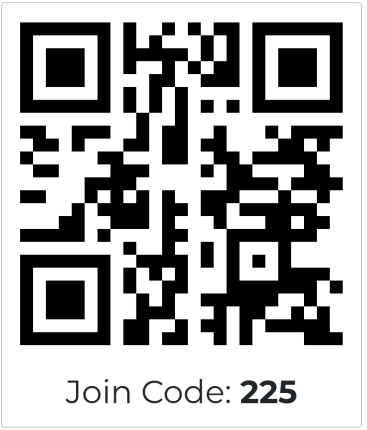
Announcements

1. MP Lists due Today!
2. Exam 2 this week! (M-W)
3. Exam 1 Grades Released
    a. Exam Regrade Form Online
4. Extra Credit Survey Form (2pts, 70% of class must respond)
5. MP Mosaics will be released soon

Extra Credit Survey!



Warm-Up Question: What is the worst case runtime of Quick Select?

KD Trees

Learning Objectives

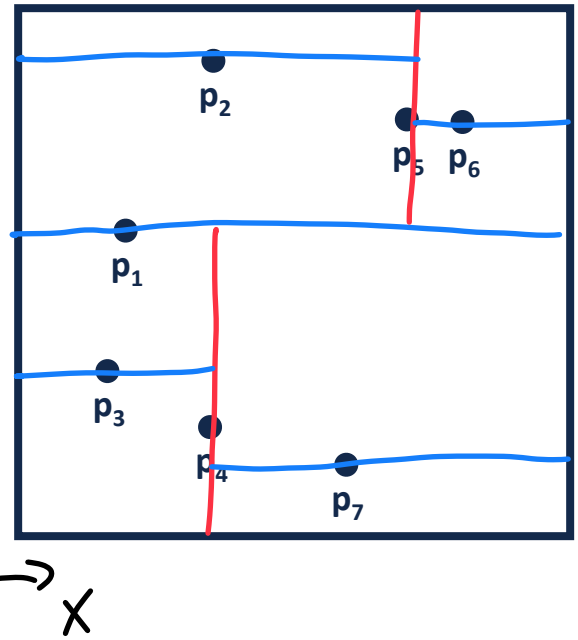
1. Understand the worst and average case runtime of Quick Select
2. Understand nearest-neighbor search with a KD-Tree
3. Define and Implement Lambda Functions

2D Range Based Searches

Consider points in 2D: $p = \{p_1, p_2, \ldots, p_n\}$

Tree Construction: Quick Select – $O(n)$

1. Find median point along a dimension and partition nodes
2. Go to next dimension
3. Recursively build left subtree
4. Recursively build right subtree



What is the runtime of building my K-D tree?

$O(n \log n)$

Quick Select Algorithm

Partitions elements about the median

Smaller Values are on the left of the median

Larger Values are on the right of the median

Faster than sorting O(n log n)

Ex. Input: [11, 6, 44, 41, 33, 57, 2]

Output: [11, 6, 2, 33, 41, 57, 44]

<33 >33

Worst Case Analysis

Ex. Input: [11, 6, 44, 41, 33, 57, 2]

Worst case: pivot is either max or min value

$$\text{Total Work} = n + (n-1) + (n-2) + \cdots + 1$$

$$= \sum_{i=0}^{n} i$$

$$= \frac{n(n+1)}{2} = O(n^2)$$

$O(n) \leftarrow E(T(n)) = cn + \frac{1}{n}\sum_{k=0}^{n-1} E[T(\max(k, n-k-1))]$

Average Case Analysis

Ex. Input: [11, 6, 44, 41, 33, 57, 2]

$$E[T(n)] = cn + \frac{1}{n}\sum_{k=0}^{n-1} E(T(\sim))$$

- cn: running quick select
- $\frac{1}{n}\sum_{k=0}^{n-1}$: averaging over any k selected
- $E(T(\sim))$: work for subproblem

K - selected pivot index
m - median index
n - # of elements in array
E - expected value
T - total amount of work done

pivot = 6

Ex. [2, 6, 11, 44, 41, 33, 57]
0 1 2 3 4 5 6
k=1

Ex. pivot = 44
[11, 6, 41, 33, 2, 44, 57]

$n-k-1$

$\max(k, n-k-1)$

k

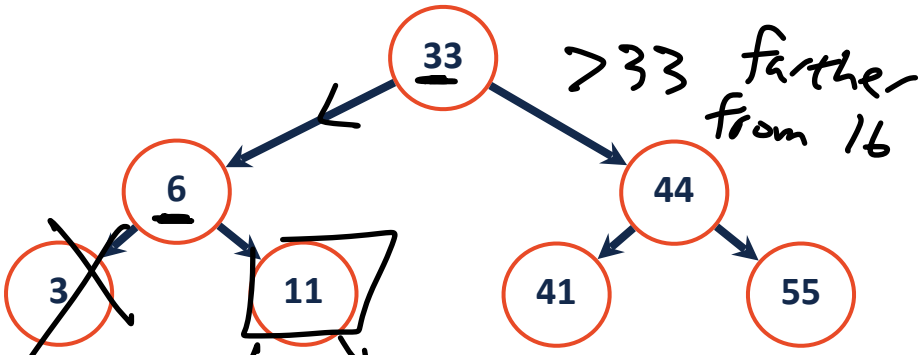
Closest Element

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

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

query point

…what point is closest to 16?



