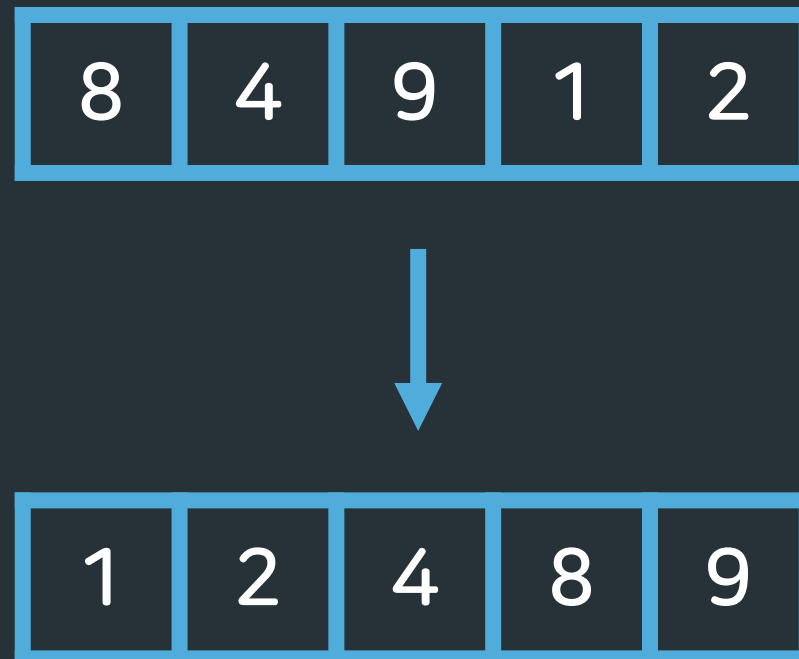


알튜비튜

정렬

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



- 정렬: 데이터를 특정한 기준에 따라 순서대로 나열하는 것

대표적인 정렬 알고리즘



$O(n^2)$

Insertion sort
Selection sort
Bubble sort

$O(n \log n)$

Quick sort
Merge sort
Heap sort

$O(n^2)$

Insertion sort
Selection sort
Bubble sort

$O(n \log n)$

Quick sort
Merge sort
Heap sort

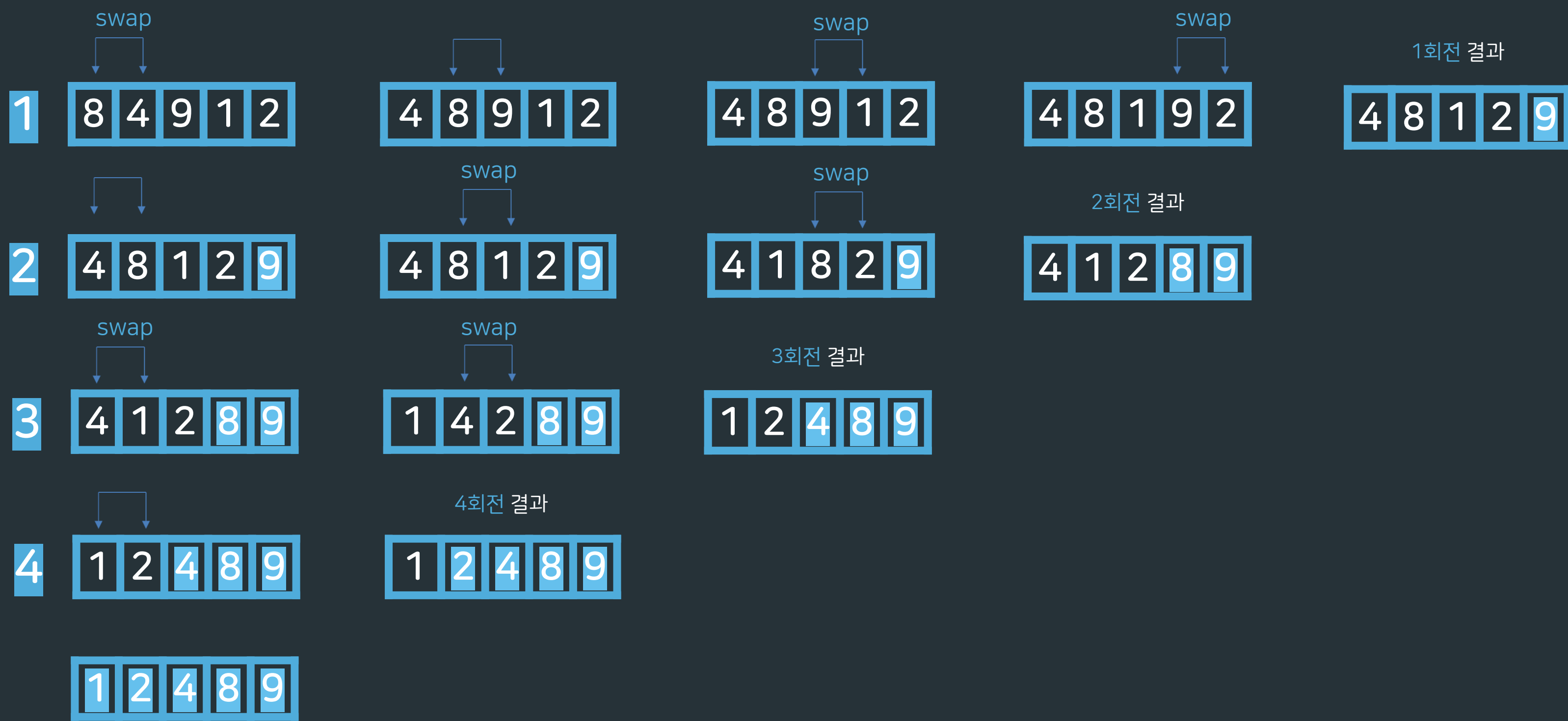
오름차순 정렬이라고 가정하고 설명합니다!

Bubble sort

- 인접한 두 원소를 비교
- (왼쪽 원소) > (오른쪽 원소) 라면 swap!
- 가장 큰 원소부터 오른쪽에 정렬됨
- 데이터가 하나씩 정렬되면서 비교에서 제외

버블 정렬

- (왼쪽 원소) > (오른쪽 원소) 라면 swap!



