05장. 선택 알고리즘

Youn-Hee Han
LINK@KOREATECH
http://link.koreatech.ac.kr

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

-아놀드 토인비

학습 목표

◈ 평균 선형 선택 알고리즘의 원리를 이해한다.

◈ 평균 선형 선택 알고리즘의 수행 시간 분석을 이해한다.

◈ 최악의 경우에도 선형 시간을 보장하는 선택 알고리즘의 원리 및 수행 시간 분석을 이해한다.

◈ 평균 선형 선택 알고리즘과 최악의 경우에도 선형 시간을 보장하는 선택 알고리즘의 관계를 이해한다.

선택 알고리즘

◈ 선택 알고리즘 문제 정의 (Find the ith smallest element in an array)

– 주어진 n개의 원소를 지닌 배열에서 i번째 작은 수 찾기

Input: A[] = {10, 3, 6, 9, 2, 4, 15, 23}, $i = 4$
Output: 6

Input: A[] = {5, −8, 10, 37, 101, 2, 9}, $i = 6$
Output: 37

- 주어진 조건
  - ➢ 주어진 배열 내 숫자는 임의로 주어진다.
  - ➢ 주어진 배열 내 숫자는 양수, 음수 또는 0이 될 수 있다.
  - ➢ $1 \leq i \leq n$

단순 전략 I

◈ 선택 알고리즘 단순 전략 I (Brute-Force)

- [문제]주어진 n개의 원소를 지닌 배열에서 i번째 작은 수 찾기
- [해결 전략]
  - 주어진 n개의 원소를 처음부터 마지막까지 살펴보면서 첫 번째 작은 수 찾기
  - 주어진 n개의 원소를 처음부터 마지막까지 살펴보면서 두 번째 작은 수 찾기
  - …
  - 주어진 n개의 원소를 처음부터 마지막까지 살펴보면서 i 번째 작은 수 찾기
- [점근적 복잡도]: $O(n^2)$

단순 전략 II

◈ 선택 알고리즘 단순 전략 II (정렬 알고리즘 활용)

- [문제]주어진 n개의 원소를 지닌 배열에서 i번째 작은 수 찾기
- [해결 전략]
  - 먼저 주어진 배열을 정렬한다.
  - 정렬된 배열에서 i번째 위치에 존재하는 수를 반환한다.

- [점근적 복잡도]
  - 정렬의 일반적인 점근적 복잡도: $O(nlogn)$
  - i번째 위치에 존재하는 수를 반환하는 복잡도: $O(1)$
  - 따라서, $O(nlogn)$

더 좋은 방법이 없을까?

◈ 선택 알고리즘 전략 고찰

– $O(nlogn)$ 보다 더 좋은, 즉 모든 경우에 $\Theta(n)$ 복잡도를 지니는 알고리즘을 만들 수는 없을까?

01. 평균 선형시간 선택 알고리즘

Quick Selection

◈ **Quick Selection 의사 코드**

- $p \le i \le r$

알고리즘 5-1	평균 선형 시간 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 ← partition(A, p, r);     ▷ [알고리즘 4-6]의 partition()과 동일
  ❶ k ← q−p+1;                  ▷ k : 기준원소가 전체에서 k번째 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
}
```

i**번째 원소는** $A[p]$

알고리즘 4-6	분할

$Quick\ Sort$ **에서 제시된 것과 동일**

```
partition(A[], p, r)
{
    x ← A[r];          ▷ 기준원소
    i ← p−1;           ▷ i는 1구역의 끝지점
    for j ← p to r−1   ▷ j는 3구역의 시작 지점
        if (A[j] ≤ x) then A[++i] ↔ A[j];
                       ▷ 의미는 i값 증가 후 A[i] ↔ A[j] 교환
    A[i+1] ↔ A[r];     ▷ 기준원소와 2구역 첫 원소 교환
    return i+1;
}
```

Quick Selection

알고리즘 5-1 평균 선형 시간 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 ← partition(A, p, r);       ▷ [알고리즘 4-6]의 partition()과 동일
  ❶ k ← q-p+1;                    ▷ k : 기준원소가 전체에서 k번째 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
}
```

