RANGES IN C++20

A brief introduction to ranges

THE ONE RANGES PROPOSAL

P0896r4 was the basis for ranges in C++20

DEFINITIONS

RANGE

Usually easiest to think of it as a pair of iterators, referring to the begin/end of a forward-iterable container (vector, list, map, etc.)

The elements of the range are actually the following:

- An iterator marking the begining
- A sentinel marking the end

SENTINEL

The sentinel can be one of three things:

- An iterator to just past the end of the range; this is normal for standard algorithms/containers
- A value, the first occurrence of which will *not* end up in the range
- A size, the number of elements in a `counted_range`

VIEW

A type of range with constant-time copy/move/size operations

(Think `std::vector`, not `std::list`)

RANGE ADAPTER

A utility that transforms a (viewable) range into a view, with custom behavior

Examples include:

- `all`, `reverse`
- `filter, transform`
- `take`, `take_while`, `drop`, `drop_while`
- `elements`, `keys`, `values`
- and others...

	(https://wg21.link/P0475R1)							
`std::bit_cast()`	P0476R2 (https://wg21.link/P0476R2)							
Integral power-of-2 operations	P0556R3 (https://wg21.link/P0556R3)	9						
Improving the return value of erase-like algorithms	P0646R1 (https://wg21.link/P0646R1)	9		19.21*				
`std::destroying_delete`	P0722R3 (https://wg21.link/P0722R3)	9	9					
`std::is_nothrow_convertible`	P0758R1 (https://wg21.link/P0758R1)	9	9	19.23*				
Add shift to `<algorithm>`	P0769R2 (https://wg21.link/P0769R2)			19.21*				
Constexpr for `std::swap()` and swap related functions	P0879R0 (https://wg21.link/P0879R0)	10						
`std::type_identity`	P0887R1 (https://wg21.link/P0887R1)	9	8	19.21*				
Concepts library	P0898R3 (https://wg21.link/P0898R3)	10		19.23*				
`constexpr` comparison operators for `std::array`	P1023R0 (https://wg21.link/P1023R0)	10	8					
`std::unwrap_ref_decay` and `std::unwrap_reference`	P0318R1 (https://wg21.link/P0318R1)	9	8	19.21*				
`std::bind_front()`	P0356R5 (https://wg21.link/P0356R5)	9						
`std::reference_wrapper` for incomplete types	P0357R3 (https://wg21.link/P0357R3)	9	8					
Fixing `operator>>(basic_istream&, CharT*)`	P0487R1 (https://wg21.link/P0487R1)		8	19.23*				
Library support for `char8_t`	P0482R6 (https://wg21.link/P0482R6)	9		19.23*				
Utility functions to implement uses-allocator construction	P0591R4 (https://wg21.link/P0591R4)	9						
DR: `std::variant` and `std::optional` should propagate copy/move triviality	P0602R4 (https://wg21.link/P0602R4)	8.1	8	19.11*				
A sane `std::variant` converting constructor	P0608R3 (https://wg21.link/P0608R3)	10	9					
`std::function`'s move constructor should be `noexcept`	P0771R1 (https://wg21.link/P0771R1)	7.2	6	19.22*				
The One Ranges Proposal	P0896R4 (https://wg21.link/P0896R4)							
Heterogeneous lookup for unordered containers	P0919R3 (https://wg21.link/P0919R3)			19.23*				
`<chrono> zero()`, `min()`, and `max()` should be `noexcept`	P0972R0 (https://wg21.link/P0972R0)	9	8	19.14*				
`constexpr` in `std::pointer_traits`	P1006R1 (https://wg21.link/P1006R1)	9	8					

COMPILER SUPPORT FOR RANGES

As of now, there is no range support baked into any compiler

INSTEAD, I USED THE FOLLOWING IMPLEMENTATIONS

RANGE-V3

- Eric Niebler's original ranges library
- This is the basis for The One Ranges Proposal
- Requires only C++11 or higher, re-implements concepts
- Has many more features than what will be in C++20

