📩 분할 정복 알고리즘

> Divide and conquer algorithm
>
> 그대로 해결할 수 없는 문제를 작은 문제로 분할하여 해결하는 방법
>
> 주로 재귀 함수(recursive function)를 통해 구현됨

분할 정복 알고리즘 수행 순서

1. Divide (분할)

문제가 분할이 가능한 경우, 2개 이상의 문제로 나눈다.

나누어진 문제가 여전히 분할이 가능하다면, 또 다시 Divde를 수행한다.

2. Conquer (정복)

분할 된 사례들을 각각 정복(conquer)한다.

이 때, 사례가 충분히 작지 않다면 재귀를 사용해서 문제를 정복한다.

3. Combine

작은 사례들에 대한 해답을 통합하여 원래 사례의 해답을 구한다.

장단점

장점

- 문제를 나눔으로써 어려운 문제를 작은 것부터 해결할 수 있음
- 병렬적으로 문제를 해결할 때 주로 사용됨

단점

- 함수를 재귀적으로 호출함에 따라 오버헤드가 발생할 수 있음
- 스택에 다양한 데이터를 보관하고 있어야하므로 스택 오버플로우가 발생하거나 과도한 메모리 사용을 하게 됨

분할 정복 알고리즘의 응용

1. 병합 정렬(Merge Sort)

> 리스트를 반으로 나눈뒤, 각각의 리스트의 요소가 0 또는 1이 될 때까지 계속해서 반으로 나누고 병합하는 과정에서 정렬하는 방법

시간 복잡도

- 최악의 경우 : O(nlogn)
- 최선의 경우 : O(nlogn)
- 평균 : O(nlogn)

공간 복잡도

- O(n)

알고리즘 순서

1. 정렬할 데이터 집합의 크기가 0 또는 1이면 이미 정렬된 것으로 본다. 그렇지 않은 경우,
2. 데이터 집합을 반으로 나눈다. (Divide)
3. 원래 같은 집합에서 나뉘어져 나온 데이터 집합 둘을 병합(Combine)하여 하나의 데이터 집합으로 만든다. 단, 병합할 때 데이터 집합의 원소는 순서에 맞춰 정렬한다. (Conquer)
4. 데이터 집합이 다시 하나가 될 때까지 3번 단계를 반복한다.