◈ Quick Selection 수행 예

– 전체에서 2번째 작은 원소 찾기

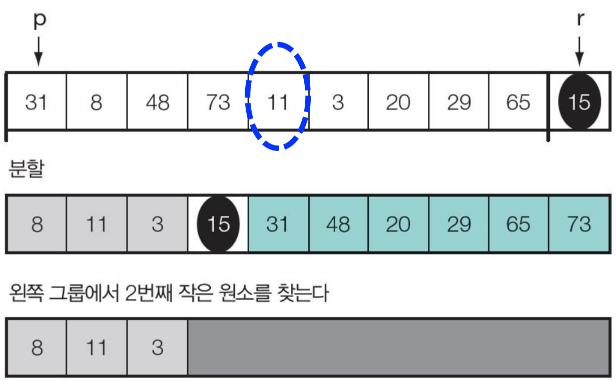

그림 5-1 선택 알고리즘의 작동 예 1 : 2번째 작은 원소 찾기

$i = 2, \quad q = 4, \quad k \leftarrow q - p + 1 = 4 - 1 + 1 = 4$

기준원소는 인덱스가 p부터 시작하는 배열에서 4번째 작은 원소

전체에서 2번째 작은 원소는
왼쪽 그룹에서 여전히 2(=i)번째 작은 원소임
→ 재귀 호출 $select(A, p, q - 1, i) = select(A, 1,3,2)$

– 전체에서 7번째 작은 원소 찾기

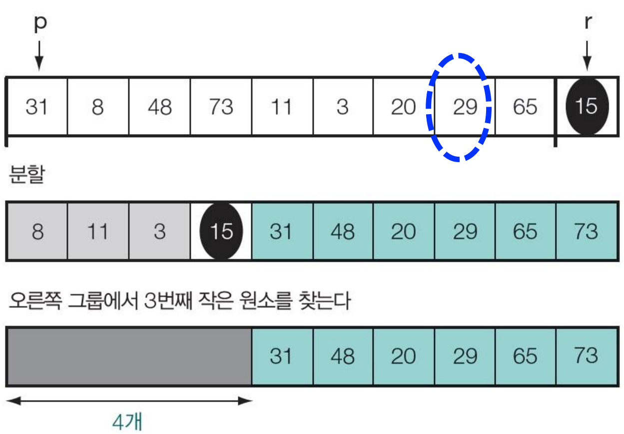

그림 5-2 선택 알고리즘의 작동 예 2 : 7번째 작은 원소 찾기

$i = 7, \quad q = 4, \quad k = q - p + 1 = 4 - 1 + 1 = 4$

기준원소는 인덱스가 p부터 시작하는 배열에서 4번째 작은 원소

전체에서 7번째 작은 원소는
오른쪽 그룹에서 3($= i - k$)번째 작은 원소임
→ 재귀 호출 $select(A, q + 1, r, i - k) = select(A, 5,10,3)$

Quick Selection

◈ Quick Selection 점근적 복잡도 분석

알고리즘 5-1	평균 선형 시간 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 ← partition(A, p, r);      ▷ [알고리즘 4-6]의 partition()과 동일
 ❶ k ← q-p+1;                   ▷ k : 기준원소가 전체에서 k번째 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
}
```

분할 [알고리즘 4-6]: $\Theta(n)$

평균의 경우 알고리즘 수행 시간의 점화식
(점화식에 부등호가 있는 이유: 기준원소가 찾고 있는 원소일 때를 고려하기 때문)

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

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

$$\cdots$$

$$\le cn$$

따라서, $T(n) = \Theta(n)$
자세한 증명은 생략
(교재 142페이지 참조)

Quick Selection

◈ Quick Selection 점근적 복잡도 분석

알고리즘 5-1	평균 선형 시간 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 ← partition(A, p, r);     ▷ [알고리즘 4-6]의 partition()과 동일
  ❶ k ← q−p+1;                 ▷ k : 기준원소가 전체에서 k번째 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
}
```

최악의 경우 알고리즘 수행 시간의 점화식

$$T(n) = T(n-1) + \Theta(n)$$

$T(n) - T(n-1) = n$
$T(n-1) - T(n-2) = n-1$
$T(n-2) - T(n-3) = n-2$
$\cdots$
$T(3) - T(2) = 3$
$T(2) - T(1) = 2$
———————————————————
$T(n) - T(1) = 2 + \cdots + (n-2) + n$
➔
$T(n) = \frac{n(n+1)}{2}$ $(\because T(1) = 1)$

