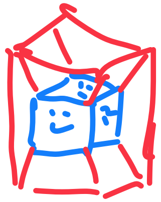
# Data Structures

# K-d Tree

CS 225

Brad Solomon & G Carl Evans

September 29, 2023



Department of Computer Science

# MP_Lists Plagiarism Report

Significant increase in plagiarism **

Still processing all the FAIR cases

Remember course policies!

# MP_Mosaic Extra Credit Extension

Todays lecture will ‘review’ several key concepts

Concepts may be new to some, extra credit is extended

Extra credit deadline: Wednesday

# Learning Objectives

AVL tree

B

Discuss (one) extension beyond BST

Introduce lambda functions in C++

Finish AVL proof and introduce B-Trees

X X

2 weeks Acen lab

# Summary of Balanced BST

claim w/o proof

**AVL Trees**

- Max height: ???? * lg(n)
- Rotations:

Zero rotations on find

One rotation on insert

O(**h**) == O(**lg(n)**) rotations on remove

# Range-based Searches

Balanced BSTs are useful structures for range-based and nearest-neighbor searches.

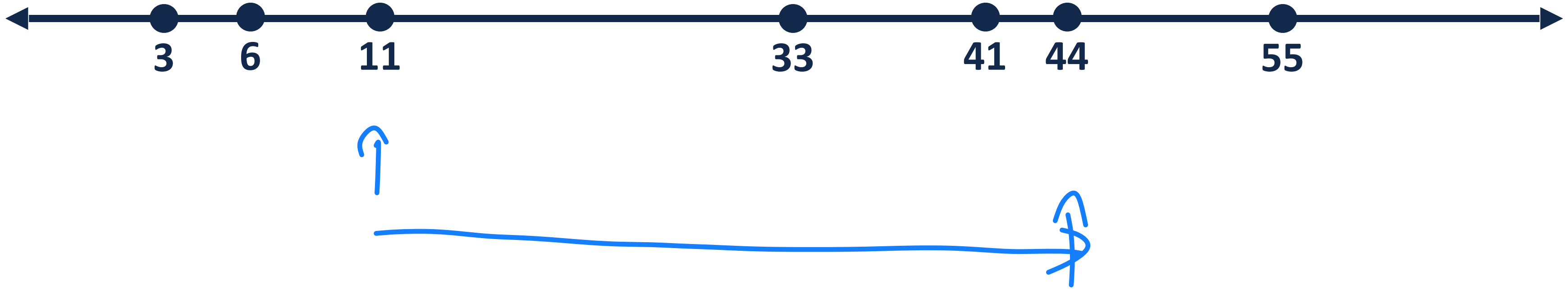
If this is fixed

**Q:** Consider points in 1D: $\mathbf{p = \{p_1, p_2, \ldots, p_n\}}$.

If I can change

…what points fall in [11, 42]?

**Ex:**



# Range-based Searches

iter. begin / iter. end / i++

11 44

**Q:** Consider points in 1D: **$p = \{p_1, p_2, \ldots, p_n\}$**.

exclusive

…what points fall in [~~11~~, 42]?

10

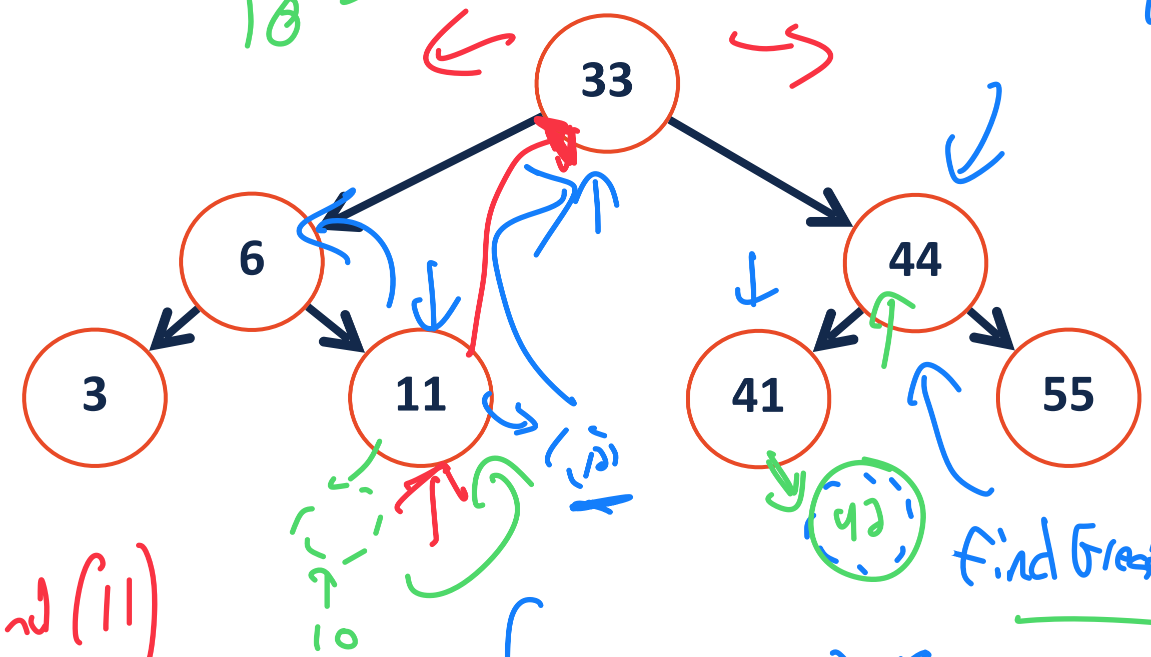
query: 11-42

↳ 11, 33, 41

12, 42

↳ 33, 41

— - -



inOrder Iterator find(11)

findGreater(42)

11 | , 33, 41, ~~44~~

end

# Red-Black Trees in C++

ADT

C++ provides us a balanced BST as part of the standard library:

```
std::map<K, V> map;
```

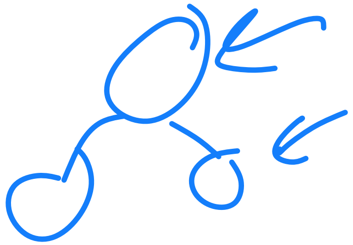
constructor

insert / find

```
V & std::map<K, V>::operator[]( const K & )
```

```
std::map<K, V>::erase( const K & )
```

remove



checker board

# Red-Black Trees in C++

C++ provides us a balanced BST as part of the standard library:

```
iterator std::map<K, V>::lower_bound( const K & );

iterator std::map<K, V>::upper_bound( const K & );
```

↳ Nearest neighbor!

begin

end

for (auto it = p.upper_bound(11); it != p.upper_bound(42); it++){

↳ ≤

print *it

↓

3

X

6

[11, 42)

?:

# Range-based Searches

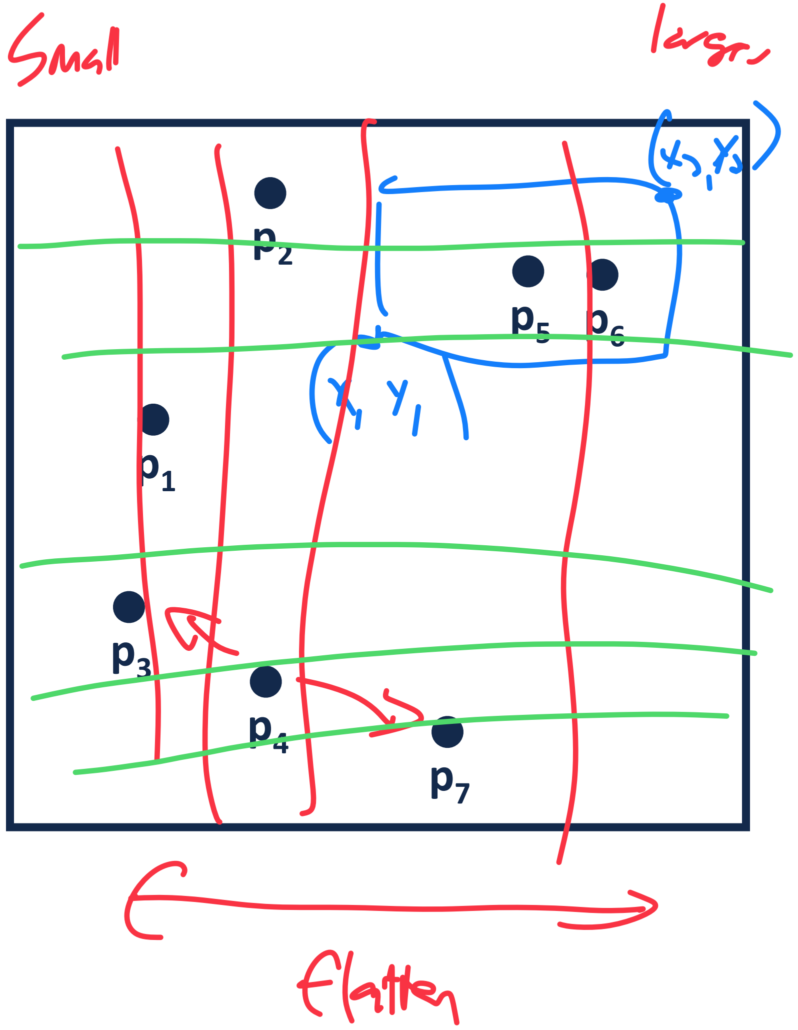
ordered
smaller
larger

Consider points in 2D: **p = {$p_1$, $p_2$, …, $p_n$}.**

**Q:** What points are in the rectangle:
**[ ($x_1$, $y_1$), ($x_2$, $y_2$) ]?**

**Q:** What is the nearest point to **($x_1$, $y_1$)**?

This is hard!

