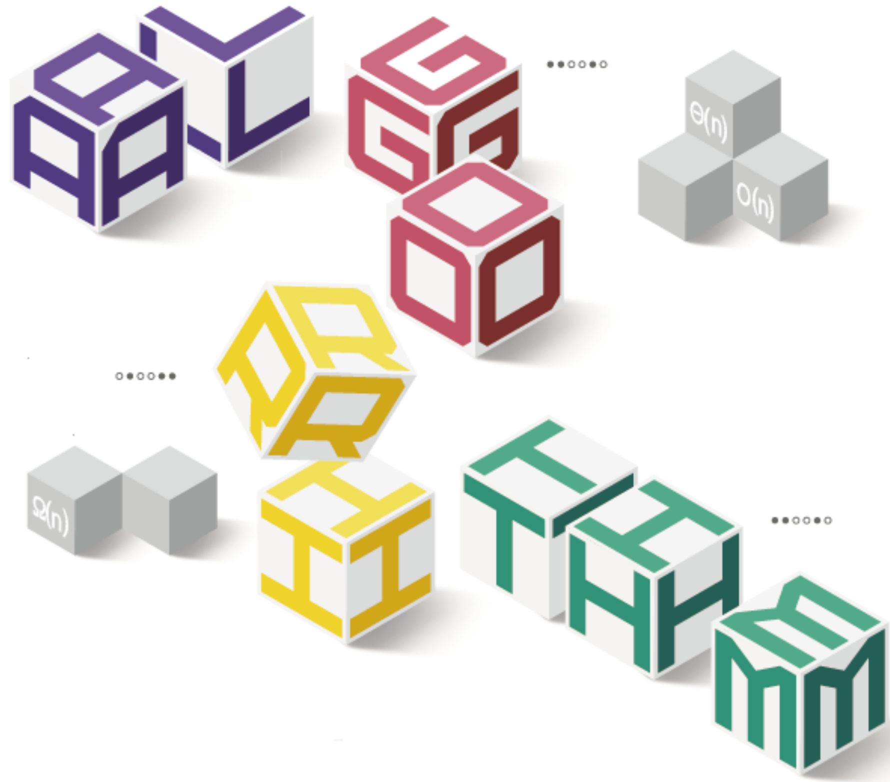


관계 중심의 사고법

쉽게 배우는 알고리즘

5장. 선택 알고리즘

5장. 선택 알고리즘

일을 시작하기 위해 기분이 내킬 때까지 기다리는 따위의 짓을
하지 않으려면 시험 제도는 좋은 훈련이 된다.

-아놀드 토인비

선택 알고리즘 (i번째 작은 수 찾기)

- 배열 A[p ... r]에서 i번째 작은 원소를 찾는다
- 두가지 알고리즘을 배운다
  - 평균적으로 선형시간이 소요되는 알고리즘
  - 최악의 경우에도 선형시간이 소요되는 알고리즘

$\Theta(n^2)$ 선택 알고리즘(원시적 방법)

```
select (A, p, r, i)
▷ 배열 A[p ... r]에서 i번째 작은 원소를 찾는다
{
        A[p ... r]에서 제일 작은 원소 A[k]를 찾는다;
        if (i == 1) then return A[k] ;
        else {
                A[k ... r-1] ← A[k+1 ... r];        ▷ shift left
                select (A, p, r-1, i-1)
        }
}
```

✓수행 시간: $\Theta(n^2)$

평균 선형시간 선택 알고리즘

select (A, p, r, i)
▷ 배열 A[p ... r]에서 i번째 작은 원소를 찾는다
{
    if (p = r) **then return** A[p] ;   ▷ 원소가 하나뿐인 경우. i는 반드시 1.
    $q \leftarrow$ partition(A, p, r) ;
    $k \leftarrow q$-p+1;   ▷ k : 기준원소가 전체에서 k 번째 작은 원소임을 의미
    if ($i < k$) **then return** select(A, p, q-1, i) ; ▷ 왼쪽 그룹으로 범위를 좁힘
    else if ($i = k$) **then return** A[q] ;   ▷ 기준원소가 바로 찾는 원소임
    else return select(A, q+1, r, i-k) ;   ▷ 오른쪽 그룹으로 범위를 좁힘
}

✓평균 수행 시간: $\Theta(n)$
✓최악의 경우 수행 시간: $\Theta(n^2)$

선택 알고리즘 작동 예 1

2번째 작은 원소 찾기

	p									r
입력배열	31	8	48	73	11	3	20	29	65	15

분할	8	11	3	15	31	48	20	29	65	73

왼쪽 그룹에서 2번째 작은 원소를 찾는다

8	11	3

선택 알고리즘 작동 예 2

7번째 작은 원소 찾기

입력배열 (p: 31, r: 15)

31	8	48	73	11	3	20	29	65	15

분할

8	11	3	15	31	48	20	29	65	73

오른쪽 그룹에서 3번째 작은 원소를 찾는다

				31	48	20	29	65	73

4개

평균 수행 시간

$$T(n) \le \frac{1}{n}\sum_{k=1}^{n} \boxed{\max[T(k\text{-}1),\ T(n\text{-}k)]} + \Theta(n)$$

