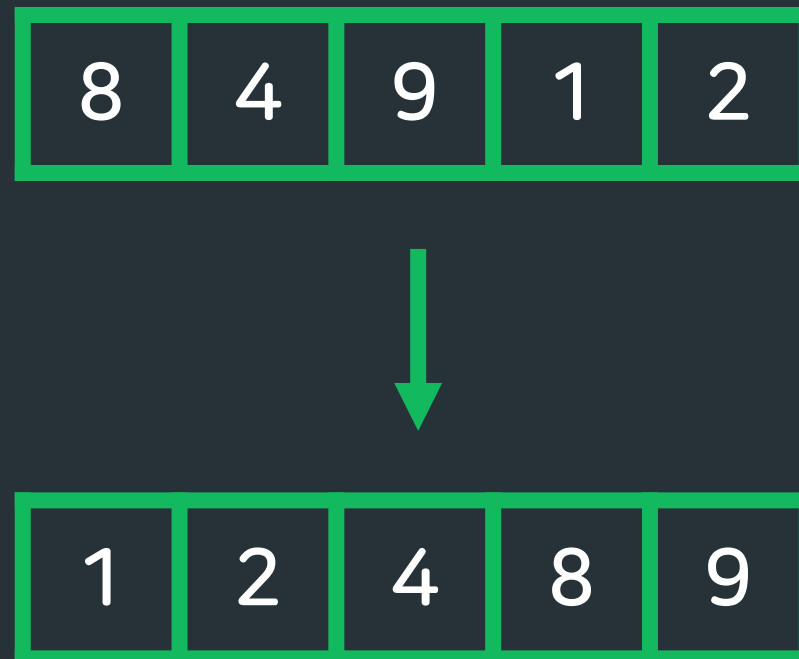


알튜비튜

정렬

배열의 원소를 정렬하는 방법에는 여러가지가 있습니다.
오늘은 그 중에서 시간 복잡도 $O(n^2)$ 의 버블 정렬과 $O(n \log n)$ 의 병합 정렬을 알아본 뒤,
STL의 sort 알고리즘에 대해 배웁니다.



