Analysis and Design of Algorithms

Lesson6: Selection Problem

张涵翠(Zhang Hancui)

Email：zhc@zstu.edu.cn

Selection Problem

✓ Selection Problem

- ***Problems: select the k-th smallest element from an array***
- ***Also called "Order Statistics"(顺序统计量)***
  - ***1-st smallest: minimum***
  - ***n-th smallest: maximum, if the length of array is n***
  - $\left\lceil \frac{n+1}{2} \right\rceil$***-th or*** $\left\lfloor \frac{n+1}{2} \right\rfloor$***-th smallest: median***
- ***Q: how to do selection? What's the time cost?***

Selection Problem

✓ Selection Problem

Case 1: *select the minimum or maximum element*

IDEA :

Comparison & Record

$$T(n) = n - 1$$

Minimum(A, n):

```
1.min = A[1]
2.for i <- 2 to n do
3.   if A[i] < min
4.       min = A[i]
5.return min
```

Selection Problem

✓ Selection Problem

Case 2: ***select the minimum and maximum element***

IDEA **:**

Solve them independently

$$\boldsymbol{T(n) = 2n - 3} \Leftarrow \begin{cases} Cost_{min} = n - 1 \\ Cost_{max} = n - 1 - 1 \end{cases}$$

Selection Problem

✓ Selection Problem

Case 2: ***select the minimum and maximum element***

IDEA :

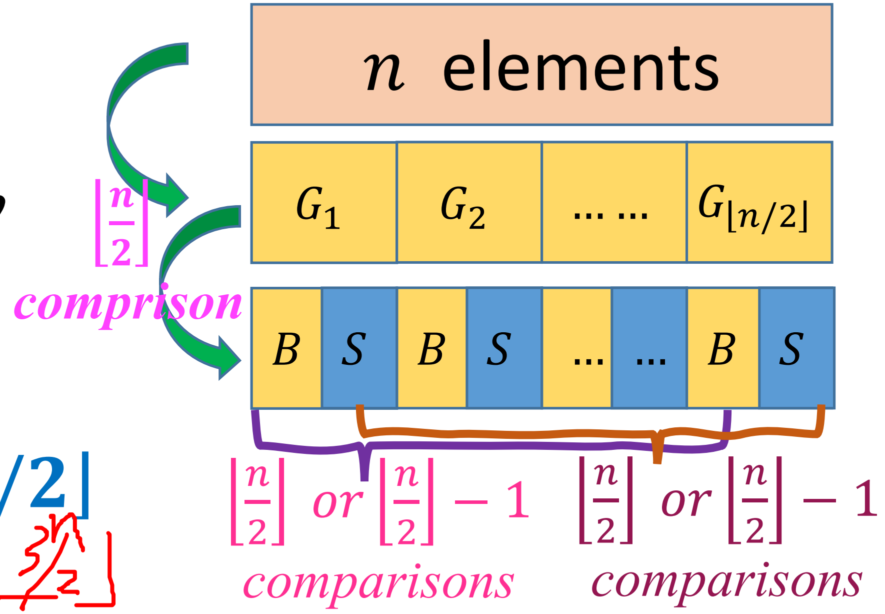
Solve them simultaneously

Divide into pairs

$\Rightarrow$ $\boldsymbol{T(n) = 3\lceil n/2 \rceil - 2 \text{ or } 3\lfloor n/2 \rfloor}$

