Data Selection

COMPSCI 220: WEEK 8.4

Instructor: Meng-Fen Chiang

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

Slides adapted from Mark Wilson, Georgy Gimel'farb, Simone Linz and Tanya Gvozdeva

Introduction

- Data selection is closely related to sorting.
- Instead of rearranges the input list, data selection aims to find the k-th smallest item.
- This is also called the item of **rank** k or the k-th order statistic.

Data Selection Task

- Previously, we have sorted a list $a[0..n-1]$.
- Data selection task: we only need to know the k-th largest/smallest element in the sorted list.
- For example, find the k-th largest element in

10	13	2	12	5	9	1	8	3	11

- Question: Do we need to sort the list to do it?
  - **Special case $\boldsymbol{k}$=1 or $\boldsymbol{k}$=$\boldsymbol{n}$**: If we need the largest/smallest element, then we can just use the algorithm of finding min/max element of list, without sorting. It will take $\Theta(n)$ time, which is better then sorting the whole list.
  - **More general cases**: Consider k as an input, which we don't know beforehand.

Quickselect

- Let's use the quicksort idea to find the k-th smallest element! We are given list $a[0..n-1]$ and $0 \leq k \leq n-1$

  - If the size of the list is 1, return the only element in it.
  - Otherwise
    - Choose a pivot. Partition the remaining items into two disjoint sublists: reorder the list by placing all items $\geq$ the pivot to follow it, and all elements $\leq$ the pivot to precede it.
    - If the new position q of the pivot equals to $k-1$ then return $a[k-1]$.
    - If $q > k-1$ then recursively find the k-th smallest on the left sublist.
    - If $q < k-1$ then recursively find the $(k-q-1)$-th smallest on the right sublist

