[CSE3081(2반)] 알고리즘 설계와 분석

2020학년도 2학기

강의자료

(2020.10.08 목요일)

서강대학교 공과대학 컴퓨터공학과
임 인 성 교수

[주제 3]

Divide-and-Conquer Techniques

and

Sorting Techniques

Selection of the *k*-th Smallest Element

- **Problem:** Given a sequence of S of *n* elements and an integer *k* (1 <= *k* <= *n*), find the *k*-th smallest element of *S*.

- **Solution 1:** Choose the smallest element repeatedly k times.

$$C = c(n-1) + c(n-2) + c(n-3) + \cdots + c(n-k) = c \cdot k \cdot n - c \cdot \frac{k(k+1)}{2}$$

$$\text{If } k = \frac{n}{2}, \text{ then } C = c \cdot \frac{n^2}{2} - c \cdot \frac{n^2+2n}{8} = O(n^2).$$

- **Solution 2:** Build a min-heap, and then extract the smallest element repeatedly k times.

$$C = c \cdot n + d \cdot k \cdot \log n$$

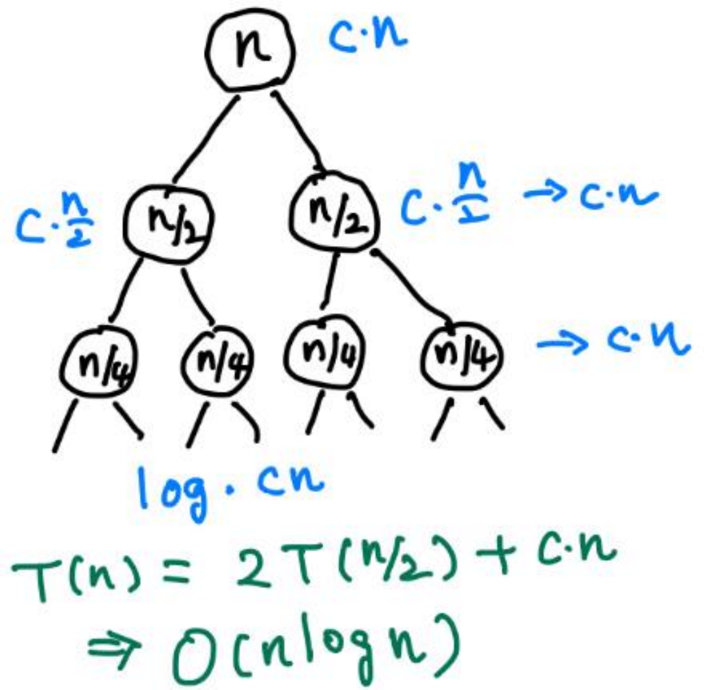
$$\text{If } k = \frac{n}{2}, \text{ then } C = c \cdot n + d \cdot \frac{n}{2} \cdot \log n = O(n \log n).$$