Ex:

closest_point = ~~33~~ 6 11

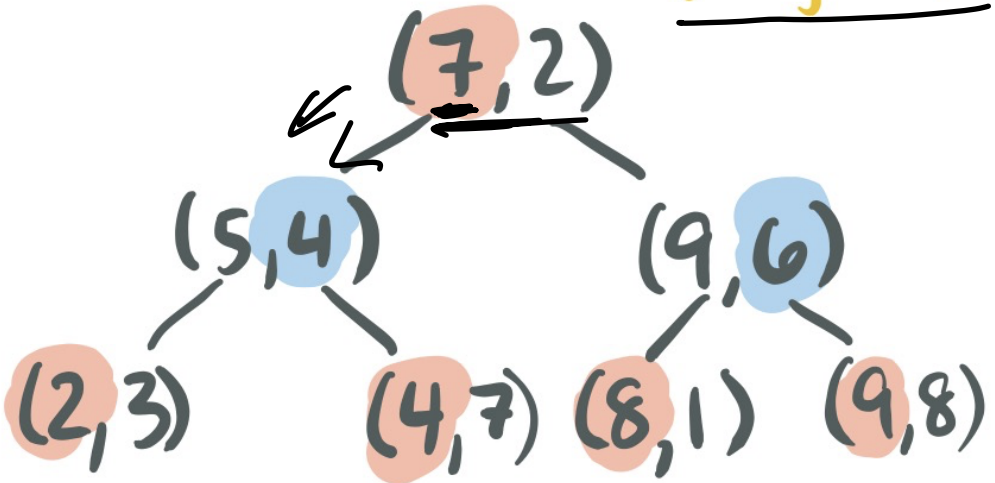
closest_distance = ~~17~~ 10 5

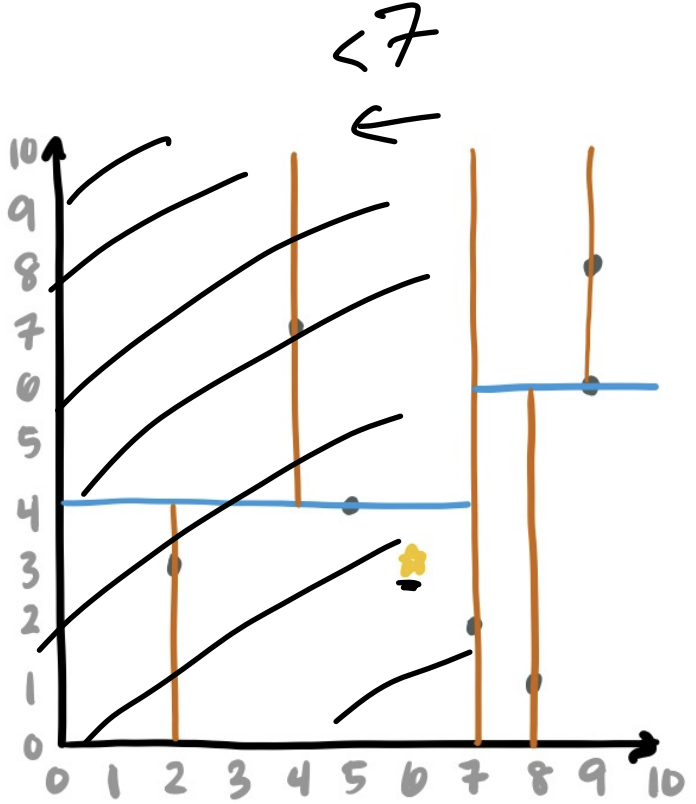
Nearest Neighbor KD Tree

