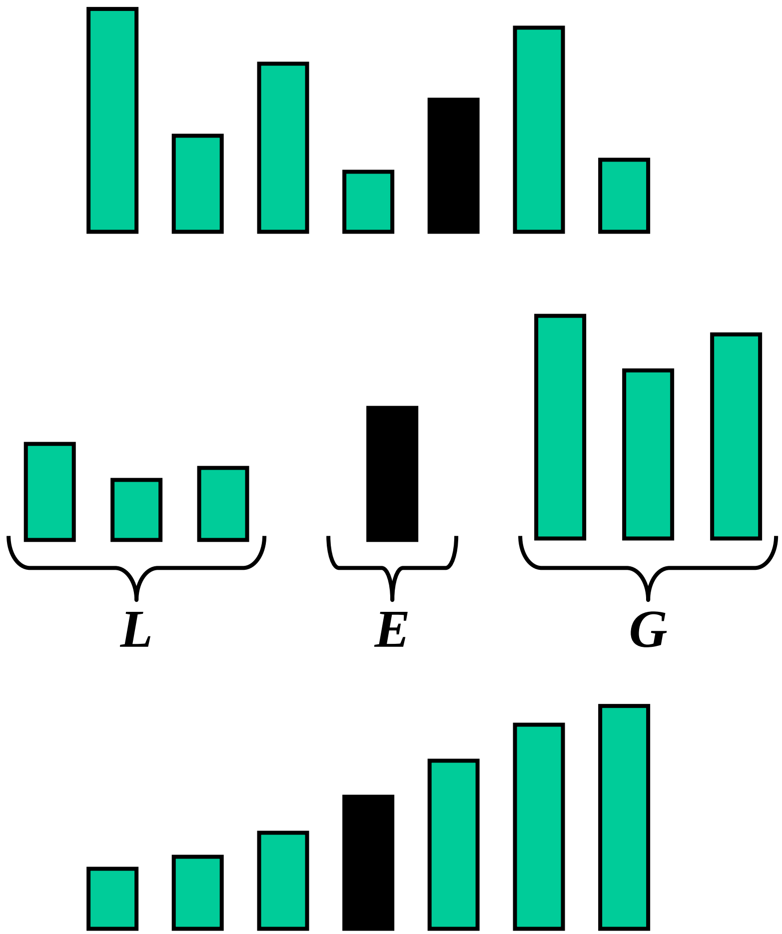
CSC 226

Algorithms and Data Structures: II

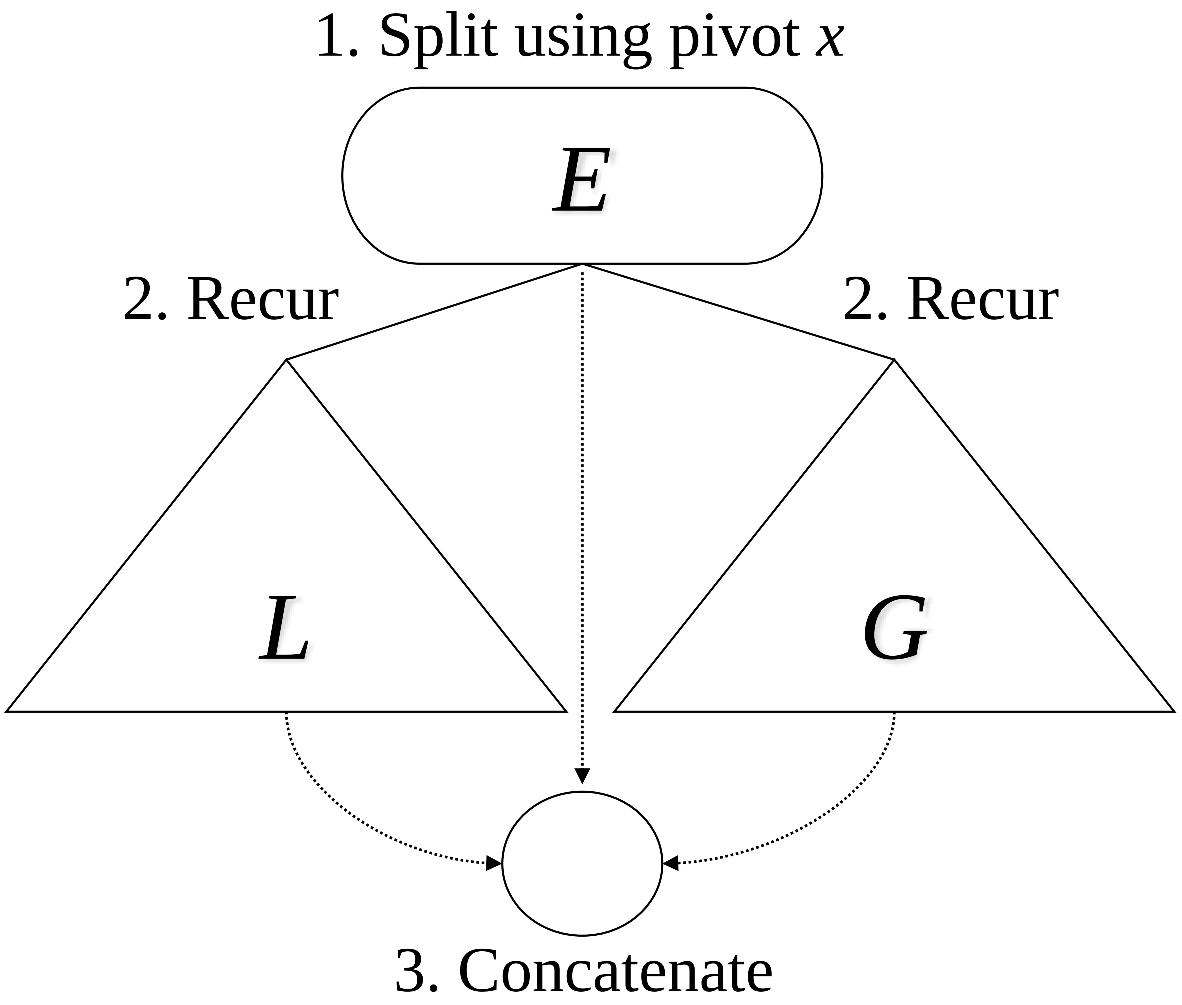
Rich Little
rlittle@uvic.ca

Quicksort based on ADT Sequence

- Quick-sort is a sorting algorithm based on the divide-and-conquer paradigm:
  - Divide: pick an element ***x*** in ***S*** (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***



Quicksort Algorithm



Quick-Sort Tree

- An execution of quick-sort is depicted by a binary tree
  - Each node represents a recursive call of quick-sort and stores
    - Unsorted sequence before the execution and its pivot
    - Sorted sequence at the end of the execution
  - The root is the initial call
  - The leaves are calls on subsequences of size 0 or 1