분할된 양쪽 중 큰 쪽을 처리하는 비용

재귀호출을 제외한 오버헤드
(분할이 대부분)

이것은 $T(n) \le cn$임을 추정 후 증명법으로 증명할 수 있다
(뒷 페이지)

$$\therefore T(n) = O(n)$$

$T(n) = \Omega(n)$임은 자명하므로 $T(n) = \Theta(n)$

$$T(n) = \frac{1}{n}\sum_{k=1}^{n} T(\max(k-1, n-k) + \Theta(n)$$

$$= \frac{2}{n}\sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} T(k) + \Theta(n)$$

Guess $T(n) \leq cn$
for some $c > 0, n_0 \geq 0$ and $\forall n \geq n_0$

<= 이 맞다.
이유는, T(n)이 cn보다
작거나 같다고 가정을
했기 때문에 귀납적 가정에
의해 작거나 같다(upper bounding)

$$= \frac{2}{n}\sum_{k=\lfloor \frac{n}{2} \rfloor}^{n-1} ck + \Theta(n)$$

$$= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor - 1} k\right) + \Theta(n)$$

$$= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{\left(\left\lfloor\frac{n}{2}\right\rfloor - 1\right)\left\lfloor\frac{n}{2}\right\rfloor}{2}\right) + \Theta(n)$$

$$\le \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{\left(\frac{n}{2}-2\right)\left(\frac{n}{2}-1\right)}{2}\right) + \Theta(n)$$

$$\le c(n-1) - \frac{c}{n}\left(\frac{n^2}{4} - \frac{3n}{2} + 2\right) + \Theta(n)$$

$$\le cn + \left(-\frac{cn}{4} + \frac{c}{2} - \frac{2c}{n}\right) + \Theta(n)$$

$$\le cn$$

We can choose $c>0$ s.t. $-\frac{cn}{4}$ dominates $\left(\frac{c}{2} - \frac{2c}{n}\right) + \Theta(n)$

최악의 경우 수행 시간

$$T(n) = T(n\text{-}1) + \Theta(n)$$

재귀호출을 제외한 오버헤드
(분할이 대부분)

분할이 $0:n$-1로 되고 큰 쪽을 처리하는 비용

$$\therefore T(n) = \Theta(n^2)$$

최악의 경우 선형시간 선택 알고리즘

- 앞에서 배운 선택 알고리즘에서
  - 수행 시간은 분할의 균형에 영향을 받는다
- 분할이 항상 1:1이면
  - $T(n) = T\left(\frac{n}{2}\right) + \Theta(n) \rightarrow T(n) = \Theta(n)$
- 분할이 항상 3:1이면
  - $T(n) \leq T\left(\frac{3n}{4}\right) + \Theta(n) \rightarrow T(n) = O(n) \rightarrow T(n) = \Theta(n)$
- 이번 알고리즘은
  - 최악의 경우 분할의 균형이 어느 정도 보장되도록 함으로써 수행 시간이 $\Theta(n)$이 되도록 한다
  - 분할의 균형을 유지하기 위한 오버헤드가 지나치게 크면 안된다

linearSelect (A, p, r, i)

▷ 배열 $A[p \ldots r]$에서 i번째 작은 원소를 찾는다

{

1) base
2) & 3)

① 원소의 총 수가 5개 이하이면 원하는 원소를 찾고 알고리즘을 끝낸다.

② 전체 원소들을 5개씩의 원소를 가진 $\lceil n/5 \rceil$개의 그룹으로 나눈다.
(원소의 총수가 5의 배수가 아니면 이중 한 그룹은 5개 미만이 된다.)

③ 각 그룹에서 중앙값을 (원소가 5개이면 3번째 원소) 찾는다.
이렇게 찾은 중앙값들을 $m_1, m_2, \ldots, m_{\lceil n/5 \rceil}$이라 하자.