버블 정렬

8	4	9	1	2
---	---	---	---	---

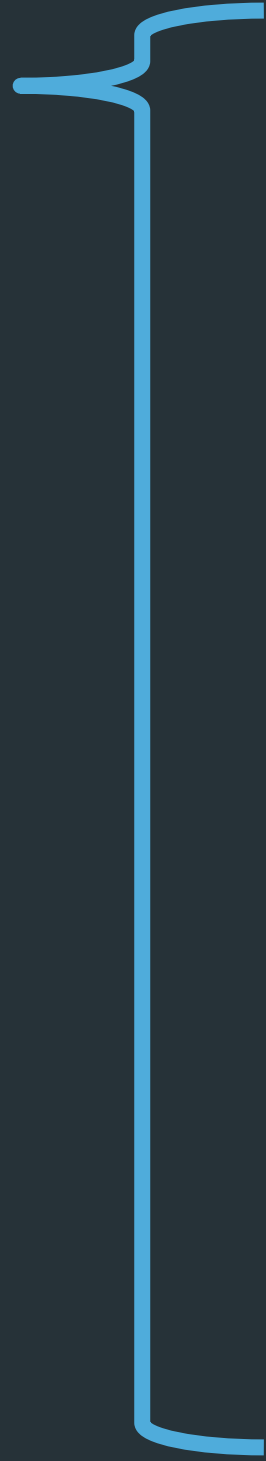
4	8	1	2	9
---	---	---	---	---

4	1	2	8	9
---	---	---	---	---

1	2	4	8	9
---	---	---	---	---

1	2	4	8	9
---	---	---	---	---

1	2	4	8	9
---	---	---	---	---



8	4	9	1	2
---	---	---	---	---

8	4	9	1	2
---	---	---	---	---

4	8	9	1	2
---	---	---	---	---

4	8	9	1	2
---	---	---	---	---

4	8	1	9	2
---	---	---	---	---

4	8	1	2	9
---	---	---	---	---

/<> 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

예제 입력1

```
5
5 2 3 4 1
```

예제 입력2

```
5
2 1 3 4 5
```

예제 출력1

```
1 2 3 4 5
```

