

Divide and Conquer

Heo song yi

CONCEPT

Divide and conquer

주어진 문제를 둘 이상의 부분으로 **나눈 뒤**,
각 문제에 대한 답을 **재귀호출**을 이용해 계산하고,
각 부분 문제의 답으로부터 **전체 문제의 답을 계산**해내는 알고리즘

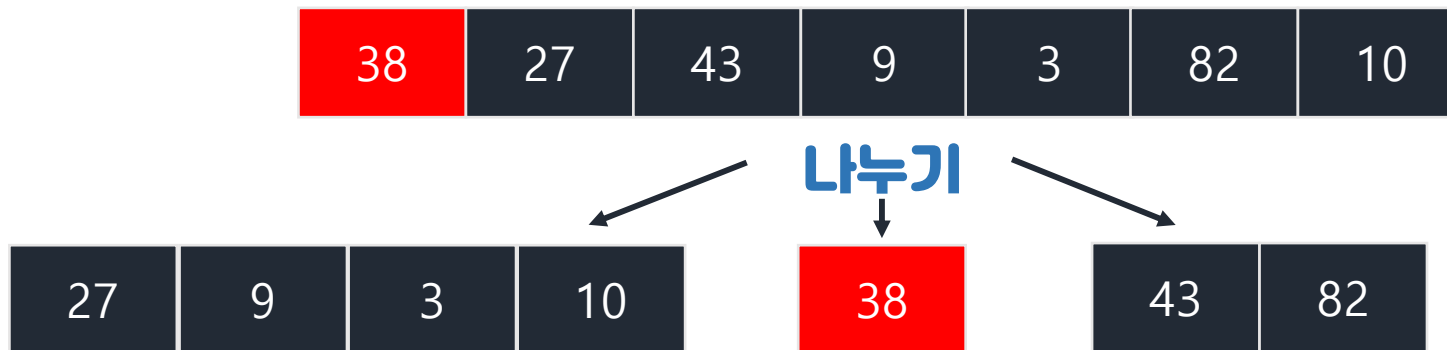
ex) 퀵 정렬, 병합 정렬

Quick Sort

Divide and conquer

주어진 문제

다음 7개의 숫자를 정렬하여라.

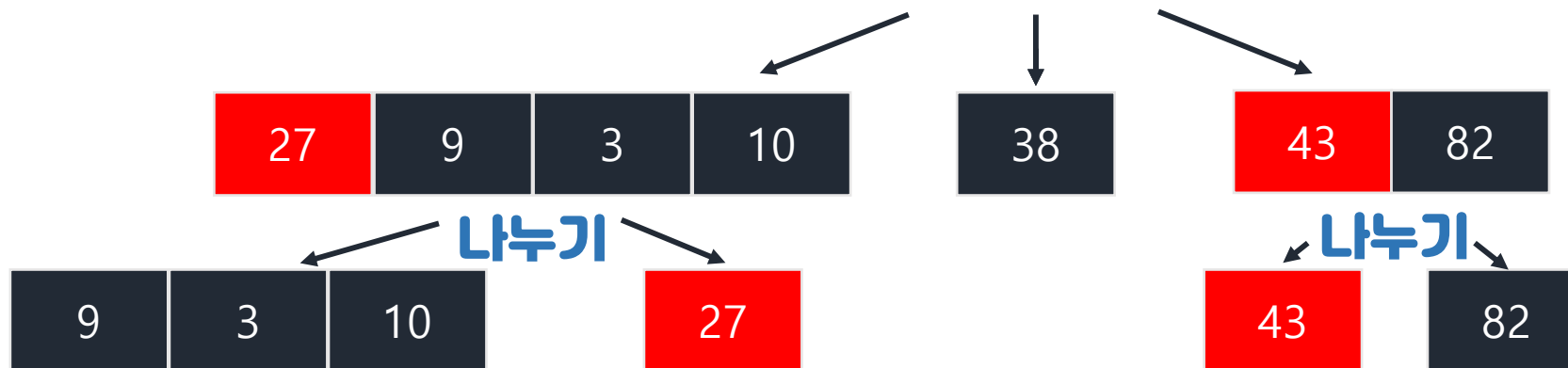


Quick Sort

Divide and conquer

주어진 문제

다음 7개의 숫자를 정렬하여라.