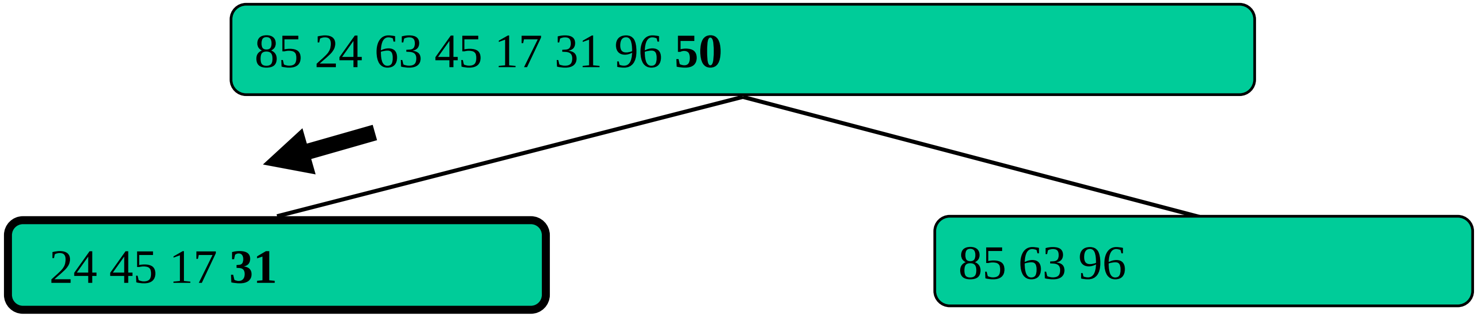
Execution Example

- Pivot selection

85 24 63 45 17 31 96 **50**

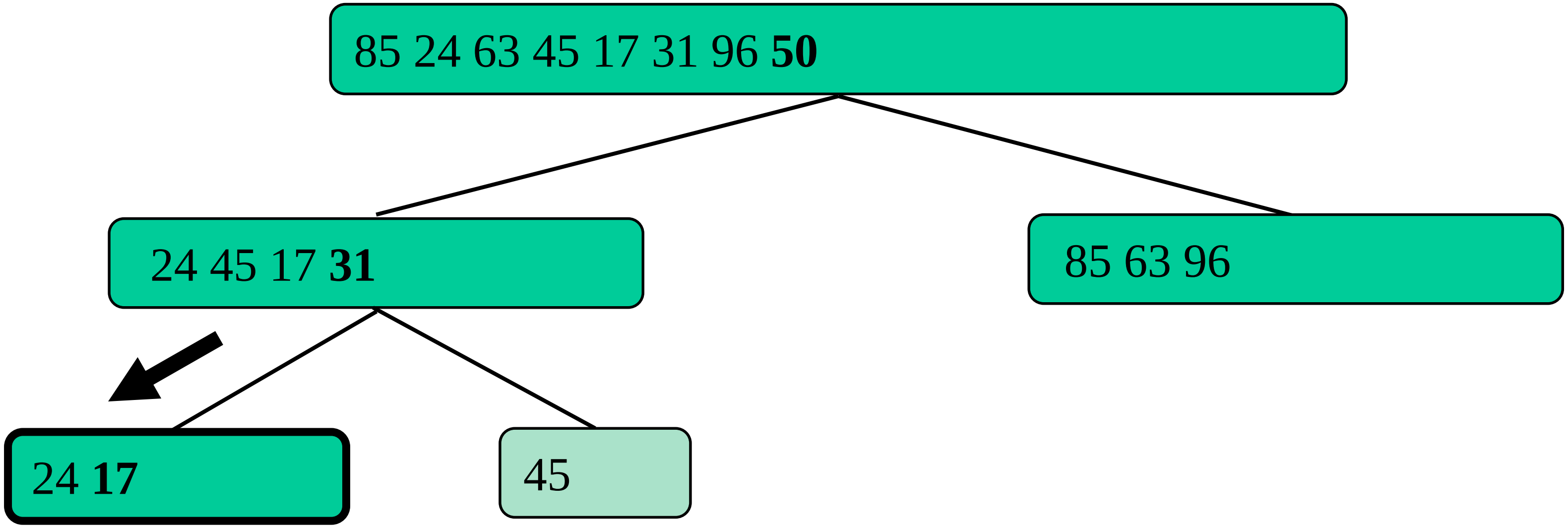
Execution Example (cont.)

- Partition, recursive call, pivot selection



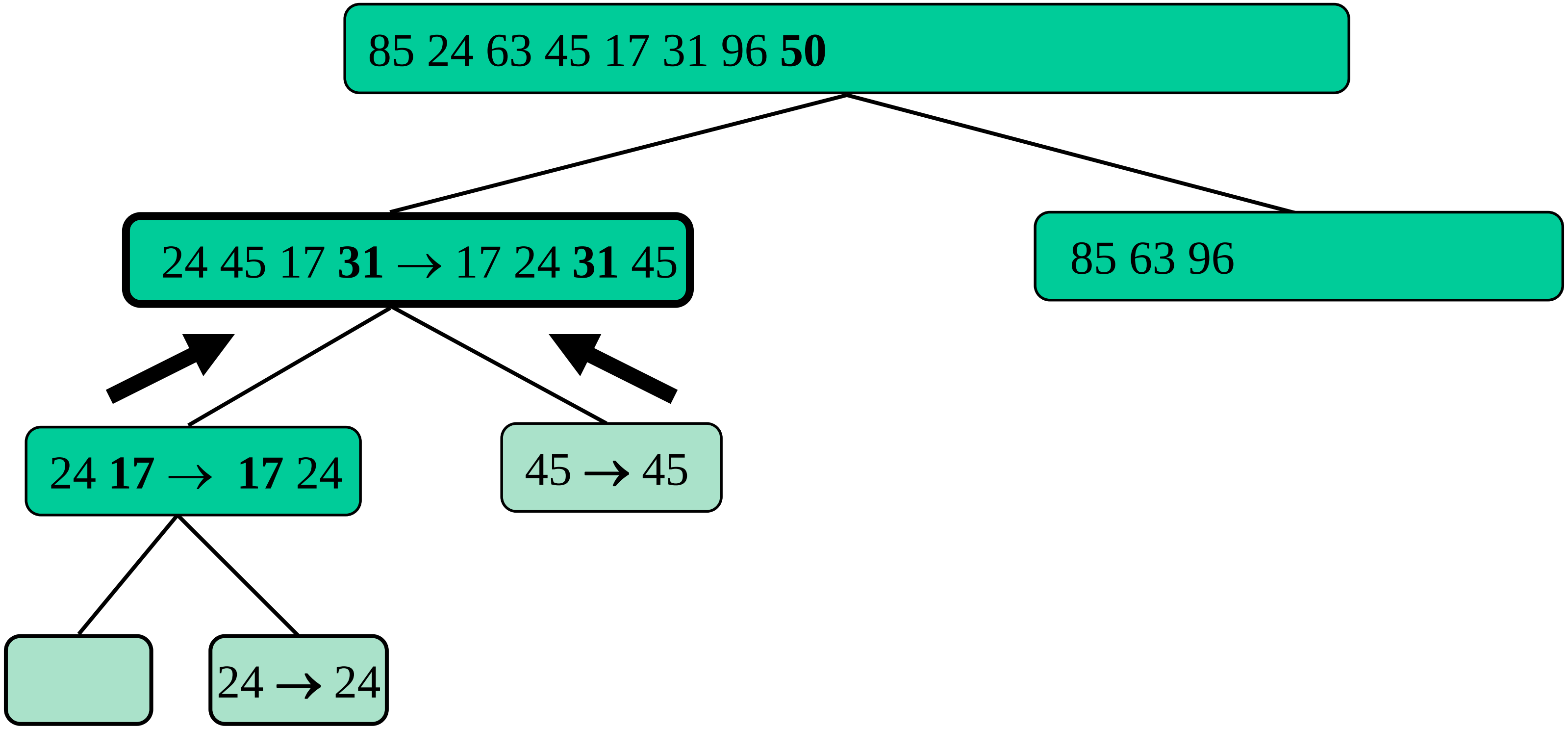
Execution Example (cont.)

- Partition, recursive call, pivot selection



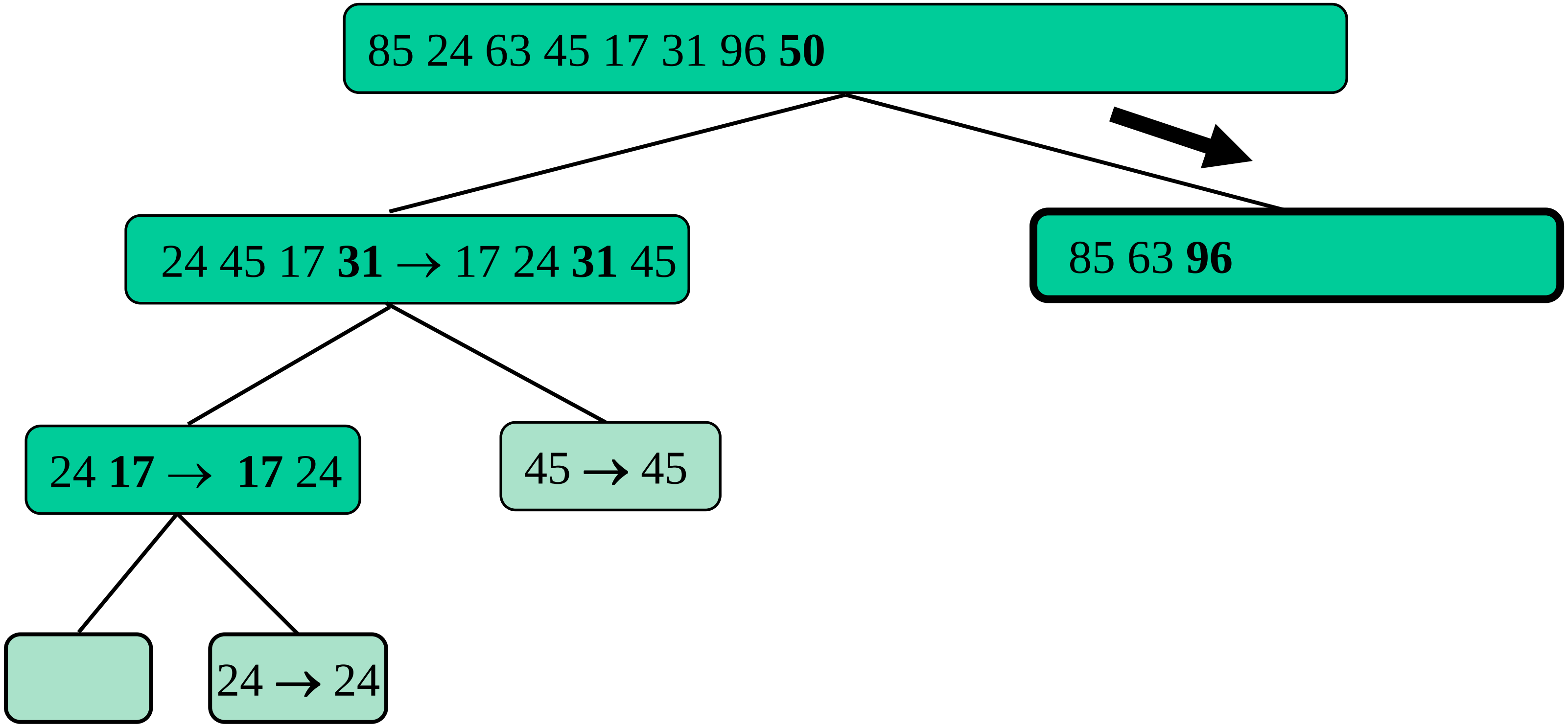
Execution Example (cont.)