$\lfloor \frac{3n}{2} \rfloor$

$T(n) = 2n - 3$

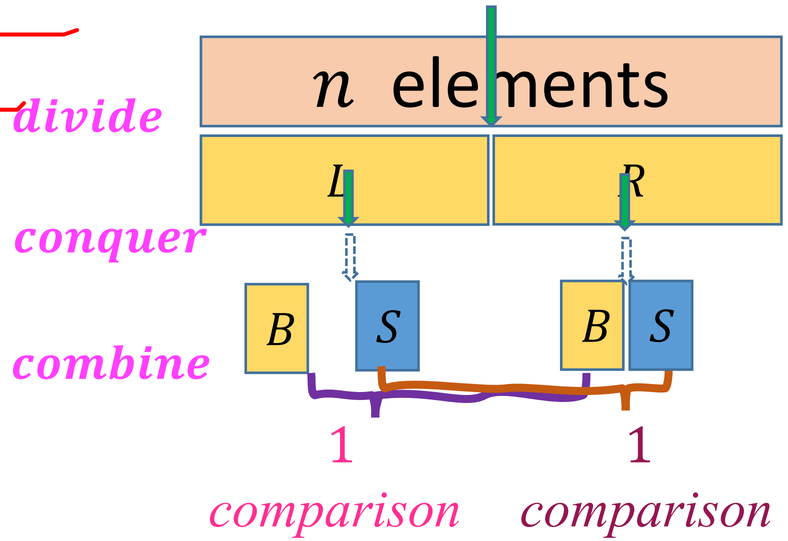


✓ Selection Problem

$T(n) = 2n - 3$ $\quad O(2n)$

Case 2: *select the minimum and maximum element*

IDEA: Merge sort



Divide & Conquer

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

$$\underline{T(2) = 1}$$

Assume $n = 2^k$

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + 2\right] + 2 = \cdots$$

$$= 2^{k-1}T\left(\frac{n}{2^{k-1}}\right) + 2^{k-1} + \cdots + 2^1$$

$$= 2^{k-1} + 2^{k-1} + \cdots + 2^1 = 3 \times 2^{k-1} - 2 = \frac{3n}{2} - 2$$

$$O\left(\frac{3n}{2}\right)$$

Selection Problem

✓ Selection Problem

Case 3: ***(General problem) select the k-th smallest element***

Method 1:

IDEA : ***Select the minimum element at every turn, execute k times***

➔ $T(n) = O(kn)$

$k \rightarrow \frac{n}{2}$ ➔ n^2

worst

Selection Problem

✓ Selection Problem

Case 3: ***(General problem) select the k-th smallest element***

Method 2:

IDEA : ***Sort A + index the k-th element A[k]***

➡ $T_{selection} = T_{sorting}(n)$

$= \Theta(nlogn)$

- ***Best:*** $\Theta(n)$
- ***worst:*** $\Theta(n^2)$

Q1: Is there any way to make the selection in expected linear time?

✓ Selection Problem

Case 3: *(General problem) select the k-th smallest element*

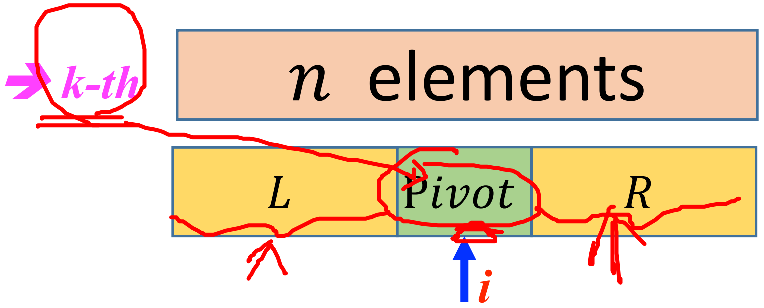
Method 3:

IDEA : *Divide&Conquer*

➡ **similar to quicksort**
but works on only one side

$$E[T_{quicksort}(n)] = \Theta(nlogn)$$

$$E[T_{selection}(n)] = \Theta(n)$$



1) $k = i$ ➔ $return\ A[i]$
2) $k \geq i$ ➔ $R\ subarry$
3) $k < i$ ➔ $L\ subarry$