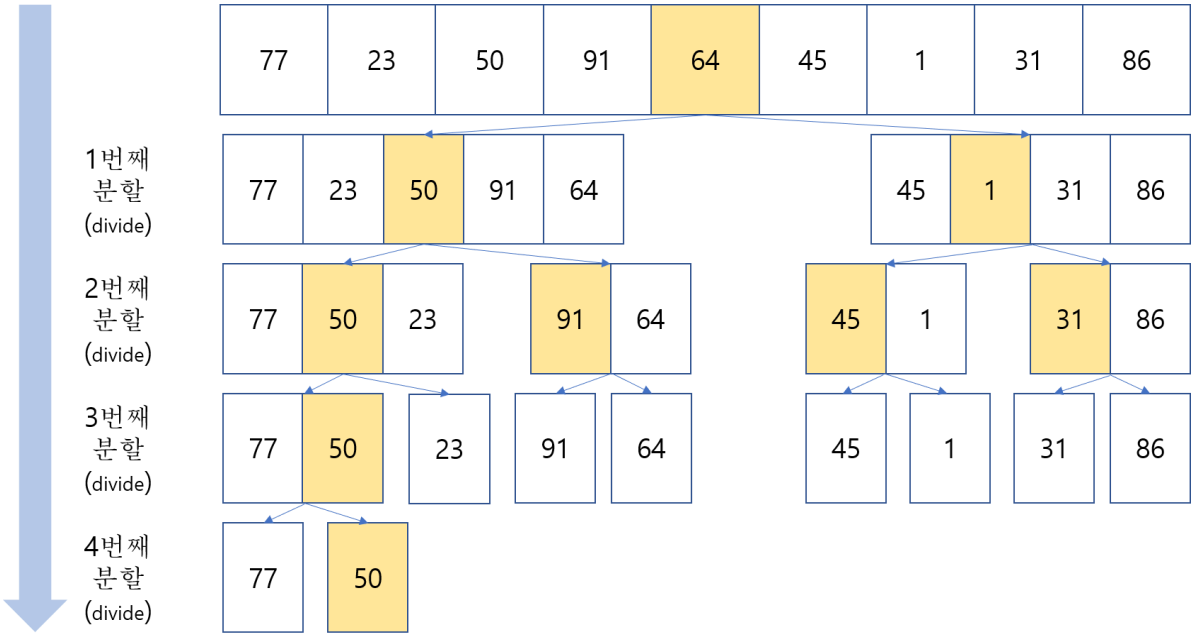
그림과 함께 이해하기

0. 데이터 집합 준비

77	23	50	91	64	45	1	31	86

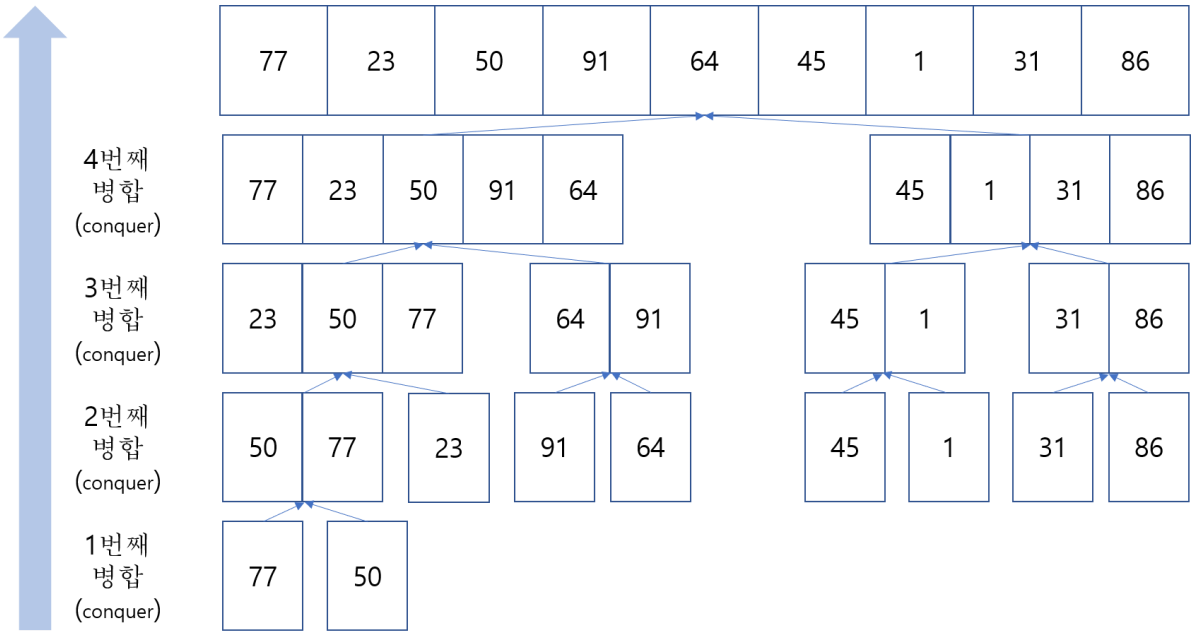
1. 데이터 집합을 반으로 나누고, 같은 집합에서 나뉘어져 나온 데이터 집합 둘을 병합하여 하나의 데이터 집합으로 만든다. (Divide)

- 색을 칠한 부분은 나누는 기준이 되는 부분



2. 같은 집합에서 나뉘어져 나온 데이터 집합 둘을 병합하여 하나의 데이터 집합으로 만든다. (Conquer & Combine)

    - 주의할 점 : 병합 시 데이터 집합의 원소는 순서에 맞춰 정렬



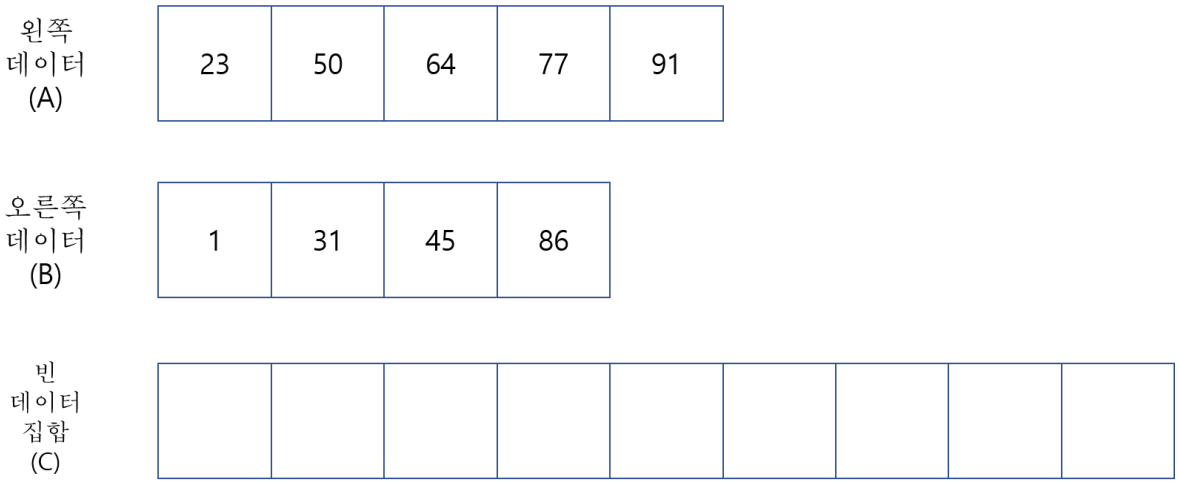
Conquer & Combine 과정 상세 설명

1. 두 데이터 집합의 크기의 합만큼의 크기를 가지는 빈 데이터 집합을 만든다.
2. 두 데이터 집합의 첫 번째 요소들을 비교하여 작은 요소를 빈 데이터 집합에 추가한다. 그리고 새 데이터 집합에 추가한 요소는 원래 데이터 집합에서 삭제한다.
3. 원래 두 데이터 집합의 요소가 모두 삭제될 때까지 2번 단계를 반복한다.

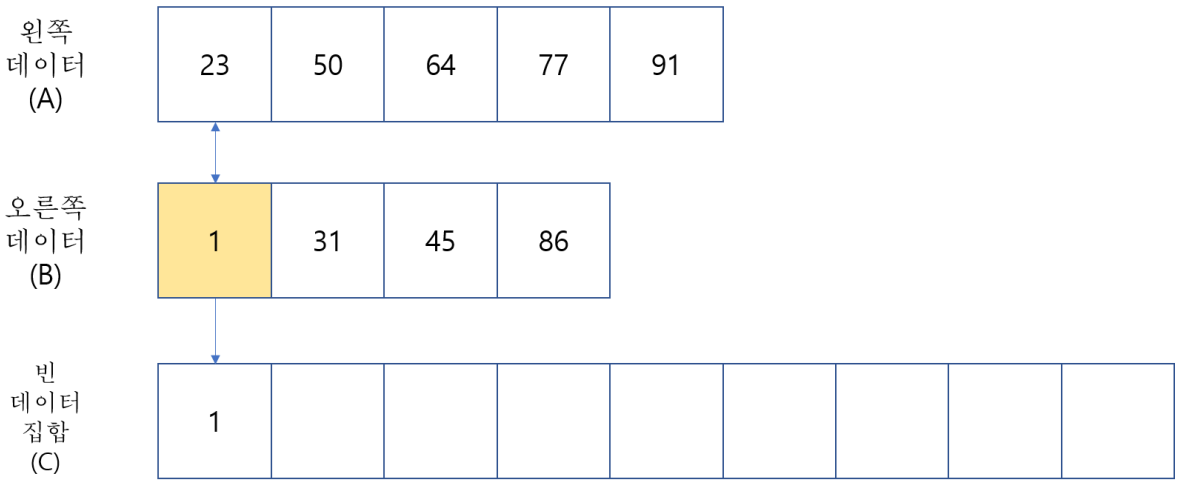
그림과 함께 이해하기

> 위의 알고리즘에서 4번째 병합(conquer)을 할 때 과정을 상세히 나타냄
>
> 따라서 각각의 데이터 배열은 1, 2, 3번째 병합 이후 정렬된 상태