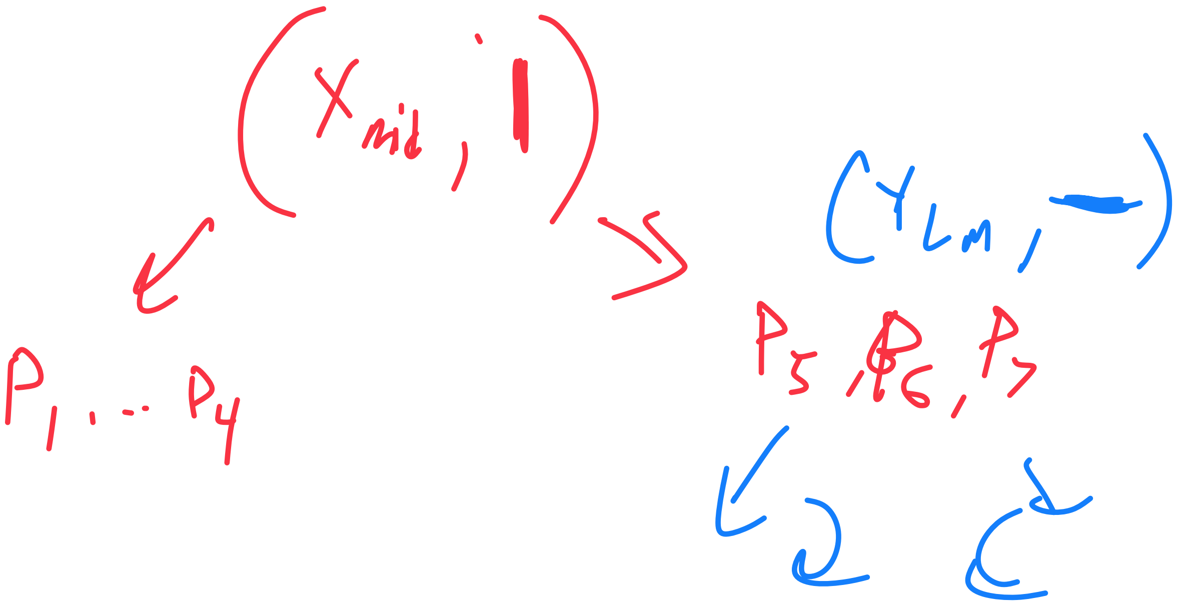


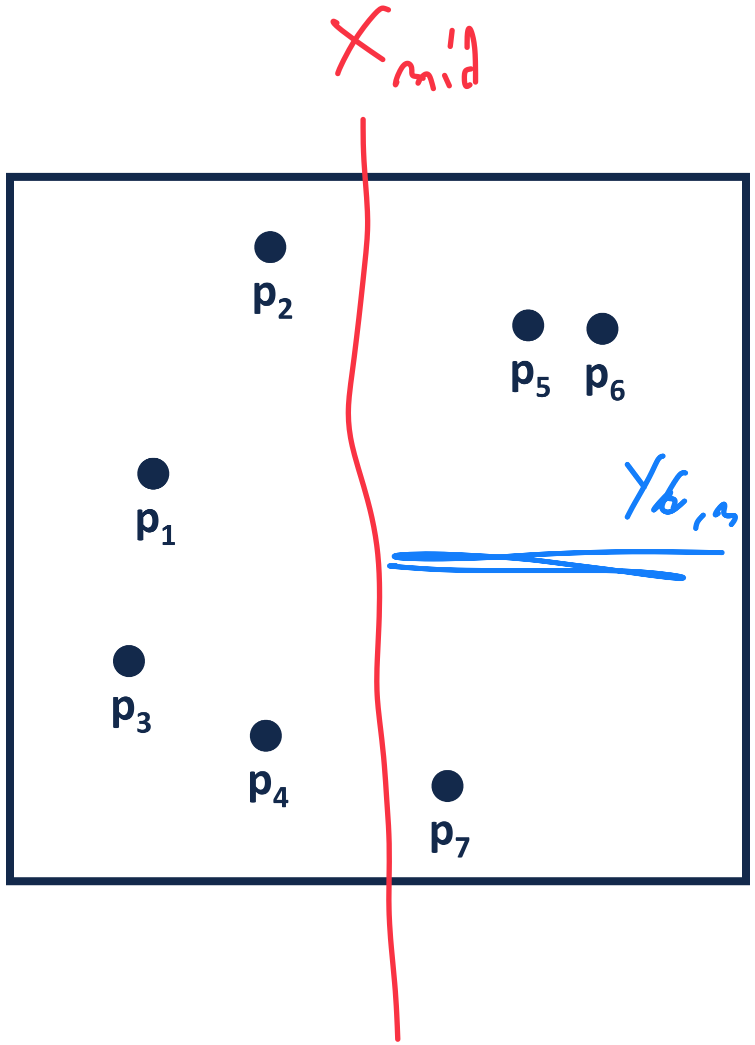
# Range-based Searches

Consider points in 2D: **p = {p$_1$, p$_2$, …, p$_n$}.**

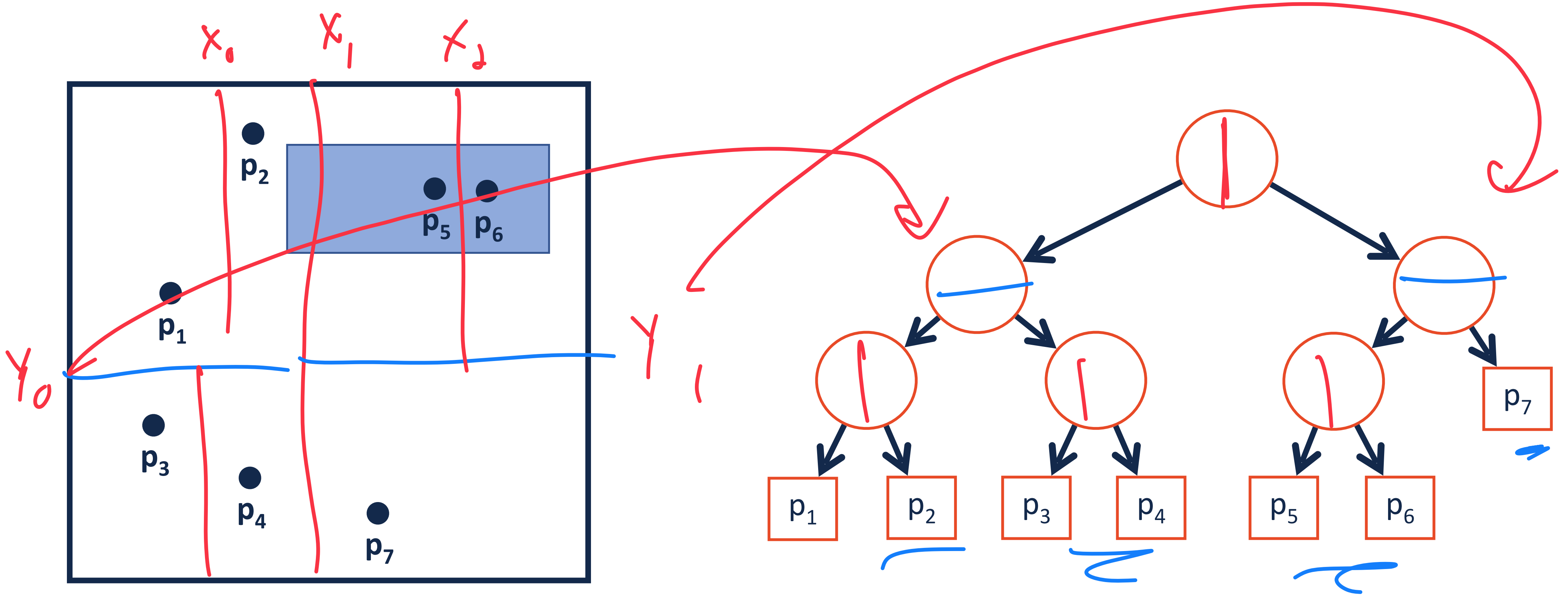
**Tree construction:**

Build a tree on changing dimensions

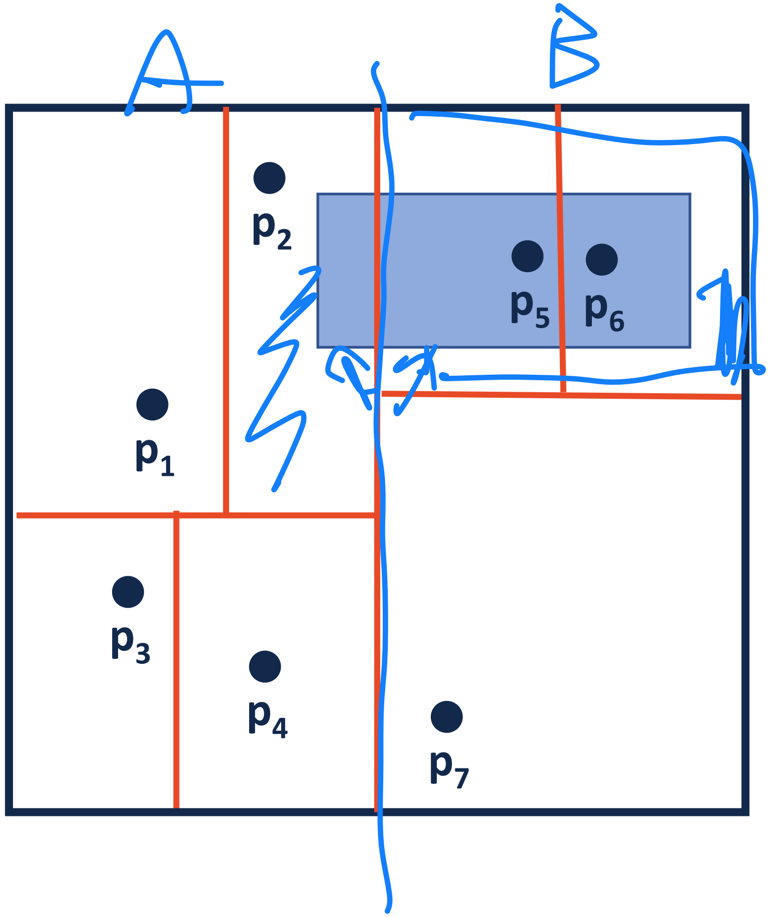


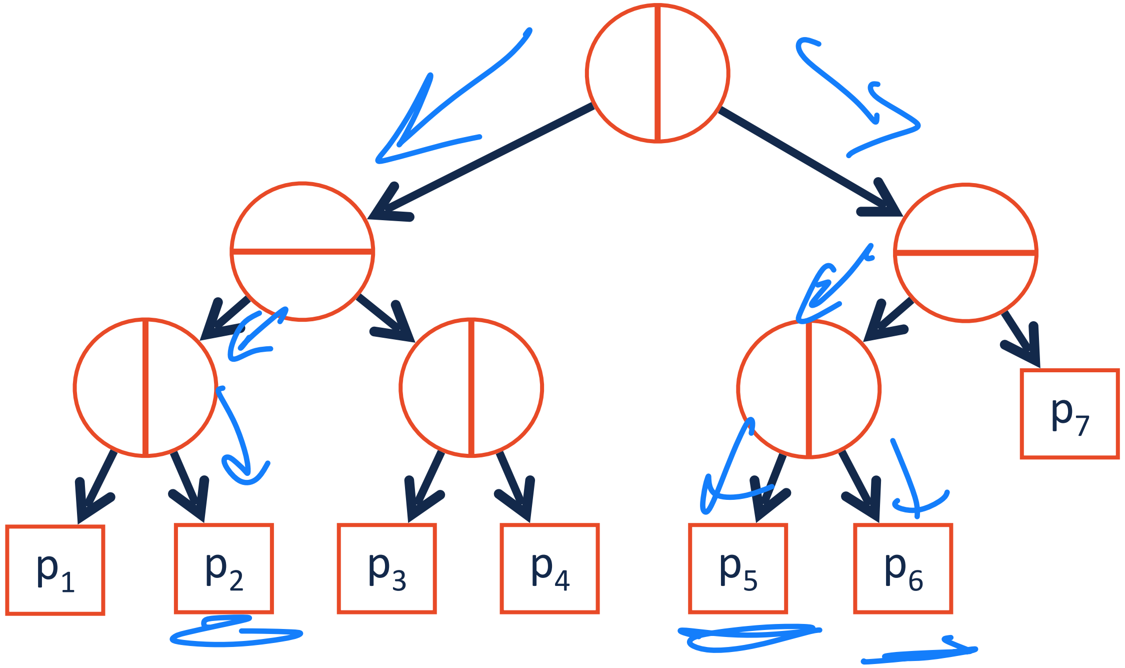


# Range-based Searches



# Range-based Searches





# Nearest Neighbor: k-d tree

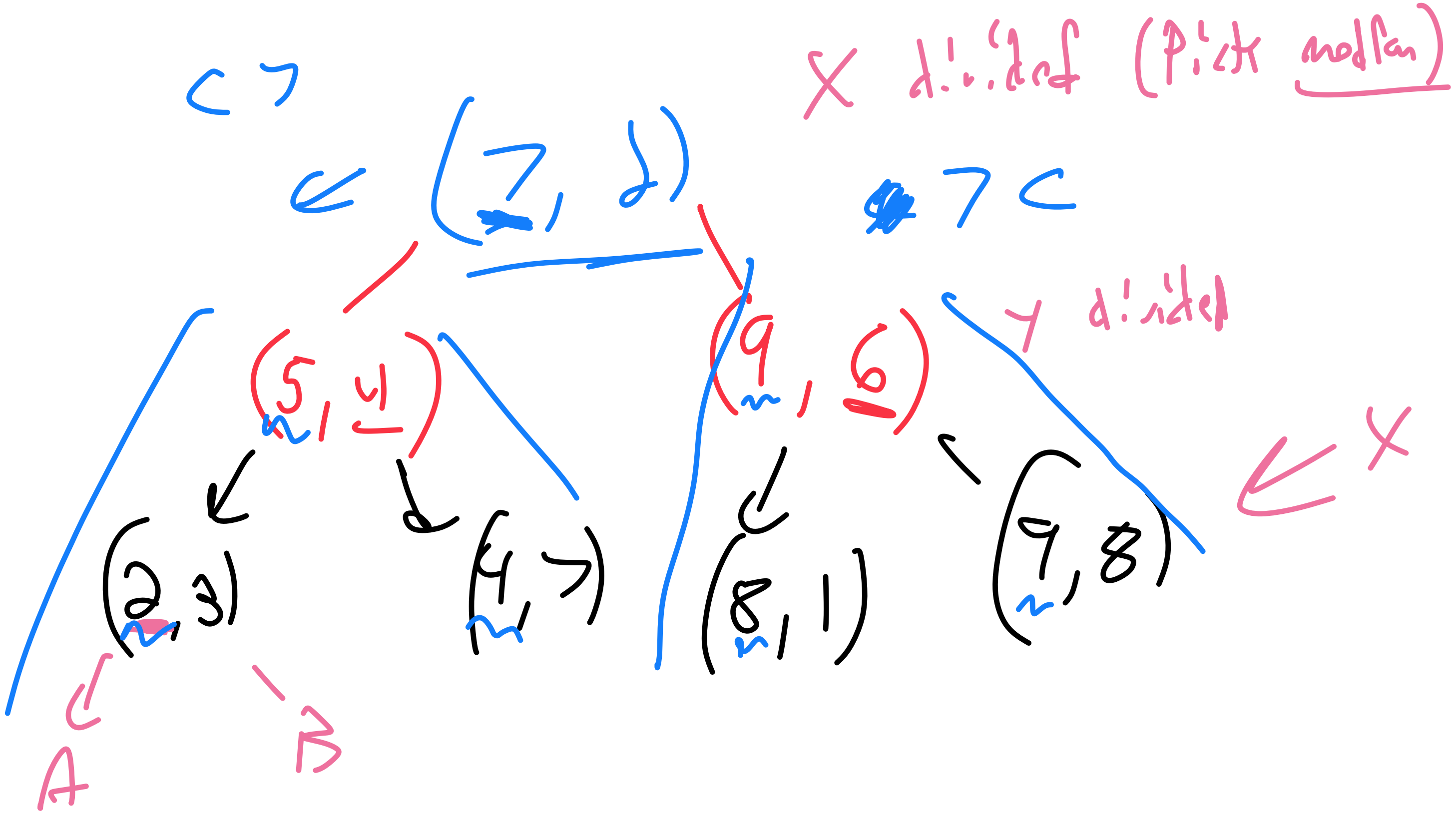