예제 출력2

```
1 2 3 4 5
```

Bubble sort

- 가장 쉽지만 가장 비효율적인 알고리즘
- N개의 입력을 정렬하는 시간 복잡도 $O(N^2)$
- 비교 횟수 $(N-1) + (N-2) + \dots + 1 = N(N-1)/2$
- 자료의 교환(SWAP)은 이동(MOVE) 3번에 해당하는 Burden한 작업
- 하나의 요소가 가장 왼쪽에서 가장 오른쪽으로 이동하기 위해서는 배열에서 모든 다른 요소들과 교환 되어야함.
- 특히, 이미 배열이 이미 정렬 완료된 상태일지라도 불필요한 회전을 수행함.

Advanced Bubble Sort

- 앞선 버블 정렬의 단점을 보완하고자 만들어짐
- 이전 회전에서 교환(SWAP)이 한번도 일어나지 않았다면 정렬이 이루어지지 않은 원소가 없다는 점을 활용.

/<> 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

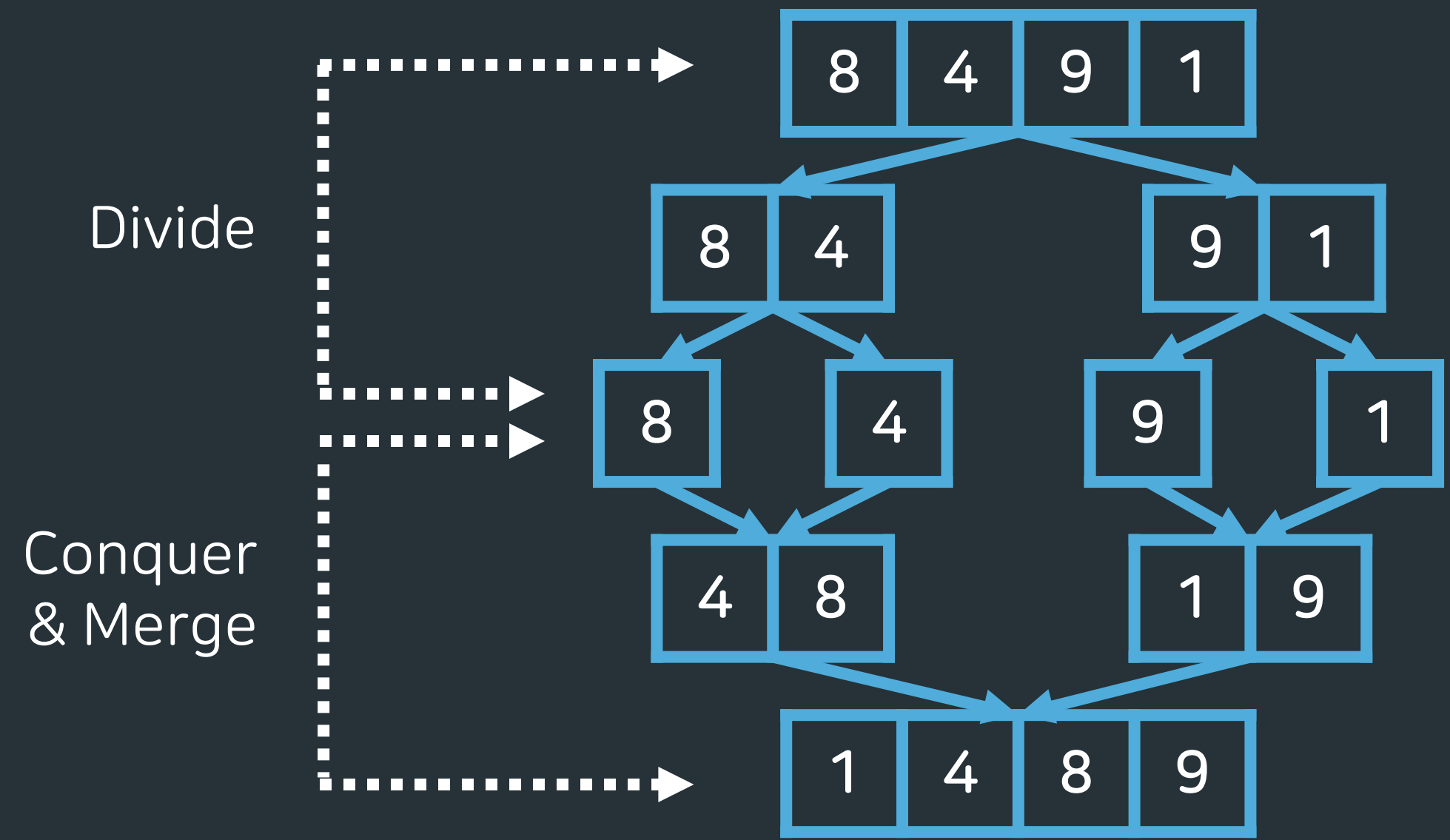
→ N의 범위가 최대 1,000이기 때문에 $O(n^2)$ 의 알고리즘이라도 시간초과가 발생하지 않음!

