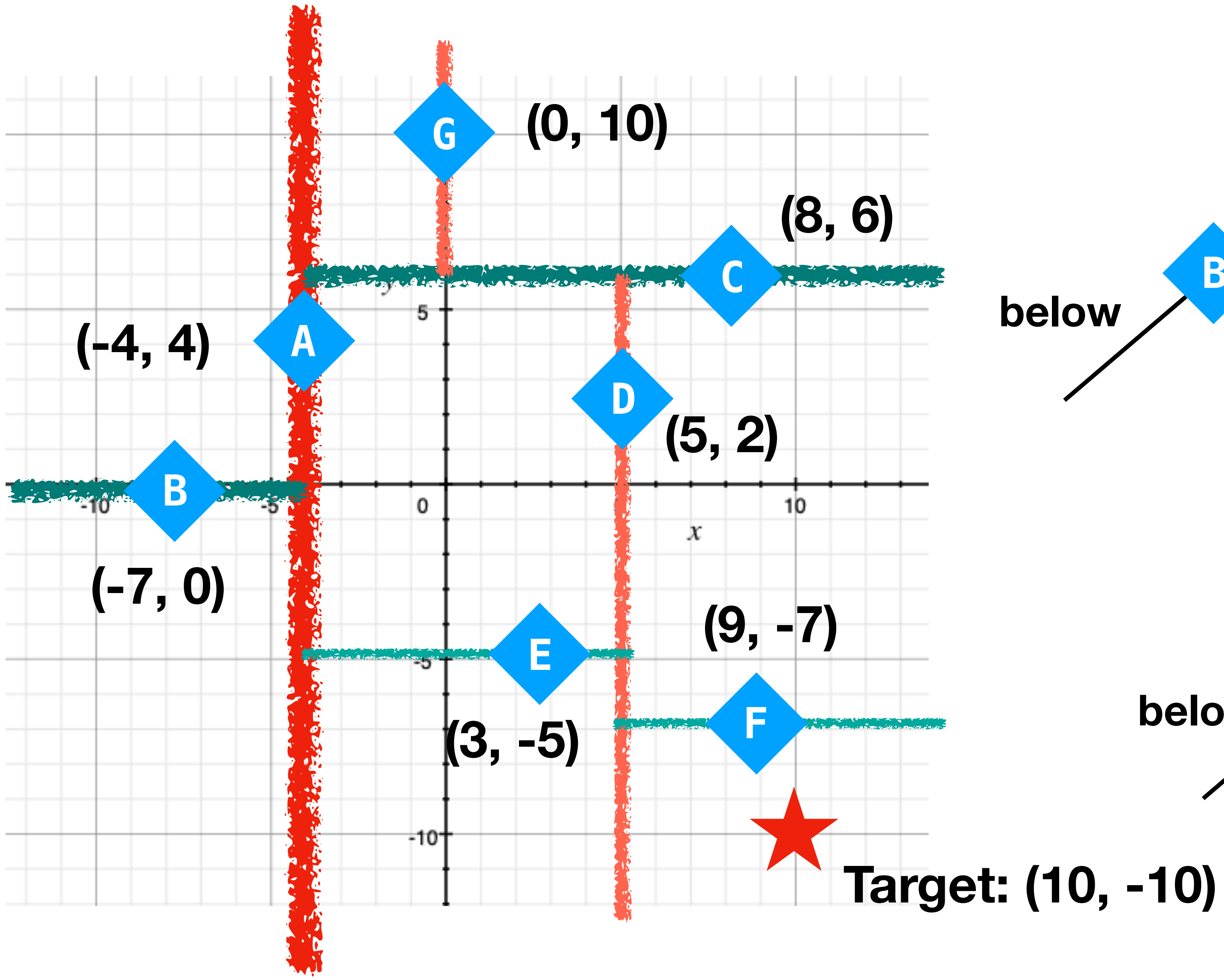
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

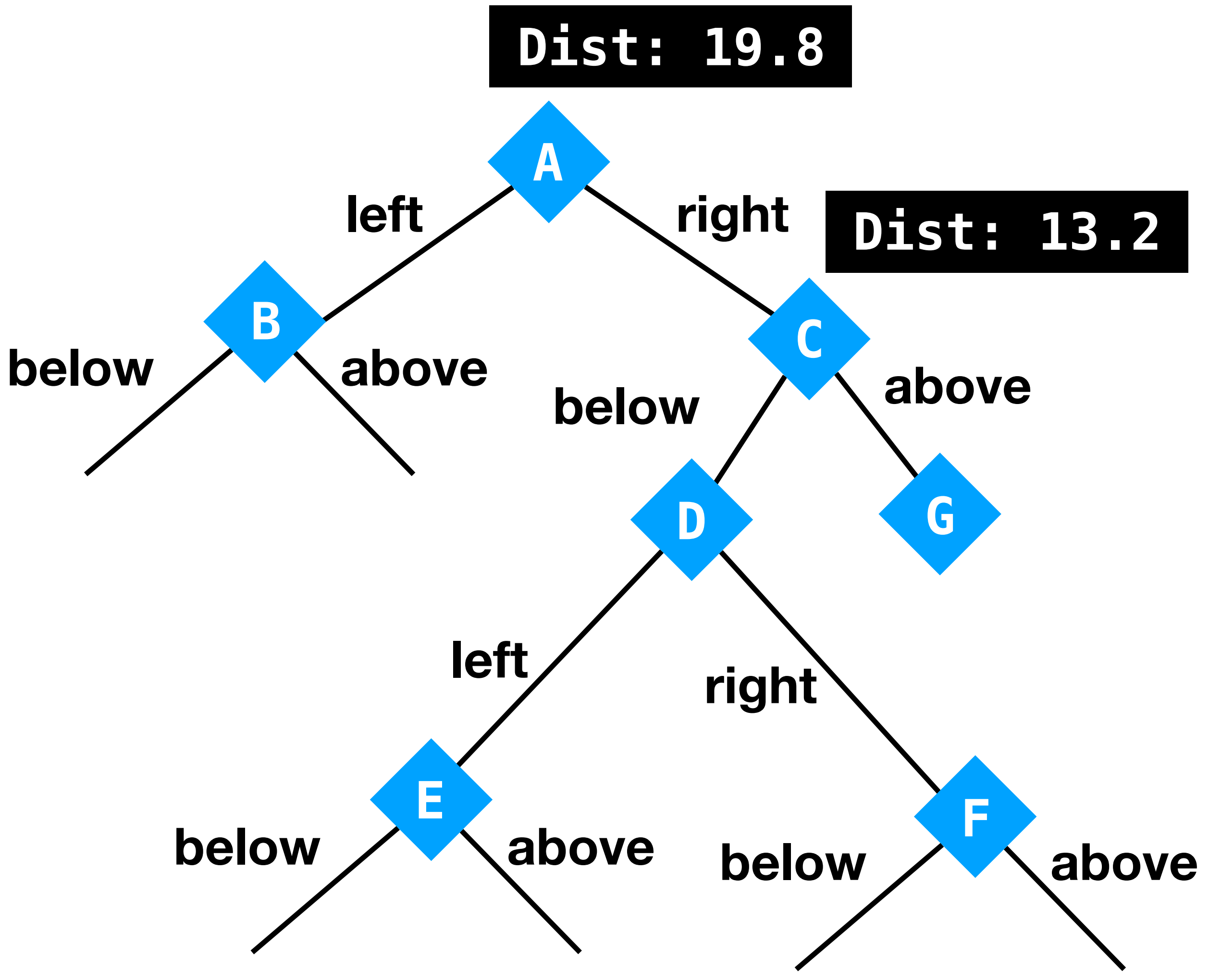




Nearest

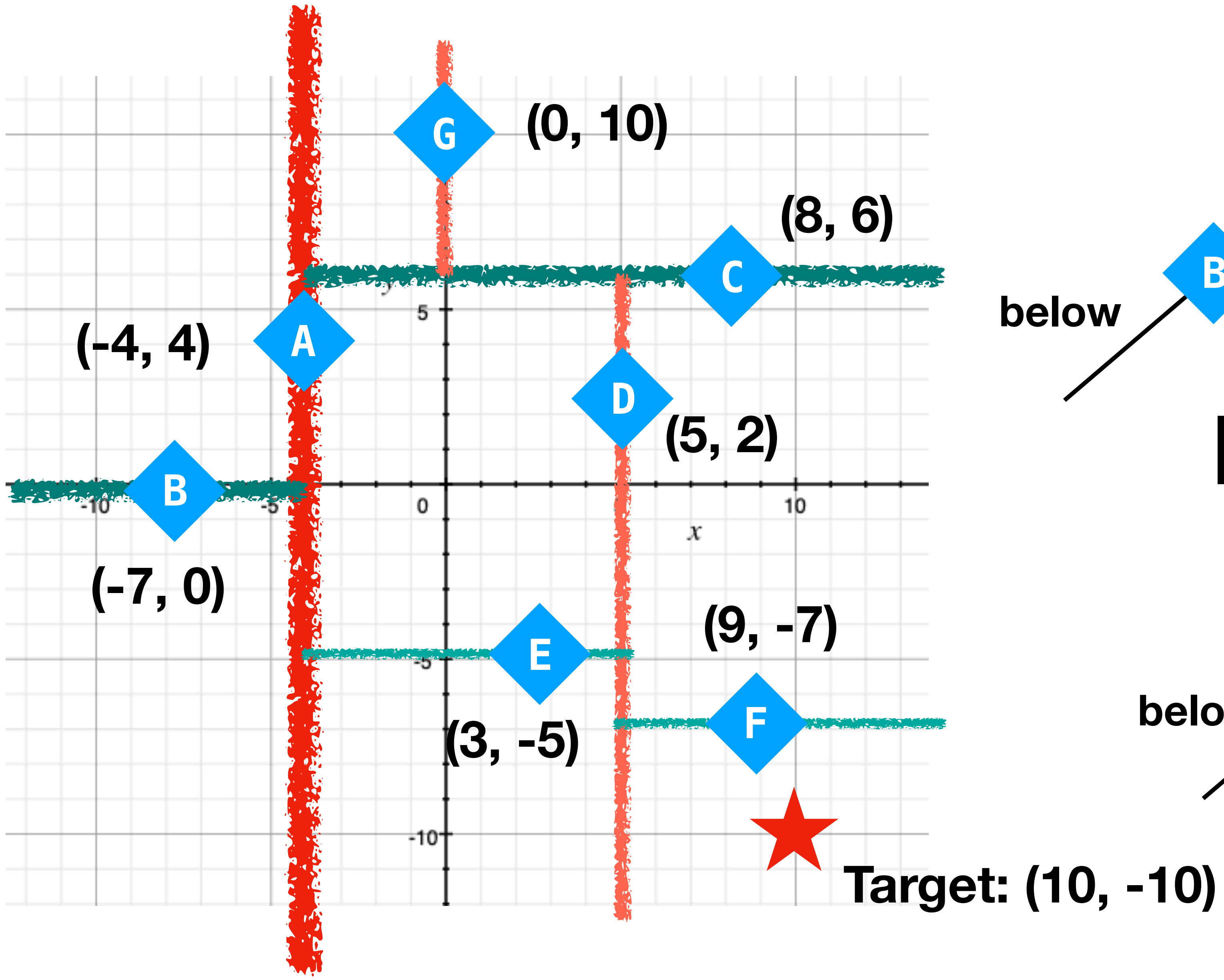
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

