Better Algorithm Intuition

Conor Hoekstra (he/him)

code_report

codereport

NVIDIA RAPIDS

https://rapids.ai

#include

1. https://github.com/codereport/Talks

“I’m not an expert, I’m just a dude.”

- Scott Schurr, CppCon 2015

About Me

- I'm a Senior Library Software Engineer for NVIDIA
  - Working on the RAPIDS AI team (http://rapids.ai)
- I am a programming language enthusiast
- I've been coding in C++ for 5+ years*
- I love `auto` (AAA)
- I prefer **east const** (1 const west)
- I love algorithms and beautiful code
- I have a YouTube channel

C++ now 2019 MAY 6-10 cppnow.org

Conor Hoekstra

Algorithm Intuition

Video Sponsorship Provided By: JetBrains

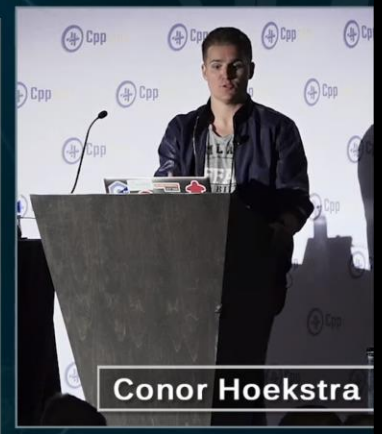
```
template<class T>
using rev = reverse_iterator<T>;

int trap(vector<int>& v) {
    vector u(v.size
    auto it = max_e
    inclusive_scan(
    inclusive_scan(
    return transfor
        std::plus<>
        std::minus<
}
```

Cppcon The C++ Conference 2019 cppcon.org

```
auto dangerous_teams(std::string const& s) -> bool {
    return s
         | views::group_by(std::equal_to{})
         | views::transform(ranges::distance)
         | ranges::any_of([](std::size_t s){
               return s >= 7;
           });
}
```

https://brevzin.github.io/c++/2019/08/22/ufcs-custom-extension/

212



Conor Hoekstra

Algorithm Intuition (part 2 of 2)

Video Sponsorship Provided By: ansatz

Goal

(once again)

- Get you excited about algorithms
- Learn a new algorithm
- Start to develop some **algorithm intuition**

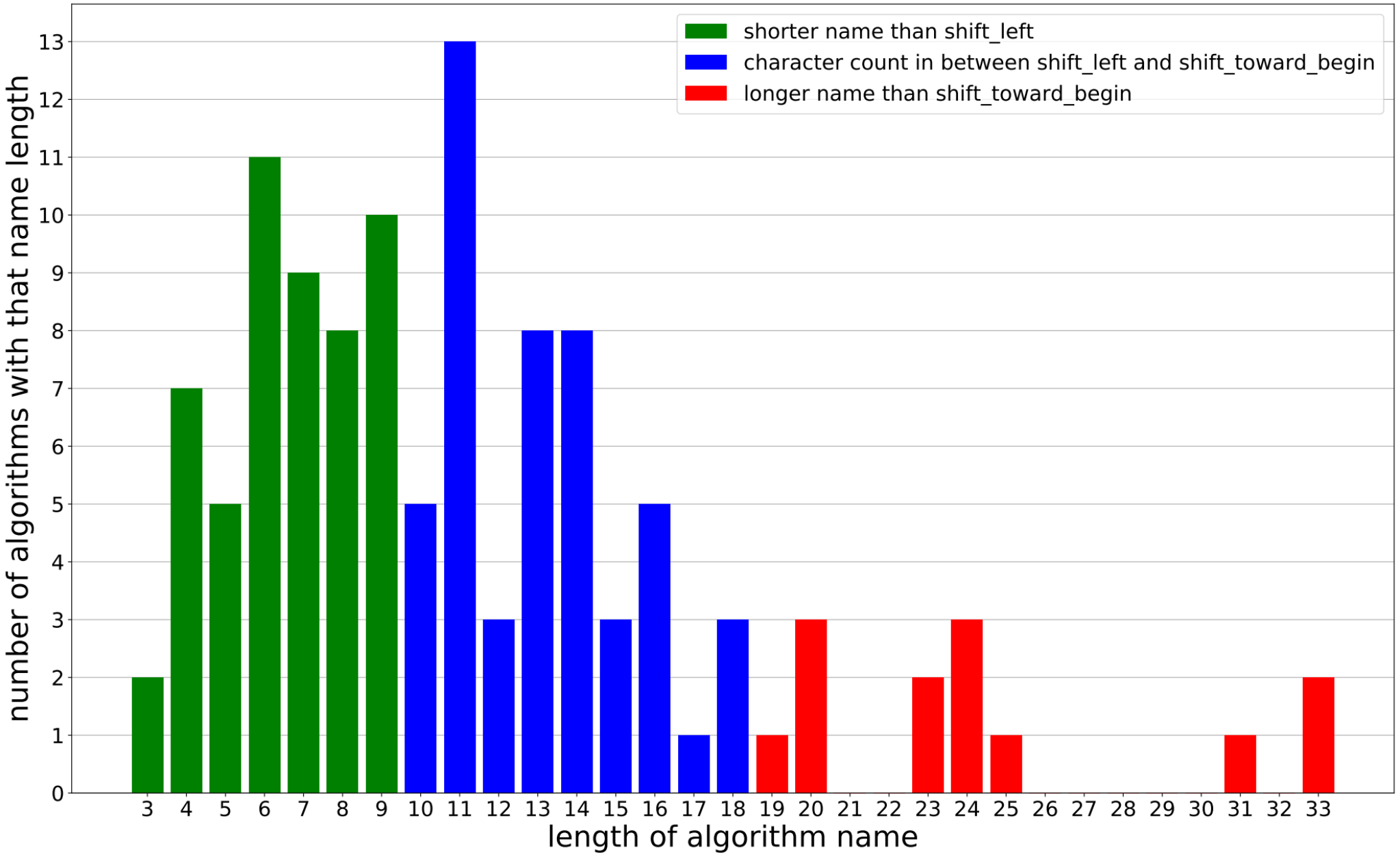
Prologue

<numeric> vs <algorithm>

Library	Pre-C++11	C++11	C++17	Grand Total
<algorithm>	66	19	3, -1	87*
<numeric>	4	1	6	11
<memory>	3	1	9	13
Grand Total	73	21	17*	111

Library	Pre-C++11	C++11	C++17	Grand Total
<algorithm>	66	19	3, -1	87*
<numeric>	4	1	6	11
<memory>	3	1	9	13
Grand Total	73	21	17*	111

Distribution of algorithm names character count



114

```
3  max
3  min

4  copy
4  fill
4  find
4  iota
4  move
4  sort
4  swap

5  clamp
5  count
5  equal
5  merge
5  qsort

6  all_of
6  any_of
6  copy_n
6  fill_n
6  minmax
6  reduce
6  remove
6  rotate
6  sample
6  search
6  unique

7  bsearch
7  copy_if
7  destroy
7  find_if
7  is_heap
7  none_of
7  replace
7  reverse
7  shuffle

8  count_if
8  find_end
8  for_each
8  generate
8  includes
8  mismatch
8  pop_head
8  search_n

9  destroy_n
9  is_sorted
9  iter_swap
9  make_heap
9  partition
9  push_heap
9  remove_if
9  set_union
9  sort_heap
9  transform

10 accumulate
10 destroy_at
10 for_each_n
10 generate_n
10 replace_if

11 equal_range
11 find_if_not
11 lower_bound
11 max_element
11 min_element
11 nth_element
11 partial_sum
11 remove_copy
11 rotate_copy
11 stable_sort
11 swap_ranges
11 unique_copy
11 upper_bound

12 partial_sort
12 replace_copy
12 reverse_copy

13 adjacent_find
13 binary_search
13 copy_backward
13 find_first_of
13 inner_product
13 inplace_merge
13 is_heap_until
13 move_backward

14 exclusive_scan
14 inclusive_scan
14 is_partitioned
14 is_permutation
14 minmax_element
14 partition_copy
14 remove_copy_if
14 set_difference

15 is_sorted_until
15 partition_point
15 replace_copy_if

16 next_permutation
16 prev_permutation
16 set_intersection
16 stable_partition
16 transform_reduce

17 partial_sort_copy

18 uninitialized_copy
18 uninitialized_fill
18 uninitialized_move

19 adjacent_difference

20 uninitialized_copy_n
20 uninitialized_fill_n
20 uninitialized_move_n

23 lexicographical_compare
23 uninitialized_construct

24 set_symmetric_difference
24 transform_exclusive_scan
24 transform_inclusive_scan

25 uninitialized_construct_n

31 uninitialized_default_construct

33 lexicographical_compare_three_way
33 uninitialized_default_construct_n
```

