DIVIDE / CONQUER

발표자 : 최 연 석

07. 분할 정복

분할정복 알고리즘 ? 주어진 문제를 둘 이상의 부분 문제로 나눈 뒤 각 문제에 대한 답을 재귀 호출을 이용해 계산하고, 각 부분 문제의 답으로부터 전체 문제의 답을 계산해 내는 풀이 방법

- 문제를 더 작은 문제로 분할하는 과정 (divide)
- 각 문제에 대해 구한 답을 원래 문제에 대한 답으로 병합하는 과정 (merge)
- 더 이상 답을 분할하지 않고 곧장 풀 수 있는 매우 작은 문제 (base case)

조건 : 문제를 둘 이상 부분 문제로 나누는 자연스러운 방법이 있어야 하며, 부분 문제의 답을 조합해 원래 문제의 답을 계산하는 효율적인 방법이 있어야한다.

07. 분할 정복

예제 1) 수열의 빠른 합과 행렬의 빠른 제곱

1 부터 n까지 수의 합을 구하는 문제

$$fastsum(n) = 1 + 2 + 3 + \cdots + n$$
$$= \left(1 + 2 + \cdots + \frac{n}{2}\right) + \left(\left(\frac{n}{2} + 1\right) + \cdots + \mathrm{n}\right)$$

n이 짝수일때

$$\left(\frac{n}{2} + 1\right) + \cdots + n = \left(\frac{n}{2} + 1\right) + \cdots + \left(\frac{n}{2} + \frac{n}{2}\right)$$

$$= \left(\frac{n}{2} \times \frac{n}{2}\right) + \left(1 + 2 + 3 + \cdots + \frac{n}{2}\right)$$

$$= \left(\frac{n}{2} \times \frac{n}{2}\right) + fastsum\left(\frac{n}{2}\right)$$

$$fastsum(n) = \frac{n^2}{4} + 2 * fastsum\left(\frac{n}{2}\right)$$

07. 분할 정복

예제 1) 수열의 빠른 합과 행렬의 빠른 제곱

1 부터 n까지 수의 합을 구하는 문제

```
def fastSum(n):
    if n == 1:
        return 1
    if n % 2 == 1:
        return fastSum(n-1) + n
    return 2*fastSum(n//2) + (n//2)*(n//2)
```

07. 분할 정복

예제 1) 수열의 빠른 합과 행렬의 빠른 제곱

시간 복잡도 분석

$recursiveSum$

n = 5;

$$recursiveSum(5) = recursiveSum(4) + 5$$

$$= recursiveSum(3) + 4 + 5$$

$$= recursiveSum(2) + 3 + 4 + 5$$

$$= recursiveSum(1) + 2 + 3 + 4 + 5$$

$$= 1 + 2 + 3 + 4 + 5 = 15$$

O(n)

$fastSum$

n = 5;

$$fastSum(5) = fastSum(4) + 5$$

$$= (2 \times fastSum(2) + 4) + 5$$

$$= (2 \times (2 \times fastSum(1) + 1) + 4) + 5$$

$$= 2 \times (2 \times 1 + 1) + 4 + 5 = 15$$

두 번에 한번꼴로 n이 절반으로 줄어드니 호출되는 횟수는 n보다 작고 시간 복잡도도 O(n) 보다 작다.

07. 분할 정복

예제1) 수열의 빠른 합과 행렬의 빠른 제곱

행렬의 거듭제곱을 구하는 문제

A^m

만약 m이 1,000,000같이 매우 큰 수라면 일일이 A를 연속해서 곱해서 값을 구하는 것은 꽤나 오랜 시간이 걸리게 된다.

$A^m = A^{m/2} \times A^{m/2}$

앞서 보인 덧셈의 예제와 마찬가지로 m을 절반으로 나눌 수 있는 짝수인 경우와 그렇지 않는 홀수인 경우로 나눠서 거듭제곱을 계산하면 훨씬 빠른 시간 안에 정답을 구할 수 있다.

07. 분할 정복

예제1) 수열의 빠른 합과 행렬의 빠른 제곱

행렬의 거듭제곱을 구하는 문제

```python
def pow(a, m):
    if m == 0:
        return identity(len(a))
    if m % 2 > 0:
        return pow(a, m-1) * a
    half = pow(a, m//2)
    return half * half
```

$a^{10} = a^5 \times a^5$

$= (a \times a^4) \times (a \times a^4)$

$= (a \times a^2 \times a^2) \times (a \times a^2 \times a^2)$

$= (a \times a \times a \times a \times a) \times (a \times a$

$\times a \times a \times a)$

예제1) 수열의 빠른 합과 행렬의 빠른 제곱

행렬의 거듭제곱을 구하는 문제

```python
def pow(a, m):
    if m == 0:
        return identity(len(a))
    if m % 2 > 0:
        return pow(a, m-1) * a
    half = pow(a, m//2)
    return half * half
```

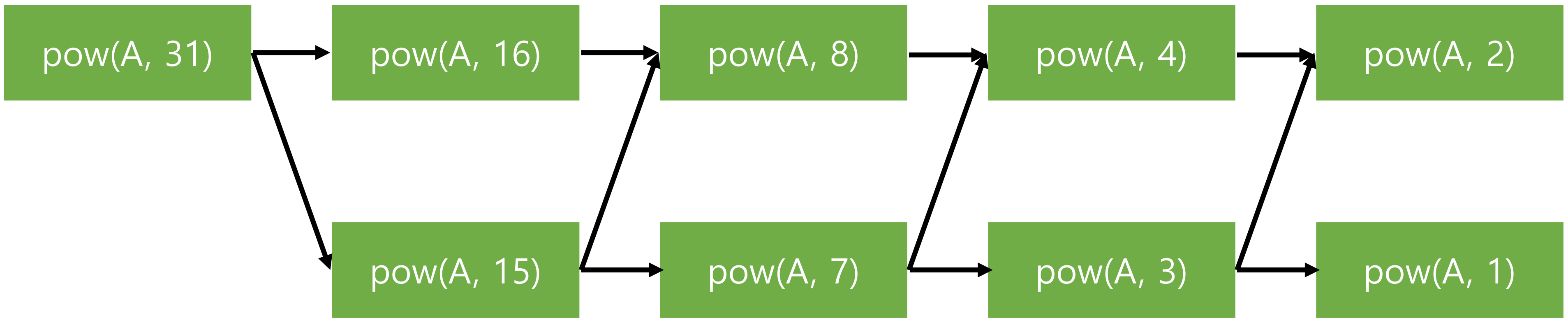
홀수인 경우

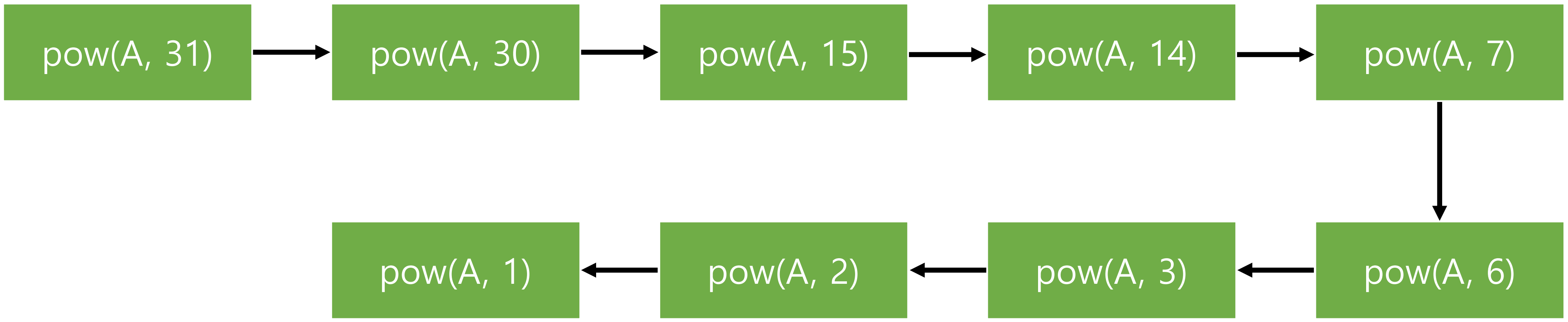
왜 m-1 과 1로 나누는가?