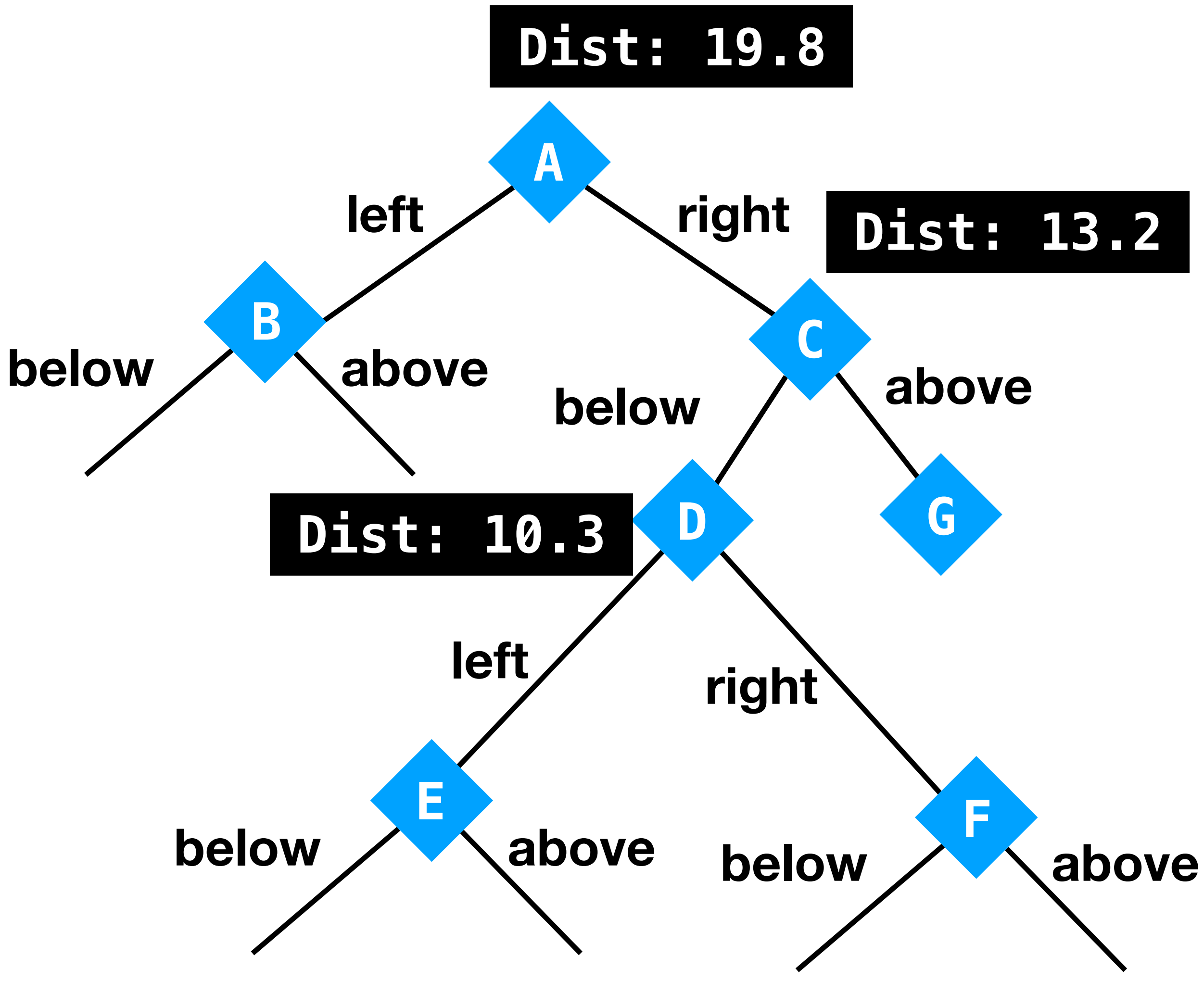




Nearest

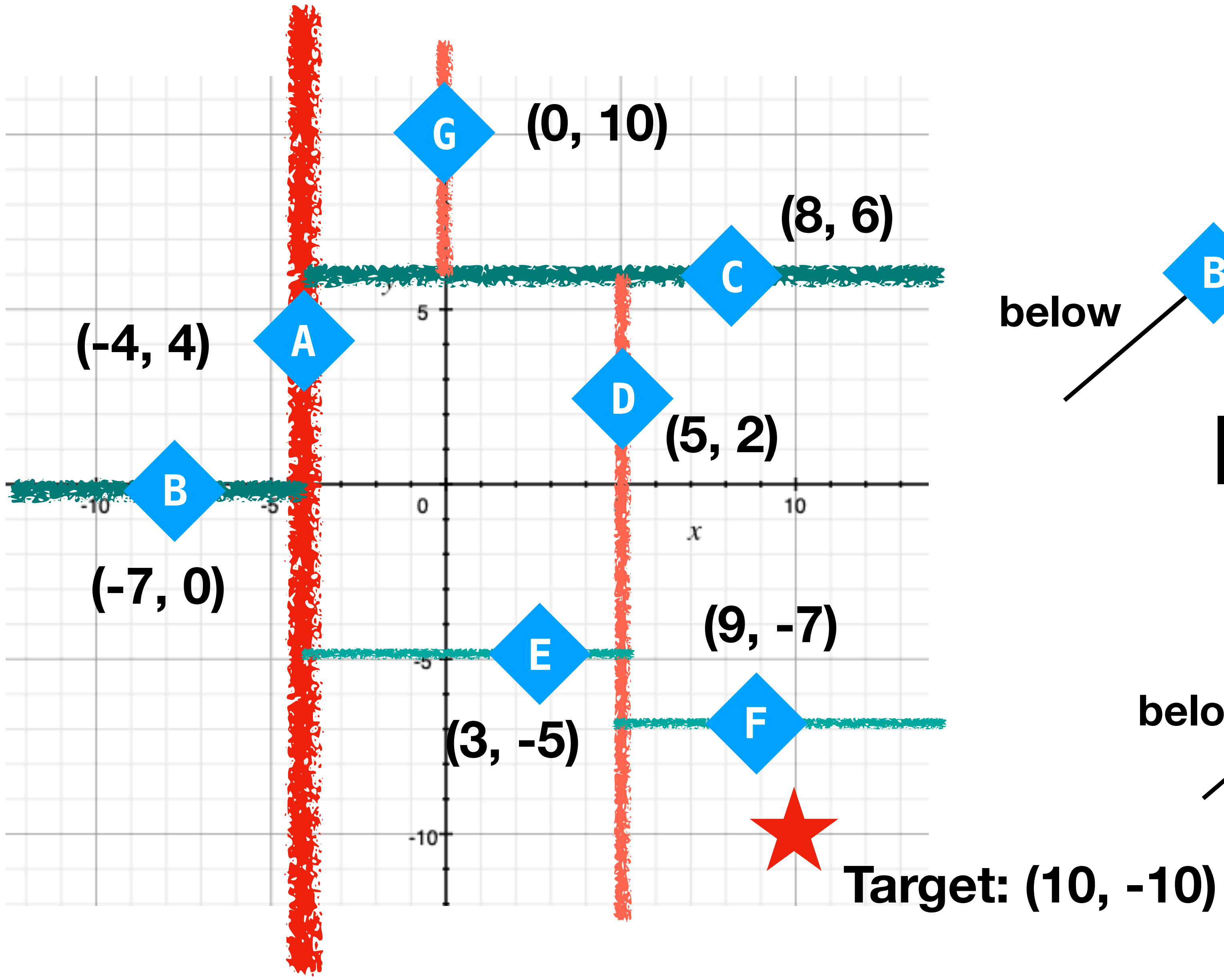
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

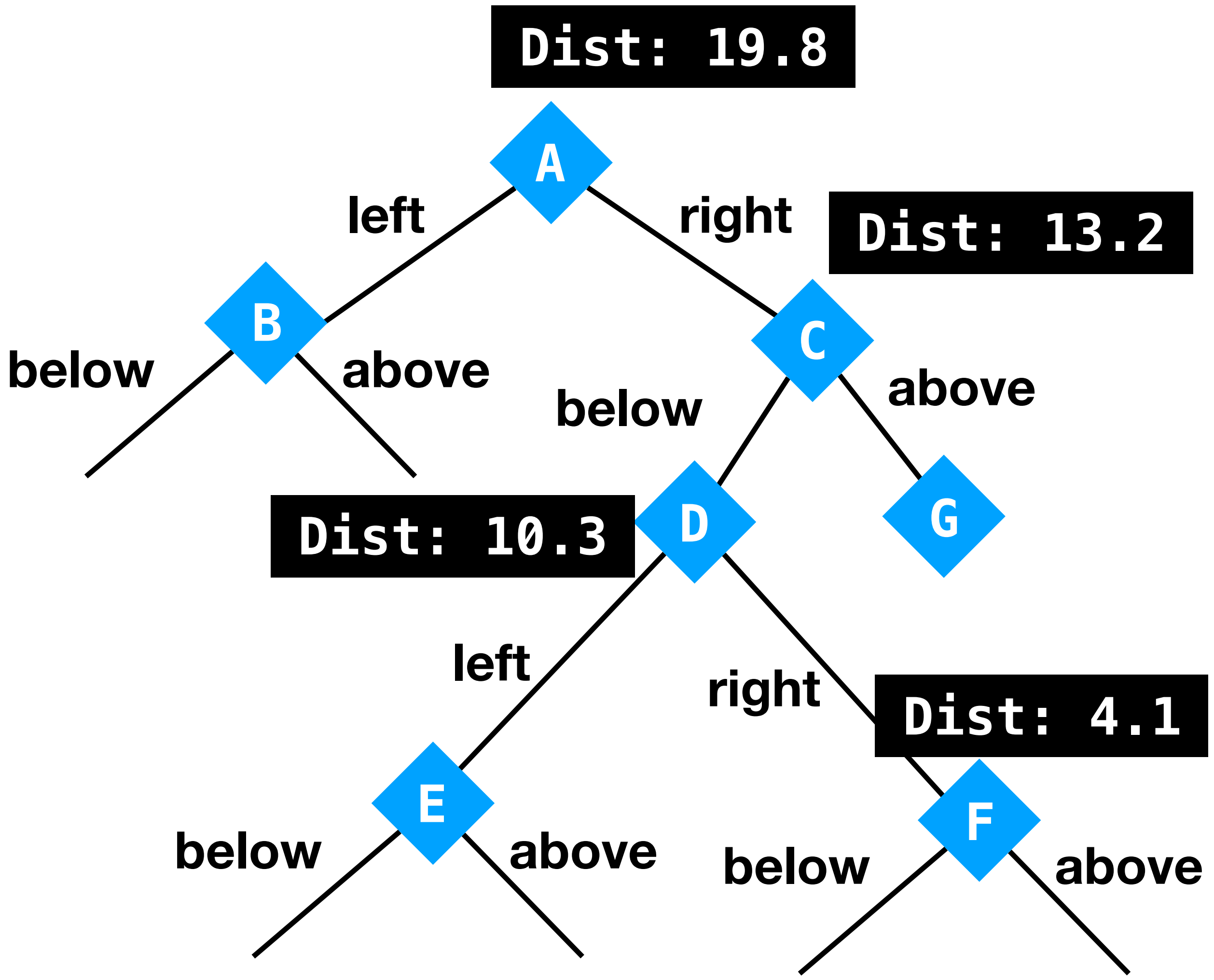




Nearest

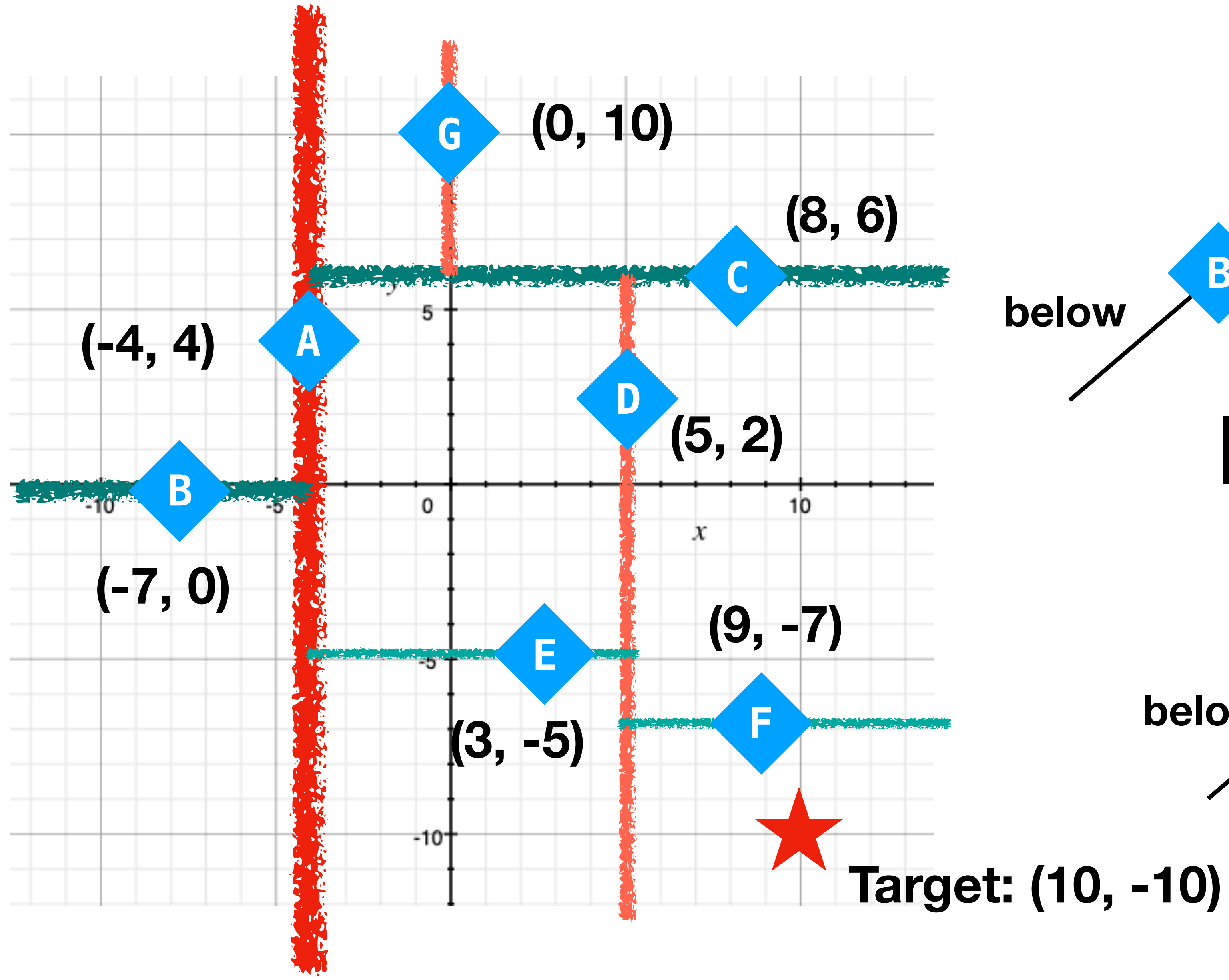
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

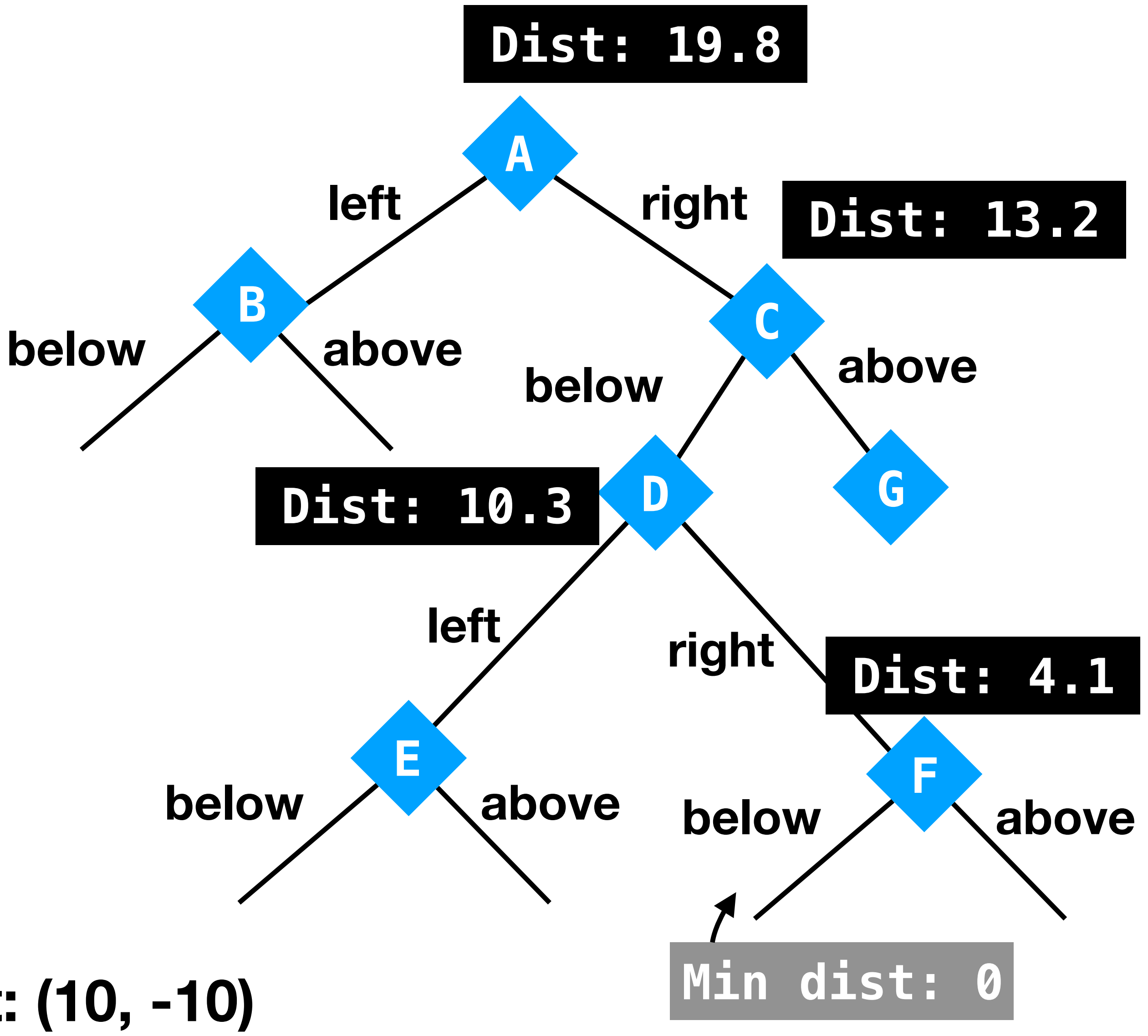




Nearest

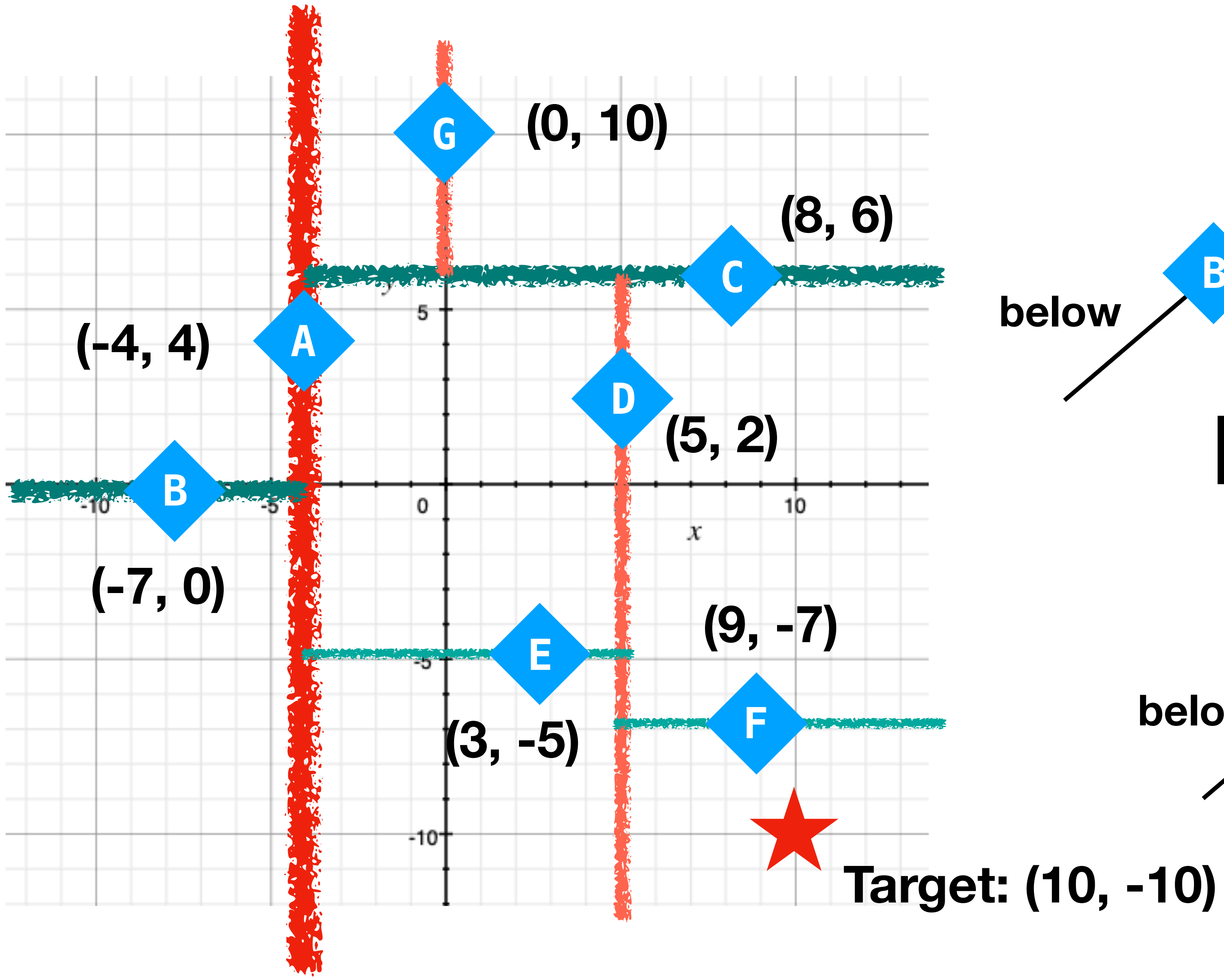
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