Merge sort

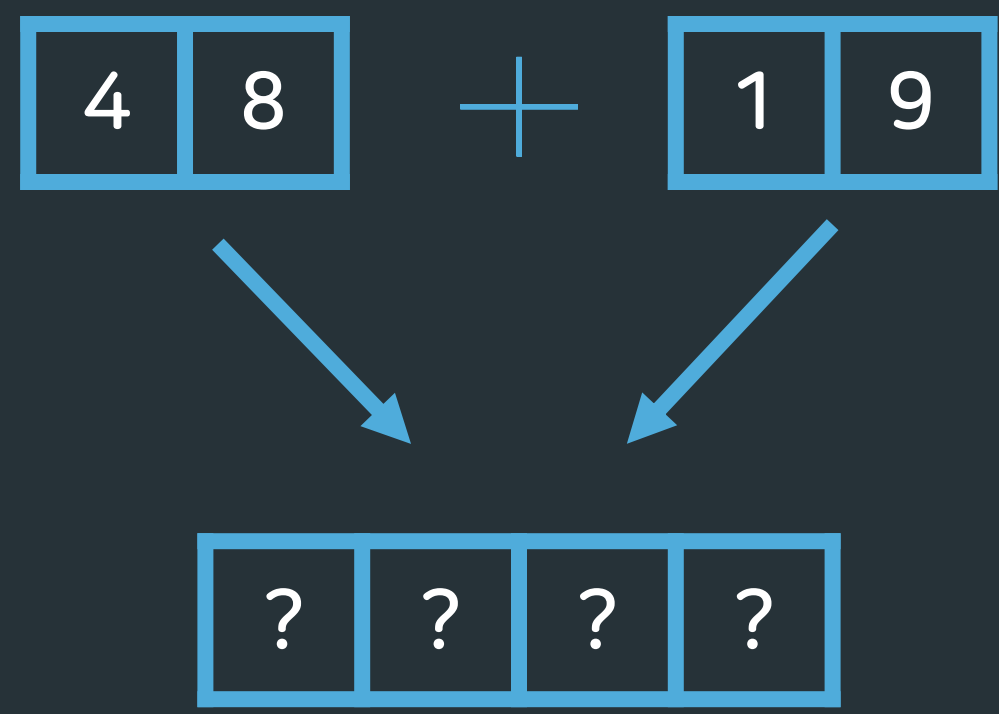
- 분할 정복(Divide and Conquer) 방식으로 설계된 알고리즘
- 하나의 배열을 정확히 반으로 나눔 (Divide)
- 나뉜 배열들을 정렬 (Conquer)
- 다시 하나의 배열로 합치기 (Merge)