07. 분할 정복

예제1) 수열의 빠른 합과 행렬의 빠른 제곱

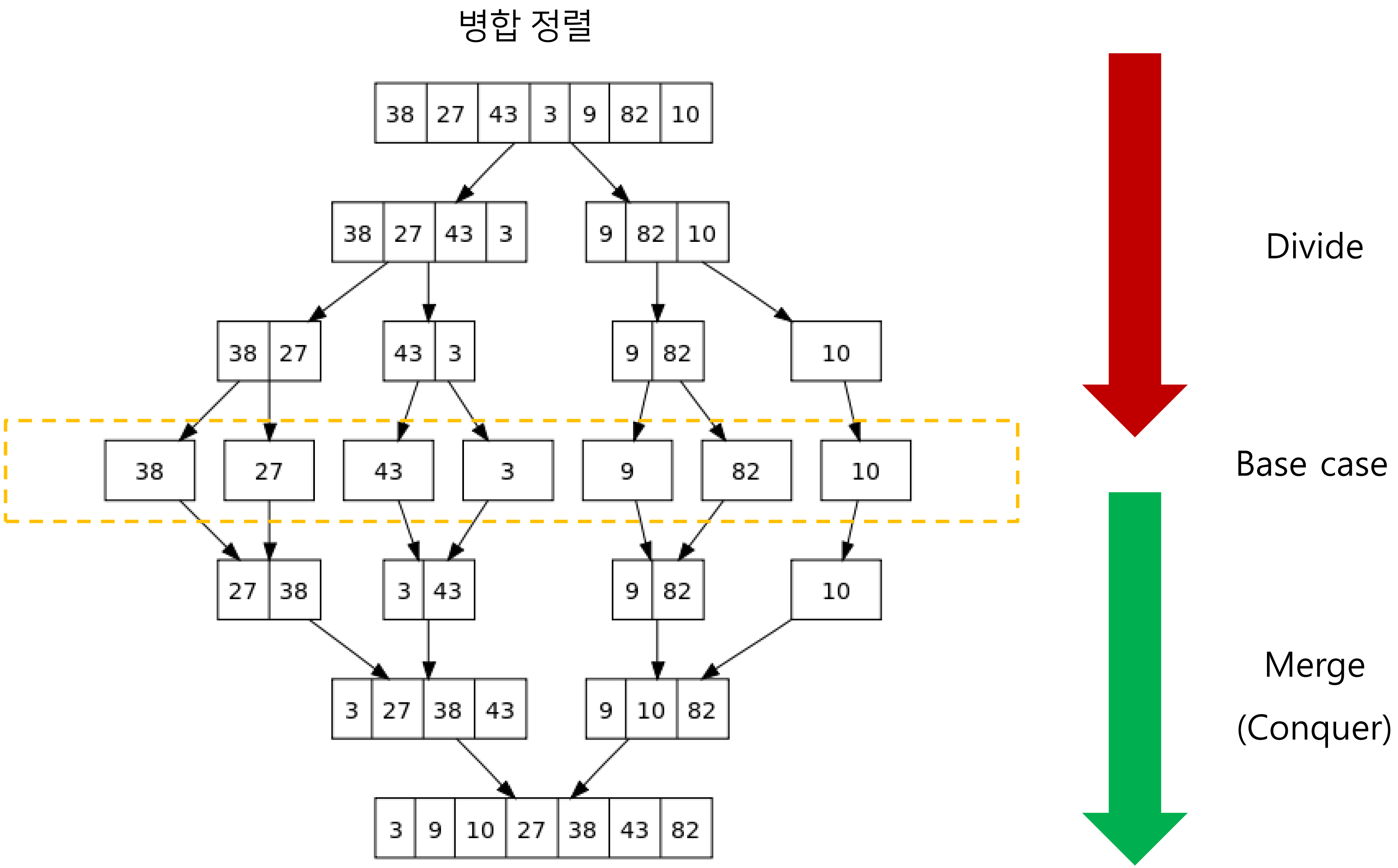
행렬의 거듭제곱을 구하는 문제





07. 분할 정복

예제2) 병합 정렬과 퀵 정렬

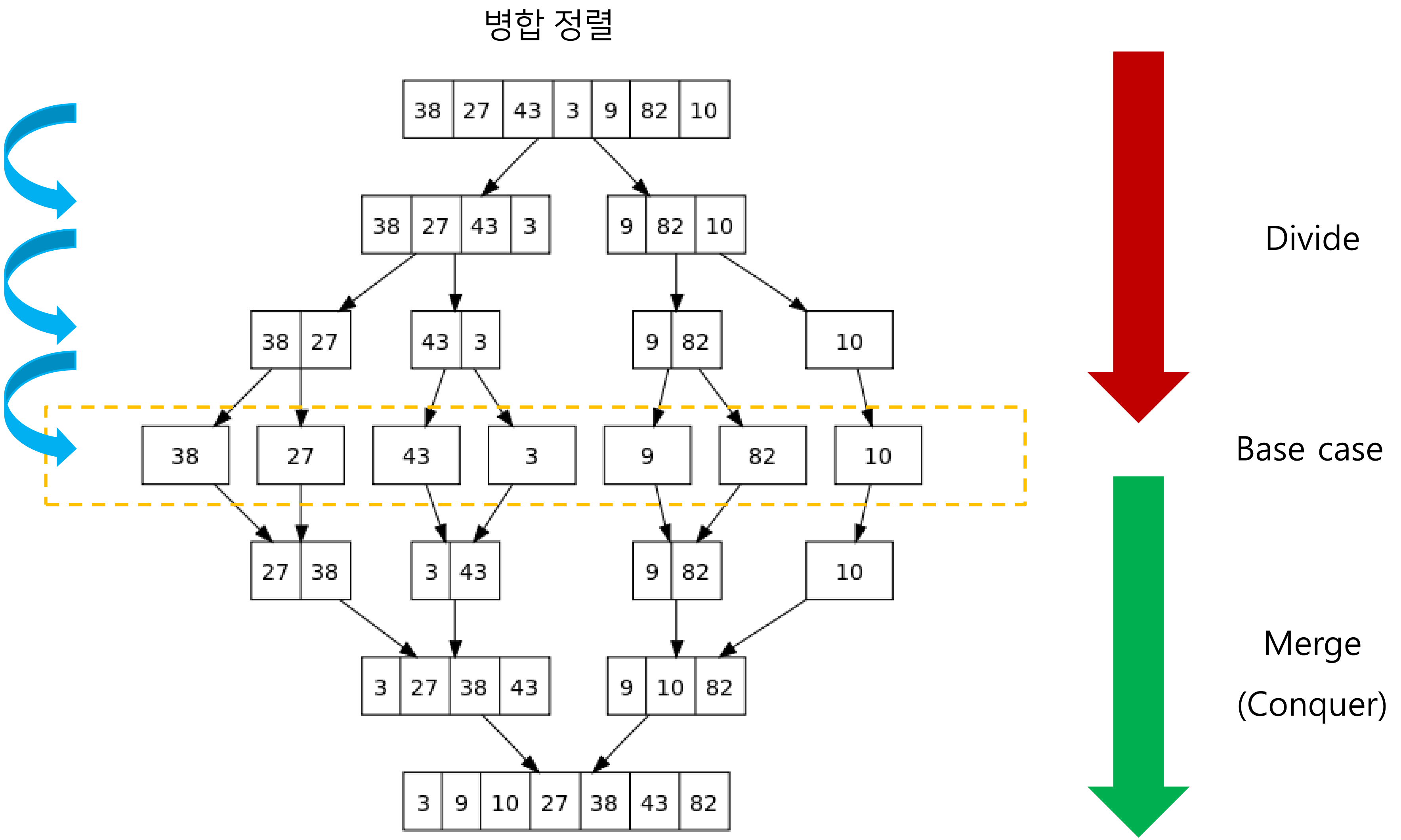


07. 분할 정복

예제2) 병합 정렬과 퀵 정렬

원소의 개수가 1개가 될 때까지 절반으로 나눈다.