Algorithm Love Club

0 Tweets

Edit profile

Algorithm Love Club

@algo_love_club

For those that want to join the Algorithm Love Club! Our leaders are @SeanParent and Kate Gregory (@gregcons)!

@algo_love_club

Join the club on Twitter

#include

Chapter 1

LeetCode Level 1

LeetCode

344. Reverse String

Easy 908 581 Favorite Share

Write a function that reverses a string.

https://leetcode.com/problems/reverse-string/

```cpp
void reverseString(std::string& s) {
    std::reverse(
        std::begin(s),
        std::end(s));
}
```

thrust::reverse

344. Reverse String

Hot | Newest to Oldest | **Most Votes** | Most Posts | Recent Activity | Oldest to Newest

Simple C++ solution `cpp`

xz2210 created at: April 21, 2016 9:32 PM | Last Reply: yellomellofello August 28, 2019 8:25 PM

One Line C++ Code (with Help of STL) `cpp`

krrk2102 created at: April 22, 2016 12:53 PM | Last Reply: nicho December 12, 2018 8:53 AM

Share my C++ solution,very easy to understand `cpp` `easy-understand`

vdvvdd created at: April 23, 2016 7:21 PM | Last Reply: lidz March 16, 2019 10:48 PM

Easy C++ 2 Line Solution [beats 92%, 100%] `2 lines` `c++` `clean-code` `+ 4 more`

Pooja0406 created at: September 4, 2019 12:44 AM | No replies yet.

[C++] One-Liner `c++` `cpp`

KasraGhodsi created at: June 23, 2019 2:18 PM | No replies yet.

C++ solution without extra memory `cpp`

magiceye07 created at: May 25, 2019 2:12 PM | No replies yet.

Simple C++ code `cpp` `solution` `solution-cpp`

neelneelpurk created at: May 11, 2019 9:36 AM | No replies yet.

< Back

Simple C++ solution

★ 567 Last Edit: October 2, 2018 10:22 AM

54

```
class Solution {
public:
    string reverseString(string s) {
        int i = 0, j = s.size() - 1;
        while(i < j){
            swap(s[i++], s[j--]);
        }

        return s;
    }
};
```

< Back

Share my C++ solution,very easy to understand

★ 479 April 23, 2016 7:21 PM

5

```
class Solution {
public:
    string reverseString(string s) {
        int start = 0;
        int end = s.length() - 1;
        char ch = 0;

        for (; start < end; start++, end--)
        {
            ch = s[start];
            s[start] = s[end];
            s[end] = ch;
        }

        return s;
    }
};
```

< Back

Easy C++ 2 Line Solution [beats 92%, 100%]

★ 83 Last Edit: September 4, 2019 12:44 AM

0

Runtime: 44 ms, faster than 92.12% of C++ online submissions for Reverse String.

Memory Usage: 15.2 MB, less than 96.34% of C++ online submissions for Reverse String.

```
void reverseString(vector<char>& s) {

        for(int start=0, end = s.size()-1; start < end; start++, end--)
            swap(s[start], s[end]);
    }
```

Back

[C++] One-Liner

★ 74 June 23, 2019 2:18 PM

0

One Line C++ Code (with Help of STL)

April 22, 2016 12:53 PM

LeetCode

709. To Lower Case

Easy 341 1188 Favorite Share

Implement function ToLowerCase() that has a string parameter str, and returns the same string in lowercase.

https://leetcode.com/problems/to-lower-case/

```
std::string toLowerCase(string s) {

    std::transform(
        std::begin(s),
        std::end(s),
        std::begin(s),
        ::tolower);

    return s;
}
```

thrust::transform

LeetCode

905. Sort Array By Parity

Easy 586 62 Favorite Share

Given an array `A` of non-negative integers, return an array consisting of all the even elements of `A`, followed by all the odd elements of `A`.

You may return any answer array that satisfies this condition.

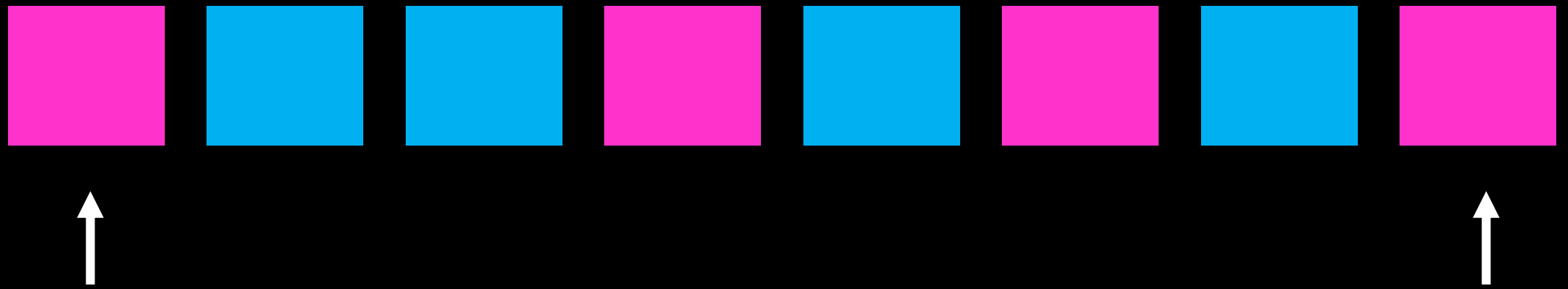
https://leetcode.com/problems/sort-array-by-parity/

```cpp
auto sortArrayByParity(vector<int>& A) {

    std::partition(
        std::begin(A),
        std::end(A),
        [](auto e) {
            return e % 2 == 0;
        });

    return A;
}
```

thrust::partition

“two finger method”

@ivan_cukic



LeetCode

283. Move Zeroes

Easy 2463 90 Favorite Share

Given an array `nums`, write a function to move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

https://leetcode.com/problems/move-zeroes/

```cpp
void moveZeroes(vector<int>& nums) {
    std::stable_partition(
        std::begin(nums),
        std::end(nums),
        [] (auto e) {
            return e != 0;
        });
}
```

thrust::stable_partition

LeetCode

973. K Closest Points to Origin

Medium 782 75 Favorite Share

We have a list of `points` on the plane. Find the `K` closest points to the origin `(0, 0)`.

(Here, the distance between two points on a plane is the Euclidean distance.)