분할 정복

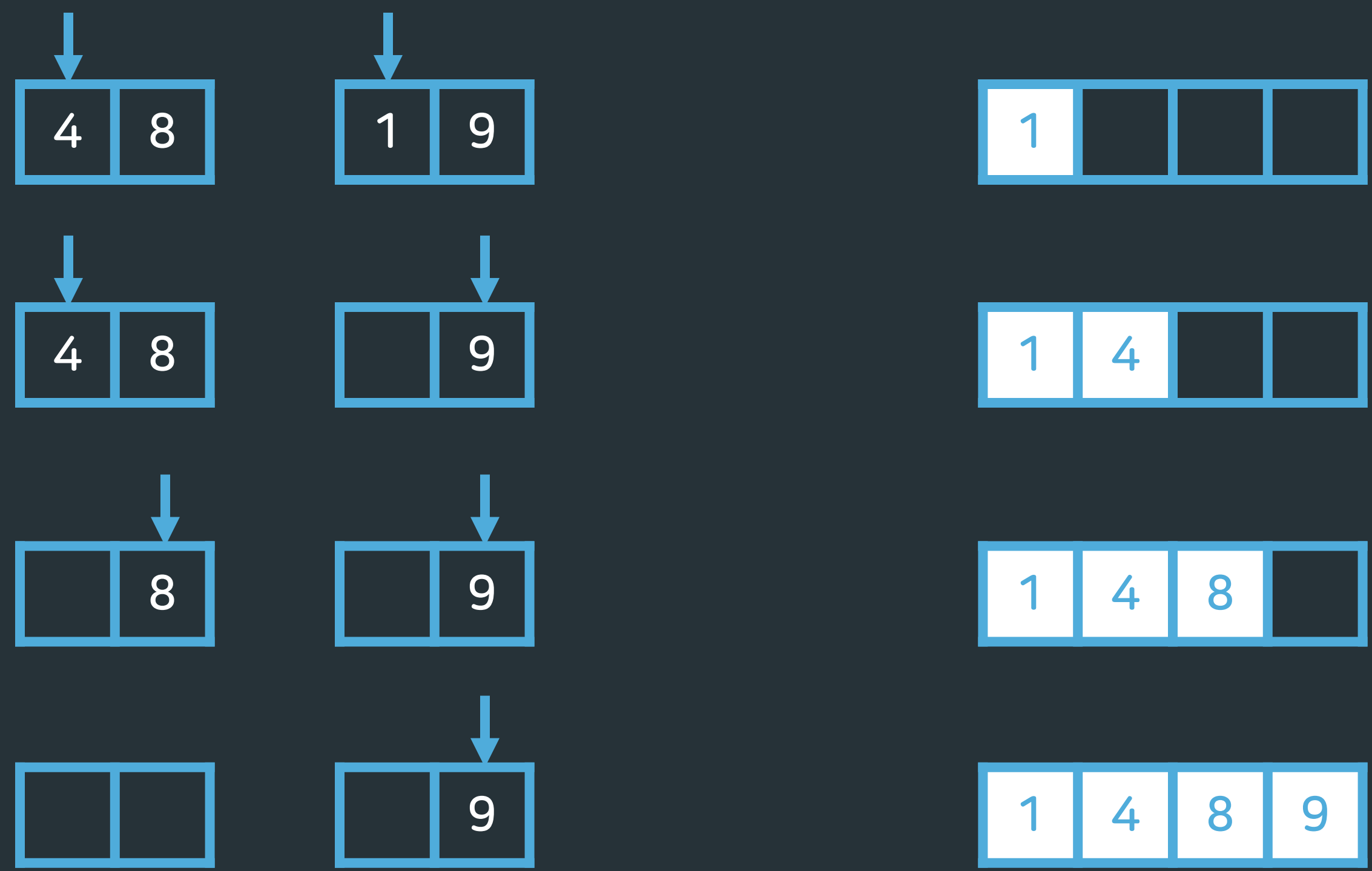
- 한 번에 해결할 수 없는 문제를 작은 문제로 분할하여 해결하는 알고리즘
- 주로 재귀 함수로 구현
- 크게 3 단계로 이루어짐
 1. Divide: 문제 분할
 2. Conquer: 쪼개진 작은 문제 해결
 3. Combine: 해결된 작은 문제들을 다시 합침