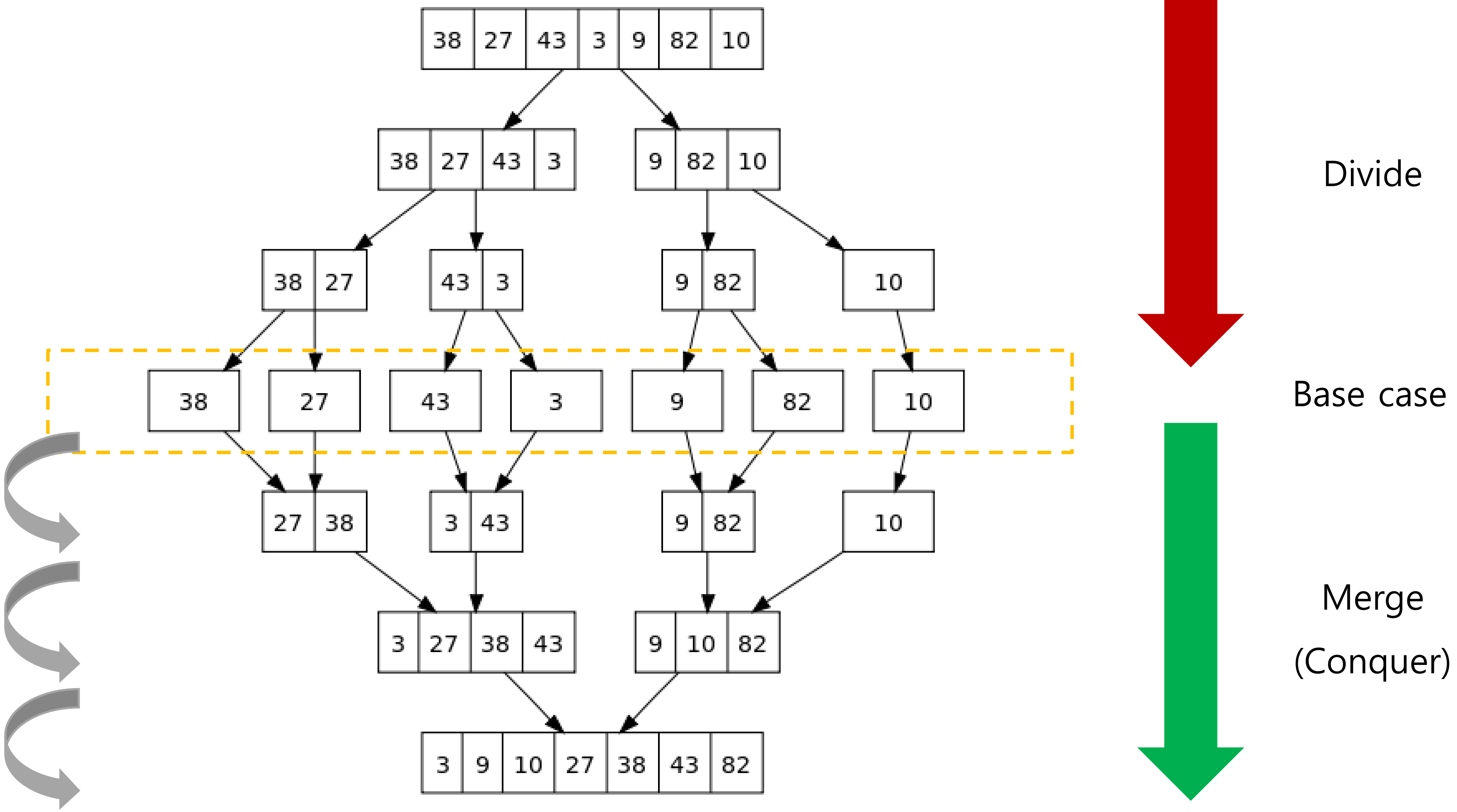
$=O(\log_2 n)$



07. 분할 정복

예제2) 병합 정렬과 퀵 정렬

병합 정렬

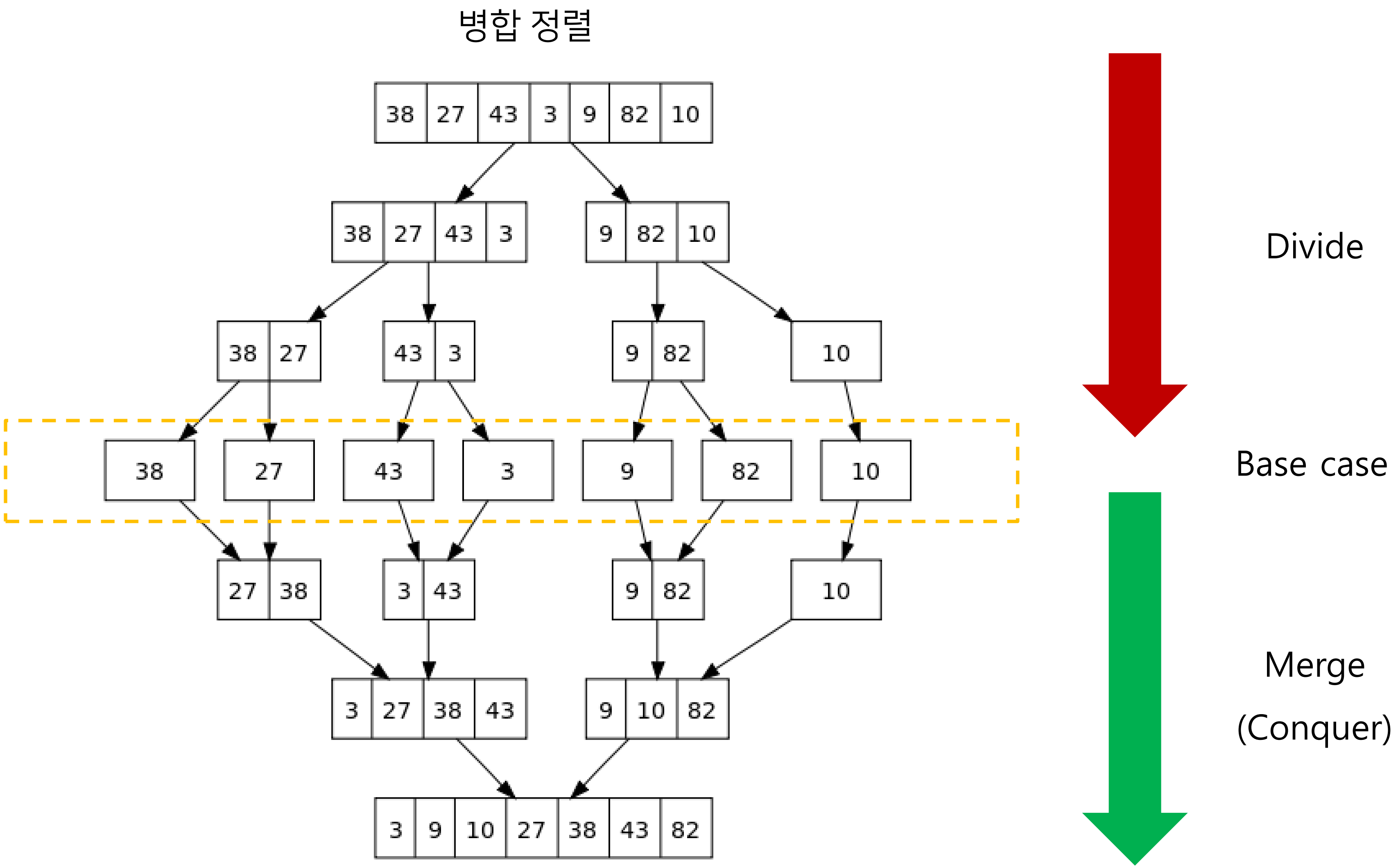


정렬하며 병합한다.