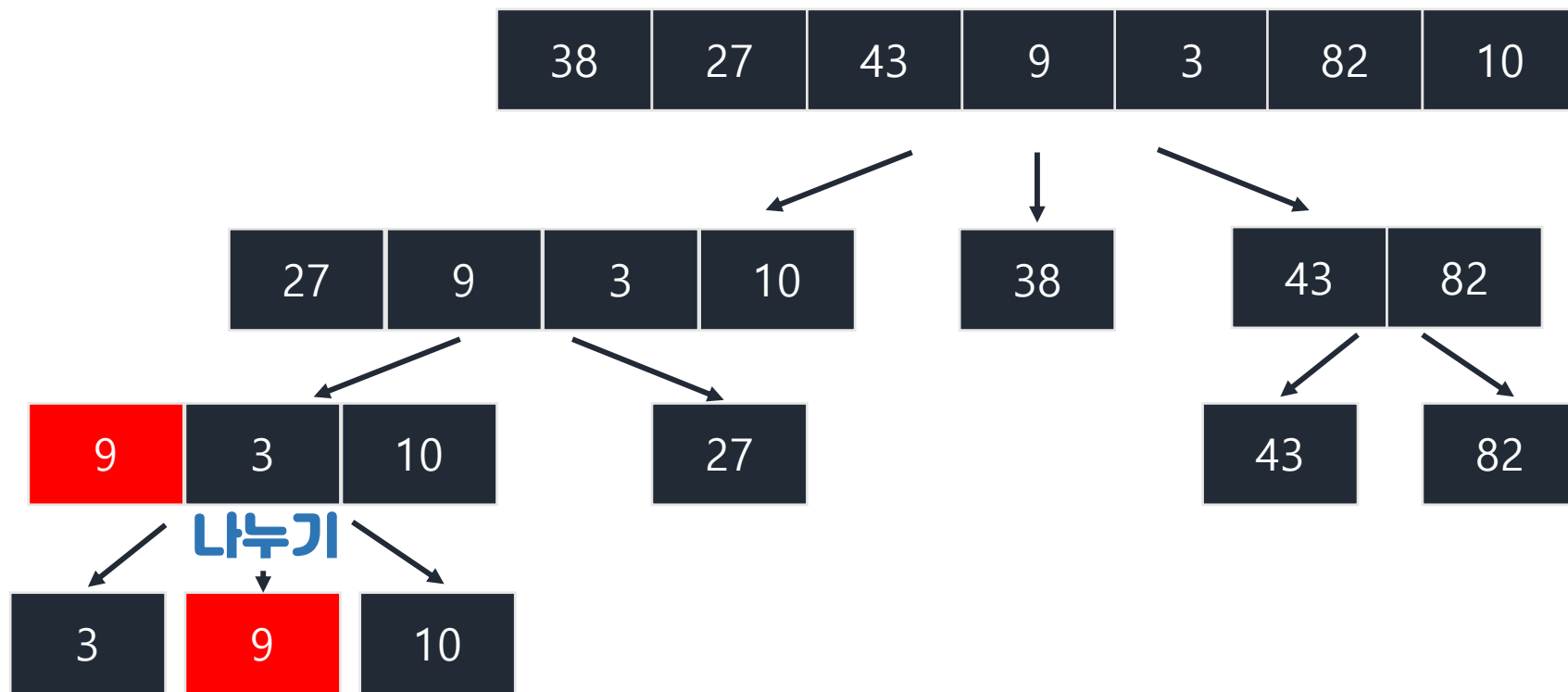


Quick Sort

Divide and conquer