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

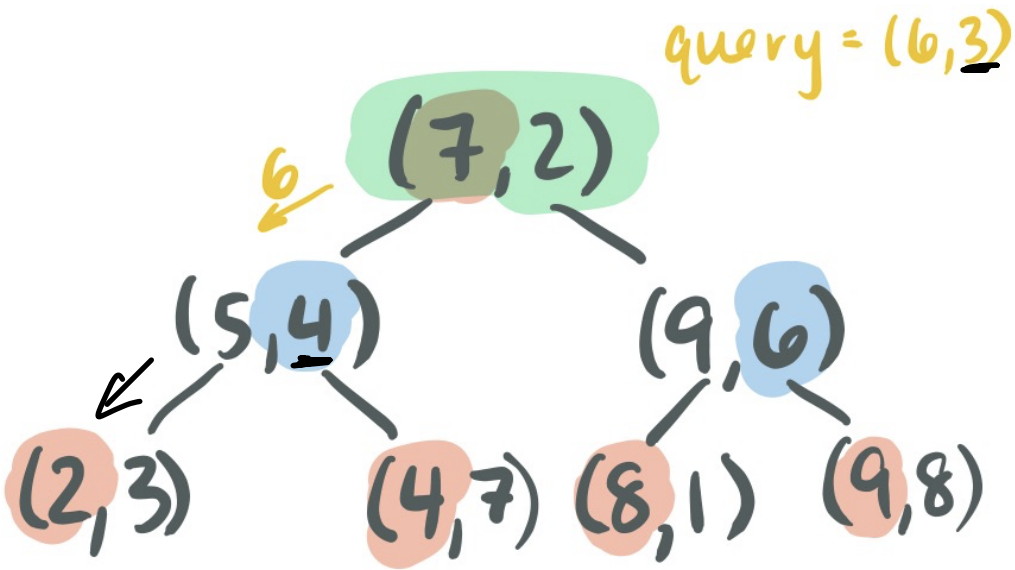
$$d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

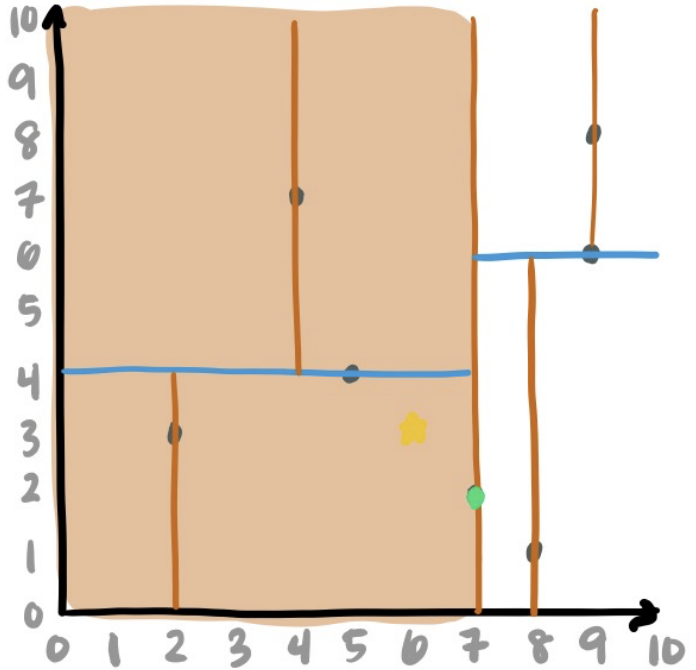
query = (6,3)