You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in.)

https://leetcode.com/problems/k-closest-points-to-origin/

```cpp
vector<vector<int>> kClosest(
    vector<vector<int>>& points, int K) {

    std::sort(
        std::begin(points),
        std::end(points),
        [](auto const& a, auto const& b) {
            return sqrt(a[0] * a[0] + a[1] * a[1]) <
                   sqrt(b[0] * b[0] + b[1] * b[1]);
        });

    return vector(
        std::begin(points),
        std::begin(points) + K);
}
```

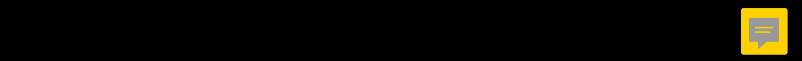
thrust::sort

```cpp
vector<vector<int>> kClosest(
    vector<vector<int>>& points, int K) {

    std::sort(
        std::begin(points),
        std::end(points),
        [](auto const& a, auto const& b) {
            return (a[0] * a[0] + a[1] * a[1]) <
                   (b[0] * b[0] + b[1] * b[1]);
        });

    return vector(
        std::begin(points),
        std::begin(points) + K);
}
```

thrust::sort

```cpp
vector<vector<int>> kClosest(
    vector<vector<int>>& points, int K) {

    std::partial_sort(
        std::begin(points),
        std::begin(points) + K,
        std::end(points),
        [](auto const& a, auto const& b) {
            return (a[0] * a[0] + a[1] * a[1]) <
                   (b[0] * b[0] + b[1] * b[1]);
        });

    return vector(
        std::begin(points),
        std::begin(points) + K);
}
```

```
vector<vector<int>> kClosest(
    vector<vector<int>>& points, int K) {

    std::nth_element(
        std::begin(points),
        std::begin(points) + K,
        std::end(points),
        [](auto const& a, auto const& b) {
            return (a[0] * a[0] + a[1] * a[1]) <
                   (b[0] * b[0] + b[1] * b[1]);
        });

    return vector(
        std::begin(points),
        std::begin(points) + K);
}
```



```
std::sort
↓
std::partial_sort
↓
std::nth_element
```

Kate Gregory

Naming is Hard: Let's Do Better

An <algorithm> story

- sort
- partial_sort

`partial_sort_copy` `top_n`

`nth_element_copy` `top_n`

`set` `unordered_set`

`nth_element_copy` `top_n`

`partial_sort_copy` `top_n_sorted`

reverse

stable_partition

transform

nth_element

for_each

sort

partition

partial_sort

std::swap ->
std::move

reverse

stable_partition

transform

nth_element

for_each

sort

partition

partial_sort

std::swap ->
std::move

reverse

stable_partition

transform

nth_element

for_each

sort

partition

partial_sort

`move`

`swap`

`stable_partition`

Chapter 2

LeetCode Level 2

LeetCode

977. Squares of a Sorted Array

Easy 543 57 Favorite Share

Given an array of integers `A` sorted in non-decreasing order, return an array of the squares of each number, also in sorted non-decreasing order.

Note:

1. `1 <= A.length <= 10000`
2. `-10000 <= A[i] <= 10000`
3. `A` is sorted in non-decreasing order.

https://leetcode.com/problems/squares-of-a-sorted-array/

```cpp
vector<int> sortedSquares(vector<int>& A) {

    std::transform(
        std::begin(A),
        std::end(A),
        std::begin(A),
        [] (auto e) {
            return e * e;
        });

    std::sort(
        std::begin(A),
        std::end(A));

    return A;
}
```

thrust::transform thrust::sort

```cpp
vector<int> sortedSquares(vector<int>& A) {
    transform(begin(A), end(A), begin(A),
        [] (auto e) { return e * e; });
    sort(begin(A), end(A));
    return A;
}
```

thrust::transform thrust::sort

20
Ranges

https://godbolt.org/z/4wQbNY

```
namespace rv = ranges::views;
namespace ra = ranges::actions;

auto sortedSquares(std::vector<int> A) {
    return A
        | rv::all
        | ra::transform([] (int e) { return e * e; })
        | ra::sort
        | ranges::to<std::vector<int>>;
}
```

thrust::transform thrust::sort

```
[1,3,2,5,4]
[1,9,4,25,16]
[1,4,9,16,25]
```

```haskell
square :: Int -> Int
square n = n * n

sortedSquares :: [Int] -> [Int]
sortedSquares = sort . map square
```

LeetCode *

961. Element Repeated Once

Easy 271 167 Favorite Share

In an array A of size N, there are $N - 1$ unique values (therefore exactly one of those values appears twice).

Return the value that is repeated 2 times.

https://leetcode.com/problems/n-repeated-element-in-size-2n-array/

```cpp
int repeatedNTimes(vector<int>& A) {

    std::sort(
        std::begin(A),
        std::end(A));

    return *std::adjacent_find(
        std::cbegin(A),
        std::cend(A));
}
```

thrust::sort

LeetCode *

33. Search in Rotated Sorted Array

Medium 3093 368 Favorite Share

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., `[0,1,2,4,5,6,7]` might become `[4,5,6,7,0,1,2]`).

If found in the array return `true`, otherwise return `false`.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

https://leetcode.com/problems/search-in-rotated-sorted-array

```cpp
bool search(vector<int>& nums, int target) {
    auto f = std::cbegin(nums);
    auto l = std::cend(nums);
    auto p = std::is_sorted_until(f, l);
    return target >= *f ? std::binary_search(f, p, target)
                        : std::binary_search(p, l, target);
}
```

thrust::is_sorted_until thrust::binary_search

```cpp
bool search(vector<int>& nums, int target) {
    auto f = std::cbegin(nums);
    auto l = std::cend(nums);
    auto p = std::partition_point(f, l,
        [x = *f] (auto const& e) {
            return e >= x;
        });
    return target >= *f ? std::binary_search(f, p, target)
                        : std::binary_search(p, l, target);
}
```

thrust::partition_point thrust::binary_search

```cpp
bool search(vector<int>& nums, int target) {
    auto f = std::cbegin(nums);
    auto l = std::cend(nums);
    auto p = std::partition_point(f, l,
        std::bind(std::greater_equal{}, _1, *f));
    return target >= *f ? std::binary_search(f, p, target)
                        : std::binary_search(p, l, target);
}
```

thrust::partition_point thrust::binary_search

LeetCode

42. Trapping Rain Water

Hard 3387 61 Favorite Share

Given *n* non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped. **Thanks Marcos** for contributing this image!

https://leetcode.com/problems/trapping-rain-water/

```cpp
template<class T>
using rev = reverse_iterator<T>;

int trap(vector<int>& v) {
    vector u(v.size(), 0);
    auto it = max_element(begin(v), end(v));
    inclusive_scan(begin(v),  next(it), begin(u),  ufo::max{});
    inclusive_scan(rbegin(v), rev(it),  rbegin(u), ufo::max{});
    return transform_reduce(cbegin(u), cend(u), cbegin(v), 0,
        std::plus<>(),
        std::minus<>());
}
```

thrust::inclusive_scan thrust::transform_reduce thrust::max_element

Chapter 4

LeetCode Level 3

Quickly mention

find_if → lower_bound

find_if → upper_bound

What about that bad loop?

```
// Next, check if the panel has moved to the other side of another panel.

auto p = find_if(begin(expanded_panels_), end(expanded_panels_),
  [&](const ref_ptr<Panel>& e){ return center_x <= e->cur_panel_center(); });

// Fix this code - panel is the panel found above.

if (panel != fixed_panel) {
   // If it has, then we reorder the panels.
   ref_ptr<Panel> ref = expanded_panels_[fixed_index];
   expanded_panels_.erase(expanded_panels_.begin() + fixed_index);
   expanded_panels_.insert(expanded_panels_.begin() + i, ref);
}
```

Quickly mention 

find_if → lower_bound

find_if → upper_bound

LeetCode

917. Reverse Only Letters

Easy 321 30 Favorite Share

Given a string `S`, return the "reversed" string where all characters that are not a letter stay in the same place, and all letters reverse their positions.

```
1ab2cd -> 1dc2ba
```

https://leetcode.com/problems/reverse-only-letters/

How many STL algorithms will it take to solve?