따라서,

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

→ 즉, 분할의 균형이 매번 극단적으로 불균형일 때

02. 최악의 경우에도 선형 시간을 보장하는 선택 알고리즘

1:9 분할 가정 → $T(n) = \Theta(n)$

◈ 1:9 분할 가정을 통한 점근적 복잡도 산출

- 재귀 호출시마다 항상 1:9로 분할이 되고 더 큰 분할 쪽으로 계속 탐색을 한다고 가정

알고리즘 5-1 평균 선형 시간 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 ← partition(A, p, r);      ▷ [알고리즘 4-6]의 partition()과 동일
  ❶ k ← q-p+1;                   ▷ k : 기준원소가 전체에서 k번째 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
}
```

- 최악의 경우 알고리즘 수행 시간의 점화식 $T(n) = T\left(\frac{9n}{10}\right) + \Theta(n)$

1:9 분할 가정 → $T(n) = \Theta(n)$

◈ 1:9 분할 가정을 통한 점근적 복잡도 산출

– 최악의 경우 알고리즘 수행 시간의 점화식

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

- 마스터 정리를 활용한 점근적 복잡도 산출 [Page 150] 연습문제 06번

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad \text{for } a \geq 1 \text{ and } b > 1$$

➢ $a = 1,\ b = \frac{10}{9},\ f(n) = \Theta(n)$

➢ 추가적인 함수 $h(n)$ 정의: $h(n) = n^{\log_b a} = n^0 = 1$

➢ 따라서, $\lim_{n \to \infty} \frac{f(n)}{h(n)} = \infty$ 이고, 충분히 큰 모든 n에 대해

$f\left(\frac{9n}{10}\right) \leq f(n)$이므로, $T(n) = \Theta(f(n)) = \Theta(n)$이다.

1:99 분할 가정 → $T(n) = \Theta(n)$

◈ 1:99 분할 가정을 통한 점근적 복잡도 산출

- 재귀 호출시마다 항상 1:99로 분할이 되고 더 큰 분할 쪽으로 계속 탐색을 한다고 가정
- 최악의 경우 알고리즘 수행 시간의 점화식 $T(n) = T\left(\frac{99n}{100}\right) + \Theta(n)$

  - 마스터 정리를 활용한 점근적 복잡도 산출 [Page 151] 연습문제 07번

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \qquad \text{for } a \geq 1 \text{ and } b > 1$$

➢ $a = 1,\ b = \frac{100}{99},\ f(n) = \Theta(n)$

➢ 추가적인 함수 $h(n)$ 정의: $h(n) = n^{\log_b a} = n^0 = 1$