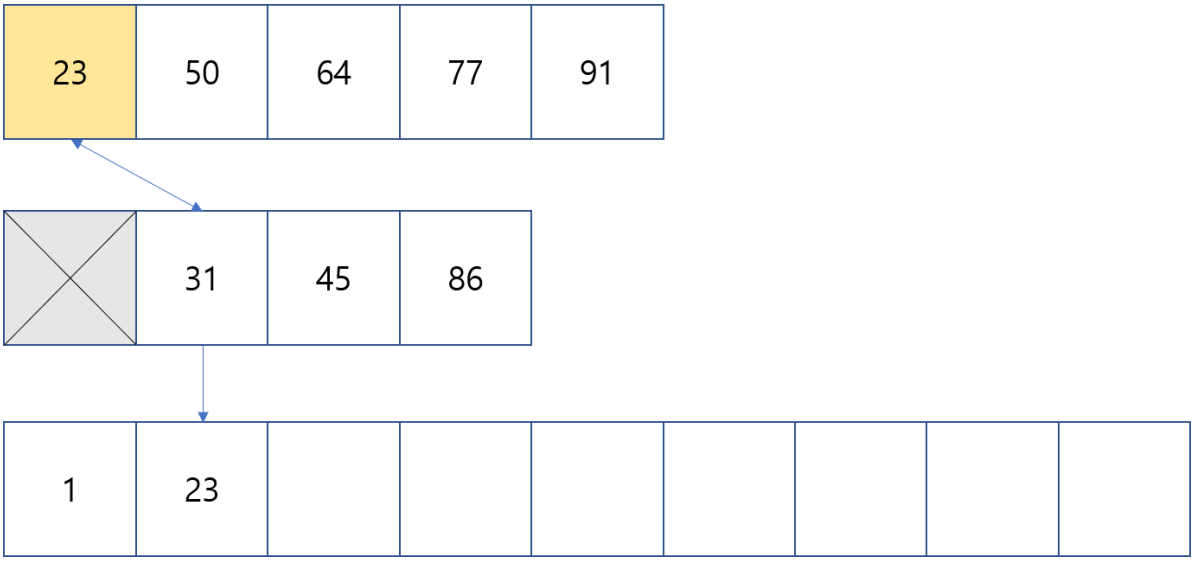
1. 두 데이터 집합의 크기의 합만큼의 크기를 갖는 빈 데이터 집합을 만든다.