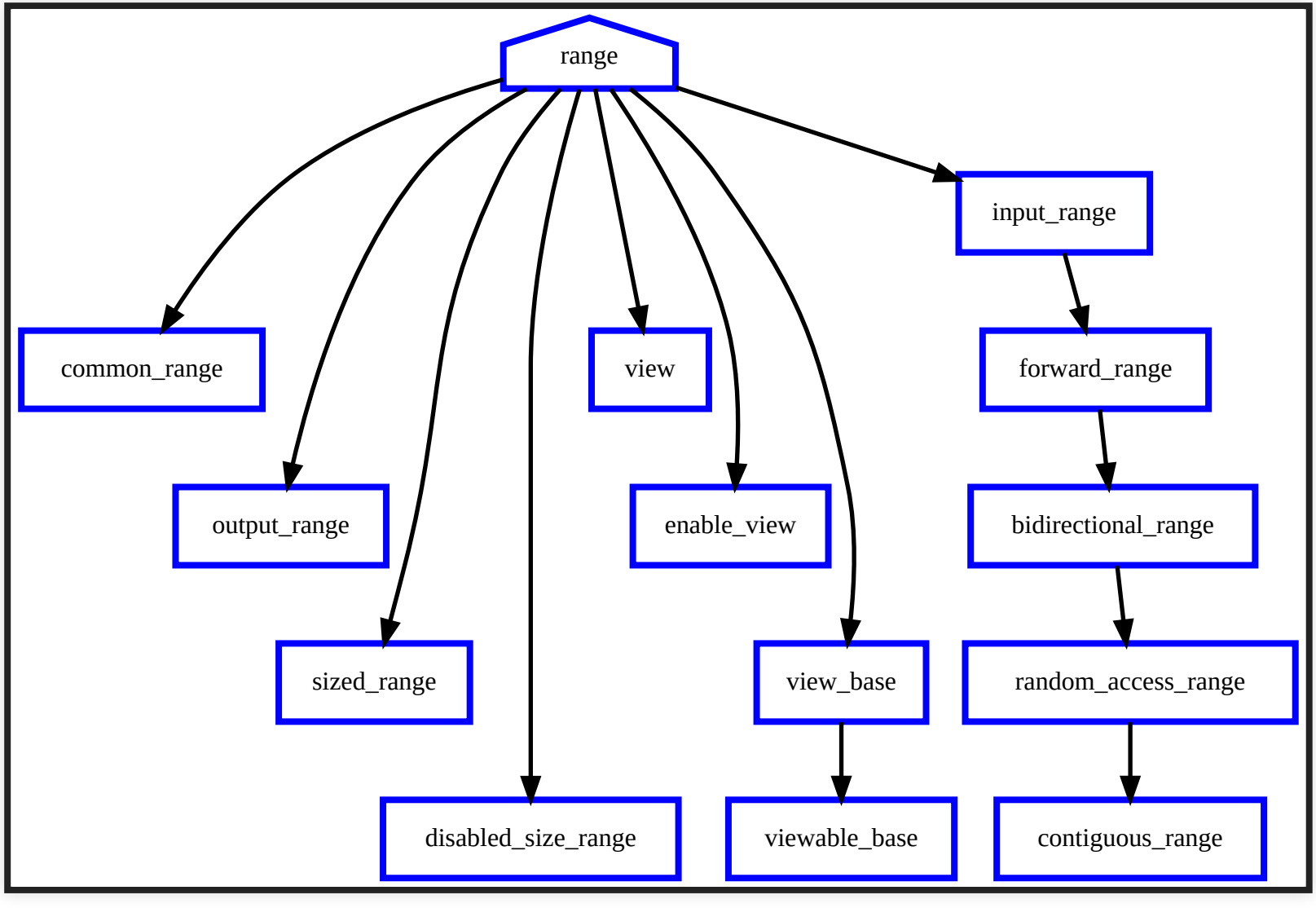
CMCSTL2

- Casey Carter's reference implementation of C++20 ranges
- Requires C++17, with `-fconcepts`
- Should exactly match what is going into the C++20 standard

NANORANGE

- Tristan Brindle's implementation of C++20 ranges
- Requires C++17, with `-fconcepts`
- Another implementation based on the C++20 standard

RANGE CONCEPT HIERARCHY



RANGE ADAPTERS IN DETAIL

There are a few ways to call range adapters that will give the exact same results:

```
// Normal function syntax
adapter(range);

// Pipe operator syntax
range | adapter;
```

EXAMPLES

FUNCTION SYNTAX

The normal function syntax looks a bit awkward, especially when chaining together several range adapters:

```
for (auto e : Ranges::views::take(
  Ranges::views::iota(17), 5))
{
  std::cout << " " << e;
}
// 17 18 19 20 21
```