1

2

3

4

```
string reverseOnlyLetters(string S) {

    std::string letters;

    std::copy_if(
        std::begin(S),
        std::end(S),
        std::back_inserter(letters),
        ::isalpha);

    std::reverse(
        std::begin(letters),
        std::end(letters));

    std::transform(
        std::begin(S),
        std::end(S),
        std::begin(S),
        [c = std::cbegin(letters)] (auto e) mutable {
            return isalpha(e) ? *c++ : e;
        });

    return S;
}
```

transform reverse copy_if

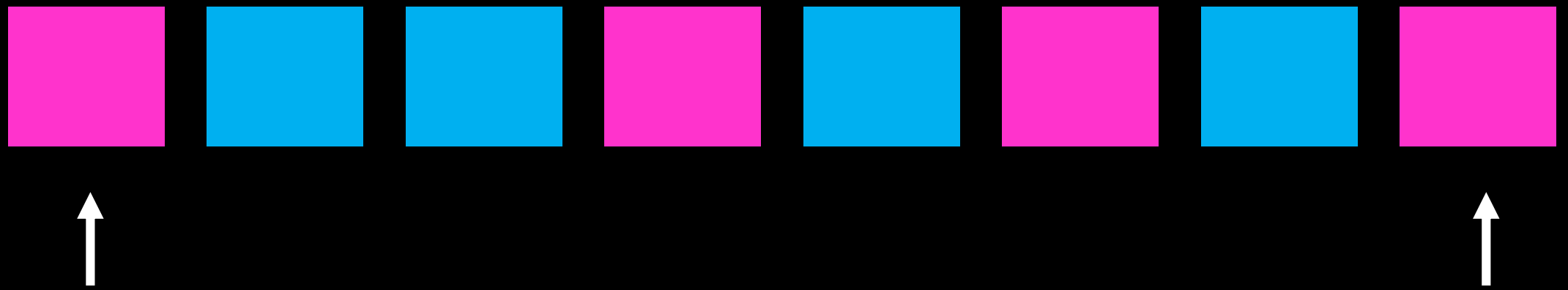
```
string reverseOnlyLetters(string S) {

    std::string letters;

    std::copy_if(
        std::crbegin(S),
        std::crend(S),
        std::back_inserter(letters),
        ::isalpha);

    std::transform(
        std::begin(S),
        std::end(S),
        std::begin(S),
        [c = std::cbegin(letters)] (auto e) mutable {
            return isalpha(e) ? *c++ : e;
        });

    return S;
}
```

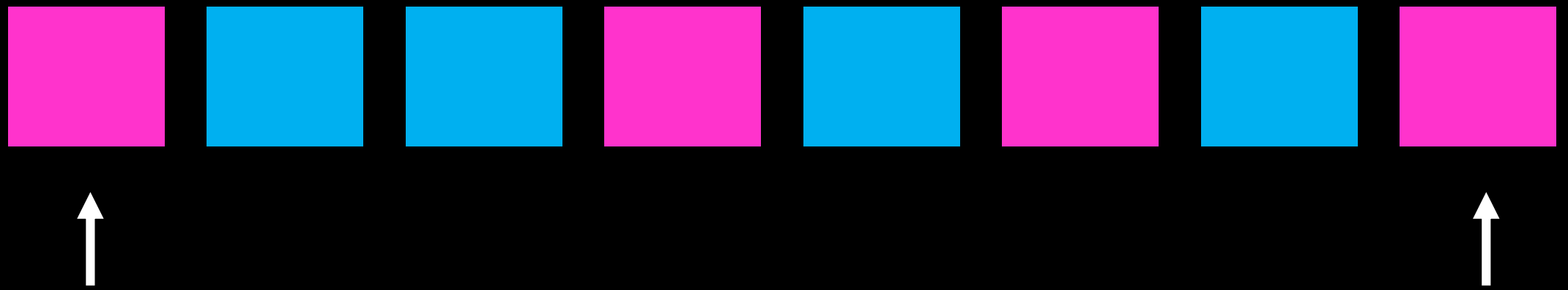
transform copy_if

```
string reverseOnlyLetters(string S) {

    std::reverse_if(
        std::begin(S),
        std::end(S),
        ::isalpha);

    return S;
}
```

```cpp
template <class B, class P>
void reverse_if(B first, B last, P pred) {
    while (first != last) {
        while (!pred(*first)) ++first;
        while (!pred(*last)) --last;
        iter_swap(first++, last++);
    }
}
```

```cpp
template <class B, class P>
void reverse_if(B f, B l, P pred) {
    while (f != l) {
        while (!pred(*f)) { ++f; if (f == l) goto done; }
        while (!pred(*l)) { --l; if (f == l) goto done; }
        iter_swap(f++, l);
        if (f == l) break;
        ++l;
    }
    done: ;
}
```





```cpp
std::string reverseOnlyLetters(std::string S) {

    std::partition(
        std::begin(S),
        std::end(S),
        [b = true] (auto e) mutable {
            if (isalpha(e)) b = !b;
            return b;
        });

    return S;
}
```

thrust::partition

How many STL algorithms will it take to solve?

1

2

3

4

Chapter 5

The

Sean Parent

Game

```
std::is_sorted
```

↑

```
std::is_sorted_until
```

```
template<class _FwdIt,
    class _Pr>
    _NODISCARD inline bool is_sorted(_FwdIt _First, _FwdIt _Last, _Pr _Pred)
    {   // test if range is ordered by predicate
    _Adl_verify_range(_First, _Last);
    const auto _UFirst = _Get_unwrapped(_First);
    const auto _ULast = _Get_unwrapped(_Last);
    return (_STD is_sorted_until(_UFirst, _ULast, _Pass_fn(_Pred)) == _ULast);
    }
```

thrust::is_sorted thrust::is_sorted_until

```
std::is_sorted
      ↑
std::is_sorted_until
      ↑
std::adjacent_find
```

```
template<class _FwdIt,
    class _Pr>
    _NODISCARD inline _FwdIt is_sorted_until(const _FwdIt _First, _FwdIt _Last, _Pr _Pred)
    {   // find extent of range that is ordered by predicate
    _Adl_verify_range(_First, _Last);
    auto _UFirst = _Get_unwrapped(_First);
    auto _ULast = _Get_unwrapped(_Last);
    if (_UFirst != _ULast)
        {
        for (auto _UNext = _UFirst; ++_UNext != _ULast; ++_UFirst)
            {
            if (_DEBUG_LT_PRED(_Pred, *_UNext, *_UFirst))
                {
                _ULast = _UNext;
                break;
                }
            }
        }

    _Seek_wrapped(_Last, _ULast);
    return (_Last);
    }
```

```
template<class _FwdIt,
    class _Pr>
    _NODISCARD inline _FwdIt adjacent_find(const _FwdIt _First, _FwdIt _Last, _Pr _Pred)
    {   // find first satisfying _Pred with successor
    _Adl_verify_range(_First, _Last);
    auto _UFirst = _Get_unwrapped(_First);
    auto _ULast = _Get_unwrapped(_Last);
    if (_UFirst != _ULast)
        {
        for (auto _UNext = _UFirst; ++_UNext != _ULast; _UFirst = _UNext)
            {
            if (_Pred(*_UFirst, *_UNext))
                {
                _ULast = _UFirst;
                break;
                }
            }
        }

    _Seek_wrapped(_Last, _ULast);
    return (_Last);
    }
```