- 1. Divide
: 입력 배열을 비슷한 크기의 2개의 부분 배열로 분할한다.
- 2. Conquer & Merge
: 부분배열을 합치면서 정렬한다.



두개의 정렬된 배열을 하나의 정렬된 배열로 합치려면?



/<> 11728번 : 배열 합치기 – Silver 5

문제

- 정렬되어 있는 두 배열 A,B가 주어질때 두 배열을 합쳐서 정렬한 결과를 출력

제한 사항

- N,M의 범위는 $1 \leq N,M \leq 1,000,000$

예제 입력1

```
2 2
3 5
2 9
```

예제 입력2

```
2 1
4 7
1
```

예제 입력3

```
4 3
2 3 5 9
1 4 7
```

예제 출력1

```
2 3 5 9
```

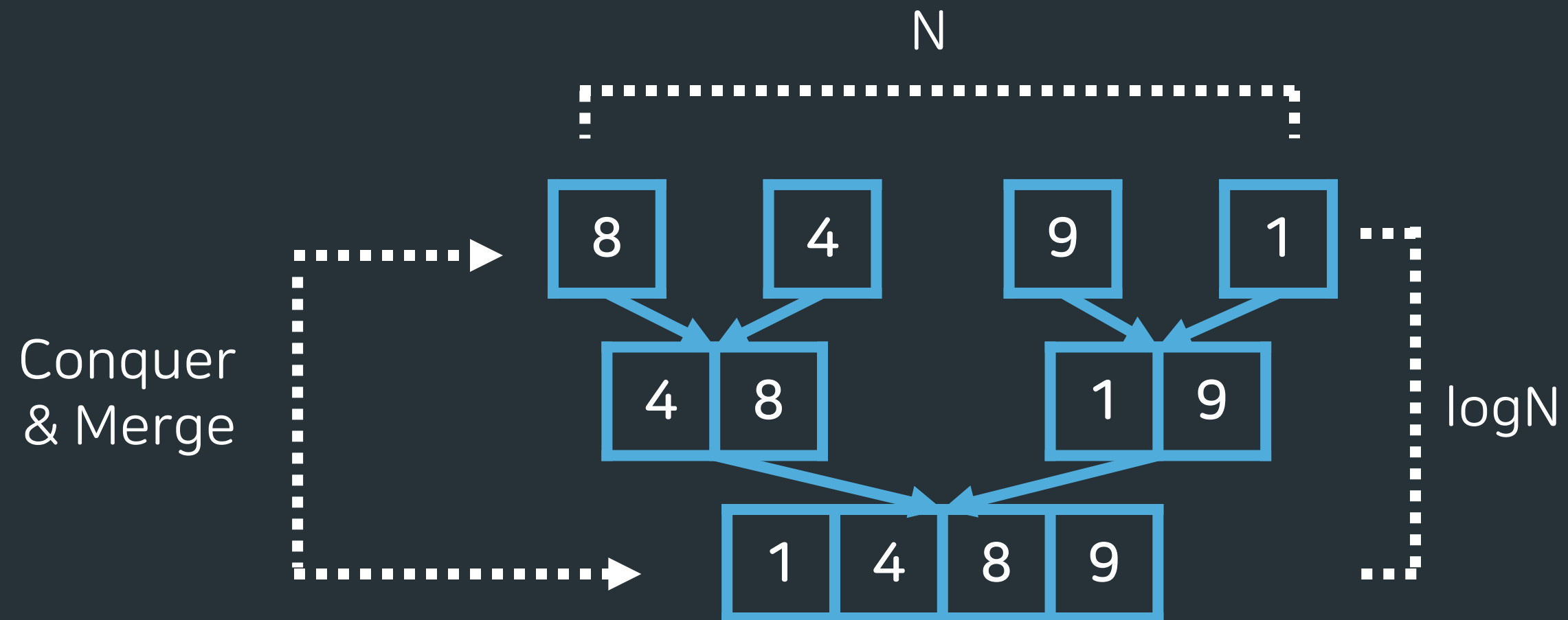
예제 출력2

```
1 4 7
```

예제 출력3

```
1 2 3 4 5 7 9
```

- 시간복잡도 $O(n \log n)$



/<> 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

예제 입력1

```
5
5 2 3 4 1
```

예제 입력2

```
5
2 1 3 4 5
```

예제 출력1

```
1 2 3 4 5
```

예제 출력2

```
1 2 3 4 5
```

/<> 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!

세상에 정렬할 일이 얼마나 많은데...



Search:

Reference <algorithm> sort

Not logged in

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>

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

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


```
● ● ●  
  
#include <algorithm>  
  
int arr[10] = { 1,4,5,2,9,8,6,10,3,7};  
    sort(arr, arr + 10);  
  
    vector<int> v = {1,4,5,2,9,8,6,10,3,7};  
    sort(v.begin(), v.end()); // sort (v.begin(),v.begin()+10)  
  
}
```

/<> 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

예제 입력

```
12
Junkyu 50 60 100
Sangkeun 80 60 50
Sunyoung 80 70 100
Soong 50 60 90
Haebin 50 60 100
Kangsoo 60 80 100
Donghyuk 80 60 100
Sei 70 70 70
Wonseob 70 70 90