THE PIPE OPERATOR

The pipe operator (|) allows for easy chaining of range adapters:

```
for (auto e : Ranges::views::iota(17)
  | Ranges::views::take(5))
{
  std::cout << " " << e;
}
// 17 18 19 20 21
```

take_while

The `take_while` range adapter only takes elements while the condition is met:

```
for (auto e : Ranges::views::iota(17)
  | Ranges::views::take_while(LessThanTwenty)
  | Ranges::views::take(5))
{
  std::cout << " " << e;
}
// 17 18 19
```

drop_while

`drop_while` is similar, but it only starts taking elements once the condition is first met:

```
for (auto e : Ranges::views::iota(17)
  | Ranges::views::drop_while(LessThanTwenty)
  | Ranges::views::take(5))
{
  std::cout << " " << e;
}
// 20 21 22 23 24
```

reverse

The `reverse` adapter reverses the view:

```
for (auto e : Ranges::views::iota(17)
  | Ranges::views::drop_while(LessThanTwenty)
  | Ranges::views::take(5)
  | Ranges::views::reverse)
{
  std::cout << " " << e;
}
// 24 23 22 21 20
```

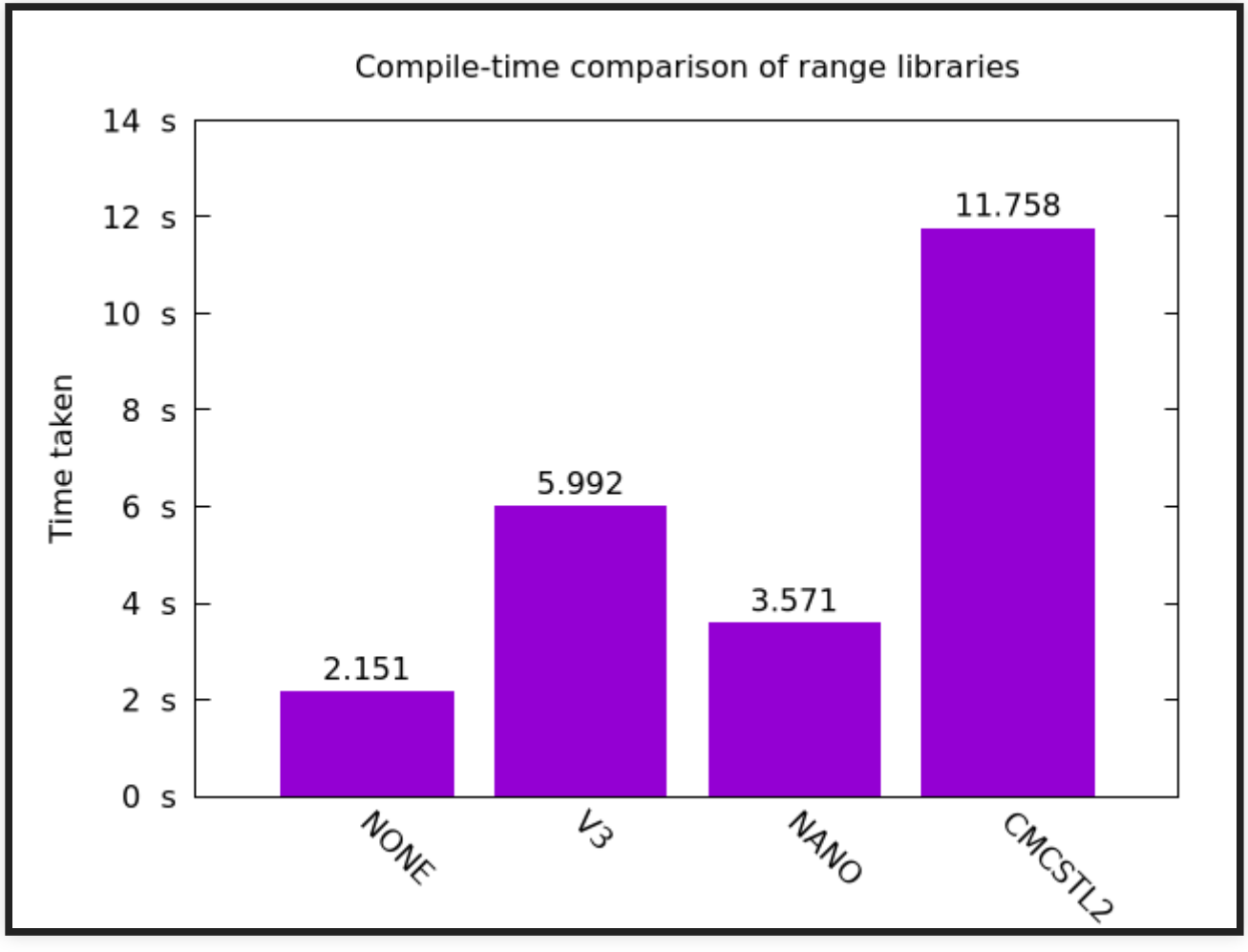
`reverse` CAN BE COSTLY

Taking the `reverse` of an unbounded range is a bad idea:

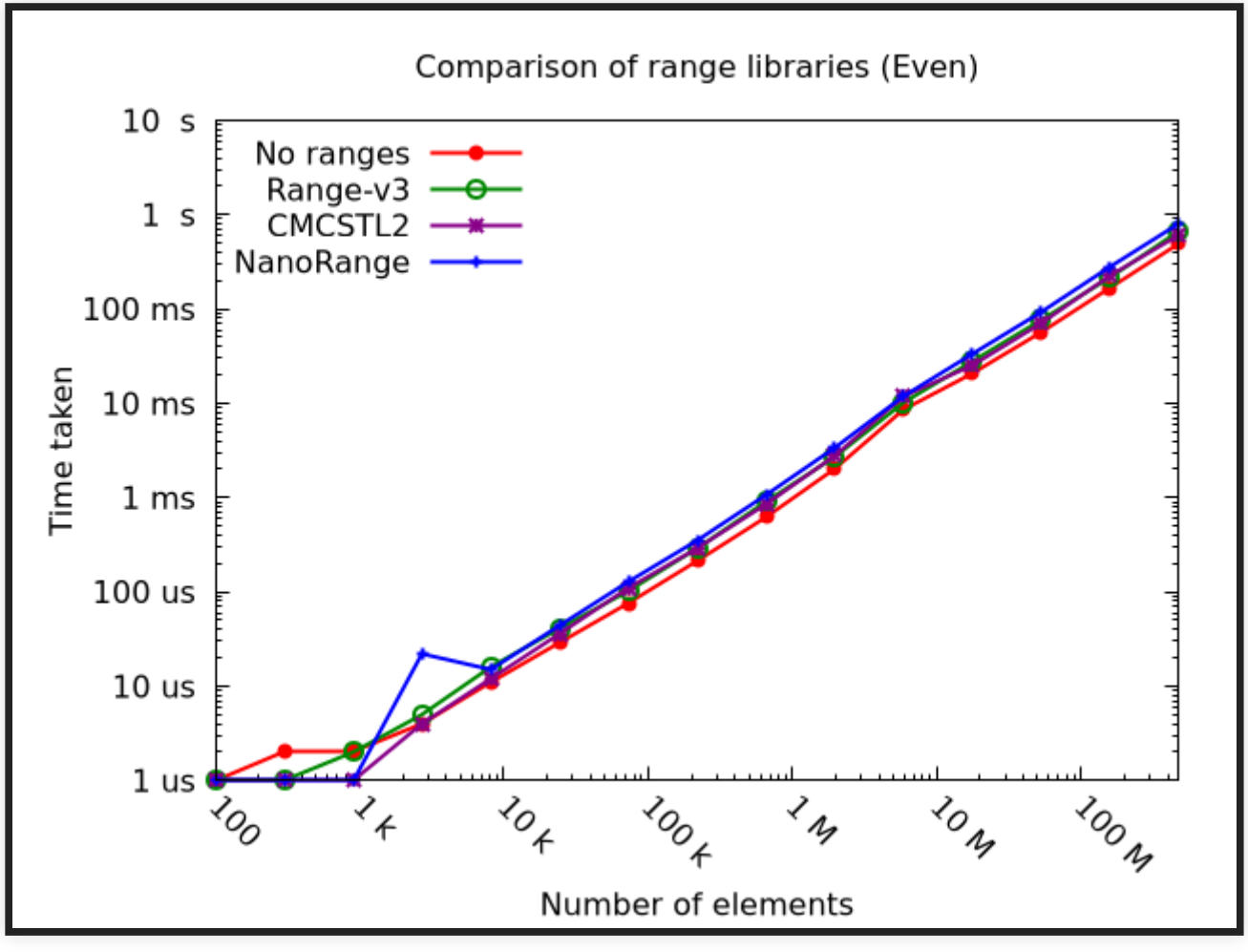
```
for (auto e : Ranges::views::iota(17)
  | Ranges::views::drop_while(LessThanTwenty)
  | Ranges::views::reverse
  | Ranges::views::take(5))
{
  std::cout << " " << e;
}
// ERROR, range is [inf - 5, inf) or so
```