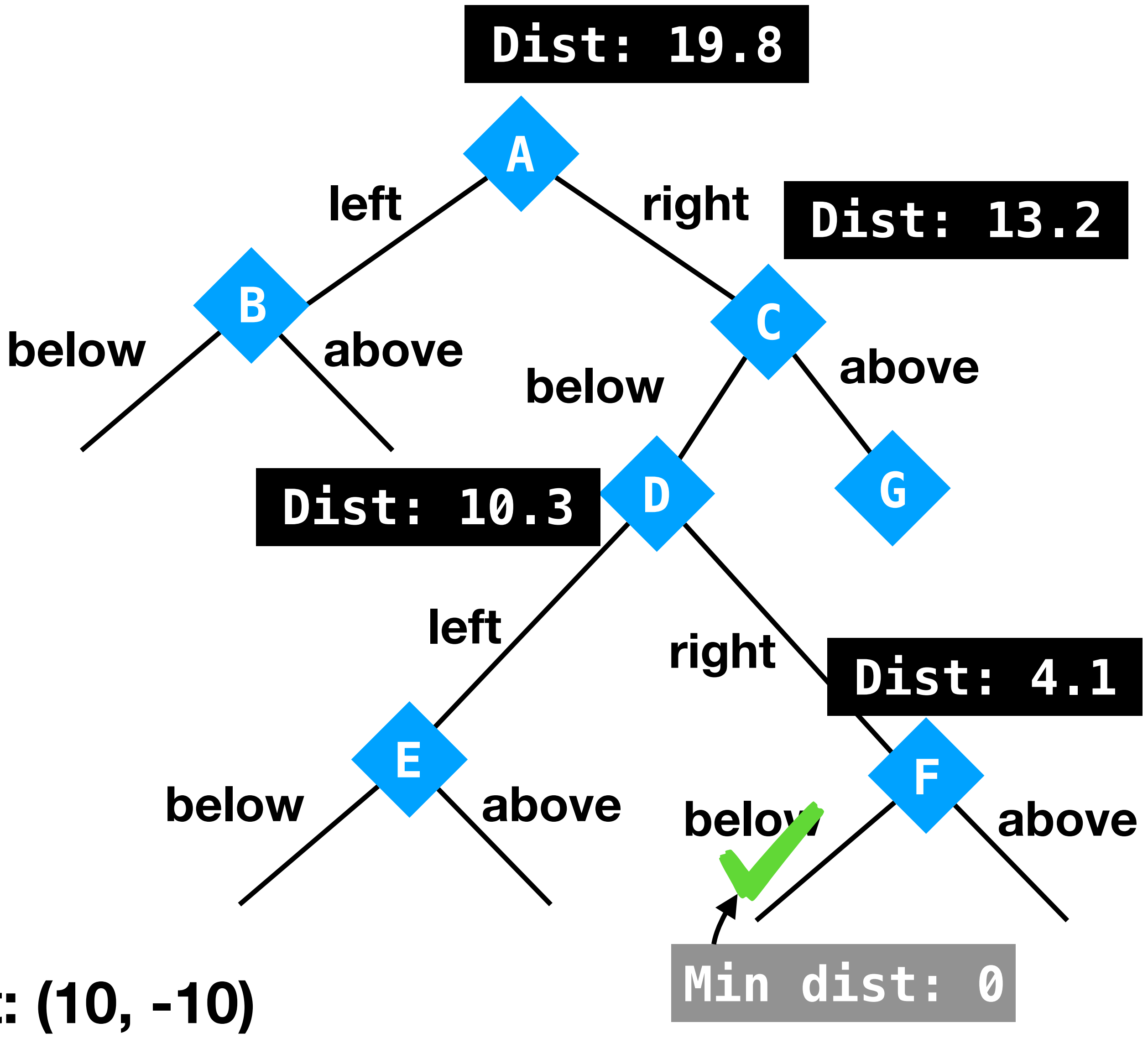




Nearest

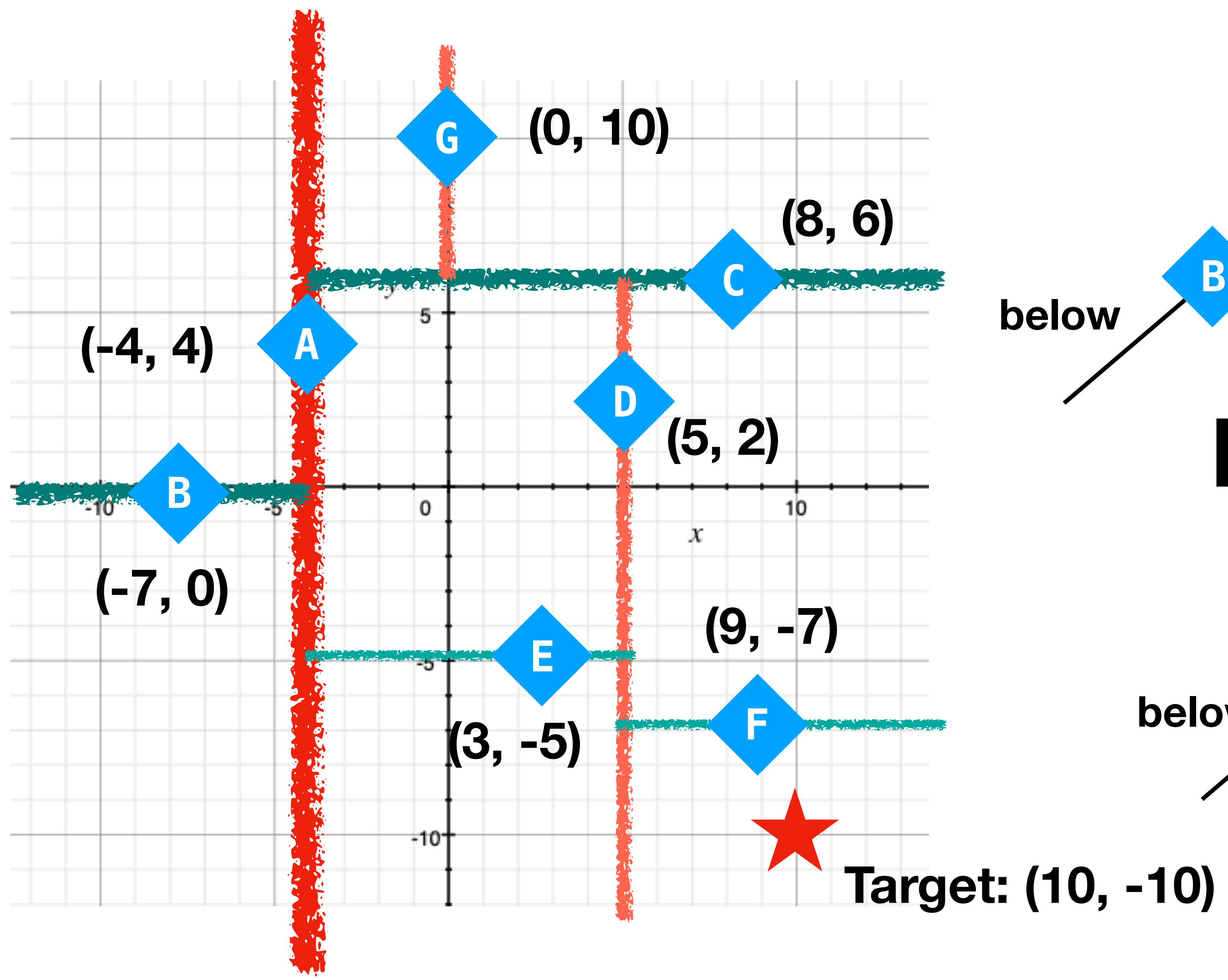
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

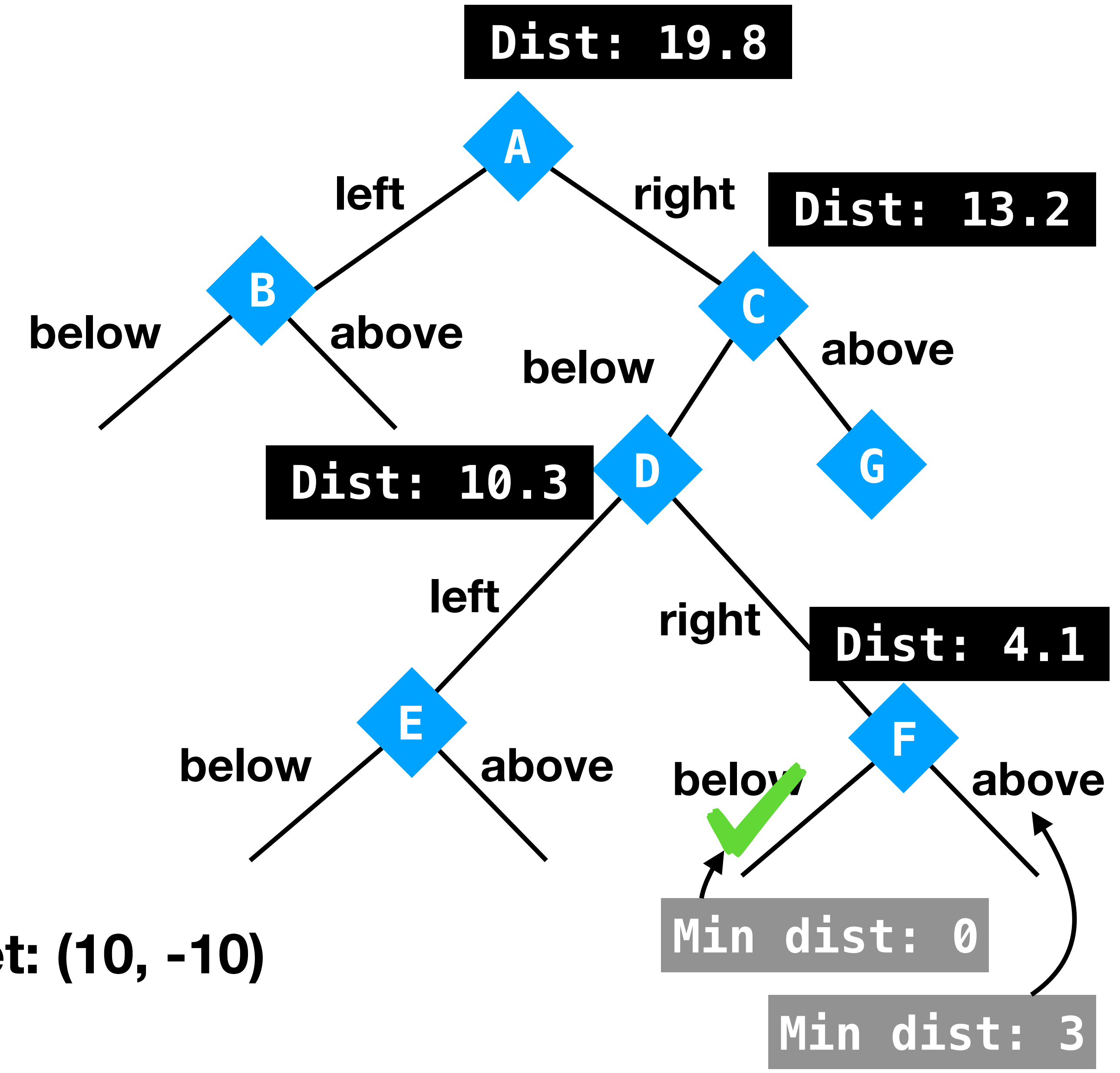




Nearest

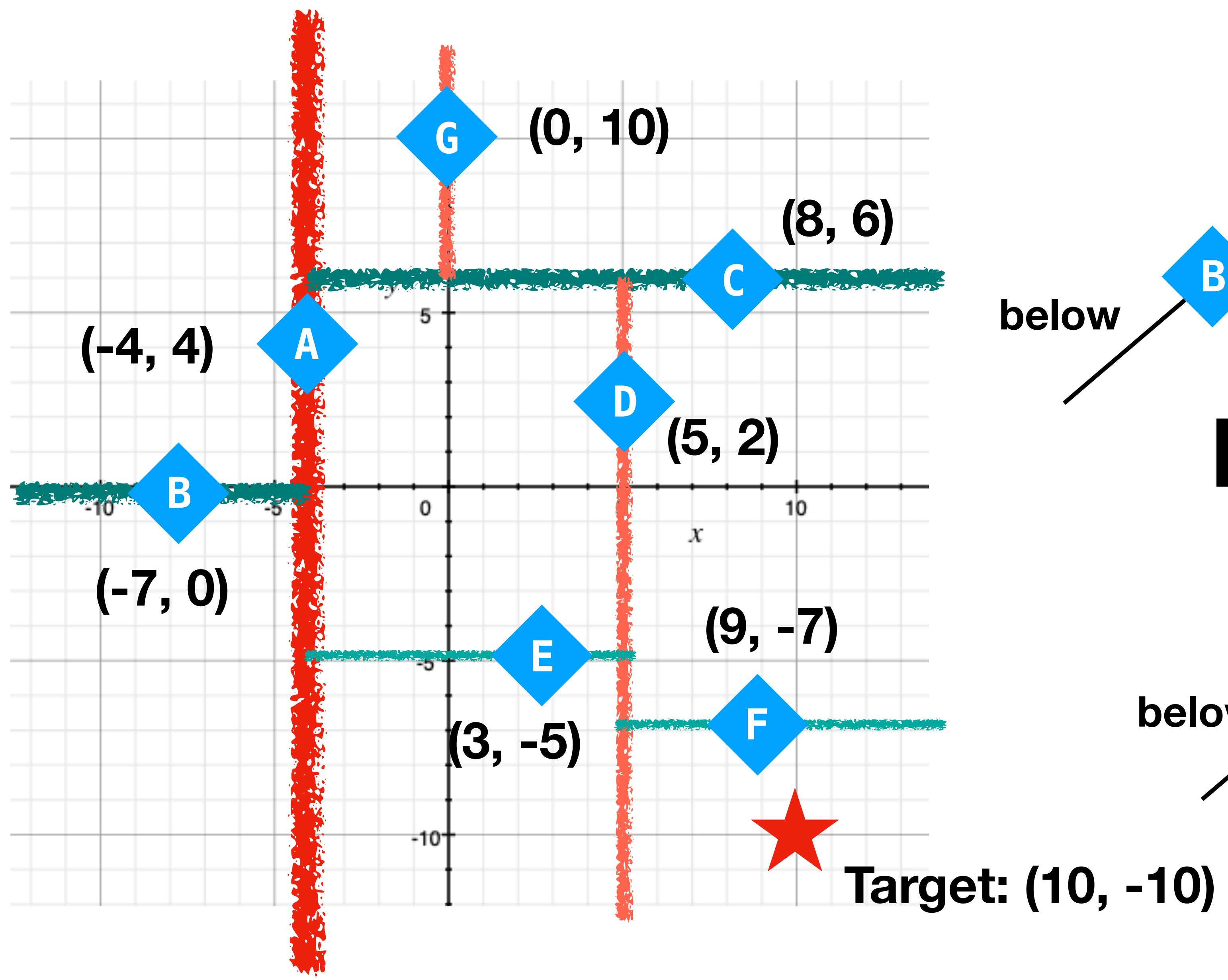
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