- Recursive call, …, base case, join



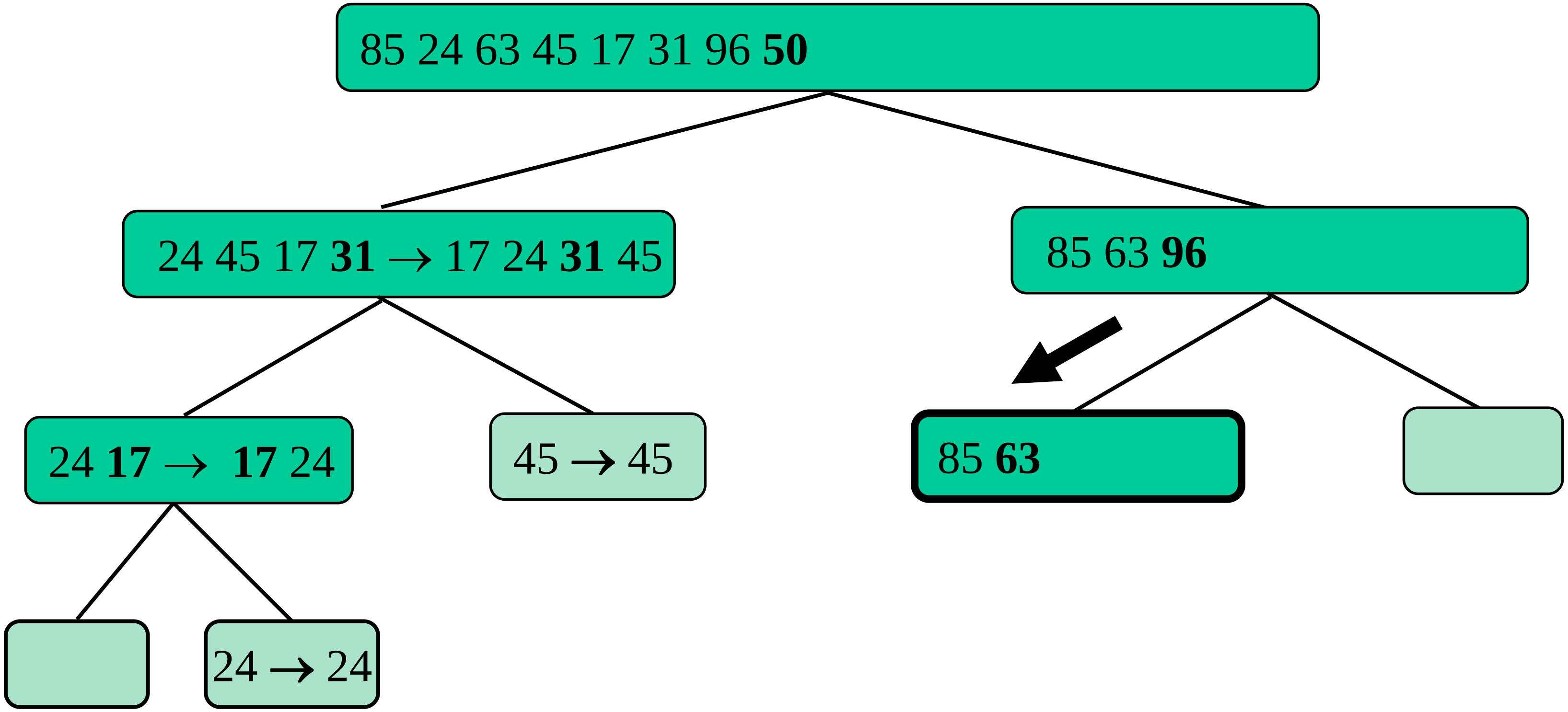
Execution Example (cont.)

- Recursive call, …, base case, join



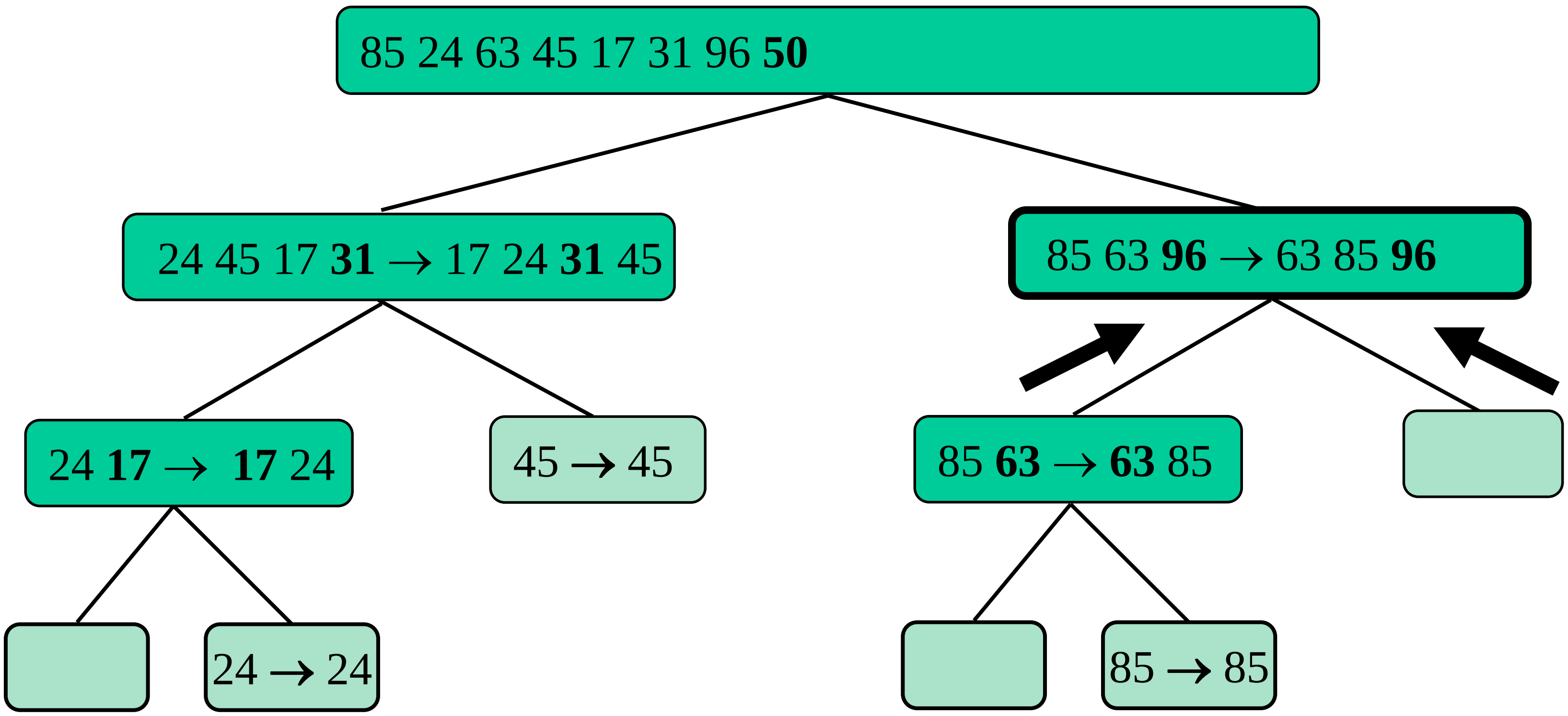
Execution Example (cont.)