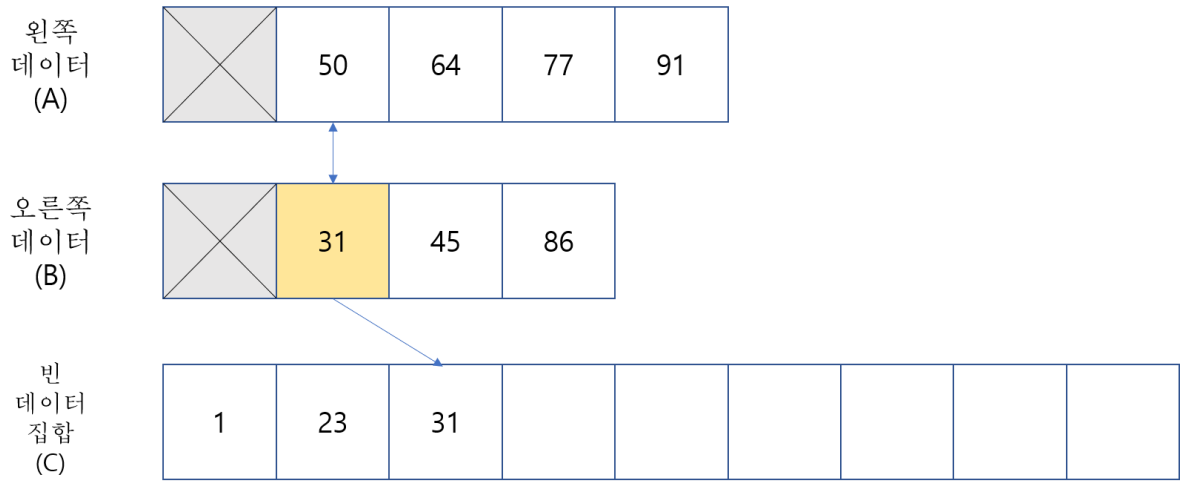
2. 첫 번째 요소들을 비교해서 작은 요소를 빈 데이터 집합에 추가



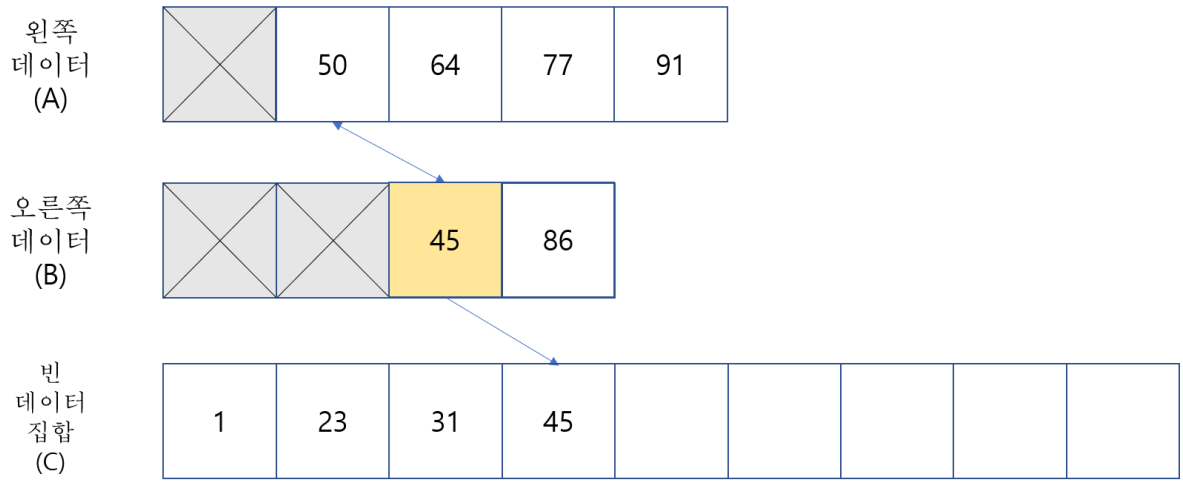
2-1. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



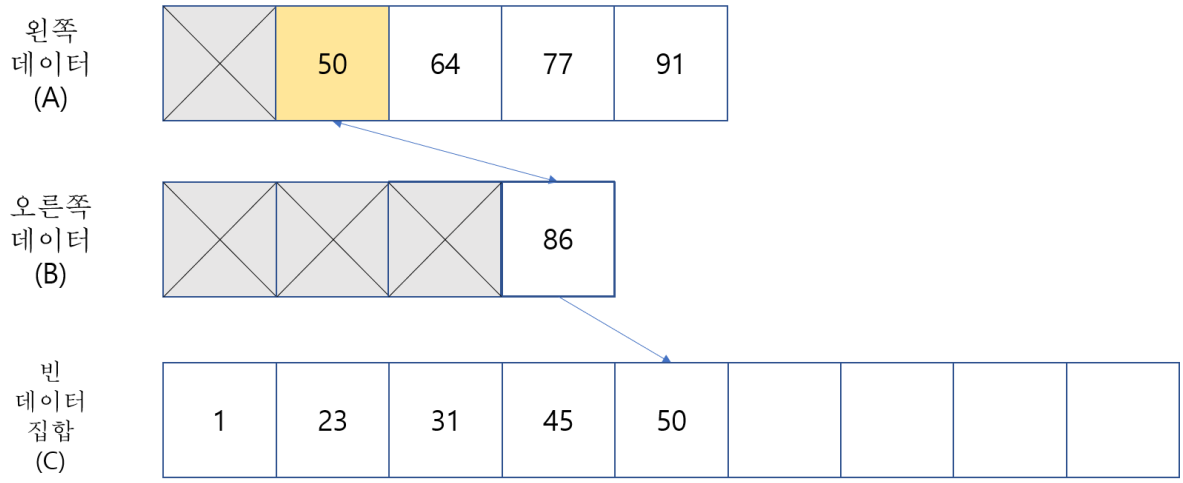
2-2. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



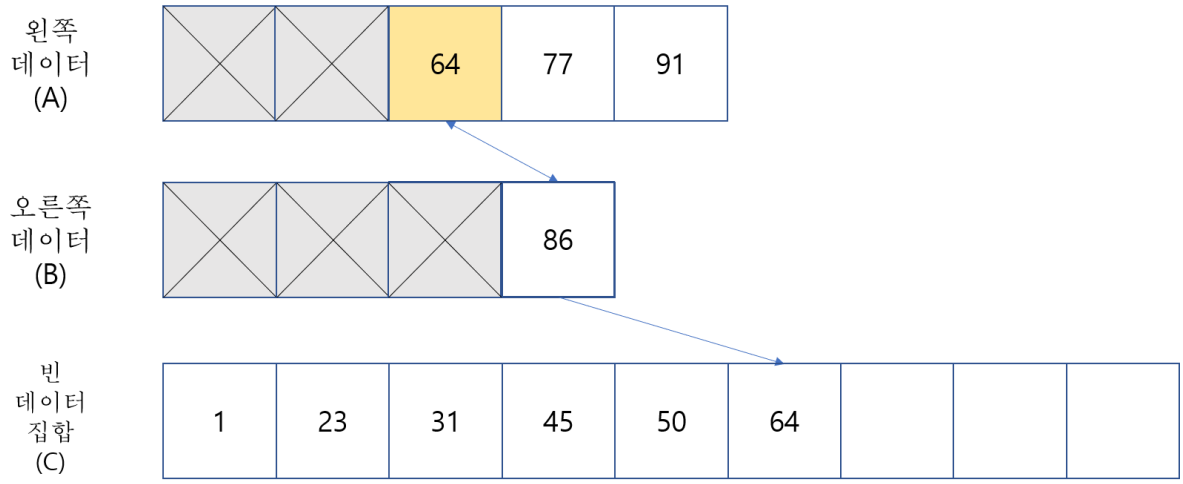
2-3. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