Quickselect | Example

$k = 4$

pivot partitioning

10	**13**	**2**	**12**	**5**	**9**	**1**	**8**	**3**	**11**
0	1	2	3	4	5	6	7	8	9

Quickselect | Example

$k = 4$

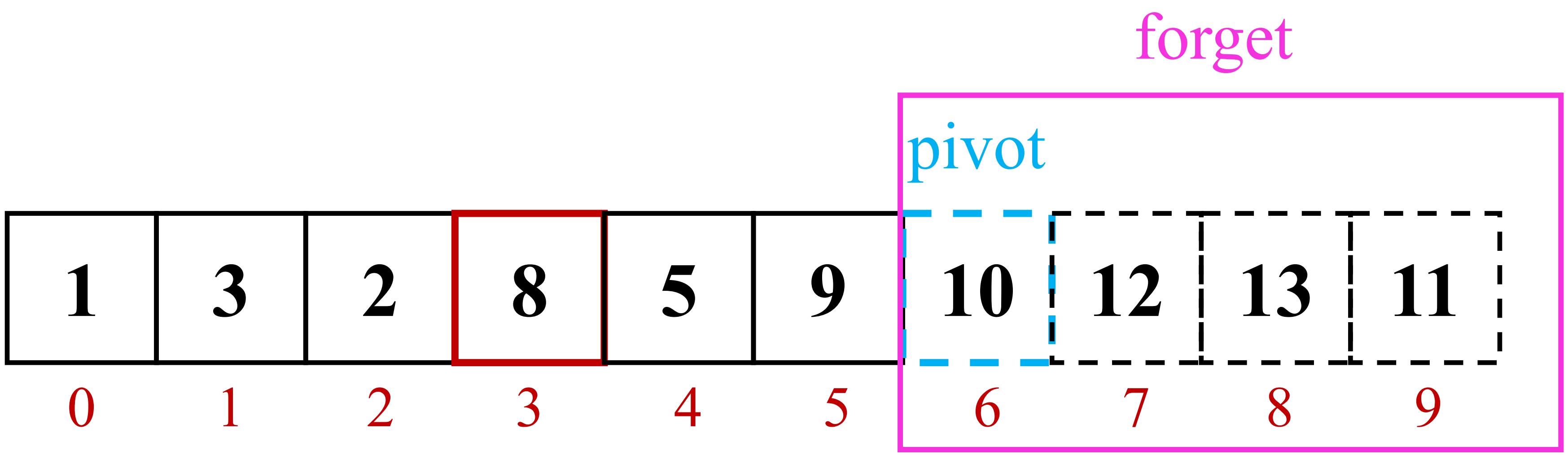
pivot

1	**3**	**2**	**8**	**5**	**9**	**10**	**12**	**13**	**11**
0	1	2	3	4	5	6	7	8	9

$k - 1 < 6$

Quickselect | Example

$k = 4$



Quickselect | Example

$k = 4$

pivot partitioning

1	**3**	**2**	**8**	**5**	**9**				
0	1	2	3	4	5	6	7	8	9

$k - 1 > 0$

Quickselect | Example

$k = 4$

forget

pivot

1	3	2	8	5	9				
0	1	2	3	4	5	6	7	8	9

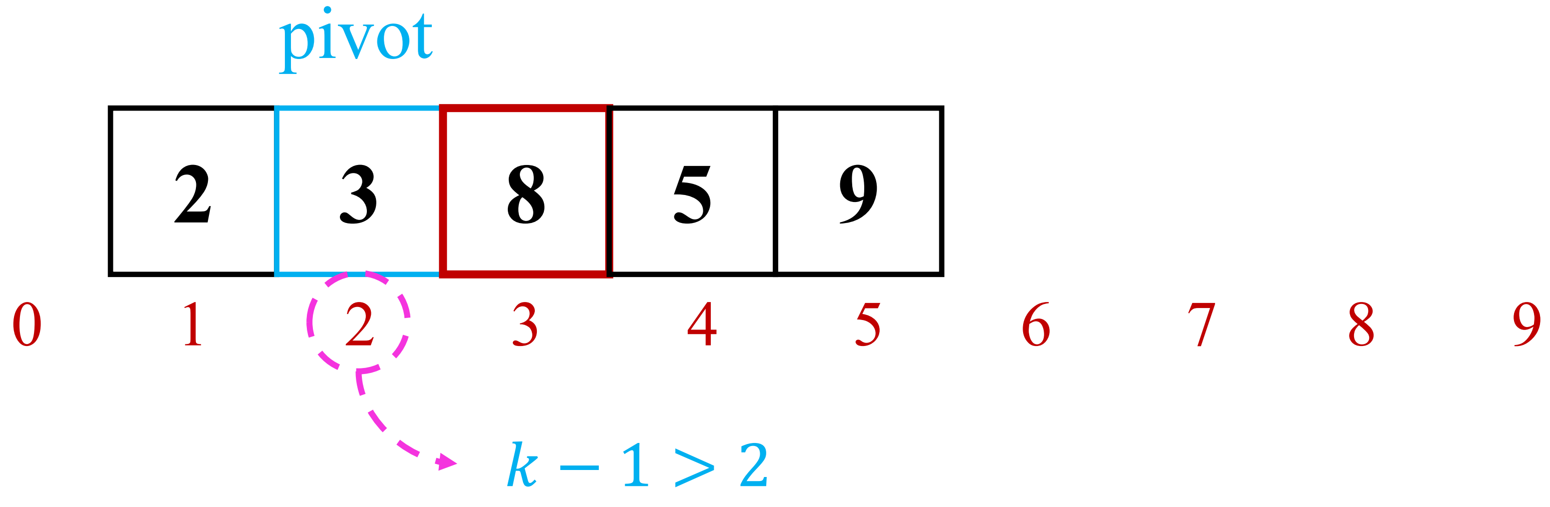
Quickselect | Example

$k = 4$

	pivot	partitioning							
	3	2	8	5	9				
0	1	2	3	4	5	6	7	8	9

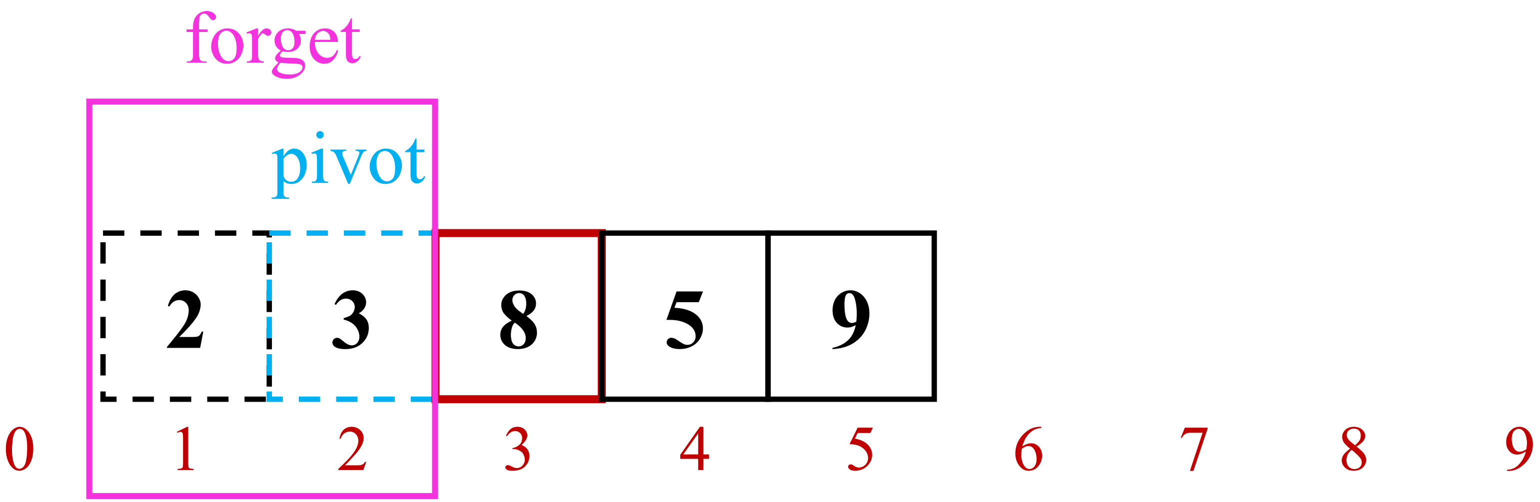
Quickselect | Example

$k = 4$



Quickselect | Example

$k = 4$



Quickselect | Example

$k = 4$

partitioning

pivot

			8	**5**	**9**				
0	1	2	3	4	5	6	7	8	9

Quickselect | Example

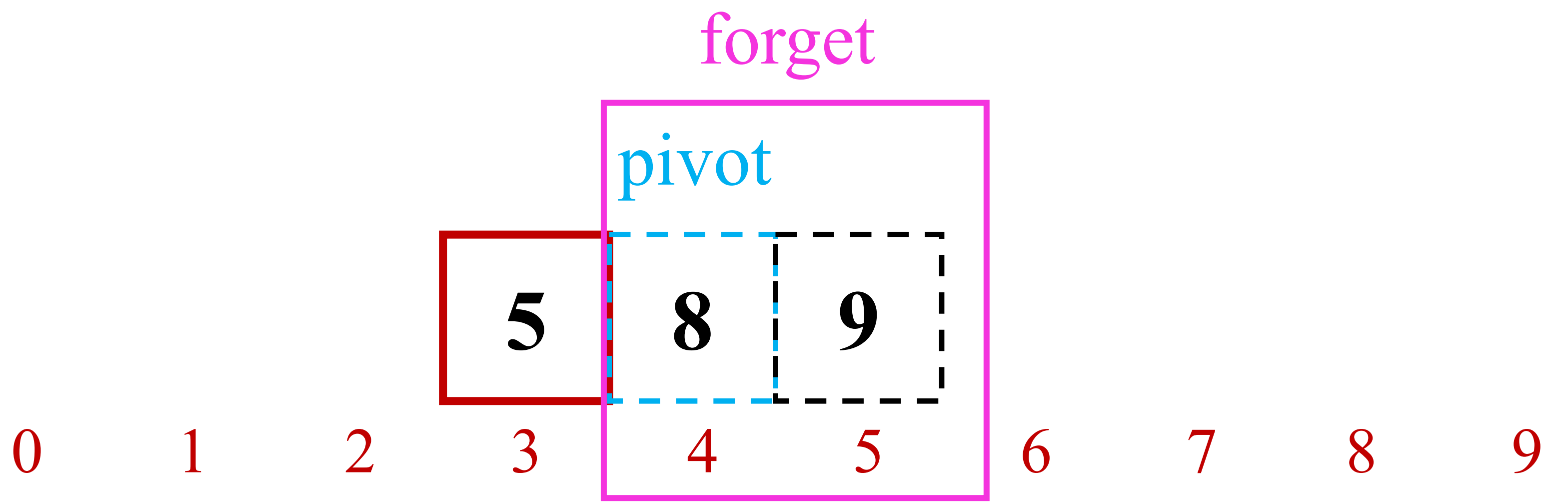
$k = 4$

