

Data Structures

KD Tree 2 (quickSelect and Lambda)

CS 225

September 25, 2026

Brad Solomon



UNIVERSITY OF
ILLINOIS
URBANA - CHAMPAIGN

Department of Computer Science

Learning Objectives

Review KD Tree construction

Introduce quickSelect algorithm

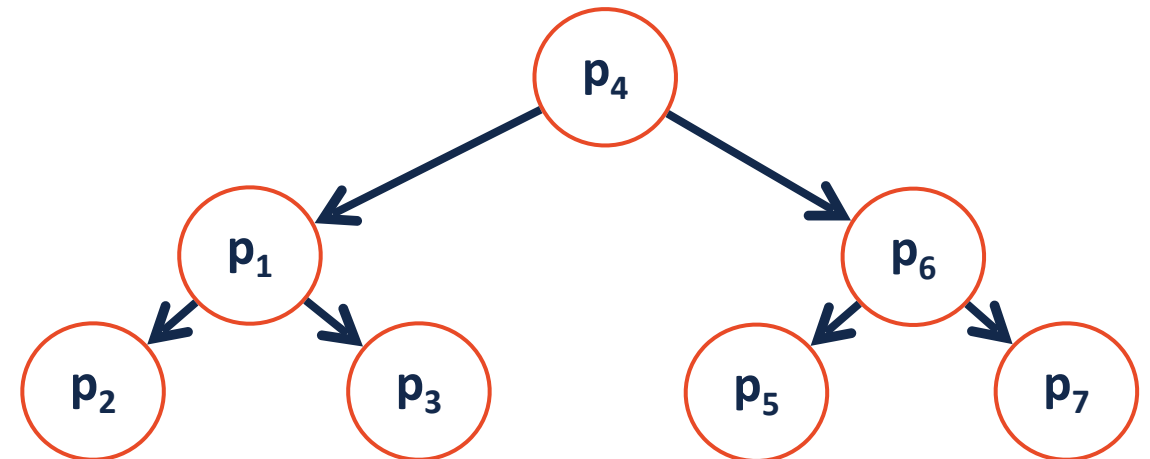
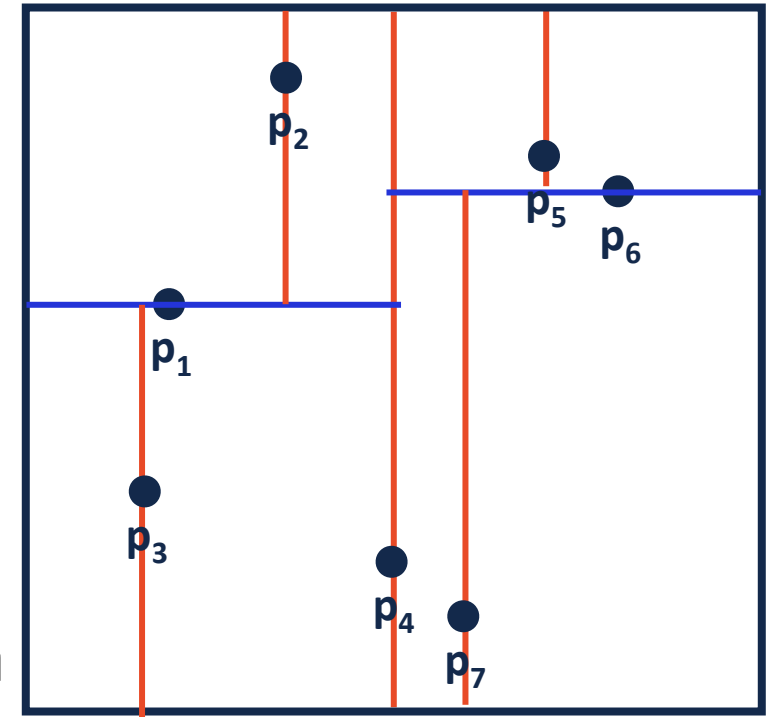
Discuss lambda functions

KD Tree Construction

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

Tree Construction:

1. Find **median point** along a particular dim
2. Recurse left and right, **swapping** dimension



Efficiently identifying median point in range

Last class we assumed we could find median point trivially!

