CSCI2100 Data Structures
Medians and Order Statistics

Irwin King

king@cse.cuhk.edu.hk
http://www.cse.cuhk.edu.hk/~king

Department of Computer Science & Engineering
The Chinese University of Hong Kong

Outline

- Order Statistics
- Overview of QuickSort
- Selection in Expected Linear Time
- Selection in Worst-Case Linear Time
- Analysis

Resources: https://www.cs.drexel.edu/~amd435/courses/cs260/lectures/L-9_2_Order_statistics_1P.pdf
www.cs.bu.edu/fac/gkollios/cs113/Slides/quicksort.ppt

What Are Order Statistics?

- The k-th order statistic is the k-th smallest element of an array.

3	4	13	14	23	27	41	54	65	75

8th order statistic

- The lower median is the $\lfloor \frac{n}{2} \rfloor$-th order statistic.
- The upper median is the $\lceil \frac{n}{2} \rceil$-th order statistic.
- If n is odd, lower and upper median are the same.

3	4	13	14	23	27	41	54	65	75

lower median upper median

What are Order Statistics?

- Selecting ith-ranked item from a collection.
  - First: $i = 1$
  - Last: $i = n$
  - Median(s): $i = \lfloor \frac{n}{2} \rfloor, \lceil \frac{n}{2} \rceil$

Order Statistics Overview

- Assume collection is unordered, otherwise trivial.

  find the ith order stat = A[i]

- Can sort first — $\Theta(n \log n)$, but can do better — $\Theta(n)$.
- We can find max and min in time (obvious).
- Can we find any order statistics in linear time? (not obvious!)

Order Statistics Overview

- How can we modify QuickSort to obtain expected-case $\Theta(n)$?
- Pivot, partition, but recur only on one set of data. No join.

QuickSort

- Given an array of *n* elements (e.g., integers):
  - If array only contains one element, return
  - Else
    - pick one element to use as pivot.
    - Partition elements into two sub-arrays:
      - Elements less than or equal to pivot
      - Elements greater than pivot
    - Quicksort two sub-arrays
    - Return results

Example

- We are given array of *n* integers to sort:

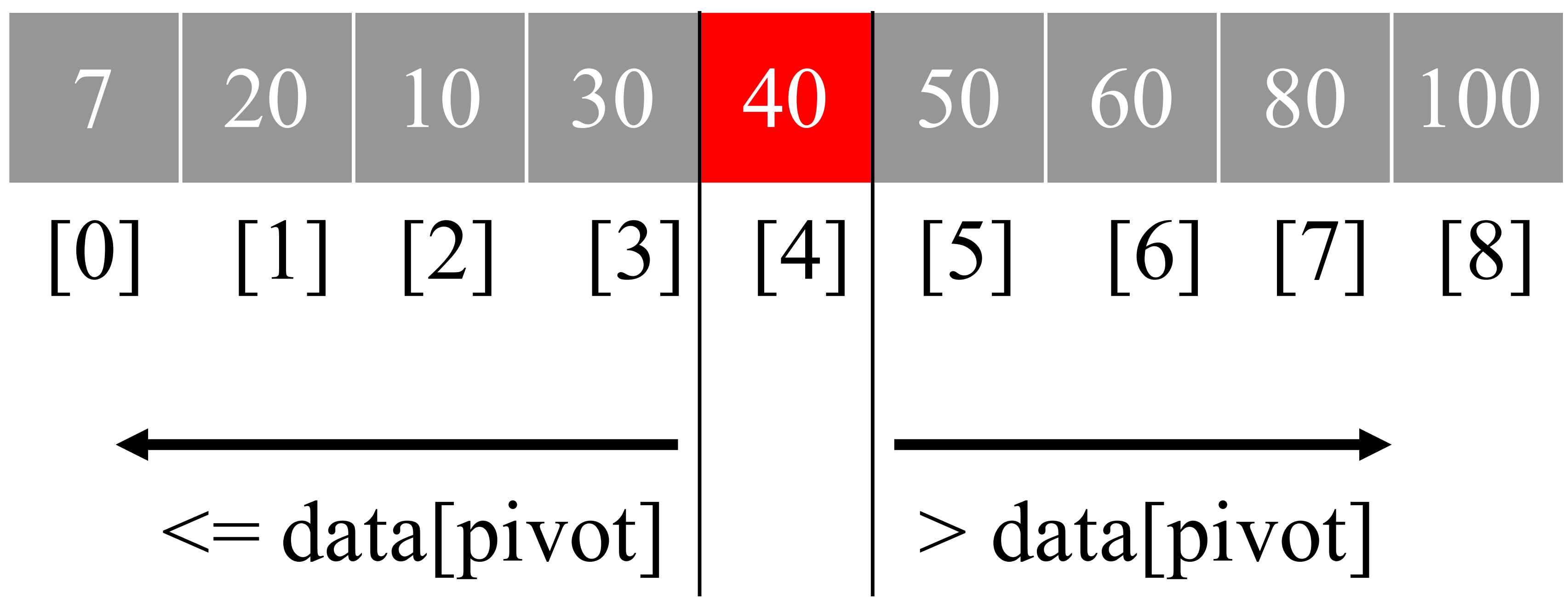
40	20	10	80	60	50	7	30	100

Pick Pivot Element

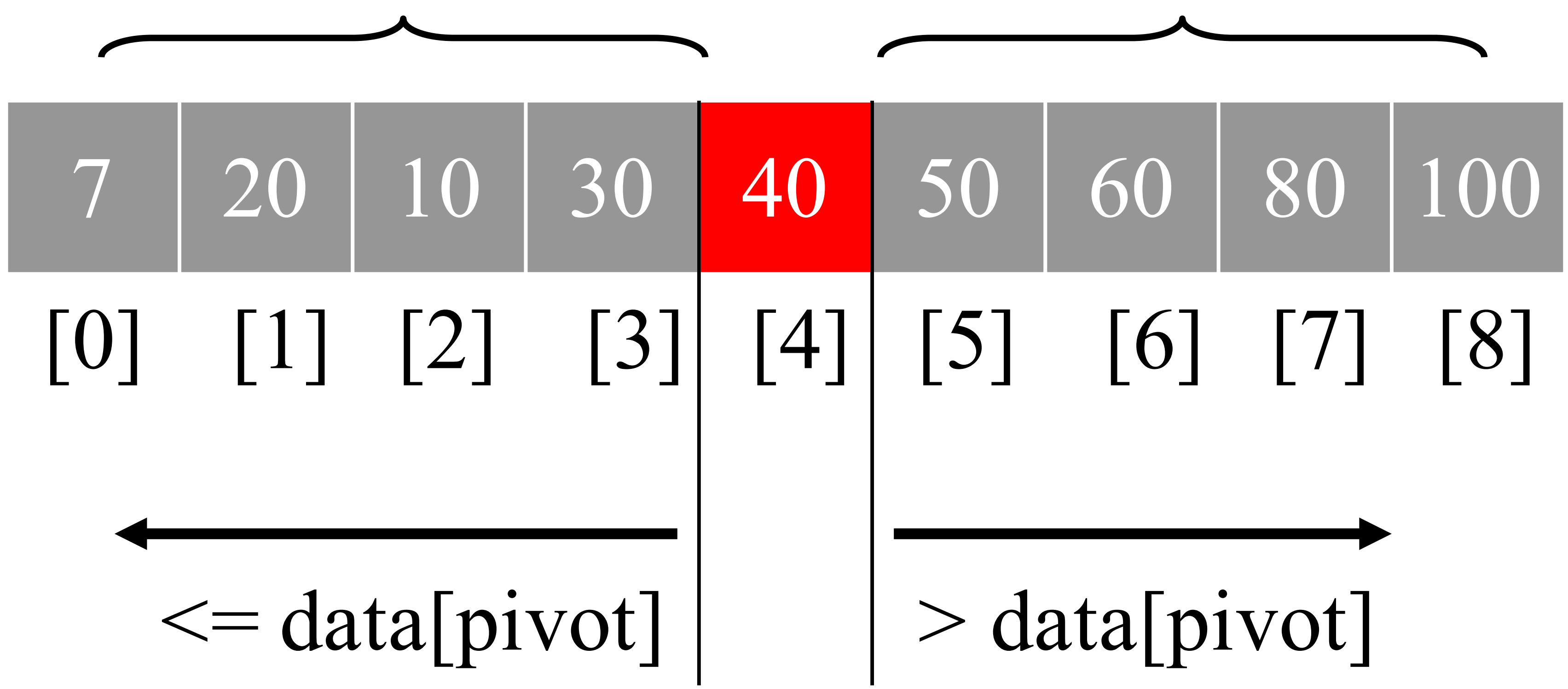
- There are a number of ways to pick the pivot element. In this example, we will use the first element in the array:

40	20	10	80	60	50	7	30	100

Partition



Recursion: Quicksort Sub-arrays



Selection in Expected Linear Time

- Randomized-Select(A[p..r],i) //looking for ith o.s.

```
if p = r
  return A[p]
q <— Randomized-Partition(A,p,r)
k <— q-p+1              //the size of the left partition
if i=k                  //then the pivot value is the answer
return A[q]
else if i < k           //then the answer is in the front
return Randomized-Select(A,p,q-1,i)
else                    //then the answer is in the back half
return Randomized-Select(A,q+1,r,i-k)
```

Example

Find the 2-nd order statistic

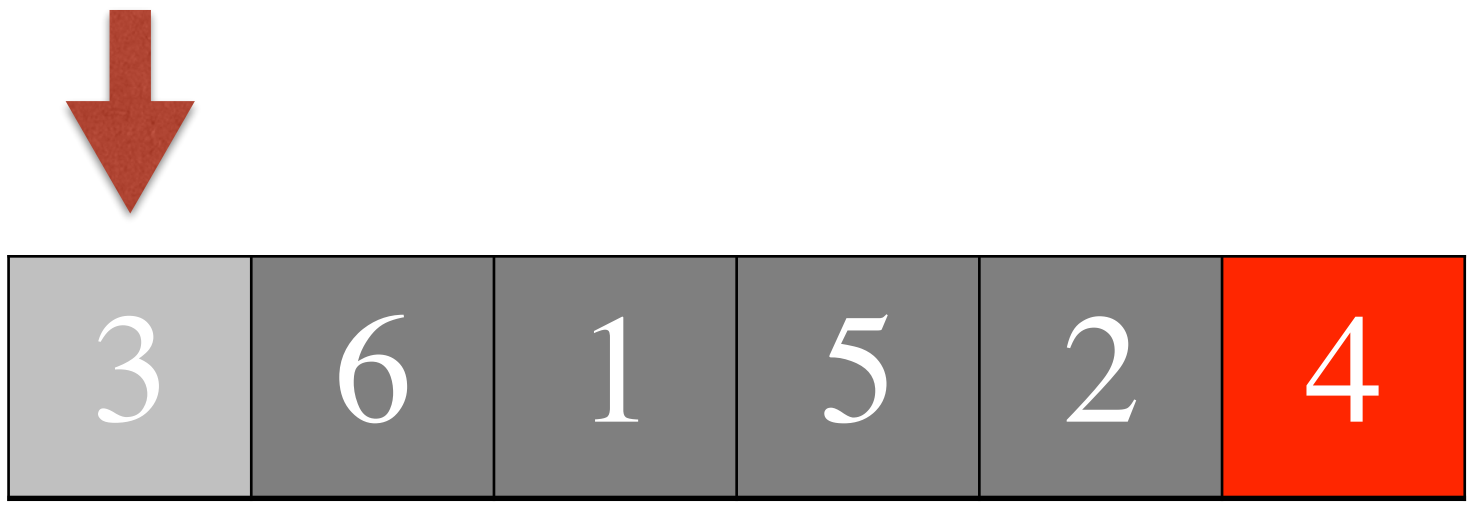
3	6	1	5	2	4

Example

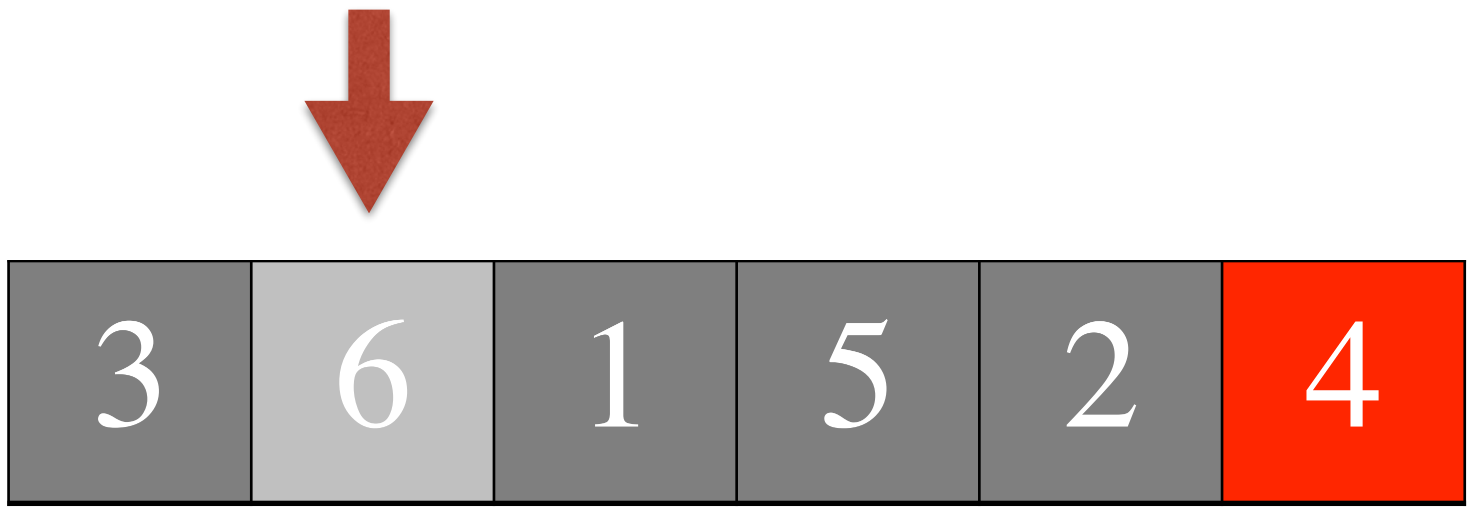
Pivot

3	6	1	5	2	4

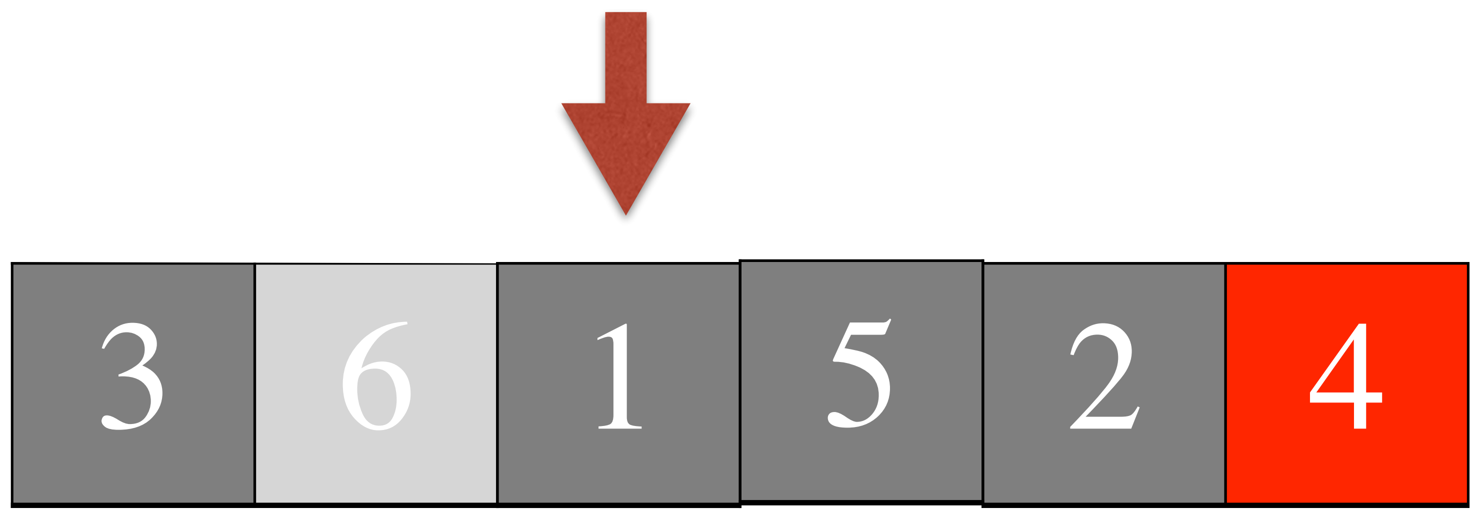
Example



Example



Example

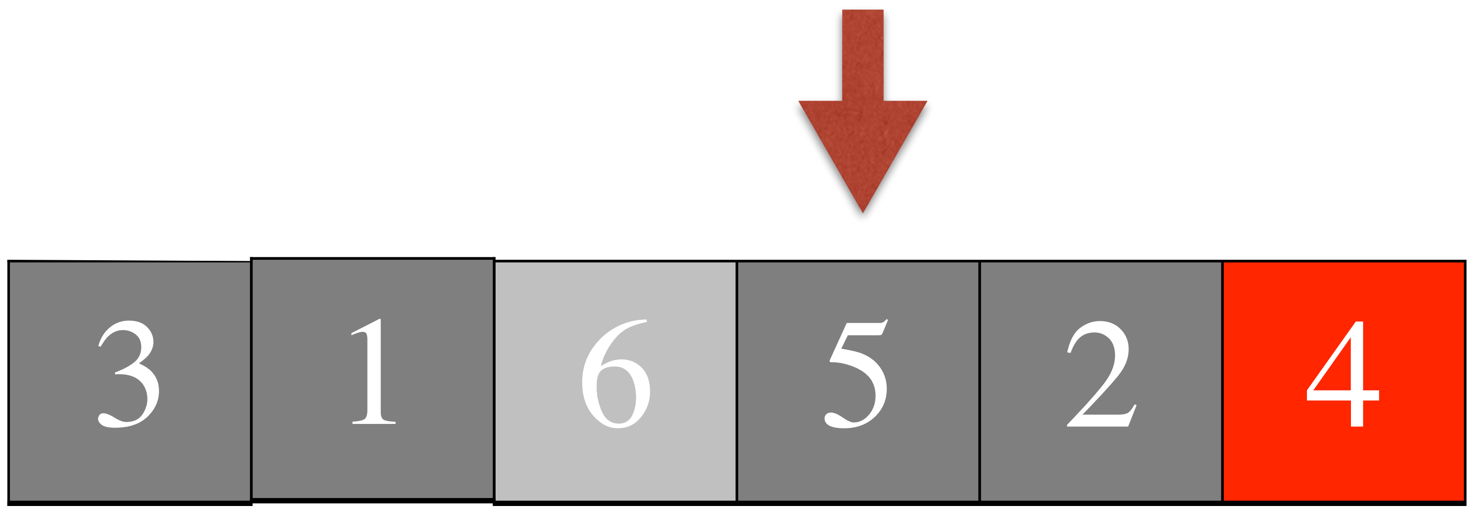


Exchange

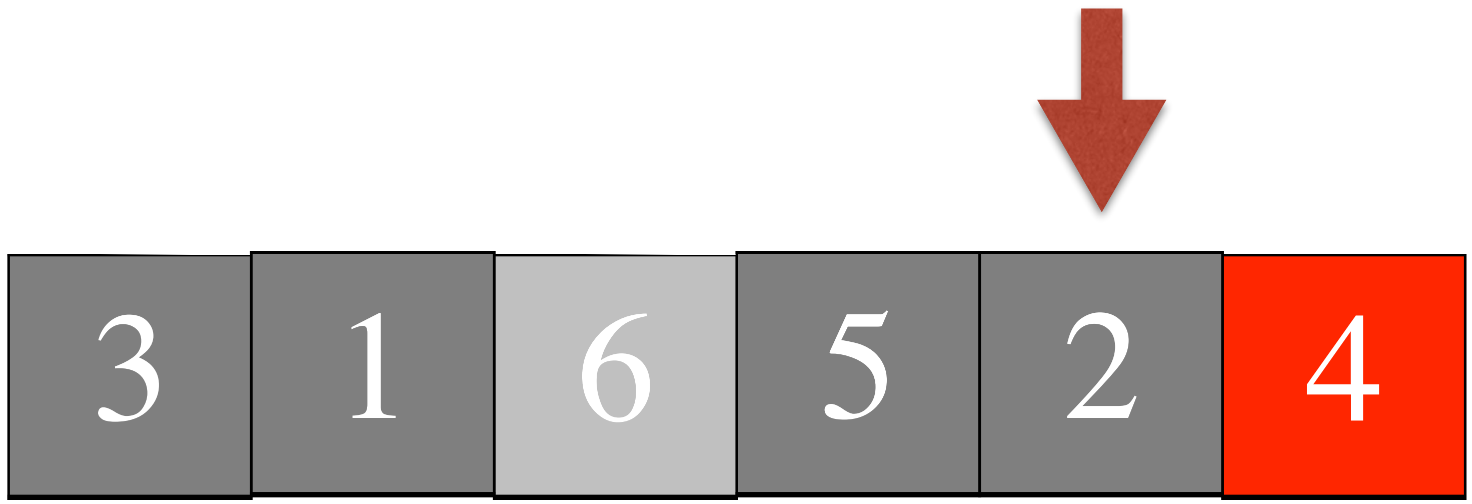
Example

3	1	6	5	2	4

Example

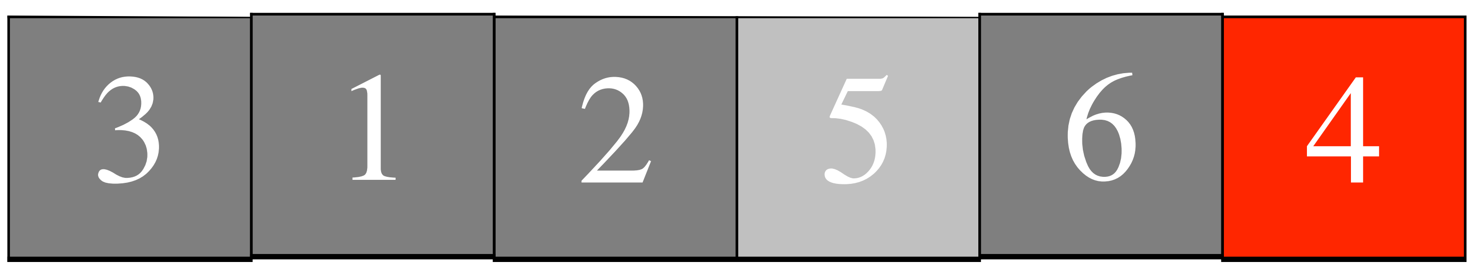


Example

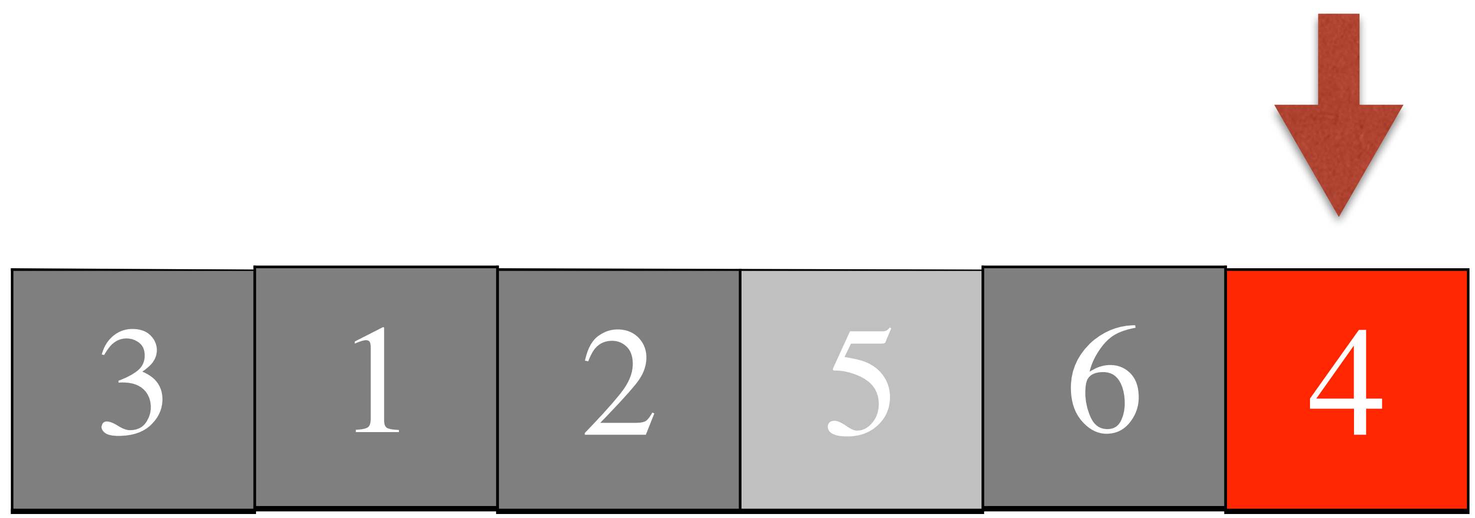


Exchange

Example



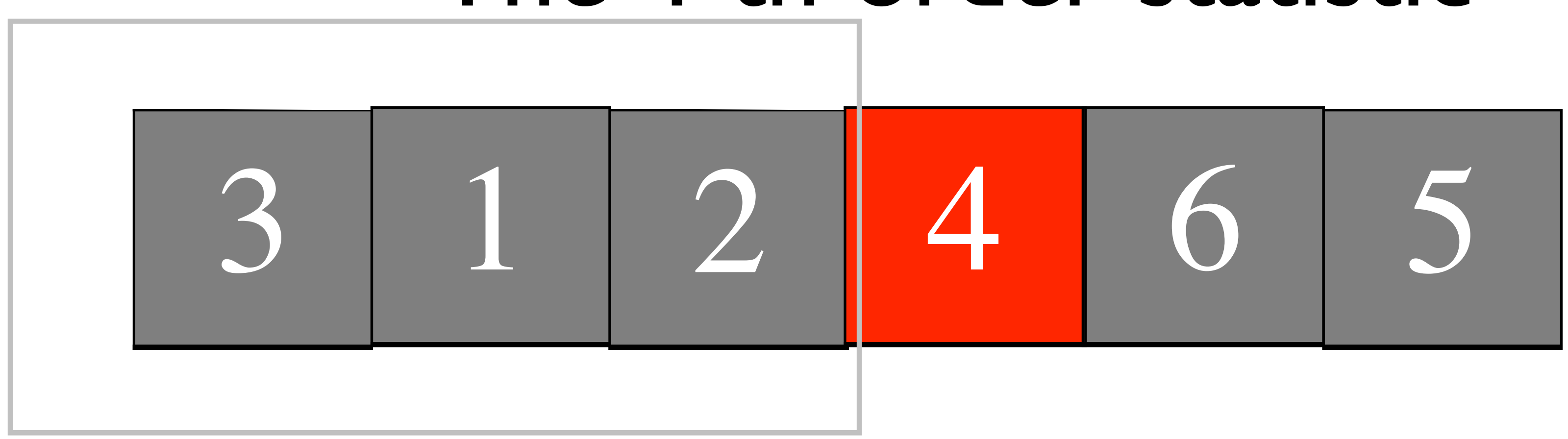