Selection Problem

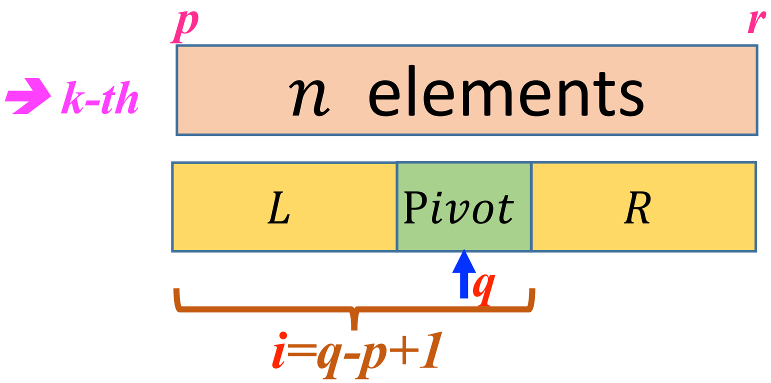
✓ Selection Problem

Case 3: ***(General problem) select the k-th smallest element***

```
RandomSelect(A,p,r,k):
if p == r return A[p]
q = Randomized-Partition(A,p,r)
i = q-p+1
if k == i return A[q]
else if k < i return RandomSelect(A,p,q-1,k)
else return RandomSelect(A,q+1,r,k-i)
```

the k-th element

#elements in lower subarray



Selection Problem

✓ Selection Problem

Example

A=

6 (p, i)	10 (j)	13	5	8	3	2	11 (r)

➔ *Find k-th smallest*

➔ *k = 7*

$k - 6$

Step1: *partition* *x=6*

4 3

2 (p)	5	3	6	8 (i)	13 (j)	10	11 (r)

q-p+1=4 **q=4**

Step2: *1 subarray* ***k=3*** *partition* ***x=8***

8 (q=1)	13 (i)	10 (j)	11

Step3: *1 subarray* ***k=2*** *partition* ***x=13***

11	10	13 (q=3)

Step4: *1 subarray* ***k=2*** *partition* ***x=11***

10	11 (q=2)

➔ ***7-th smallest element is : 11***

Selection Problem

✓ Selection Problem --*Divide&Conquer:* ***analyze***

◆ ***any split of constant proportionality yields a balanced partition***

➔***best-case***

e.g. $\frac{1}{10}$ & $\frac{9}{10}$ ➔$T(n) = T\left(\frac{9n}{10}\right) + \Theta(n)$

Master method:

$a = 1, b = \frac{10}{9}, n^{log_b^a} = n^{log_{10/9}^1} = n^0 = 1$

$f(n) = \Theta(n) = \Omega(n^{log_b^a + \varepsilon})$ ➔**Case 3:**$T(n) = \Theta(f(n))$
$= \Theta(n)$

Selection Problem

✓ **Selection Problem** ***--Divide&Conquer: analyze***

- ***Unbalanced partition: 0 & n-1*** ➔ ***worst-case***

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

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

arithmetic series

Selection Problem

✓ Selection Problem --*Divide&Conquer: analyze*

- ***Expectation:*** *like quicksort, depends on partition* ➔ *average-case*
  - ✓ *Needs a random indicator: j,* $0 \le j \le n-1$
  - ✓ *Needs a random variable:* x_j, $0 \le j \le n-1$

$$x_j = \begin{cases} 1, & \text{generate a split } j: n-j-1 \\ 0, & \text{otherwise} \end{cases}$$

$$T(n) = \begin{cases} j=0: & max\{T(0), T(n-1)\} + \Theta(n) \\ j=1: & max\{T(1), T(n-2)\} + \Theta(n) \\ & \dots \\ j=n-1: & max\{T(n-1), T(0)\} + \Theta(n) \end{cases}$$

all cases

$$E[T(n)] = E\left[\sum_{j=0}^{n-1} x_j \left(max\{T(j), T(n-j-1)\} + \Theta(n)\right)\right]$$

Selection Problem

✓ **Selection Problem** ***--Divide&Conquer: analyze***

Calculating expectation:

$$
\begin{aligned}
E[T(n)] &= E\left[\sum_{j=0}^{n-1} x_j\left(max\{T(j), T(n-j-1)\} + \Theta(n)\right)\right] \\
&= \sum_{j=0}^{n-1} E\left[x_j\left(max\{T(j), T(n-j-1)\} + \Theta(n)\right)\right] \\
&= \sum_{j=0}^{n-1} E\left[x_j\right] \cdot E[max\{T(j), T(n-j-1)\} + \Theta(n)] \\
&= \sum_{j=0}^{n-1} \frac{1}{n} \cdot E[max\{T(j), T(n-j-1)\} + \Theta(n)] \\
&= \frac{1}{n}\sum_{j=0}^{n-1} E[max\{T(j), T(n-j-1)\} + \frac{1}{n}\sum_{j=0}^{n-1} E[\Theta(n)] \\
&= \frac{2}{n}\sum_{j=\lfloor n/2 \rfloor}^{n-1} E[T(j)] \ + \Theta(n) \\
&\le cn
\end{aligned}
$$