				pivot					
			5	**8**	**9**				
0	1	2	3	4	5	6	7	8	9

$k - 1 < 4$

Quickselect | Example

$k = 4$



Quickselect | Example

$k = 4$

			$\boxed{\mathbf{5}}$						
0	1	2	3	4	5	6	7	8	9

Algorithm 1 Quickselect.

Require: $0 \le i \le j \le n-1, 1 \le k \le j-i+1$

function QUICKSELECT(list $a[0..n-1]$, int i, int j, int k)

  if $i < j$ **then**

    $p \leftarrow$ a[i] ▷ pivot element

    $q \leftarrow PARTITION(a, i, j, p)$ ▷ put p in correct position

    if $q - i = k - 1$ **then**

      return $a[q]$

    else if $q - i > k - 1$ **then**

      $QUICKSELECT(a, i, q-1, k)$ ▷ look in left half

    else

      $QUICKSELECT(a, q+1, j, k-q+i-1)$ ▷ look in right half

$\boldsymbol{a[0]}$	…	$\boldsymbol{a[i-1]}$	$\boldsymbol{a[i]}$	…	$\boldsymbol{a[j]}$	…	$\boldsymbol{a[n-1]}$

Quickselect works here

Time Complexity Analysis

- Claim: The average-case running time is $\Theta(n)$.

- Proof: The pivot can be at one of the n positions with equal probability for a list of size n after partitioning. Let's say the pivot is at position p.

$$[a_0, a_1, a_2, \dots, a_p, \dots, a_{n-1}]$$

- Then, we have p elements on the left sublist, and $n - p - 1$ elements on the right one.
- Let $T(n)$ be the average-case running time function for input of size n
- For each position p, the element we want to find could be one of the following three cases:
  - Case 1: The element is at position p, just return without any operation.
  - Case 2: The element is in the left sublist, we need $T(p)$ operations to find it.
  - Case 3: The element is in the right sublist. we need $T(n - p - 1)$ operations to find it.
  - For simplicity, we can merge case 2 to either case 1 or case 3, then on average, we need $(T(p) + T(n - p - 1))/2$ time.

$$[\overbrace{a_0, a_1, a_2, \ldots}^{T(p)}, a_p, \overbrace{\ldots, a_{n-1}}^{T(n-p-1)}]$$

Pivot is at position p:

Assume each event is equally likely to occur.