Example



Exchange

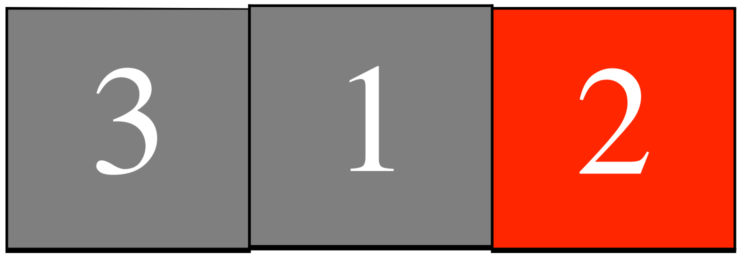
Example

The 4-th order statistic

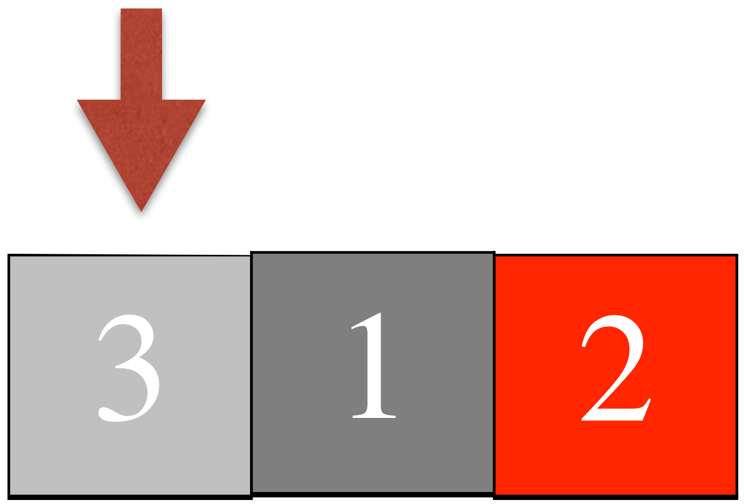


Need to find the 2-nd order statistic in the sub-array

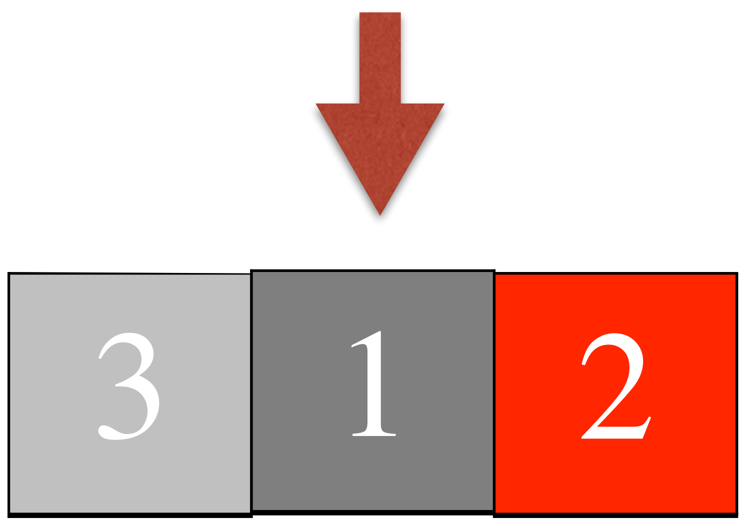
Example



Example



Example

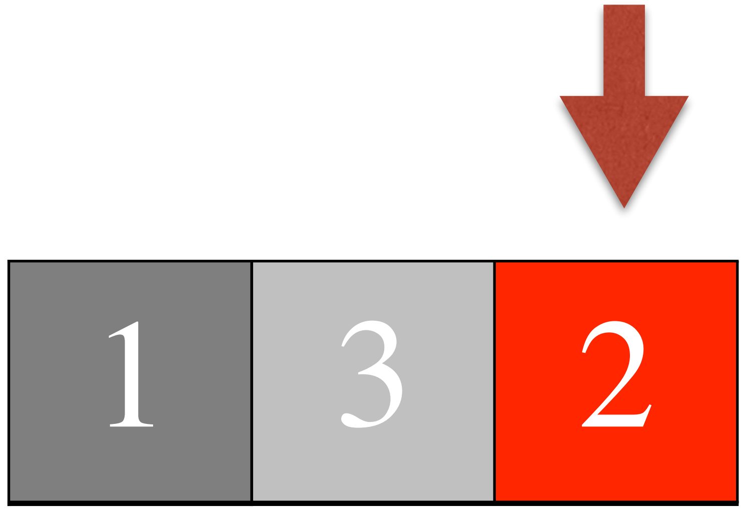


Exchange

Example

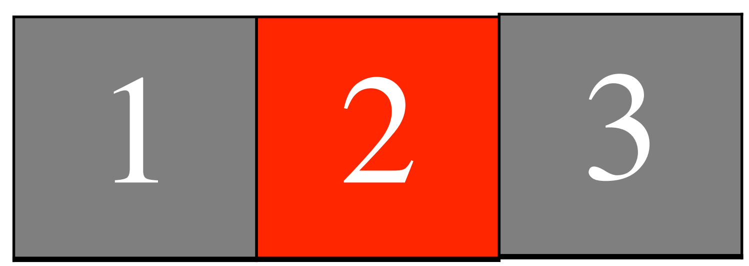
1	3	2

Example



Exchange

Example



The 2-nd order statistic

Randomized Selection: Analysis

- Analyzing RandomizedSelect()
  - Worst case: partition always 0:n-1