(Use substitution method to prove)

Selection Problem

✓ Selection Problem --*Divide&Conquer: analyze*

Calculating expectation:

Guess:

$E[T(n)] \le cn$

$$E[T(n)] = \frac{2}{n}\sum_{j=\lfloor n/2\rfloor}^{n-1} E[T(j)] + \Theta(n)$$

$$\le \frac{2}{n}\sum_{j=\lfloor n/2\rfloor}^{n-1} cj + \Theta(n)$$ *(arithmetic series)*

$$= \frac{2c}{n} \times \frac{3n^2-2n}{8} + \Theta(n)$$

$$= \frac{3c}{4}n - \frac{c}{2} + \Theta(n)$$

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

$$\le cn$$

➔ $c \to \infty, for\ all\ n \ge n_0$
there is $\frac{c}{4}n + \frac{c}{2} - \Theta(n) \ge 0$

selection in expected linear time

Selection Problem

✓ Selection Problem *--Divide&Conquer: analyze*

- ***Make the worst-case in linear time***
  - ***The running time relies on the partition of pivot***
  - ***Randomized-Selection: expected in linear time with the high probability***
  - ***Worst-case: the situation we always need to do consideration***

Q2: how to get a good pivot without the high probability to lower $T(n)$?

Selection Problem

✓ Selection Problem --*Divide&Conquer:* ***analyze***

- ***Make the worst-case in linear time***

Idea:

generate pivot recursively ➔ guarantee a good split

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \quad \textbf{\textit{VS.}} \quad T(n) = T\left(\frac{n}{2}\right) + \Theta(n)$$

➔ ***Make the sum of recursions cost $\leq$ cn***

➔ ***Select***

Selection Problem

✓ Selection Problem *--Divide&Conquer: analyze*

- ***Make the worst-case in linear time*** ➔ ***Select***

Select(A,k):

1.Divide *n* into $g = \lceil n/5 \rceil$ groups, each has 5 elements

2.Recursively Sort&Select the median m of each group g, $(0 \le g \le \lceil n/5 \rceil)$,
and Let $M = \{m_g, 0 \le g \le \lceil n/5 \rceil\}$

3. x <- Select(M, $\lceil |M|/2 \rceil$)

4.Partition A around x, *get 4 parts:* p_1, p_2, p_3, p_4

5.Let $\mathrm{i} = \#\mathrm{element} \le x$, then x = i-th smallest

6.if k == i return x

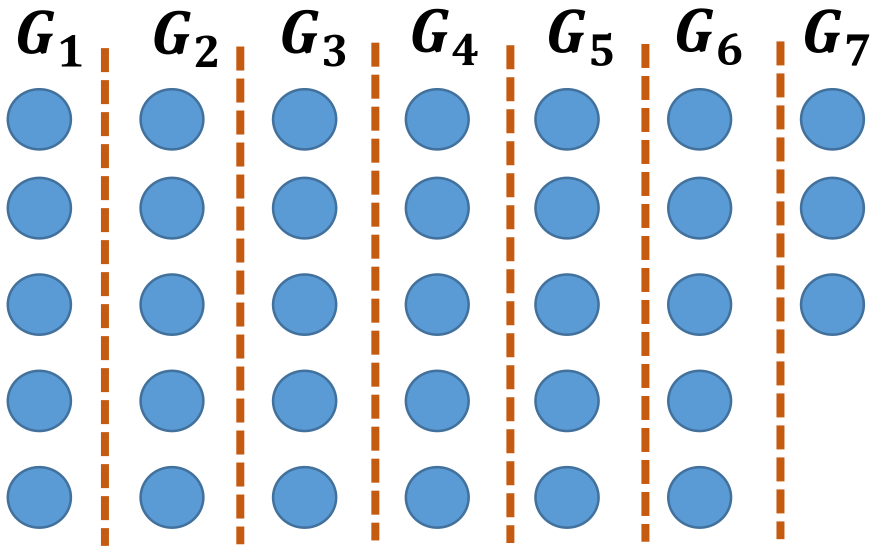
7.else if k < i return Select(A',i)