- Recursive call, …, base case, join



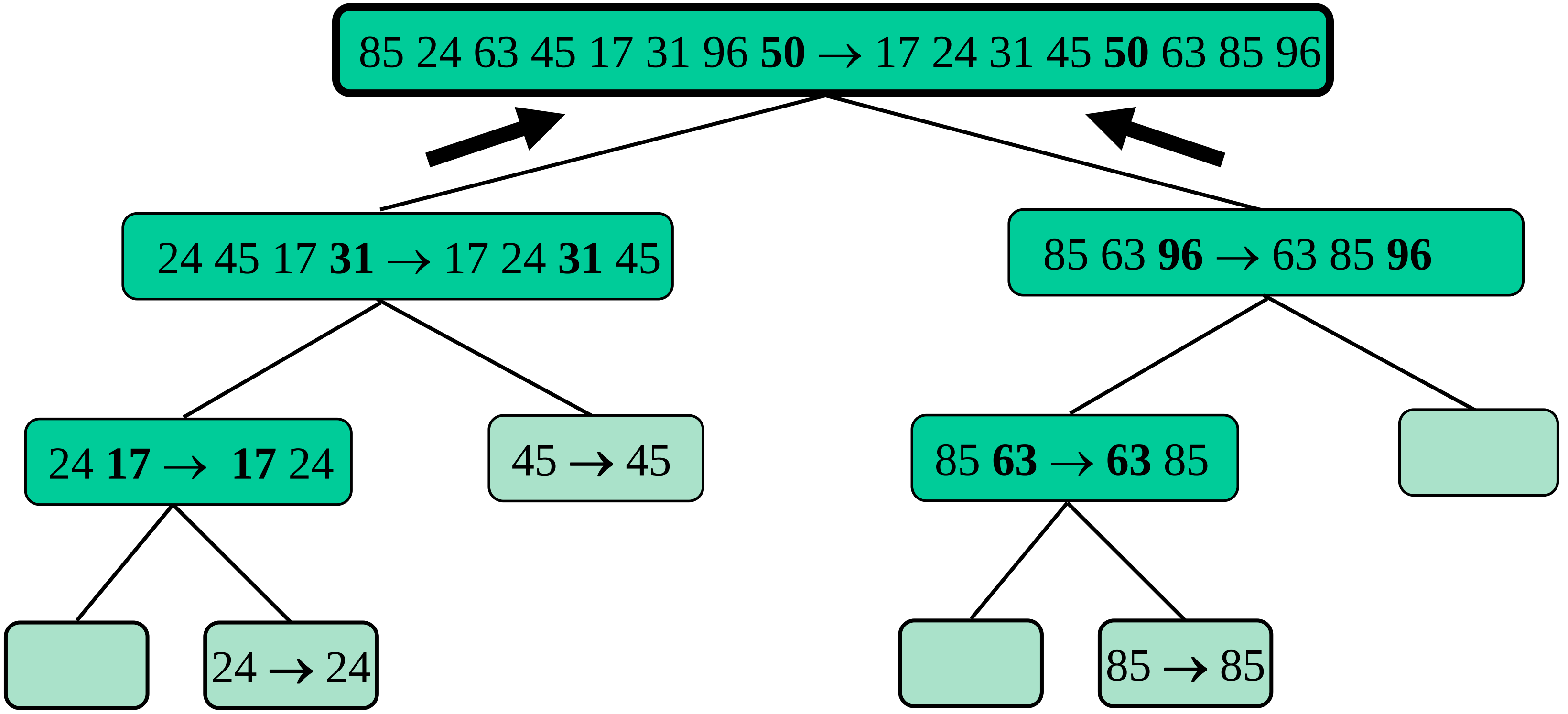
Execution Example (cont.)

- Recursive call, …, base case, join

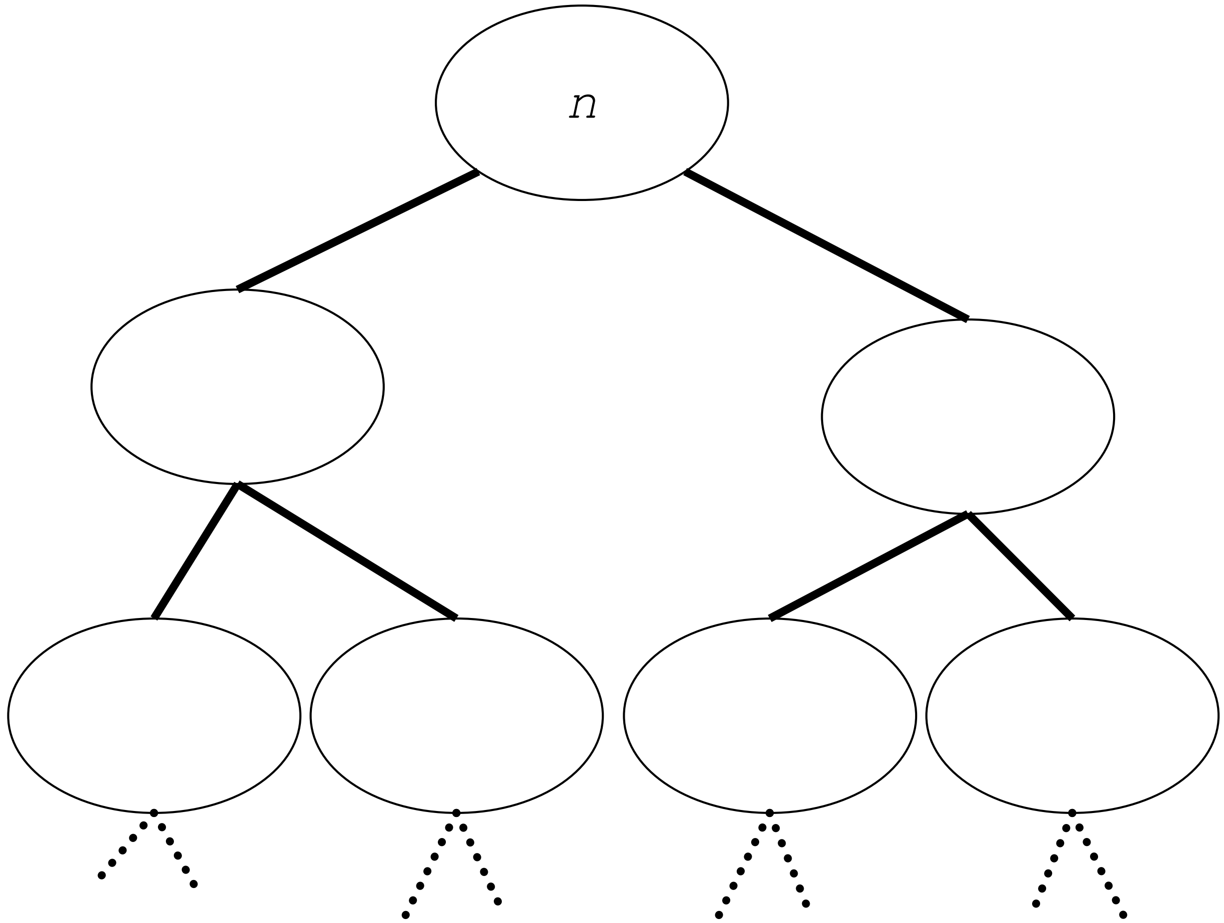


Execution Example (cont.)

- Join, join

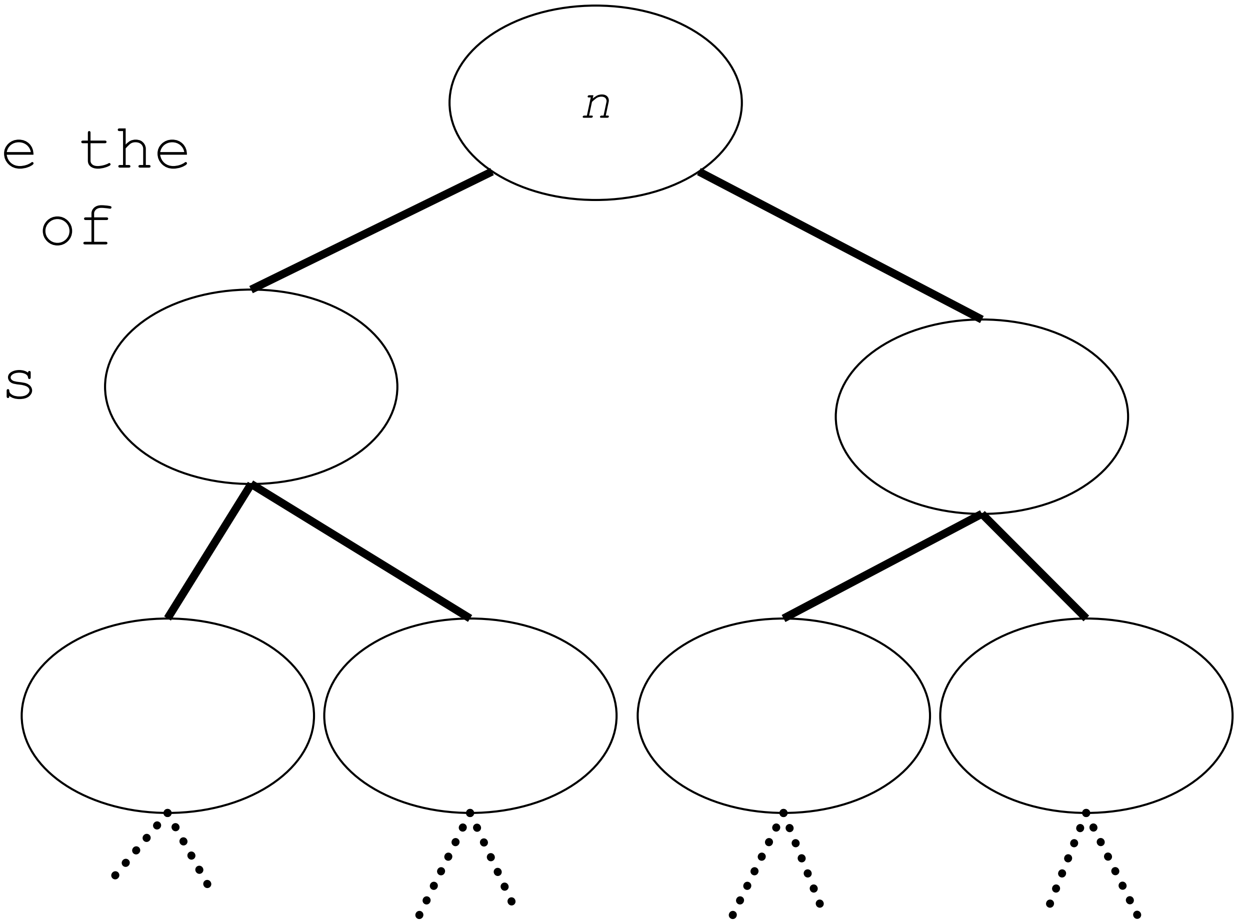


How long can a branch in the Quicksort tree be in the worst case?