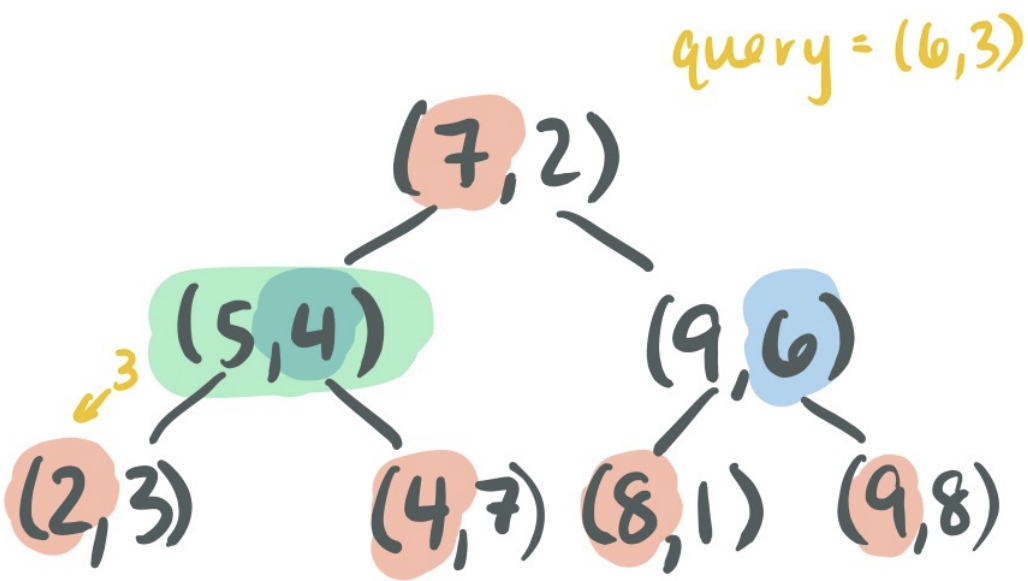


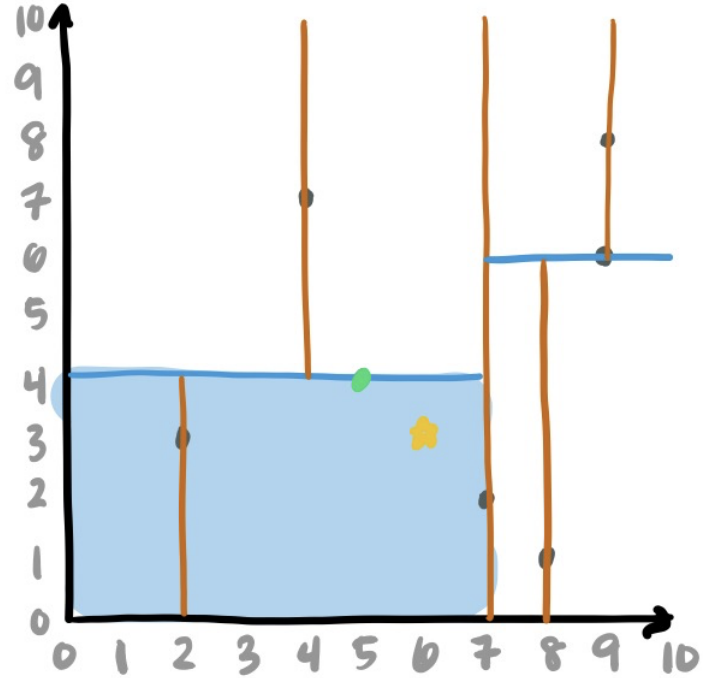
Nearest Neighbor KD Tree





Nearest Neighbor KD Tree

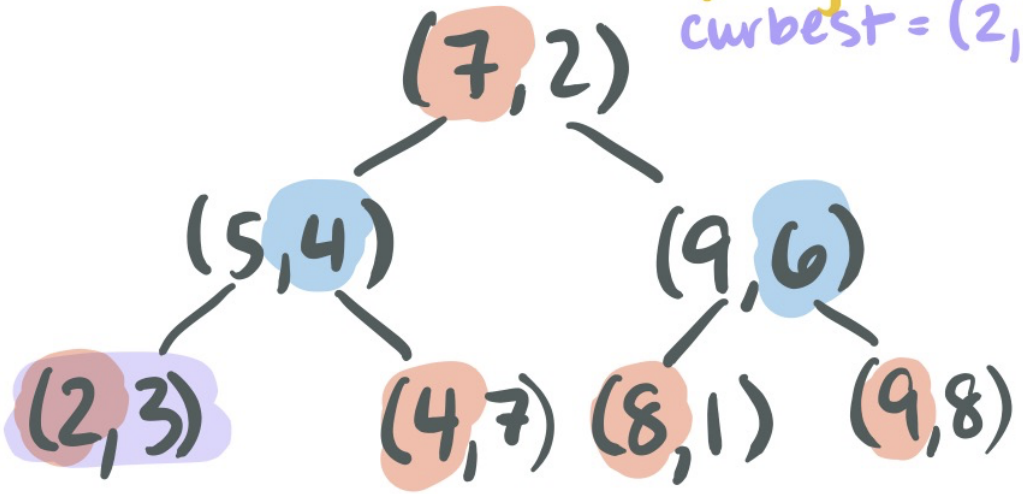


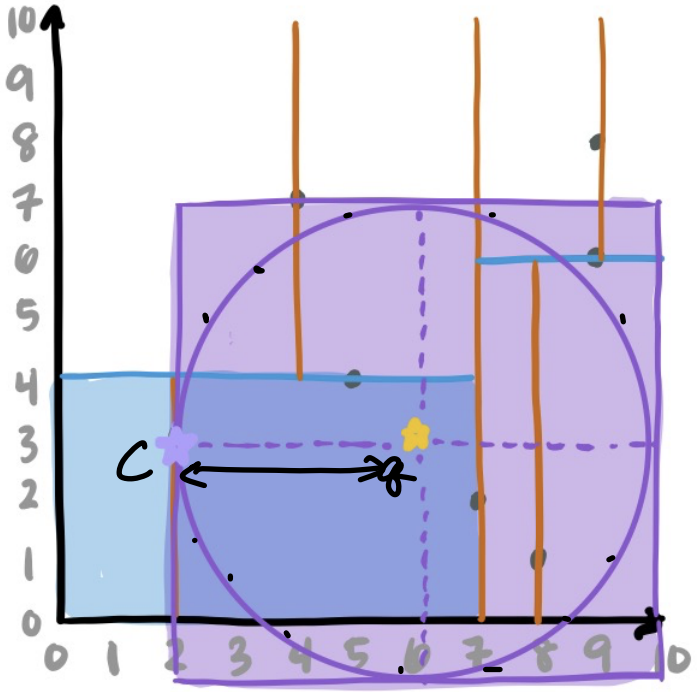


Nearest Neighbor KD Tree

Once a candidate is found, compute distance

query = (6,3)
curbest = (2,3)