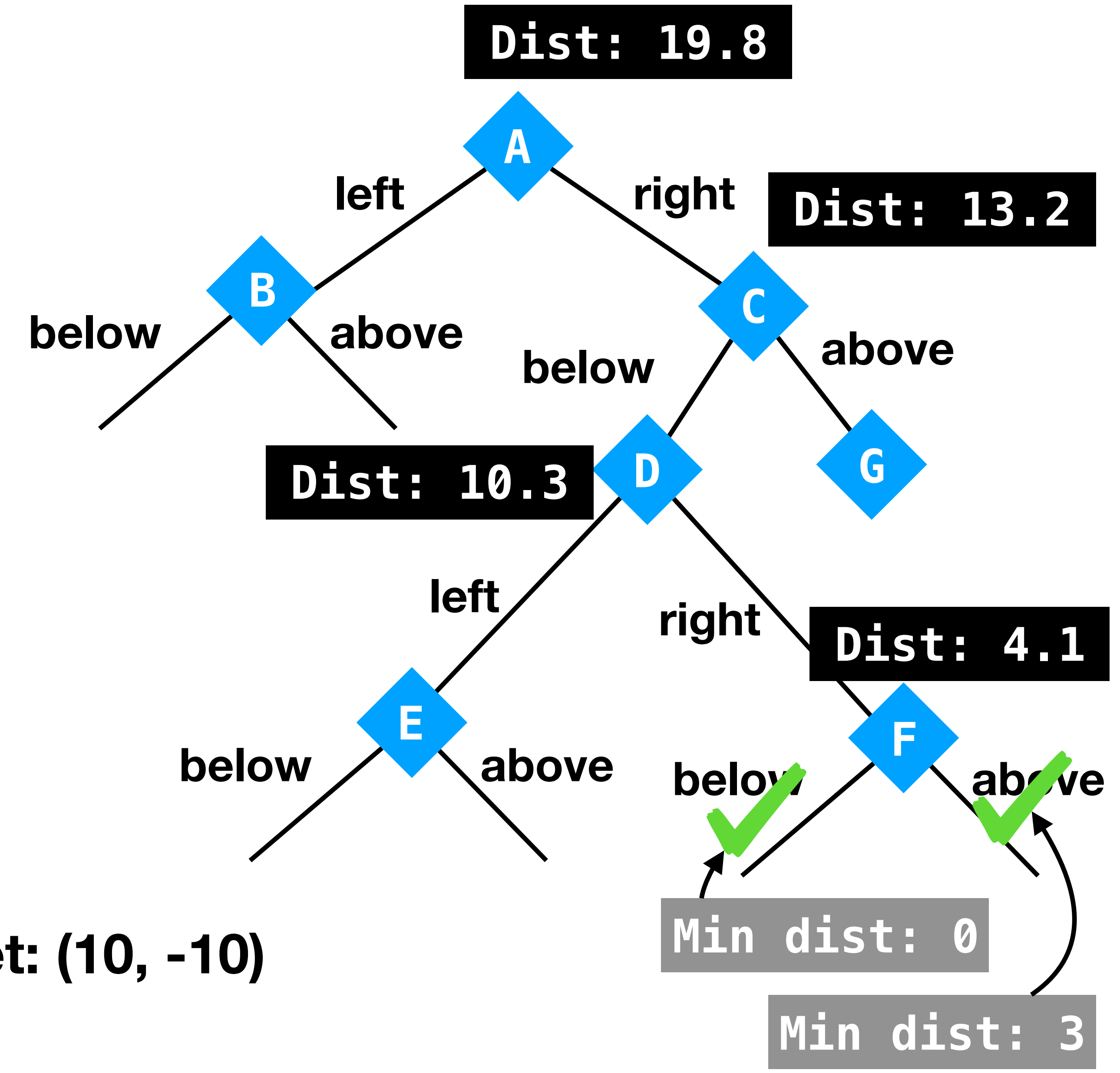




Nearest

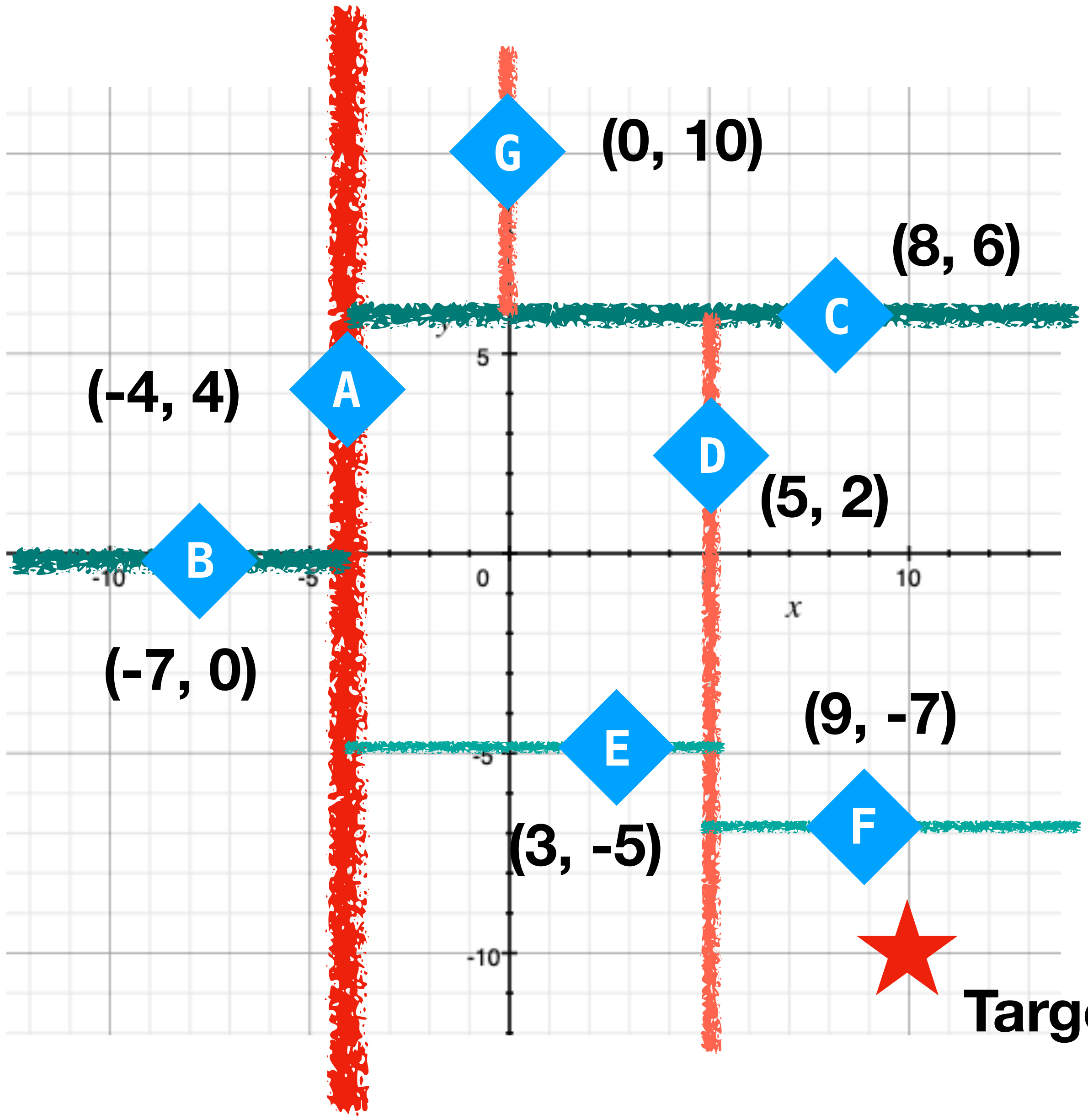
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

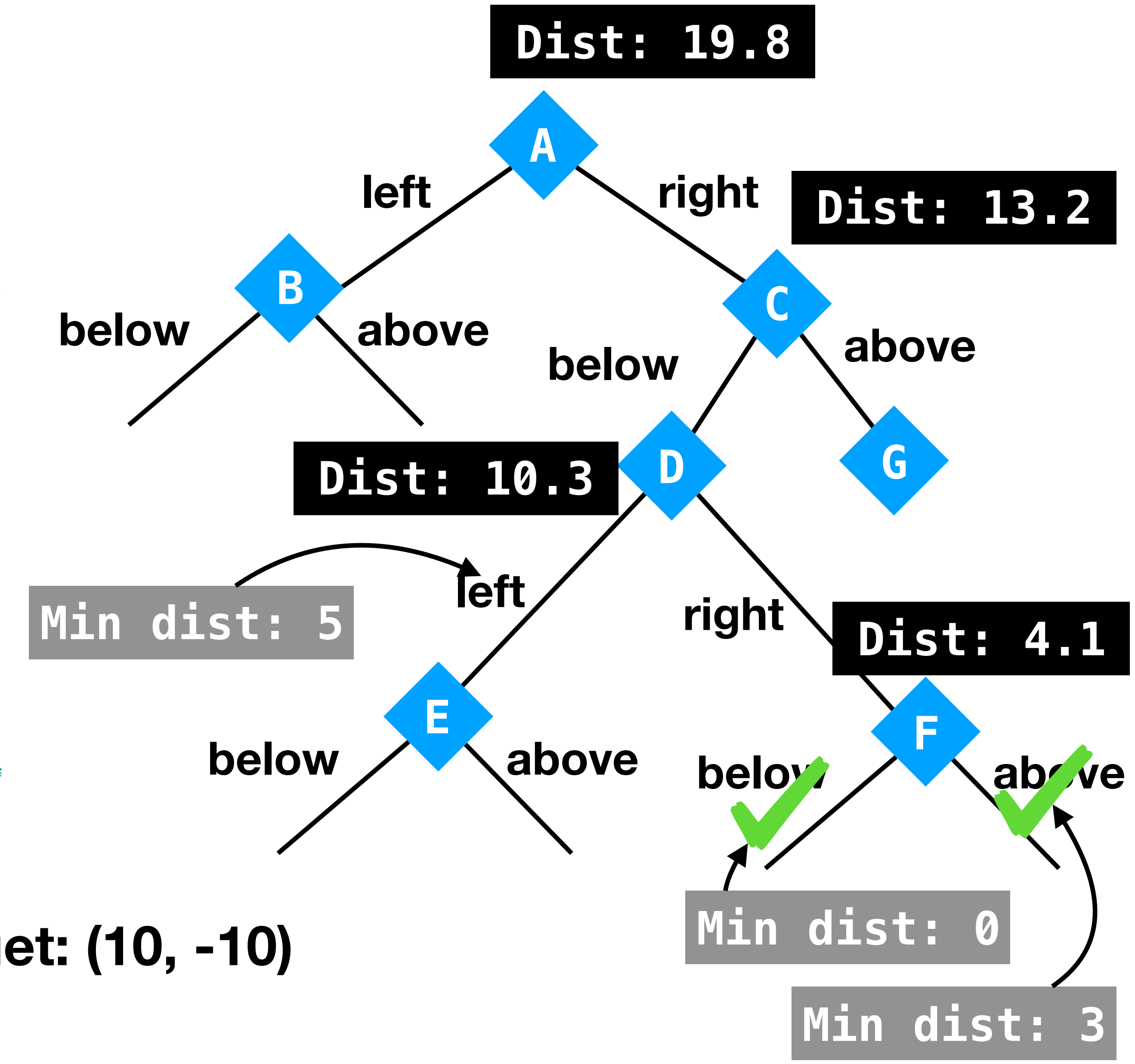




Nearest

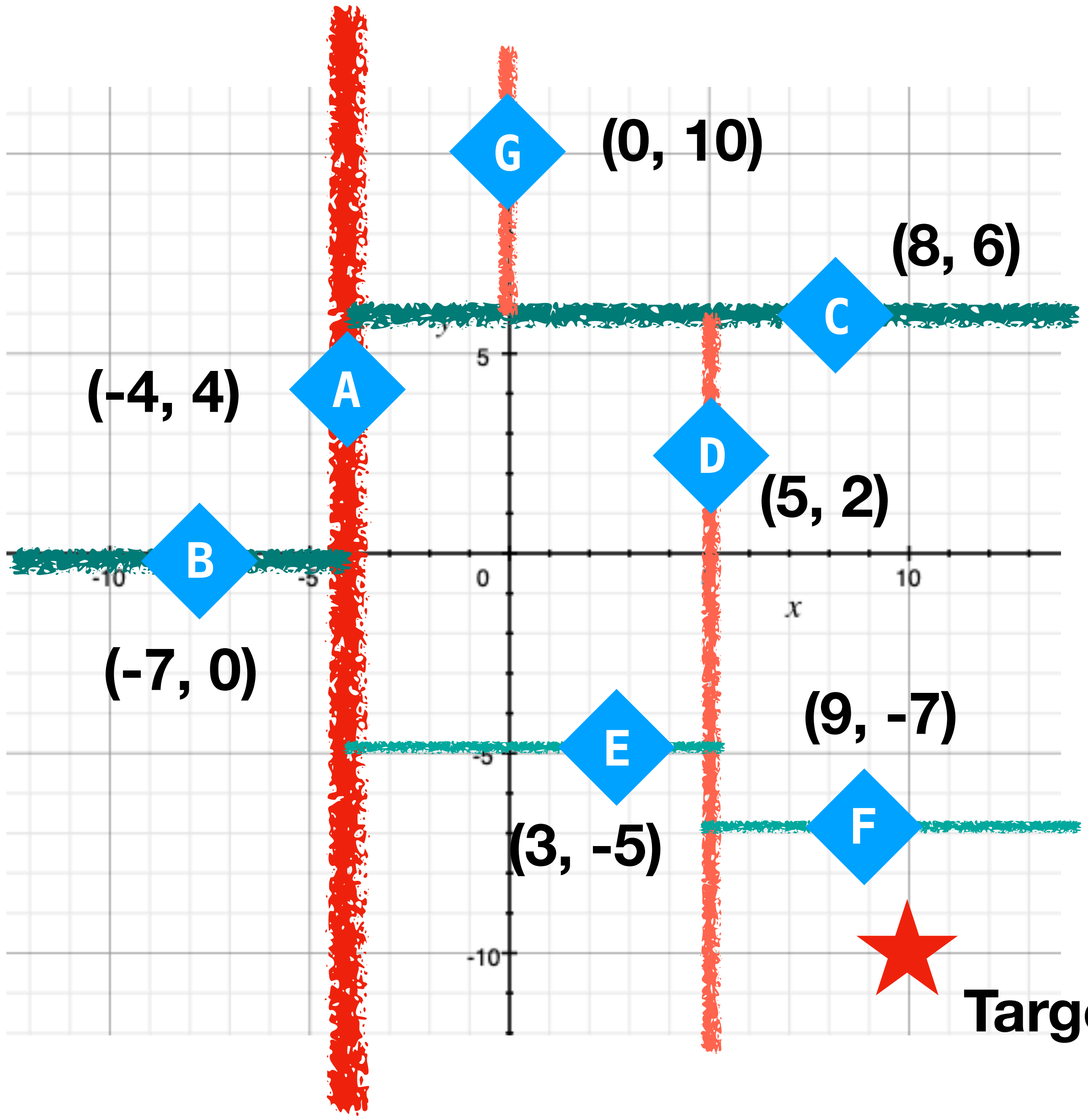
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.