(x, y, z)

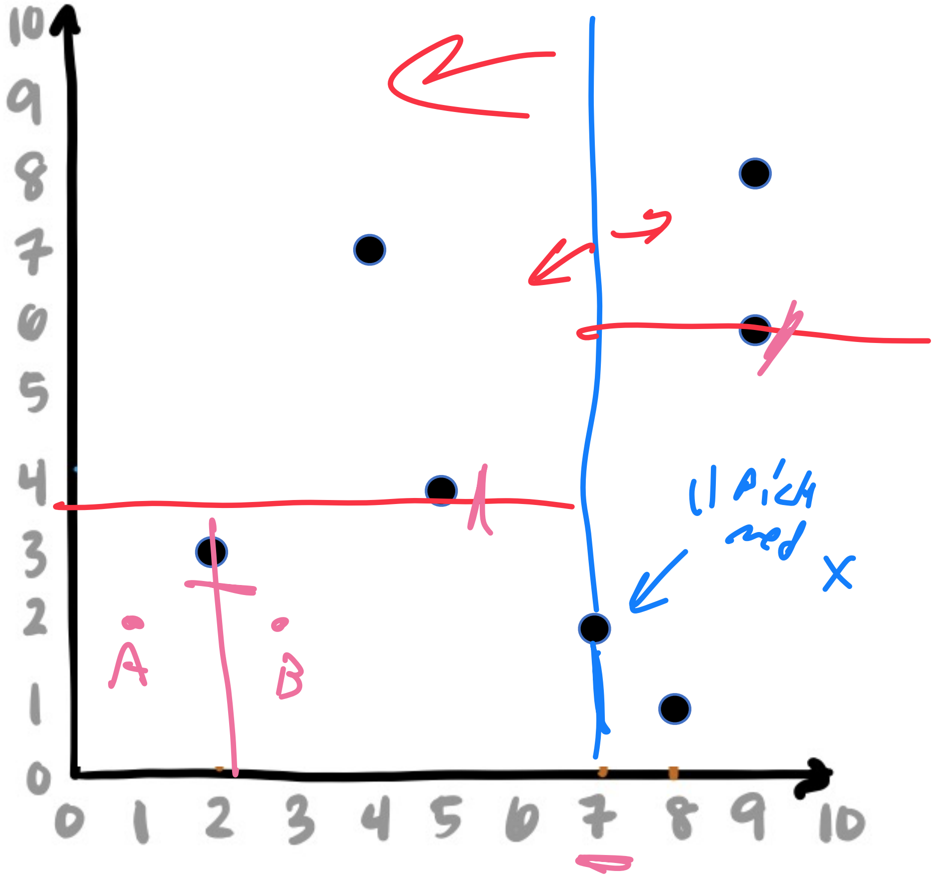
(x, y)

A **k-d tree** is similar but splits on points:

(7,2), (5,4), (9,6), (4,7), (2,3), (8,1), (9,8)

start w/ x-dim

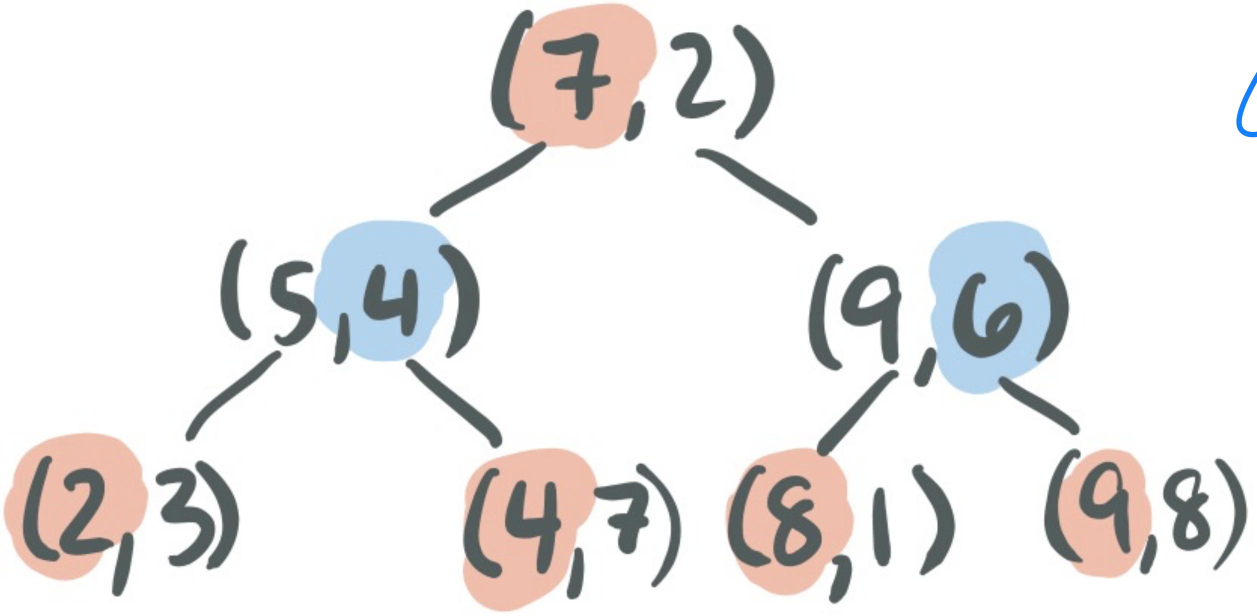


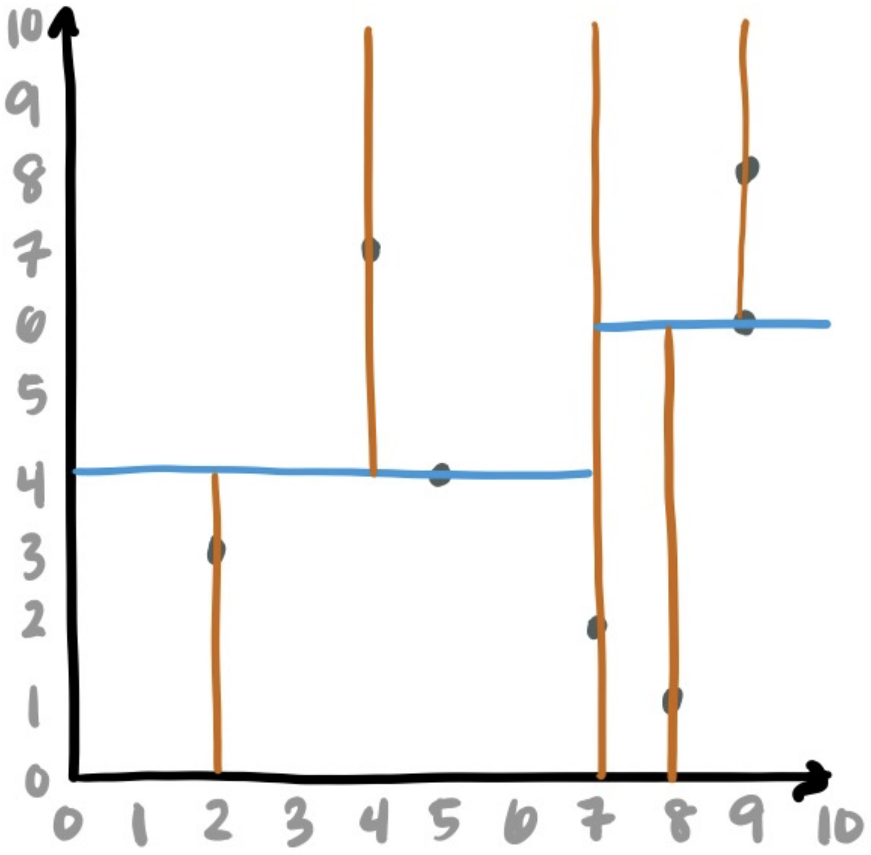


# Nearest Neighbor: k-d tree

Split all dimensions

↳ Pick median value in each dimension





# Nearest Neighbor: k-d tree

This is hard!

