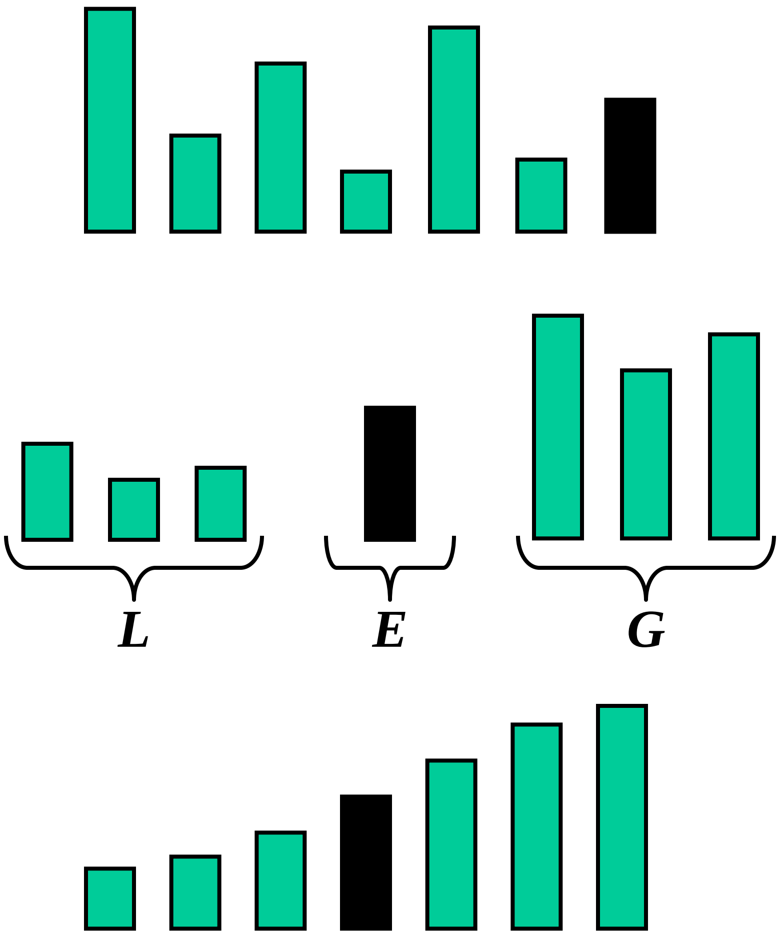
CSC 225

Algorithms and Data Structures I
Fall 2014
Rich Little

Quicksort as discussed in Textbook based on ADT Sequence

- Quick-sort is a randomized sorting algorithm based on the divide-and-conquer paradigm:
  - Divide: pick a random element ***x*** (called pivot) and partition ***S*** into
    - ***L*** elements less than ***x***
    - ***E*** elements equal ***x***
    - ***G*** elements greater than ***x***
  - Recur: sort ***L*** and ***G***
  - Conquer: join ***L***, ***E*** and ***G***



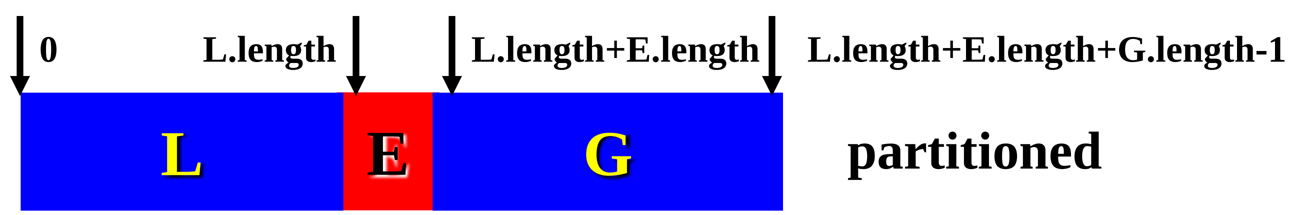
In-Place Quick-Sort

- How can Quick-sort be implemented to run in-place?
- Use an array
- Use the array to store the subarrays for all the recursive calls

Randomized QuickSelect

Input: Sequence S containing n elements, integer $k \leq n$
Output: k^{th} smallest element in sorted sequence S

if S.length() = 1 **then return** S
Let L, E, G be empty sequences
$p \leftarrow$ pickRandomPivot(S)
partition(L, E, G, S, p)
if $k \leq L$.length() **then return** QuickSelect(L, k)
else if $k \leq L$.length() + E.length() **then return** p
else return QuickSelect(G, $k - L$.length() $- E$.length())