$=O(n)$

07. 분할 정복

예제2) 병합 정렬과 퀵 정렬

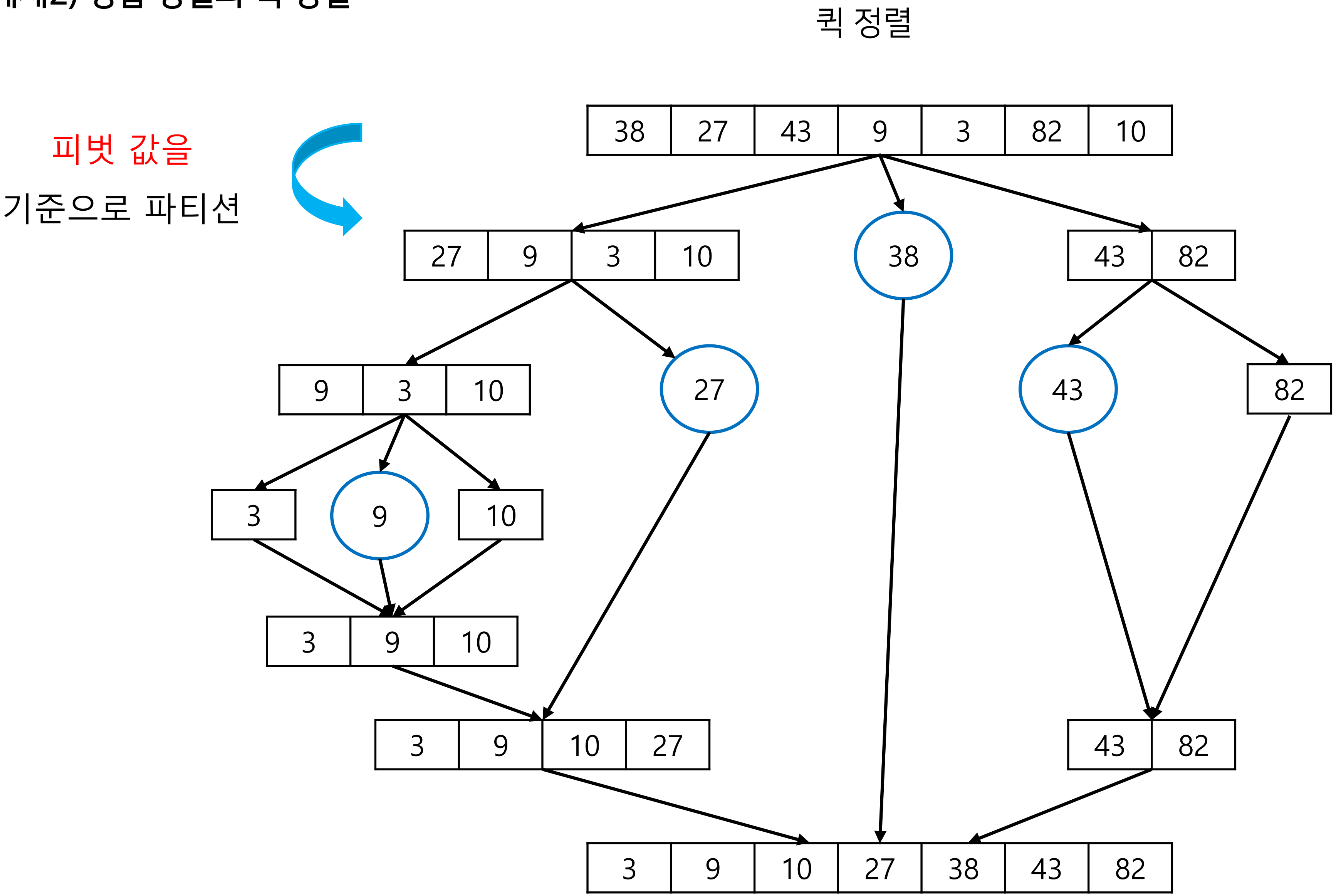


시간복잡도

$=O(n \log_2 n)$

07. 분할 정복

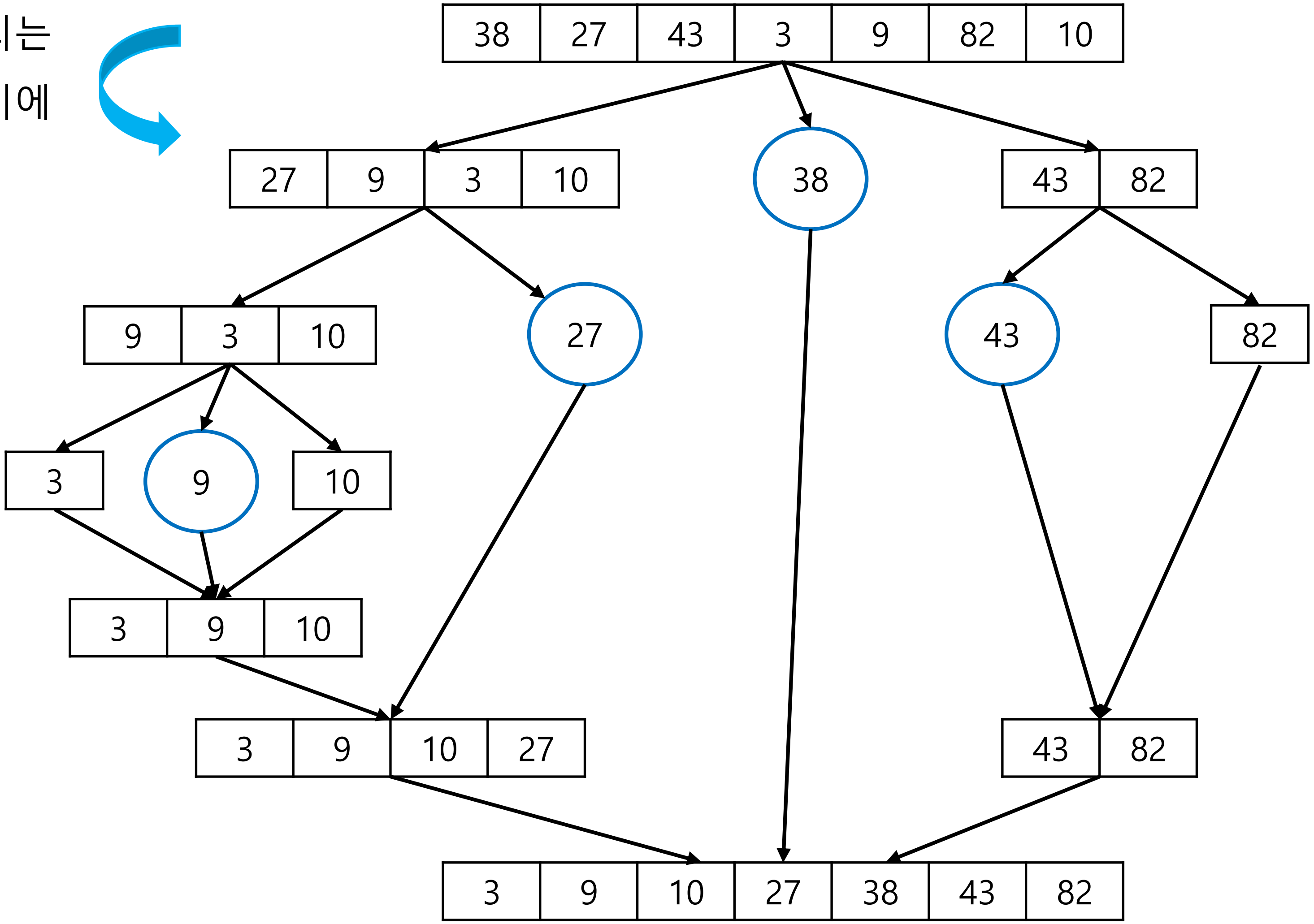
예제2) 병합 정렬과 퀵 정렬



예제2) 병합 정렬과 퀵 정렬

퀵 정렬

파티션 하는데 걸리는 시간은 배열의 길이에 비례



최악 시간 복잡도

$O(n^2)$

평균 시간 복잡도

$O(n\log_2 n)$

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

<u>1234 와 5678을 곱해보자!</u>

			1	2	3	4
		X	5	6	7	8
			9	8	7	2
		8	6	3	8	
	7	4	0	4		
6	1	7	0			
7	0	0	6	6	5	2

			1	2	3	4
		X	5	6	7	8
			8	16	24	32
		7	14	21	28	
	6	12	18	24		
5	10	15	20			
5	16	34	60	61	52	32

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

1234 와 5678을 곱해보자!

			1	2	3	4
		X	5	6	7	8
			9	8	7	2
		8	6	3	8	
	7	4	0	4		
6	1	7	0			
7	0	0	6	6	5	2

2

			1	2	3	4
		X	5	6	7	8
			8	16	24	32
		7	14	21	28	
	6	12	18	24		
5	10	15	20			
5	16	34	60	61	52	32

1

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

<u>1234 와 5678을 곱해보자!</u>

[4, 3, 2, 1]	a[ ]				1	2	3	4
[8, 7, 6, 5]	b[ ]			X	5	6	7	8
					8	16	24	32
				7	14	21	28	
			6	12	18	24		
		5	10	15	20			
[32, 52, 61, 60, 34, 16, 5]	c[ ]	5	16	34	60	61	52	32

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

<u>1234 와 5678을 곱해보자!</u>

[4, 3, 2, 1] ← **a[ ]** ← 1 2 3 4

[8, 7, 6, 5] ← **b[ ]** ← 5 6 7 8

```
for i in range(len(a)):
    for j in range(len(b)):
        c[ i + j ] += a[i] * b[j]
```

			1	2	3	4
		X	5	6	7	8
			8	16	24	32
		7	14	21	28	
	6	12	18	24		
5	10	15	20			
5	16	34	60	61	52	32

[32, 52, 61, 60, 34, 16, 5] ← **c[ ]** ← 5 16 34 60 61 52 32

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

1234 와 5678을 곱해보자!

```
def multiply(a, b):
    c = [0]*(len(a) + len(b) - 1)
    for i in range(len(a)):
        for j in range(len(b)):
            c[i + j] += a[i] * b[j]
    normalize(c)
    return c
```

07. 분할 정복

예제3) 카라츠바의 빠른 곱셈 알고리즘

<u>1234 와 5678을 곱해보자!</u>

c = [32, 52, 61, 60, 34, 16, 5] $\xrightarrow{\text{자리 수 조정}}$ [7, 0, 0, 6, 6, 5, 2]

&

뒤 집기

예제3) 카라츠바의 빠른 곱셈 알고리즘

<u>1234 와 5678을 곱해보자!</u>

```
def normalize(num):
    num.append(0)
    for i in range(len(num)-1):
        if num[i] < 0:
            borrow = (abs(num[i]) + 9) // 10
            num[i+1] -= borrow
            num[i] += borrow * 10
        else:
            num[i+1] += num[i] // 10
            num[i] %= 10
    ans = []
    while num:
        ans.append(num.pop())
    return int(''.join(map(str, ans)))
```

[32, 52, 61, 60, 34, 16, 5]

[2, 55, 61, 60, 34, 16, 5]