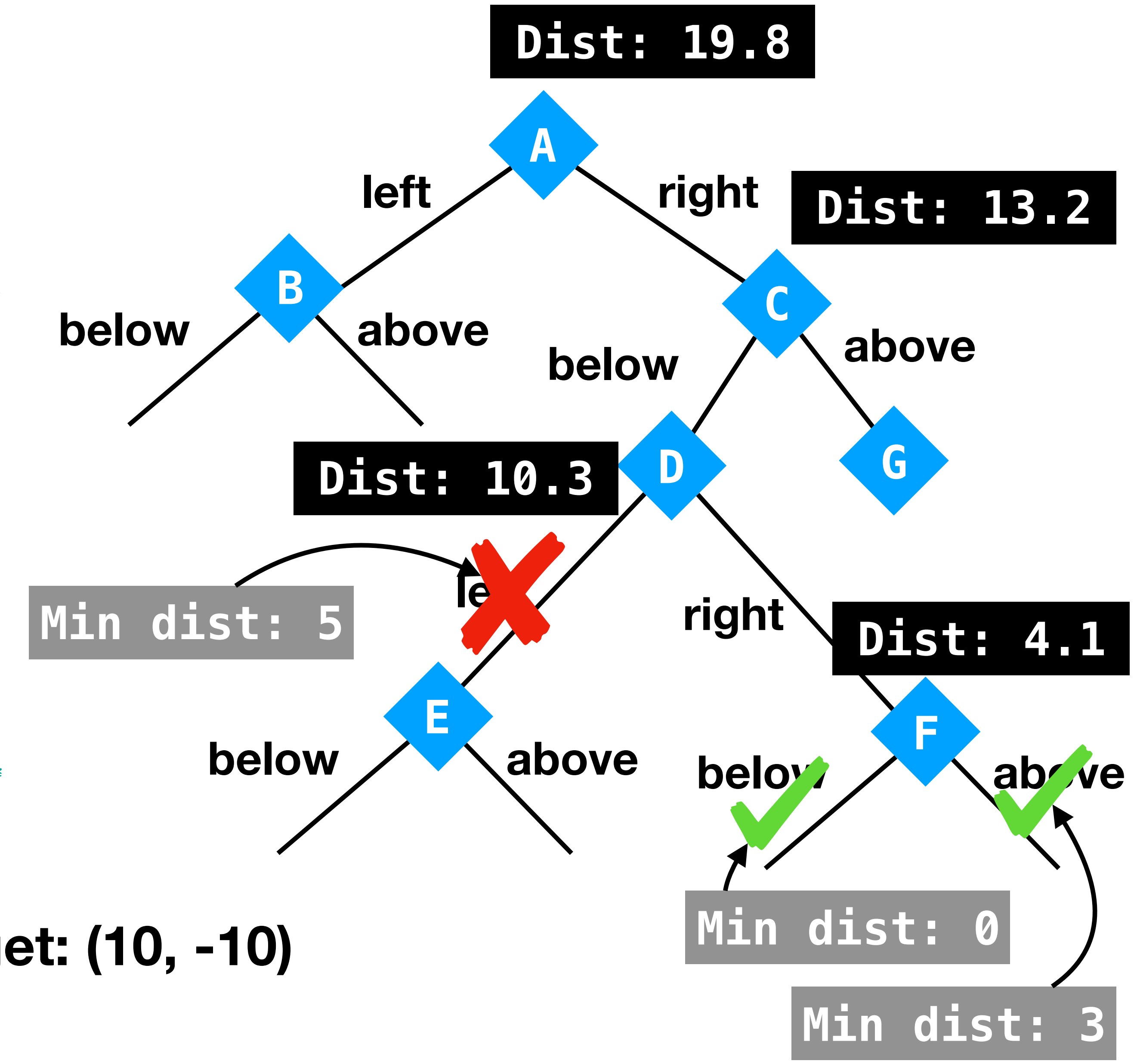


Nearest

- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

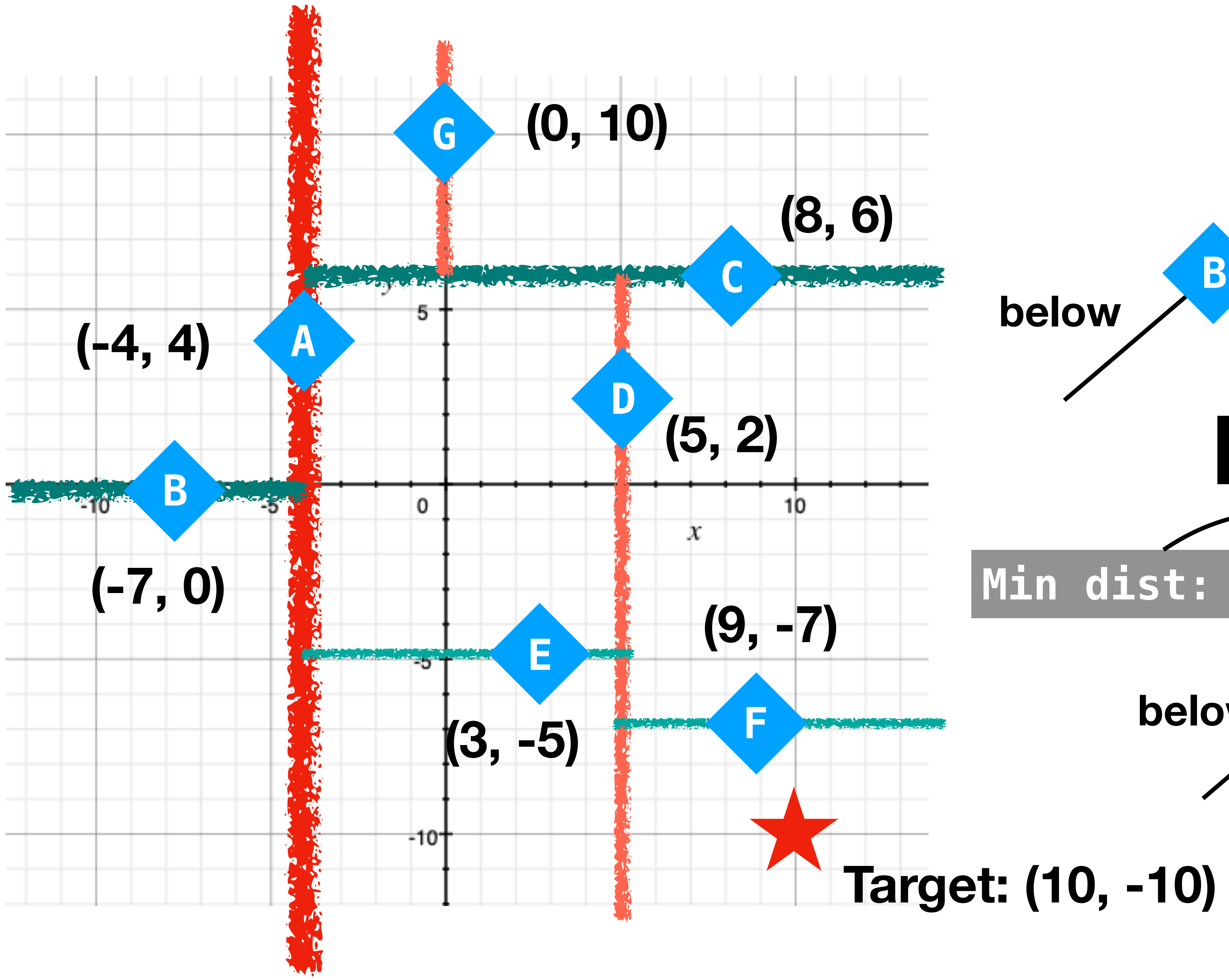


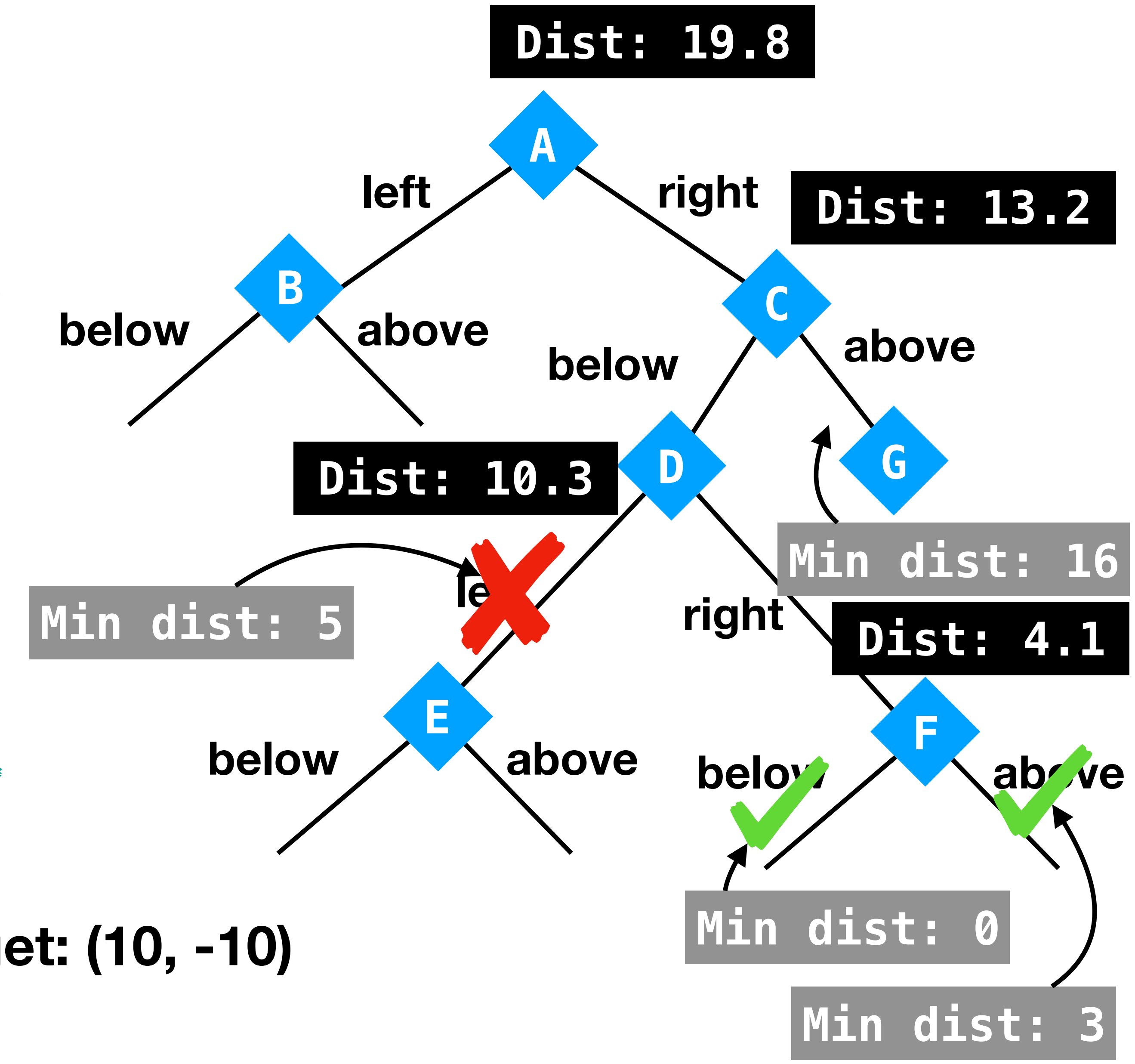
Target: (10, -10)



Nearest

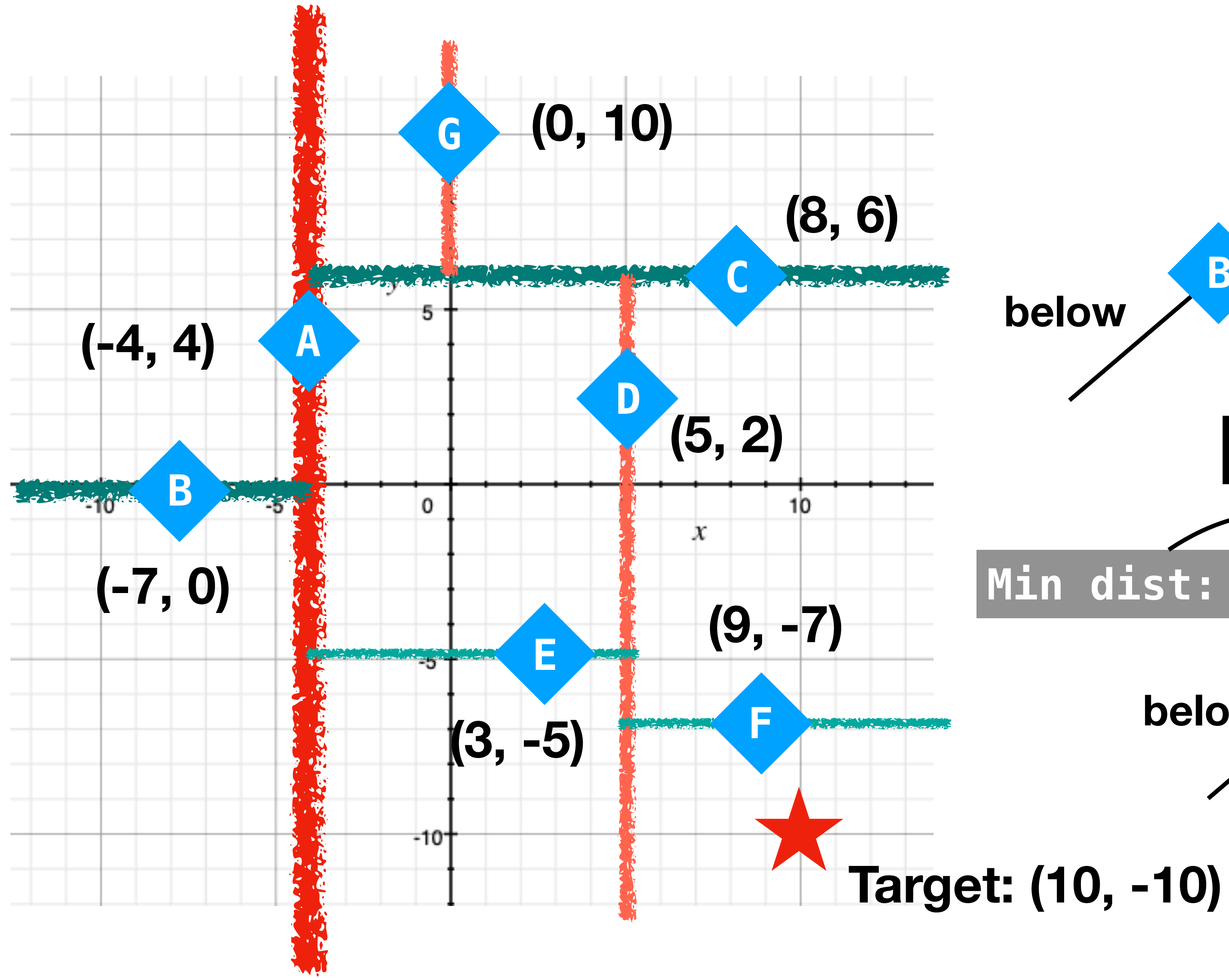
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