This construction seems easy conceptually but…

1. Review, understand, and use **quickselect**

standard!

2. Review, understand, and use **lambda functions**

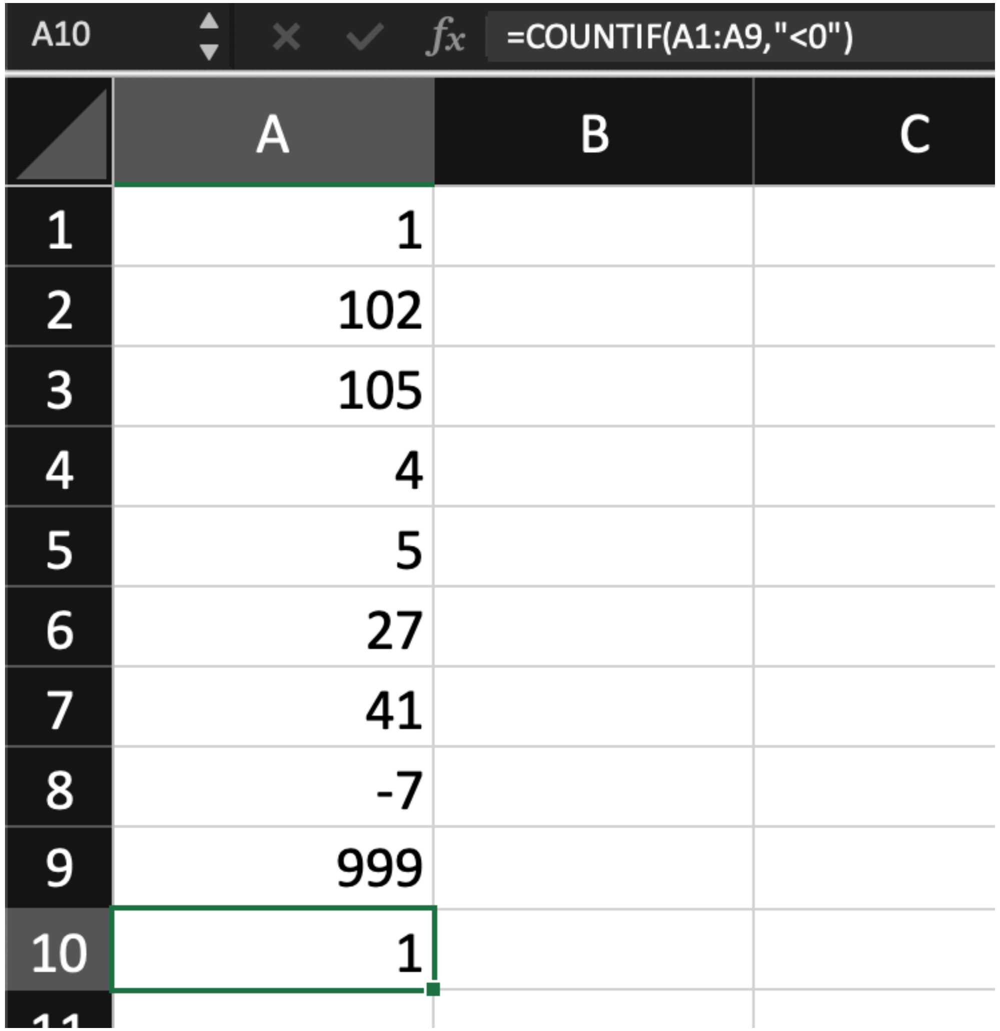
# Functions as arguments

Consider the function from Excel
COUNTIF(*range*, *criteria*)

"Iterator" start is A1

end is A9

comp | pred is "<0"



A10 =COUNTIF(A1:A9,"<0")

|  | A | B | C |
|---|---|---|---|
| 1 | 1 |  |  |
| 2 | 102 |  |  |
| 3 | 105 |  |  |
| 4 | 4 |  |  |
| 5 | 5 |  |  |
| 6 | 27 |  |  |
| 7 | 41 |  |  |
| 8 | -7 |  |  |
| 9 | 999 |  |  |
| 10 | 1 |  |  |

# Functions as arguments

predicate

**Countif.hpp**

```
template <typename Iter, typename Pred>
int Countif(Iter begin, Iter end, Pred pred) {
  int count = 0;
  auto cur = begin;

  while(cur != end) {
    if(pred(*cur))
      ++count;
    ++cur;
  }

  return count;
}
```

select( begin, end, itr, k, cmp)

called here

is big (from 2 slides later)

pred is a function!

this is run (3 slide later)

↳ functions have memory addresses!

↳ Treat like variable

# Lambda Functions in C++

**main.cpp**

```
bool isNegative(int num) { return (num < 0); }

class IsNegative {
public:
    bool operator() (int num) { return (num < 0); }
};

int main() {
  std::vector<int> numbers = {1, 102, 105, 4, 5, 27, 41, -7, 999};

  auto isnegl = [](int num) { return (num < 0); };
  auto isnegfp = isNegative;
  auto isnegfunctor = IsNegative();

  cout << "There are " << Countif(numbers.begin(), numbers.end(), ________)
    << " negative numbers" << std::endl;

```

Normal function

class method

&, *

Cmp?

# Lambda Functions in C++

Build on args

most useful

[ capture ]( arg list input ){ function body code }

??

we can use value as variable

when function is defined it captures a value

the value of captured variables is fixed when defined

It does not see the actual variable

# Lambda Functions in C++

main.cpp

← 10;

10;

```
29   int big;
30
31
32   std::cout << "How big is big? ";
33   std::cin >> big;
34
35   auto isbig = [big](int num) { return (num >= big); };
36
37   std::cout << "There are " << Countif(numbers.begin(), numbers.end(), isbig)
38     << " big numbers" << std::endl;
}
```

If ~~[illegible]~~ , isbig is defined here

changing big doesnt matter

change big doesnt matter!

3

3

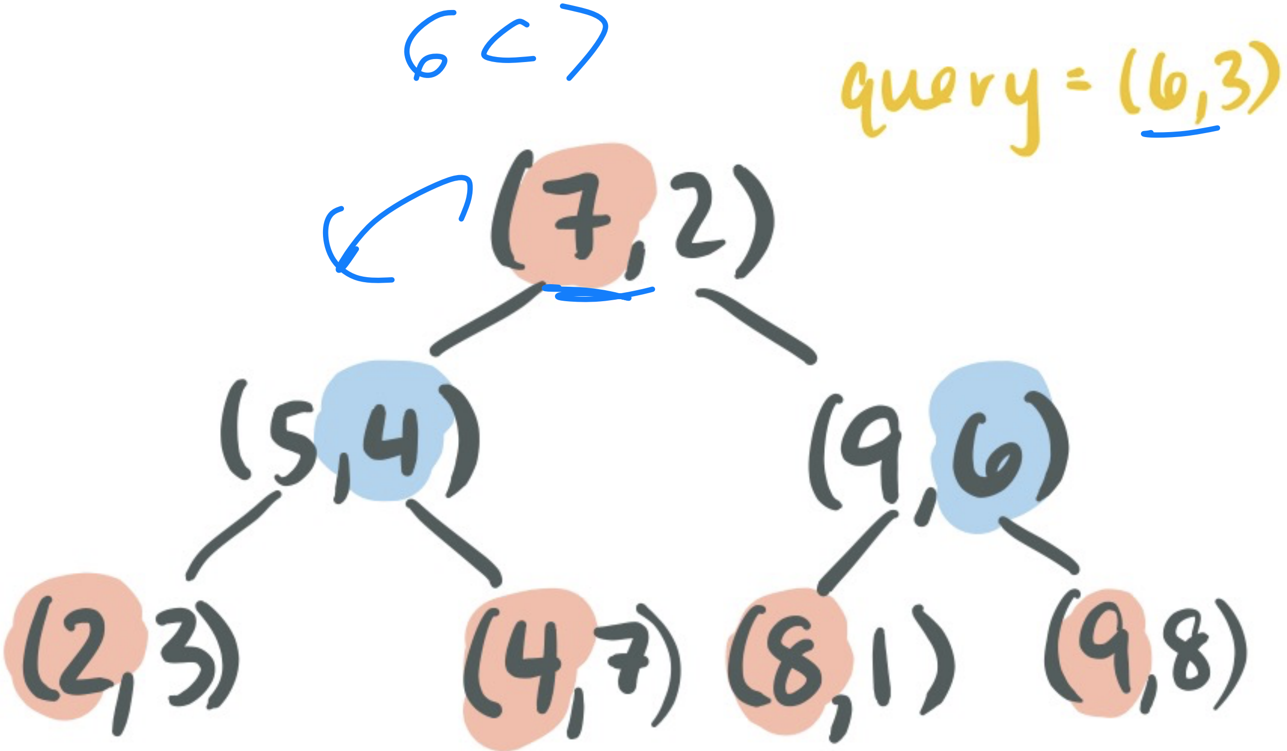
Value of big @ 35 is value always in isbig