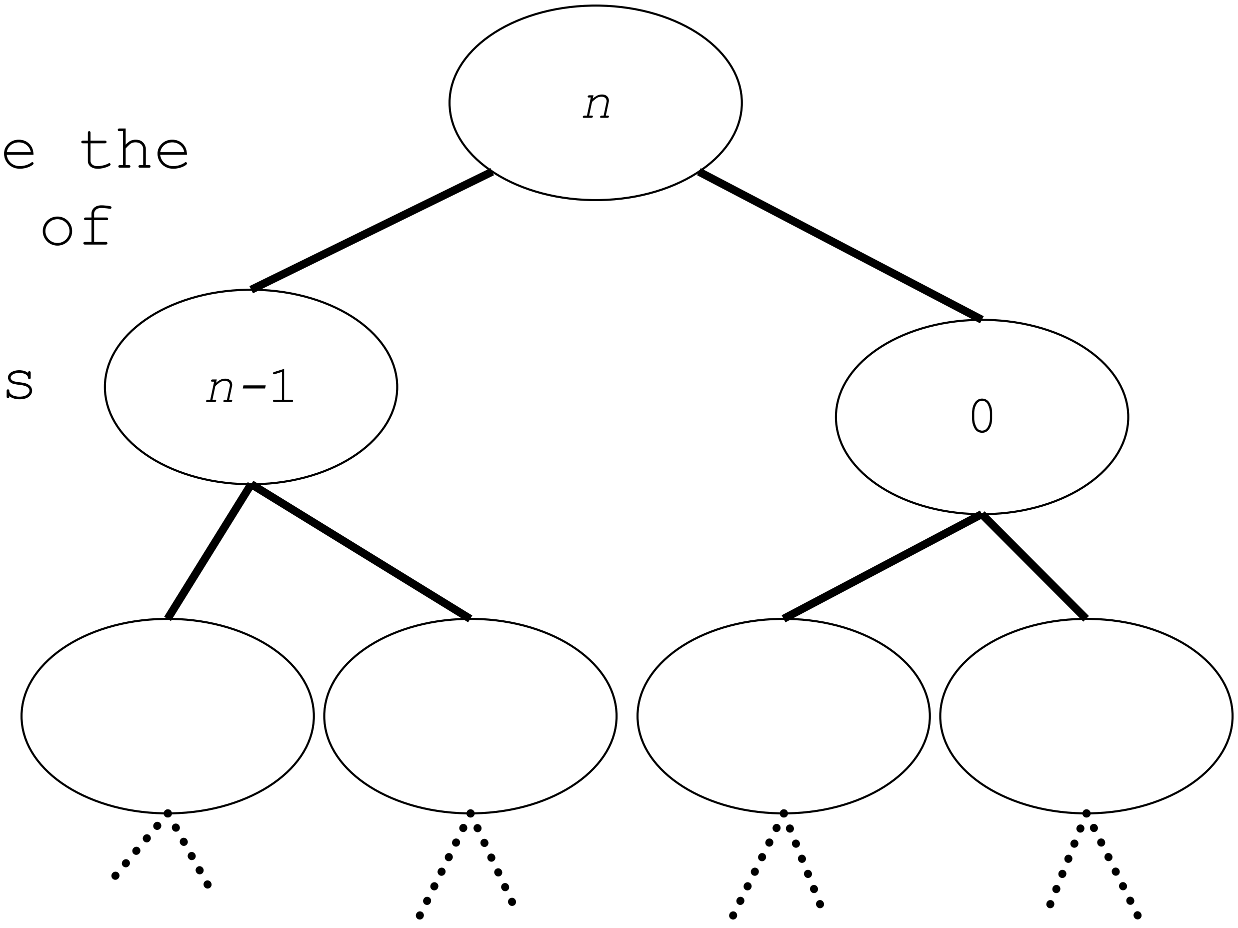
The pivot p and the length of sequences L and G

Let x be the largest of all elements



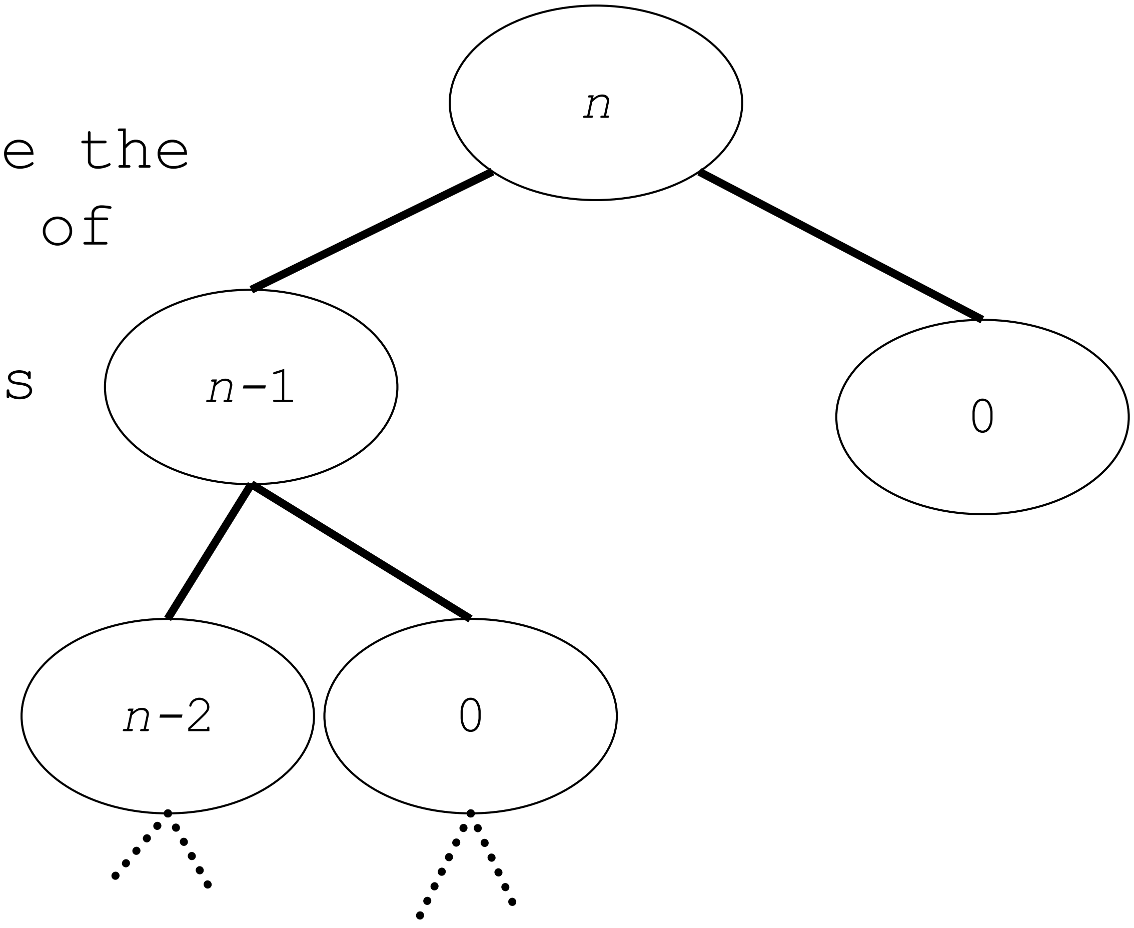
The pivot element and the length of sequences *L* and *G*

Let x be the largest of all elements



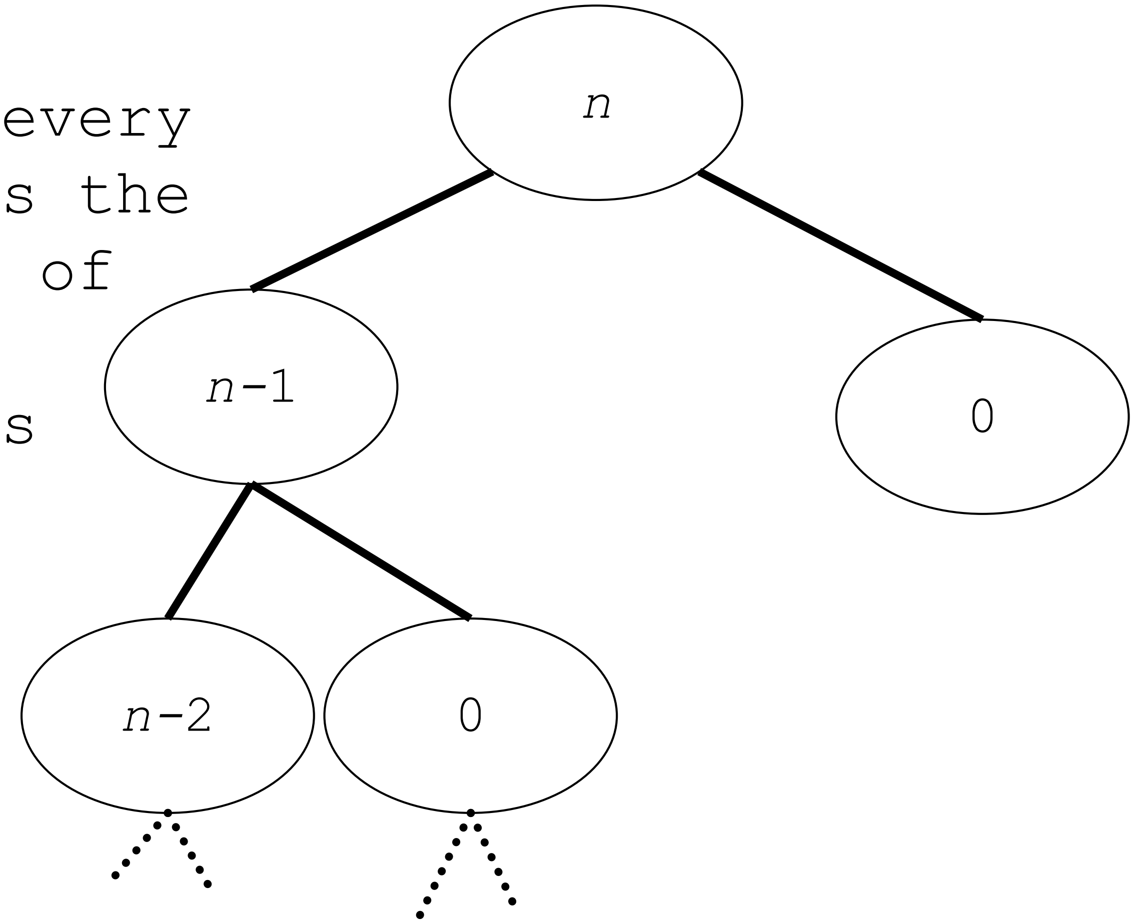
The pivot element and the length of sequences L and G

