점화식과 점근적 복잡도의 분석

교재: 쉽게 배우는 알고리즘: 관계 중심의 사고법
저자: 문병로
출판사: 한빛미디어, 2013년 발행

수업 목표

- 재귀 알고리즘과 점화식의 관계를 이해한다.
- 점화식의 점근적 분석을 이해한다.

점화식의 이해

- 점화식
  - 어떤 함수를 자신보다 더 작은 변수에 대한 함수와의 관계로 표현한 것
- 예
  - $a_n = a_{n-1} + 2$
  - $f(n) = n\,f(n-1)$
  - $f(n) = f(n-1) + f(n-2)$
  - $f(n) = f(n/2) + n$

- *여러 알고리즘의 수행 시간을 점화식으로 표현할 수 있다.*

용어를 알아보자!

as·ymp·tot·ic (asymptotical) 수학

미국식 [æsimtátik(əl)] | 영국식 [-tɔ́-]

두산동아 | YBM | 영영사전

형용사

형용사

(수학) **점근선의**

asymptotic property
점근성

an asymptotic circle[line]
점근원[선]

점근 漸近 [발음 : 점 : 근]

파생어 : 점근하다

명사

점점 가까워짐.

점화식

위키백과, 우리 모두의 백과사전.

수학에서 **점화식**(Recurrence relation)이란 수열의 항 사이에서 성립하는 관계식을 말한다. 즉, 수열 $\{a_n\}$ 의 각 항 a_n 이 함수 f를 이용해서

$$a_{n+1} = f(a_1, a_2, ..., a_n)$$

처럼 귀납적으로 정해져 있을 때, 함수 f를 수열 $\{a_n\}$ 의 점화식이라고 하며, 또한, 수열 $\{a_n\}$ 은 점화식 f로 정의된다고 한다.

점화식을 **푼다**는 것은 귀납적으로 주어진 수열 $\{a_n\}$ 의 일반항 a_n 을 n 의 명시적인 식(explicit formula)으로 나타내는 것을 말한다.

점화-식 漸化式 [발음 : 점 : 화식]

활용 : 점화식만[점 : 화싱만]

명사

<수학> **수열의 항(項) 사이에 성립하는 관계식.**

점화식과 점근적 표기법

- 점근적 표기법
  - 알고리즘의 수행시간을 분석하는 핵심 도구
  - $\boldsymbol{O}(n)$, $\boldsymbol{\Omega}(n)$, $\Theta(n)$
- 점화식으로 표현된 알고리즘의 점근적 분석을 통해 점근적 표기법으로 나타낼 수 있고 이를 통해 임의 알고리즘의 수행시간을 분석할 수 있다.
- 알고리즘 → 점화식 → 점근적 분석 → 점근적 표기

재능

반복재귀

점화식의 점근적 분석 방법

- 반복대치
  - 더 작은 문제에 대한 함수로 반복해서 대치해 나가는 해법
- 추정후 증명
  - 결론을 추정하고 수학적 귀납법으로 이용하여 증명하는 방법
- 마스터 정리
  - 형식에 맞는 점화식의 복잡도를 바로 알 수 있다.

반복 대치

pseudo code

의사코드

$\Theta(n)$

$\frac{sum(n)}{T(n)}$

```
factorial(n)  {
                                                C
   if (n=1) return 1 ;          ---------------- (1)

        return n * factorial(n-1) ;   ------------ (2)

}
```

$T(n)$, $T(n-1)$

✓ n에 비례하는 시간이 소요된다.

factorial, sum 재귀호출

$$\begin{cases} T(n) = T(n-1) + c \\ T(1) = 1 \ (\ T(1) \leq c) \end{cases}$$

1개자료 처리시간 $\leq$

$$
\begin{aligned}
T(n) &= T(n-1) + c \\
&= T(n-2) + c + c = T(n-2) + 2c \\
&= T(n-3) + c + 2c = T(n-3) + 3c \\
&\cdots \\
&= T(1) + (n-1)c \\
&\leq c + (n-1)c \\
&= cn \\
&= O(n)
\end{aligned}
$$