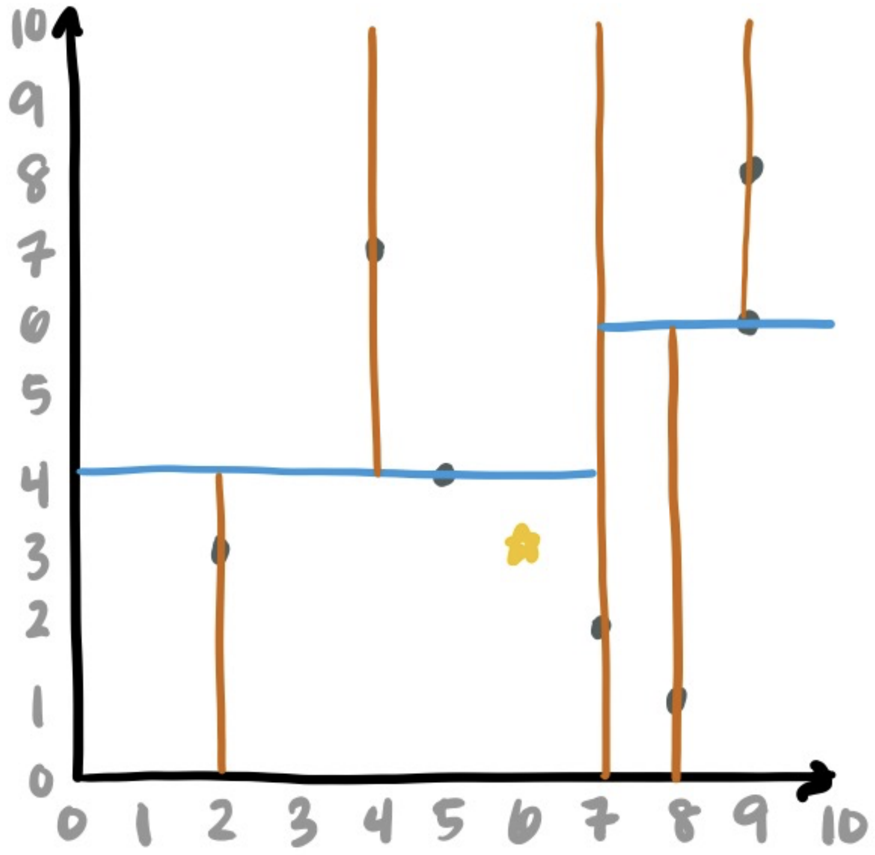
big is 10 always

# Nearest Neighbor: k-d tree

change dim every level

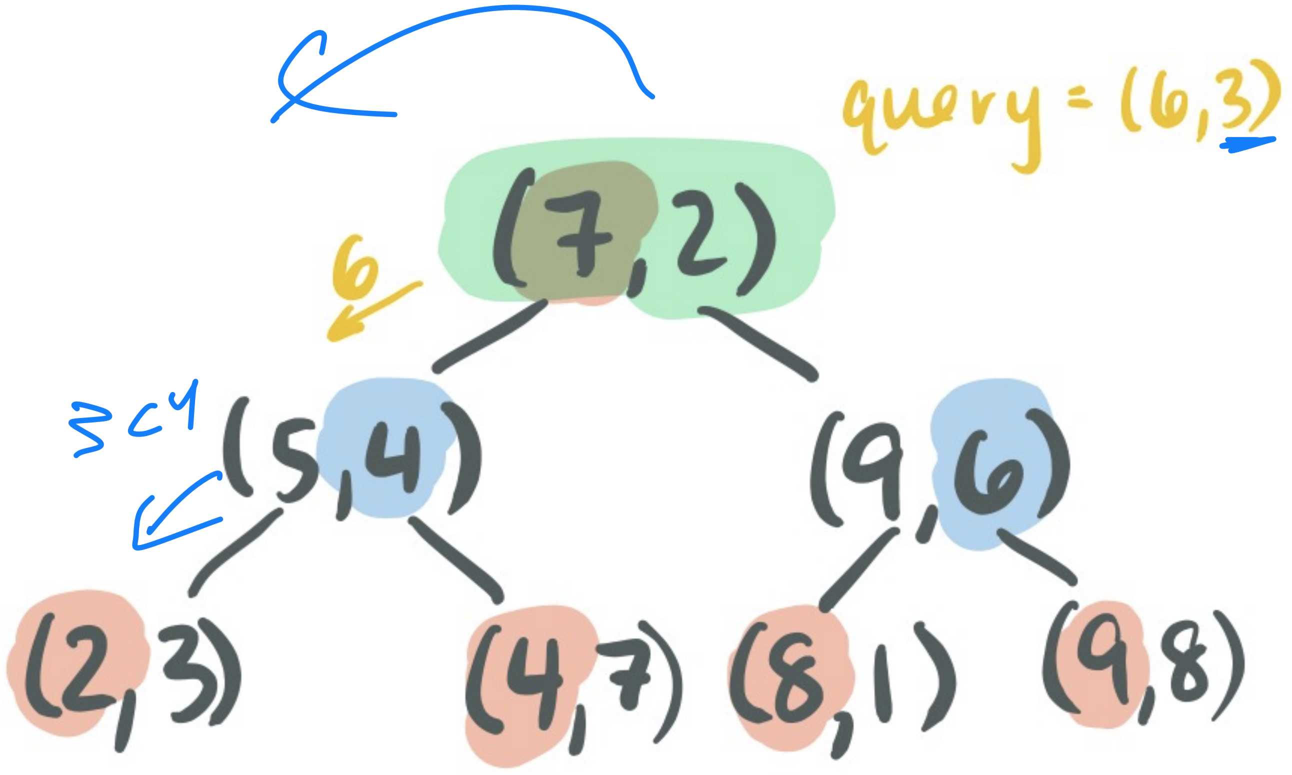
When querying a k-d tree, it acts like a BST* at first…