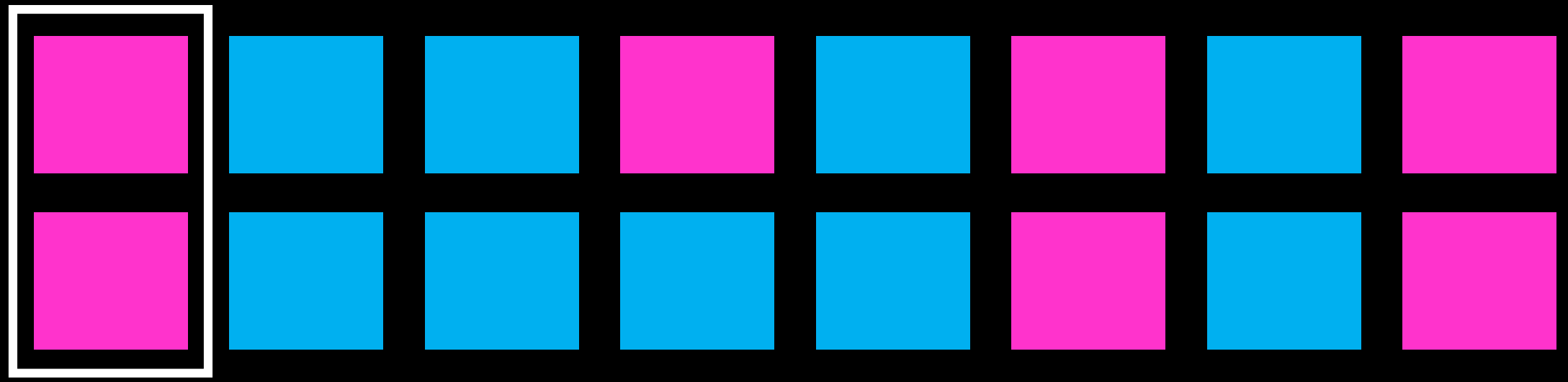
```
template<class _FwdIt,
        class _Pr>
        _NODISCARD inline _FwdIt adjacent_find(const
_FwdIt _First, _FwdIt _Last, _Pr _Pred)
        {       // find first satisfying _Pred with successor
        _Adl_verify_range(_First, _Last);
        auto _UFirst = _Get_unwrapped(_First);
        auto _ULast = _Get_unwrapped(_Last);
        if (_UFirst != _ULast)
                {
                for (auto _UNext = _UFirst; ++_UNext != _ULast; _UFirst = _UNext)
                        {
                        if (_Pred(*_UFirst, *_UNext))
                                {
                                _ULast = _UFirst;
                                break;
                                }
                        }
                }

        _Seek_wrapped(_Last, _ULast);
        return (_Last);
        }
```

```
template<class _FwdIt,
        class _Pr>
        _NODISCARD inline _FwdIt is_sorted_until(const
_FwdIt _First, _FwdIt _Last, _Pr _Pred)
        {       // find extent of range that is ordered by predicate
        _Adl_verify_range(_First, _Last);
        auto _UFirst = _Get_unwrapped(_First);
        auto _ULast = _Get_unwrapped(_Last);
        if (_UFirst != _ULast)
                {
                for (auto _UNext = _UFirst; ++_UNext != _ULast; ++_UFirst)
                        {
                        if (_DEBUG_LT_PRED(_Pred, *_UNext, *_UFirst))
                                {
                                _ULast = _UNext;
                                break;
                                }
                        }
                }

        _Seek_wrapped(_Last, _ULast);
        return (_Last);
        }
```

```cpp
#include <boost/hana/functional/flip.hpp>

template<class F, class P>
F is_sorted_until(F first, F last, P pred) {
    return std::adjacent_find(
        first,
        last,
        boost::hana::flip(pred));
}
```

```
std::is_sorted
      ↑
std::is_sorted_until
      ↑
std::adjacent_find
```

```
std::is_sorted
      ↑
std::is_sorted_until
      ↑
std::adjacent_find
      ↑
std::mismatch
```



```cpp
template <class I, class P>
I adjacent_find(I first, I last, P pred) {
    return std::mismatch(
        first,
        std::prev(last),
        std::next(first),
        [pred] (auto const& a, auto const& b) {
            return !pred(a, b);
        }).first;
}
```

```cpp
template <class I, class P>
I adjacent_find(I first, I last, P pred) {
    return std::mismatch(
        first,
        std::prev(last),
        std::next(first),
        std::not_fn(pred)).first;
}
```

zip_find

transform_find

zip_find

```
template <class F, class F2, class P>
auto mismatch(F first, F last, F2 first2, P pred) {
    std::zip_find(
        first,
        last,
        first2,
        std::not_fn(pred));
}
```

adjacent_find

```
template <class I, class P>
I adjacent_find(I first, I last, P pred) {
    return zip_find(
        first,
        std::prev(last),
        std::next(first),
        pred).first;
}
```

```
mismatch -> zip_find
transform -> zip_with
inner_product -> zip_reduce
transform_reduce -> zip_reduce

equal -> zip_reduce(logical_and, equal_to)
```

```
template<class _InIt1,
    class _InIt2,
    class _Pr> inline
    bool _Equal_unchecked1(_InIt1 _First1, const _InIt1 _Last1,
        _InIt2 _First2, _Pr _Pred, false_type)
    {   // compare [_First1, _Last1) to [_First2, ...) using _Pred,
    for (; _First1 != _Last1; ++_First1, (void)++_First2)
        {
        if (!_Pred(*_First1, *_First2))
            {
            return (false);
            }
        }

    return (true);
```

```cpp
template <class InputIterator1,
          class InputIterator2,
          class BinaryPredicate>
inline bool equal(InputIterator1 first1,
                  InputIterator1 last1,
                  InputIterator2 first2,
                  BinaryPredicate binary_pred) {
    return mismatch(
        first1,
        last1,
        first2,
        binary_pred).first == last1;
}
```

http://stepanovpapers.com/butler.hpl.hp/stl/stl/ALGOBASE.H

```
template <class InputIterator1,
          class InputIterator2,
          class BinaryPredicate>
inline bool equal(InputIterator1 first1,
                  InputIterator1 last1,
                  InputIterator2 first2,
                  BinaryPredicate binary_pred) {
    return mismatch(
        first1,
        last1,
        first2,
        binary_pred).first == last1;
}
```

http://stepanovpapers.com/butler.hpl.hp/stl/stl/ALGOBASE.H

```
template <class InputIterator1, class InputIterator2, class BinaryPredicate>
inline bool equal(InputIterator1 first1, InputIterator1 last1,
                  InputIterator2 first2, BinaryPredicate binary_pred) {
    return mismatch(first1, last1, first2, binary_pred).first == last1;
}
```

```
mismatch -> zip_find
transform -> zip_with
inner_product -> zip_reduce
transform_reduce -> zip_reduce
```

```
equal -> zip_reduce(logical_and, equal_to)
    equal -> mismatch() == end()
```

includes, merge, search, set_*, swap_range

mismatch -> zip_find

adjacent_difference -> adjacent_transform

reverse
transform
partition
stable_partition
sort
partial_sort
nth_element
remove_if

unique
for_each
copy
adjacent_find
partial_sort_copy
partition_point
binary_search
prev_permutation

is_sorted
is_sorted_until
find_if
equal_range
lower_bound
copy_if
mismatch
equal

Conclusion

Chapter 1:

sort | partial_sort | nth_element

remove_if | stable_partition

Chapter 2: is_sorted_until | partition_point (… is log n)