Let x be the largest of all elements

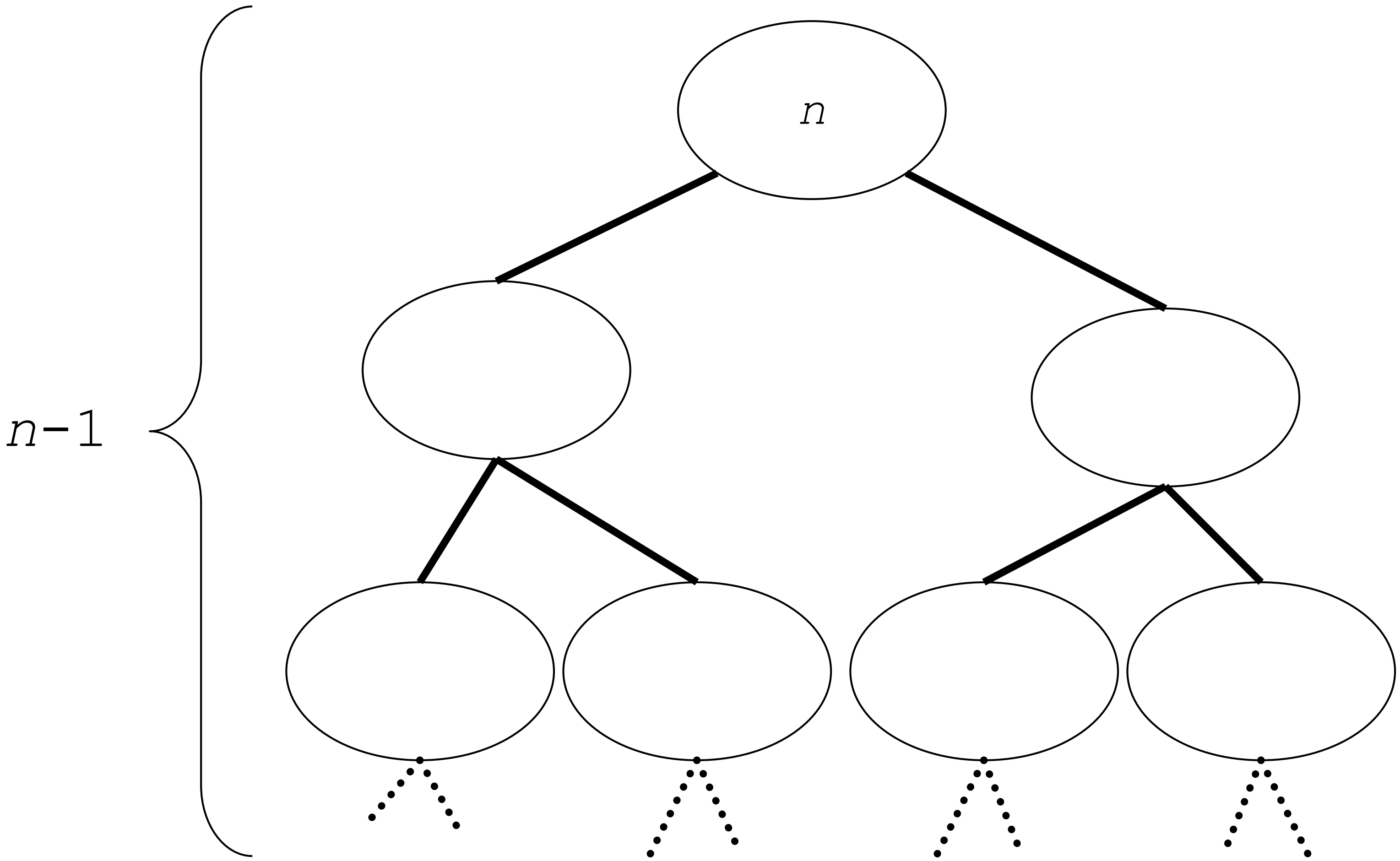


What sequences require the longest branch?

Assume every pivot is the largest of all the elements to sort



How long can be a branch in the quick-sort tree?



Algorithm quickSort(S)

```
if |S| > 1 then
  p ← pickPivot(S)
  partition(L,E,G,S,p)
  quickSort(L)
  quickSort(G)
  Concatenate(S,L,E,G)
end
```

What is the worst-case running time of Quicksort?

- To make life simple, assume $n - 1$ units of time to partition, 1 unit of time to pick pivot and 0 time to concatenate (inline).
- Then, the total time $T(n)$ is given by,

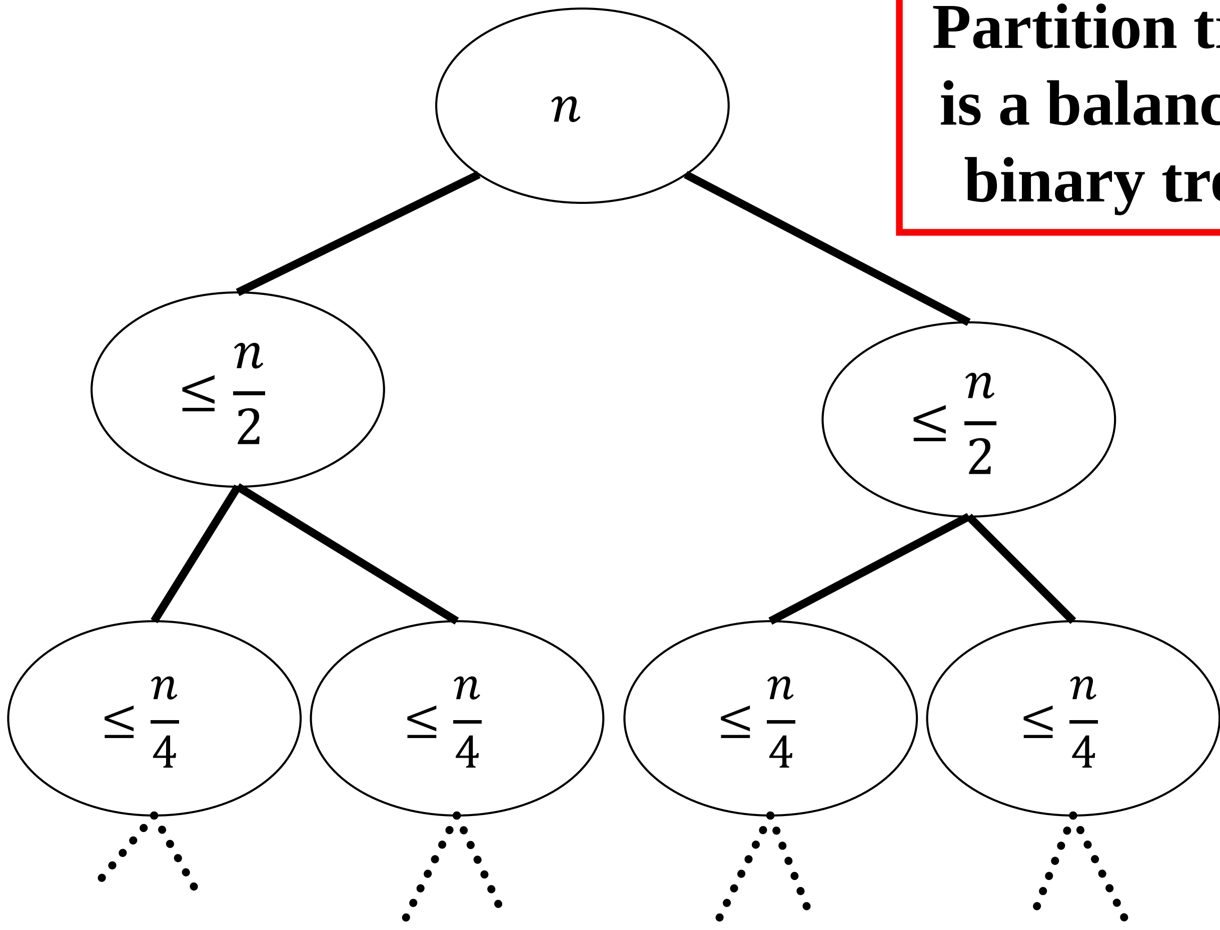
$$\boxed{T(n) = \begin{cases} 1, & \text{if } n = 1 \\ T(n-1) + n, & \text{otherwise} \end{cases}}$$

- By repeated substitution this is equal to

$$T(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \text{ which is } \Theta(\boldsymbol{n^2})$$

When is Quicksort fastest?