➢ **Can we design an O(n)-time algorithm?**

Observation

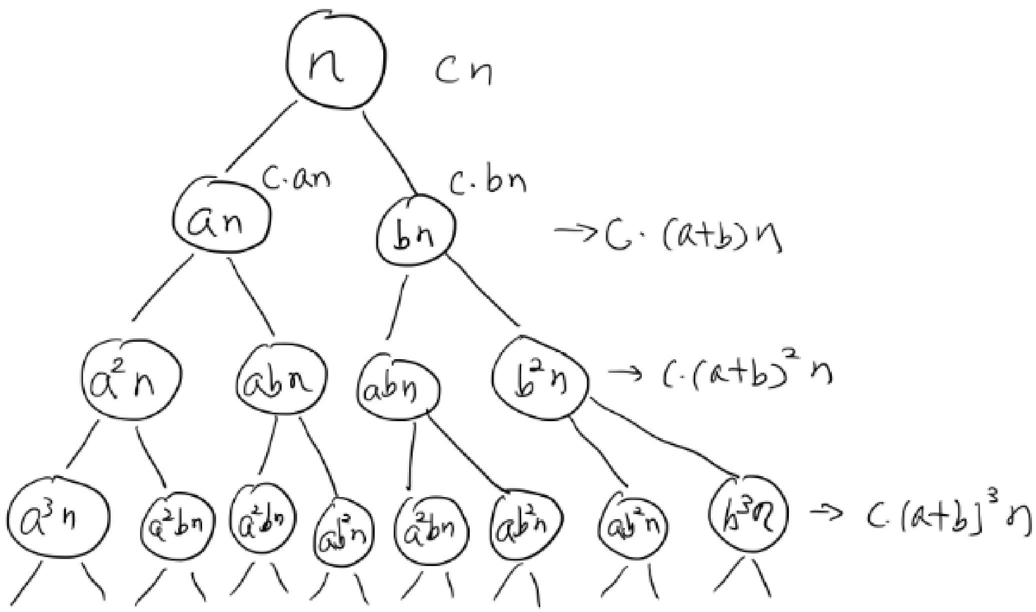
- **At least O(n) time is necessary.**

- **If we use a divide-and-conquer scheme like the merge sort,**



What about $T(n) = 3T(n/3) + cn$?

- Can we design an O(n)-time algorithm for this selection problem?
  - What about **$T(n) = T(an) + T(bn) + cn$ with $a + b < 1$**?



$$\text{Cost}: \; cn\{1 + (a+b) + (a+b)^2 + \cdots\}$$

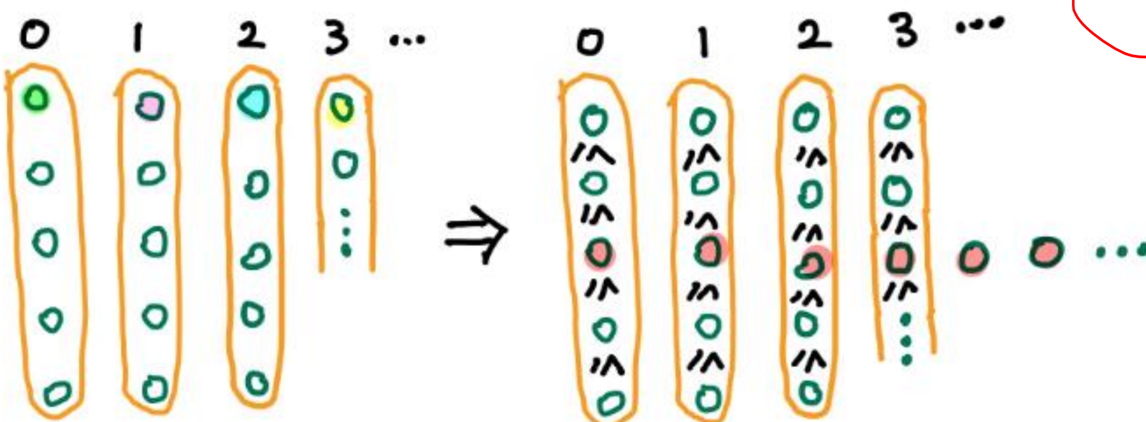
$$\leq cn \frac{1}{1-(a+b)} \Rightarrow O(n)$$

Algorithm

$\lfloor 21/5 \rfloor = 4$ $\lceil 21/5 \rceil = 5$

- **Step 1:** Divide S into floor(|S|/5) sequence of 5 elements each with up to four leftover elements. n

- **Step 2:** Sort each 5-element sequence.



$|M| = n/5$

$n = 200$

$k = \frac{n}{5} \cdot \frac{1}{2} = \frac{n}{10}$

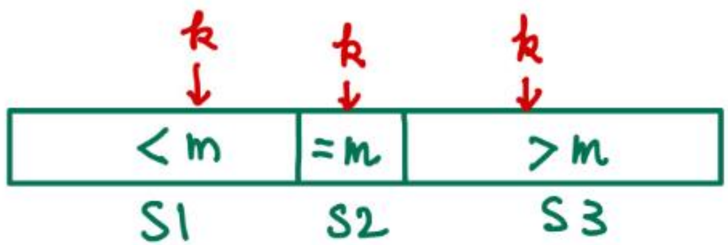
- **Step 3:** Let M be the sequence of medians of the 5-element sets.
  Then, let m be the median of the elements in M.

- **Step 4:** Let **S1**, **S2**, and **S3** be the sequences of elements in S **less than**, **equal to**, and **greater than m**, respectively.
  - **If** |S1| >= k, **then** find the **k-th** smallest element of **S1**.
  - **else if** (|S1| + |S2| >= k), then **m** is the k-th smallest element of S.
  - **else** find the **(k – |S1| - |S2|)-th** smallest element of **S3**.