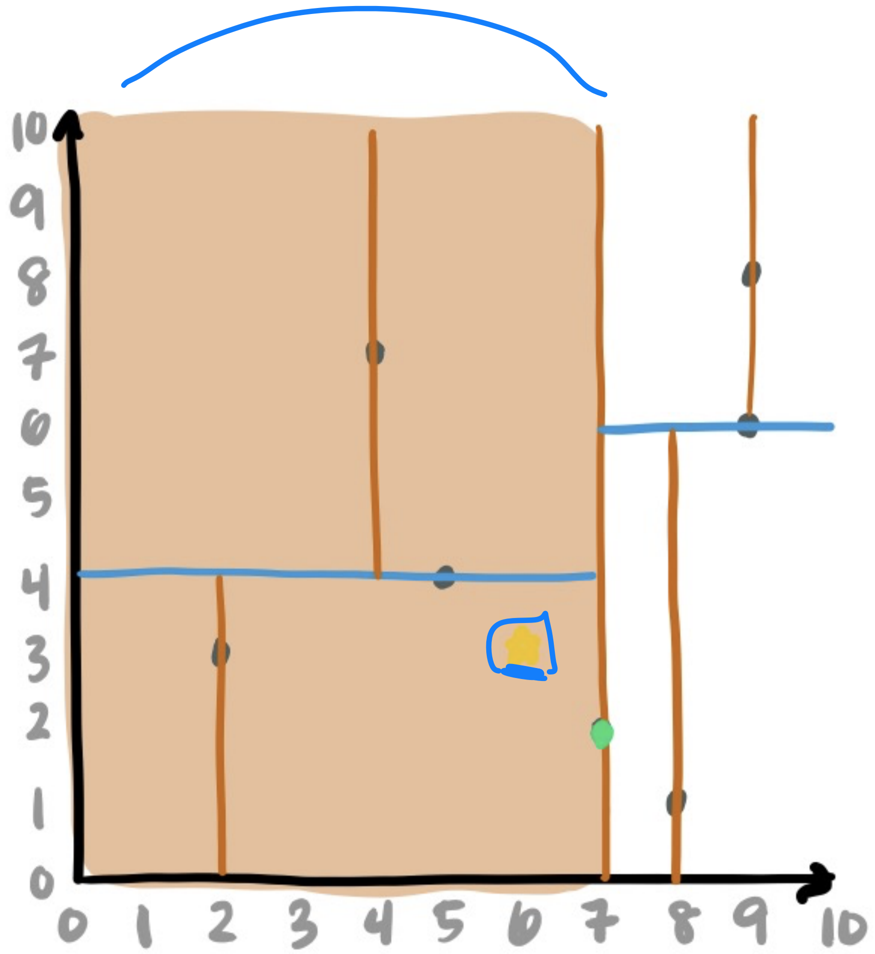




# Nearest Neighbor: k-d tree

When querying a k-d tree, it acts like a BST* at first…

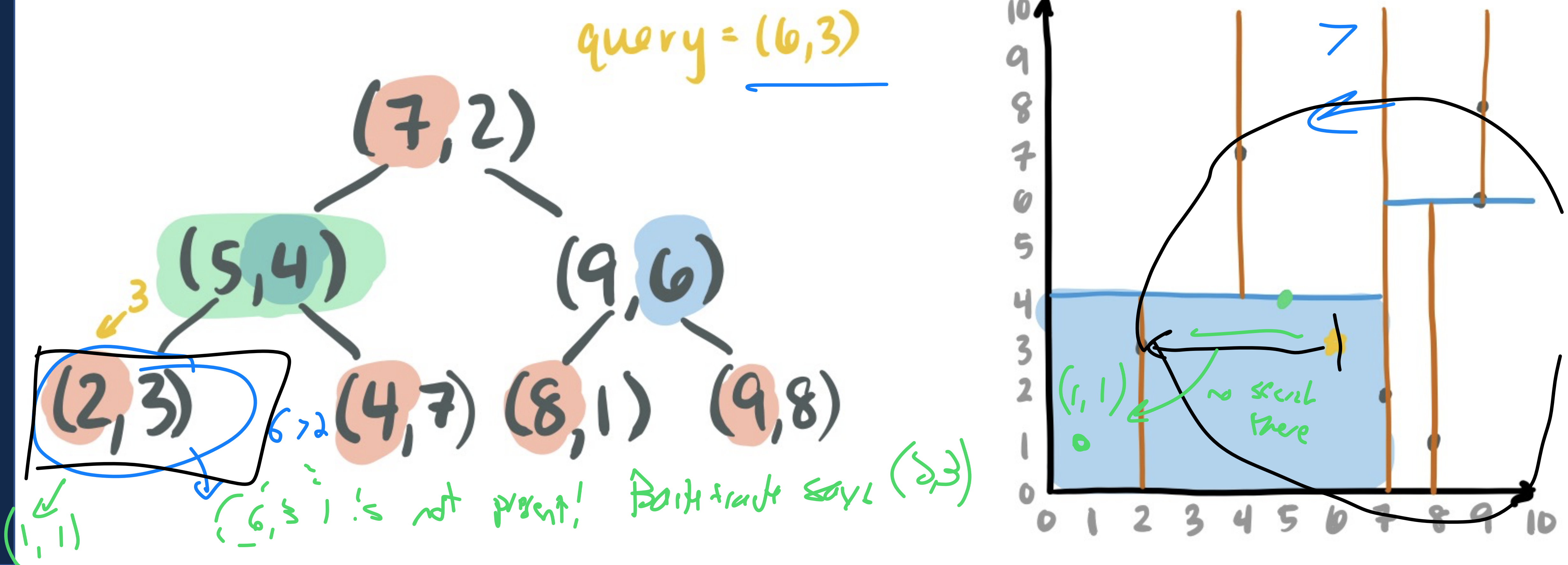




# Nearest Neighbor: k-d tree

When querying a k-d tree, it acts like a BST* at first…

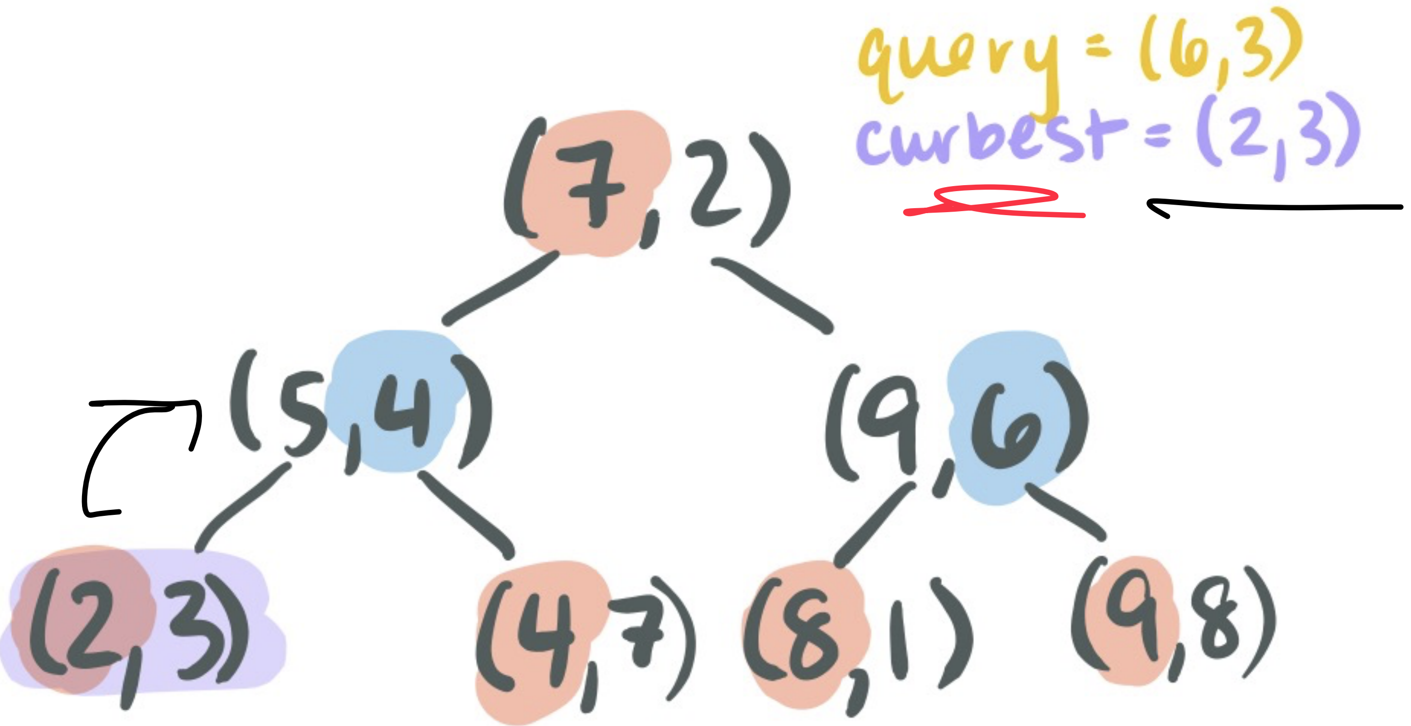
↳ But if not exact need to find nearest neighbor

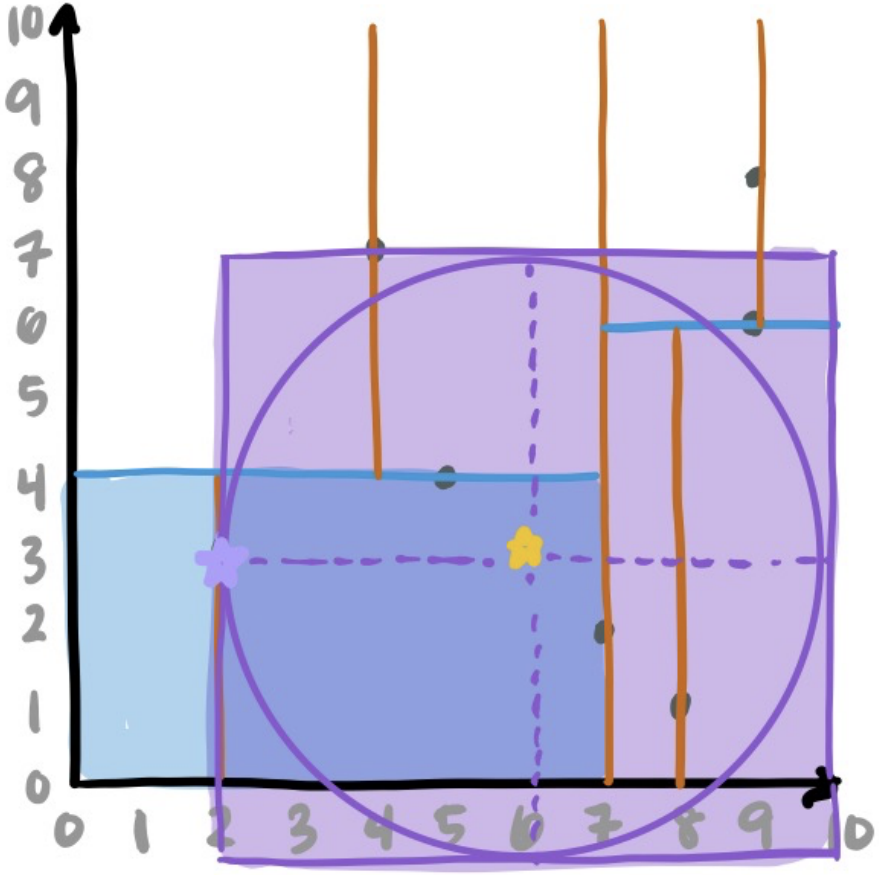


# Nearest Neighbor: k-d tree

When querying a k-d tree, it acts like a BST* at first…

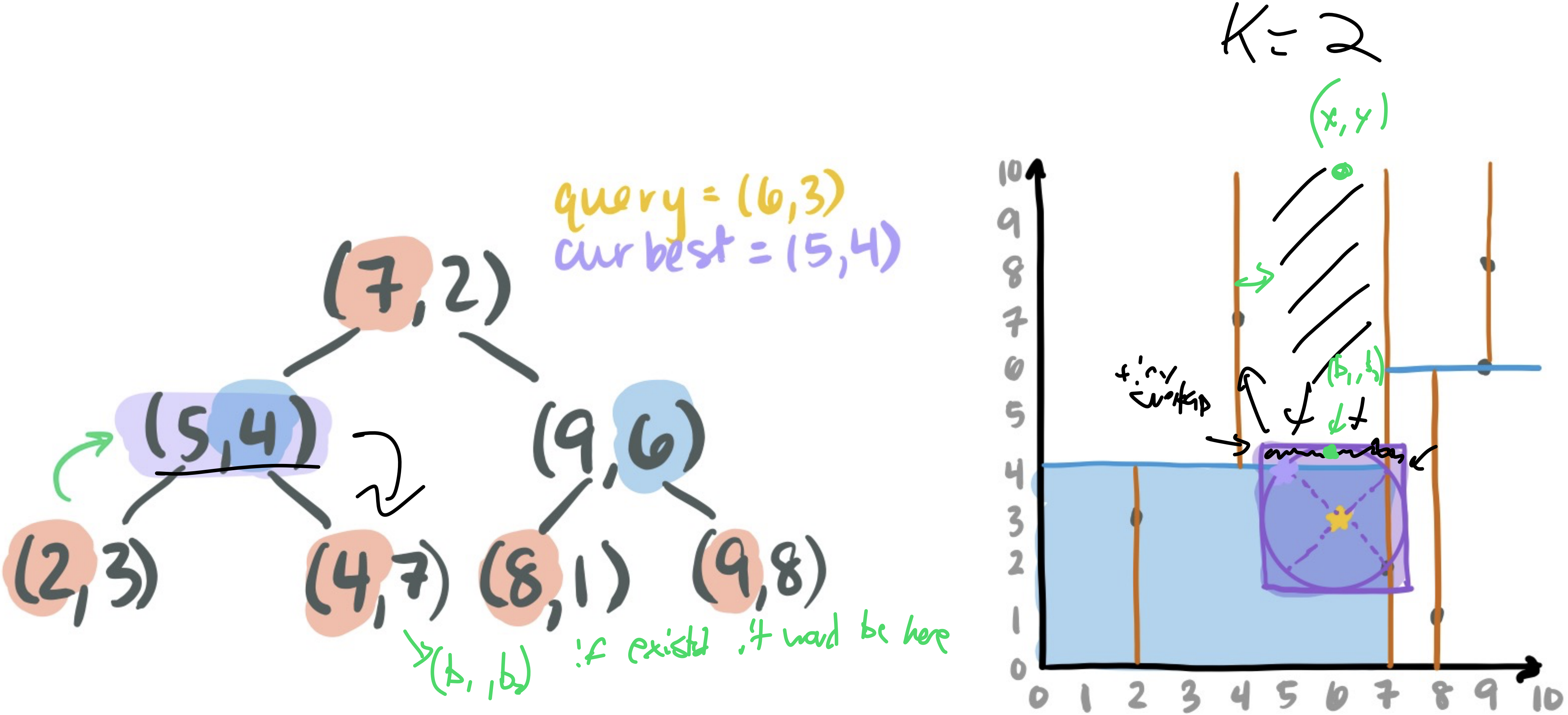
checked first 2,3 vs 5,4 (against 6,3)