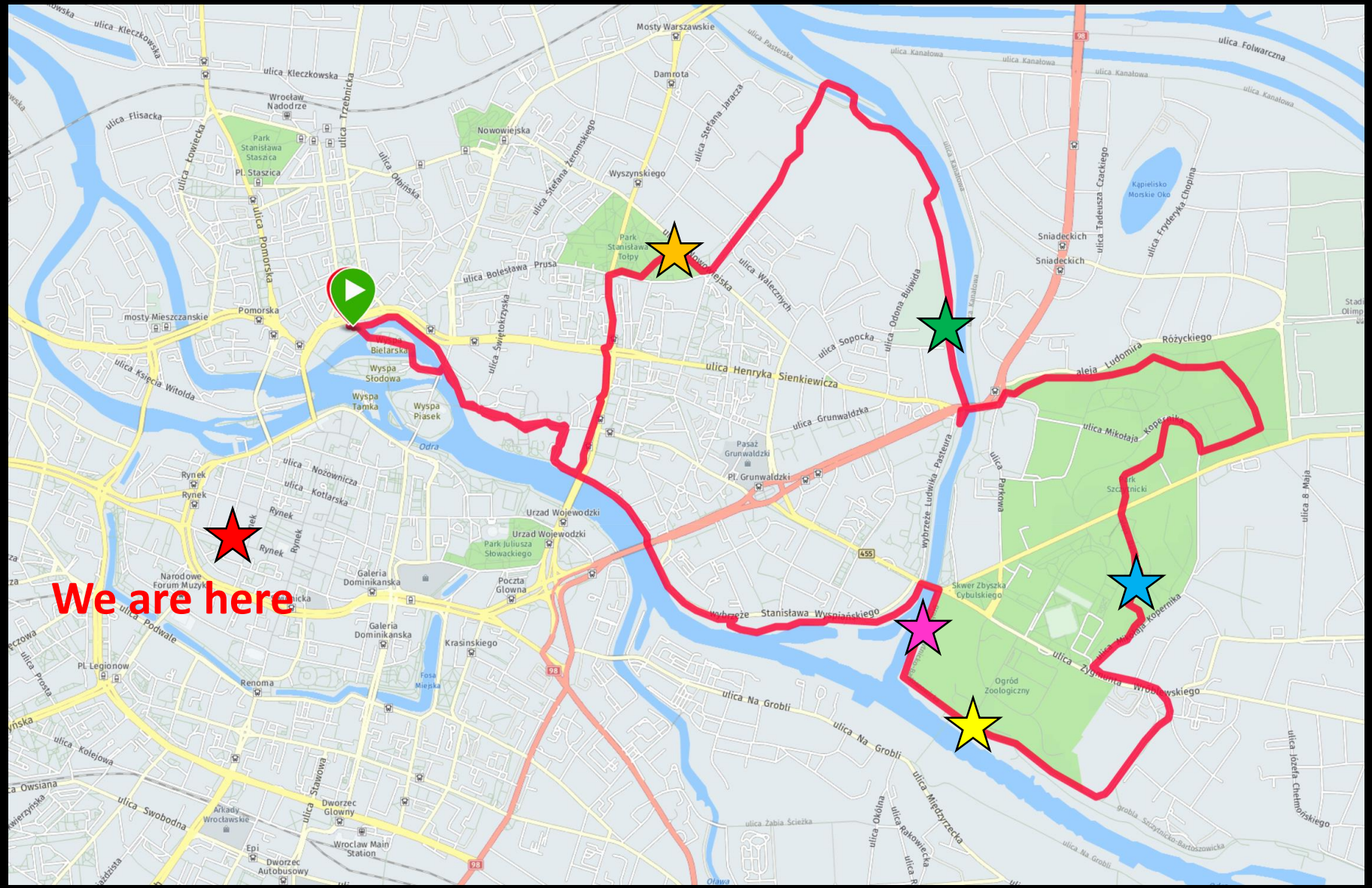
Chapter 3: transform is both zipWith and map

Chapter 4:

find_if | lower_bound / upper_bound

partition | reverse_if

Chapter 5: is_sorted | is_sorted_until | adjacent_find | mismatch


We are here
Mosty Warszawskie
Damrota
ulica Folwarczna
ulica Kanałowa
ulica Kleczkowska
Wrocław Nadodrze
ulica Flisacka
ulica Trzebnicka
Park Stanisława Staszica
Pl. Staszica
Nowowiejska
ulica Stefana Jaracza
ulica Stefana Żeromskiego
Wyszynskiego
ulica Łowiecka
ulica Pomorska
ulica Obornicka
Park Stanisława Tołpy
ulica Bolesława Prusa
ulica Walecznych
ulica Sopocka
ulica Odona Bujwida
Sniadeckich
ulica Tadeusza Czackiego
ulica Fryderyka Chopina
Kąpielisko Morskie Oko
mosty Mieszczanskie
Pomorska
ulica Księcia Witolda
Bielarska
Wyspa Słodowa
Wyspa Tamka
Wyspa Piasek
ulica Świętokrzyska
ulica Henryka Sienkiewicza
ulica Grunwaldzka
Pasaż Grunwaldzki
Pl. Grunwaldzki
aleja Ludomira Różyckiego
ulica Mikołaja Kopernika
Odra
ulica Nożownicza
ulica Kotlarska
Rynek
Urzad Wojewodzki
Park Juliusza Słowackiego
wybrzeże Ludwika Pasteura
ulica Parkowa
455
Narodowe Forum Muzyki
Galeria Dominikanska
Poczta Glowna
Skwer Zbyszka Cybulskiego
ulica Podwale
wybrzeże Stanisława Wyspiańskiego
ulica Mikołaja Kopernika
PL. Legionow
Krasinskiego
Renoma
Fosa Miejska
98
Ogród Zoologiczny
ulica Na Grobli
ulica Kolejowa
ulica Prosta
ulica Swobodna
Arkady Wrocławskie
ulica Sławkowa
Dworzec Glowny
Epi
Dworzec Autobusowy
Wroclaw Main Station
ulica Międzyrzecka
ulica Rakowiecka
ulica Żabia Ścieżka
ulica Józefa Chełmońskiego
grobla Szczytnicko-Bartoszowicka
Oława

Thank you!

https://github.com/codereport/Talks/

Conor Hoekstra

code_report

codereport

Questions?

https://github.com/codereport/Talks/

Conor Hoekstra

code_report

codereport

Bonus Slides

LeetCode

1051. Height Checker

Easy 90 708 Favorite Share

Students are asked to stand in non-decreasing order of heights for an annual photo.

Return the minimum number of students not standing in the right positions. (This is the number of students that must move in order for all students to be standing in non-decreasing order of height.)

https://leetcode.com/problems/height-checker/

```cpp
int heightChecker(vector<int>& heights) {

    std::vector<int> h_copy(heights.size());

    std::partial_sort_copy(
        std::begin(heights),
        std::end(heights),
        std::begin(h_copy),
        std::end(h_copy));

    return std::inner_product(
        std::begin(heights),
        std::end(heights),
        std::begin(h_copy),
        0,
        std::plus{},
        std::not_equal_to{});
}
```

thrust::inner_product

LeetCode

1053. Previous Permutation With One Swap

Medium 62 161 Favorite Share

Given an array `A` of positive integers (not necessarily distinct), return the lexicographically largest permutation that is smaller than `A`, that can be **made with one swap** (A *swap* exchanges the positions of two numbers `A[i]` and `A[j]`). If it cannot be done, then return the same array.

```
[3,2,1]     ->   [3,1,2]
[1,1,5]     ->   [1,1,5]
[1,9,4,6,7] -> [1,7,4,6,9]
```

https://leetcode.com/problems/previous-permutation-with-one-swap/

How many STL algorithms will it take to solve?

