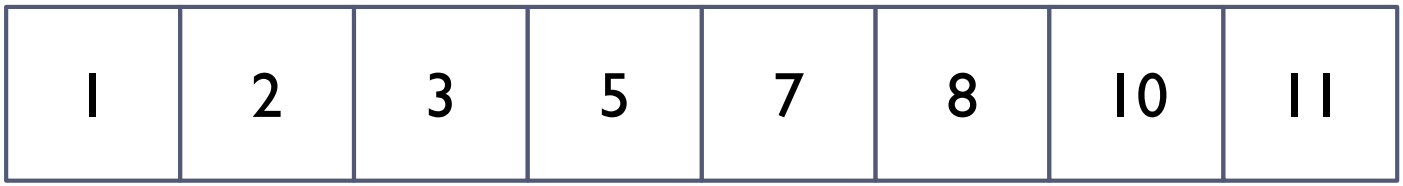
2013 Fall Semester SMP

Day 5 – Search & Sort

Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search



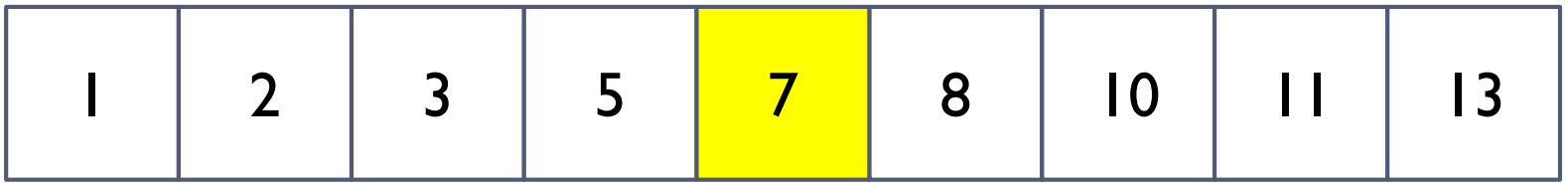
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (5)

1	2	3	5	7	8	10	11	13

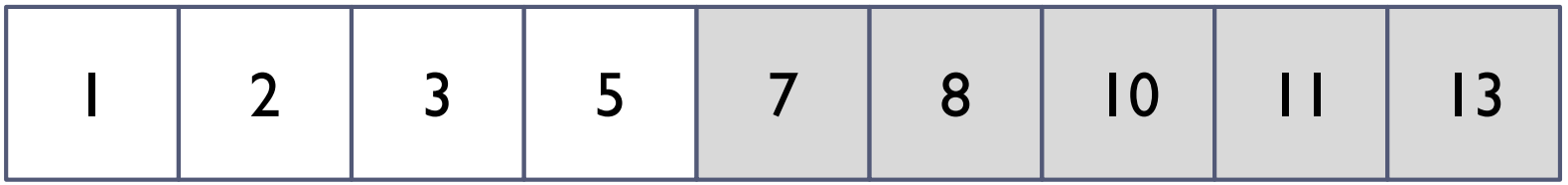
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2)



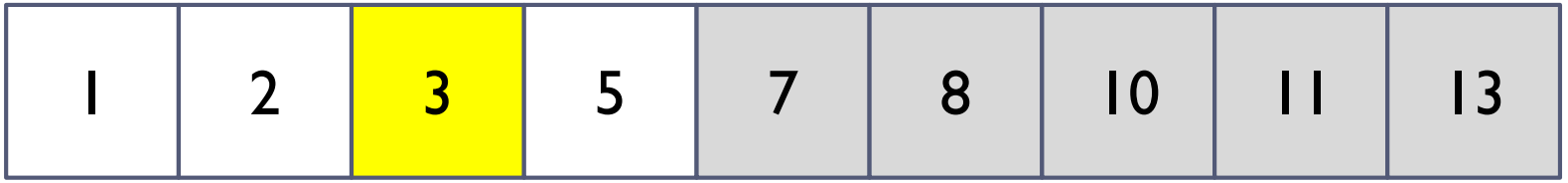
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2)



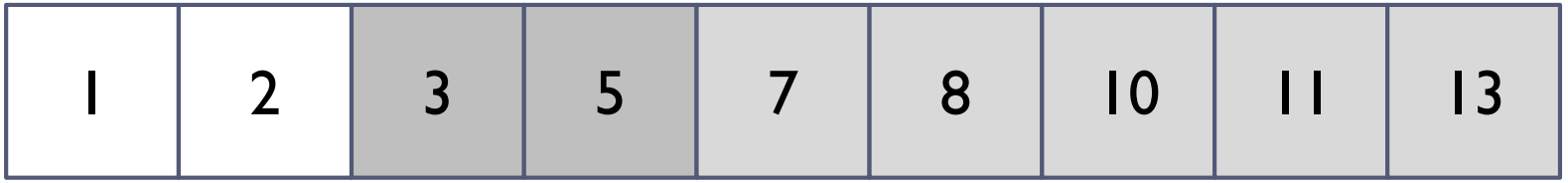
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2)



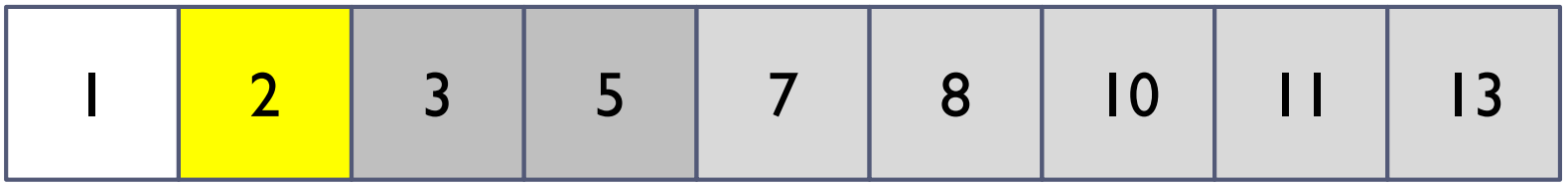
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2)



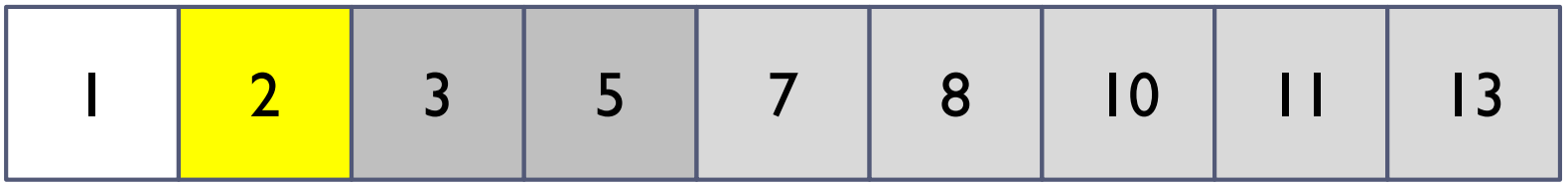
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2)



Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (2) **(!!)**



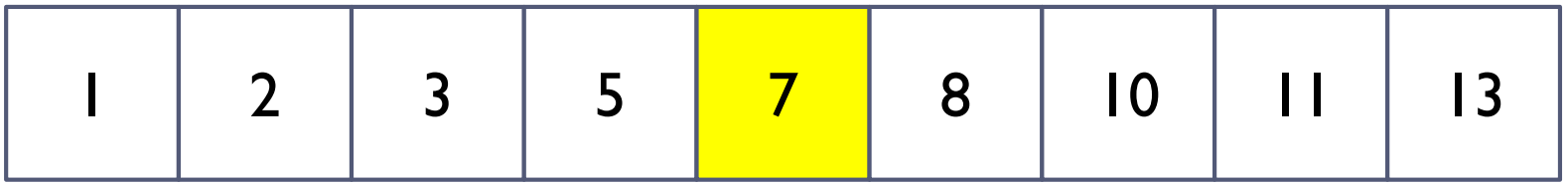
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)

1	2	3	5	7	8	10	11	13

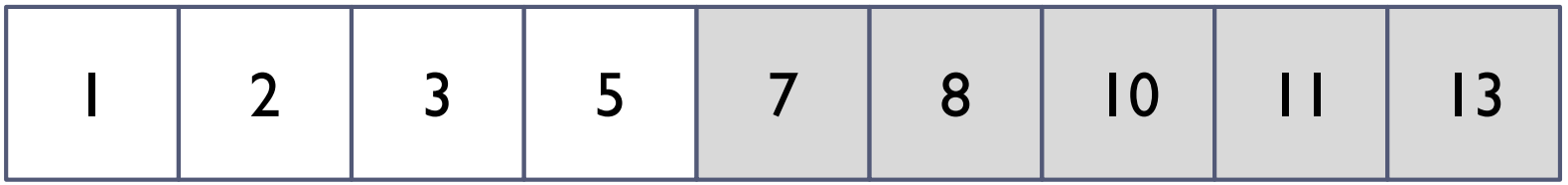
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)



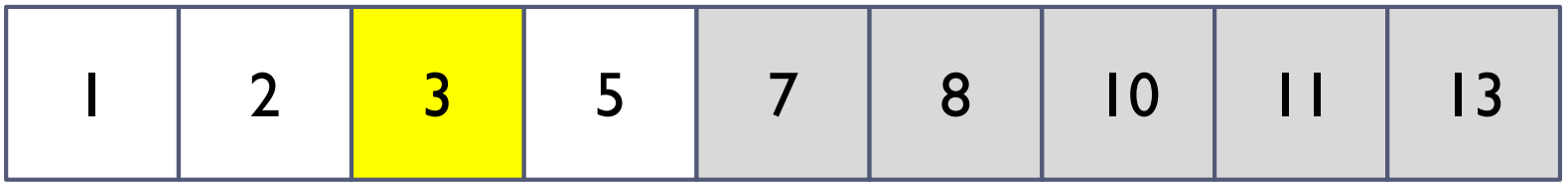
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)



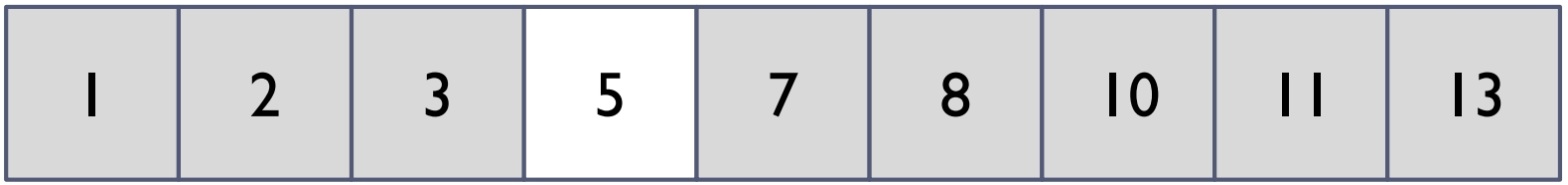
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)



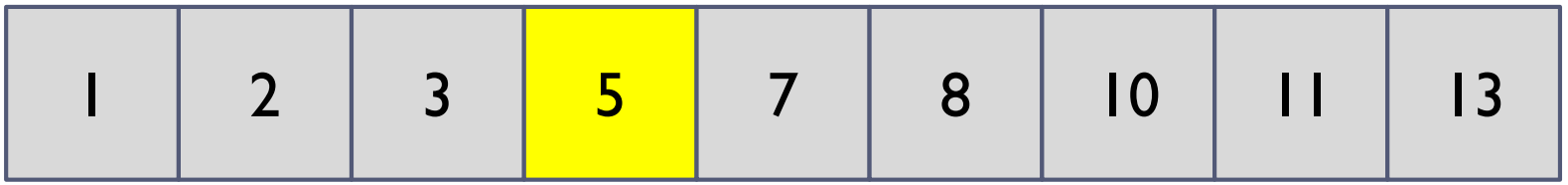
Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)



Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)



Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4)

1	2	3	5	7	8	10	11	13

Search

- 원하는 값을 찾는다
  - O(N)
- Data가 정렬되어 있다면?
  - O(log N)만에 가능!
  - ➔ Binary Search
- Search (4) **(??)**

1	2	3	5	7	8	10	11	13

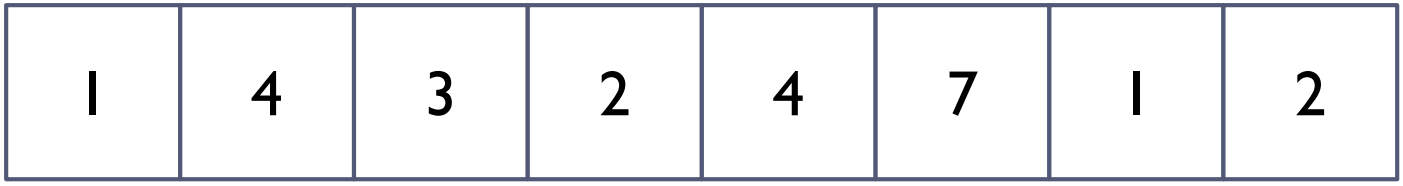
Merge Sort

- Divide & Conquer의 가장 기본적인 형태
  - 일단 나누고, 각각에 대해서 작업을 수행한 다음
  - 하나로 합친다!
- 매우 중요함
  - Divide & Conquer의 대표적 모델로써 중요하고
  - O(n log n) 의 효율적인 정렬 알고리즘이라 중요하다
  - 응용이 될 가능성이 매우 큼