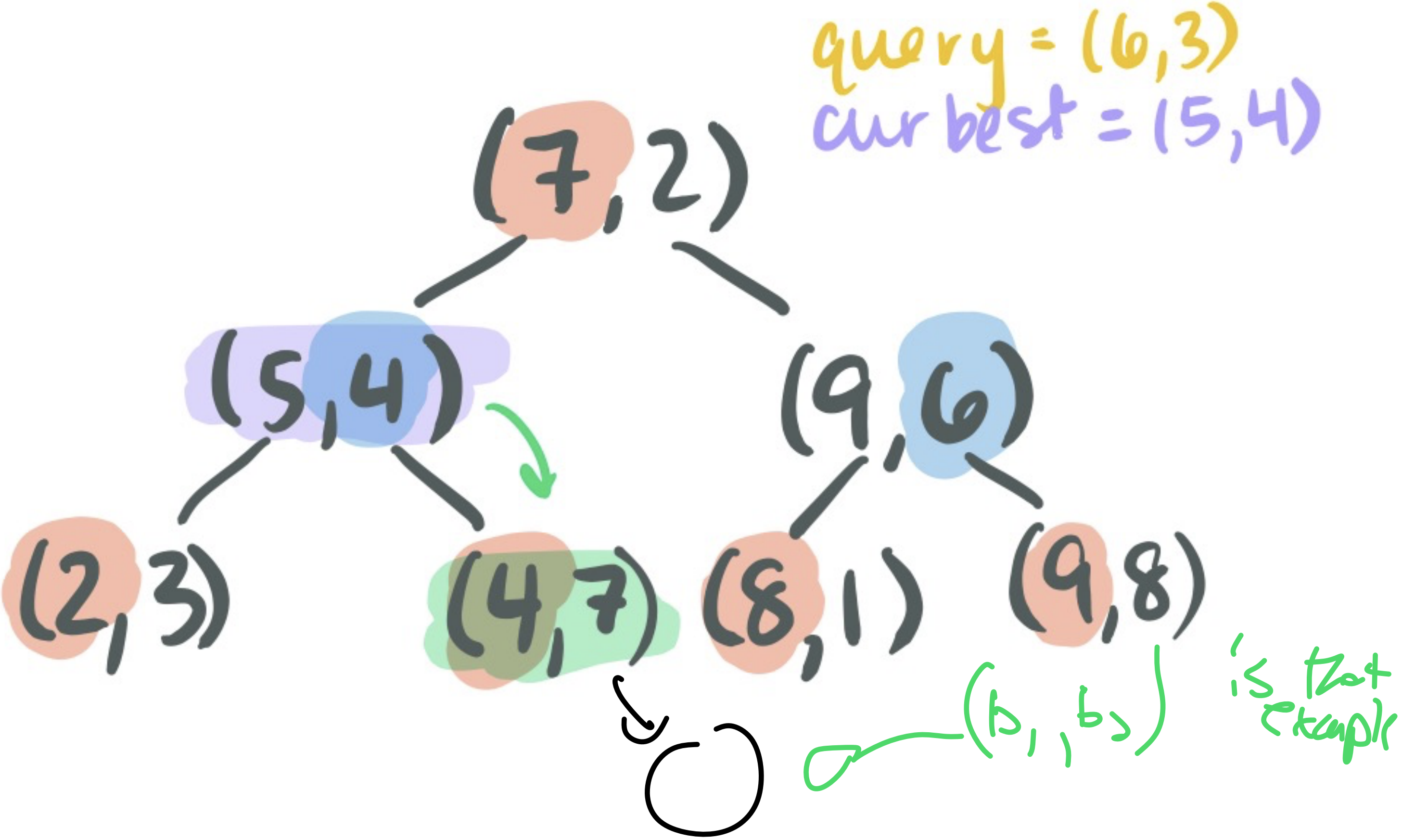


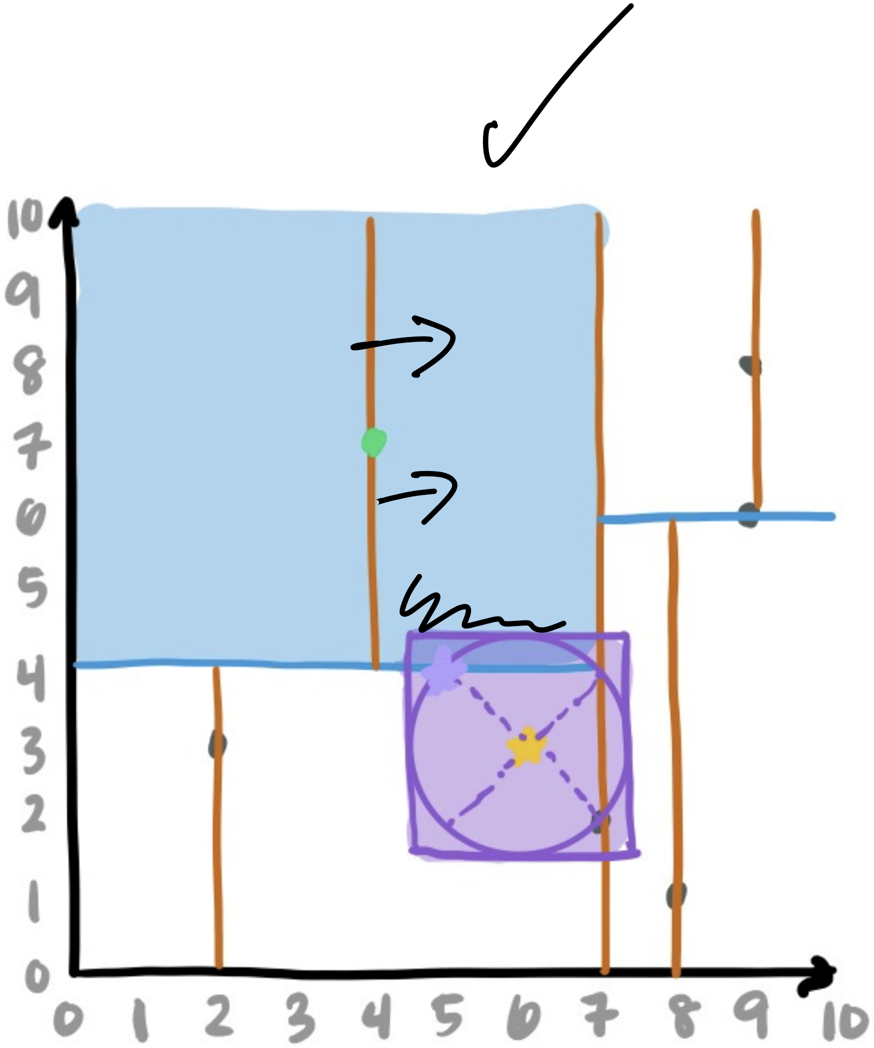
# Nearest Neighbor: k-d tree

**Backtracking:** start recursing backwards -- store "best" possibility as you trace back



# Nearest Neighbor: k-d tree

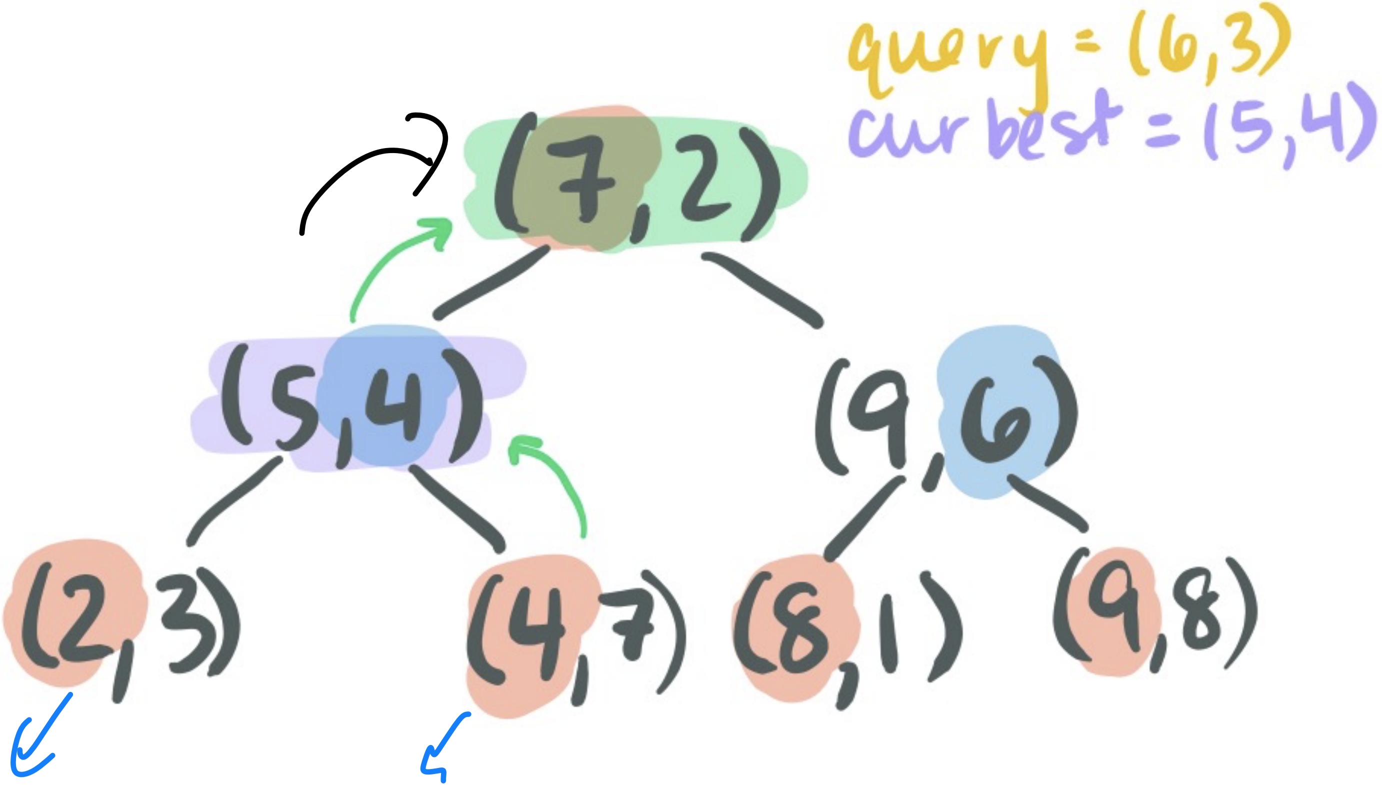


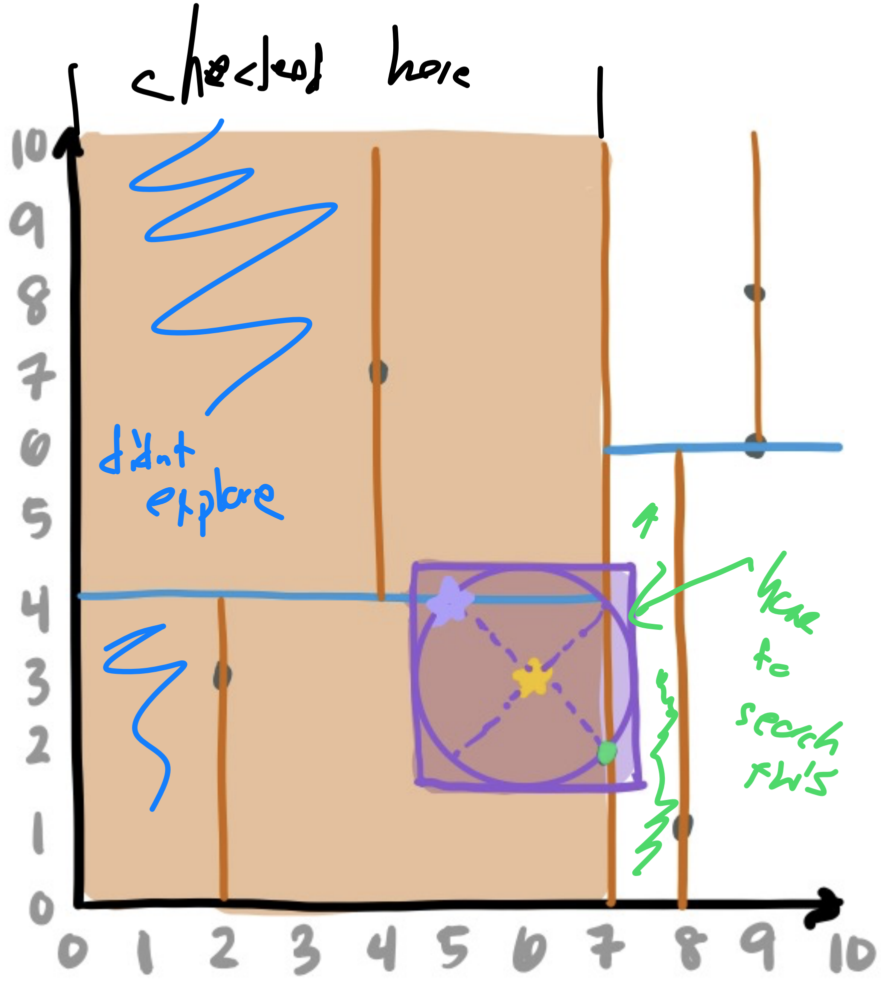


# Nearest Neighbor: k-d tree

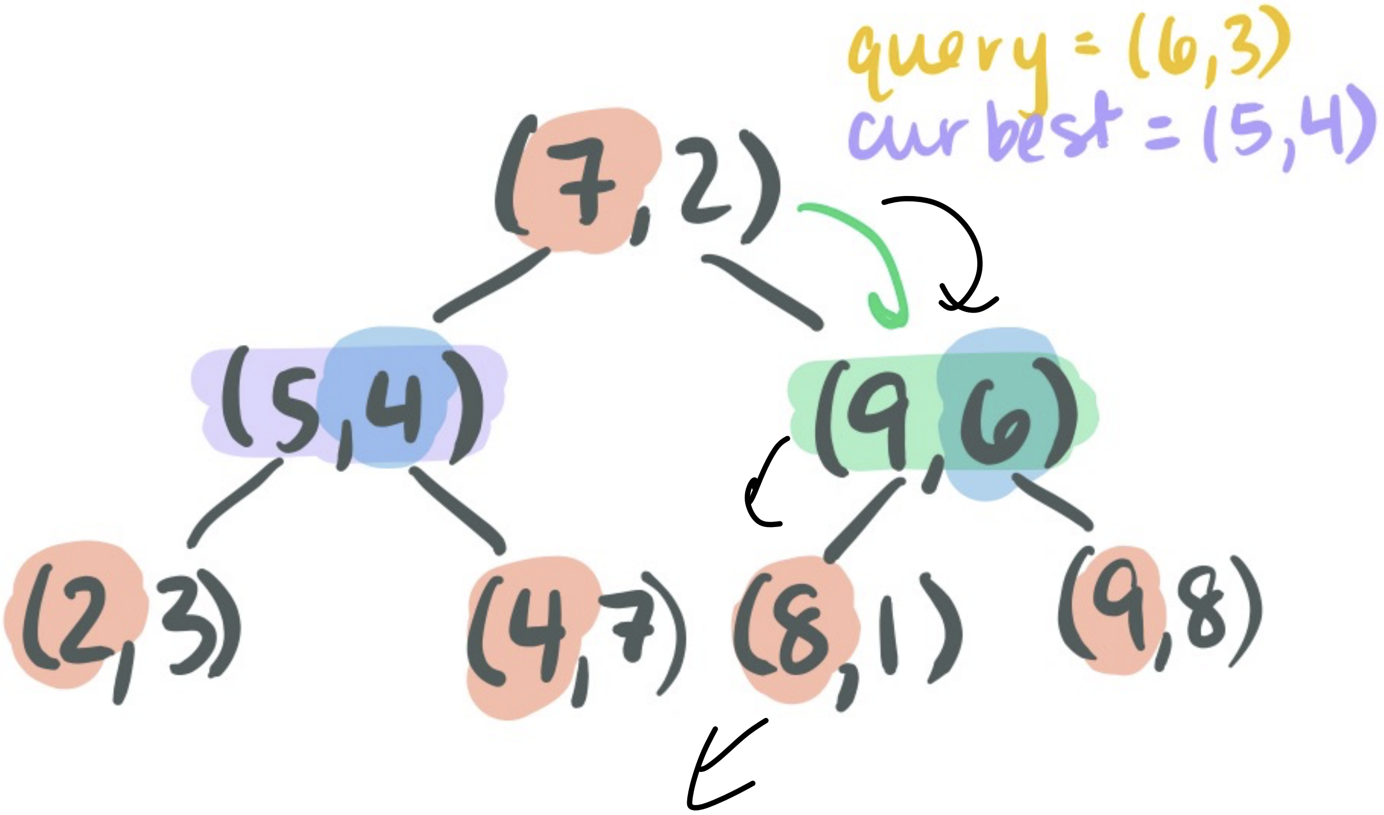
On ties, use `smallerDimVal` to determine which point remains `curBest`