RESULTS

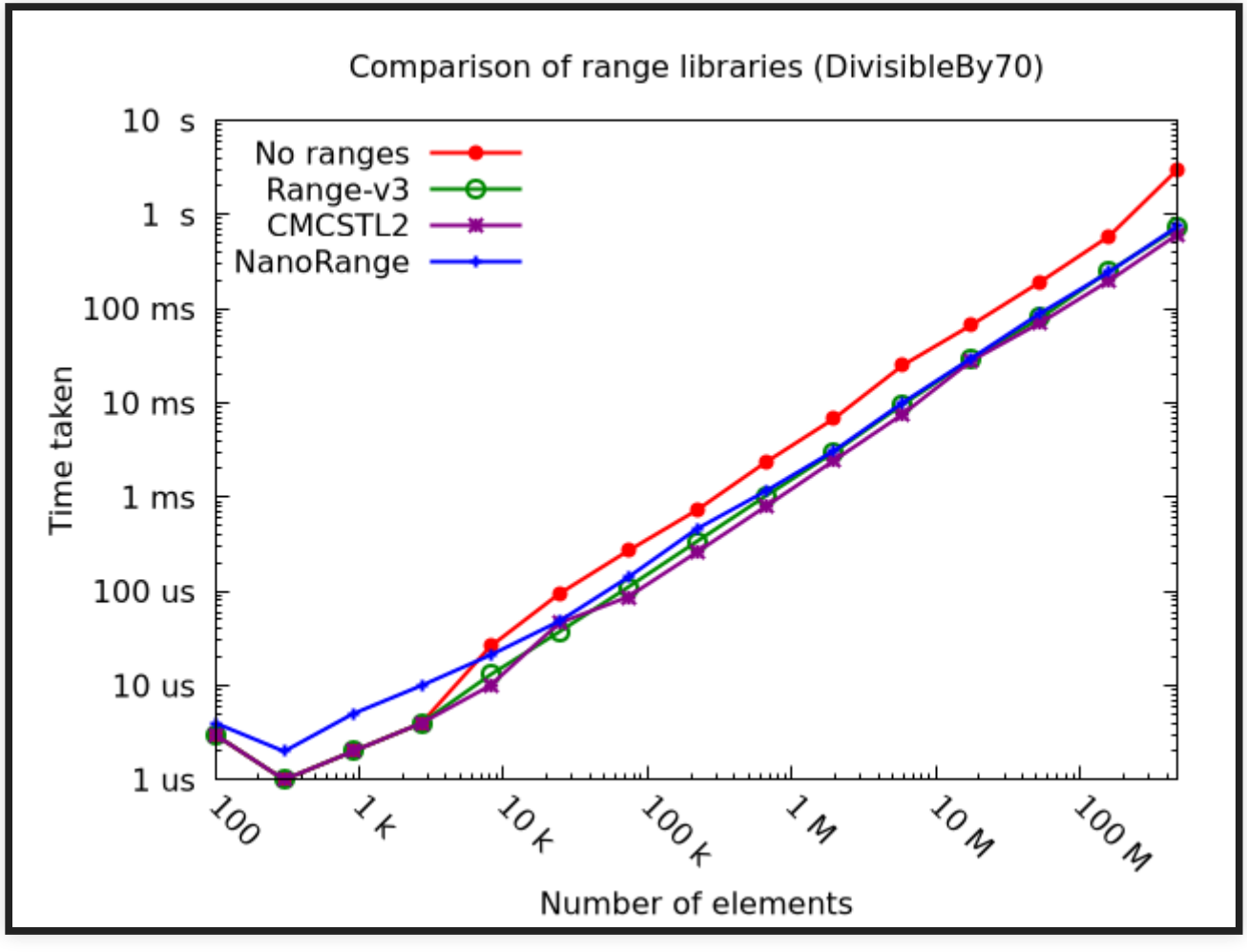
COMPILE TIME COMPARISON



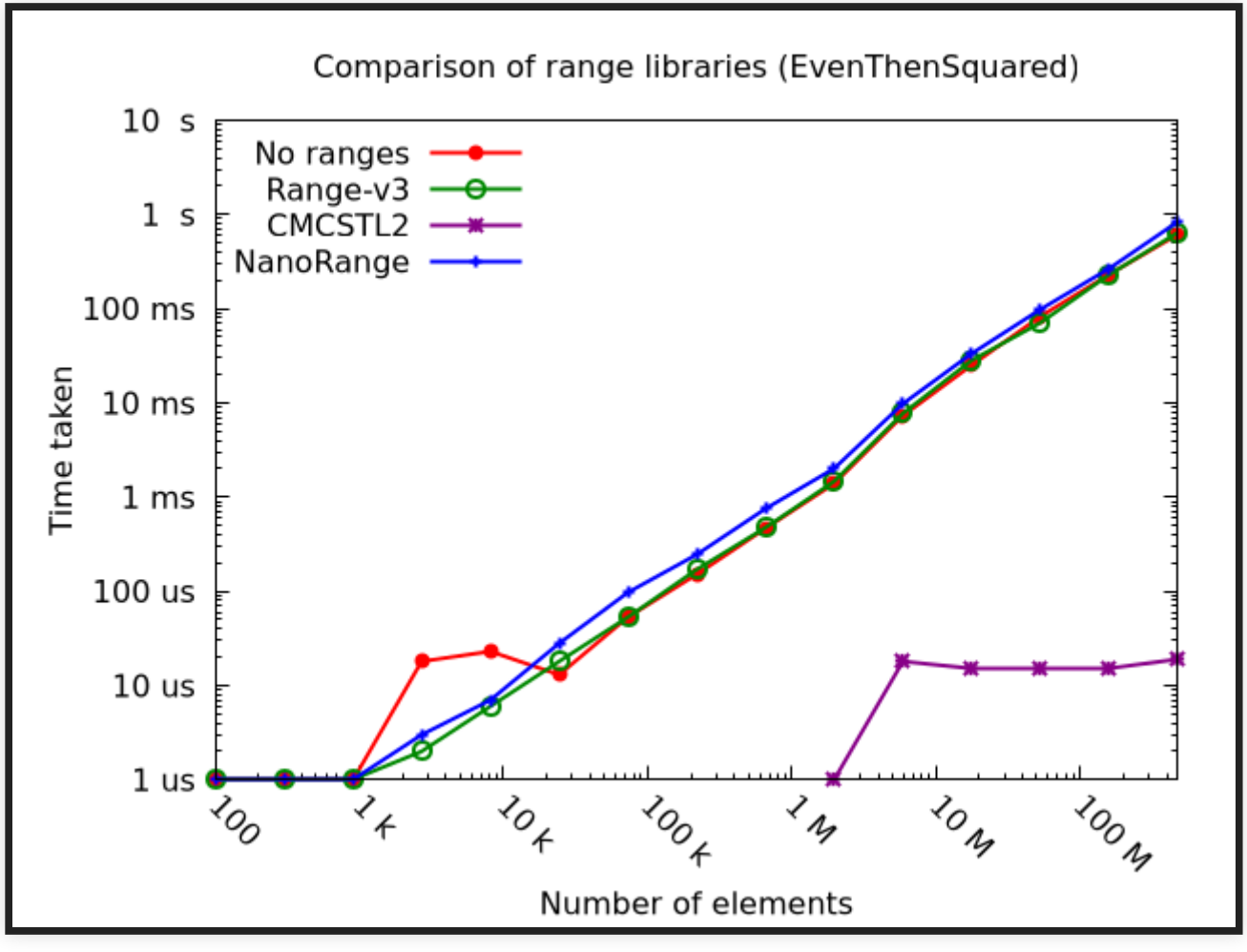
RUN-TIME - IS EVEN



RUN-TIME - DIVISIBLE BY SEVENTY

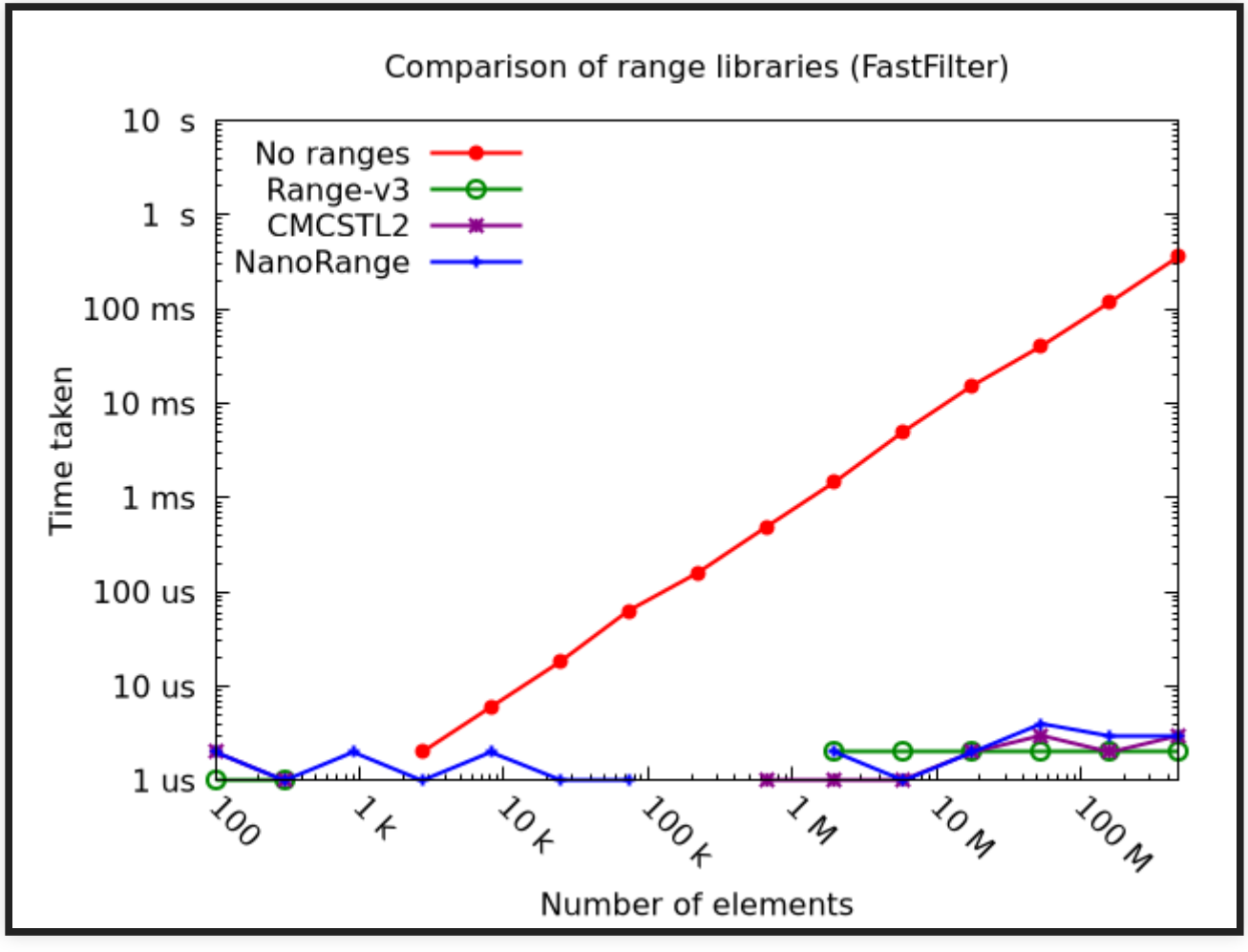


RUN-TIME - IS EVEN SQUARED



(Maybe a bug with CMCSTL2 here, I had to work around it)

RUN-TIME - FAST FILTER



CODE AND PRESENTATION

The source code is available at my GitHub page:
https://github.com/ejricha/examples

The presentation is available as well:
https://github.com/ejricha/presentations

Also posted on the CppMaryland GitHub page:
https://github.com/cppmaryland/presentations

REFERENCES

- https://en.cppreference.com/w/cpp/ranges
- http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0896r4.pdf
- http://ericniebler.com/2018/12/05/standard-ranges/
- https://www.fluentcpp.com/2018/12/07/algorithms-on-ranges/
- https://meetingcpp.com/mcpp/slides/2018/C++%20Concepts%20and%20Ranges%20-%20How%20to%20use%20them.pdf

YouTube Videos

CppCon 2019: Dvir Yitzcha...

CppCon 2019: Chris Di Bel...

CppCon 2019: Chris Di Bel...

CppCon 2019: Jeff Garlan...

CppCon 2019: Tristan Brin...

Introduction to C++ Ranges

C++ Concepts and Ranges...

QUESTIONS?