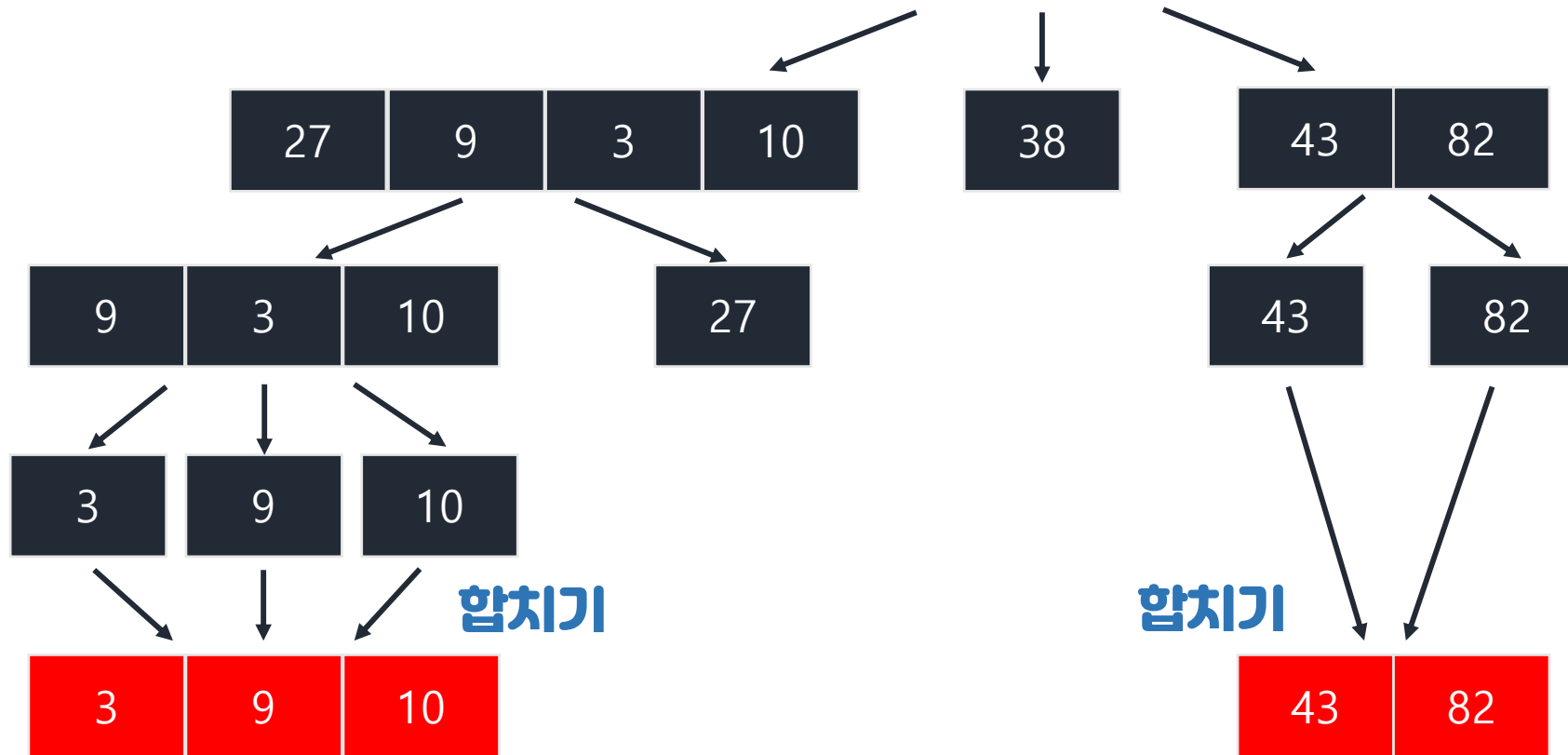
주어진 문제

다음 7개의 숫자를 정렬하여라.