Merge Sort

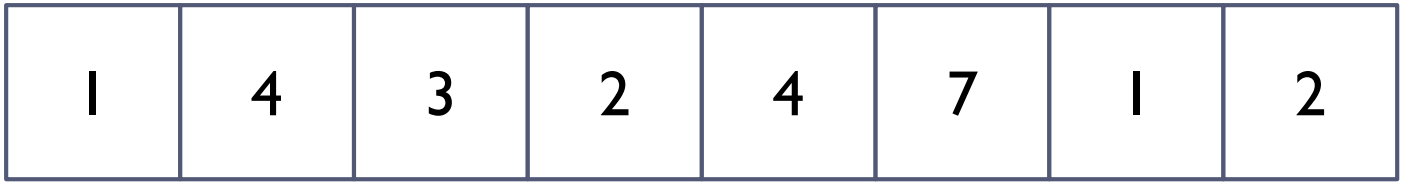
- Merge Sort의 절차
  - 1. 일단 두 부분으로 나눈다
  - 2. 각각의 부분을 어떻게든 정렬한다.
  - 3. 두 부분을 합친다

Merge Sort



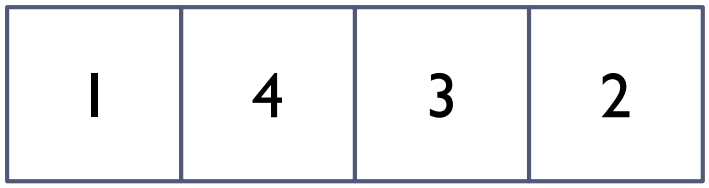
Merge Sort

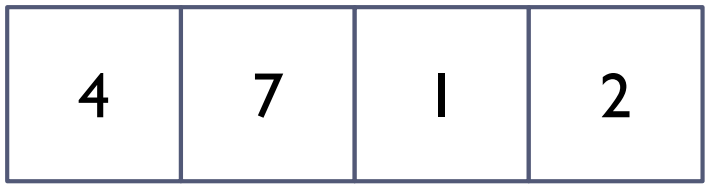
- 1. 일단 나눈다



Merge Sort

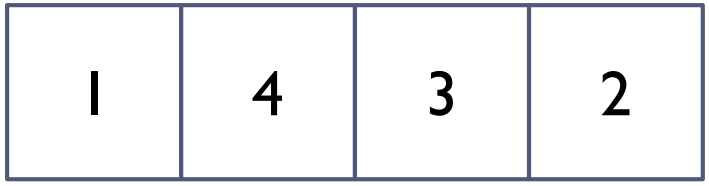
- 1. 일단 나눈다

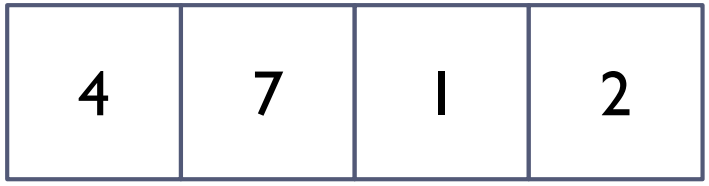




Merge Sort

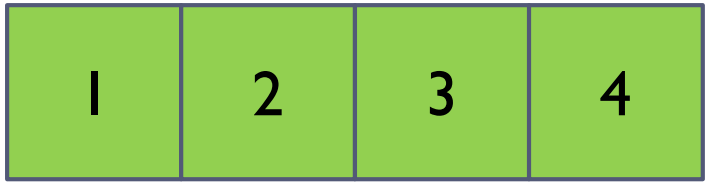
- 2. 어떻게든 둘을 정렬한다

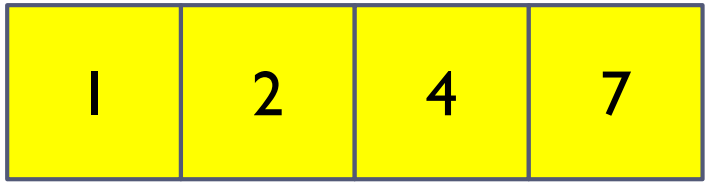




Merge Sort

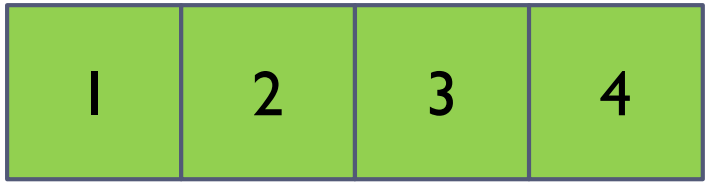
- 2. 어떻게든 둘을 정렬한다

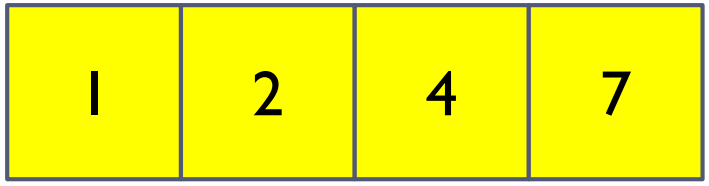




Merge Sort

- 3. 다시 하나로 합친다





Merge Sort

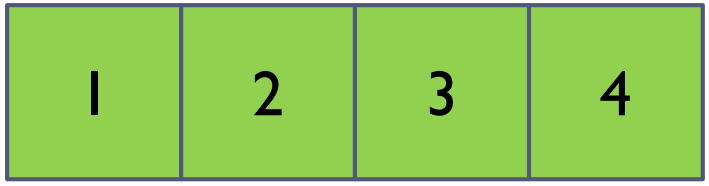
- 3. 다시 하나로 합친다

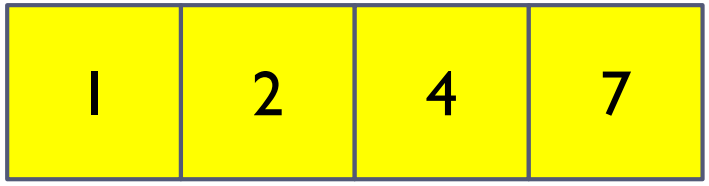
Merge Sort

- 3. 다시 하나로 합친다 ?????