$T(1)=T\{n-(n-1)\}$

Mergesort (병합정렬)

```
mergeSort(A[ ], p, r)
▷ A[p ... r]을 정렬한다
{
    if (p < r) then {
        q ← (p+r)/2;  ------------------------  ① ▷ p, q의 중간 지점 계산
        mergeSort(A, p, q);  ----------------  ② ▷ 전반부 정렬
        mergeSort(A, q+1, r); --------------  ③ ▷ 후반부 정렬
        merge(A, p, q, r);  ------------------  ④ ▷ 병합
    }
}

merge(A[ ], p, q, r)
{
    정렬되어 있는 두 배열 A[p ... q]와 A[q+1 ... r]을 합하여
    정렬된 하나의 배열 A[p ... r]을 만든다.
}
```

Mergesort의 작동 예

정렬할 배열이 주어짐

31	3	65	73	8	11	20	29	48	15

배열을 반반으로 나눈다

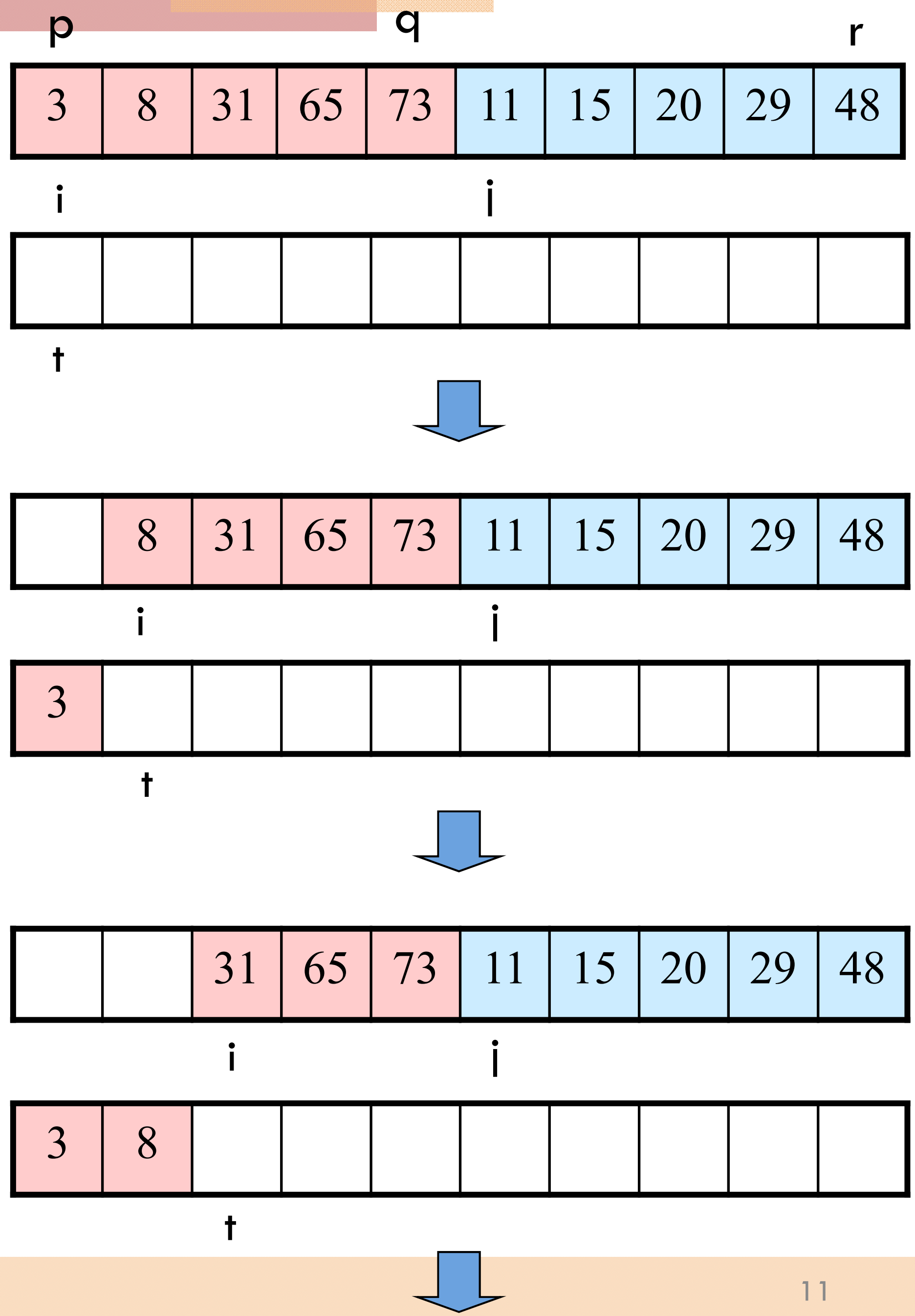
31	3	65	73	8	11	20	29	48	15	1

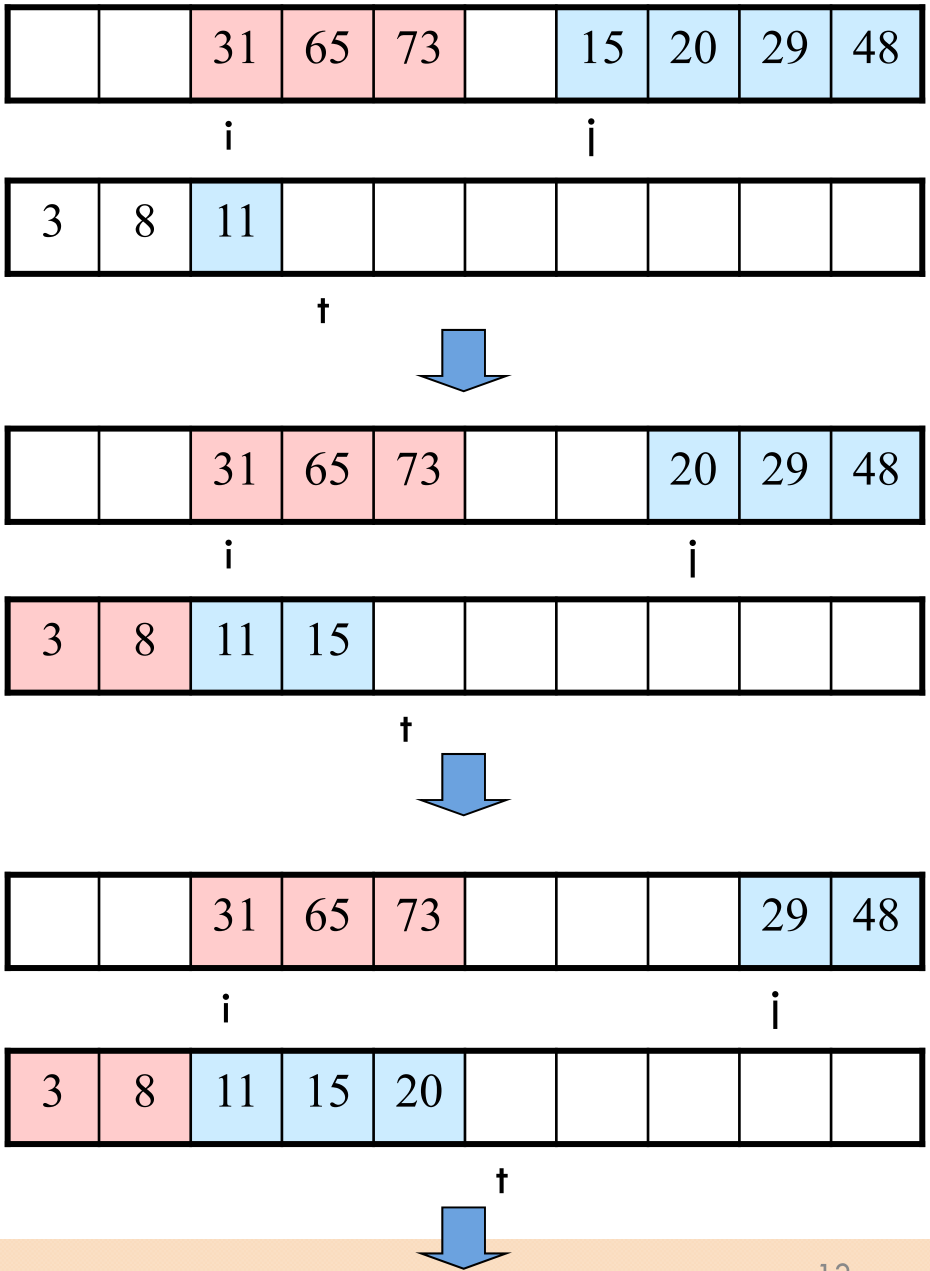
각각 독립적으로 정렬한다

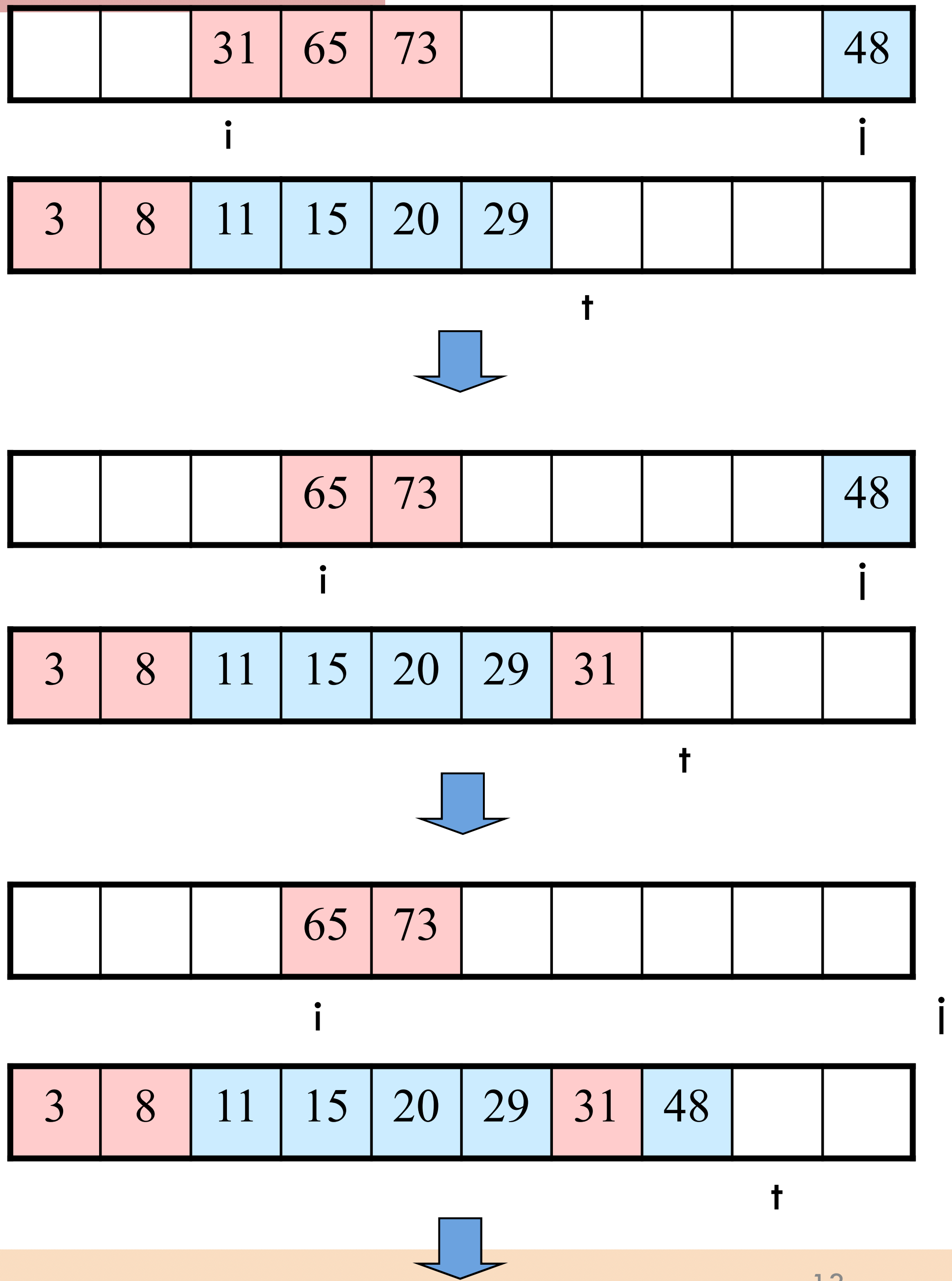
3	8	31	65	73	11	15	20	29	48	2 3

병합한다 (정렬완료)

3	8	11	15	20	29	31	48	65	73	4


p
q
r
3 8 31 65 73 11 15 20 29 48
i
j
t
8 31 65 73 11 15 20 29 48
i
j
3
t
31 65 73 11 15 20 29 48
i
j
3 8
t


31 65 73 15 20 29 48
i j
3 8 11
t
31 65 73 20 29 48
i j
3 8 11 15
t
31 65 73 29 48
i j
3 8 11 15 20
t


31 65 73 48
i j
3 8 11 15 20 29
t
65 73 48
i j
3 8 11 15 20 29 31
t
65 73
i j
3 8 11 15 20 29 31 48
t

i j