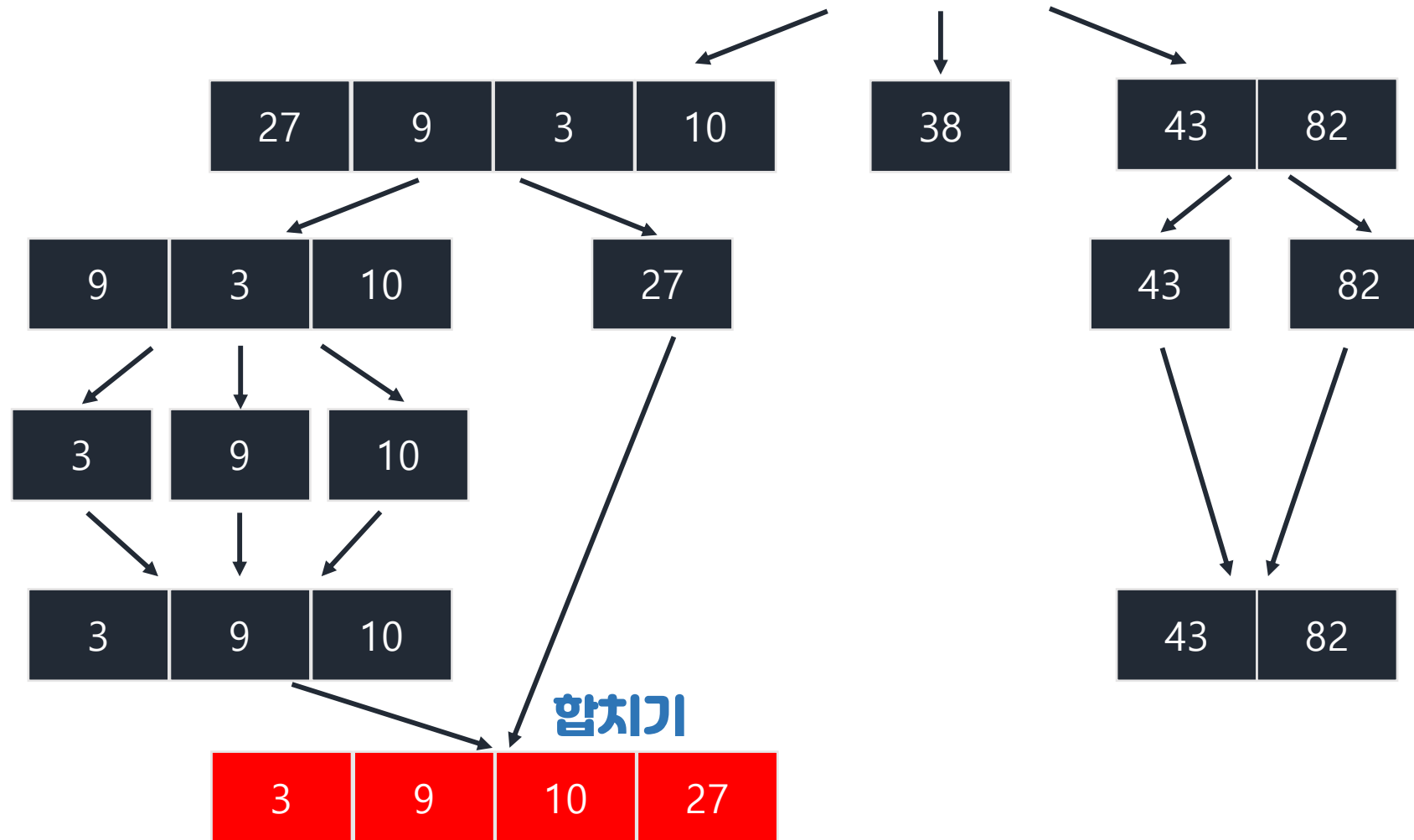
Quick Sort

Divide and conquer



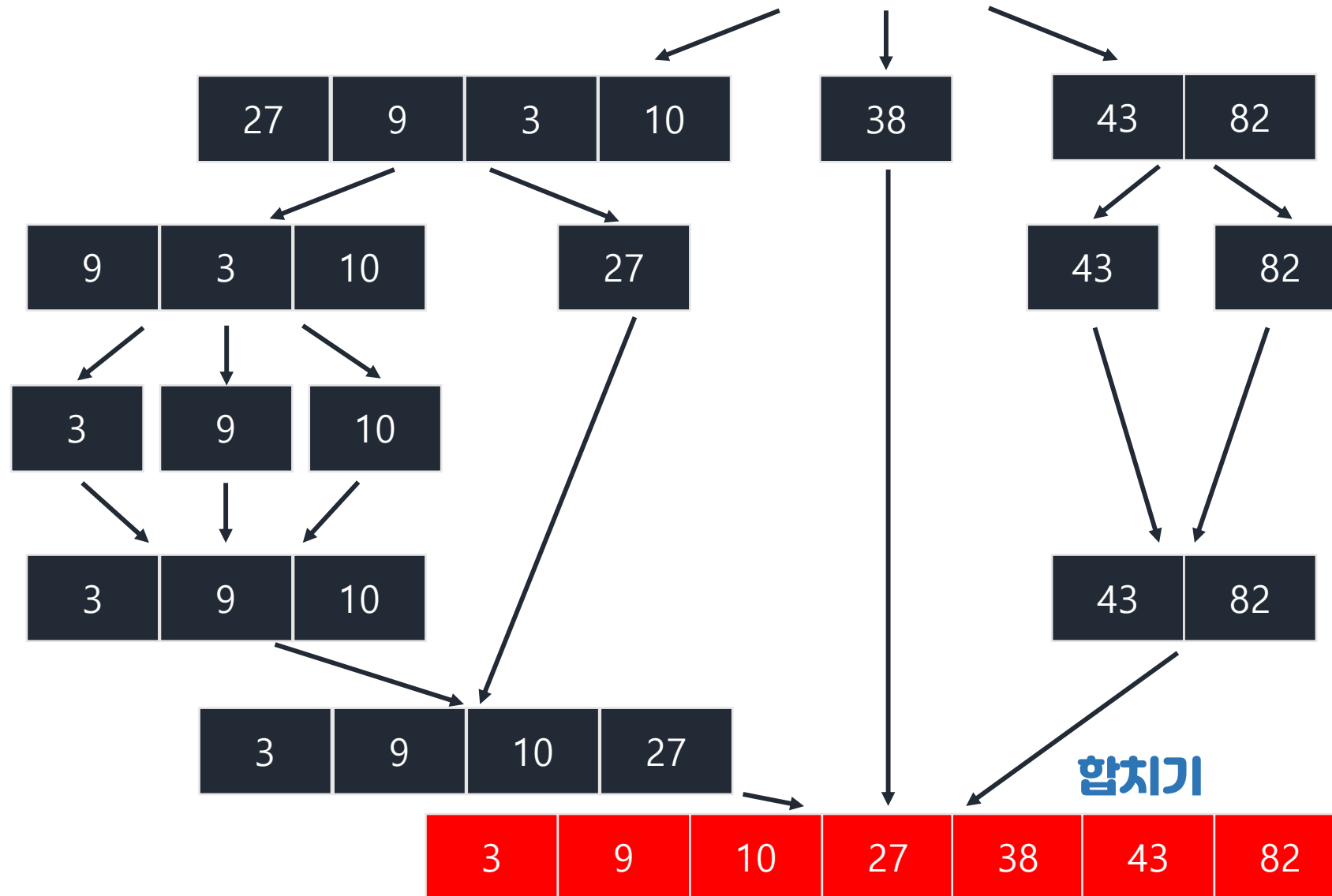
Quick Sort

Divide and conquer



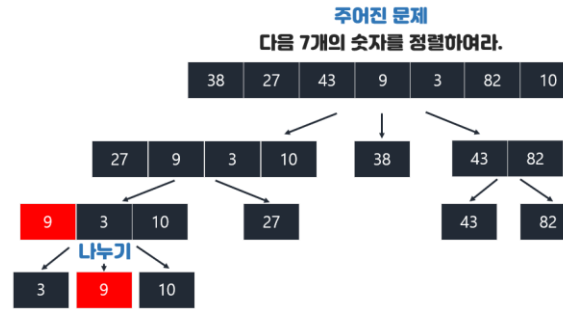
Quick Sort

Divide and conquer



Quick Sort

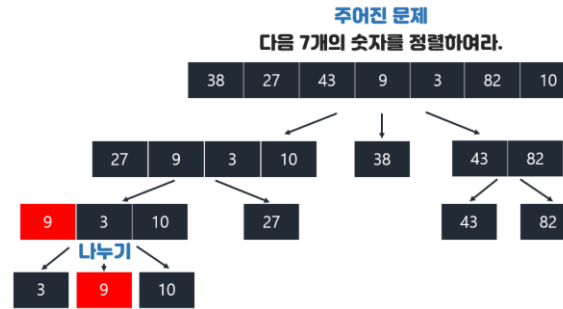
Divide and conquer



임의의 **기준 수(pivot)**를 지정한 후,
기준보다 작거나 같은 숫자를 왼쪽,
더 큰 숫자를 오른쪽으로 보낸다.

Quick Sort

Divide and conquer



임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.

!!반복!!

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



왼 -> 오 : 기준 수 보다 큰 값을 찾는다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



왼 -> 오 : 기준 수 보다 큰 값을 찾는다.

