Sorting

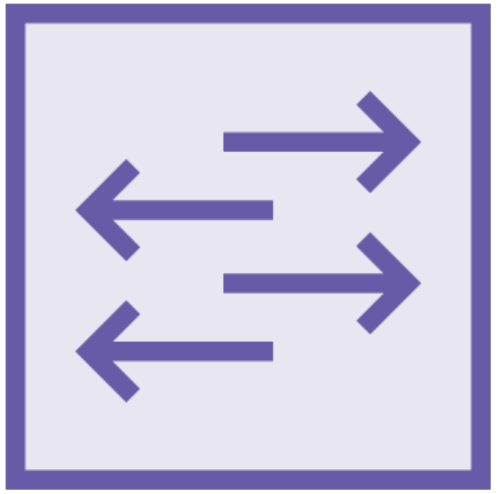
Kate Gregory

@gregcons www.gregcons.com/kateblog

Sorting

sort(it1, it2)

Overload operator<



sort(it1, it2, lambda)

Stable Sort

Sort, then sort again by another property

Should “ties” on the second sort be in their original order?

You decide

Is It Sorted?

is_sorted(it1, it2)

is_sorted(it1, it2, lambda)

Find the Largest, or Smallest, or ...

Unsorted collection	Sorted collection
Checks every element	
max_element()	Largest is last, smallest is first
min_element()	
find()	Can do a binary search for any value
	upper_bound()
	lower_bound()

Shuffle

```
#include <random>

// . . .

random_device randomdevice;

mt19937 generator(
        randomdevice());

shuffle(it1, it2,generator)
```

- Bring in another header
- Once your collection is in place
- Create a random device
- Create a Mersenne Twister to use as a generator
- Pass the generator to shuffle

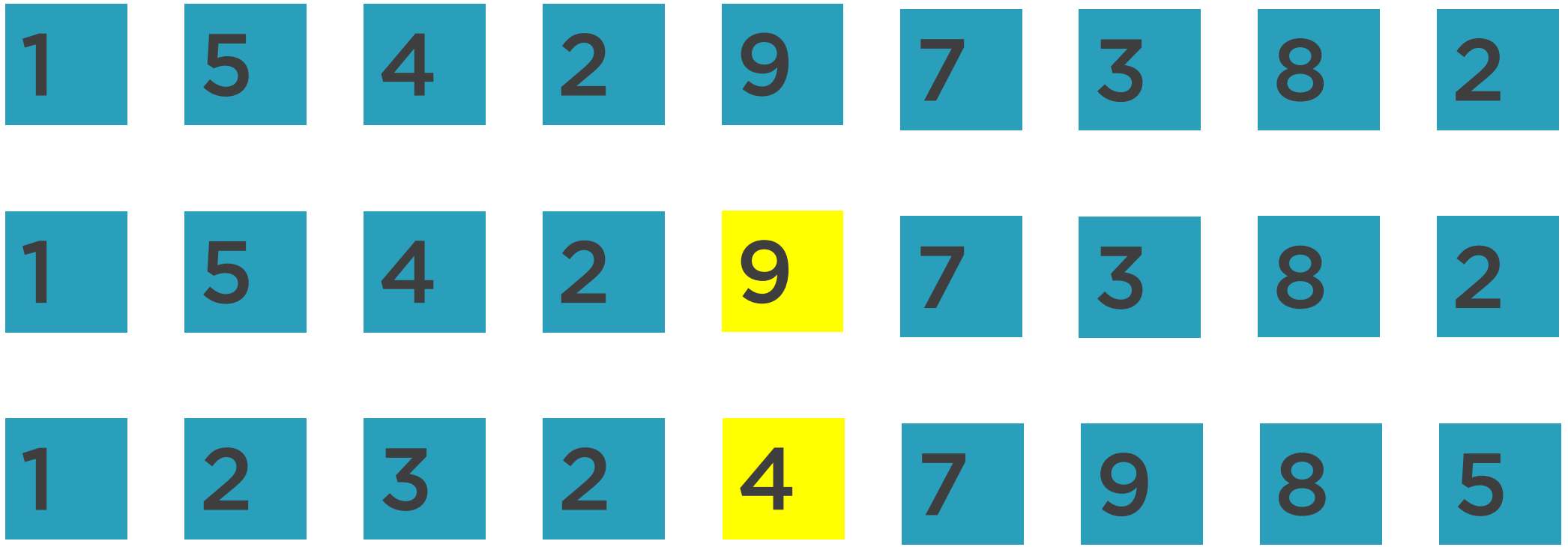
Partial Sorting

partial_sort()

is_sorted_until

partial_sort_copy()

Nth Element



What are you trying to do?

Summary

- Sorting is trivial for elements with <
- Provide a lambda for complete control
- When sorting multiple times, consider stable_sort
- Want to know if it's sorted? is_sorted()
- In a sorted collection, faster searches are available
- You can easily shuffle (unsort) as well
- You can also do partial sorts and rough groupings if that's all you need