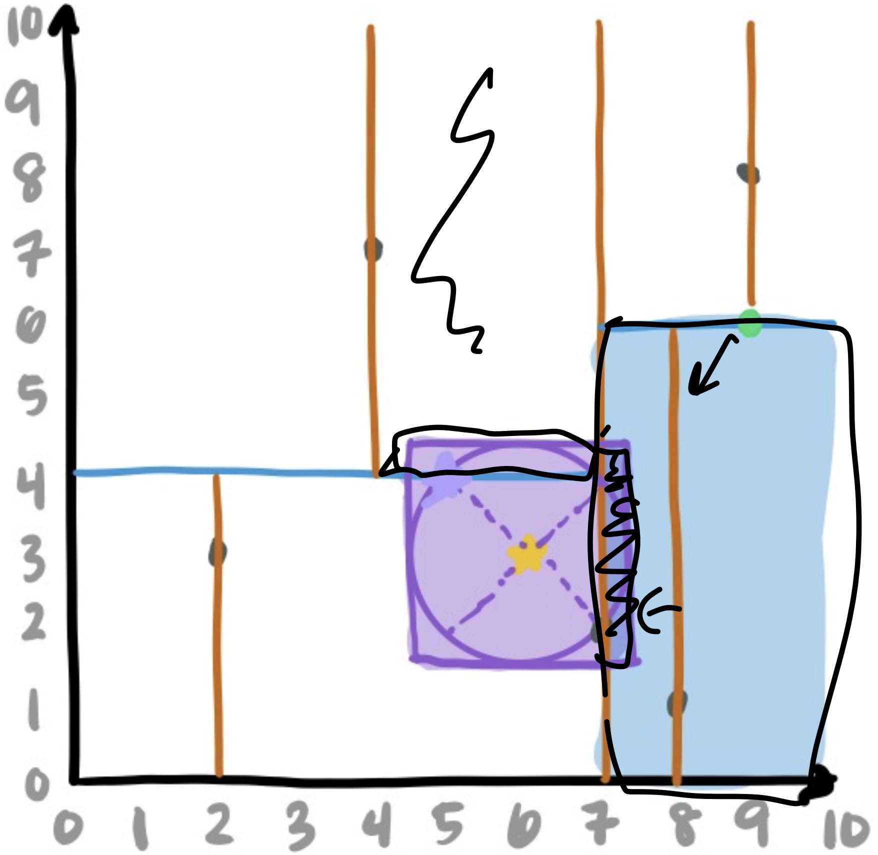
7,2 and 5,4 are equally away





# Nearest Neighbor: k-d tree

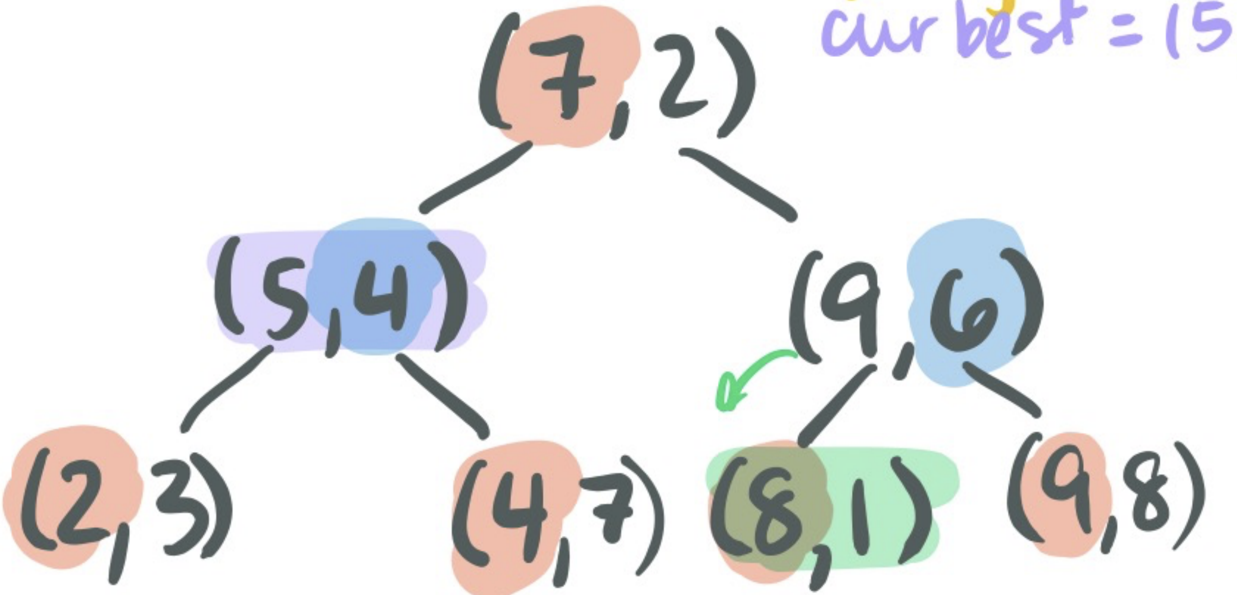


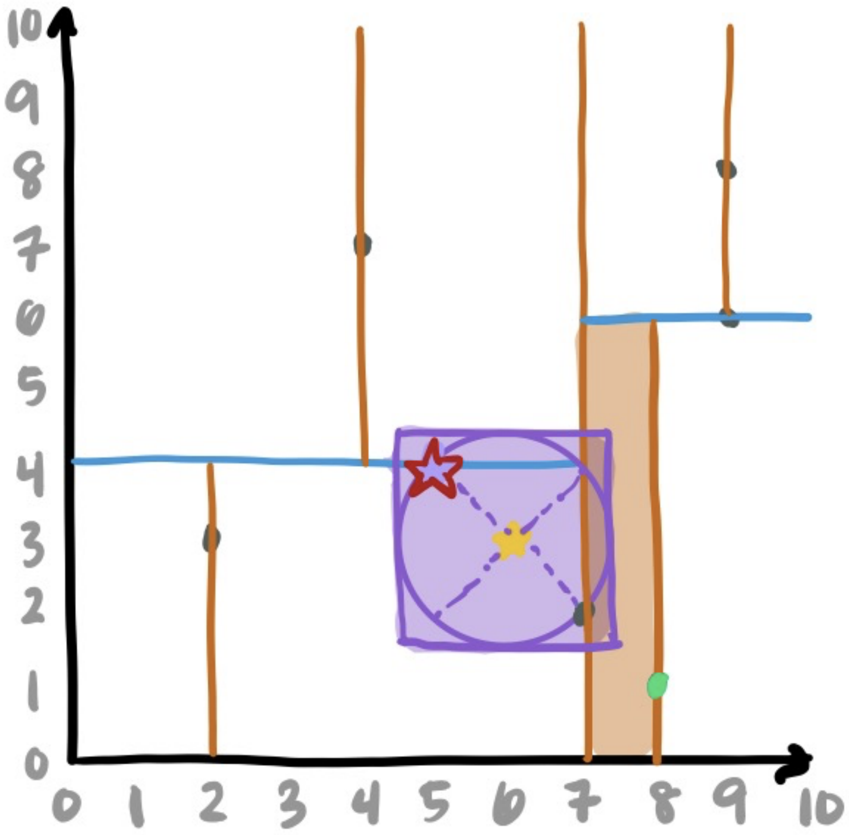


# Nearest Neighbor: k-d tree

query = (6,3)

cur best = (5,4)





BEST: (5,4)

# Nearest Neighbor: k-d tree

**Final tips:**

The mp_mosaic writeup is long. **READ IT**

The suggestions in the writeup should be followed carefully