➢ 따라서, $\lim_{n\to\infty} \frac{f(n)}{h(n)} = \infty$ 이고, 충분히 큰 모든 n에 대해 $f\left(\frac{99n}{100}\right) \leq f(n)$이므로, $T(n) = \Theta(f(n)) = \Theta(n)$이다.

1:n 분할 가정 → $T(n) = \Theta(n)$

◈ 분할의 균형이 아주 나빠 보여도 일정한 상수비만 유지하며 분할이 된다면 점근적 복잡도는 항상 $T(n) = \Theta(n)$

– 즉, 분할의 균형을 어느 정도 까지 한정할 수 있음을 보장할 수 있으면 점근적 복잡도는 항상 $T(n) = \Theta(n)$

– 하지만, 분할의 균형을 맞추는 데 필요한 오버헤드가 발생!

선형 시간 보장 선택 알고리즘

◈ 선형 시간 보장 선택 알고리즘

알고리즘 5-2	최악의 경우 선형 시간 선택 알고리즘

linearSelect(A, p, r, i) ▷ $A[p \cdots r]$에서 i번째 작은 원소를 찾는다.

{

1 원소의 총수가 5개 이하이면 i번째 원소를 찾고 알고리즘을 끝낸다.

2 전체 원소를 5개씩의 원소를 가진 $\lceil \frac{n}{5} \rceil$개의 그룹으로 나눈다.
(원소의 총수가 5의 배수가 아니면 이 중 한 그룹은 5개 미만이 된다.)

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

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

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

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

}

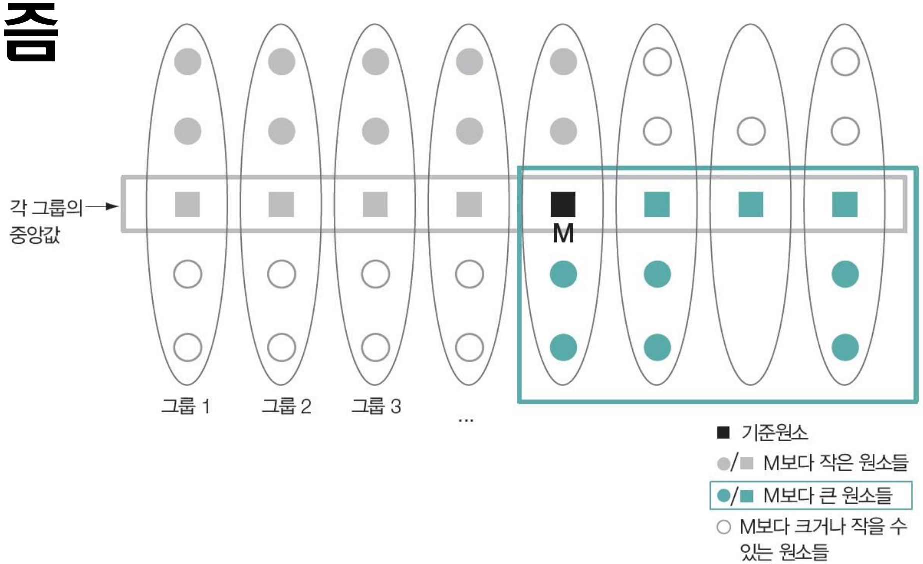

그림 5-3 기준원소를 중심으로 분할된 상황을 그림으로 표현한 예

선형 시간 보장 선택 알고리즘

◈ 선형 시간 보장 선택 알고리즘

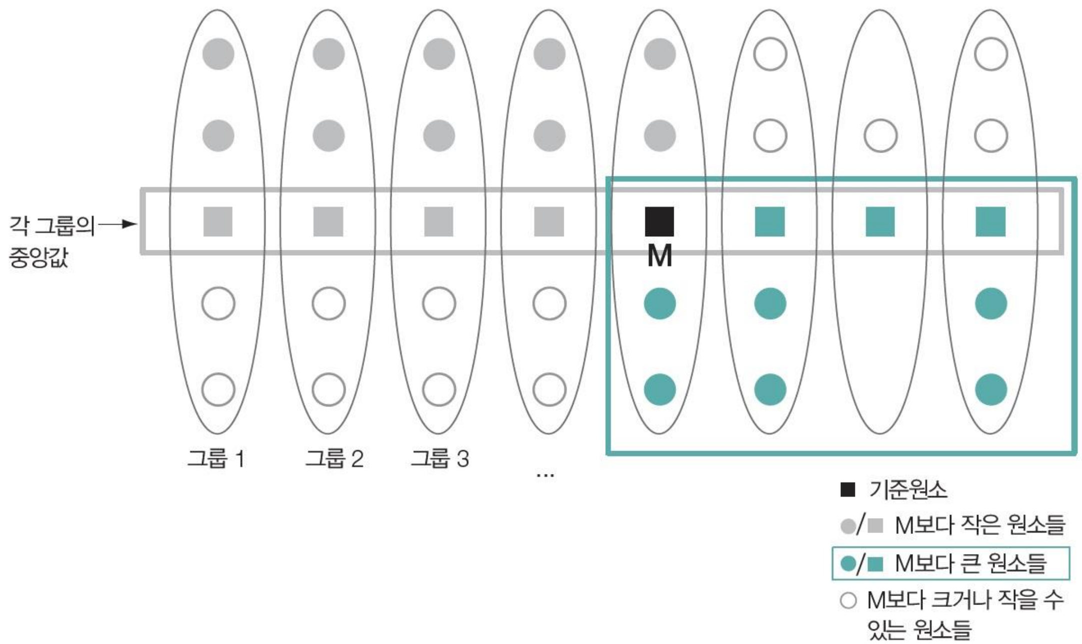

그림 5-3 기준원소를 중심으로 분할된 상황을 그림으로 표현한 예

기준 원소: M

이 그룹의 최소 원소 개수

$$3 \times \frac{n}{10} - 2$$

(∵전체 원소를 5개씩 나누면 그룹의 수는 $\frac{n}{5}$
다시 그것을 절반으로 나누면 그룹의 수는 $\frac{n}{10}$
각 그룹당 3개의 원소를 가지며
마지막 그룹에는 많으면 2개의 원소 부재)

따라서, 두 개 그룹의 원소 비율 $= \frac{3n}{10} - 2 : n - \left(\frac{3n}{10} - 2\right) = \frac{3n}{10} - 2 : \frac{7n}{10} + 2 \approx 3:7$

즉, **분할의 균형을 어느 정도 까지 한정할 수 있음을 보장** 가능 → $T(n) = \Theta(n)$

그렇다면, 이러한 분할을 하는 데 소요되는 오버헤드는?

선형 시간 보장 선택 알고리즘

◈ 선형 시간 보장 선택 알고리즘 분석

– linearSelect 알고리즘 수행시간: $T(n)$

알고리즘 5-2	최악의 경우 선형 시간 선택 알고리즘