Merge Sort

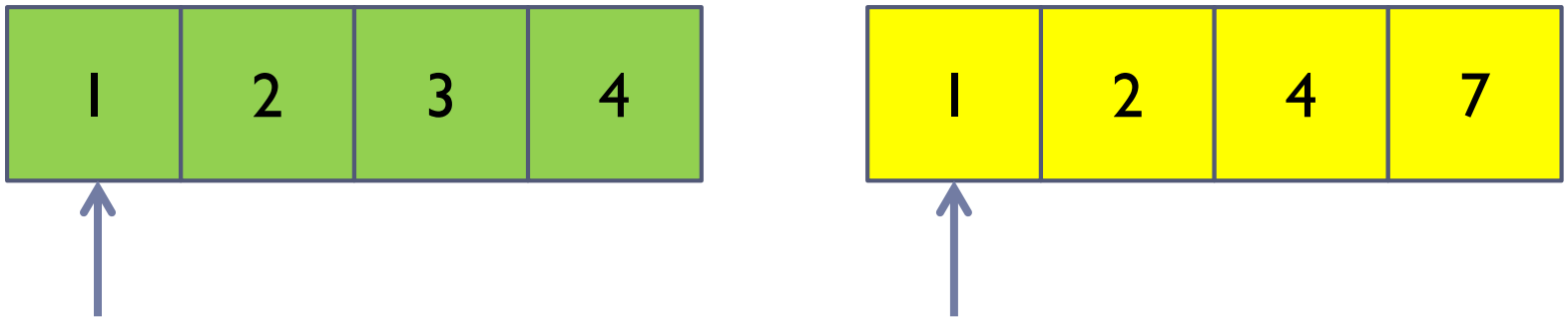
- 3. 다시 하나로 합친다





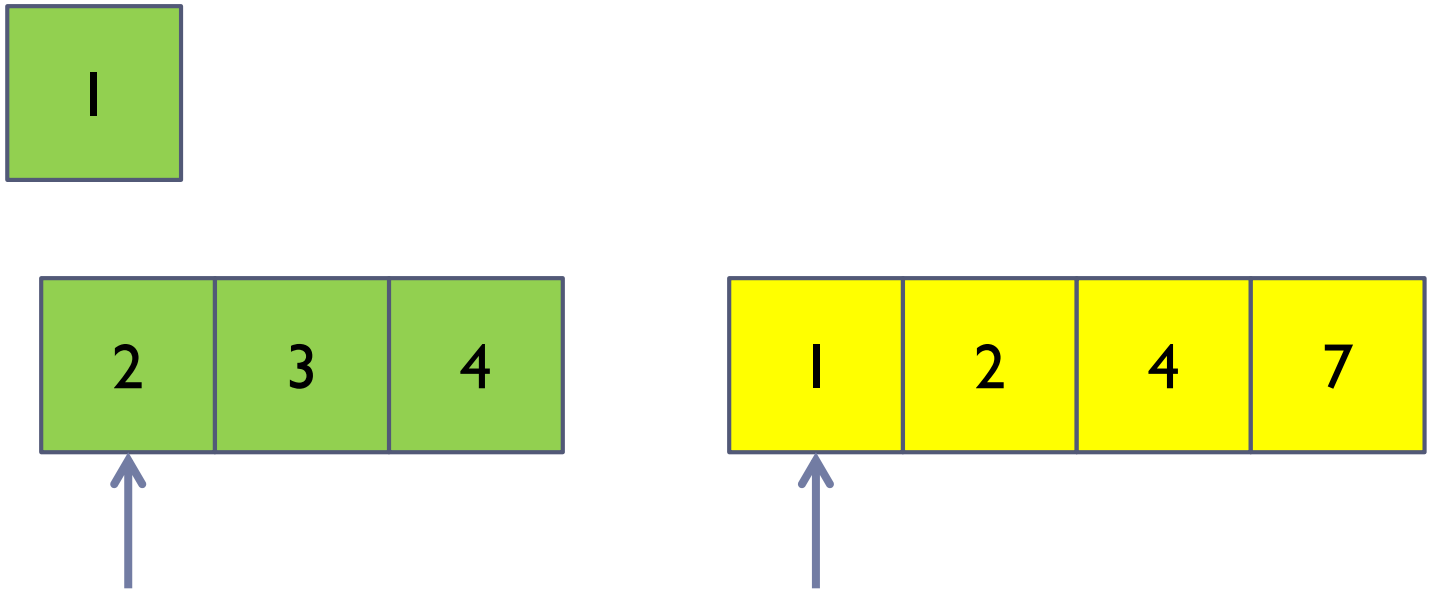
Merge Sort

- 3. 다시 하나로 합친다



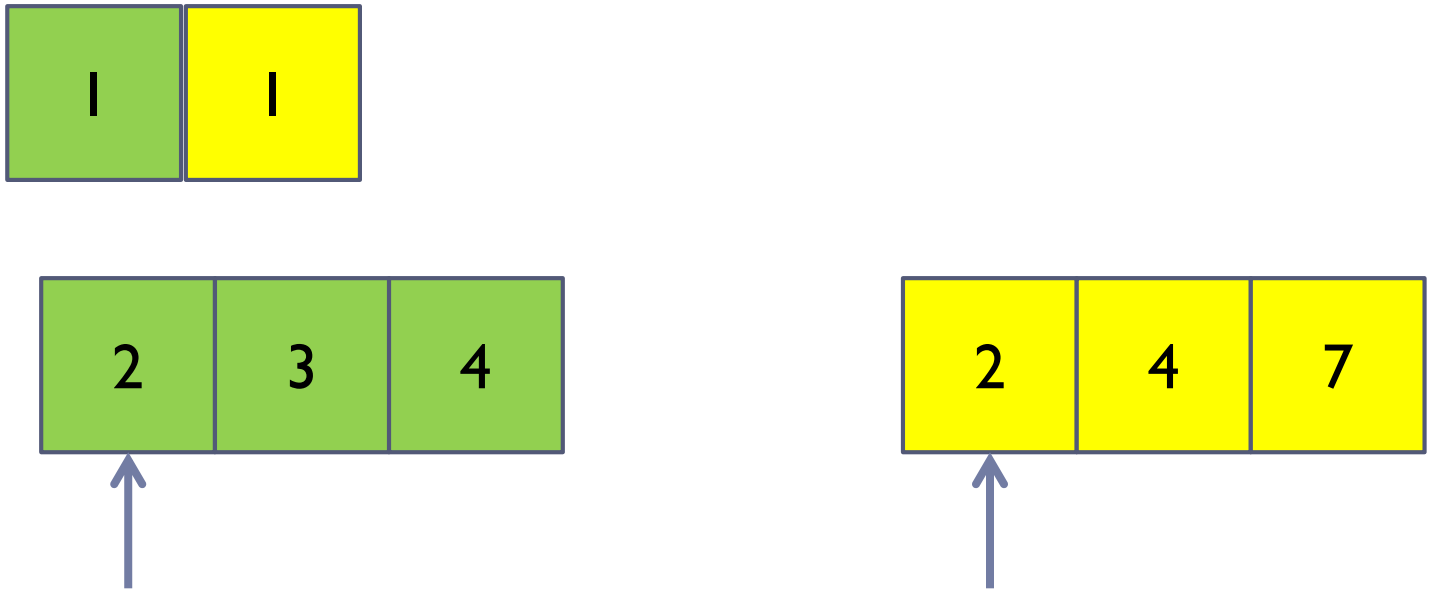
Merge Sort

- 3. 다시 하나로 합친다



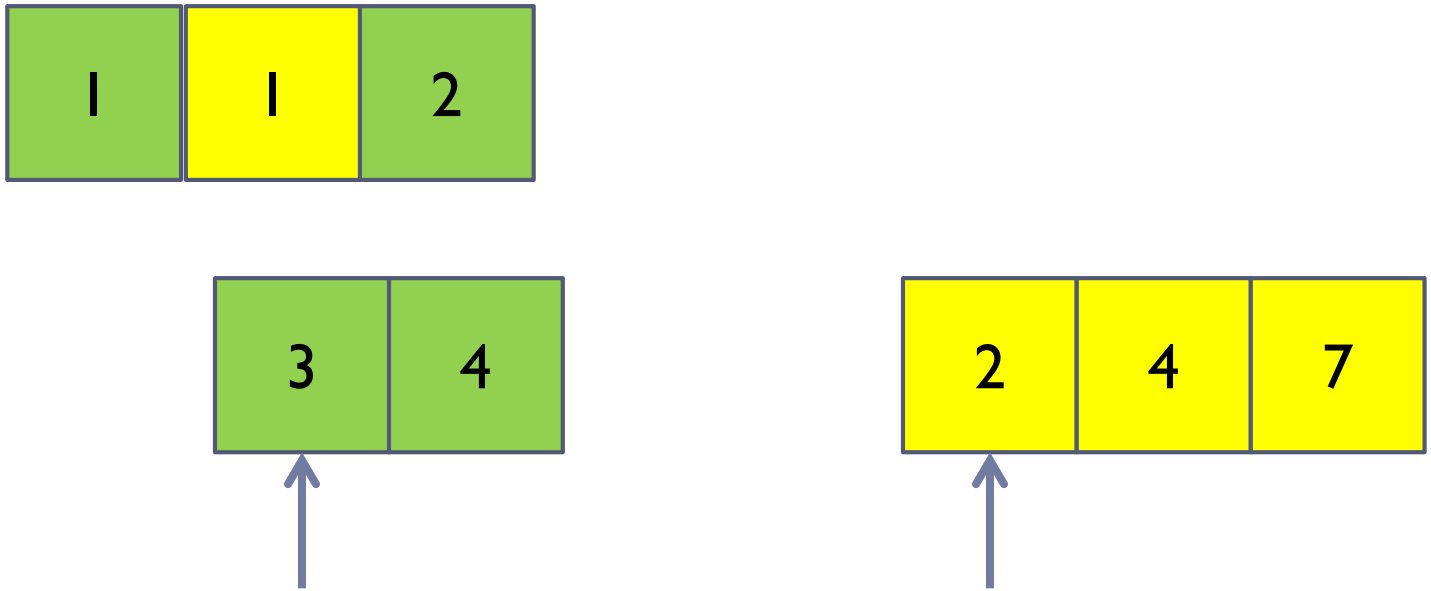
Merge Sort

- 3. 다시 하나로 합친다



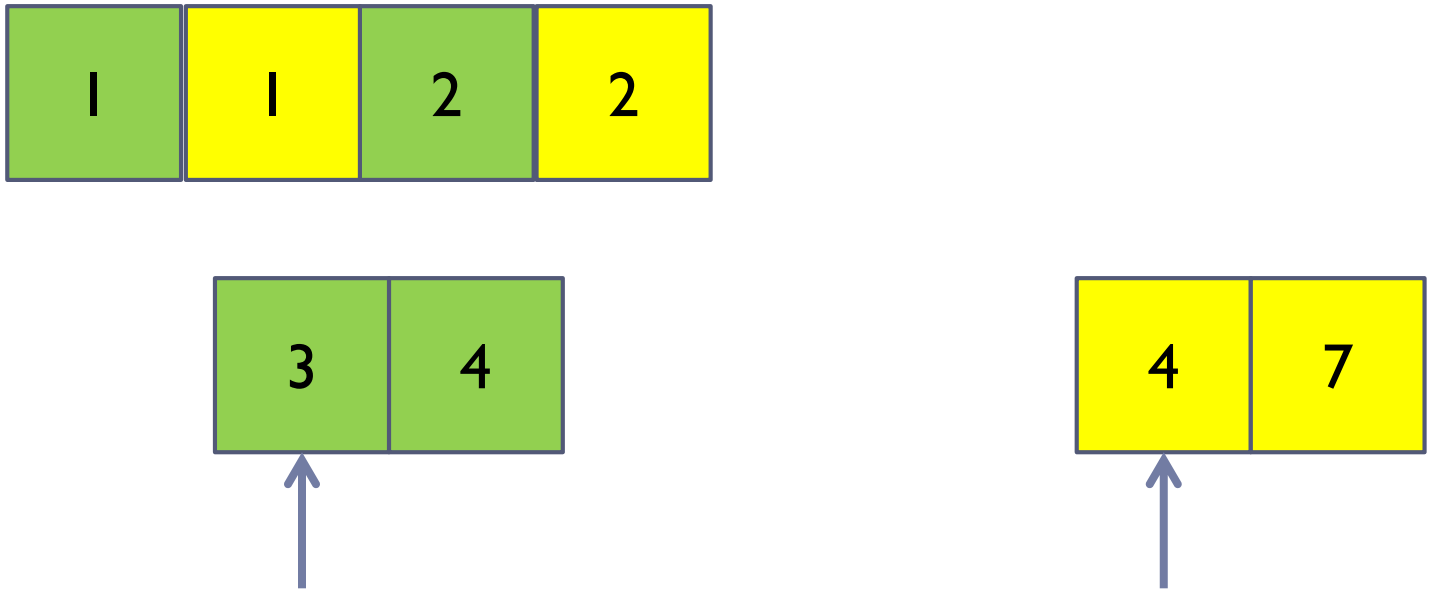
Merge Sort

- 3. 다시 하나로 합친다



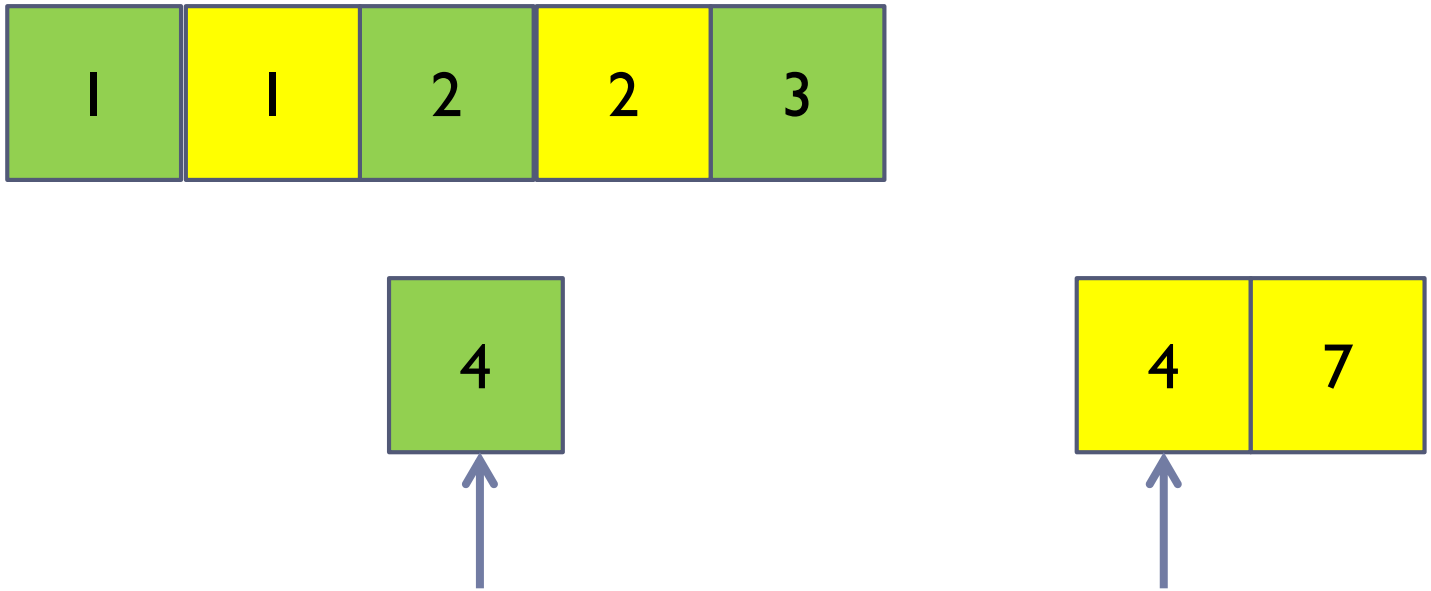
Merge Sort

- 3. 다시 하나로 합친다



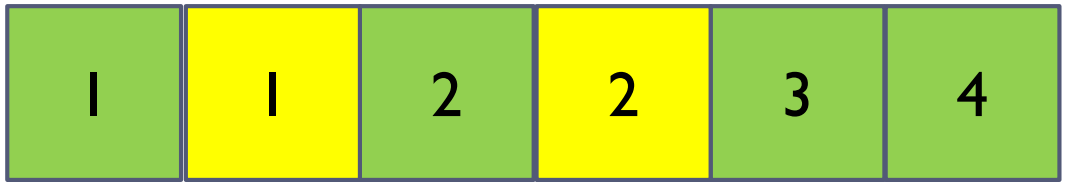
Merge Sort

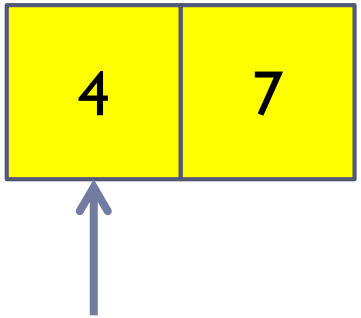
- 3. 다시 하나로 합친다



Merge Sort

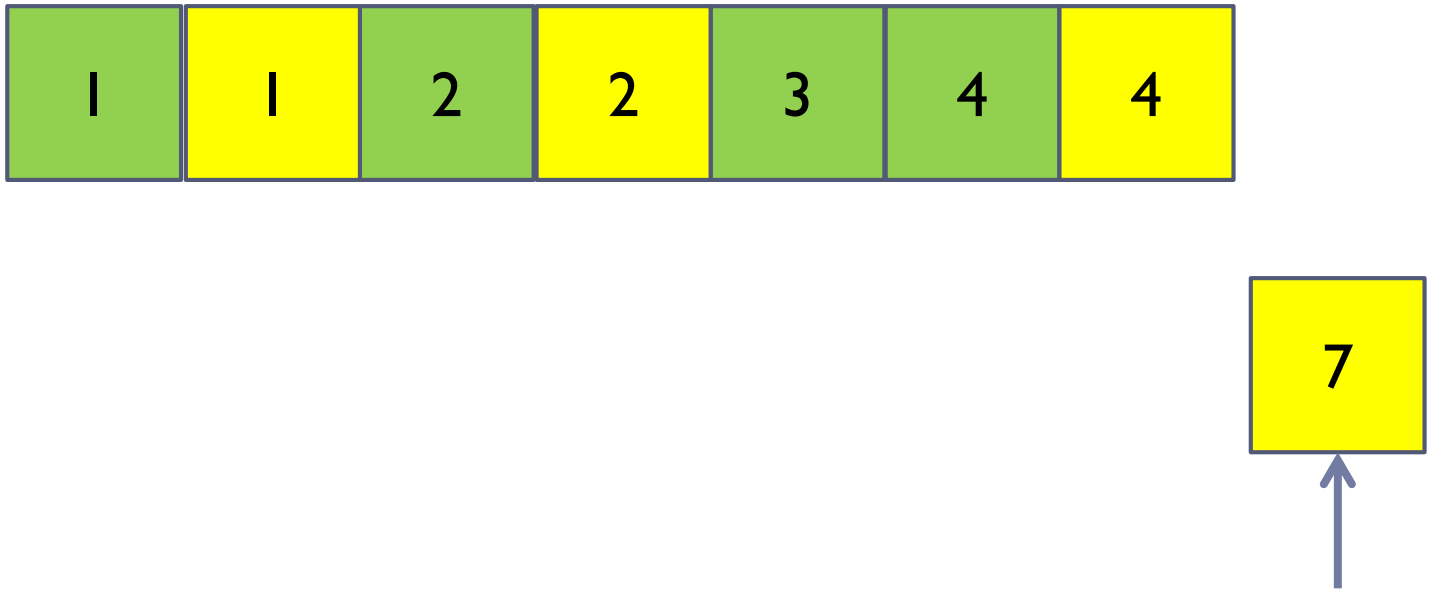
- 3. 다시 하나로 합친다





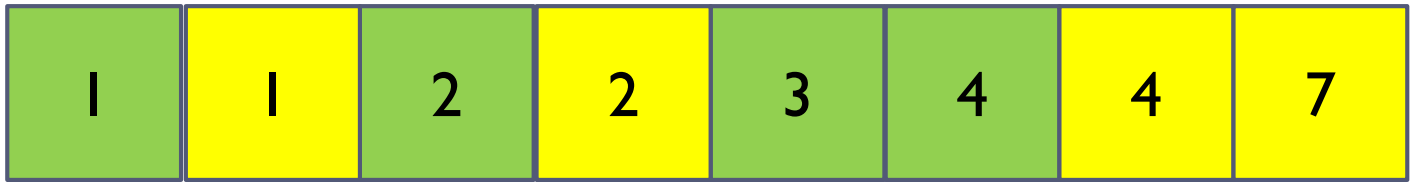
Merge Sort

- 3. 다시 하나로 합친다



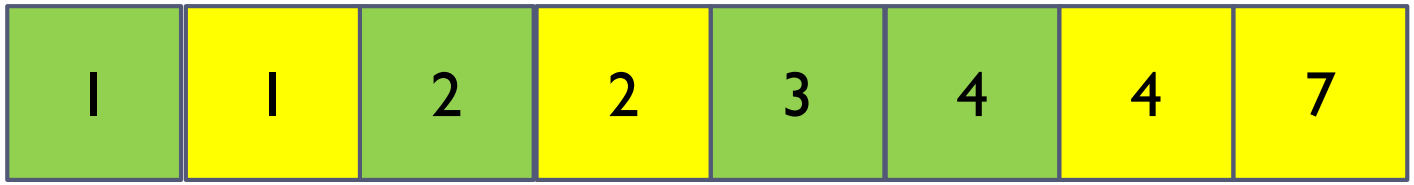
Merge Sort

- 3. 다시 하나로 합친다



Merge Sort

- 3. 다시 하나로 합친다 (!!!)