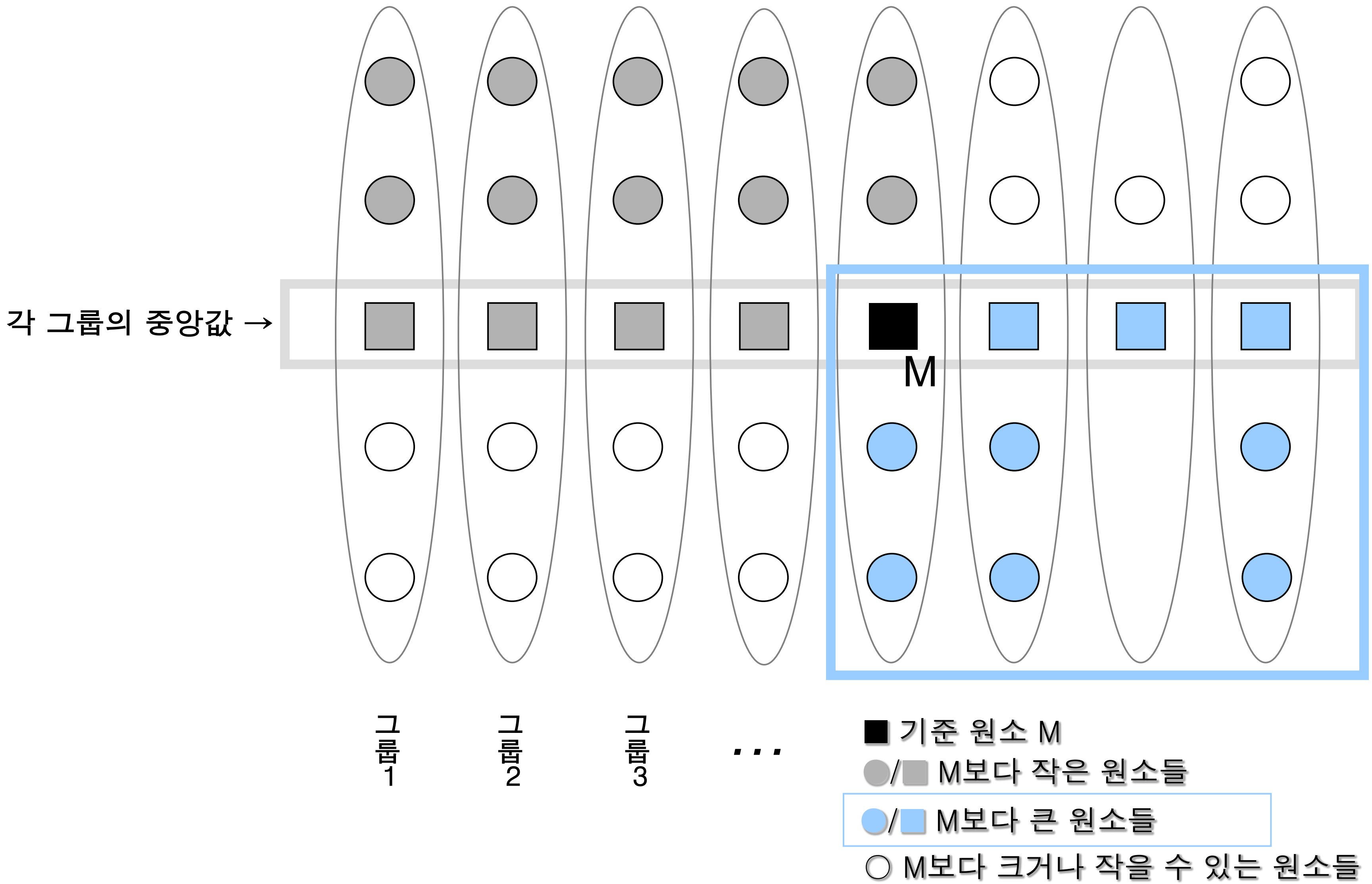
④ $m_1, m_2, \ldots, m_{\lceil n/5 \rceil}$들의 중앙값 M을 재귀적으로 구한다. 원소의 총수가 홀수면 중앙값이 하나이므로 문제가 없고, 원소의 총수가 짝수일 경우는 두 중앙값 중 아무거나 임의로 선택한다. ▷ call linearSelect( )

⑤ M을 기준원소로 삼아 전체 원소를 분할한다(M보다 작거나 같은 것은 M의 왼쪽에, M보다 큰 것은 M의 오른쪽에 오도록).

⑥ 분할된 두 그룹 중 적합한 쪽을 선택하여 단계 ①~⑥을 재귀적으로 반복한다. ▷ call linearSelect( )

}

기준 원소를 중심으로 한 대소 관계



최악의 경우 수행 시간

분할의 균형을 위한 overhead

$$T(n) \le \boxed{T(\lceil n/5 \rceil)} + T(7n/10 + 2) + \boxed{\Theta(n)}$$

④ ⑥ ①②③⑤

이것은 $T(n) \le cn$임을 추정 후 증명법으로 증명할 수 있다
(뒷 페이지)

$$\therefore T(n) = O(n)$$

적어도 linear time으로 한번씩의 비교는 일어나므로, 오메가 n은 자명하다!

$T(n) = \Omega(n)$임은 자명하므로 $T(n) = \Theta(n)$

$$T(n) \le T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 2\right) + \Theta(n)$$

1더해서 upper bound 없애주고, =이 아니라 <=을 쓴듯.

$$\le T\left(\frac{n}{5} + 1\right) + T\left(\frac{7n}{10} + 2\right) + \Theta(n)$$

n0는 n/5 +1 혹은 7n/10 +2 이 n보다 작은 범위를 말한다.

Assume $T(k) \le ck \;\forall k, n_0 \le k < n$
(귀납적 가정을 가장 충실하게 표현한 것)

$$\le c\left(\frac{n}{5} + 1\right) + c\left(\frac{7n}{10} + 2\right) + \Theta(n)$$

$$= c\left(\frac{9n}{10} + 3\right) + \Theta(n)$$

$$= cn - \frac{cn}{10} + 3c + \Theta(n)$$

따라서, T(n) = O(n)이 성립한다.

$$\le cn$$

We can choose $c>0$ s.t. $-\frac{cn}{10}$ dominates $3c + \Theta(n)$