Possibility 1: $p = 0$	$(1/n)$	
Possibility 2: $p = 1$	$(1/n)$	
Possibility 3: $p = 2$	$(1/n)$	
…	…	
Possibility n: $p = n - 1$	$(1/n)$	

Case 1: quickselect the left sublist

Case 2: r-th order statistics is at p

Case 3: quickselect the right sublist

At any index p, the average cost is $\frac{\text{Case 1+Case3}}{2} = \frac{T(p)+T(n-p-1)}{2}$

$$T(n) = \frac{1}{n}\left[\frac{T(0)+T(n-1)}{2} + \frac{T(1)+T(n-2)}{2} + \frac{T(2)+T(n-3)}{2} \ldots + \frac{T(n-1)+T(0)}{2}\right] + partition$$

$$T(n) = \frac{1}{2n}[T(0)+T(1)+T(2)+\cdots+T(n-1)+T(n-1)+T(n-2)+T(n-3)+\cdots+T(0)] + partition$$

$$[\overbrace{a_0, a_1, a_2, \ldots}^{T(p)}, a_p, \overbrace{\ldots, a_{n-1}}^{T(n-p-1)}]$$

$$T(n) = \frac{1}{2n}[T(0) + T(1) + T(2) + \cdots + T(n-1) + T(n-1) + T(n-2) + T(n-3) + \cdots + T(0)] + partition$$

$$T(n) = \frac{2}{2n}[T(0) + T(1) + T(2) + \cdots + T(n-1)\,] + partition$$

$$T(n) = \frac{1}{n}[T(0) + T(1) + T(2) + \cdots + T(n-1)\,] + cn \qquad (c > 0)$$

$$nT(n) = [T(0) + T(1) + T(2) + \cdots + T(n-1)\,] + cn^2 \qquad \text{Eq.(1)}$$

$$(n-1)T(n-1) = [T(0) + T(1) + T(2) + \cdots + T(n-2)\,] + c(n-1)^2 \qquad \text{Eq.(2)}$$

Eq.(1)-Eq.(2)

$$nT(n) - (n-1)T(n-1) = T(n-1) + c(n^2 - (n^2 - 2n + 1))$$

$$nT(n) = nT(n-1) + c(2n-1)$$

$$T(n) = T(n-1) + 2c - \frac{c}{n}$$

$$[\underbrace{a_0, a_1, a_2, \ldots}_{T(p)}, a_p, \underbrace{\ldots, a_{n-1}}_{T(n-p-1)}]$$

Top-down Telescoping

Summation

$$\left\{\begin{aligned}
T(n) &= T(n-1) + 2c - \frac{c}{n} \\
T(n-1) &= T(n-2) + 2c - \frac{c}{n-1} \\
T(n-2) &= T(n-3) + 2c - \frac{c}{n-2} \\
T(n-3) &= T(n-4) + 2c - \frac{c}{n-3} \\
&\cdots \\
T(1) &= T(0) + 2c - \frac{c}{1}
\end{aligned}\right.$$

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

$$T(n) = 2nc - cH_n \qquad \rhd H_n \in O(\log n)$$

$$T(n) \in \Theta(n)$$

Time Complexity Analysis (Contd.)

- The average-case running time is $\Theta(n)$.

- Proof: Given that the partitioning is linear, we can have the running time function as follows:

$$T(n) = \frac{1}{2n}\sum_{p=0}^{n-1} T(p) + T(n-p-1) + cn = \frac{2}{2n}\sum_{p=0}^{n-1} T(p) + cn$$

$$nT(n) = \sum_{p=0}^{n-1} T(p) + cn^2$$

$$(n-1)T(n-1) = \sum_{p=0}^{n-2} T(p) + c(n-1)^2 \Longrightarrow T(n) = T(n-1) + \frac{c(2n-1)}{n} = T(n-1) + 2c - \frac{c}{n}$$

By telescoping, we have $T(n) = 2cn - cH_n \in \Theta(n)$

SUMMARY

- Best case $\Theta(1)$: the pivot from the first iteration happens to be what we are looking for.
- Worst case $\Theta(n^2)$: the same as Quicksort. Very unbalanced left and right sublists.
- Average case $\Theta(n)$