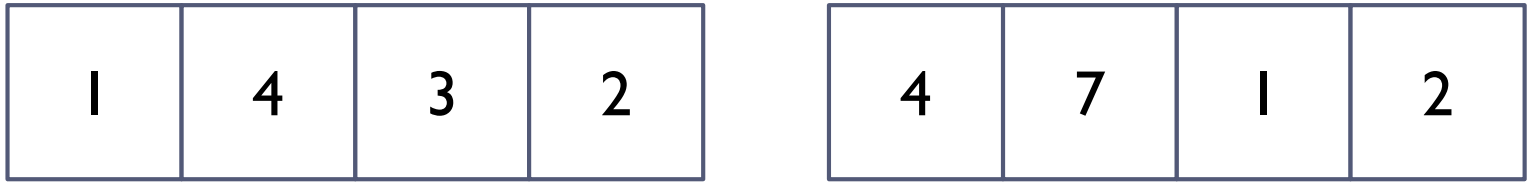
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!

1	4	3	2	4	7	1	2

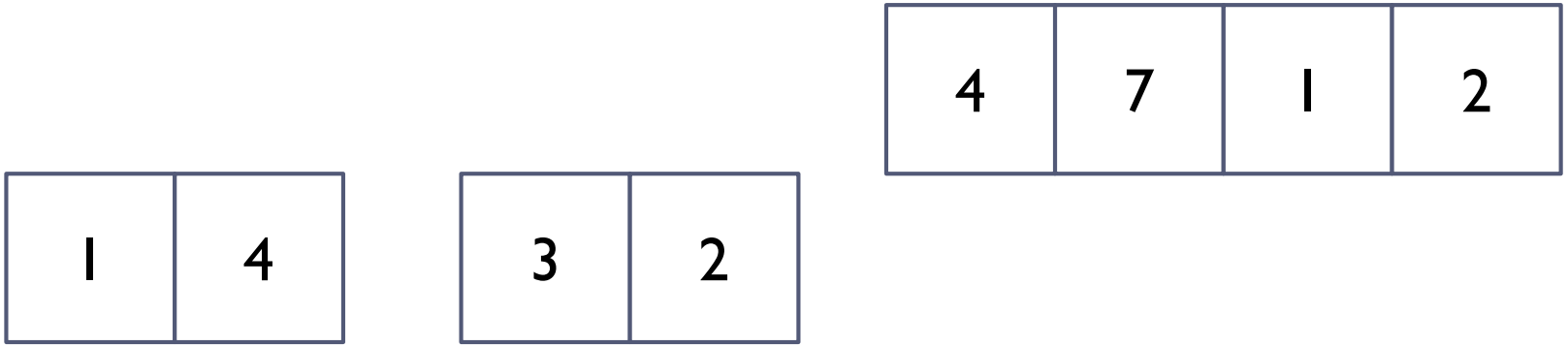
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



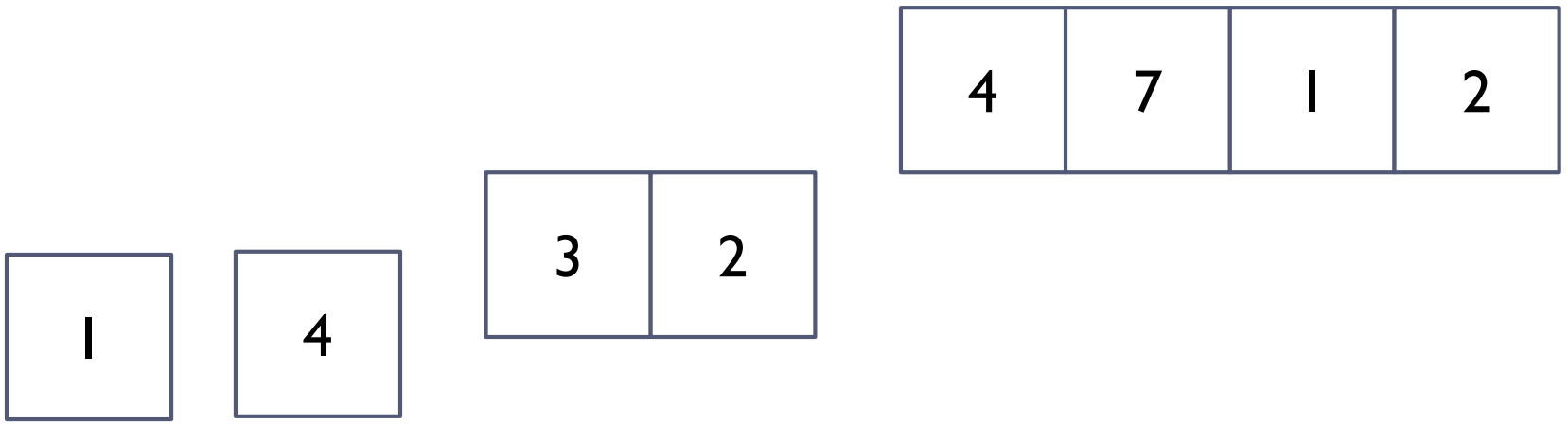
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



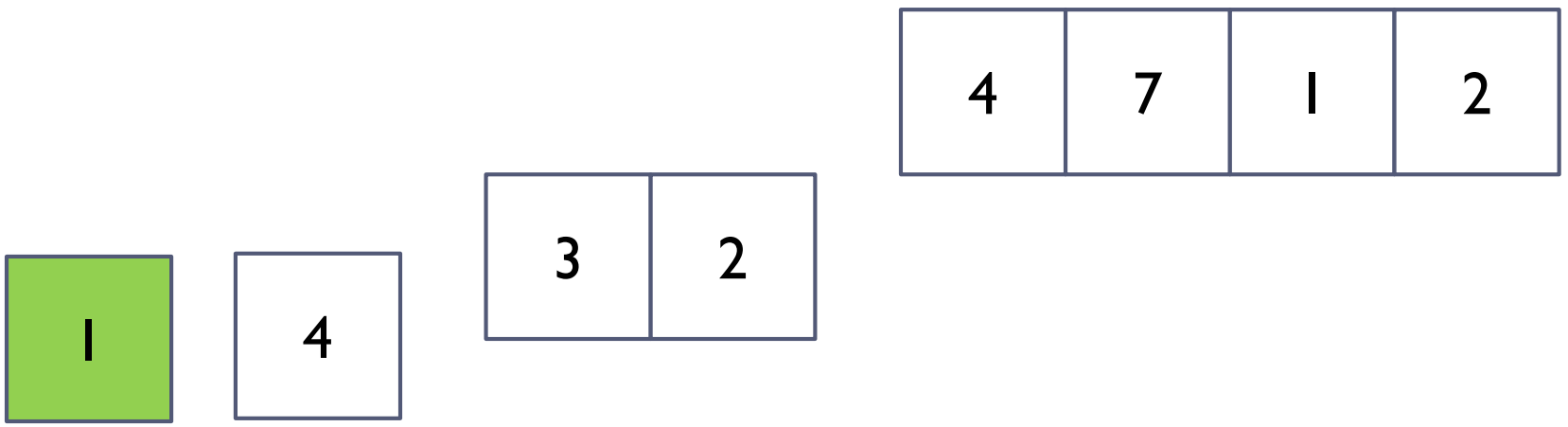
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



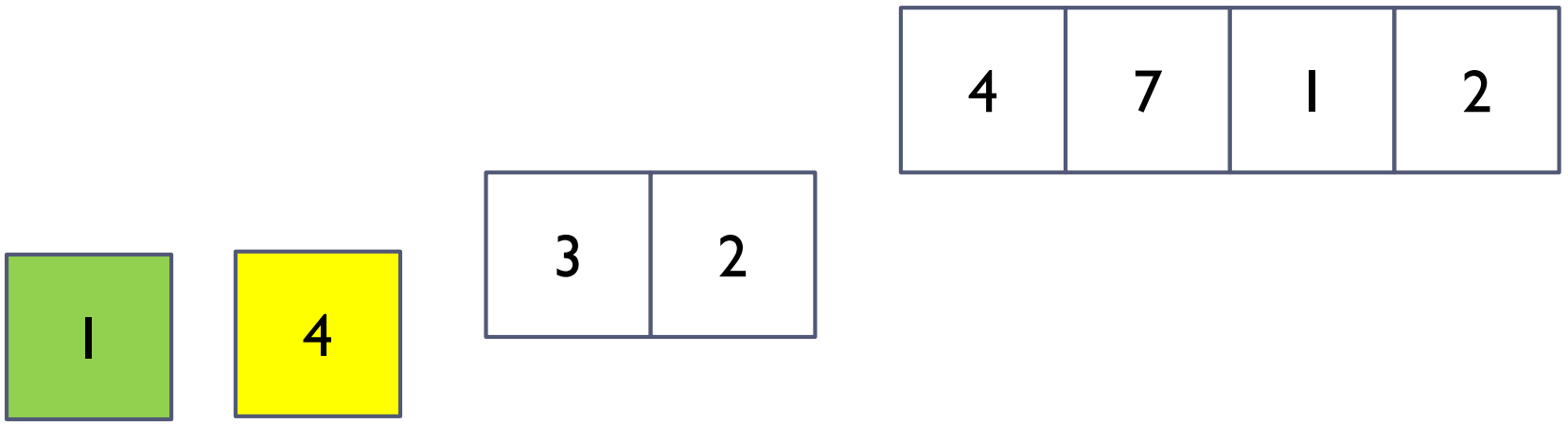
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



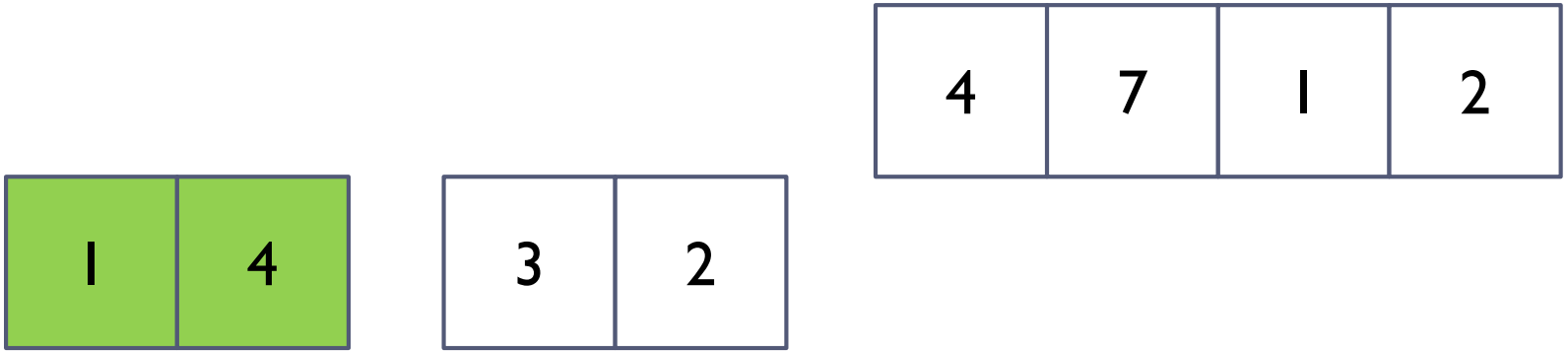
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



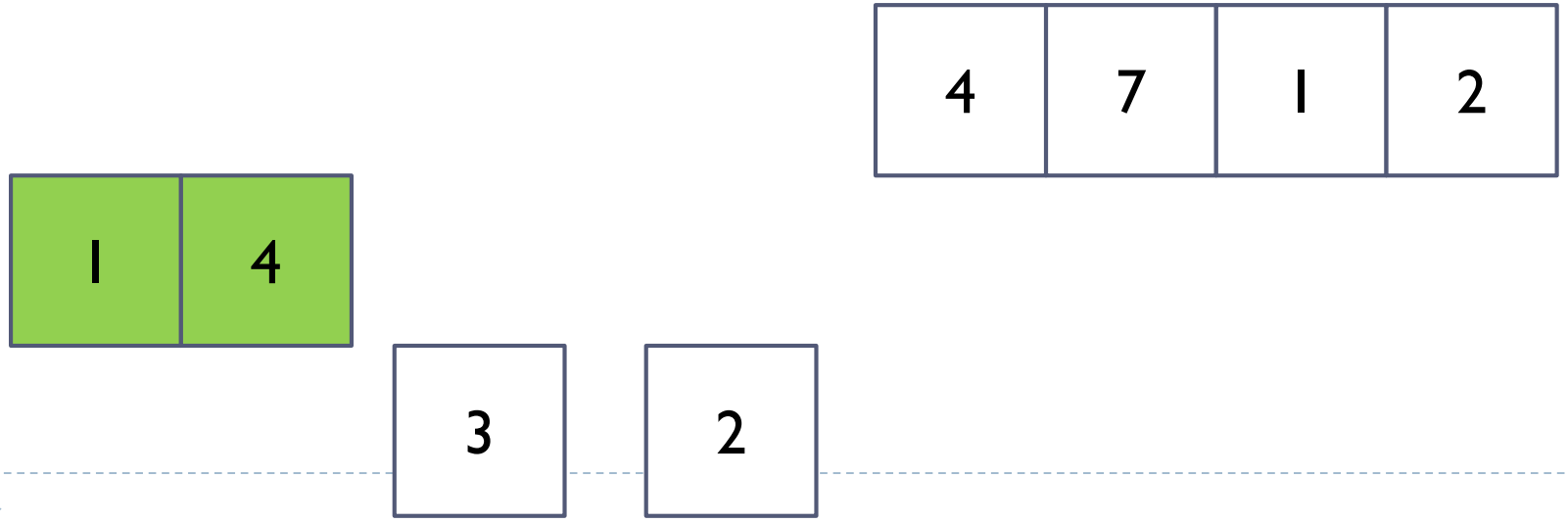
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