Partition tree is a balanced binary tree



What is the best-case running time of Quicksort?

- To make life simple, assume $n-1$ units of time to partition, 1 unit of time to pick pivot and 0 time to concatenate (inline).
- Then, the total time $T(n)$ is given by,

$$\boxed{T(n) \leq \begin{cases} 1, & \text{if } n = 1 \\ 2T\left(\frac{n}{2}\right) + n, & \text{otherwise} \end{cases}}$$

- By repeated substitution this is equal to

$$T(n) \leq n + n \log n \text{ which is } \Theta(\boldsymbol{n \log n})$$

Order Statistics or Selection

Problem

Given a sequence of n objects satisfying the total order property and an integer $k \le n$, determine the k^{th} smallest object.

Selecting the k^{th} Smallest Element

- Sort the objects and return the k^{th} from the left (i.e., k^{th} smallest element) $O(n \log n)$
- How to improve on $O(n \log n)$?
  - How much improvement is possible?
  - Best case and worst-case?
  - Modify a known sorting algorithm
  - Develop an algorithm from scratch

Modify Quicksort: quickSelect(S,k)

Input: Sequence S containing n elements, integer $k \leq n$

Output: k^{th} smallest element in sorted sequence S

if $n = 1$ **then**
  return S
Let L, E, G be empty sequences
$p \leftarrow$ pickPivot(S)
Partition(L, E, G, S, p)
if $k \leq |L|$ **then**
  return quickSelect(L, k)
else if $k \leq |L| + |E|$ **then**
  return p
else
  return quickSelect(G, $k - |L| - |E|$)

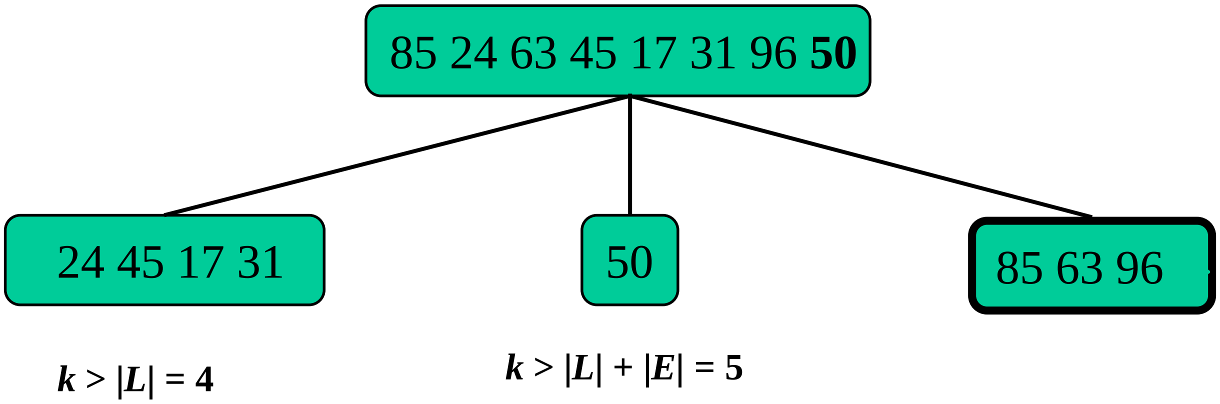
Execution Example

- quickSelect([85,24,63,45,17,31,96,50],*6*)

85 24 63 45 17 31 96 **50**

Execution Example (cont.)

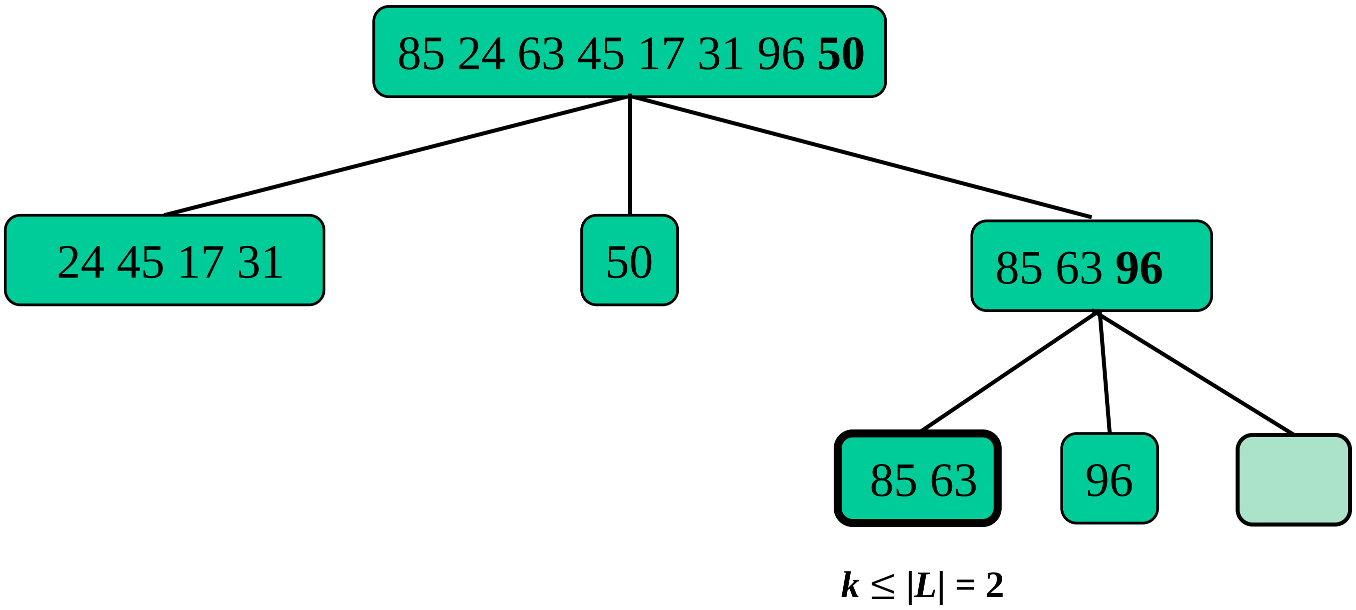
- Here $k = 6$



- Call quickSelect(G,k-$|L|$-$|E|$=1)

Execution Example (cont.)

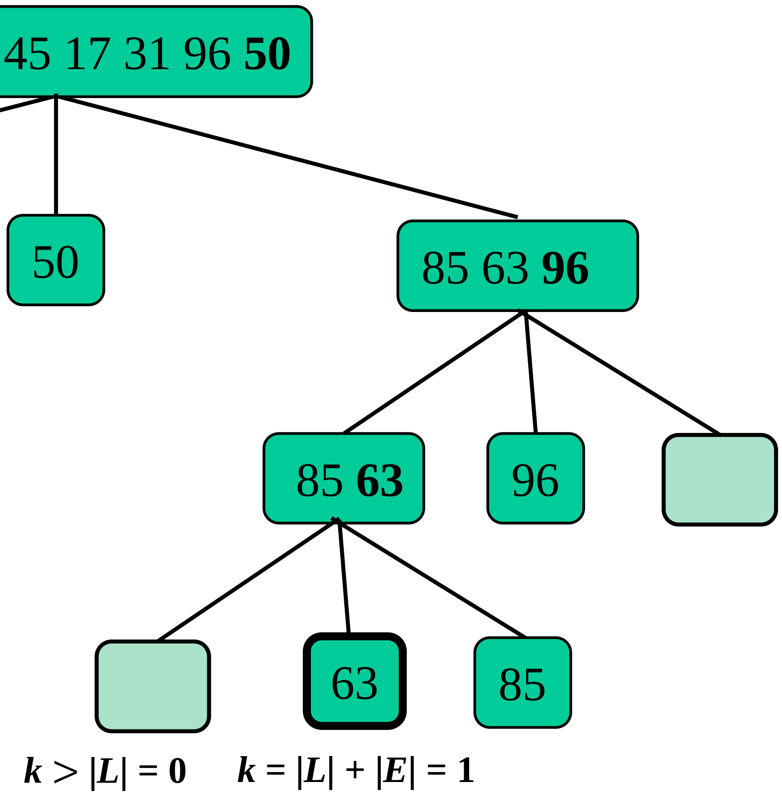
- quickSelect([85,63,96],1)



- Call quickSelect(L,1)

Execution Example (cont.)

- quickSelect([85,63],1)



- return 63

What is the worst-case running time of Quickselect?