Improve Quickselect to LinearSelect

Input: Sequence S containing n elements, integer $k \leq n$
Output: k^{th} smallest element in sorted sequence S

if S.length() = 1 **then return** S
Let L, E, G be empty sequences

$p \leftarrow$ pickCleverPivot(S)
partition(L, E, G, S, p)

if $k \leq$ L.length() **then return** Linearselect(L, k)
else if $k \leq L$.length() + E.length() **then return** p
else return Linearselect(G, k – L.length() – E.length())

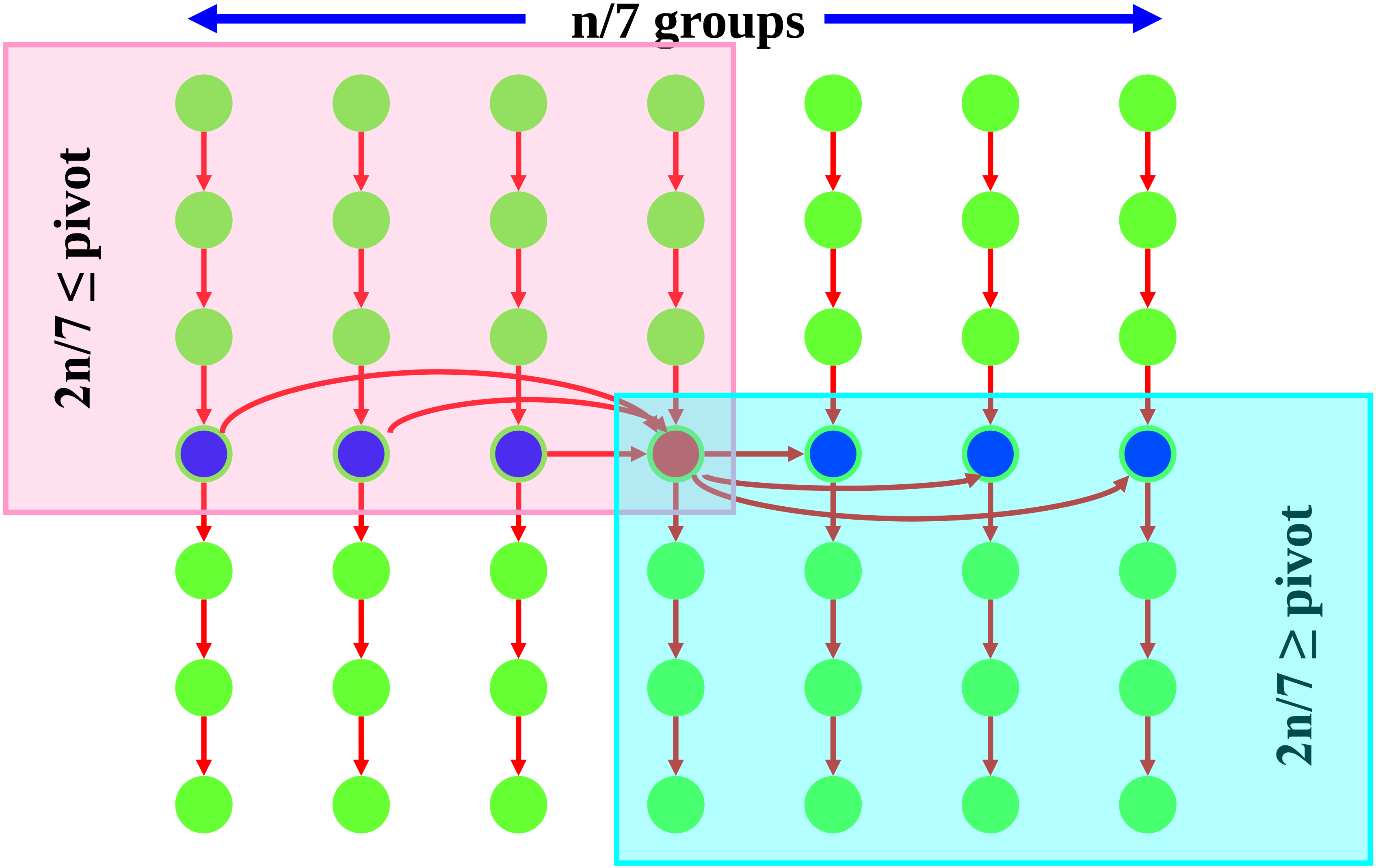
How to determine a good pivot?