1

2

3

4

[1,1,5]

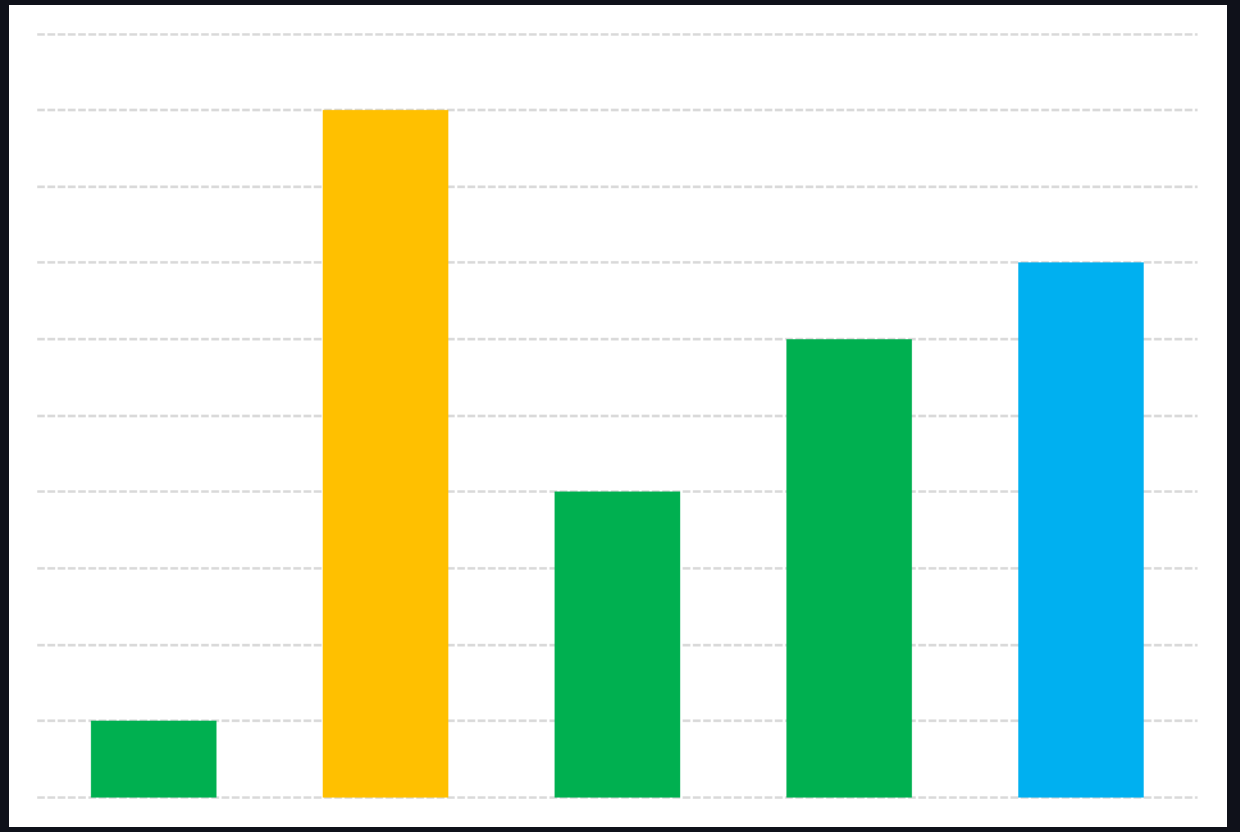
```
vector<int> prevPermOpt1(vector<int>& A) {
    prev_permutation(A.begin(), A.end());
    return A;
}
```

[1,9,4,6,7]

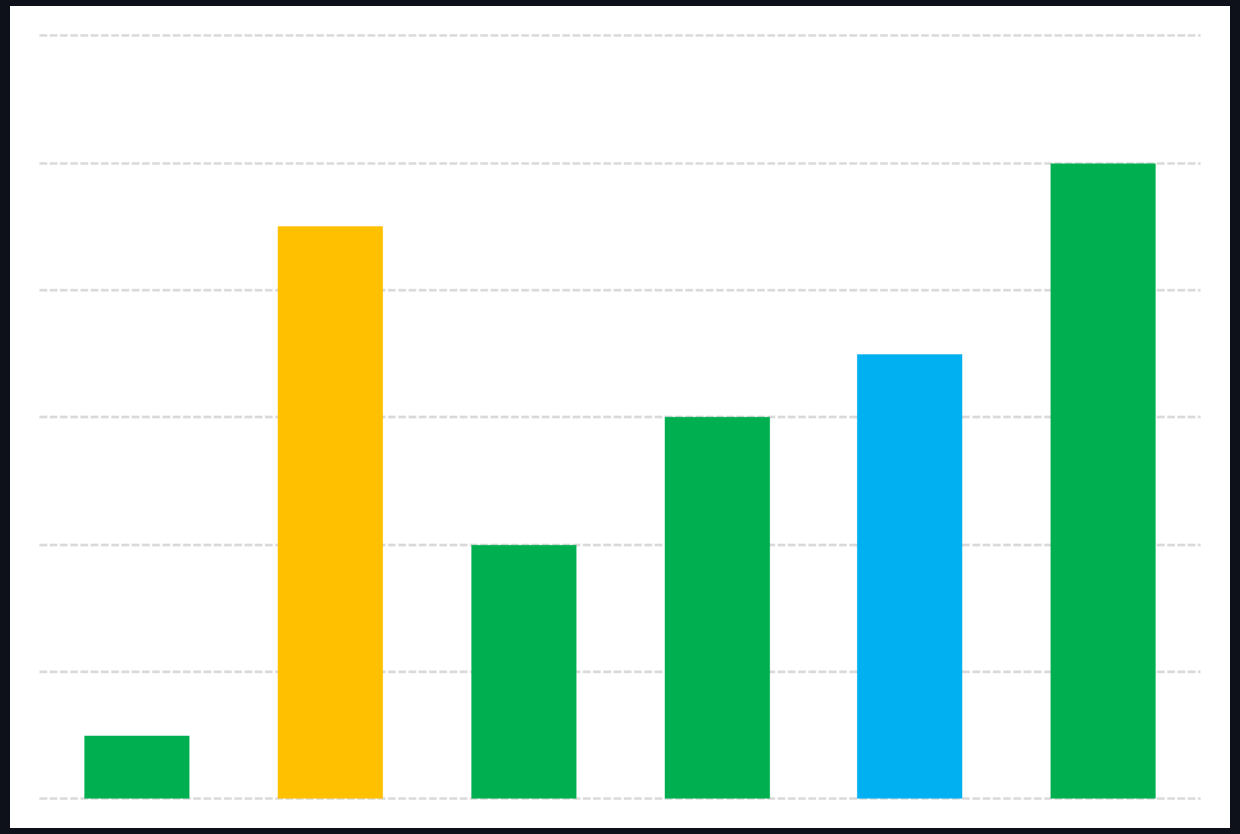
[1,7,4,6,9]

```
vector<int> prevPermOpt1(vector<int>& A) {
    if (!is_sorted(A.begin(), A.end()))
        prev_permutation(A.begin(), A.end());
    return A;
}
```

thrust::is_sorted



[1,9,4,6,7]