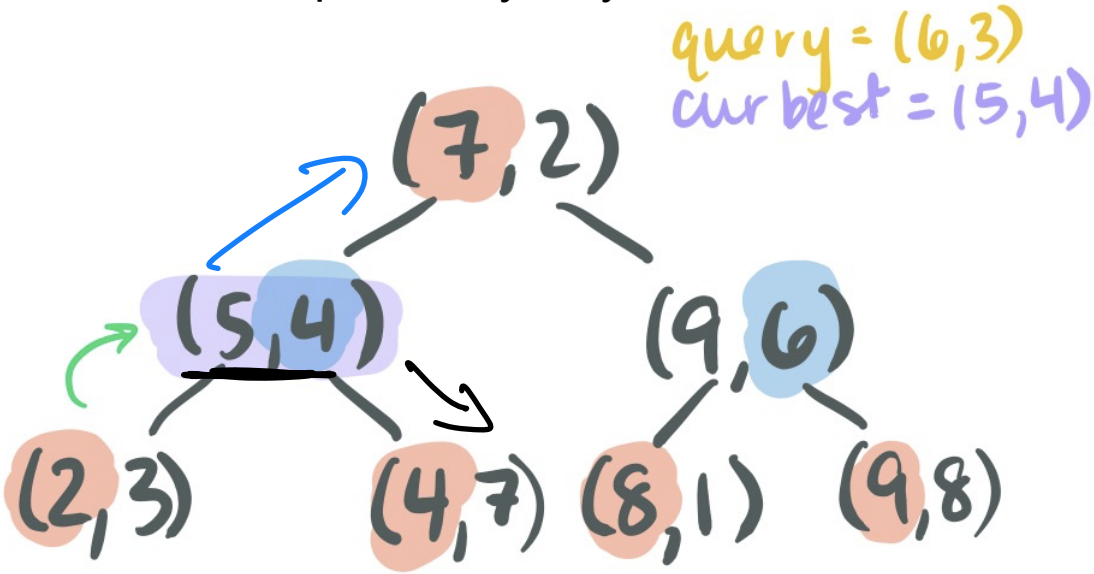


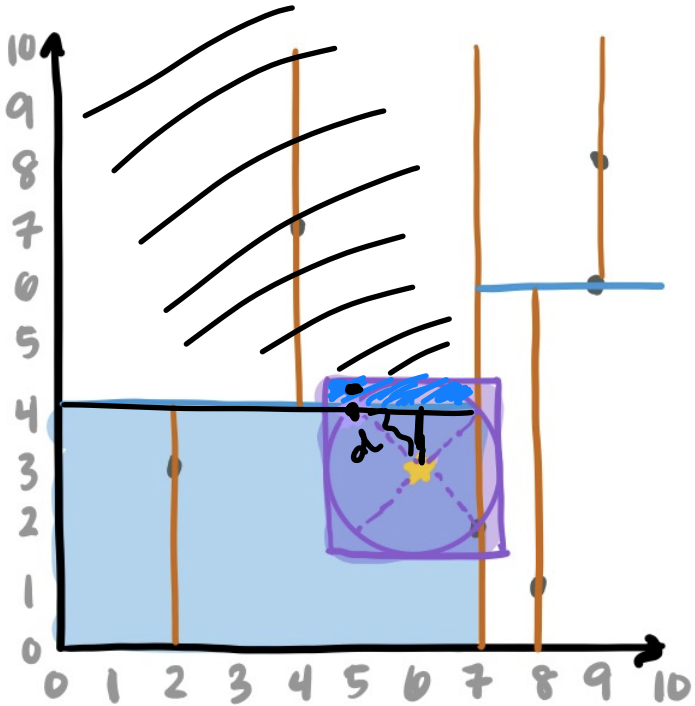


Nearest Neighbor KD Tree

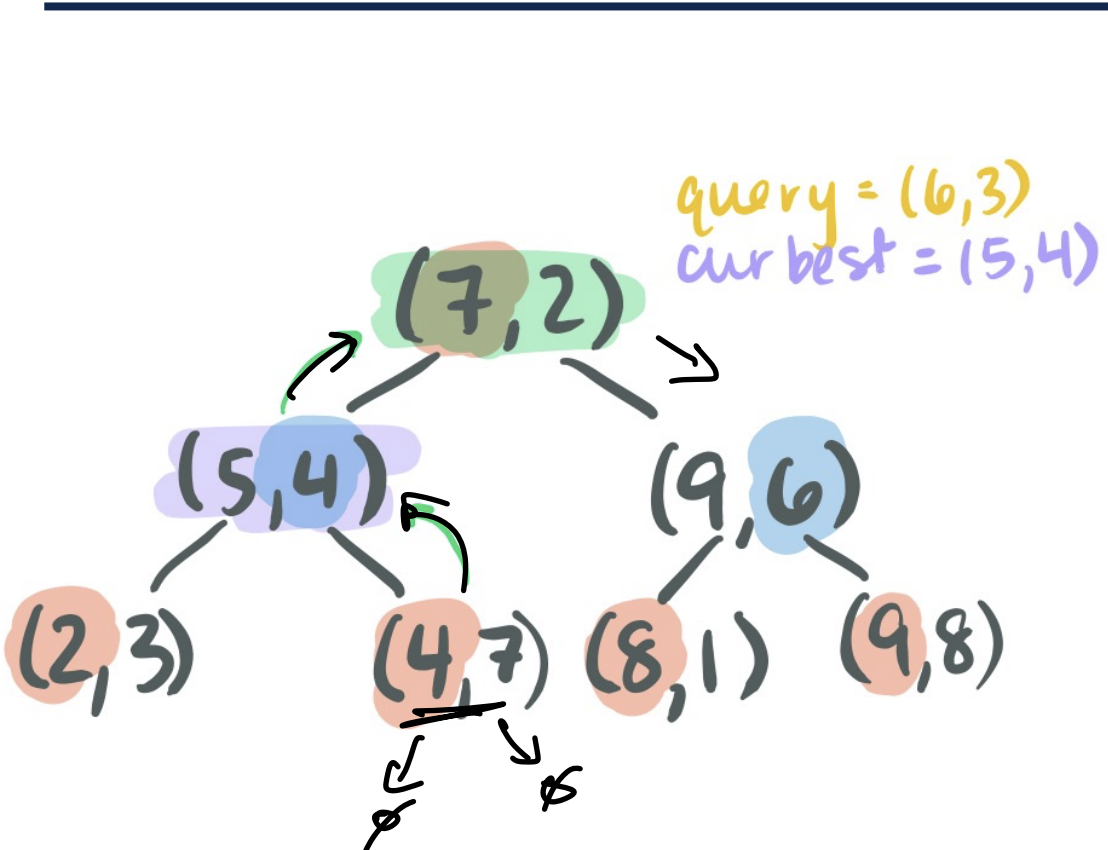