$|S1| = \alpha n$

$|S3| = \alpha n$

$\frac{1}{5} + \alpha < 1$



$n \quad n/5 + \alpha \cdot n$

$n > \frac{n}{5} + \alpha n$

$T(n) = T(\frac{n}{5}) + T(\alpha n) + cn$

A divide-and-conquer strategy

```
    procedure SELECT(k, S):
1.  if |S| < 50 then
        begin
2.          sort S;
3.          return kth smallest element in S
        end
    else
        begin
4.          divide S into ⌊|S|/5⌋ sequences of 5 elements each
5.          with up to four leftover elements;
6.          sort each 5-element sequence;
7.          let M be the sequence of medians of the 5-element sets;
8.          m ← SELECT(⌈|M|/2⌉, M);
9.          let S1, S2, and S3 be the sequences of elements in S less
              than, equal to, and greater than m, respectively;
10.         if |S1| ≥ k then return SELECT(k, S1)
            else
11.             if (|S1| + |S2| ≥ k) then return m
12.             else return SELECT(k − |S1| − |S2|, S3)
        end
```

Fig. 3.10. Algorithm to select kth smallest element.

- **Facts**
  - **At least one-fourth of the elements of *S*** are less than or equal to *m*.
  - **At least one-fourth of the elements of *S*** are greater than or equal to *m*.
  - $|S_1| <= 3n/4$ and $|S_3| <= 3n/4$

S_1: the set of all elements less than m

S_2: the set of all elements equal to m

S_3: the set of all elements greater than m

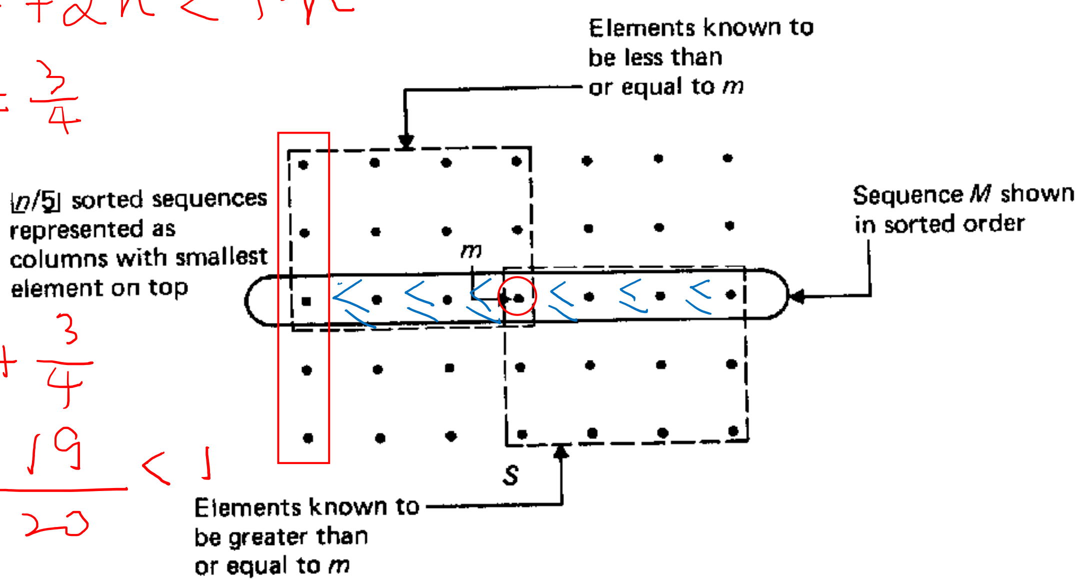
M

S1 or S3

At least $floor(\frac{n}{10})$ elements of M are greater than or equal to m.
For each of these elements, there are two distict elements of S which are at least as large.
Thus, $|S_1| \leq n - 3 * floor(\frac{n}{10}) \rightarrow |S_1| \leq \frac{3n}{4}$ for $n \geq 50$.

$\frac{n}{5} + \alpha n < 1 \cdot n$

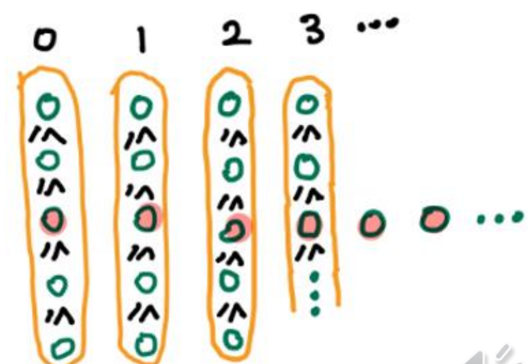
$\alpha = \frac{3}{4}$



$\frac{1}{5} + \frac{3}{4}$

$= \frac{19}{20} < 1$

Fig. 3.11 Partitioning of *S* by Algorithm 3.6.