[2, 5, 66, 60, 34, 16, 5]

[2, 5, 6, 66, 34, 16, 5]

[2, 5, 6, 6, 40, 16, 5]

[2, 5, 6, 6, 0, 20, 5]

[2, 5, 6, 6, 0, 0, 7]

예제3) 카라츠바의 빠른 곱셈 알고리즘

1234 와 5678을 곱해보자!

```
def multiply(a, b):
    c = [0]*(len(a) + len(b) - 1)
    for i in range(len(a)):
        for j in range(len(b)):
            c[i + j] += a[i] * b[j]
    normalize(c)
    return c
```

$O(n^2)$

예제3) 카라츠바의 빠른 곱셈 알고리즘

처음 128자리 수

그 다음 128자리 수

$$a = a_1 \times 10^{128} + a_0$$

$$b = b_1 \times 10^{128} + b_0$$

$$a \times b$$

$$= (a_1 \times 10^{128} + a_0) \times (b_1 \times 10^{128} + b_0)$$

$$= a_1 \times b_1 \times 10^{256} + (a_1 \times b_0 + a_0 \times b_1) \times 10^{128} + a_0 \times b_0$$

예제3) 카라츠바의 빠른 곱셈 알고리즘

$$a = a_1 \times 10^{128} + a_0$$

$$b = b_1 \times 10^{128} + b_0$$

$$a \times b$$

$$= (a_0 \times 10^{128} + a_0) \times (b_1 \times 10^{128} + b_0)$$

$$= \underbrace{a_1 \times b_1}_{z_0} \times 10^{256} + \underbrace{(a_1 \times b_0 + a_0 \times b_1)}_{z_1} \times 10^{128} + \underbrace{a_0 \times b_0}_{z_2}$$

$$= (a_0 + a_1) \times (b_0 + b_1) - z_0 - z_2$$

예제3) 카라츠바의 빠른 곱셈 알고리즘

1234 와 5678을 곱해보자!

$$O(3^k) = O(3^{\log n}) = O(n^{\log 3})$$

```python
def karatsuba(a, b):
    an = len(a)
    bn = len(b)
    if an < bn:
        return karatsuba(b, a)
    if an == 0 or bn == 0:
        return
    if an <= 50:
        return multiply(a, b)
```

```python
    half = an // 2
    a1 = a[:half]
    a0 = a[half:]
    b1 = b[:min(half, bn)]
    b0 = b[min(half, bn):bn]

    z2 = karatsuba(a0, b0)
    z0 = karatsuba(a1, b1)
    z1 = karatsuba(a0+a1, b0+b1) - z0 - z2
    ret = z0*(10**(2*half)) + z1*(10**half) + z2
    return ret
```

07. 분할 정복

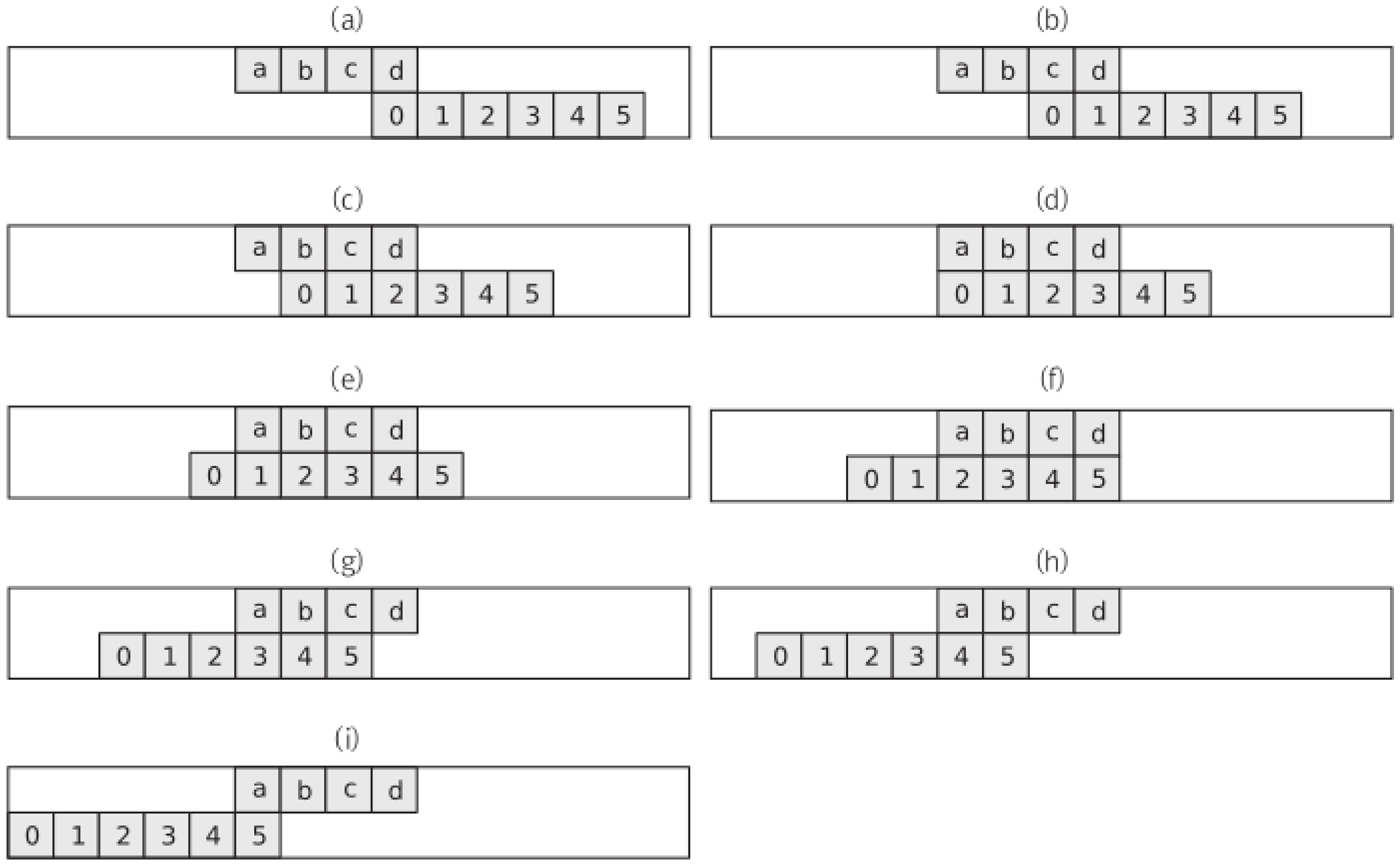
예제5) 팬미팅

실제입력 : FFFMMM
MMMFFF
FFFFFFFFFFFF
FFFFM

멤버들 a,b,c,d

팬 0,1,2,3,4,5

남자끼리는 포옹 x, 전체가 다 포옹하게 되는 횟수는 몇 번인가?



예제5) 팬미팅

남자끼리는 포옹 x, 전체가 다 포옹하게 되는 횟수는 몇 번인가?

곱셈으로 변경!

					A_2	A_1	A_0
	X	B_5	B_4	B_3	B_2	B_1	B_0
					$A_2 \cdot B_0$	$A_1 \cdot B_0$	$A_0 \cdot B_0$
				$A_2 \cdot B_1$	$A_1 \cdot B_1$	$A_0 \cdot B_1$	
			$A_2 \cdot B_2$	$A_1 \cdot B_2$	$A_0 \cdot B_2$		
		$A_2 \cdot B_3$	$A_1 \cdot B_3$	$A_0 \cdot B_3$			
	$A_2 \cdot B_4$	$A_1 \cdot B_4$	$A_0 \cdot B_4$				
$A_2 \cdot B_5$	$A_1 \cdot B_5$	$A_0 \cdot B_5$					
C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

07. 분할 정복

예제5) 팬미팅

<u>남자끼리는 포옹 x, 전체가 다 포옹하게 되는 횟수는 몇 번인가?</u>

<u>곱셈으로 변경!</u>

A_2	A_1	A_0

←

B_0	B_1	B_2	B_3	B_4	B_5

					A_2	A_1	A_0
	X	B_5	B_4	B_3	B_2	B_1	B_0
					$A_2 \cdot B_0$	$A_1 \cdot B_0$	$A_0 \cdot B_0$
				$A_2 \cdot B_1$	$A_1 \cdot B_1$	$A_0 \cdot B_1$	
			$A_2 \cdot B_2$	$A_1 \cdot B_2$	$A_0 \cdot B_2$		
		$A_2 \cdot B_3$	$A_1 \cdot B_3$	$A_0 \cdot B_3$			
	$A_2 \cdot B_4$	$A_1 \cdot B_4$	$A_0 \cdot B_4$				
$A_2 \cdot B_5$	$A_1 \cdot B_5$	$A_0 \cdot B_5$					
C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

남자를 1, 여자를 0으로 표기한다면
남자와 남자가 만났을 경우에만
C[i]에 1이 저장된다.
따라서 C[i] = 0일 때 모두가 포옹하게 되고
배열 C에서 0의 개수가 정답이 된다.