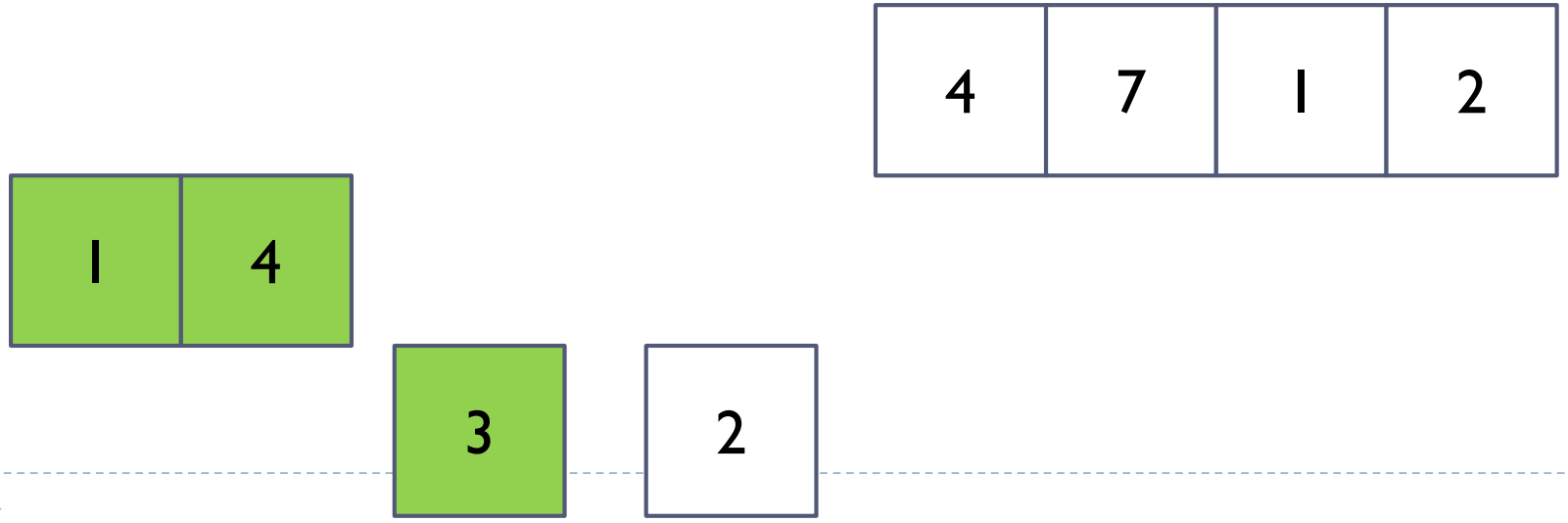
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



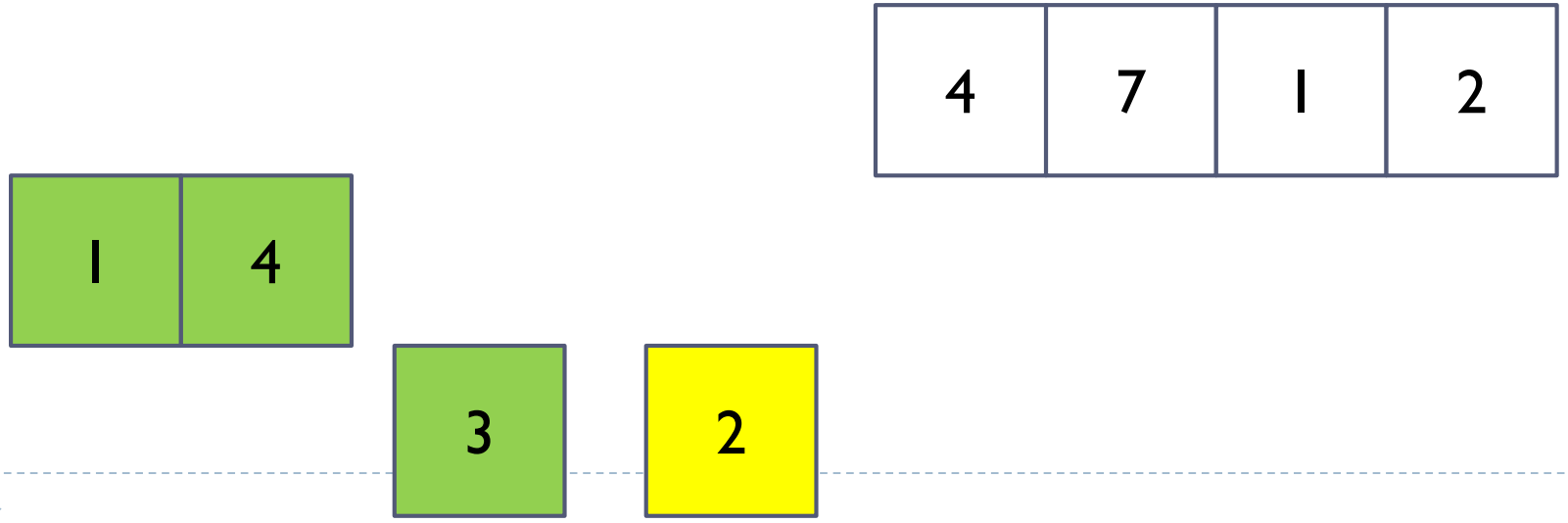
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



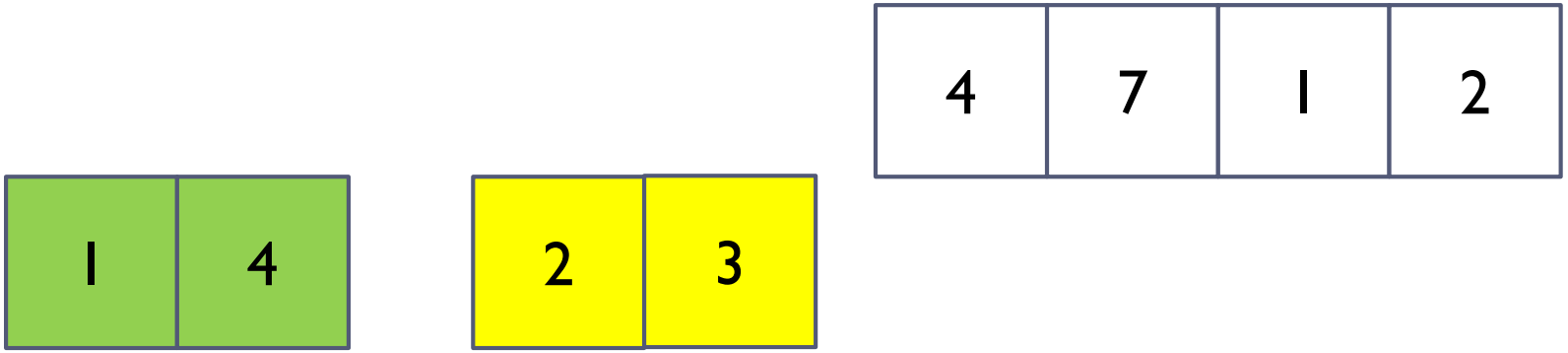
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



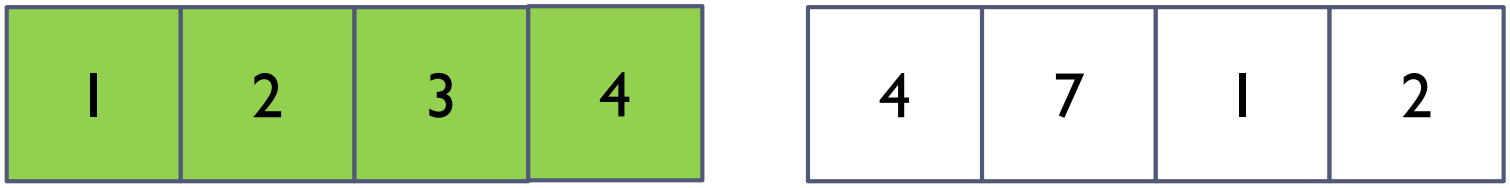
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



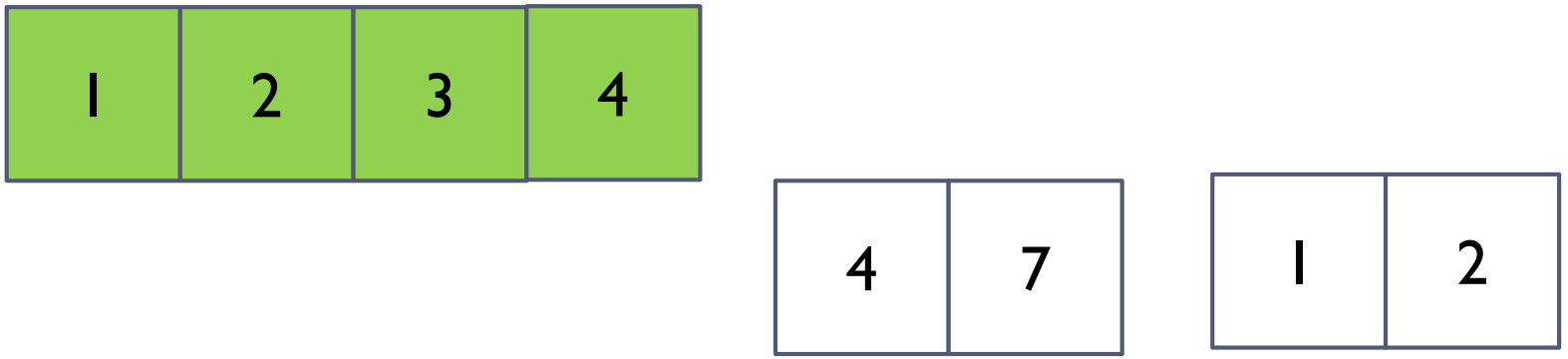
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



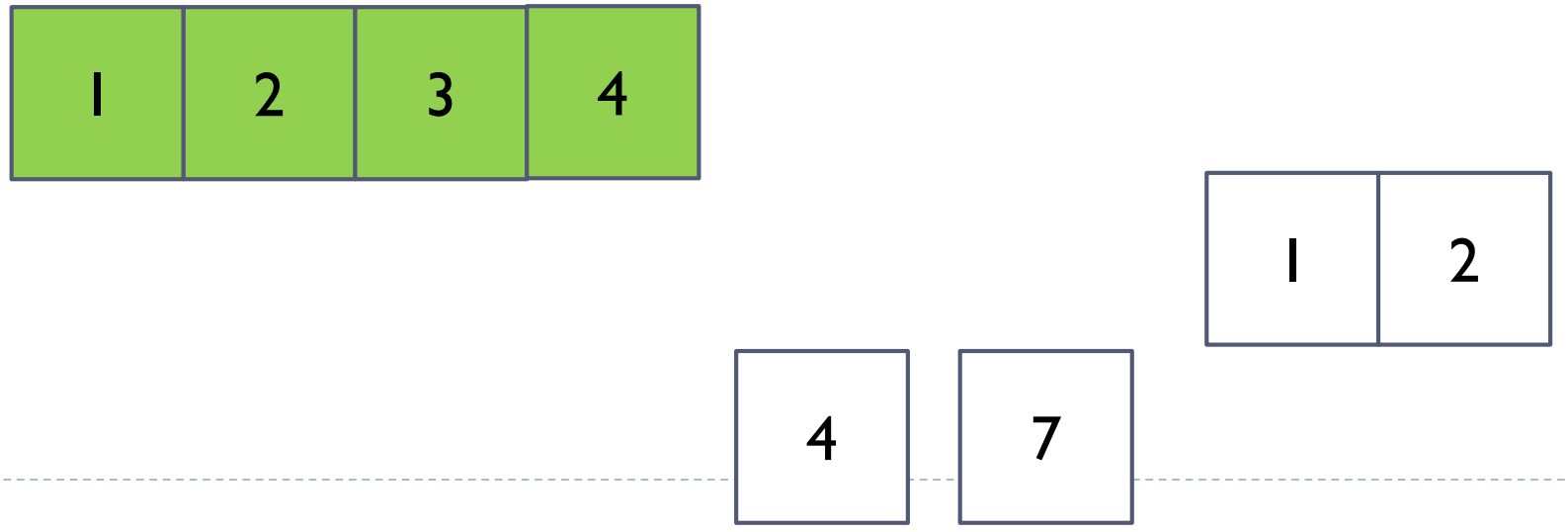
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