대표적인 정렬 알고리즘



$O(n^2)$

Insert sort
Selection sort
Bubble sort

$O(n \log n)$

Quick sort
Merge sort
Heap sort

대표적인 정렬 알고리즘

$O(n^2)$

Insert sort
Selection sort
Bubble sort

$O(n \log n)$

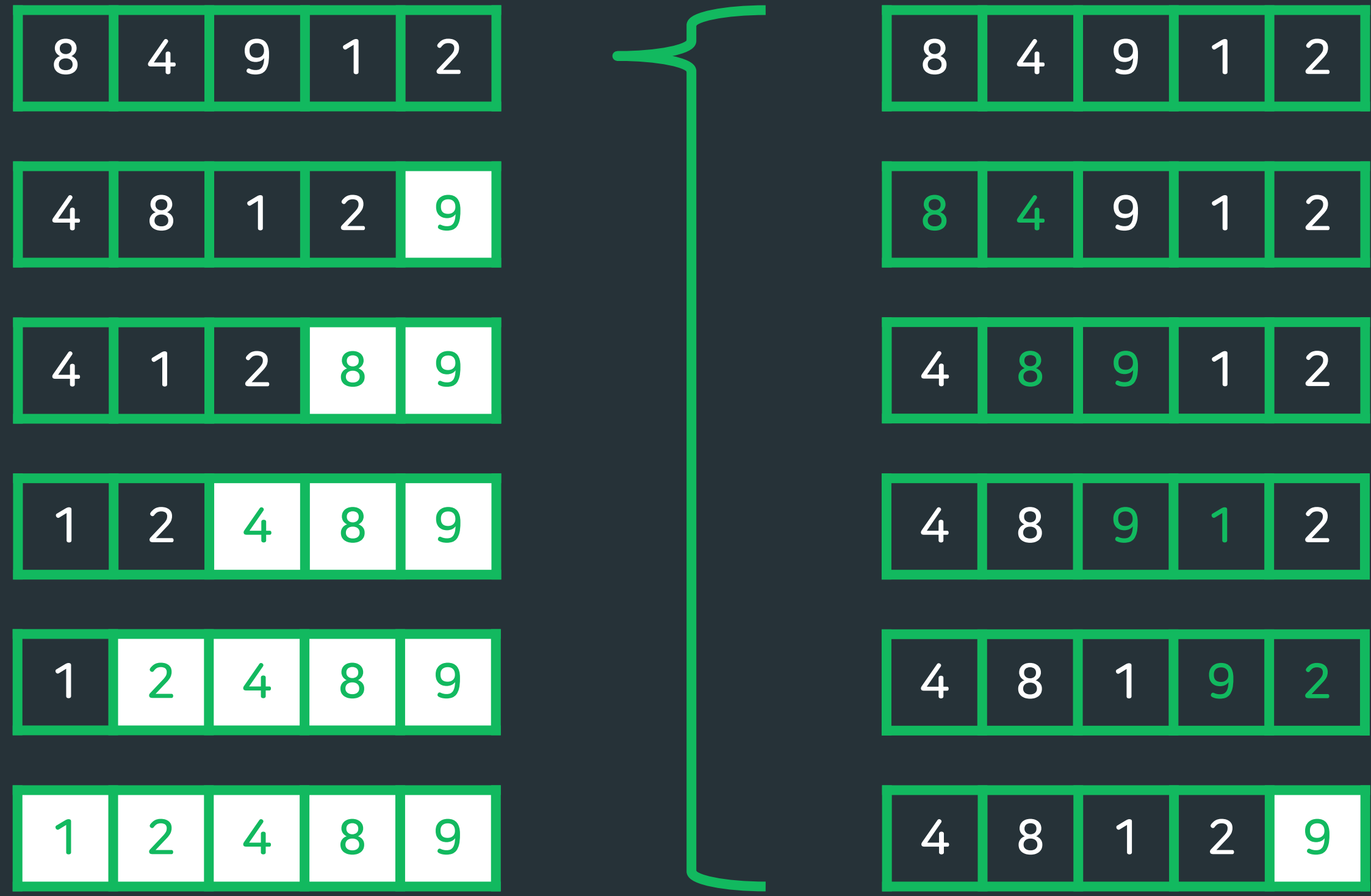
Quick sort
Merge sort
Heap sort

오름차순 정렬이라고 가정하고 설명합니다!