- To make life simple, assume $n-1$ units of time to partition, and 1 unit of time to pick pivot.
- Then, the total time $T(n)$ is given by,

$$\boxed{T(n) = \begin{cases} 1, & \text{if } n = 1 \\ T(n-1) + n, & \text{otherwise} \end{cases}}$$

- By repeated substitution this is equal to

$$T(n) = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \text{ which is } \Theta(\boldsymbol{n^2})$$

What is the best-case running time of Quicksort?

- To make life simple, assume $n - 1$ units of time to partition and 1 unit of time to pick pivot.
- Then, the total time $T(n)$ is given by,

$$\boxed{T(n) \leq \begin{cases} 1, & \text{if } n = 1 \\ T\left(\frac{n}{2}\right) + n, & \text{otherwise} \end{cases}}$$

- By repeated substitution this is equal to

$$T(n) \leq \sum_{i=0}^{\log_2 n} n \left(\frac{1}{2}\right)^i \text{ which is } \Theta(\boldsymbol{n})\textbf{!}$$

CSC 226

Algorithms and Data Structures: II

Rich Little
rlittle@uvic.ca

Order Statistics or Selection

Problem

Given a sequence of n objects satisfying the total order property and an integer $k \leq n$, determine the k^{th} smallest object.

quickSelect(S,k)

Input: Sequence S containing n elements, integer $k \leq n$

Output: k^{th} smallest element in sorted sequence S

if $n = 1$ **then**
    return S
Let L, E, G be empty sequences
$p \leftarrow$ pickPivot(S)
Partition(L, E, G, S, p)
if $k \leq |L|$ **then**
    return quickSelect(L, k)
else if $k \leq |L| + |E|$ **then**
    return p
else
    return quickSelect(G, $k - |L| - |E|$)

linearSelect(S,k)

Input: Sequence S containing n elements, integer $k \leq n$

Output: k^{th} smallest element in sorted sequence S

if $n = 1$ **then**
  return S
Let L, E, G be empty sequences
$p \leftarrow$ pickCleverPivot(S)
partition(L, E, G, S, p)
if $k \leq |L|$ **then**
  return linearSelect(L, k)
else if $k \leq |L| + |E|$ **then**
  return p
else
  return linearSelect(G, $k - |L| - |E|$)

pickCleverPivot(S)

Input: Sequence S containing n elements

Output: The median of medians of subsets of size 7

Divide S into $g = \lceil n/7 \rceil$ subsequences, $S_1, \ldots, S_g$ of size 7
(one may be < 7)

for $i \leftarrow 1$ to g **do**
$x_i \leftarrow$ median of S_i

$p \leftarrow$ linearSelect($\{x_1, \ldots, x_g\}, \lceil g/2 \rceil$)

return p

How to determine a good pivot?

- Divide S into equal-sized groups of 5 or 7 elements—we use groups of size 7
  - Thus, $\lceil n/7 \rceil$ groups of size 7 (possibly one with less)
  - Takes $O(1)$ time (in-place)

- Sort each group of size 7 completely
  - Using 21 comparisons which is optimal for 7 elements
  - Takes $\lceil n/7 \rceil * 21 \approx \boldsymbol{3n}$ time

- Determine the median of each group
  - Pick the middle element of each group O(1)
  - Gather all medians in a sequence or at the beginning of the array - takes $\approx \boldsymbol{n}$ time

- Use linearSelect recursively to determine the *median of medians*
  - If the running time of LinearSelect is $T(n)$, then to compute the median of $\lceil n/7 \rceil$ medians takes roughly $\boldsymbol{T(n/7)}$ time
  - The median of all the group medians is our clever new pivot

- Time complexity of clever pivot computation (give or take constant values)
  - $\boldsymbol{4n + T(n/7)}$

linearSelect(S,k)

Input: Sequence S containing n elements, integer $k \leq n$

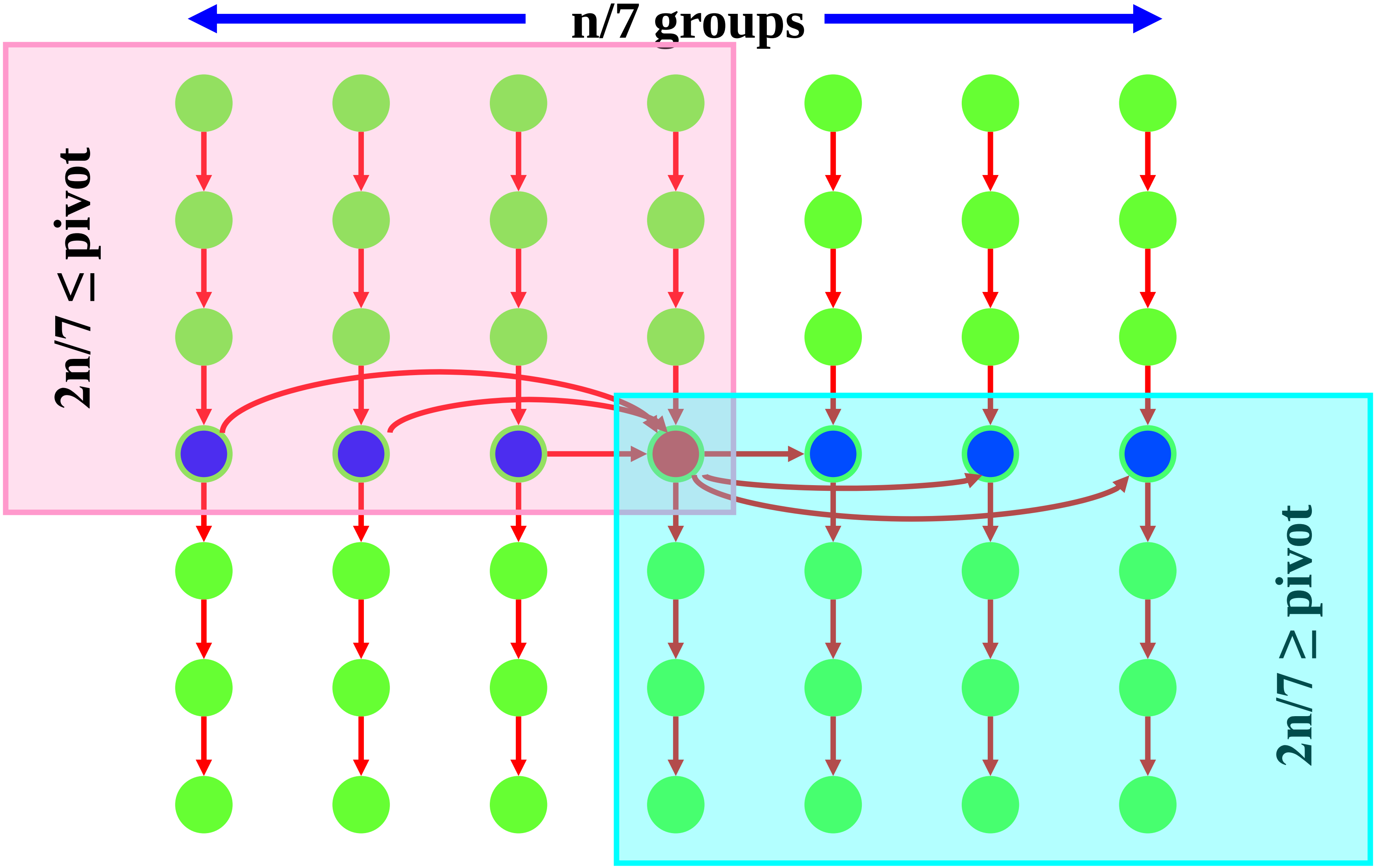
Output: k^{th} smallest element in sorted sequence S

if n = 1 **then**
    return S
Let L, E, G be empty sequences
$p \leftarrow$ pickCleverPivot(S)
partition(L, E, G, S, p)
if $k \leq |L|$ **then**
    return linearSelect(L, k)
else if $k \leq |L| + |E|$ **then**
    return p
else
    return linearSelect(G, $k - |L| - |E|$)

$\boldsymbol{4n + T(n/7)}$
$\boldsymbol{n}$

What is the cost of the recursive call?

Clever Pivot Selection



Clever Pivot Selection

- By selecting the pivot this way, we guarantee that $4 \times \left\lceil \frac{n/7}{2} \right\rceil$ elements are smaller than the pivot. (Similarly, larger than the pivot)
- Thus, in the worst case $2n/7$ elements at partitioning are in L and $5n/7$ are in G (or vice versa)
- Thus, we continue searching for the k^{th} element in $5n/7$ elements
- Thus, the conquer step takes $\boldsymbol{T(5n/7)}$ time

Time Complexity of LinearSelect

- Clever pivot selection $4n + T(n/7)$
- Partition 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

Proof.

Guess $T(n) \le cn$

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

$$\le 5n + cn/7 + 5cn/7$$

$$35n + cn + 5cn \le 7cn$$

$$35 \le c$$

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

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
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

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

Then sort each group

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

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

n/5 elements are split up with each partition step

Running Time Analysis of LinearSelect with Groups of Size 5

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

- We prove $\mathrm{T}(n)$ is $O(n)$

Solving Recurrence Equation by Guessing

Proof.

$$\text{Guess } T(n) \le cn$$

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

$$\le 5n + cn/5 + 7cn/10$$

$$50n + 2cn + 7cn \le 10cn$$

$$50 \le c$$

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

Fundamental Result of Computer Science

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