Backtracking: start recursing backwards

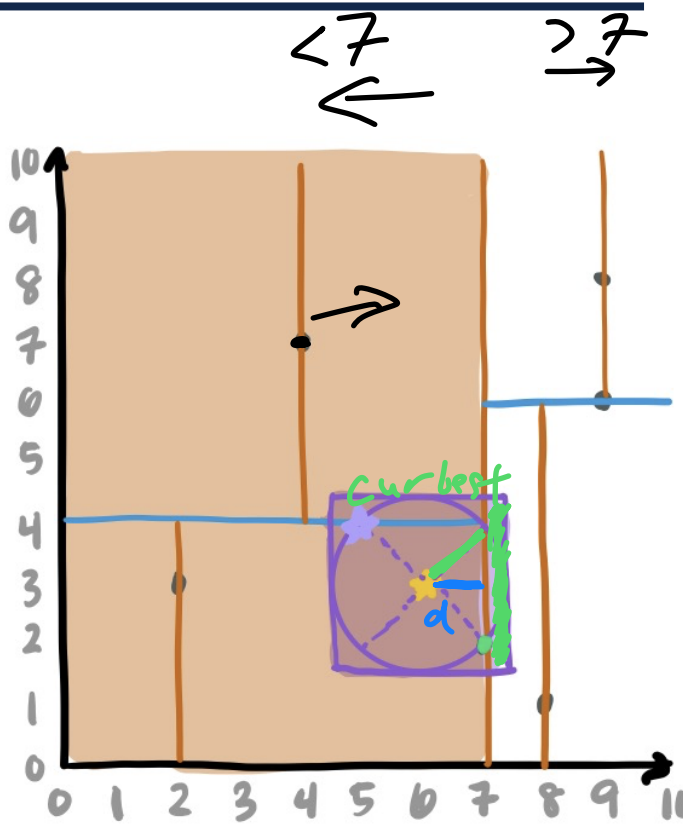
-- store “best” possibility as you trace back