3	8	11	15	20	29	31	48	65	73

t

Mergesort의 수행시간

```
mergeSort(A[ ], p, r)
{
      if (p < r) then {
            q ← (p+r)/2;   ------------------------- ①  ▷ p, q의 중간 지점 계산
            mergeSort(A, p, q);  ---------------- ②  ▷ 전반부 정렬
            mergeSort(A, q+1, r); --------------- ③  ▷ 후반부 정렬
            merge(A, p, q, r);  ------------------ ④  ▷ 병합
      }
}
merge(A[ ], p, q, r)
{
      정렬되어 있는 두 배열 A[p ... q]와 A[q+1 ... r]을 합하여
      정렬된 하나의 배열 A[p ... r]을 만든다.
}
```

수행시간의 점화식: $T(n) = 2T(n/2) +$ 오버헤드

✓크기가 n인 병합정렬 시간은 크기가 n/2인 병합정렬을 2번 하고 나머지 오버헤드를 더한 시간이다

$$T(n) \le 2T\left(\frac{n}{2}\right) + n$$

$$T(1) = 1,\ n = 2^k,\ k = \log_2 n$$

$$\begin{aligned}
T(n) &\le 2T\left(\frac{n}{2}\right) + n \\
&\le 2\left(2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right) + n = 2^2 T\left(\frac{n}{2^2}\right) + 2n \\
&\le 2^2\left(2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\right) + 2n = 2^3 T\left(\frac{n}{2^3}\right) + 3n \\
&\ldots \\
&\le 2^k T\left(\frac{n}{2^k}\right) + kn = 2^k T(1) + kn \\
&\le n + n \log n \\
&= O(n \log n)
\end{aligned}$$

Question &
Answer