... But how do we actually do this?

Efficiently identifying median point in range

Last class we assumed we could find median point trivially!

... But how do we actually do this?

Naive: Sort the list!

Problem:

Efficiently identifying median point in range

Last class we assumed we could find median point trivially!

... But how do we actually do this?

Naive: Sort the list!

Problem: We have to do this for each node!

Given median partition, at level i there are 2^k nodes

Each node has to sort $\frac{n}{2^k}$ points to find its own median.

Runtime: $\sim O(n \log n)$ per level * $O(\log n)$ levels = $O(n \log^2 n)$

Efficiently identifying median point in range

Last class we assumed we could find median point trivially!

... But how do we actually do this?

Better: We don't need to sort all items, we just need to find median!

QuickSelect Algorithm

An algorithm that can find the k -th smallest item in an unsorted list
... without fully sorting the list

How can this be useful for K-D Tree Construction?

QuickSelect Algorithm

Join Code: 225



Goal: Find the k-th smallest value without sorting

1. Pick a pivot (randomly)
2. **Partition** into smaller / larger around the pivot value
3. Recurse into the partition that contains k-th smallest value

When do we stop this recursion? What is our base case?

QuickSelect Algorithm

Goal: Find the k-th smallest value without sorting

Ex: Find the third smallest value in

7	2	9	4	1	8	5
---	---	---	---	---	---	---

Partition 1 randomly selects 5 as pivot

2	4	1	5	7	9	8
---	---	---	---	---	---	---

QuickSelect Algorithm

Goal: Find the k-th smallest value without sorting

Ex: Find the third smallest value in

7	2	9	4	1	8	5
---	---	---	---	---	---	---

Partition 1 randomly selects 5 as pivot

2	4	1	5	7	9	8
---	---	---	---	---	---	---

Partition 2 randomly selects 2 as pivot

1	2	4
---	---	---

QuickSelect Algorithm



Goal: Find the k-th smallest value without sorting

Ex: Find the third smallest value in

7	2	9	4	1	8	5
---	---	---	---	---	---	---

Partition 1 randomly selects 5 as pivot

2	4	1	5	7	9	8
---	---	---	---	---	---	---

Partition 2 randomly selects 2 as pivot

1	2	4
---	---	---

Partition 3 randomly "selects" 4 as pivot

4

Final Output: 3rd smallest value is 4!

1	2	4	5	7	9	8
---	---	---	---	---	---	---

QuickSelect Partition Algorithm

Goal: Partition items around pivot value as “smaller” or “larger”

1. Select a random pivot

Ex: $k = 3$

6	1	0	4	7	9	2	3
---	---	---	---	---	---	---	---

QuickSelect Partition Algorithm

Goal: Partition items around pivot value as “smaller” or “larger”

1. Select a random pivot

Ex: $k = 3$

6	1	0	4	7	9	2	3
---	---	---	---	---	---	---	---

2. Swap pivot value to end of array

6	1	0	3	7	9	2	4
---	---	---	---	---	---	---	---

QuickSelect Partition Algorithm

Goal: Partition items around pivot value as “smaller” or “larger”

1. Select a random pivot

Ex: $k = 3$

6	1	0	4	7	9	2	3
---	---	---	---	---	---	---	---

2. Swap pivot value to end of array

6	1	0	3	7	9	2	4
---	---	---	---	---	---	---	---

3. Partition elements

Small = 4

1	0	3	2	7	9	6	4
---	---	---	---	---	---	---	---

QuickSelect Partition Algorithm

Goal: Partition items around pivot value as “smaller” or “larger”