Bubble sort

- 인접한 두 원소를 비교
- (왼쪽 원소) > (오른쪽 원소)라면 swap!
- 가장 큰 원소부터 오른쪽에 정렬됨

버블 정렬



/<> 2750번 : 수 정렬하기 - Bronze 1

문제

- N개의 수를 오름차순 정렬

제한 사항

- N의 범위는 $1 \leq N \leq 1,000$
- 각각의 수 k는 $-1,000 \leq k \leq 1,000$ 이며 중복되지 않음

/<> 2750번 : 수 정렬하기 - Bronze 1

문제

- N개의 수를 오름차순 정렬

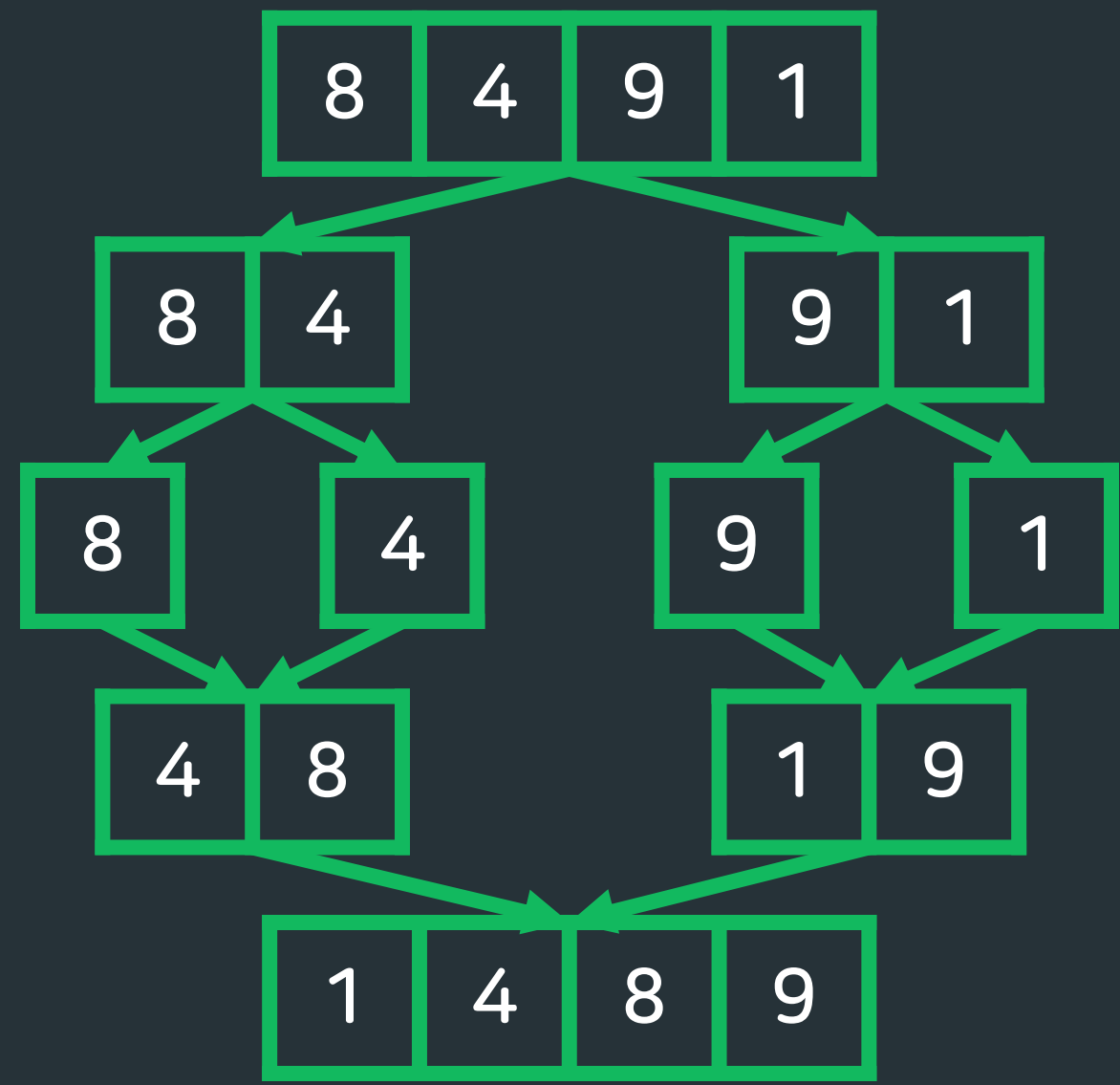
제한 사항

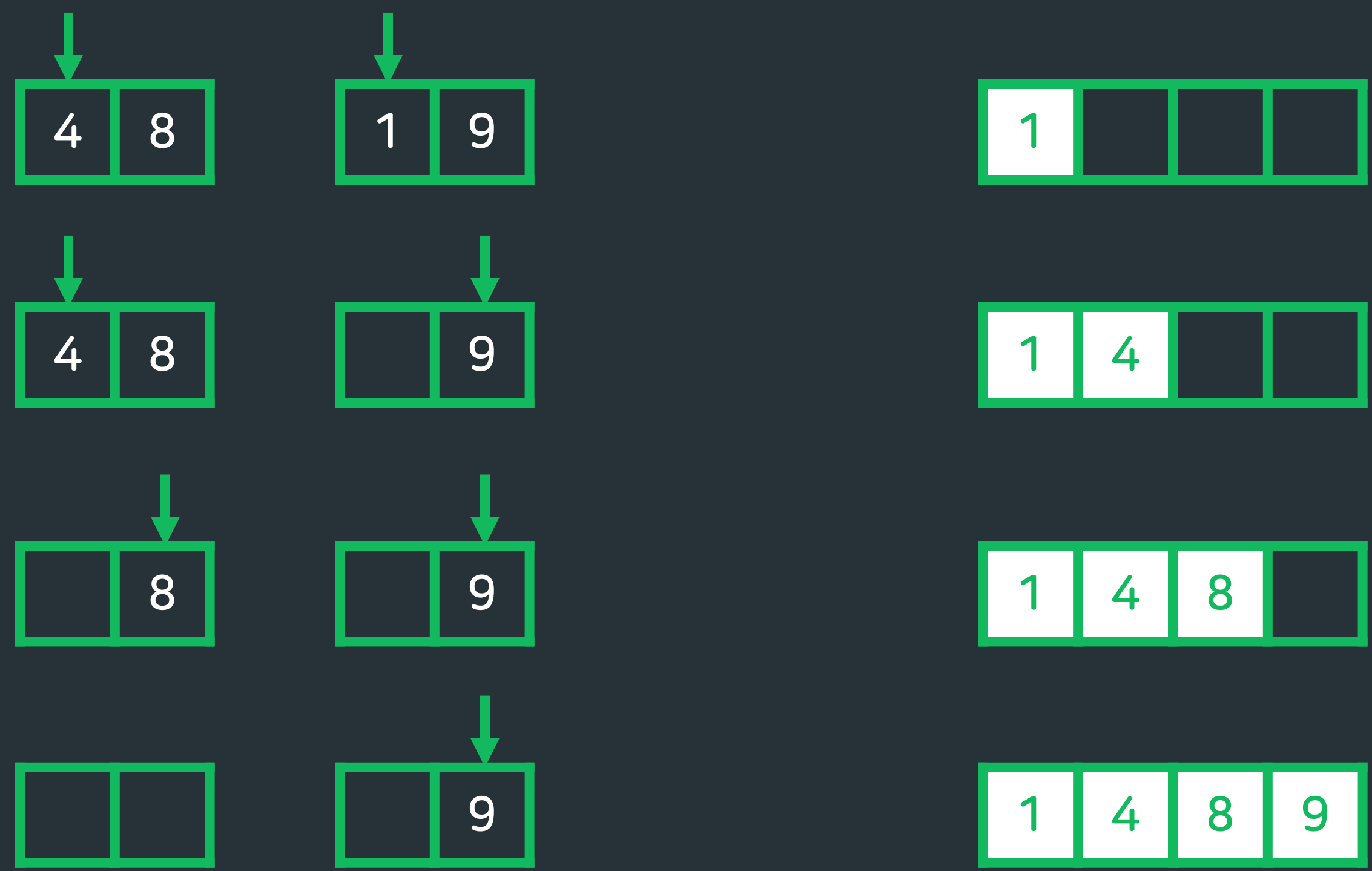
- N의 범위는 $1 \leq N \leq 1,000$
- 각각의 수 k는 $-1,000 \leq k \leq 1,000$ 이며 중복되지 않음

-> N의 범위가 최대 1,000이기 때문에 $O(n^2)$ 의 알고리즘이라도 시간초과가 발생하지 않음!

Merge sort

- 하나의 배열을 **같은 크기로 나눔** (Divide)
- 나뉜 배열들을 **정렬** (Conquer)
- 다시 하나의 배열로 **합치기** (Merge)





/<> 2751번 : 수 정렬하기 2 - Silver 5

문제

- N개의 수를 오름차순 정렬

제한 사항

- N의 범위는 $1 \leq N \leq 1,000,000$
- 각각의 수 k는 $-1,000,000 \leq k \leq 1,000,000$ 이며 중복되지 않음

/<> 2751번 : 수 정렬하기 2 - Silver 5


문제

- N개의 수를 오름차순 정렬

제한 사항

- N의 범위는 $1 \leq N \leq 1,000,000$
- 각각의 수 k는 $-1,000,000 \leq k \leq 1,000,000$ 이며 중복되지 않음

-> N의 범위가 최대 1,000,000이기 때문에 $O(n^2)$ 의 알고리즘이라면 시간초과!



Search: Go

Not logged in

register log in

Reference <algorithm> sort

You were redirected to cplusplus.com/sort || See search results for: "sort"

C++

Information

Tutorials

Reference

Articles

Forum

Reference

C library:

Containers:

Input/Output:

Multi-threading:

Other:

<algorithm>

<bitset>

<chrono>

<codecvt>

<complex>

<exception>

<functional>

<initializer_list>

<iterator>

<limits>

<locale>

<memory>

<new>

<numeric>

<random>

<ratio>

<regex>

<stdexcept>

<string>

<system_error>

<tuple>

<typeindex>

<typeinfo>

function template

std::sort

<algorithm>

default (1)

template <class RandomAccessIterator>
void sort (RandomAccessIterator first, RandomAccessIterator last);

custom (2)

template <class RandomAccessIterator, class Compare>
void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);

Sort elements in range

Sorts the elements in the range [first,last) into ascending order.