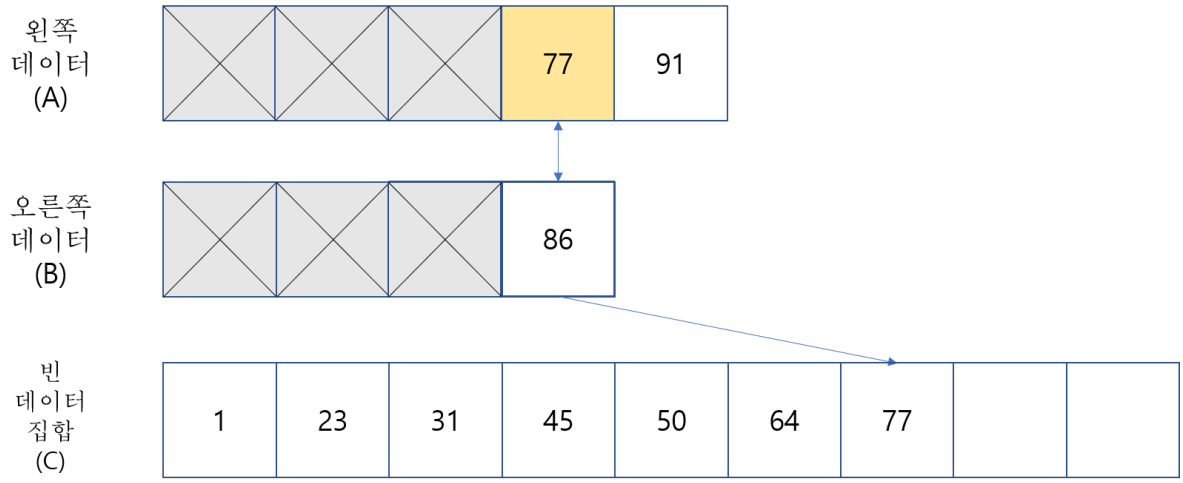
2-4. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



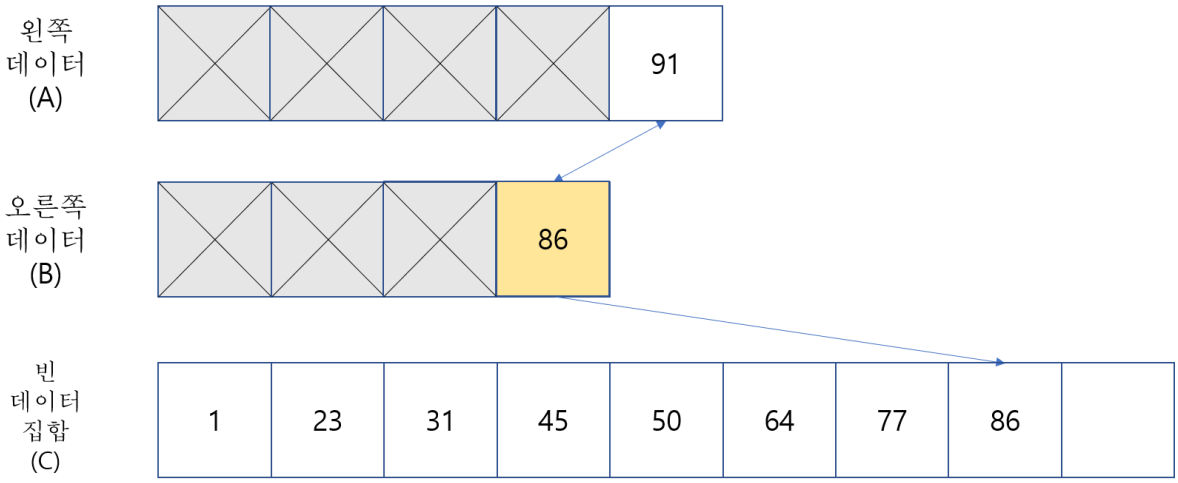
2-5. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



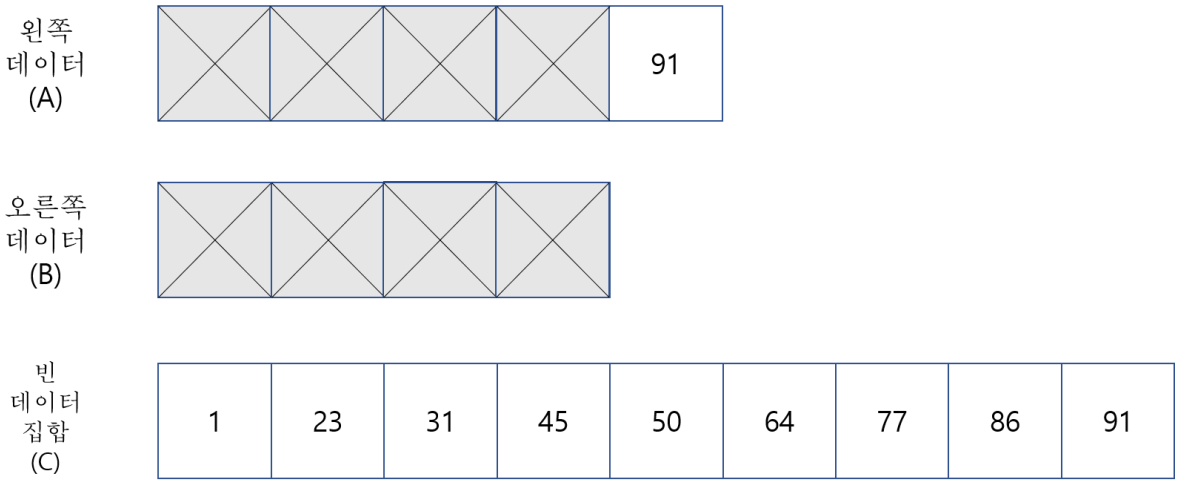
2-6. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