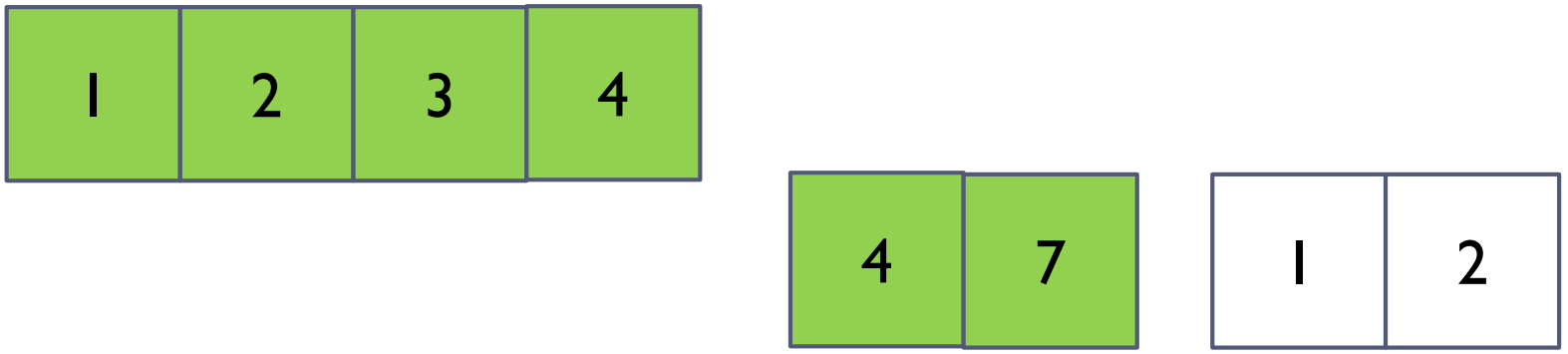
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



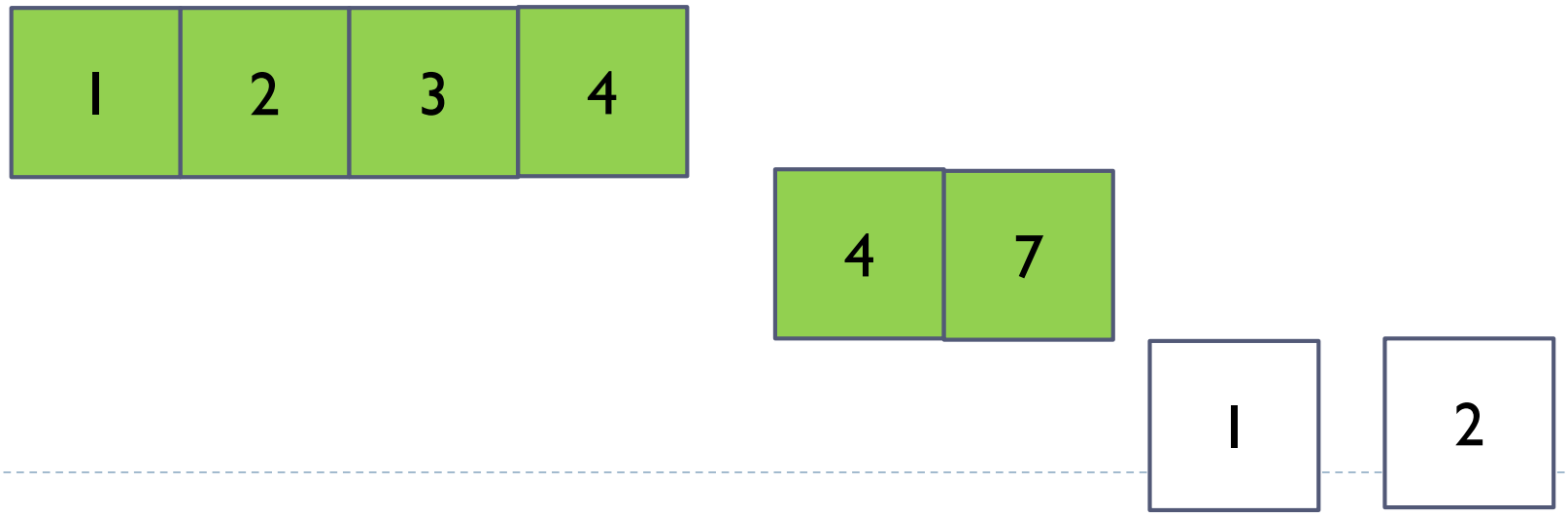
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



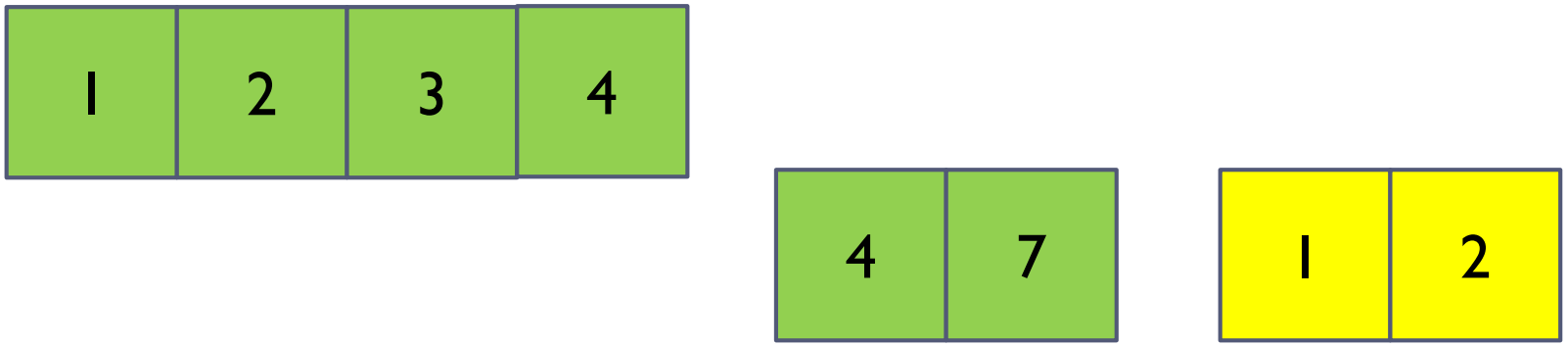
Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



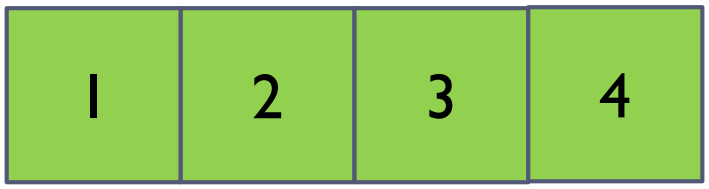
Merge Sort

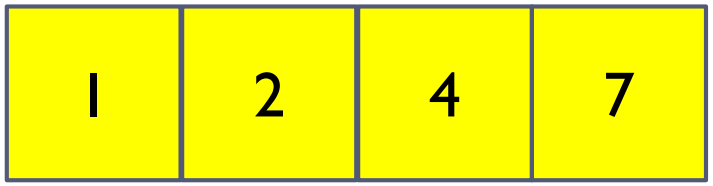
- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!



Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!





Merge Sort

- 각각의 부분을 어떻게든 정렬한다??
  - ➔ 또 다시 Merge Sort로 정렬!

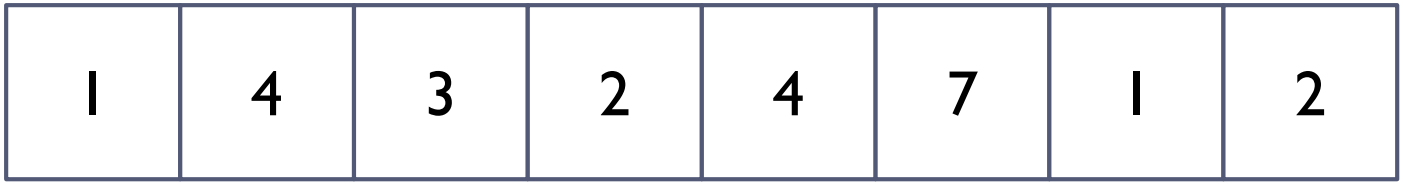
Merge Sort

- Analysis
  - 합치는 데에 걸리는 시간 ➔ O(N+M)
  - 따라서, 합치는 모든 연산 횟수를 더해보면 O(N log N)
- 전형적인 Divide & Conquer

Quick Sort

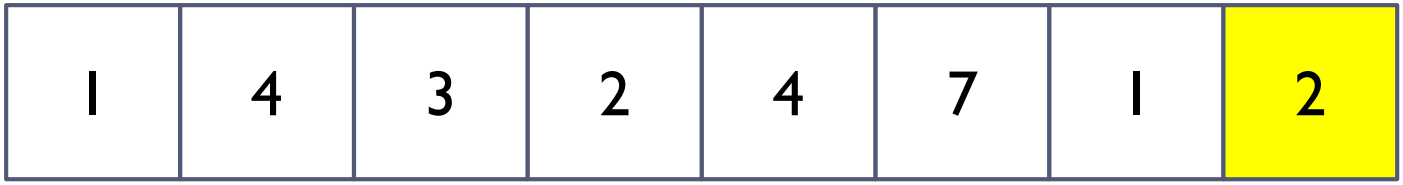
- 역시 Divide & Conquer
  - 하지만 Merge Sort와 같이 전형적인 D&C는 아님
  - 역시 엄청난 아이디어!
- 매우 중요함!!
  - 정렬 중에서 가장 빠르다고 알려져 있음
  - 하지만 역시 Case by Case. 최악의 경우 O(n^2)
  - 기본적으로 O(n log n) sort 하면 quick sort정도는 알고있어야…

Quick Sort



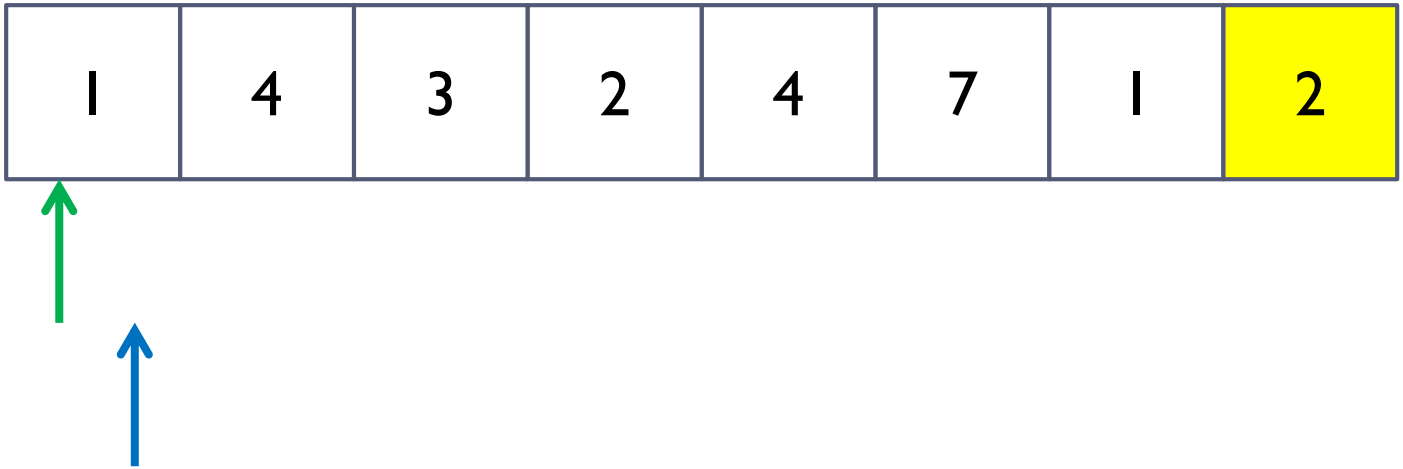
Quick Sort

- 1. Pivot을 정한다



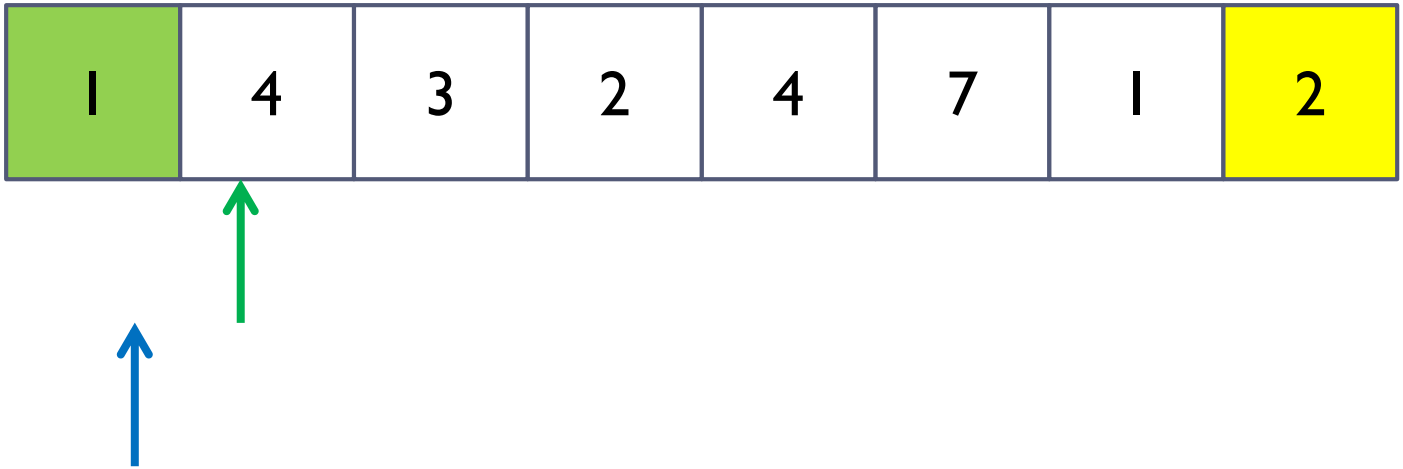
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