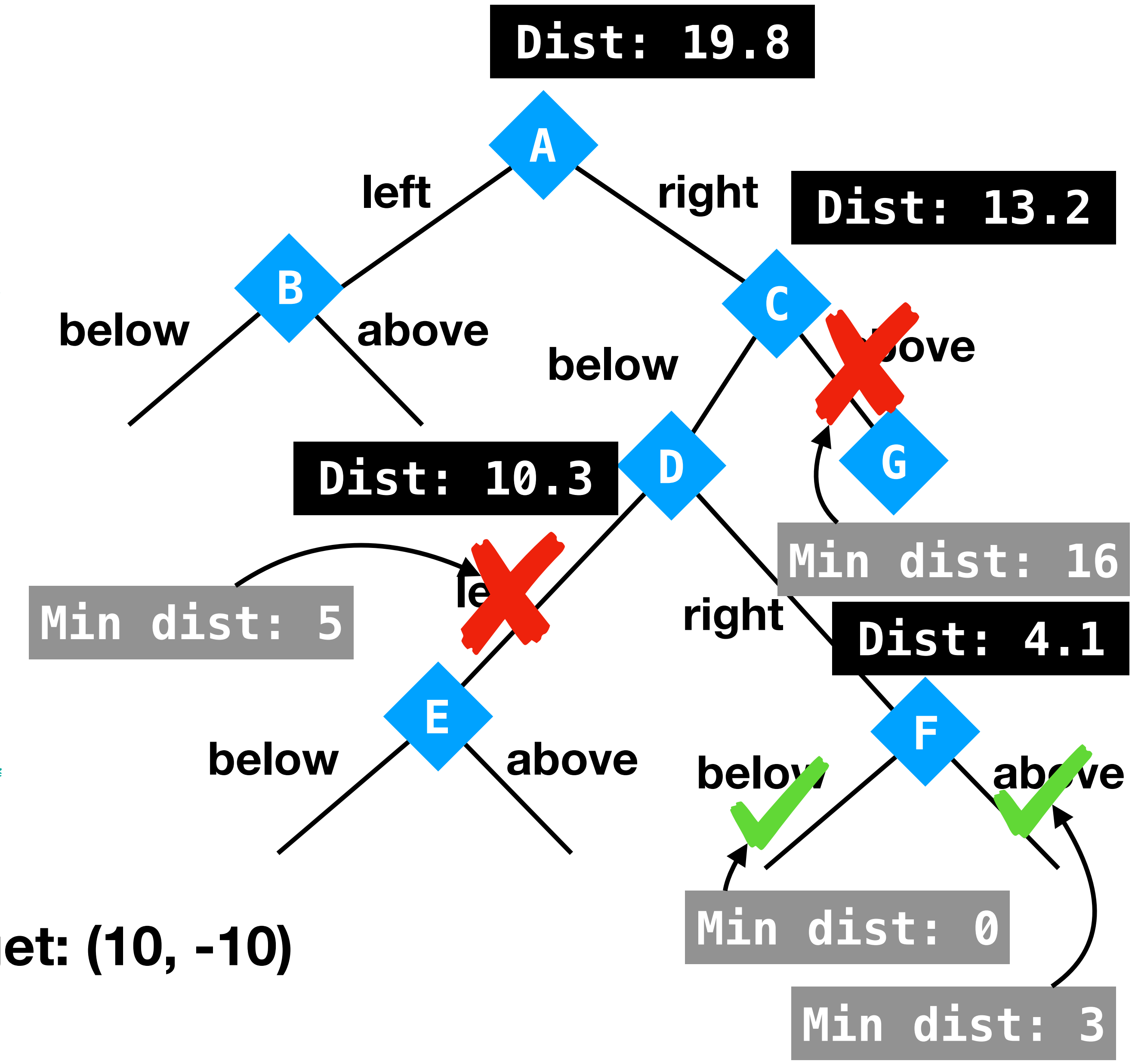




Nearest

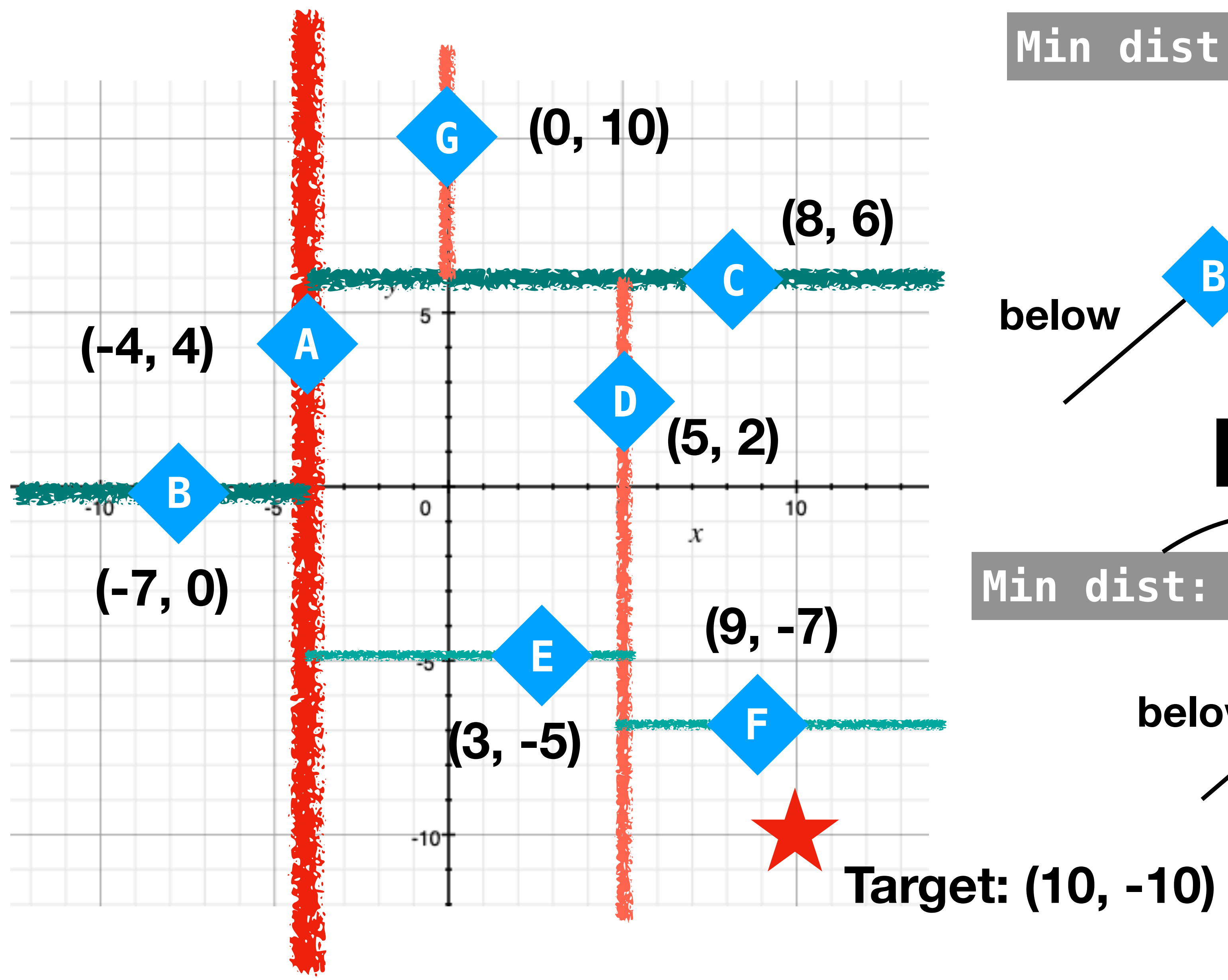
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

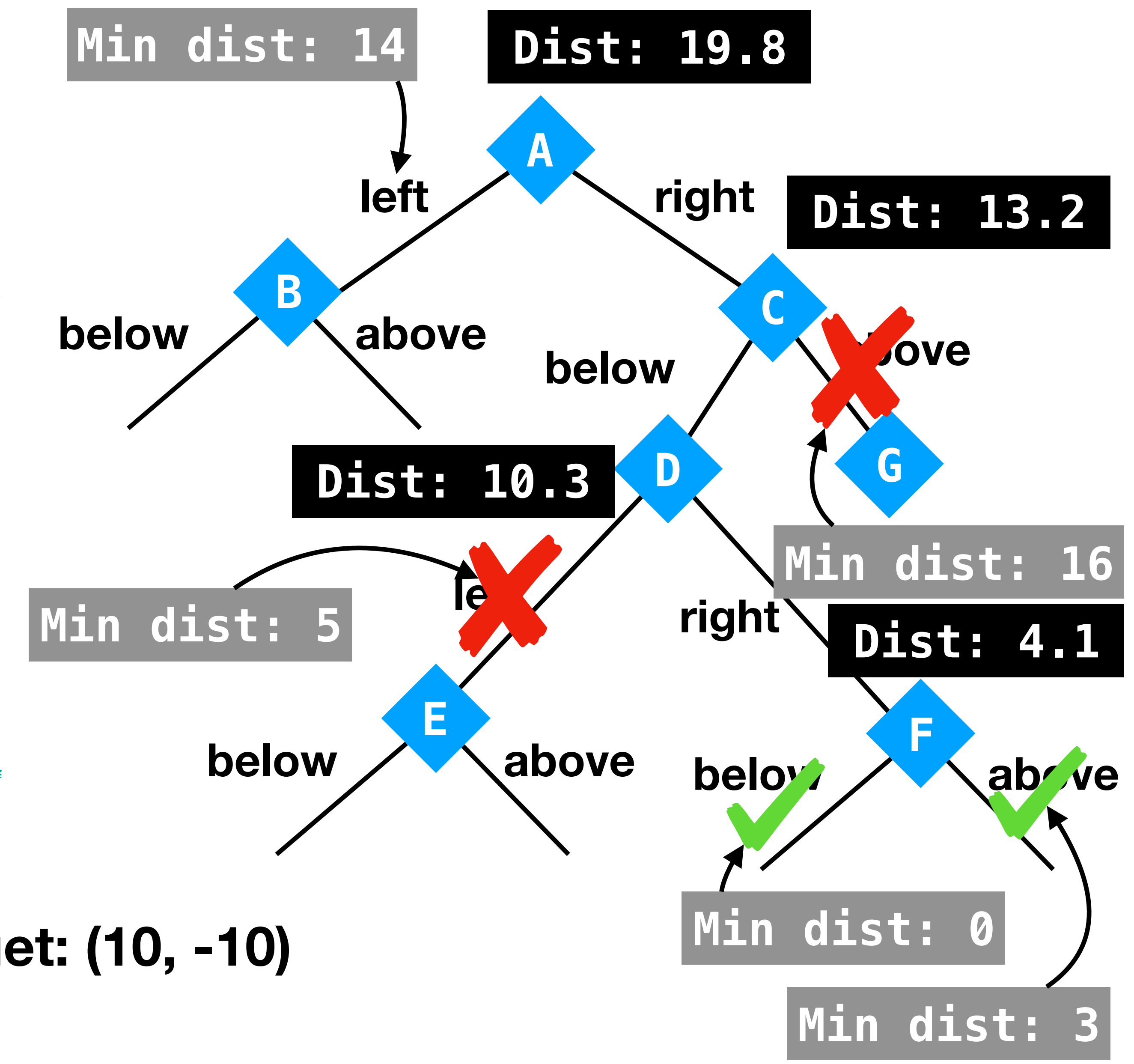




Nearest

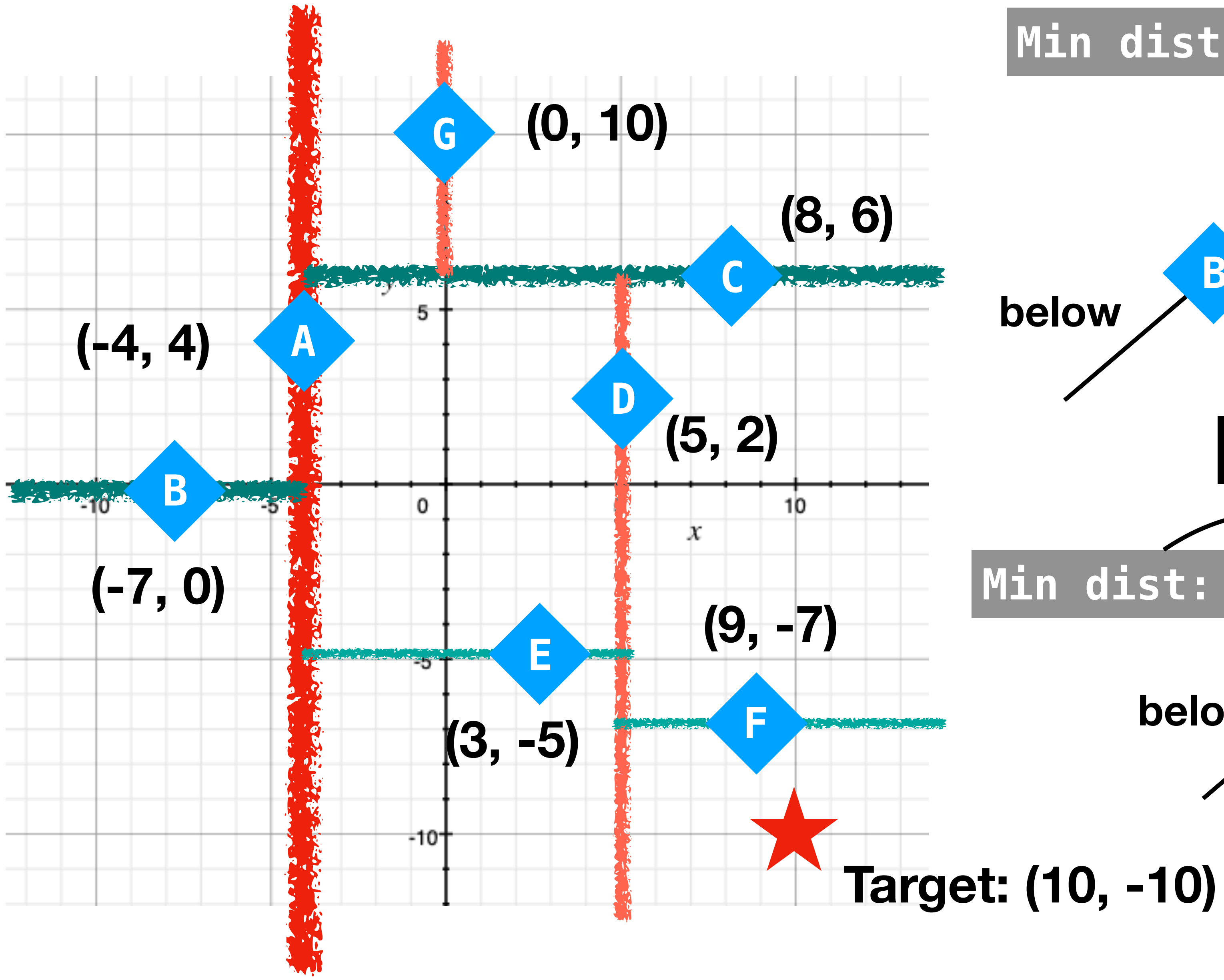
- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

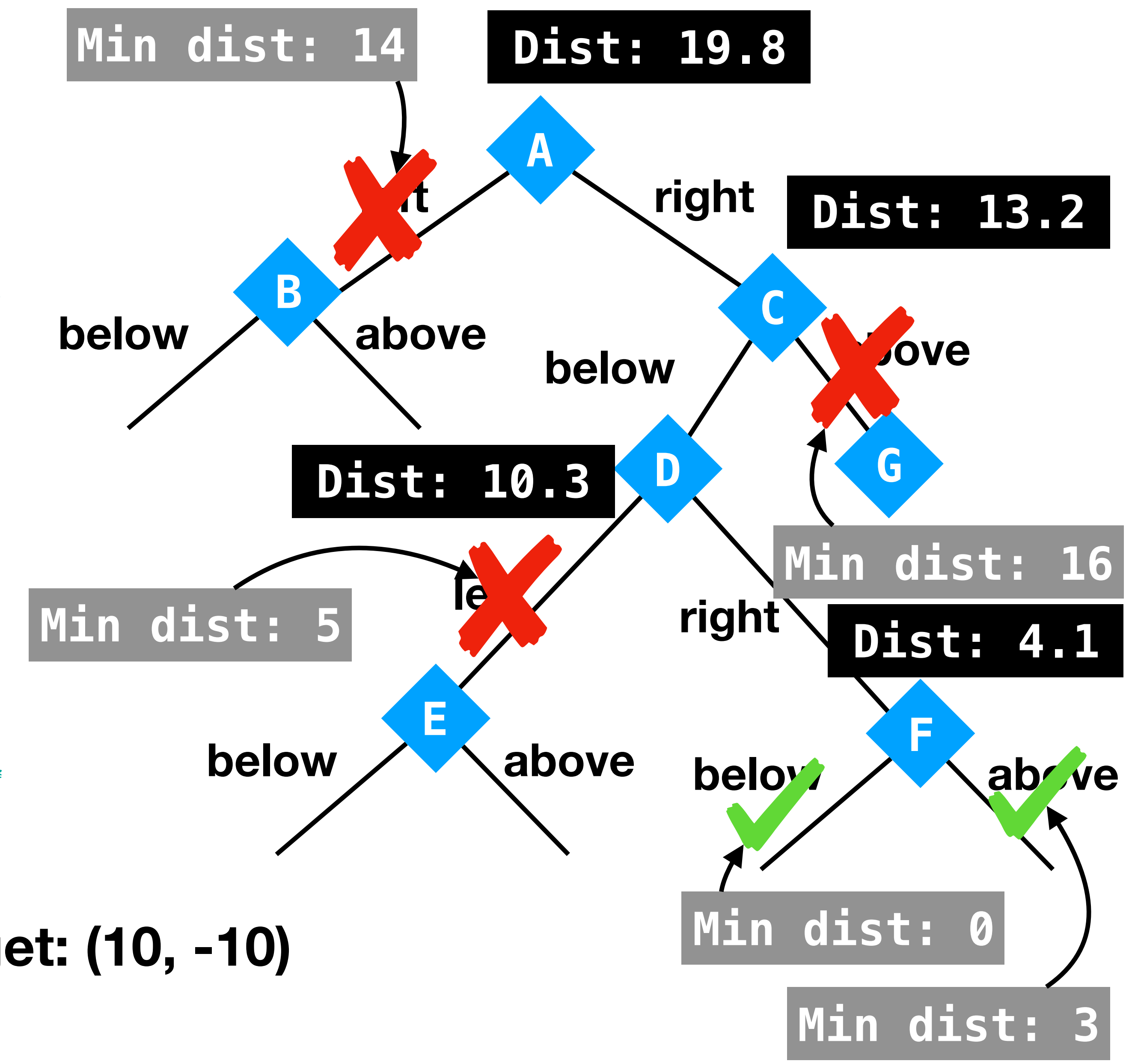




Nearest

- Idea: Search though tree, pruning branches that can't possibly have better distance than one found so far. Check "better" side first.