07. 분할 정복

예제5) 팬미팅

카라츠바 알고리즘으로 구현

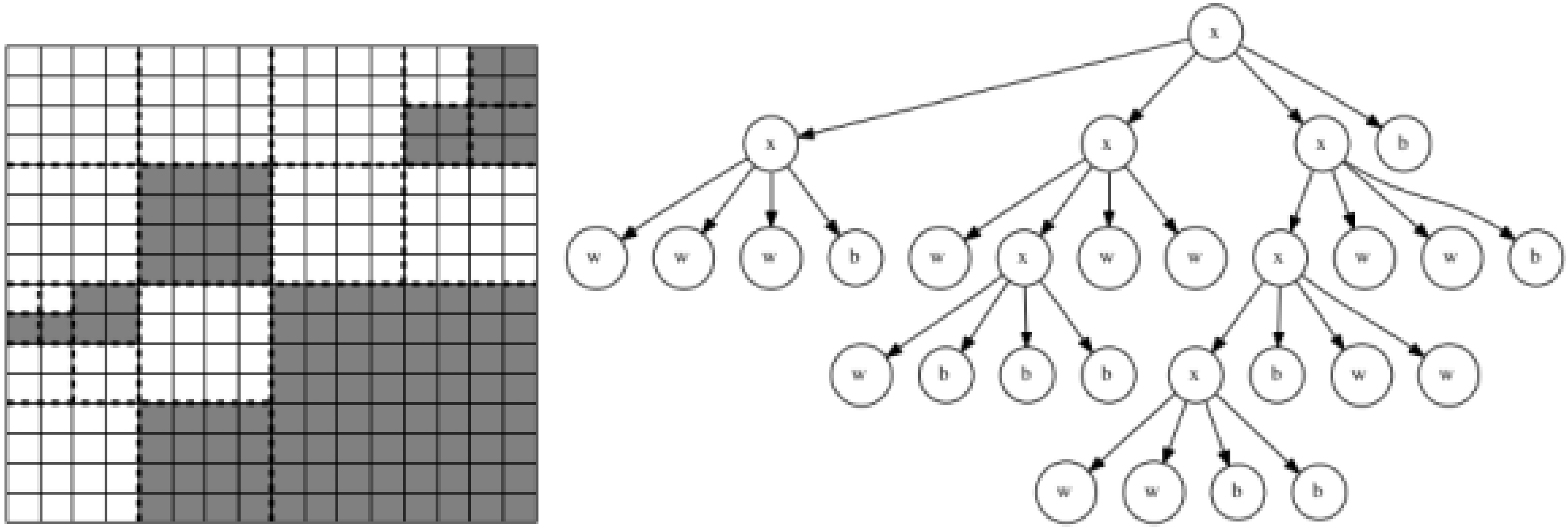
```python
def hugs(members, fans):
    n = len(members)
    m = len(fans)

    a = [0] * n
    b = [0] * m
    for i in range(n):
        if members[i] == "M":
            a[i] = 1
    for i in range(m):
        if fans[i] == 'M':
            b[i] = 1

    C = karatsuba(a, b)
    allHugs = C.count(0)
    return allHugs
```

$\sim O(n^{\log 3})$

예제4) 쿼드 트리 문제 뒤집기



(a) (b)

X (왼쪽 위 부분의 압축결과)(오른쪽 위 부분의 압축 결과)(왼쪽 아래 부분의 압축결과)(오른쪽 아래 부분의 압축결과)

xxwww bxwxw bbbww xxxww bbbww wwbb

07. 분할 정복

예제4) 쿼드 트리 문제 뒤집기

입력 값.		출력 값.
w		w
xbwwb	뒤집기 →	xwbbw
xbwxwbbwb		xxbwwbbbw
xxwwwbxwxwbbbwwxxxwwbbbwwwwbb		xxwbxwwxbbwwbwbxwbwwxwwwxbbwb

예제4) 쿼드 트리 문제 뒤집기

쿼드 트리

1) 압축해제하기

idx = -1로 초기화

```
def decompress(it, x, y, size):
    global idx
    idx += 1
    head = it[idx]
    if head == 'b' or head == 'w':
        for dx in range(size):
            for dy in range(size):
                decompressed[x+dx][y+dy] = head
    else:
        half = size//2
        decompress(it, x, y, half)
        decompress(it, x, y+half, half)
        decompress(it, x+half, y, half)
        decompress(it, x+half, y+half, half)
```

예제4) 쿼드 트리 문제 뒤집기

쿼드 트리

2) 바로 뒤집기

idx = -1로 초기화

```python
def reverse(it):
    global idx
    idx += 1
    head = it[idx]
    if head == 'b' or head == 'w':
        return head
    upperLeft = reverse(it)
    upperRight = reverse(it)
    lowerLeft = reverse(it)
    lowerRight = reverse(it)

    return 'x' + lowerLeft + lowerRight + upperLeft + upperRight
```