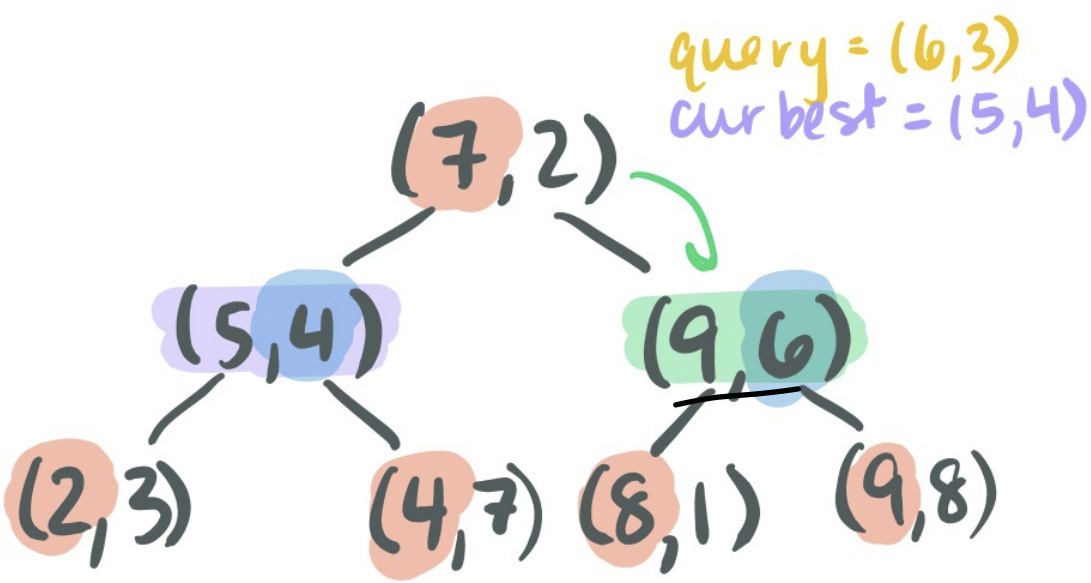


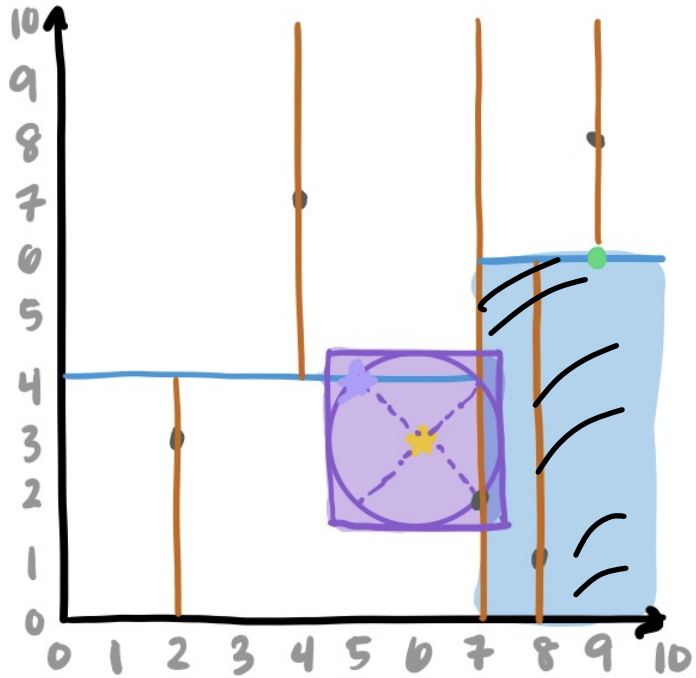
Nearest Neighbor KD Tree





Nearest Neighbor KD Tree





Build KD : $O(n \log n)$ $T(n) = T(\frac{n}{2}) + l = O(n \log n)$

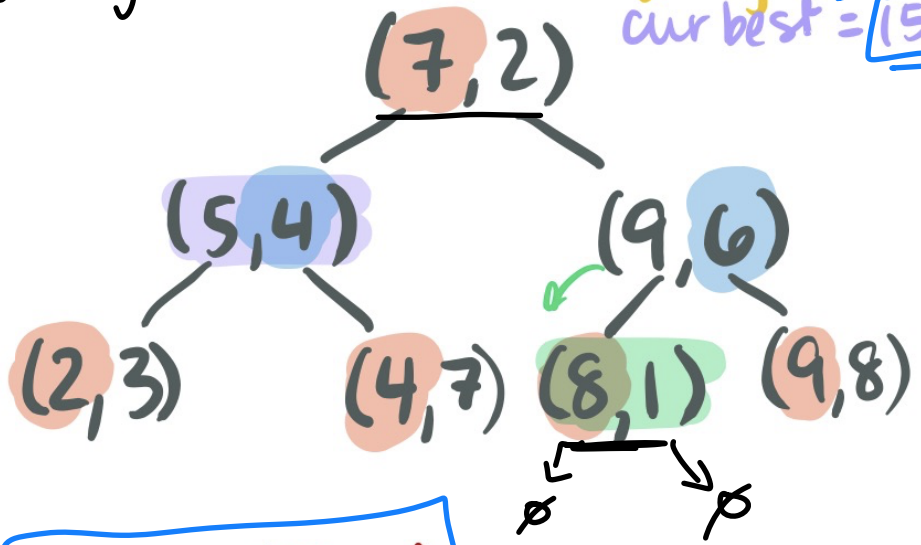
Nearest Neighbor KD Tree

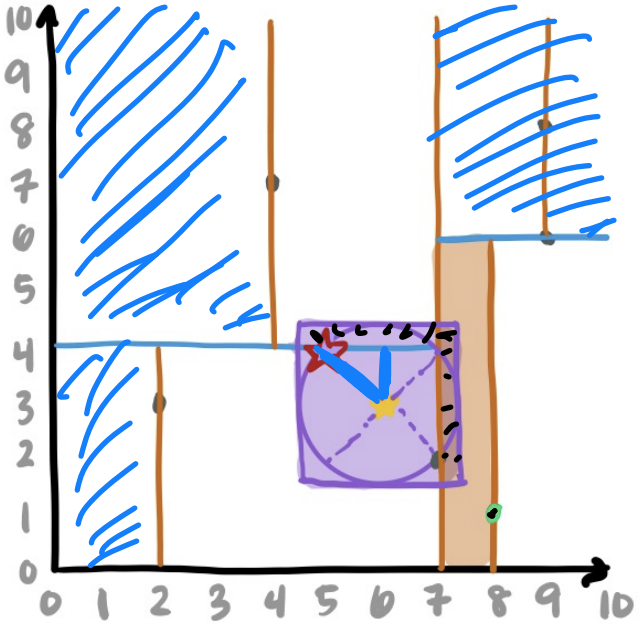
Worst Case: $O(n)$ Brute Force: $O(n)$

Average Case: $O(\log n)$

query = (6,3)

cur best = (5,4)





BEST: (5,4)

Definition of Lambda Function

A inline function without a name

	Speed (↑)	Code Size (↓)	
Lambda	✓	X	– Short functions called infrequently
Normal	X	✓	

Motivation

Surplus

= COUNTIF(B2:B13, “>=10”)

	Product	Quantity
1	**Product**	**Quantity**
2	**Oranges**	20
3	**Lemonade**	10
4	**Pies**	12
5	**Biscuits**	3
6	**Bananas**	20
7	**Cakes**	10
8	**Pizza**	5
9	**Orange Juice**	4
10	**Croissants**	15
11	**Grapefruits**	4
12	**Ice Cream**	12
13	**Apples**	20
14		
15	**Surplus**	8

CountIF Implementation

```
1  template <typename Iter, typename Cond>
2  int Countif(Iter begin, Iter end, Cond cond) {
3    int count = 0;
4    auto cur = begin;
5
6    while(cur != end) {
7      if(cond(*cur))
8        ++count;
9      ++cur;
10   }
11
12   return count;
13 }
```

```
1  bool isNegative(int num) { return (num < 0); }
2
3  class IsNegative {
4  public:
5      bool operator() (int num) { return (num < 0); }
6  };
7
8  int main() {
9    std::vector<int> numbers = {1, 102, 105, 4, 5, 27};
10
11   auto isnegl = [](int num) { return (num < 0); };
12   auto isnegfp = isNegative;
13   auto isnegfunctor = IsNegative();
14
15   std::cout << "There are " <<
     Countif(numbers.begin(), numbers.end(), isnegl)
       << " negative numbers" << std::endl;
16 }
```

Functor

isnegl
isnegfp
isnegfunctor

Implementation of Lambda Functions

[Capture](Input){ body }

Example Function

```
1   int big;
2   std::cout << "How big is big? ";
3   std::cin >> big;
4
5   auto isbig = [big](int num) { return (num >= big); };
6
7   std::cout << "There are " <<
      Countif(numbers.begin(), numbers.end(), isbig)
      << " big numbers" << std::endl;
8 }
```

input threshhold