Sanghyun 70 70 80
nsj 80 80 80
Taewhan 50 60 90
```

예제 출력

```
Donghyuk
Sangkeun
Sunyoung
nsj
Wonseob
Sanghyun
Sei
Kangsoo
Haebin
Junkyu
Soong
Taewhan
```

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 뭘까요??

std::sort

- 인자로 배열의 처음 시작 위치와, 끝 위치를 보내줌
- default 값은 오름차순 정렬
- 내림차순 정렬은 세 번째 인자에 `greater<>()` 을 넣어서
- 세 번째 인자에 비교함수(`cmp`)를 넣어서 원하는 조건대로 정렬할 수 있음!
- `cmp(int a, int b)`는 `a`가 `b`의 앞에 와야할 때 `true`를 리턴하도록 하기.
- 비교함수가 `false`를 리턴할 경우에는 `swap`하는 것임을 주의!

주의사항

- compare 함수는 두 값이 같을 때(혹은 우선순위가 같을 때) false를 반환 해야 한다.

```
bool cmp(int a, int b) {  
    if (a >= b) return true;  
    return false;  
}
```

X

```
bool cmp(int a, int b) {  
    if (a > b) return true;  
    return false;  
}
```

O

- 비교 함수의 인자로 STL 혹은 클래스 객체를 전달 시 reference를 사용하기

```
bool cmp(string a, string b) {  
    return a < b; //사전순으로  
}
```

```
bool cmp(const string& a, const string& b) {  
    return a < b; //사전순으로  
}
```

정리

- 정렬 알고리즘은 종류가 많다. (Insertion, Selection, Bubble, Merge, Quick, ...)
- 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!

- 비교함수 작성 시 인자를 넘겨줄 때 왜 `const`와 `&`를 사용할까요?
- 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. 우리 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 `comp`없이 정렬하면 어떻게 될까요?

2문제 이상 선택

- /<> 11651번 : 좌표 정렬하기 2 - Silver 2
- /<> 1758번 : 알바생 강호 - Silver 4
- /<> 1431번 : 시리얼 번호 - Silver 3
- /<> 1946번 : 신입 사원 - Silver 1
- /<> 1026번 : 보물 - Silver 4