- By clever pivot selection

- Divide S into groups of equal-sized groups of 5 or 7 elements—we will use groups of size 7
  - Thus, $n/7$ groups of size 7
  - $T(n) = O(1)$
- Sort each group of size 7 completely
  - Using 21 comparisons which is optimal for 7 elements
  - $T(n) = n/7 * 21 = 3n$
- Determine the median of each group
  - Pick the middle element of each group $T(n) = O(1)$
  - Gather all medians in a sequence or at the beginning of the array $T(n) = n$
- Use LinearSelect recursively to determine the median of medians
  - If the running time of LinearSelect is $T(n)$, then to compute the median of n/7 medians takes $T(n/7)$ time
  - The median of all the group medians is our clever new pivot

- Time complexity of clever pivot computation
  - $T(n) = 4n + T(n/7)$

Clever Pivot Selection



Clever Pivot Selection

- By selecting the pivot this way, we guarantee to split up $2n/7$ elements at partitioning
- Thus, we continue searching for the k^{th} element in $5n/7$ elements
- Thus, the conquer step takes T(5n/7) time

Time Complexity of LinearSelect

- Clever pivot selection $4n + T(n/7)$
- Partition $T(n) = n$
- Conquer recursive call $T(5n/7)$

- LinearSelect $T(n) = 5n + T(n/7) + T(5n/7)$

- **Theorem**
  - The worst-case $T(n)$ of LinearSelect is $O(n)$.
  - Blum, Floyd, Pratt, Rivest, Tarjan 1972

Solving Recurrence Equation by Guessing

$\boxed{\text{Proof.}}$

$$\text{Guess}\, T(n) = kn$$

$$T(n) = 5n + T(n/7) + T(5n/7)$$

$$kn = 5n + kn/7 + 5kn/7$$

$$7kn = 35n + kn + 5kn$$

$$7k = 35 + 6k$$

$$k = 35$$

$$T(n) = 35n \in O(n)$$

Worst-case Analysis

- **Theorem.**
  The worst-case $T(n)$ of Quicksort is $O(n^2)$.
- **Theorem.**
  The expected-case $T(n)$ of Randomized Quicksort is $O(n \log n)$.

- **Theorem.**
  The expected-case $T(n)$ of Randomized QuickSelect is $O(n)$.
- **Theorem.**
  The worst-case $T(n)$ of Randomized QuickSelect is $O(n^2)$.
- **Theorem.**
  The worst-case $T(n)$ of LinearSelect is $O(n)$.

Example LinearSelect

12 17 13 1 4	21 3 29 5 7	14 8 22 18 6	2 15 84 13 12	103 19 71 8 17

Divide *S* into *n*/5 groups of size 5

Then sort each group

12 17 13 1 4	21 3 29 5 7	14 8 22 18 6	2 15 84 13 12	103 19 71 8 17
1 4 12 13 17	3 5 7 21 29	6 8 14 18 22	2 12 13 15 84	8 17 19 71 103

Determine the Median of each Group and the Median of the Medians

12 17 13 1 4	21 3 29 5 7	14 8 22 18 6	2 15 84 13 12	103 19 71 8 17
1 4 **12** 13 17	3 5 **7** 21 29	6 8 **14** 18 22	2 12 13 15 84	8 17 **19** 71 103

12 17 13 1 4	21 3 29 5 7	14 8 22 18 6	2 15 84 13 12	103 19 71 8 17
1 4 **12** 13 17	3 5 **7** 21 29	6 8 **14** 18 22	2 12 13 15 84	8 17 **19** 71 103

Median of medians

Determine a lower bound on the size of *L*

12 17 13 1 4   21 3 29 5 7   14 8 22 18 6   2 15 84 13 12   103 19 71 8 17

1 4 **12** 13 17   3 5 **7** 21 29   6 8 **14** 18 22   2 12 13 15 84   8 17 **19** 71 103

Median of medians

L

1	3	2	6	8
4	5	12	8	17
12	7	13	14	19
13	21	15	18	71
17	84	29	22	103

R

n/5 elements are split up with each partition step

Running Time Analysis of LinearSelect with Groups of Size 5

$$T(n) = \begin{cases} b & \text{if } n \le 70 \\ 5n + T(n/5) + T(7n/10) & \text{if } n > 70 \end{cases}$$

- We prove T(n) is O(n)

Solving Recurrence Equation by Guessing

Proof.

$$\text{Guess}\, T(n) = kn$$

$$T(n) = 5n + T(n/5) + T(7n/10)$$

$$kn = 5n + kn/5 + 7kn/10$$

$$10kn = 50n + 2kn + 7kn$$

$$10k = 50 + 9k$$

$$k = 50$$

$$T(n) = 50n \in O(n)$$

Fundamental Result of Computer Science

- **Theorem.**
  Selecting the k^{th} smallest, largest or median element from a set of n elements takes $O(n)$ time in the worst-case.