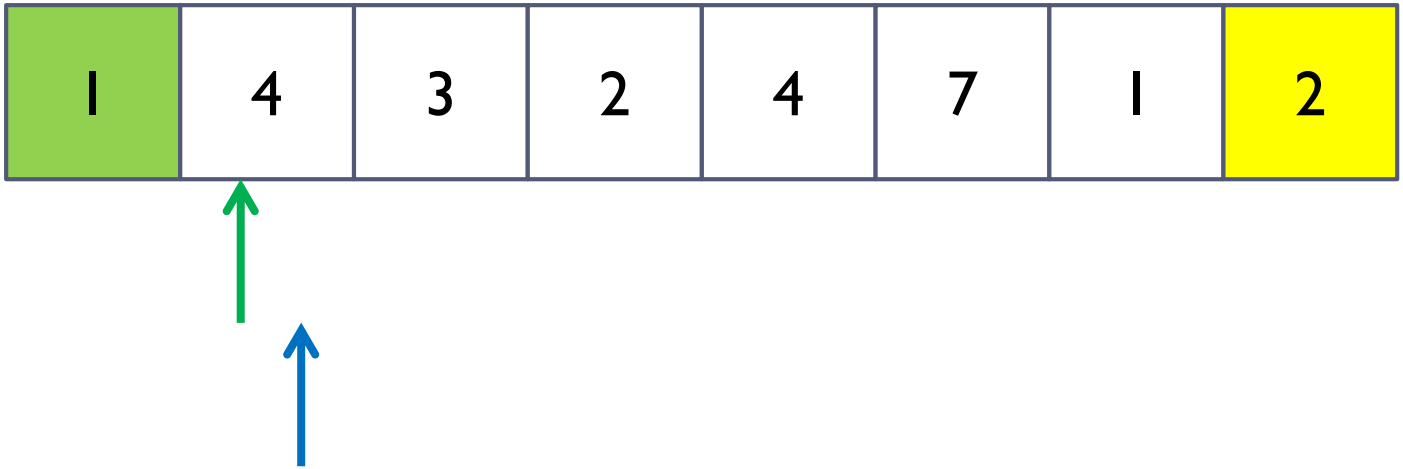
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



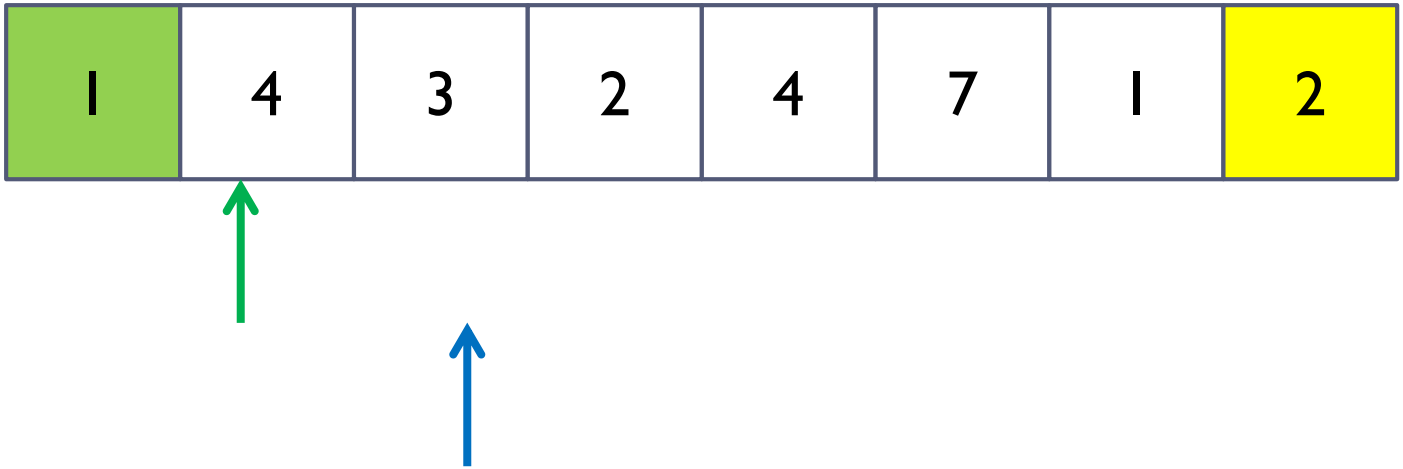
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



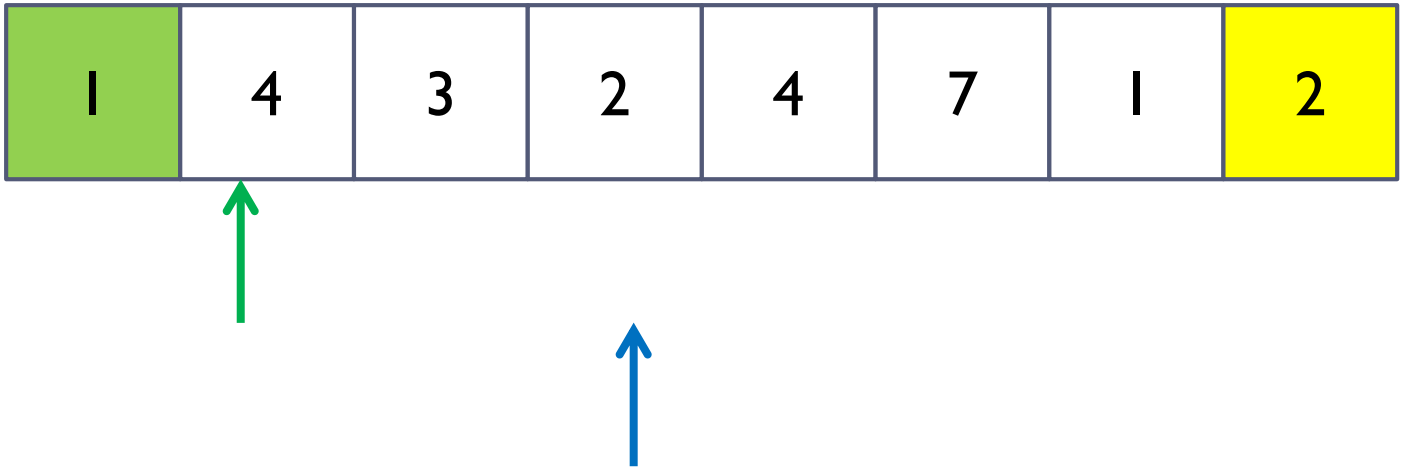
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



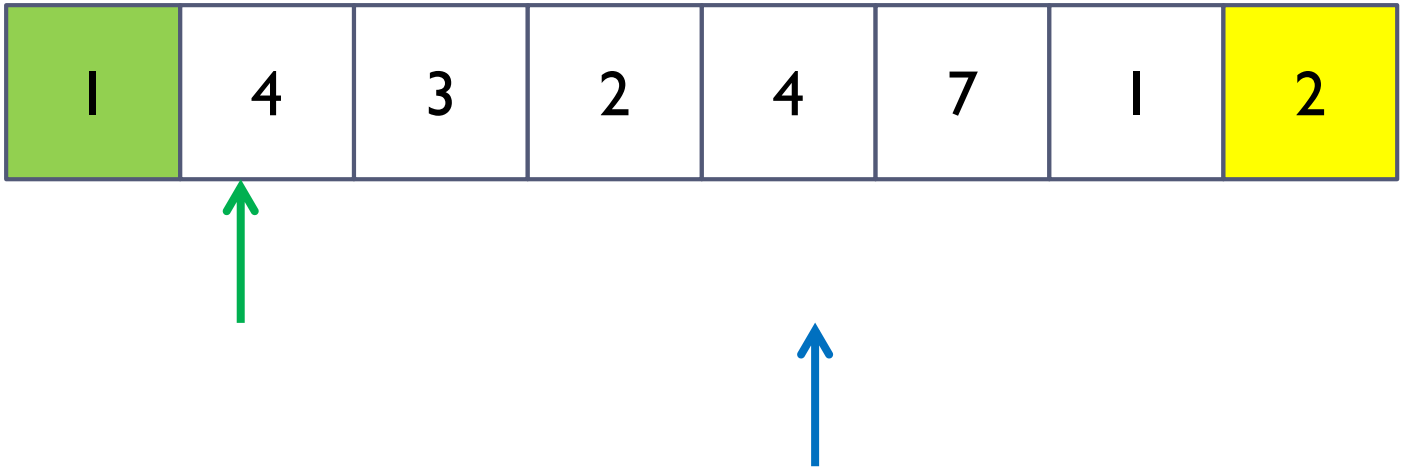
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



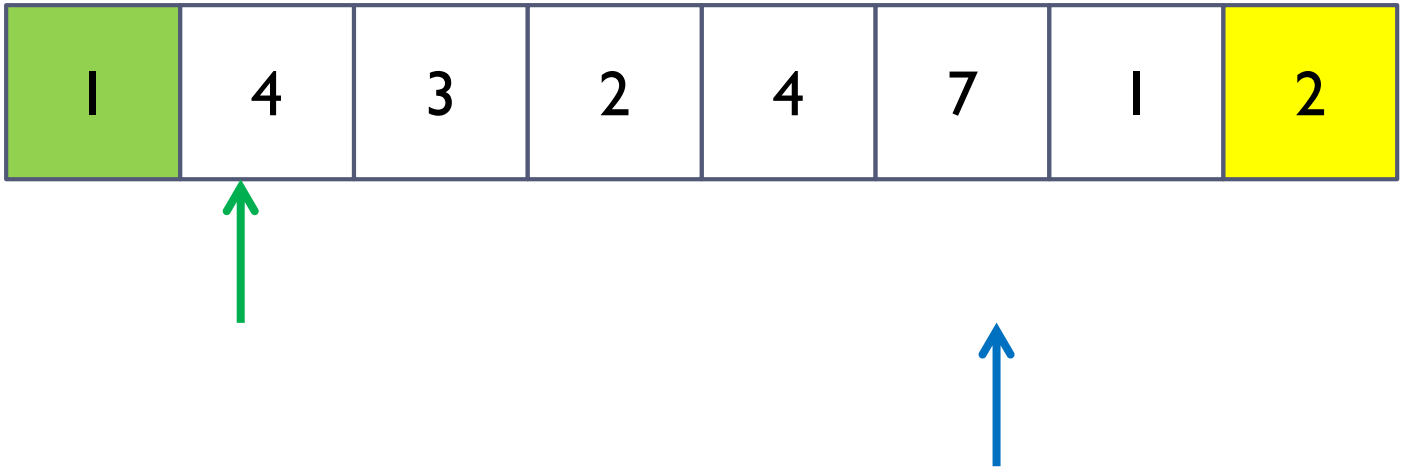
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



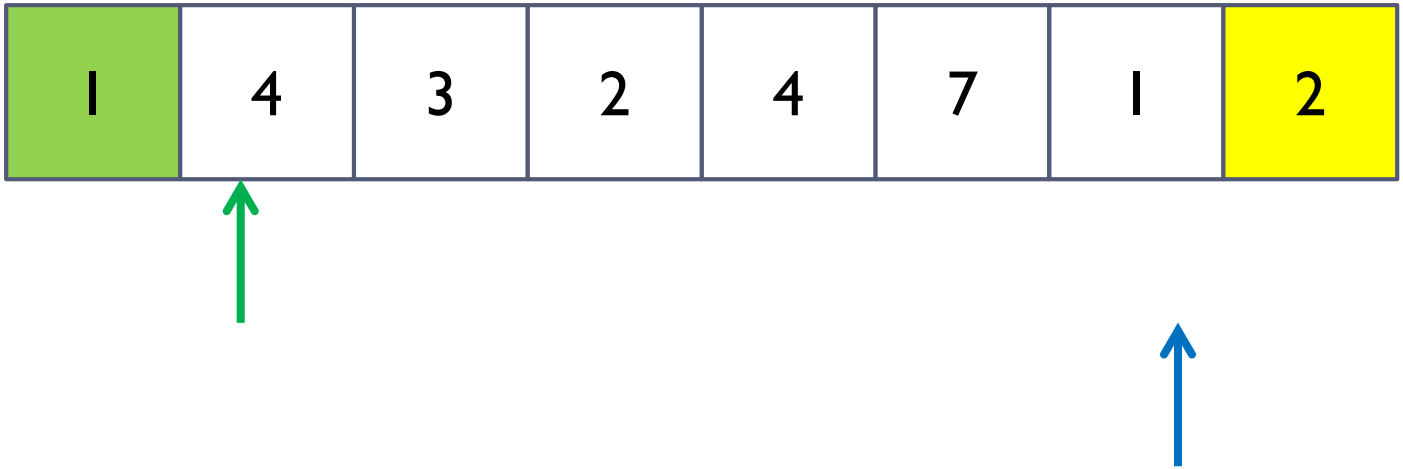
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : I ~ p-I 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