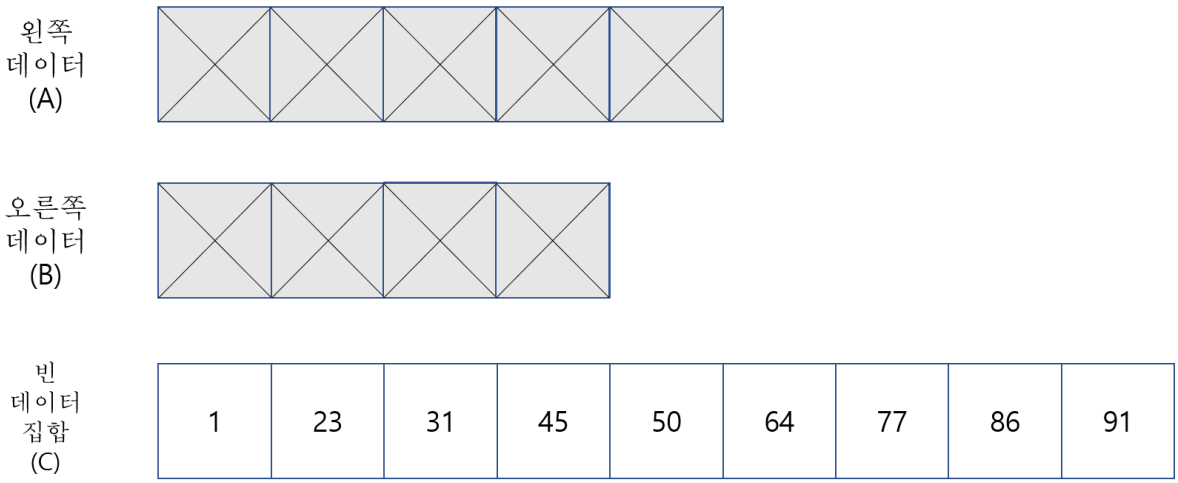
2-7. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제 후, 각각의 데이터 집합의 첫번째 요소끼리 비교



2-8. 요소가 한 개만 남아있으므로, 해당 요소를 C 데이터 집합에 추가



2-9. C 데이터 집합에 추가된 요소는 기존의 데이터 집합에서 삭제



2. 퀵 정렬 (Quick Sort)

> 임의의 원소를 pivot(피벗)으로 선정하고, 피벗을 기준으로 요소들을 분리하는 과정을 반복하여 정렬하는 방법

시간 복잡도

- 최악의 경우 : O(n^2)
- 최선의 경우 : O(nlogn)
- 평균 : O(nlogn)

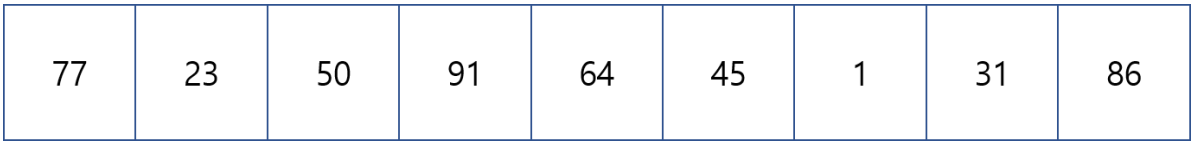
알고리즘 순서

1. 정렬할 리스트의 크기가 0 또는 1이면 이미 정렬된 것으로 본다. 그렇지 않은 경우,
2. 리스트 가운데서 하나의 원소를 임의로 고른다. 이 때 고른 원소를 피벗(pivot)이라고 한다.
3. 피벗 앞에는 피벗보다 값이 작은 모든 원소들이 오고, 피벗 뒤에는 피벗보다 값이 큰 모든 원소들이 오도록 피벗을 기준으로 리스트를 둘로 나눈다. (Divide & Conquer)

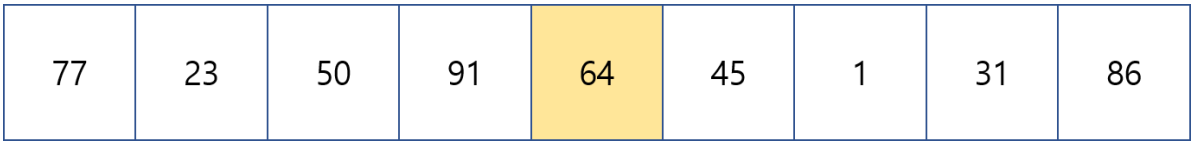
   분할을 마친 후 피벗은 움직이지 않는다.
4. 분할된 두 개의 작은 리스트에 대해 리스트의 크기가 0이나 1이 될 때까지 재귀적으로 이 과정을 반복한다.

그림과 함께 이해하기

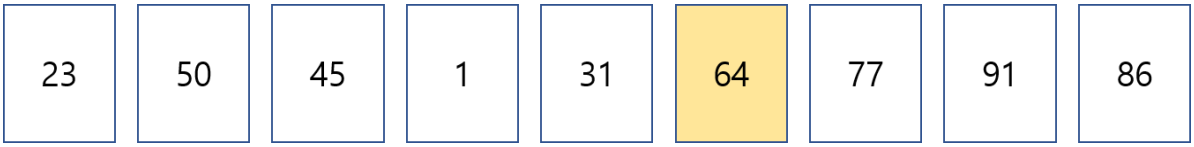
0. 데이터 집합 준비



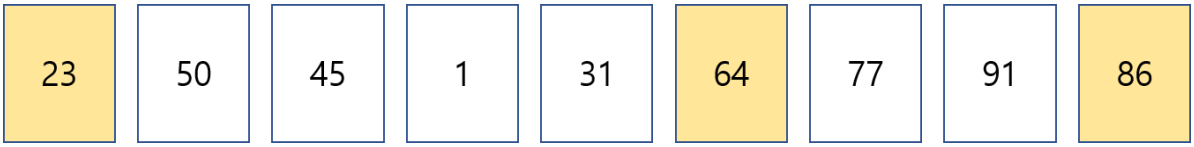
1. pivot 선정 (임의의 원소 선정) : 64
   - 이 때, pivot의 위치는 변하지 않는다.