07. 분할 정복

예제4) 쿼드 트리 문제 뒤집기

<u>쿼드 트리</u>

<u>2) 바로 뒤집기</u>

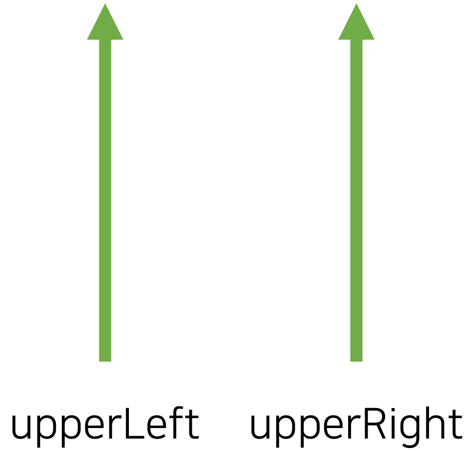
x	b	w	x	w	b	b	w	b

예제4) 쿼드 트리 문제 뒤집기

쿼드 트리

2) 바로 뒤집기

x	b	w	x	w	b	b	w	b

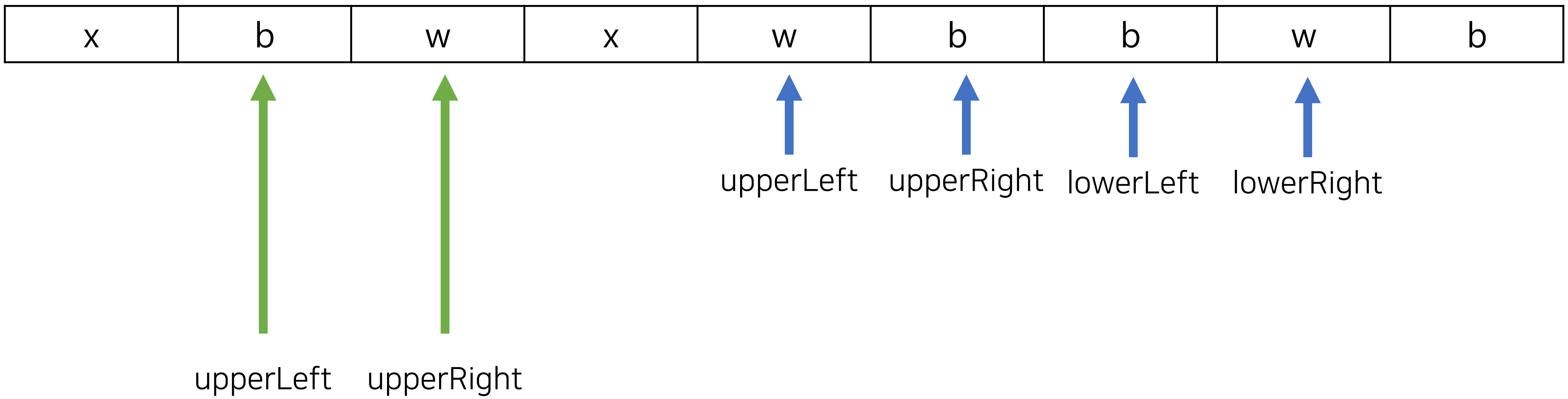


07. 분할 정복

예제4) 쿼드 트리 문제 뒤집기

쿼드 트리

2) 바로 뒤집기

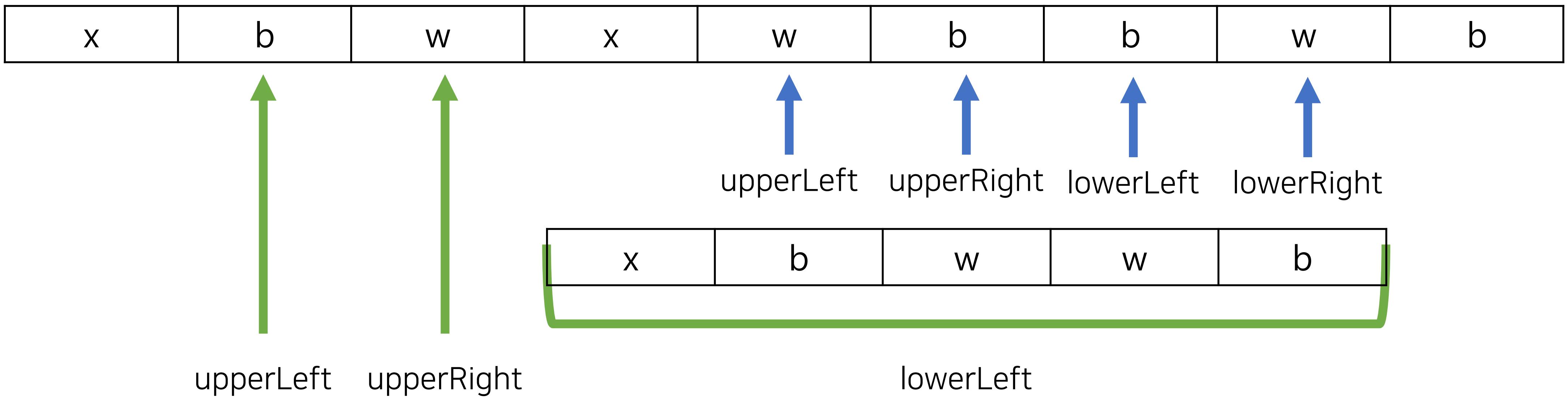


07. 분할 정복

예제4) 쿼드 트리 문제 뒤집기

<u>쿼드 트리</u>

<u>2) 바로 뒤집기</u>

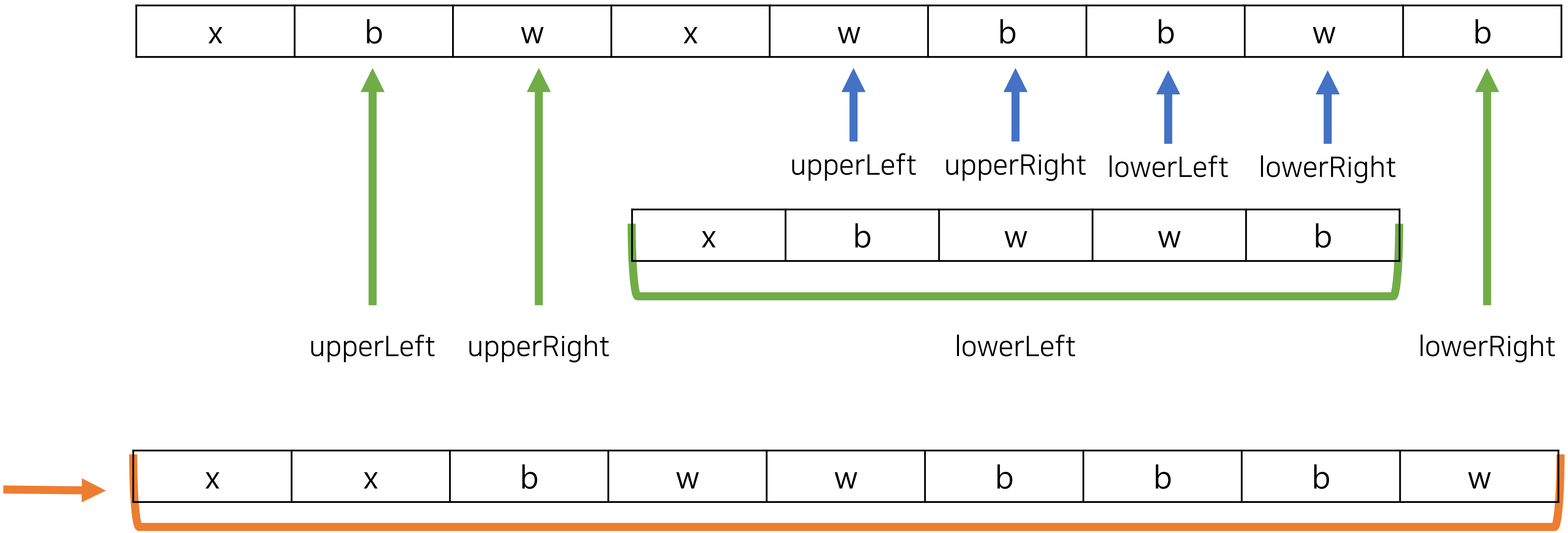


07. 분할 정복

예제4) 쿼드 트리 문제 뒤집기

쿼드 트리

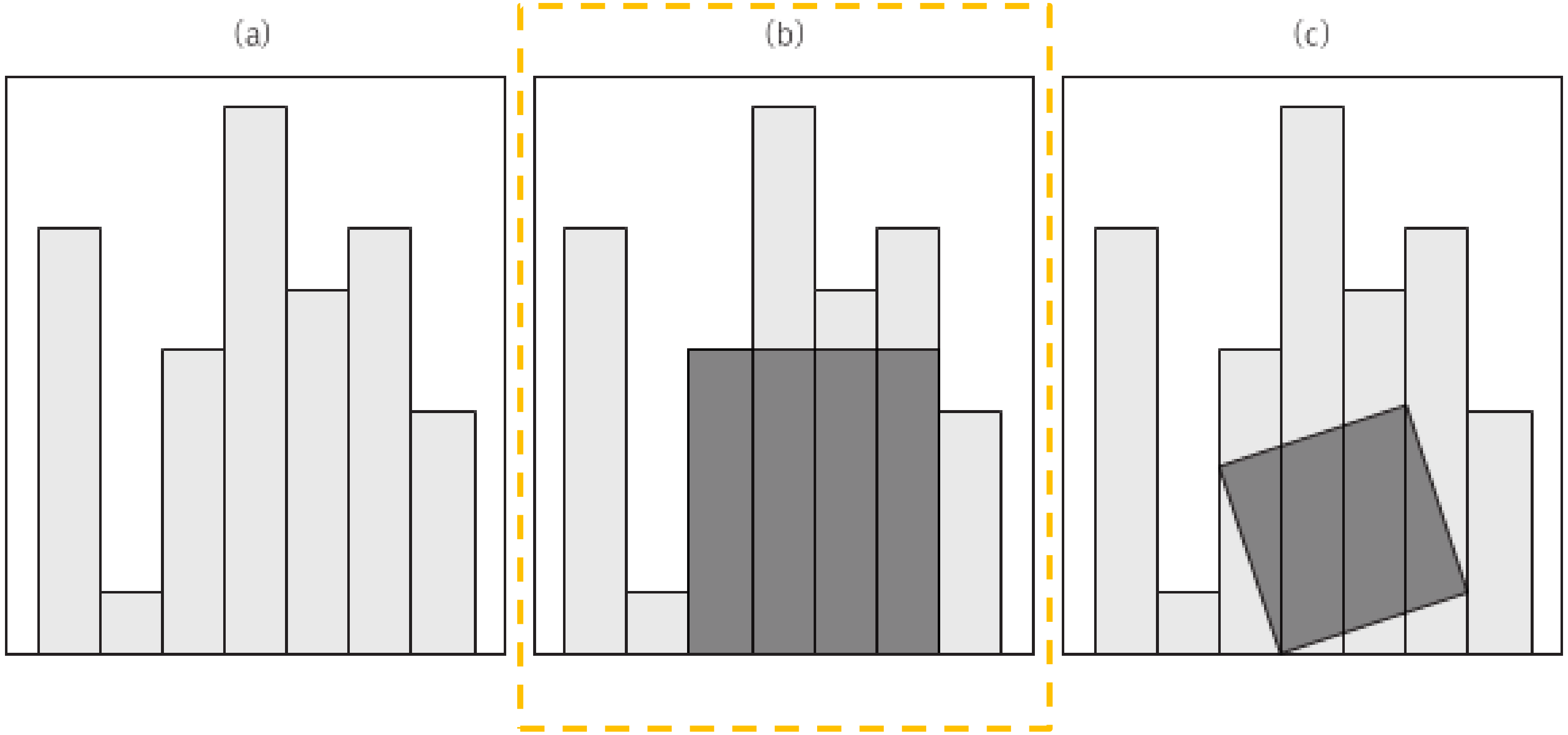
2) 바로 뒤집기



07. 분할 정복

예제5) 울타리 잘라내기

잘라 낼 수 있는 직사각형의 최대 크기

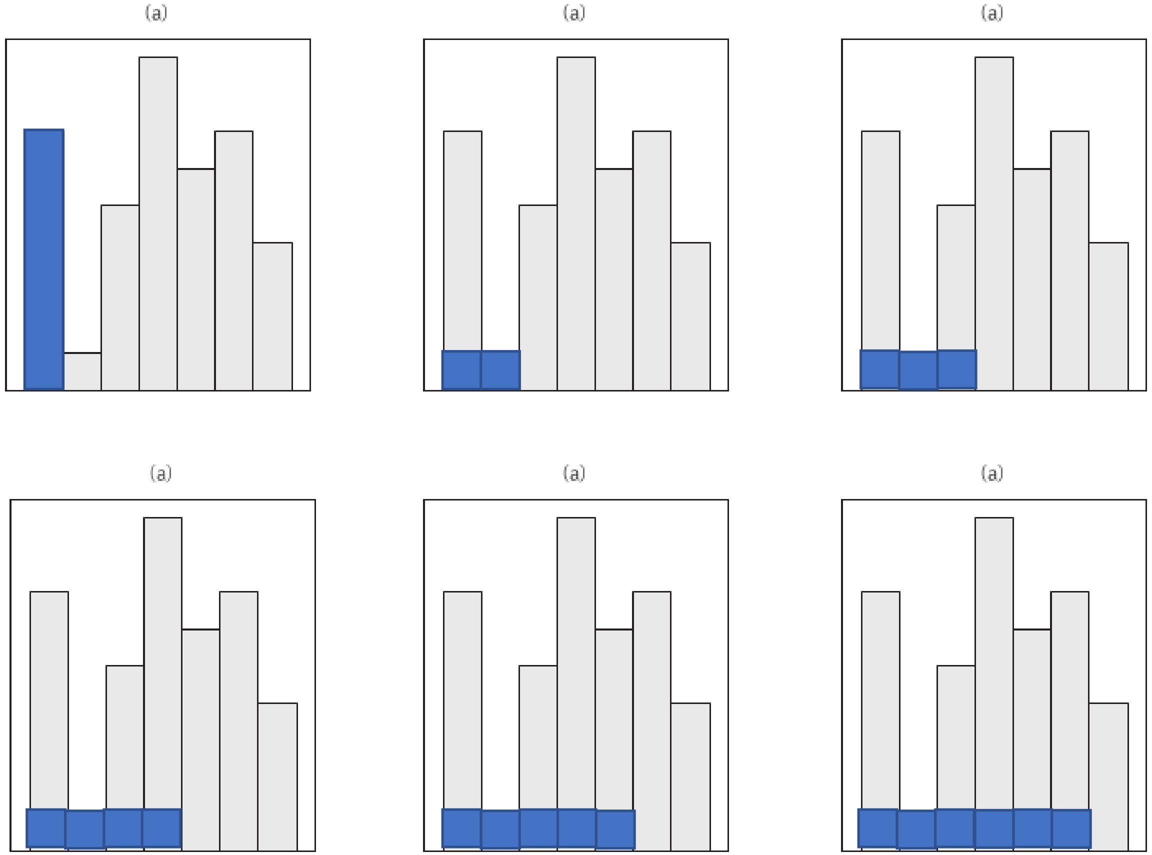


예제5) 울타리 잘라내기

잘라 낼 수 있는 직사각형의 최대 크기

1) BruteForce ~ $O(n^2)$

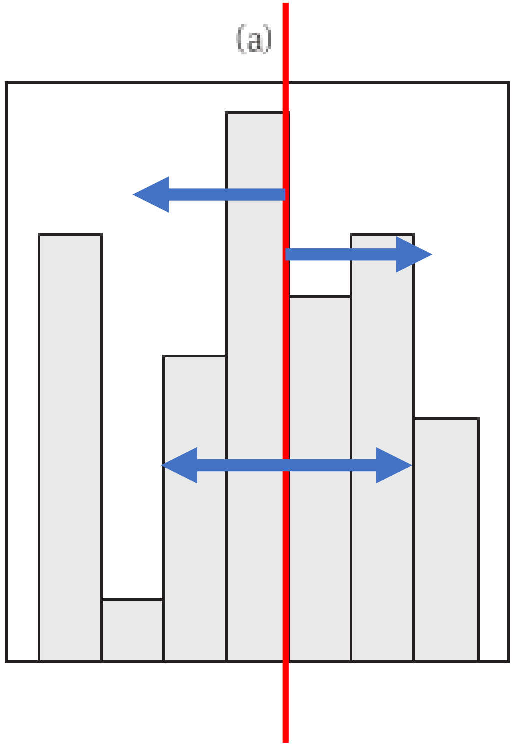
```python
def bruteForce(h):
    ret = 0
    n = len(h)
    for i in range(n):
        minHeight = h[i]
        for j in range(i, n):
            minHeight = min(minHeight, h[j])
            ret = max(ret, (j - i + 1) * minHeight)
    return ret
```



예제5) 울타리 잘라내기

잘라 낼 수 있는 직사각형의 최대 크기

2) Divide & Conquer ~ $O(n \log_2 n)$



- 가장 큰 직사각형을 왼쪽 부분 문제에서만 잘라낼 수 있다.
- 가장 큰 직사각형을 오른쪽 부분 문제에서만 잘라낼 수 있다.
- 가장 큰 직사각형은 왼쪽 부분 문제와 오른쪽 부분 문제에 걸쳐 있다.

07. 분할 정복

예제5) 울타리 잘라내기

잘라 낼 수 있는 직사각형의 최대 크기

2) Divide & Conquer ~ $O(n \log_2 n)$

```python
def divideConquer(left, right):
    if left == right:
        return h[left]
    mid = (left + right) // 2
    ret = max(divideConquer(left, mid), divideConquer(mid+1, right))

    lo = mid
    hi = mid + 1
    height = min(h[lo], h[hi])
    ret = max(ret, height * 2)

    while left < lo or hi < right:
        if hi < right and (lo == left or h[lo-1] < h[hi+1]):
            hi += 1
            height = min(height, h[hi])
        else:
            lo -= 1
            height = min(height, h[lo])

        ret = max(ret, height * (hi - lo + 1))
    return ret
```

절반으로 계속 나눠서 탐색
$O(\log_2 n)$

길이가 n인 전체를 탐색
$O(n)$

07. 분할 정복

결론

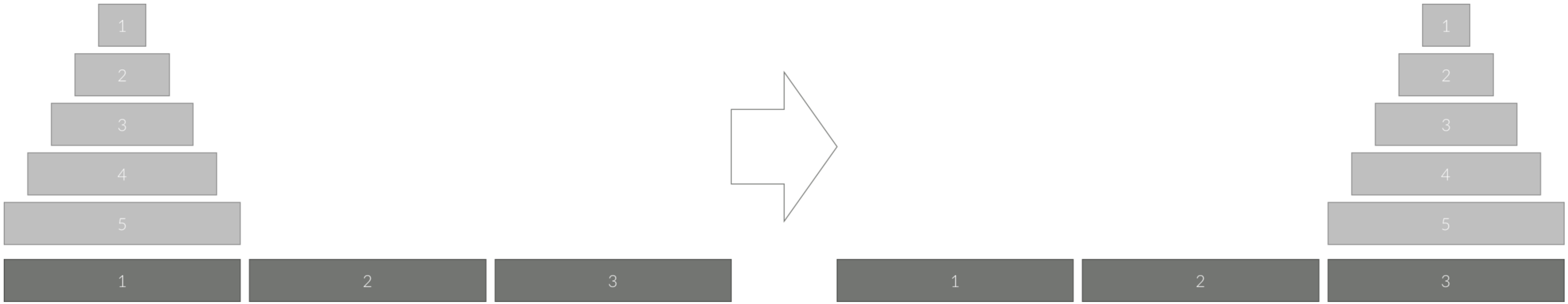
분할정복 알고리즘 ? 주어진 문제를 둘 이상의 부분 문제로 나눈 뒤 각 문제에 대한 답을 재귀 호출을 이용해 계산하고, 각 부분 문제의 답으로부터 전체 문제의 답을 계산해 내는 풀이 방법

- 문제를 더 작은 문제로 분할하는 과정 (divide)
- 각 문제에 대해 구한 답을 원래 문제에 대한 답으로 병합하는 과정 (merge)
- 더 이상 답을 분할하지 않고 곧장 풀 수 있는 매우 작은 문제 (base case)

조건 : 문제를 둘 이상 부분 문제로 나누는 자연스러운 방법이 있어야 하며, 부분 문제의 답을 조합해 원래 문제의 답을 계산하는 효율적인 방법이 있어야한다.

07. 분할 정복

예제6) 하노이 탑



1. 한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.

2.쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.

07. 분할 정복

예제6) 하노이 탑

```python
cnt = 0


def solve(n, x, y):
    global cnt
    if n == 0:
        return
    solve(n-1, x, 6 - (x + y))
    cnt += 1
    print(x, y)
    solve(n-1, 6 - (x + y), y)
```