오 -> 왼 : 기준 수 보다 작은 값을 찾는다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



두 수를 바꾼다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



왼 -> 오 : 기준 수 보다 큰 값을 찾는다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



왼 -> 오 : 기준 수 보다 큰 값을 찾는다.

오 -> 왼 : 기준 수 보다 작은 값을 찾는다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



작은 값이 큰 값보다 앞에 있을 때에는, 피벗 값을 작은 값으로 바꿔준다.

Quick Sort

Divide and conquer

임의의 **기준 수(pivot)**를 지정한 후,
기준보다 **작거나 같은 숫자를 왼쪽**,
더 큰 숫자를 오른쪽으로 보낸다.



작은 값이 큰 값보다 앞에 있을 때에는, 피벗 값을 작은 값으로 바꿔준다.

Quick Sort

Divide and conquer

38	27	43	9	3	82	10
----	----	----	---	---	----	----

원 -> 오 : 기준 수 보다 큰 값을 찾는다.



```
#include <stdio.h>
```

```
int number = 7;
```

```
int data[7] = {38, 27, 43, 9, 3, 82, 10};
```

```
void quickSort(int *data, int start, int end) {
```

```
    // start: 첫번째 원소의 인덱스(0), 마지막 원소의 인덱스(6)
```

```
    if (start >= end) {
```

```
        return;
```

```
    }
```

```
    int key = start; //키(피벗)는 첫번째 원소의 인덱스(0)
```

```
    int i = start + 1;
```

```
    int j = end;
```

```
    int temp;
```

```
    while(i <= j) { //엇갈릴 때까지 반복
```

```
        while(data[i] <= data[key]) { //키 값보다 큰 값을 만날 때까지
```

```
            i++;
```

```
        }
```

Quick Sort

Divide and conquer

38	27	43	9	3	82	10
----	----	----	---	---	----	----

오 -> 왼 : 기준 수 보다 작은 값을 찾는다.

```
while(data[j] >= data[key] && j > start) { //키 값보다 작은 값을 만날 때까지  
    j--;  
}
```

Quick Sort

Divide and conquer

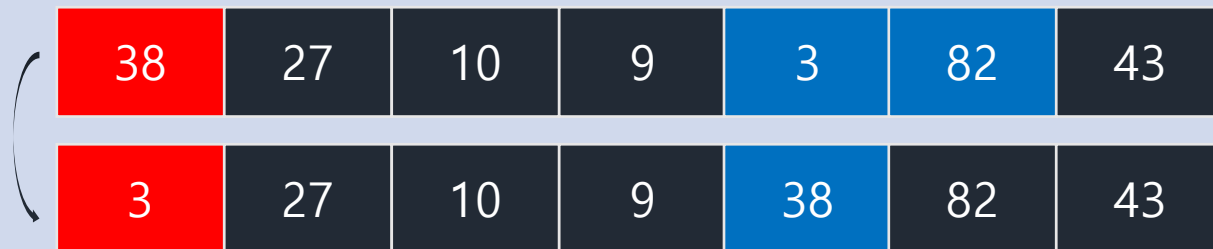
38	27	10	9	3	82	43
----	----	----	---	---	----	----

두 수를 바꾼다.

```
}  
if (i > j) {  
    temp = data[j];  
    data[j] = data[key];  
    data[key] = temp;  
} else {  
    temp = data[j];  
    data[j] = data[i];  
    data[i] = temp;  
}
```

Quick Sort

Divide and conquer



38	27	10	9	3	82	43
3	27	10	9	38	82	43

작은 값이 큰 값보다 앞에 있을 때에는,
피벗 값을 작은 값으로 바꿔준다.

```
}  
if (i > j) {  
    temp = data[j];  
    data[j] = data[key];  
    data[key] = temp;  
} else {  
    temp = data[i];
```

Quick Sort

Divide and conquer

3	27	10	9	38	82	43
---	----	----	---	----	----	----

피벗의 왼쪽부분과 오른쪽부분을 켜 정렬시킨다.

```
quickSort(data, start, j - 1);  
quickSort(data, j + 1, end);  
}
```

```
int main(void) {  
    quickSort(data, 0, number - 1);  
    for (int i=0; i < number; i++) {  
        printf("%d ", data[i]);  
    }  
}
```

Q&A

Divide and conquer