2. pivot을 기준으로 divide



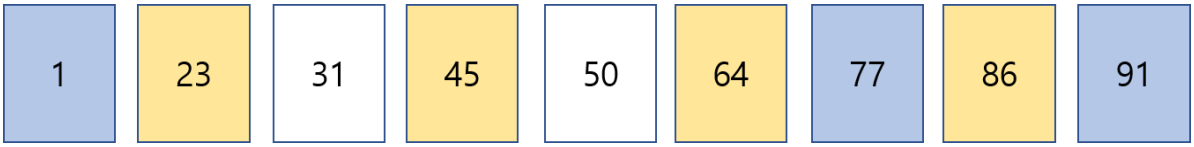
3. pivot 선정 : 23, 86 (임의로 선정)
   - 이 때, pivot의 위치는 변하지 않는다.



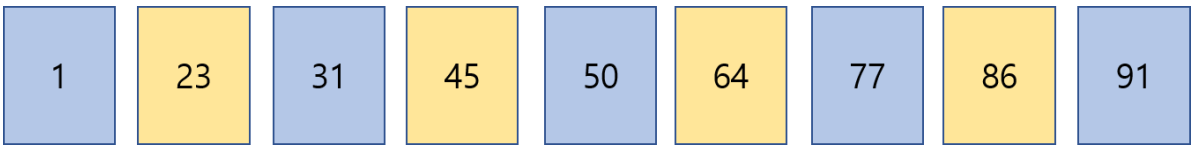
4. pivot을 기준으로 divide
   - 1, 77, 91은 정렬 완료 (파란색으로 표시)

5. pivot 선정 : 45 (임의로 선정)
   - 이 때, pivot의 위치는 변하지 않는다.

6. pivot을 기준으로 divide



7. 정렬 완료



3. 거듭 제곱 (Exponentiation)

기존의 거듭 제곱

- 자신을 n번 곱해야 하므로 O(n)의 시간이 소요

  ex)

$$C^8 = C * C * C * C * C * C * C * C$$

Divide and Conquer을 사용한 거듭 제곱

- C의 n승에서, C^2의 n//2승으로 바꾸어 계산하므로 O(nlogn)의 시간 소요

  ex)

$$C^8 = C^4 * C^4$$

  - 지수가 짝수인 경우 : n(지수)를 반으로 나눠서 곱함 (n//2 승)

    ex)

$$C^n = C^{n/2} * C^{n/2}$$

- 지수가 홀수인 경우 : n(지수)에서 1을 빼고 반으로 나눠서 곱하고 자신을 한번 더 곱해줌

  ex)

$$C^n = C^{(n-1)/2} * C^{(n-1)/2} * C$$

4. 그 외

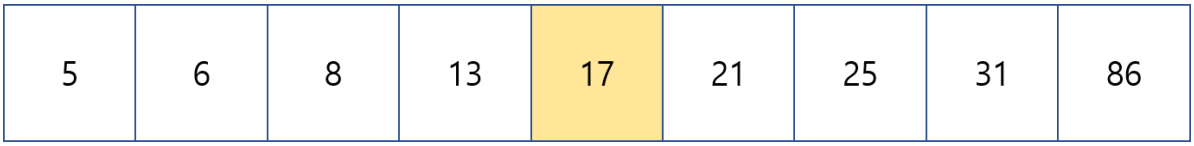
> 이분 검색, 최댓 값 찾기 등에 사용될 수 있음

1) 이분 검색

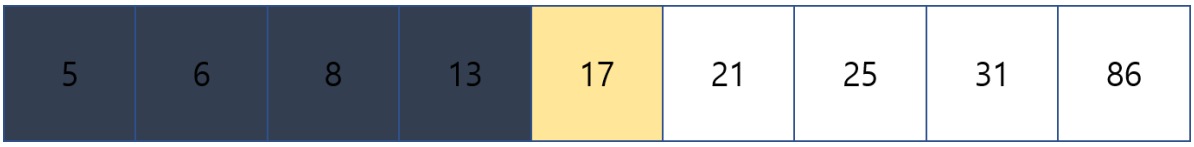
- ex) 10개의 **정렬된 배열**에서 `21`이라는 값을 가진 인덱스 찾기

0. 정렬된 배열

1. 중간 값(`17`)을 기준으로 2개의 리스트로 분할



2. 찾고자 하는 값(`21`)이 중간 값(`17`)보다 크므로 왼쪽은 사용 X



3. 중간 값(`25`)을 기준으로 2개의 리스트로 분할

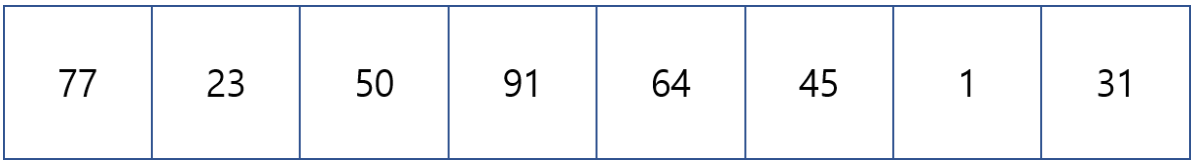
4. 중간 값(`25`)보다 찾고자 하는 값(`21`)이 더 작으므로 오른쪽은 사용 X