The elements are compared using operator< for the first version, and *comp* for the second.

Equivalent elements are not guaranteed to keep their original relative order (see *stable_sort*).

Parameters

first, last

Random-access iterators to the initial and final positions of the sequence to be sorted. The range used is [first,last), which contains all the elements between *first* and *last*, including the element pointed by *first* but not the element pointed by *last*.
RandomAccessIterator shall point to a type for which *swap* is properly defined and which is both *move-constructible* and *move-assignable*.

comp

Binary function that accepts two elements in the range as arguments, and returns a value convertible to bool. The value returned indicates whether the element passed as first argument is considered to go before the second in the specific *strict weak ordering* it defines.
The function shall not modify any of its arguments.
This can either be a function pointer or a function object.

Return value

none

/<> 10825번 : 국영수 - Silver 4

문제

- 도현이네 반 학생 N명의 이름과 국어, 영어, 수학 점수가 주어진다.
- 다음의 조건으로 학생들을 정렬하자.
 1. 국어 점수가 감소하는 순서
 2. 국어 점수가 같다면 영어 점수가 증가하는 순서
 3. 국어 점수와 영어 점수가 같다면 수학 점수가 감소하는 순서
 4. 모든 점수가 같으면 이름이 사전 순으로 증가하는 순서

제한 사항

- N의 범위는 $1 \leq N \leq 100,000$
- 점수의 범위는 $1 \leq \text{score} \leq 100$
- 이름은 알파벳 대소문자로 이루어진 10자리 이하의 문자열

Hint

1. 구조체... 기억나시나요?
2. 분명히 아까 쓴 sort 함수는 인자(parameter)가 2개였는데?

std::sort

<algorithm>

```
default (1)  template <class RandomAccessIterator>
              void sort (RandomAccessIterator first, RandomAccessIterator last);
custom (2)   template <class RandomAccessIterator, class Compare>
              void sort (RandomAccessIterator first, RandomAccessIterator last, Compare comp);
```

이건 뭡까요??

정리

- 정렬 알고리즘은 종류가 많다.
- 근데 그냥 구현하지 말고 **sort 함수** 쓰자!
- **default** 값은 **오름차순** 정렬, **내림차순** 정렬은 **greater<>()**, 그 밖의 정렬은 **comp** 정의하기.
- **comp** 정의할 때는 헛갈리지 말기! **sort**는 **comp**가 **false**를 반환해야 **swap**됨! (sort는...?)
- 정렬 알고리즘은 **그리디** 문제에 쓰이는 경우가 많아요!

이것도 알아보세요!

- 정렬 알고리즘 중엔 시간 복잡도가 **$O(n)$** 인 **계수 정렬(Counting sort)**이 있어요.
 1. **어떻게** 겨우 **$O(n)$** 만에 정렬을 할 수 있을까요?
 2. **우린** 그럼 **왜** 계수 정렬을 쓰지 않고 **$O(n \log n)$** 의 정렬 알고리즘을 사용하는 걸까요?
- 정렬 알고리즘은 **stable sort**와 **unstable sort**로 나눌 수 있어요. **이건** 어떤 개념일까요?
- 자료형이 **`pair<int, int>`**인 배열을 **comp없이 정렬**하면 어떻게 될까요?

필수

- /<> 1316번 : 그룹 단어 체커 - Silver 5
- /<> 13458번 : 시험 감독 - Bronze 2

3문제 이상 선택

- /<> 11651번 : 좌표 정렬하기 - Silver 5
- /<> 1026번 : 보물 - Silver 4
- /<> 1431번 : 시리얼 번호 - Silver 3
- /<> 11399번 : ATM - Silver 3
- /<> 1946번 : 신입 사원 - Silver 1
- /<> 10994번 : 별 찍기 - 19 - Silver 4