8.else return Select(A'', k-i)

✓ Selection Problem --*Divide&Conquer:* ***analyze***

Example:

$n = 33$



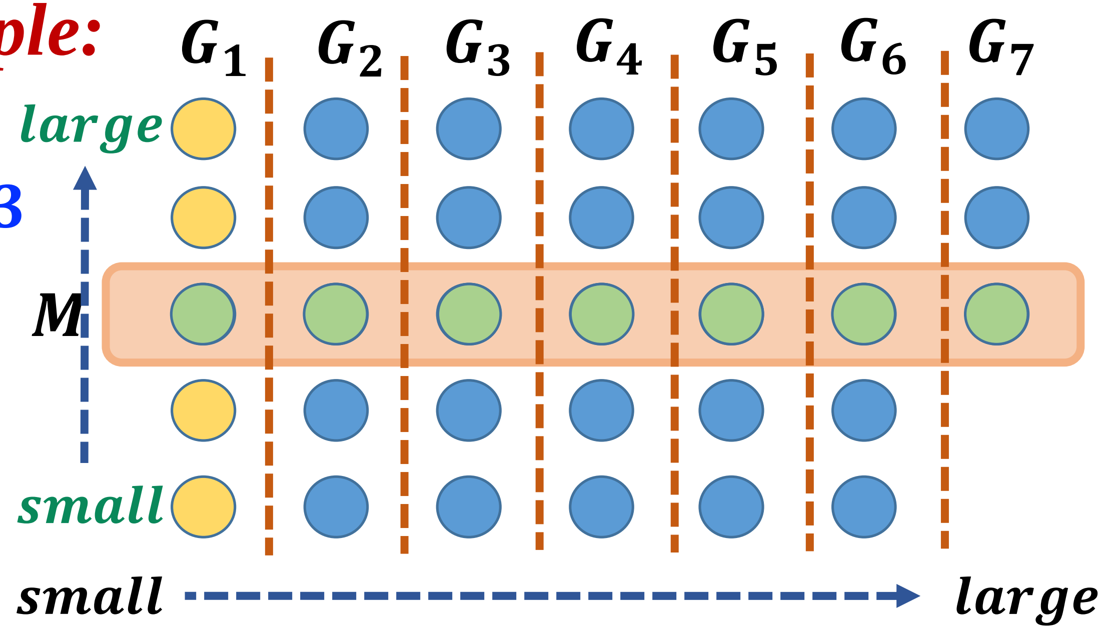
Step$_1$: *divide into* $\lceil n/5 \rceil$ *groups*

$\lceil 5/2 \rceil = 3$

Step$_2$: *recursively select* median *item of each group*

✓ Selection Problem --*Divide&Conquer: analyze*

Example:



$Step_1$: *divide into* $\lceil n/5 \rceil$ *groups*

$\lceil 5/2 \rceil = 3$

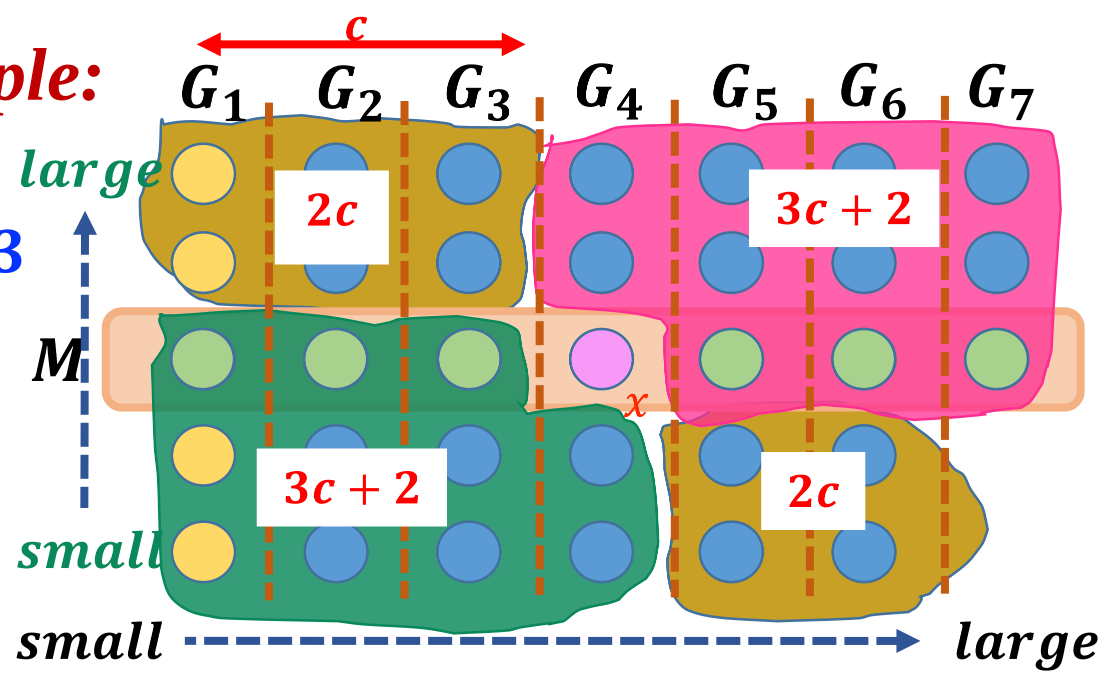
$Step_2$: *recursively select median item* m *of each group*

$Step_3$: *find x: the median of* M

Selection Problem

✓ Selection Problem --*Divide&Conquer: analyze*

Example:



Step$_1$: *divide into* $\lceil n/5 \rceil$ *groups* $\lceil 5/2 \rceil = 3$

Step$_2$: *recursively select median item* m *of each group*

Step$_3$: *find x: the median of M* → *get 4 fields*

Step$_4$: *compare with position of x vs. k*

let c *be the* ***column*** *of left-up field*

$\#totalColumn \Rightarrow \mathbf{2c+1} = \lceil n/5 \rceil$

→ $\mathbf{n} = 5 \times \#totalColumn = 5(2c+1)$

$\#left_{up} \Rightarrow 2c$ $\#right_{up} \Rightarrow 3c+2$

$\#left_{bottom} \Rightarrow 3c+2$ $\#right_{bottom} \Rightarrow 2c$

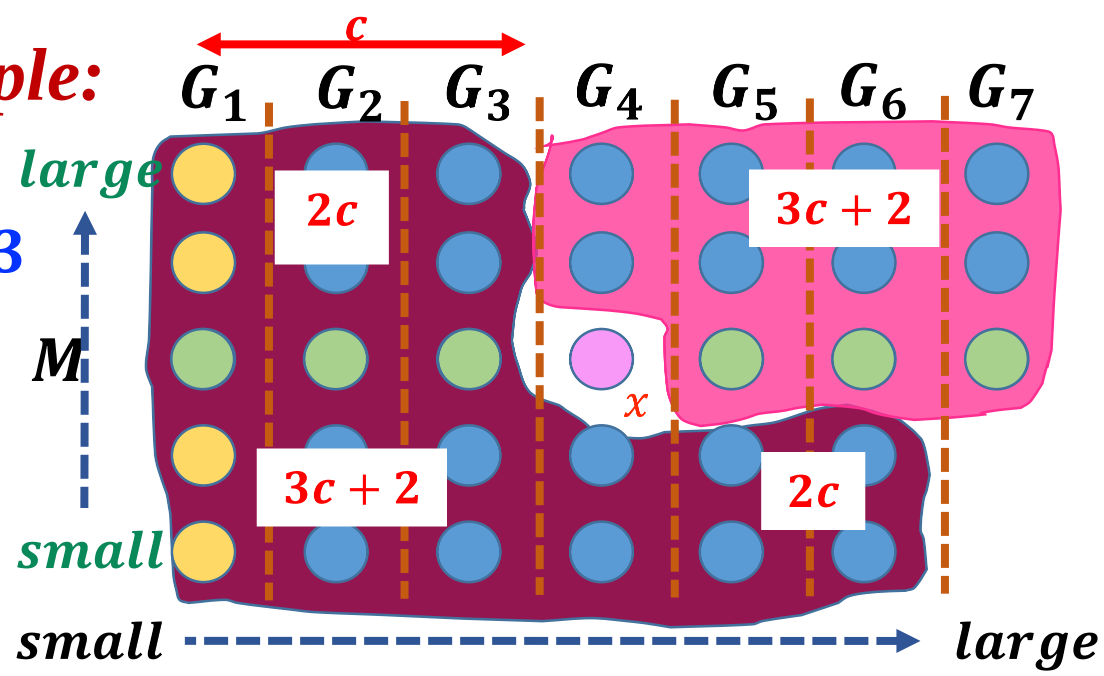
→ **Worst-case: subproblem as large as possible**

Selection Problem

✓ Selection Problem --*Divide&Conquer: analyze*

Let $T(n)$ is the time of n-element ➔ **$T(7n/10)$ be the time of subproblem**

Example:



$Step_1$: divide into $\lceil n/5 \rceil$ groups $\Theta(1)$

$Step_2$: recursively sele $T(\lceil n/5 \rceil)$ m of each group $O(n)$ $\lceil 5/2 \rceil = 3$

$Step_3$: find x: the median of M ➔ get 4 fields

$Step_4$: compare with position of x vs. k

➔ $n = 5 \times \#totalColumn = 5($ $T(7n/10)$

$\#worst = 7c + 2 = \frac{7n}{10} - \frac{3}{2}$

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

➔ **Worst-case: subproblem as large as possible**

Selection Problem

✓ Selection Problem --*Divide&Conquer: analyze*

Let $T(n)$ is the time of n-element

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

Recursion tree:

$T(n)$ → cn → $T(n/5)$, $T(7n/10)$

costs

cn → $cn/5$, $c7n/10$ ⇢ cn

$cn/5$ → $cn/25$, $c7n/50$; $c7n/10$ → $c7n/50$, $c49n/100$ ⇢ $0.9cn$

$cn/25$, $c7n/50$, $c7n/50$, $c49n/100$ ⇢ $0.81cn$

… …

$\Theta(1)$ $\Theta(1)$

$$T(n) \le cn\left(1 + 0.9 + 0.9^2 + \cdots\right)$$
$$= O(n)$$

Q: *What is the running time of **Select** if the elements are divided into **groups of 7** or **groups of 3**?*

Selection Problem

Conclusion:

1. **Main idea:**
   ① **Divide-and-Conquer**
   ② **Recurrences**
2. **Running time**
   ① **Minimum or maximum:** $T(n) = n - 1$
   ② **Minimum and maximum :** $T(n) = O(3\lfloor n/2 \rfloor)$
   ③ **Randomized Selection:** $expected\ in\ Linear\ time\ T(n) = O(n)$
   ④ **Select:** $worst - case\ in\ Linear\ time\ T(n) = O(n)$
3. **Randomized Selection vs. Randomized Quicksort**
   ① **Key** : choose pivot randomly
   ② **Subproblem**: one vs. two

Selection Problem

Exercise : *suppose we use* **RandomSelect** *to select the minimum element of the array* $A = [3, 2, 9, 0, 7, 5, 4, 8, 6, 1]$. *Describe a sequence of partitions that results in a worst-case performance of* **RandomSelect.**

Exercise : *In the algorithm* **Selection,** *the input elements are divided into groups of 5, will it work in linear time too if they are divided into groups of 7? And argue that why* **Selection** *does not run in linear time if groups of 3?*