- **Time Complexity**

$$\begin{cases} T(n) \le d & \text{for } n \le 49 \\ T(n) \le T(\frac{n}{5}) + T(\frac{3n}{4}) + cn & \text{for } n \ge 50 \end{cases} \longrightarrow T(n) \le kn \text{ for } k = \max\{d, 20c\}$$

- Input size n = |S|
- |M| <= ceil(n/5)
- |S1| <= 3n/4
- |S3| <= 3n/4

```
        procedure SELECT(k, S):
1.      if |S| < 50 then
            begin
2.              sort S;
3.              return kth smallest element in S
            end
        else
            begin
4.              divide S into ⌊|S|/5⌋ sequences of 5 elements each
5.              with up to four leftover elements;
6.              sort each 5-element sequence;
7.              let M be the sequence of medians of the 5-element sets;
8.              m ← SELECT(⌈|M|/2⌉, M);
9.              let S1, S2, and S3 be the sequences of elements in S less
                  than, equal to, and greater than m, respectively;
10.             if |S1| ≥ k then return SELECT(k, S1)
                else
11.                 if (|S1| + |S2| ≥ k) then return m
12.                 else return SELECT(k − |S1| − |S2|, S3)
            end
```

Fig. 3.10. Algorithm to select kth smallest element.

Selection Algorithm: Complexity Analysis

Theorem For some positive constants c and d, if the following recurrence relation holds:

$$\begin{cases} T(n) \leq d, & \text{for } n \leq 49, \\ T(n) \leq T(\frac{n}{5}) + T(\frac{3n}{4}) + cn, & \text{for } n \geq 50, \end{cases}$$

then $T(n) = O(n)$.

Proof We want prove that $T(n) \leq kn$ for some constant k and for all $n \geq 1$.

i) *Base case*

$T(n) \leq d \leq dn$ for all $n \geq 1$. Therefore, $T(n) \leq kn$ for all $1 \leq n \leq 49$ if we select k such that $k \geq d$.

ii) *Inductive step*

Assume that $n \geq 50$ and $T(m) \leq km$ for all $m < n$. Then,

$$\begin{aligned} T(n) &\leq T(\frac{n}{5}) + T(\frac{3n}{4}) + cn \\ &\leq k\frac{n}{5} + k\frac{3n}{4} + cn = \frac{19}{20}kn + cn \\ &= kn + (c - \frac{k}{20})n \leq kn \text{ if } k \geq 20c. \end{aligned}$$

So, if we choose k such that $k = \max(d, 20c)$, $T(n) \leq kn$ for all $n \geq 50$, completing the proof. □

Master Theorem 1

[Neapolitan 2.8]

- Let *a*, *b*, and *c* be nonnegative constants. The solution to the recurrence $\boldsymbol{T(1) = 1}$, and $\boldsymbol{T(n) = aT(n/c) + bn}$, for $n > 1$ for n a power of *c* is

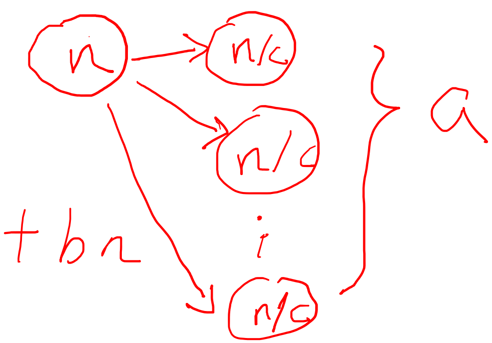
  ① $T(n) = O(n)$, if $a < c$, $n > \frac{a}{c} n$

  ② $T(n) = O(n \log n)$, if $a = c$,

  ③ $T(n) = O(n^{\log_c a})$, if $a > c$.

Prove this by induction!

$a = 3$
$c = 2$



$O(n^{\log_2 3}) \longleftarrow T(n) = 3T(\frac{n}{2}) + bn$

- Avoid divided-and-conquer if, for example,
  - An instance of size *n* is divided into two or more instances each almost of size *n*.
  - An instance of size *n* is divided into almost *n* instance of size *n*/*c*, where *c* is a constant.

$n^{\log_2 3} = n^{1.59} = n \cdot n^{0.59}$

The divide-and-conquer strategy often leads to efficient algorithms, although not always!