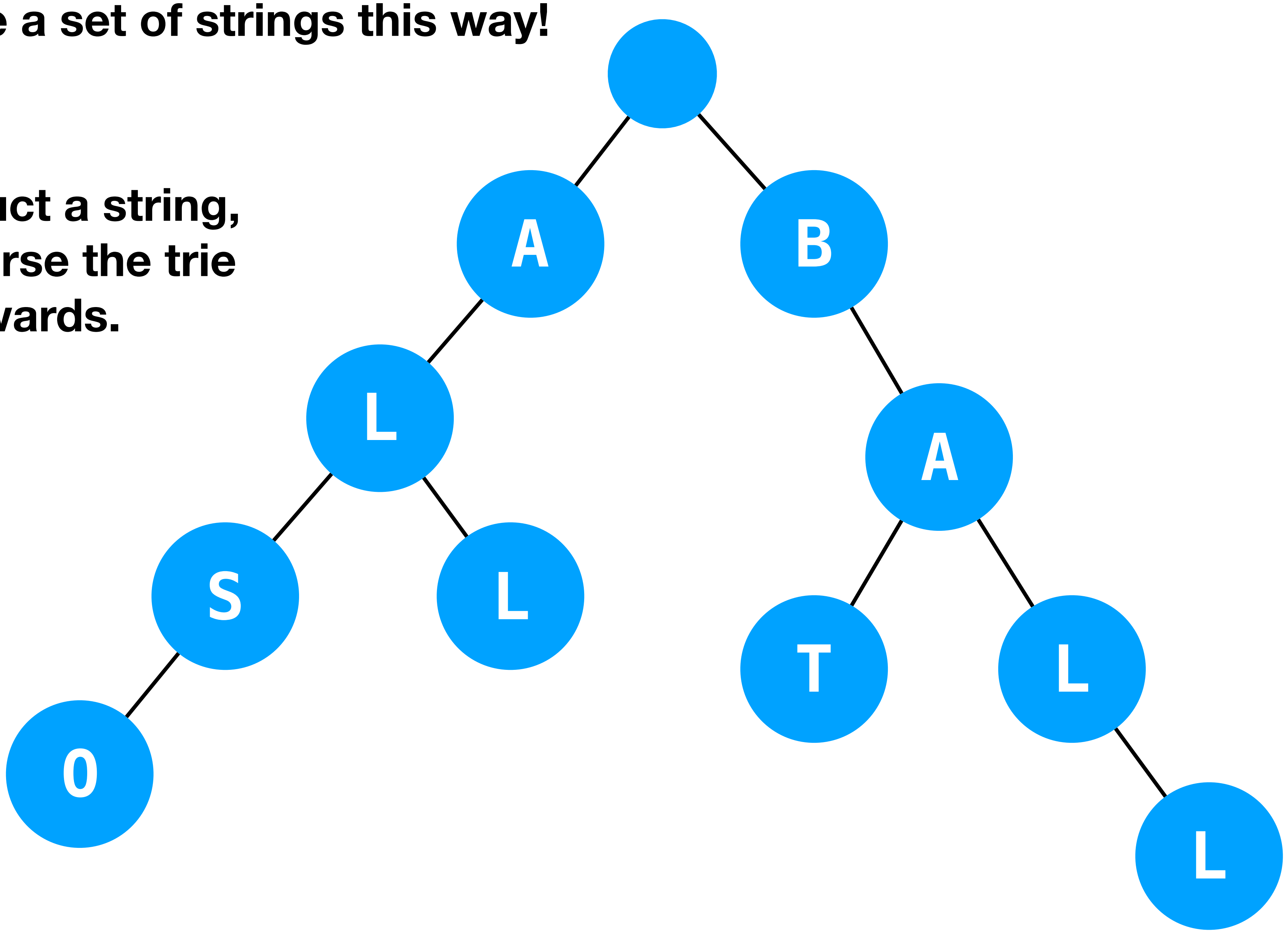




Tries

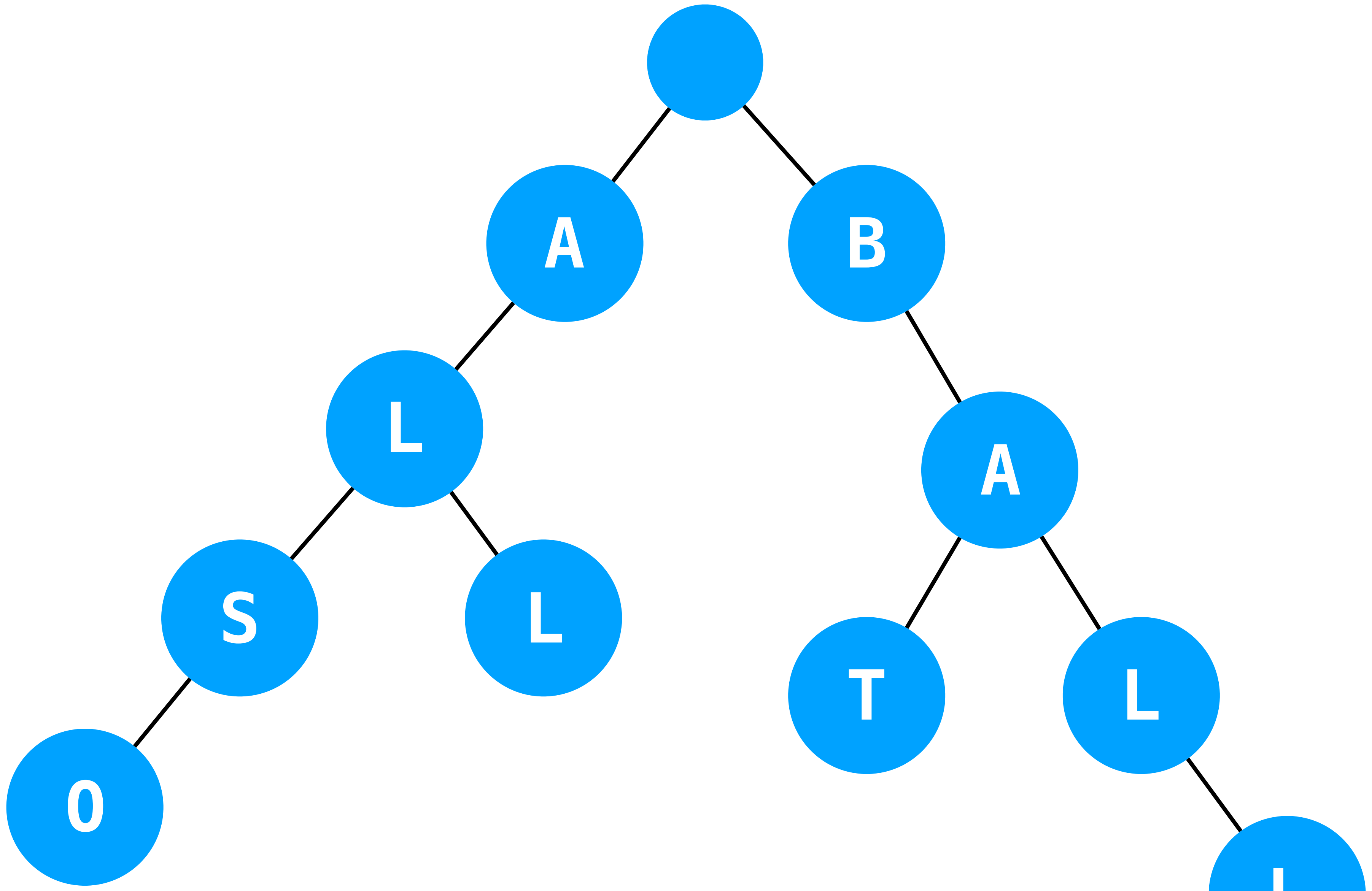
Idea: Store a single letter at each node.
We can make a set of strings this way!

To reconstruct a string, simply traverse the trie downwards.



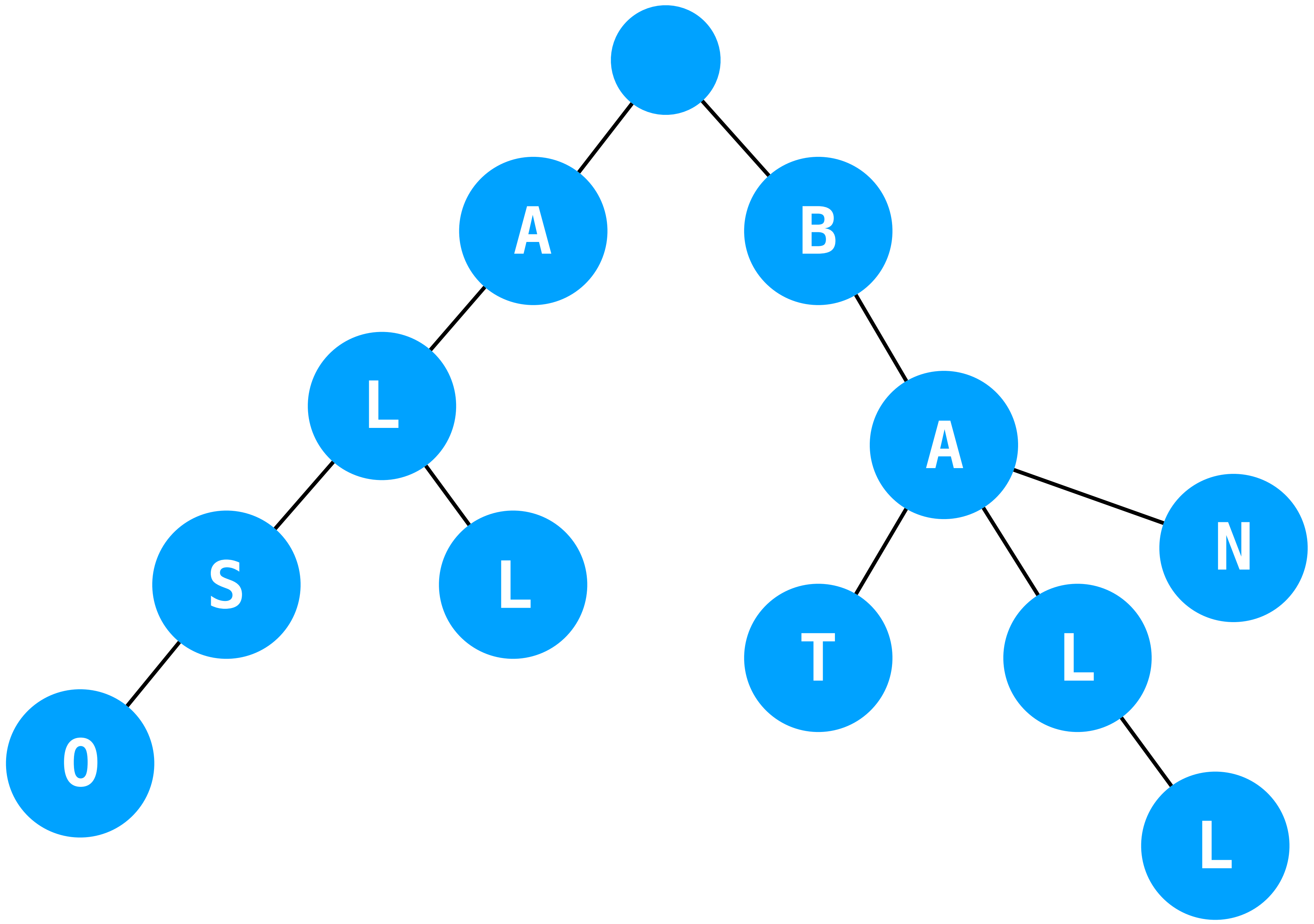
Tries

Add "BAND" to this trie.



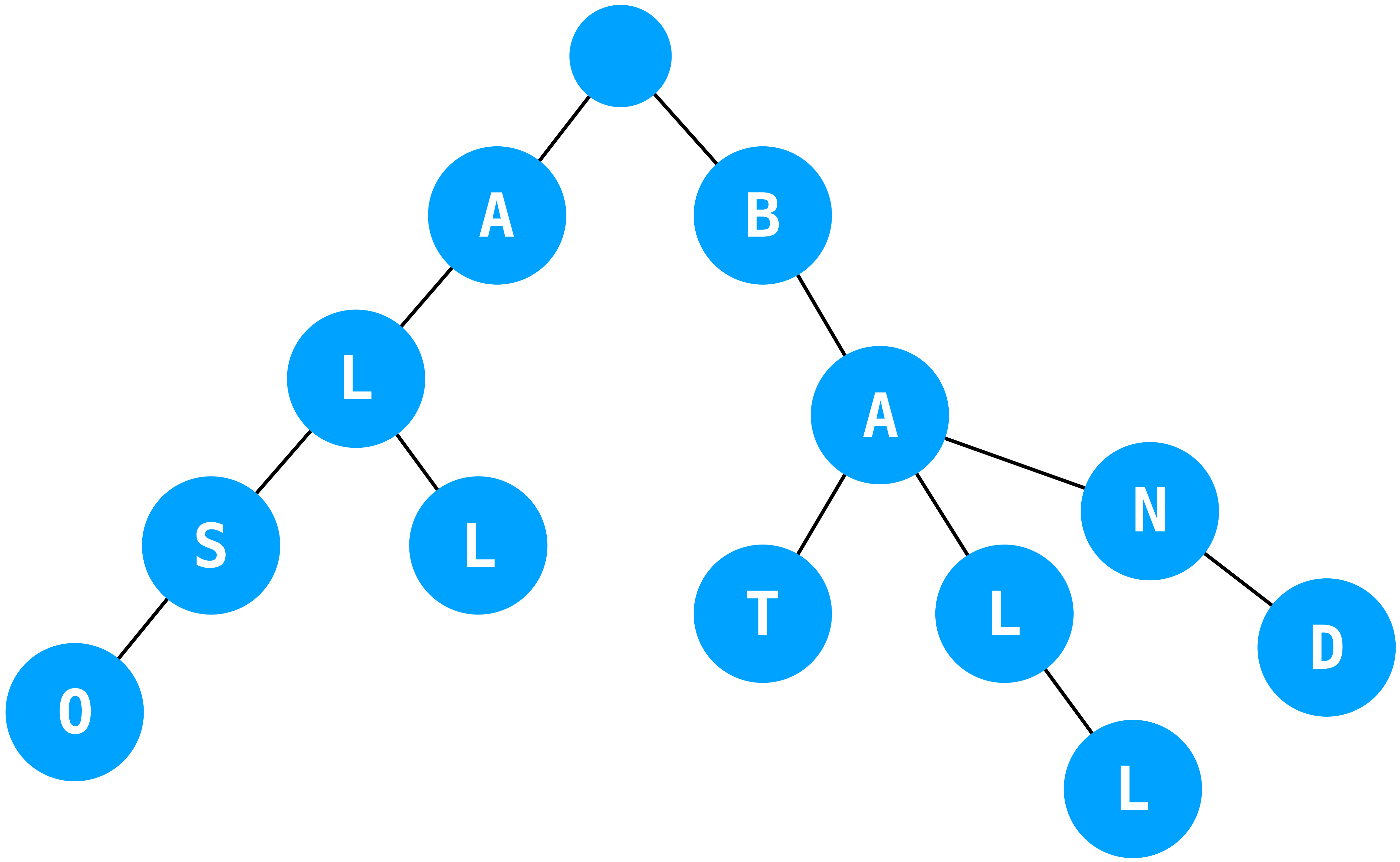
Tries

Add "BAND" to this trie.