1. Select a random pivot

Ex: $k = 3$

6	1	0	4	7	9	2	3
---	---	---	---	---	---	---	---

2. Swap pivot value to end of array

6	1	0	3	7	9	2	4
---	---	---	---	---	---	---	---

3. Partition elements

Small = 4

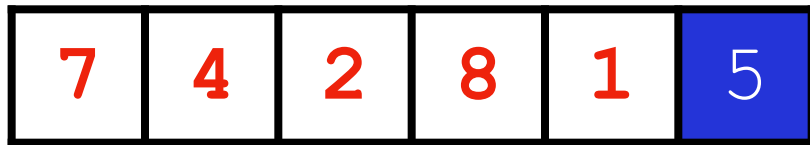
1	0	3	2	7	9	6	4
---	---	---	---	---	---	---	---

4. Swap pivot value to its proper place

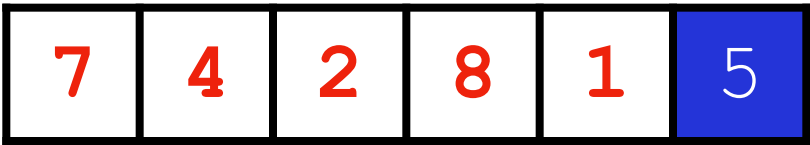
1	0	3	2	4	9	6	7
---	---	---	---	---	---	---	---

```
1 swap(arr[k], arr[length-1]);  
2  
3 for (int i = 0; i < length; i++){  
4     if (arr[i] < arr[length-1]){  
5         swap(arr[i], arr[small]);  
6         small++;  
7     }  
8 }  
9 swap(arr[small], arr[length-1]);
```

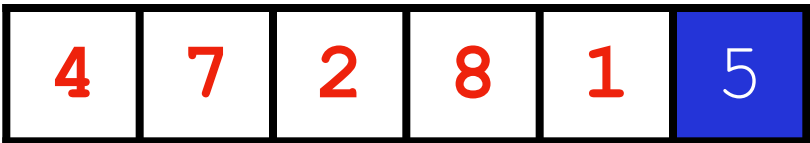
1D Partition Example with pivot $k = 2$ on $[7, 4, 5, 8, 1, 2]$



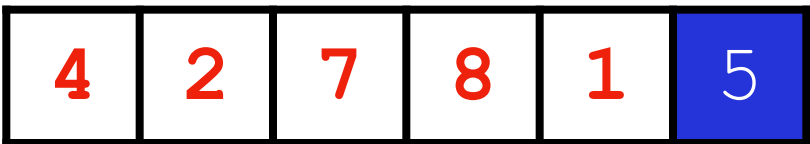
Index = 0, Small = 0



Index = 1, Small = 0



Index = 2, Small = 1



Index = 3, Small = 2

Join Code: 225



```
1 swap(arr[k], arr[length-1]);
2
3 for (int i = 0; i < length; i++){
4     if (arr[i] < arr[length-1]){
5         swap(arr[i], arr[small]);
6         small++;
7     }
8 }
9 swap(arr[small], arr[length-1]);
```

1D Partition Example with pivot $k = 2$ on $[7, 4, 5, 8, 1, 2]$

Input Array	Input Index	Input Small
7, 4, 2, 8, 1, 5	0	0
7, 4, 2, 8, 1, 5	1	0
4, 7, 2, 8, 1, 5	2	1
4, 2, 7, 8, 1, 5	3	2
	4	
	5	



```
1 swap(arr[k], arr[length-1]);
2
3 for (int i = 0; i < length; i++){
4     if (arr[i] < arr[length-1]){
5         swap(arr[i], arr[small]);
6         small++;
7     }
8 }
9 swap(arr[small], arr[length-1]);
```

1D Partition Example with pivot $k = 2$ on $[7, 4, 5, 8, 1, 2]$

Input Array	Input Index	Input Small
7, 4, 2, 8, 1, 5	0	0
7, 4, 2, 8, 1, 5	1	0
4, 7, 2, 8, 1, 5	2	1
4, 2, 7, 8, 1, 5	3	2
4, 2, 7, 8, 1, 5	4	2
4, 2, 1, 8, 7, 5	5	3

Final Output:

4	2	1	5	7	8
---	---	---	---	---	---

QuickSelect Big O

Join Code: 225

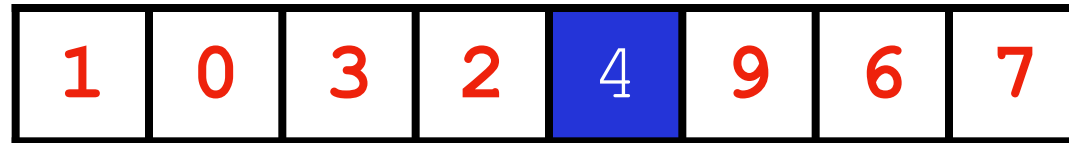


What is the worst case partition **value** for the following array?

6	1	0	4	7	9	2	3
---	---	---	---	---	---	---	---

QuickSelect Big O

Why is the actual median value the best case partition value?



QuickSelect Big O

We are not sorting the array so it should be faster than $O(n \log n)$!

Average Case:

Big O:

QuickSelect Big O



We are not sorting the array so it should be faster than $O(n \log n)$!

Average Case: $O(N)$

First partition always $O(N)$

Random pivot **expectation:**

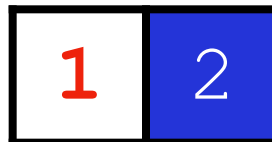
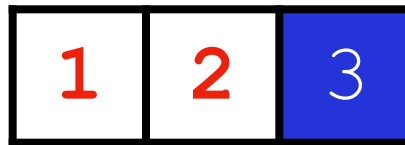
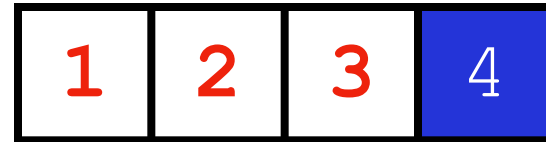
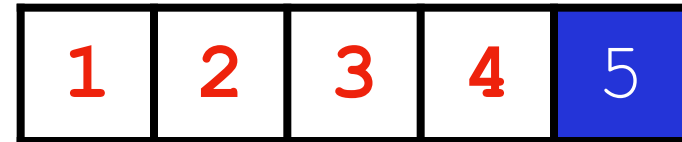
Given k as pivot index