```
linearSelect(A, p, r, i)        ▷ A[p … r]에서 i번째 작은 원소를 찾는다.
{
    1 원소의 총수가 5개 이하이면 i번째 원소를 찾고 알고리즘을 끝낸다.
    2 전체 원소를 5개씩의 원소를 가진 ⌈n/5⌉개의 그룹으로 나눈다.
      (원소의 총수가 5의 배수가 아니면 이 중 한 그룹은 5개 미만이 된다.)
    3 각 그룹에서 중앙값(원소가 5개이면 3번째 원소)을 찾는다.
      이렇게 찾은 중앙값들을 m_1, m_2, …, m_⌈n/5⌉이라 하자.
    4 m_1, m_2, …, m_⌈n/5⌉들의 중앙값 M을 재귀적으로 구한다.
      원소의 총수가 홀수이면 중앙값이 하나이므로 문제가 없고,
      원소의 총수가 짝수이면 두 중앙값 중 임의로 선택한다.    ▷ call linearSelect()
    5 M을 기준원소로 삼아 전체 원소를 분할한다(M보다 작거나 같은 것은 M의 왼쪽에,
      M보다 큰 것은 M의 오른쪽에 오도록).                    ▷ call partition()
    6 분할된 두 그룹 중 적합한 쪽을 선택해 단계 1~6을 재귀적으로 반복한다.
                                                              ▷ call linearSelect()
}
```

단계 1: $\Theta(1)$

단계 2: $\Theta(n)$

단계 3: 각 그룹의 크기는 항상 5임. 따라서, 그룹 내에서 3번째 원소 찾는 작업: $\Theta(1)$
총 그룹의 개수: $\left\lceil \frac{n}{5} \right\rceil$
그러므로, $\Theta(n)$

단계 4: 대상 개수: $\left\lceil \frac{n}{5} \right\rceil$
다시, 이들의 중앙값을 선택하는 문제이므로,
결국 linearSelect() 호출 필요
그러므로, $T\left(\left\lceil \frac{n}{5} \right\rceil\right)$

단계 5: 분할 [알고리즘 4-6]
그러므로, $\Theta(n)$

단계 5: 더 큰 그룹의 원소 개수: $\frac{7n}{10} + 2$. 그러므로,
최악의 경우: $T\left(\frac{7n}{10} + 2\right)$

선형 시간 보장 선택 알고리즘

◈ 선형 시간 보장 선택 알고리즘 분석

– linearSelect 알고리즘 수행시간: $T(n)$

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

- 위에서 기준원소를 잘 선택하는 오버헤드 부분은 $T\left(\left\lceil \frac{n}{5} \right\rceil\right) + \Theta(n)$
- 이 식을 조금 더 전개하면 다음과 같음

$$\begin{aligned} T(n) &\le T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 2\right) + \Theta(n) \\ &\le T\left(\frac{n}{5} + 1\right) + T\left(\frac{7n}{10} + 2\right) + \Theta(n) \end{aligned}$$

선형 시간 보장 선택 알고리즘

◈ 선형 시간 보장 선택 알고리즘 분석

– linearSelect 알고리즘 수행시간: $T(n)$

점근적 복잡도 유도

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

(귀납적 가정) $n_0 \le k \le n$인 모든 k에 대해서 $T(k) \le ck$라고 가정 (n_0는 경계치)

(귀납적 전개)

$\frac{n}{5}+1 \le n$ 그리고 $\frac{7n}{10}+2 \le n$ 이라는 가정 → 즉, $n \ge 7$
→ 따라서, 경계치 $n_0 = 7$

$$\begin{aligned}
T(n) &\le T\left(\frac{n}{5}+1\right)+T\left(\frac{7n}{10}+2\right)+\Theta(n)\\
&\le c\left(\frac{n}{5}+1\right)+c\left(\frac{7n}{10}+2\right)+\Theta(n)\\
&= c\left(\frac{9n}{10}+3\right)+\Theta(n)\\
&= cn-\frac{cn}{10}+3c+\Theta(n)\\
&\le cn
\end{aligned}$$

$-\frac{cn}{10}$가 $3c+\Theta(n)$을 압도할 수 있는 c 선택 가능

따라서, <u>최악의 경우에도</u> $T(n) = \Theta(n)$

Questions & Answers