5. 값 찾기

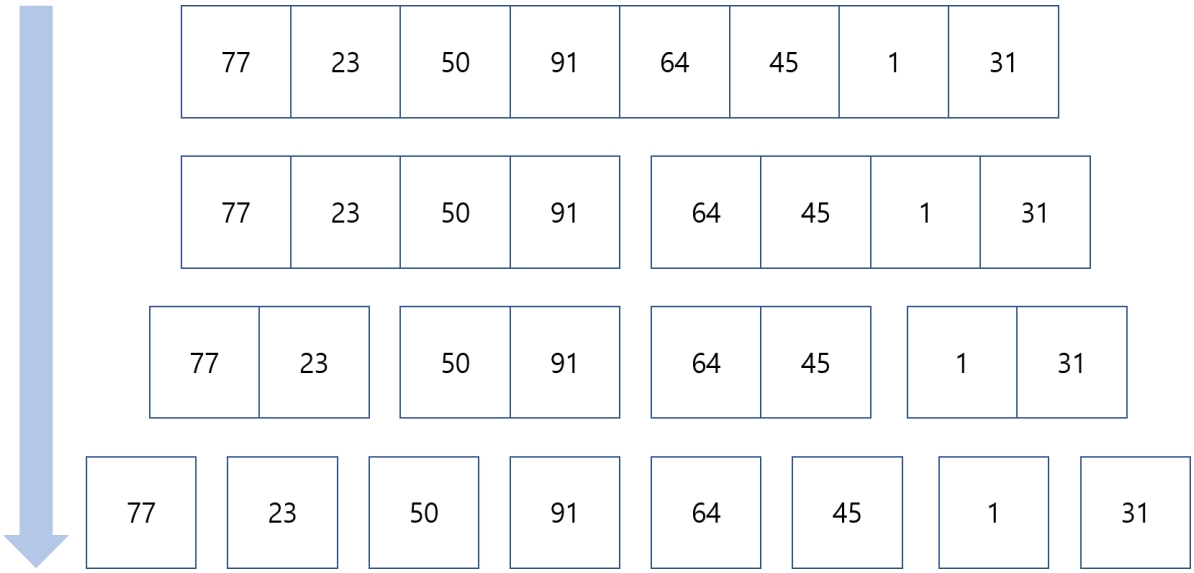
2) 최댓 값 찾기

- ex) 정렬되지 않은 리스트에서 최대 값 찾기

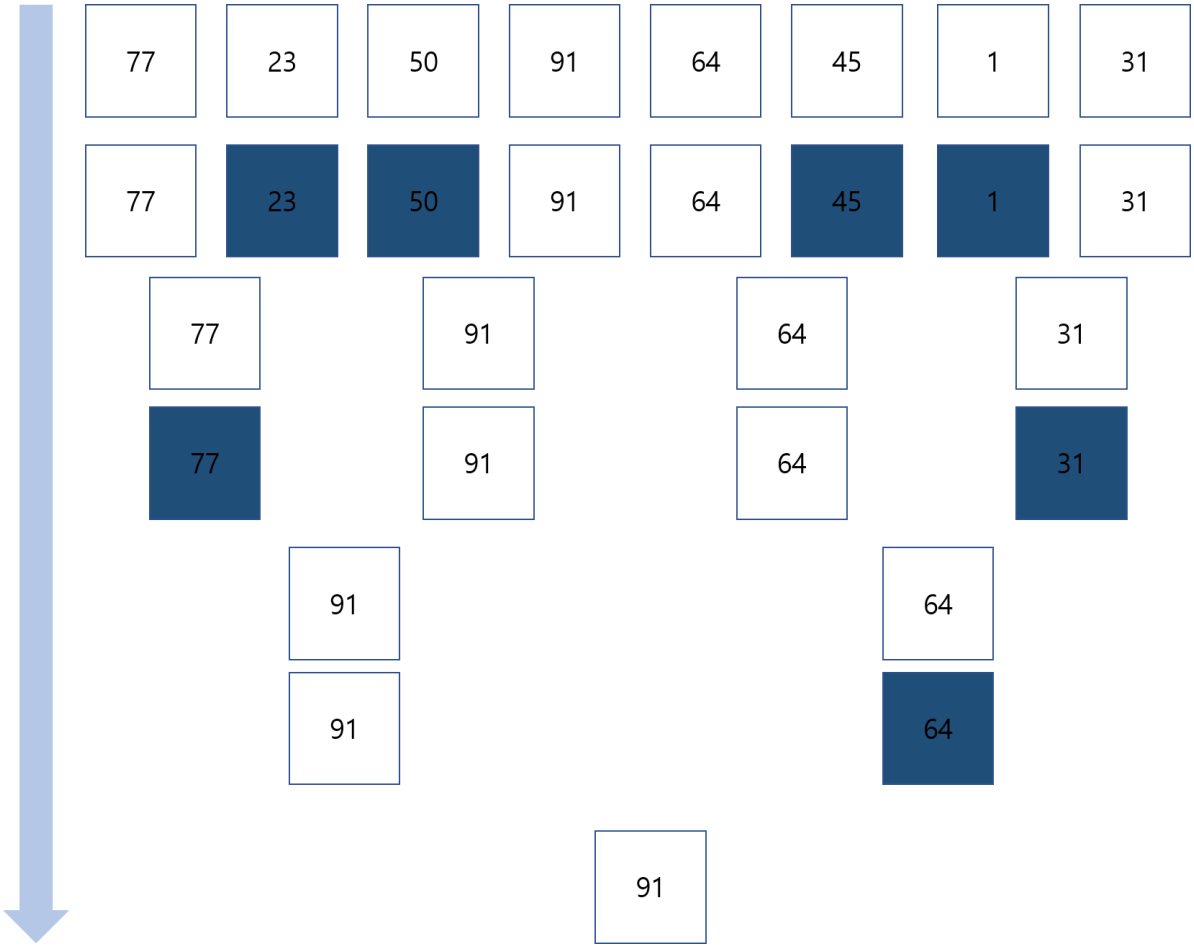
0. 정렬되지 않은 리스트



1. 분할하기(Divide)



2. 해결하며 합치기 (Conquer & Combine)



출처

https://ko.wikipedia.org/wiki/%EB%B6%84%ED%95%A0%EC%A0%95%EB%B3%B5%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

https://sinseonc.tistory.com/10

https://janghw.tistory.com/entry/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-Divide-and-Conquer-%EB%B6%84%ED%95%A0%EC%A0%95%EB%B3%B5

https://ko.wikipedia.org/wiki/%ED%80%B5_%EC%A0%95%EB%A0%AC

https://kimch3617.tistory.com/entry/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-%EB%B6%84%ED%95%A0%EC%A0%95%EB%B3%B5%EB%B2%95-Divide-and-Conquer