$$\frac{1}{n} \sum_{k=0}^{n-1} \mathbb{E}[\max(k, n - k - 1)]$$

Unproven: This is also $O(N)$

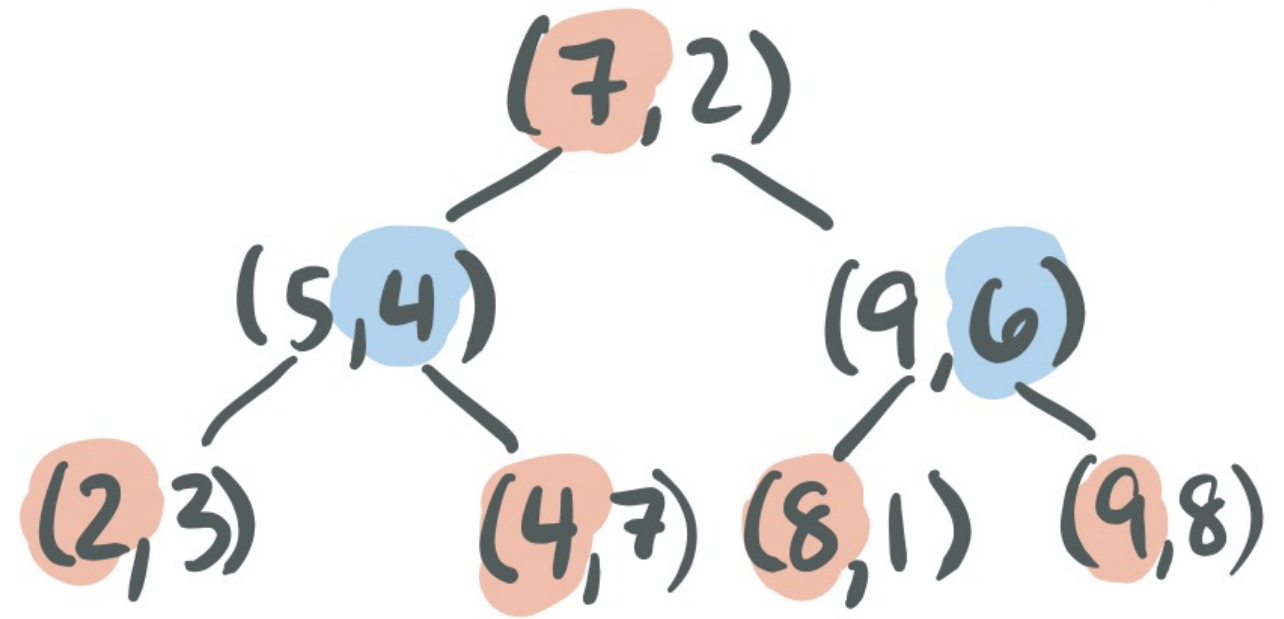
Big O: $O(N^2)$

Ex: Find 1st smallest value



QuickSelect in a KD Tree

How do we use a 1D algorithm in a k-dimension tree constructor?



Hint: **Lambda functions!**

Functions as arguments

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

A10 ✕ ✓ fx =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		
11			

Functions as arguments

Countif.hpp

```
10 template <typename Iter, typename Pred>
11 int Countif(Iter begin, Iter end, Pred pred) {
12     int count = 0;
13     auto cur = begin;
14
15     while(cur != end) {
16         if(pred(*cur))
17             ++count;
18         ++cur;
19     }
20
21     return count;
22 }
```

Lambda Functions in C++

Here are several ways to write a function as an object

main.cpp

```
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, 41, -7, 999};
10
11     auto isnegl = [](int num) { return (num < 0); };
12     auto isnegfp = isNegative;
13     auto isnegfunctor = IsNegative();
14
15     cout << "There are " << Countif(numbers.begin(), numbers.end(), _____)
16         << " negative numbers" << std::endl;
17 }
```

Lambda Functions in C++

An inline function without a name

[Capture](Arg List){ Function Body }

Lambda Functions in C++

An inline function without a name

[Capture](Arg List){ Function Body}

Capture: Takes the value of object based on when the lambda was defined, NOT the current value of the object!

Arg List: Standard way of inputting into a function

Function Body: Code can use both capture vars and arg vars

Lambda Functions in C++



main.cpp

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

Lambda Functions in C++



main.cpp

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

Useful for mp_mosaics!

KD-Tree will split points in one dimension

When comparing, we need to remember what dimension we are in!

Final Tips

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

The suggestions in the writeup should be followed carefully

```
1  template <typename RandIter, typename Comparator>
2  void select(RandIter start, RandIter end, RandIter k, Comparator cmp)
3  {
4      /**
5       * @todo Implement this function!
6       */
7
8  }
9
```

Understanding 'randIter'

An iterator is a container giving access in different ways:

Forward

Bidirectional

Random Access

Implementing quickselect with RandIter

Random Access Iterator lets you:

Swap items using `std::swap()`

```
1  template <typename RandIter, typename Comparator>
2  void BlackBox(RandIter A, RandIter B)
3  {
4
5      std::swap(*A, *B) ;
6
7
8  }
9
```

Hint: We just covered when you might want to swap in quick select!

Implementing quickselect with RandIter

Random Access Iterator lets you:

Access container indices using math operations

```
randIter A;
```

```
auto nth = *(A + n);
```

Get distance between two iterators

```
randIter A, B;
```

```
A < B;           // True if A is earlier in container than B
```

```
A - B;           // The distance between A and B
```

Implementing quickselect with RandIter

Random Access Iterator lets you:

Do most things you'd expect an array to be able to do!

The power of the **Interface!**

https://en.cppreference.com/w/cpp/iterator/random_access_iterator