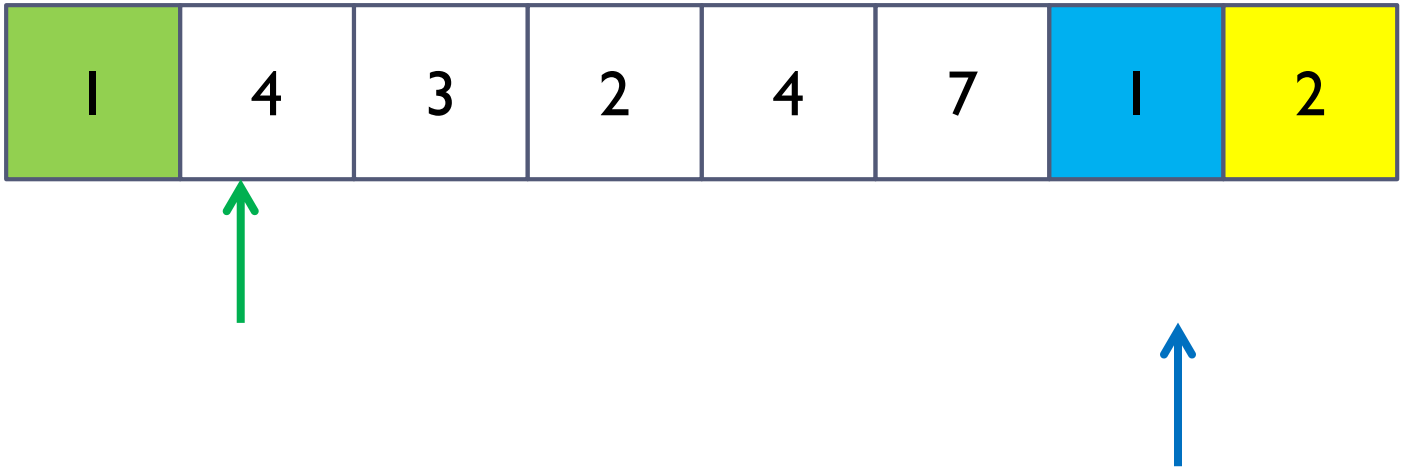
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



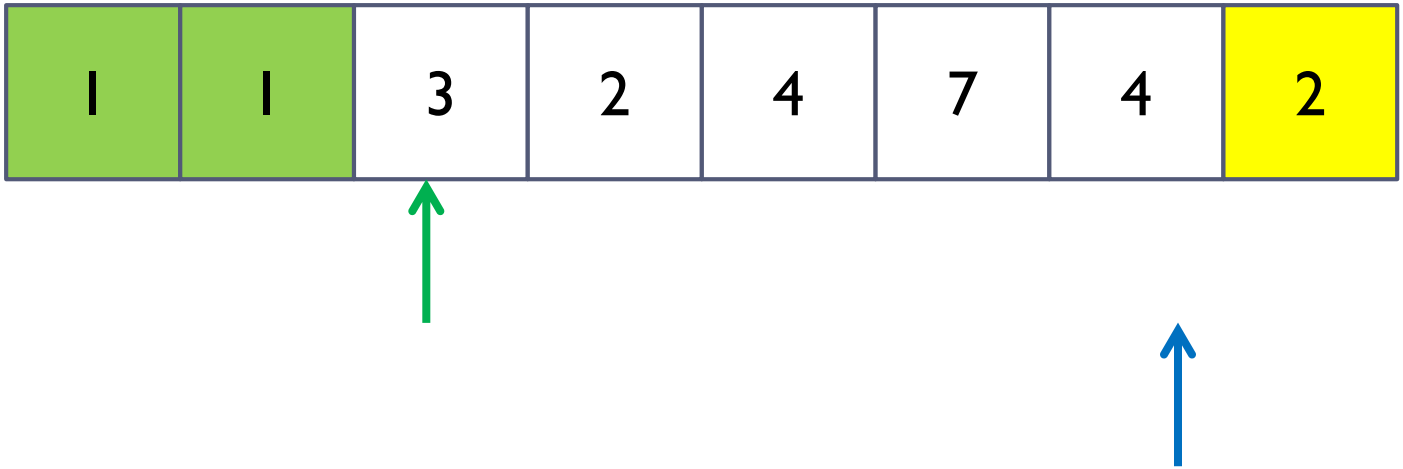
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



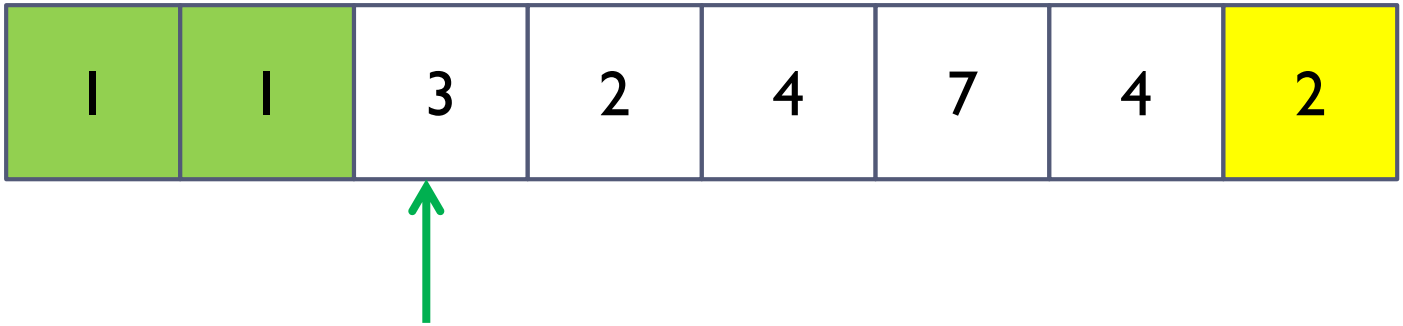
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



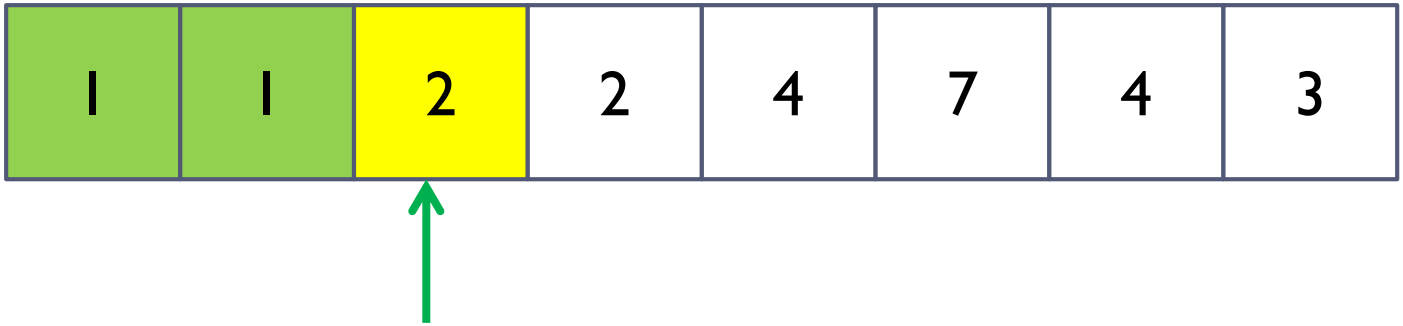
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



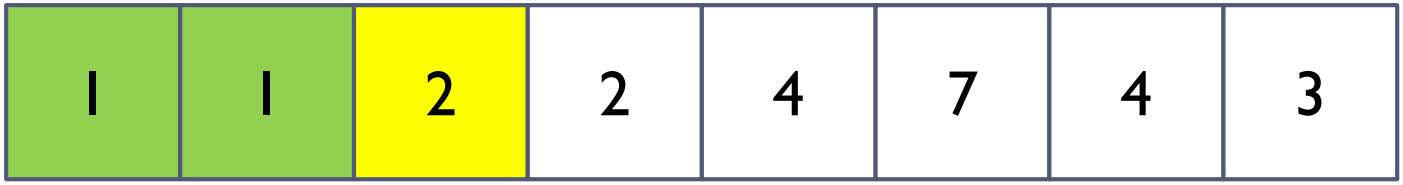
Quick Sort

- 2. Pointer 2개를 만든다. (p, q)
  - P : 1 ~ p-1 까지는 모두 pivot보다 작음
  - Q : Data[q]가 pivot보다 작음



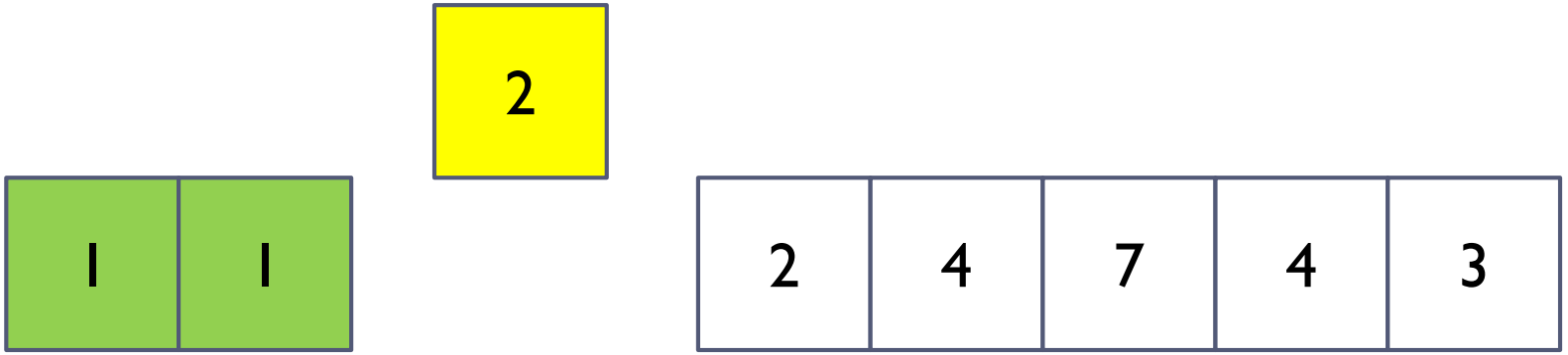
Quick Sort

- 3. 두 부분을 나눈다



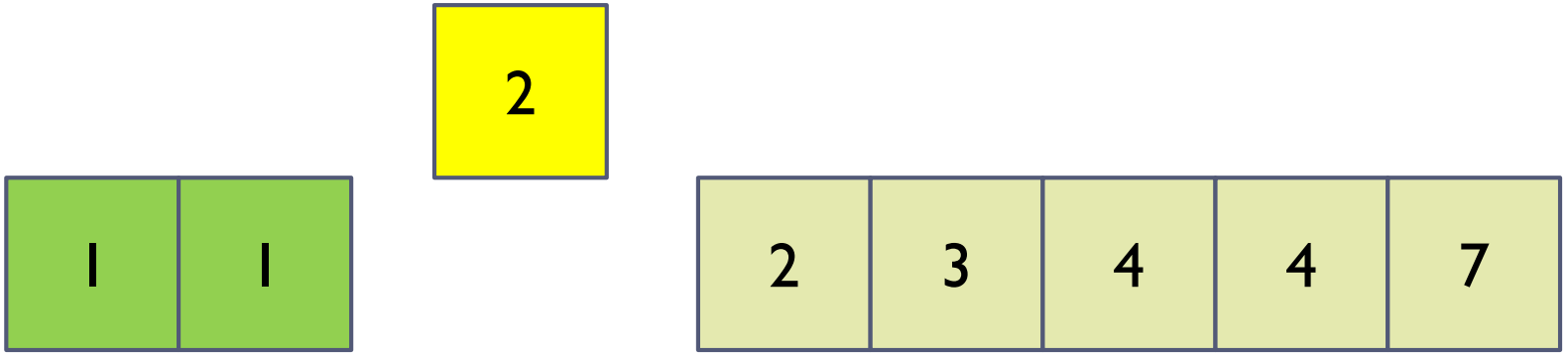
Quick Sort

- 3. 두 부분을 나눈다



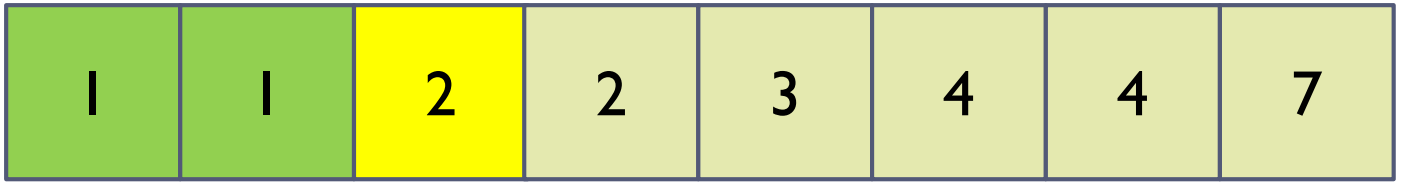
Quick Sort

- 3. 각 부분을 어떻게든 sort한다



Quick Sort

- 4. 합친다



Quick Sort

- Analysis
  - Pivot을 정하고 그 위치를 정확히 정하는데 O(N)
  - 계속해서 반 씩 나누는 것을 관찰하면 O(N log N)
- But… 최악의 경우에는?
  - 반 씩 나누어지는 것이 아니라 아예 나누어지지 않을 경우
  - O(N^2)
  - 이를 막기 위한 여러 가지 방법이 있습니다.