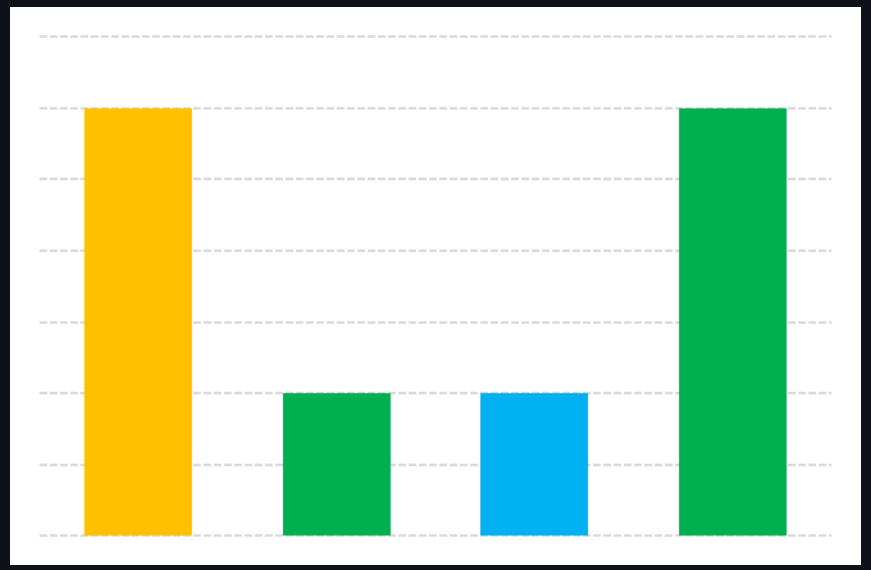
[1,9,4,6,7,10]

[3,1,1,3]

```
vector<int> prevPermOpt1(vector<int>& A) {
    if (is_sorted(A.begin(), A.end())) return A;
    auto i = is_sorted_until(A.rbegin(), A.rend(), greater<>());
    auto j = find_if(A.rbegin(), i, [&](auto e) { return e < *i; });
    iter_swap(i, j);
    return A;
}
```

thrust::is_sorted thrust::is_sorted_until thrust::find_if

1,1,3,3
1,3,1,3
1,3,3,1
3,1,1,3
3,1,3,1
3,3,1,1



[3,1,1,3]

1,1,3,3

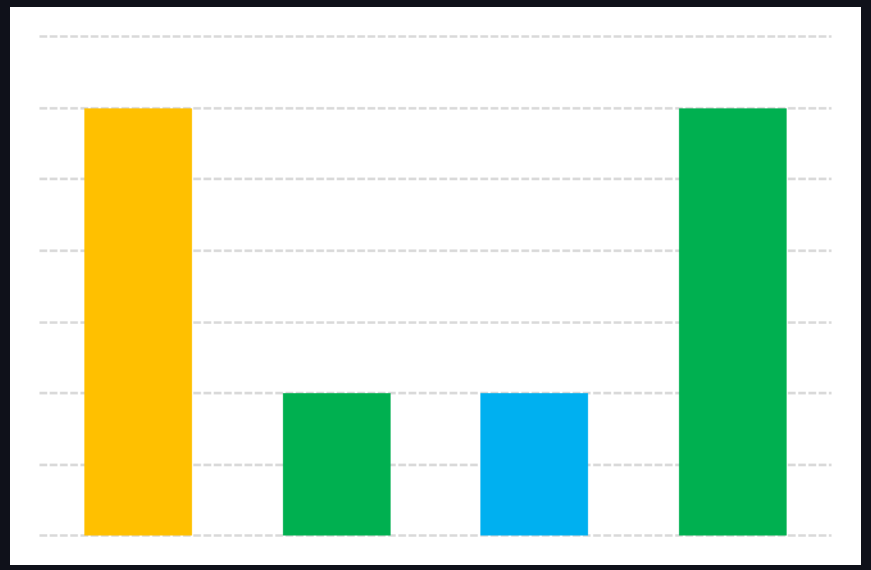
1,3,1,3

1,3,3,1

3,1,1,3

3,1,3,1

3,3,1,1



[3,1,1,3]

1. is_sort_until from back -> i
2. find_if < i from back -> x
3. find “range of all values equal to x”
4. swap furthest x (from back) with i

1,1,3,3

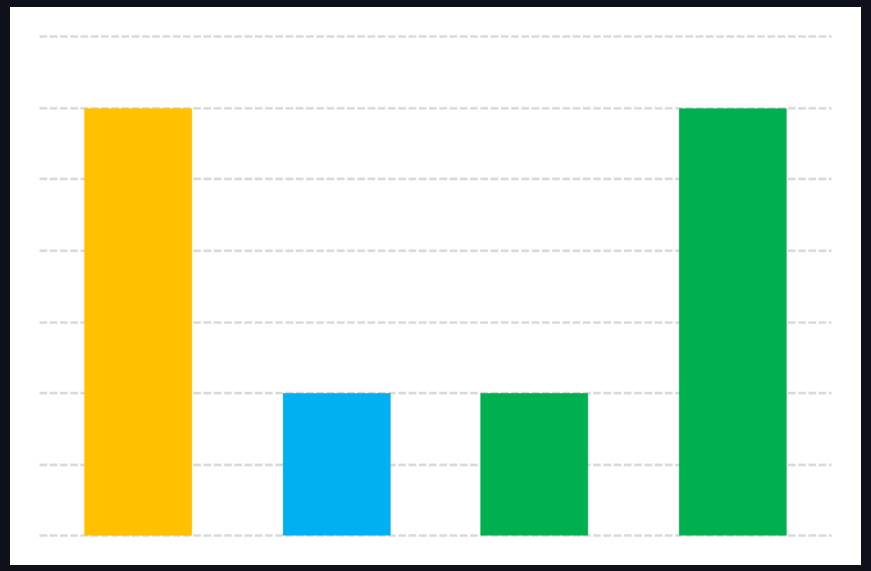
1,3,1,3

1,3,3,1

3,1,1,3

3,1,3,1

3,3,1,1



[3,1,1,3]

```cpp
vector<int> prevPermOpt1(vector<int>& A) {
    if (is_sorted(A.begin(), A.end())) return A;
    auto i = is_sorted_until(A.rbegin(), A.rend(), greater<>());
    auto j = find_if(A.rbegin(), i, [&](auto e) { return e < *i; });
    auto p = equal_range(i.base(), A.end(), *j);
    iter_swap(i, p.first);
    return A;
}
```

thrust::is_sorted thrust::is_sorted_until thrust::find_if thrust::equal_range

```cpp
vector<int> prevPermOpt1(vector<int>& A) {
    auto i = is_sorted_until(A.rbegin(), A.rend(), greater<>());
    if (i != A.rend()) {
        auto j = find_if(A.rbegin(), i, [&](auto e) { return e < *i; });
        auto p = equal_range(i.base(), A.end(), *j);
        iter_swap(i, p.first);
    }
    return A;
}
```

thrust::is_sorted_until thrust::find_if thrust::equal_range

What about that bad loop?

```
// Next, check if the panel has moved to the other side of another panel.

auto p = find_if(begin(expanded_panels_), end(expanded_panels_),
  [&](const ref_ptr<Panel>& e){ return center_x <= e->cur_panel_center(); });

// Fix this code - panel is the panel found above.

if (panel != fixed_panel) {
   // If it has, then we reorder the panels.
   ref_ptr<Panel> ref = expanded_panels_[fixed_index];
   expanded_panels_.erase(expanded_panels_.begin() + fixed_index);
   expanded_panels_.insert(expanded_panels_.begin() + i, ref);
}
```

```cpp
template<typename F, typename T>
F first_less_than(F f, F l, T val) {
    auto it = std::lower_bound(f, l, val);
    return it == f ? l : --it;
}

vector<int> prevPermOpt1(vector<int>& A) {
    auto i = is_sorted_until(A.rbegin(), A.rend(), greater<>());
    if (i != A.rend()) {
        auto j = first_less_than(i.base(), A.end(), *i);
        auto p = equal_range(i.base(), A.end(), *j);
        iter_swap(i, p.first);
    }
    return A;
}
```

thrust::is_sorted_until thrust::lower_bound thrust::equal_range

How many STL algorithms will it take to solve?

1

2

3

4