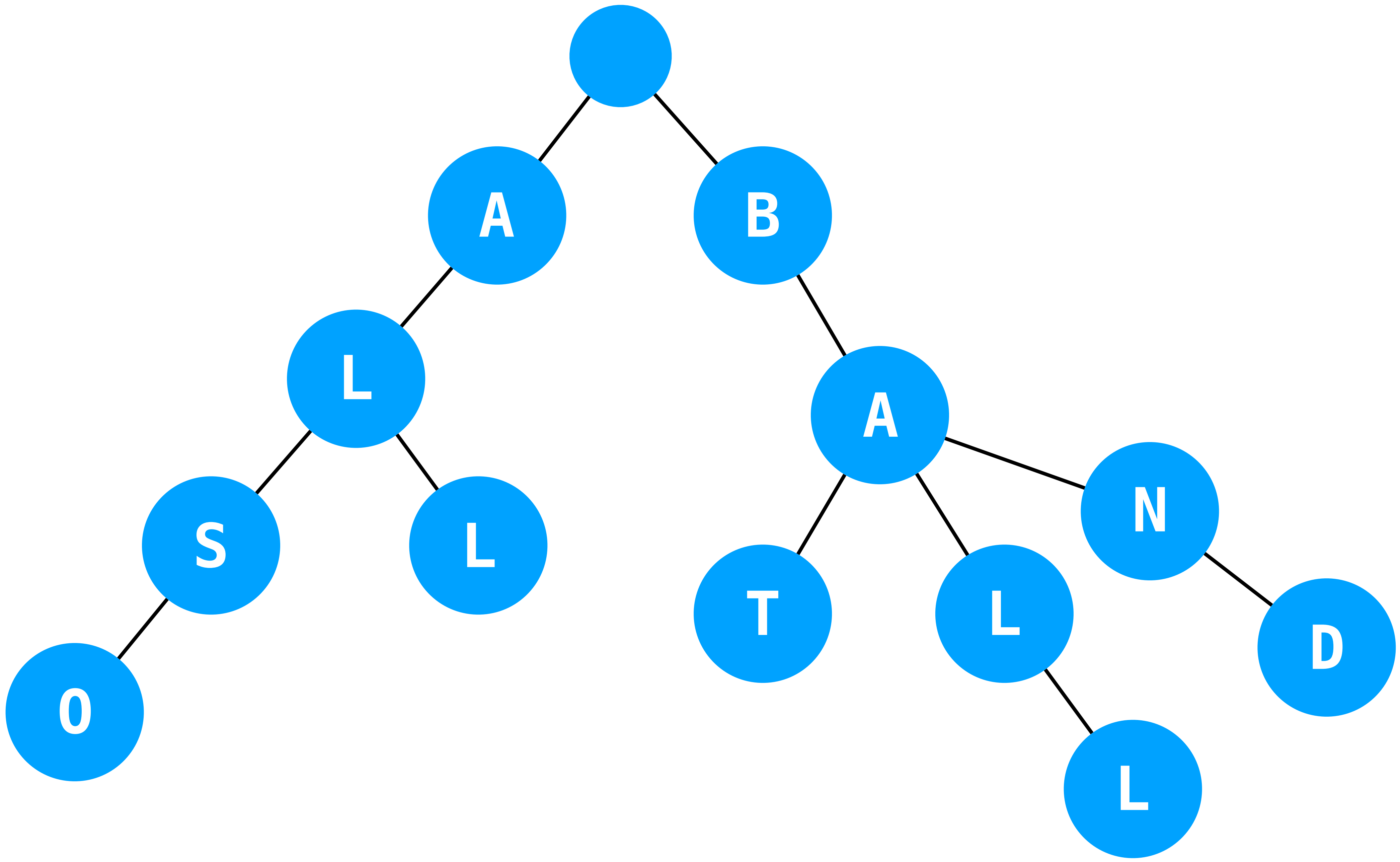
Tries

Add "BAND" to this trie.



Tries

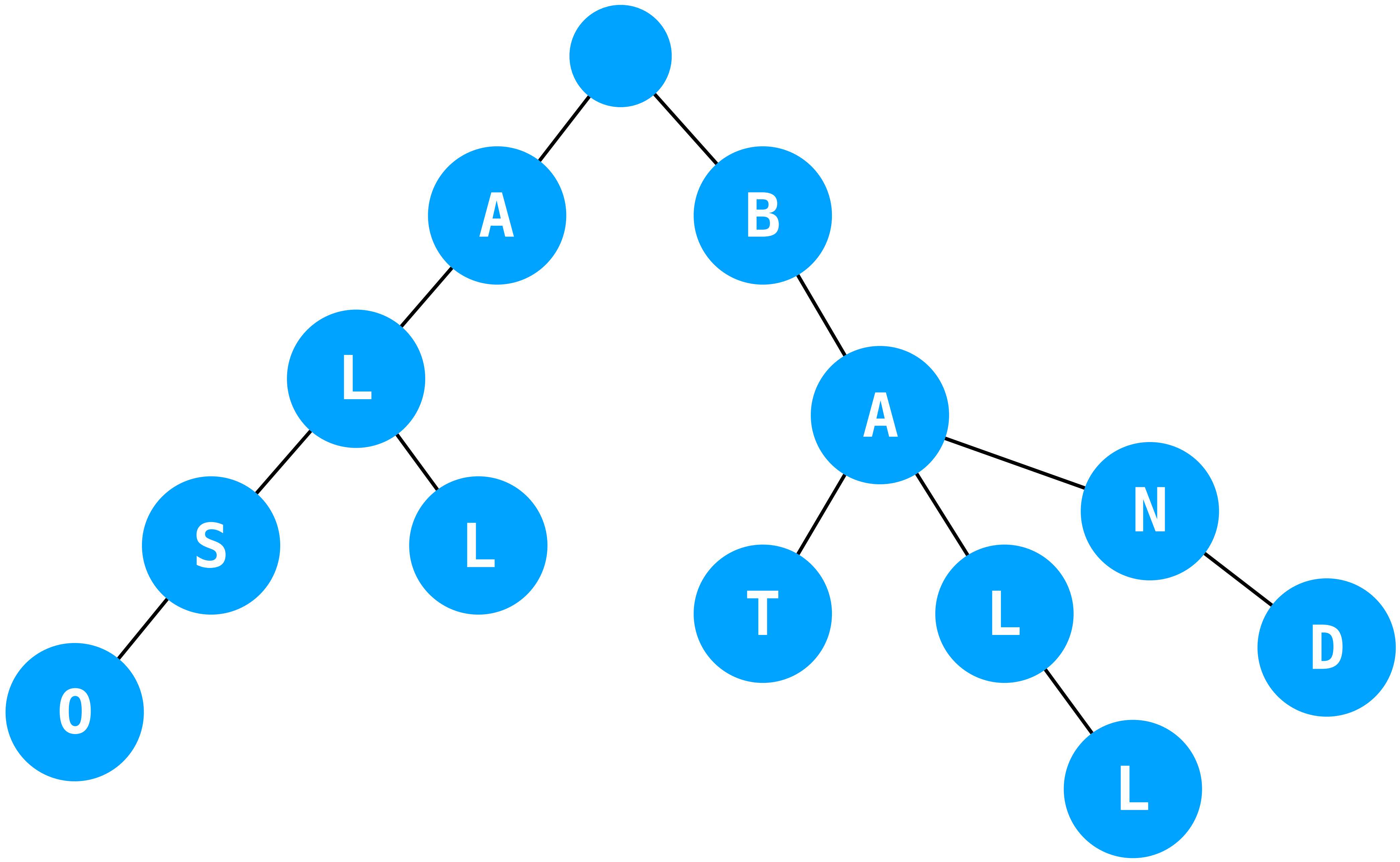
Add "BAN" to this trie.



Tries

Add "BAN" to this trie.

...oof

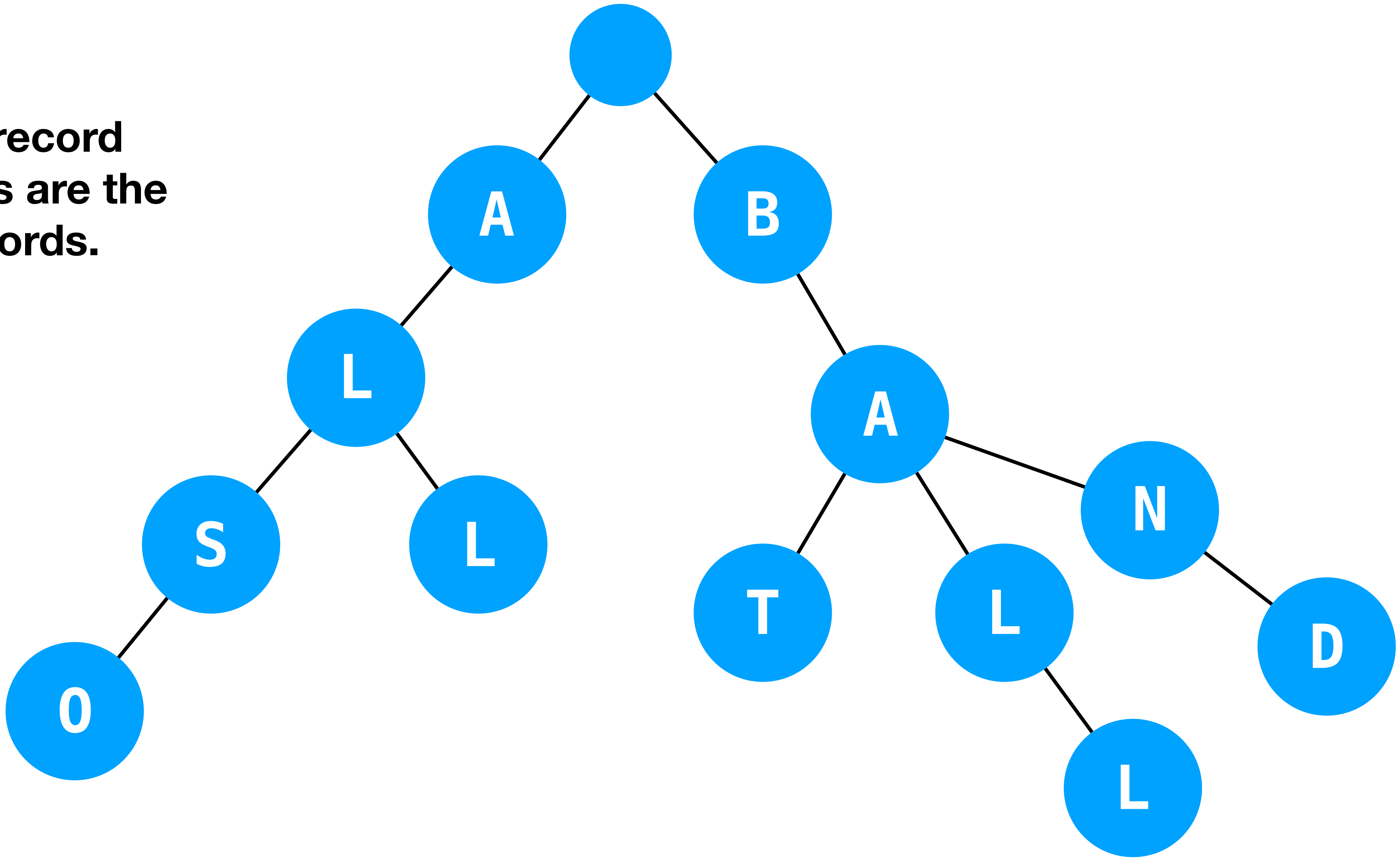


Tries

Add "BAN" to this trie.

...oof

Solution: record which nodes are the end of words.

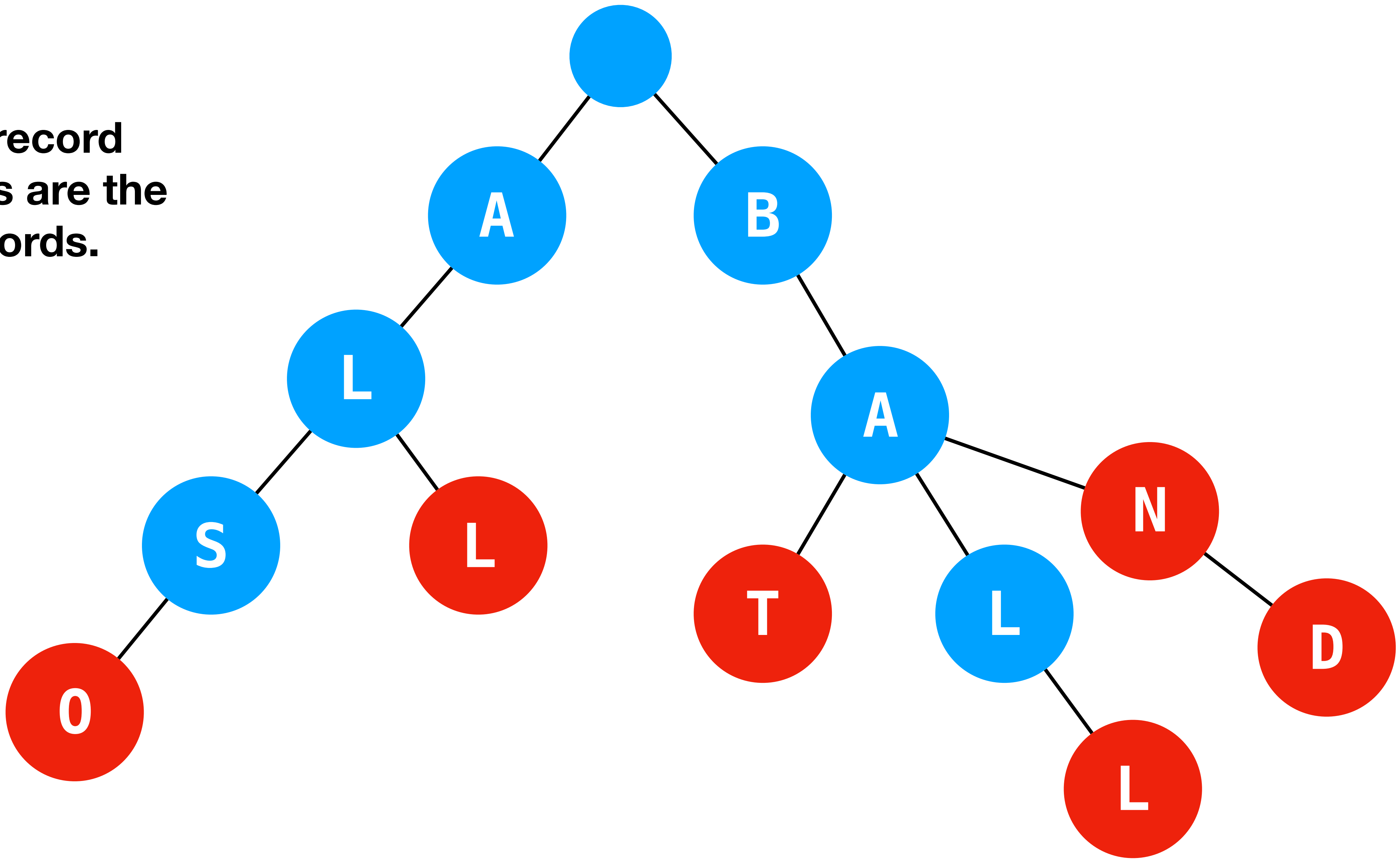


Tries

Add "BAN" to this trie.

...oof

Solution: record which nodes are the end of words.



A
B
L
A
S
L
N
T
L
D
O
L