    - $T(n) = T(n-1) + O(n) = O(n^2)$
    - No better than sorting!
  - "Best" case: suppose a 9:1 partition
    - $T(n) = T(9n/10) + O(n) = O(n)$
    - Better than sorting!
  - Average case: $O(n)$ remember from QuickSort

Worst-Case Linear-Time Selection

- Randomized algorithm works well in practice
- What follows is a worst-case linear time algorithm, really of theoretical interest only
- Basic idea:
  - Guarantee a good partitioning element
  - Guarantee worst-case linear time selection
- Warning: Non-obvious & unintuitive algorithm ahead!
- Blum, Floyd, Pratt, Rivest, Tarjan (1973)

Worst-Case Linear-Time Selection

- The algorithm in words:
  1. Divide n elements into groups of 5
  2. Find median of each group (How? How long?)
  3. Use Select() recursively to find median x of the n/5
  4. Partition the n elements around x. Let k = rank(x)
  5. if (i == k) then return x

     if (i < k) then use Select() recursively to find ith smallest element in first partition
     else (i > k) use Select() recursively to find (i-k)th smallest element in last partition

Order Statistics: Algorithm

Select(A,n,i):	$T(n)$
Divide input into groups of size 5.	$O(n)$
/* Partition on median-of-medians */	
medians = array of each group's median.	$O(n)$
pivot = Select(medians, n/5 , n/10)	$T(\lceil \frac{n}{5} \rceil)$
Left Array L and Right Array G = partition(A, pivot)	
/* Find ith element in L, pivot, or G */	
k = |L| + 1	$O(1)$
If i=k, return pivot	$O(1)$
If i<k, return Select(L, k-1, i)	$T(k)$
If i>k, return Select(G, n-k, i-k)	$T(n-k)$

Order Statistics: Analysis

less

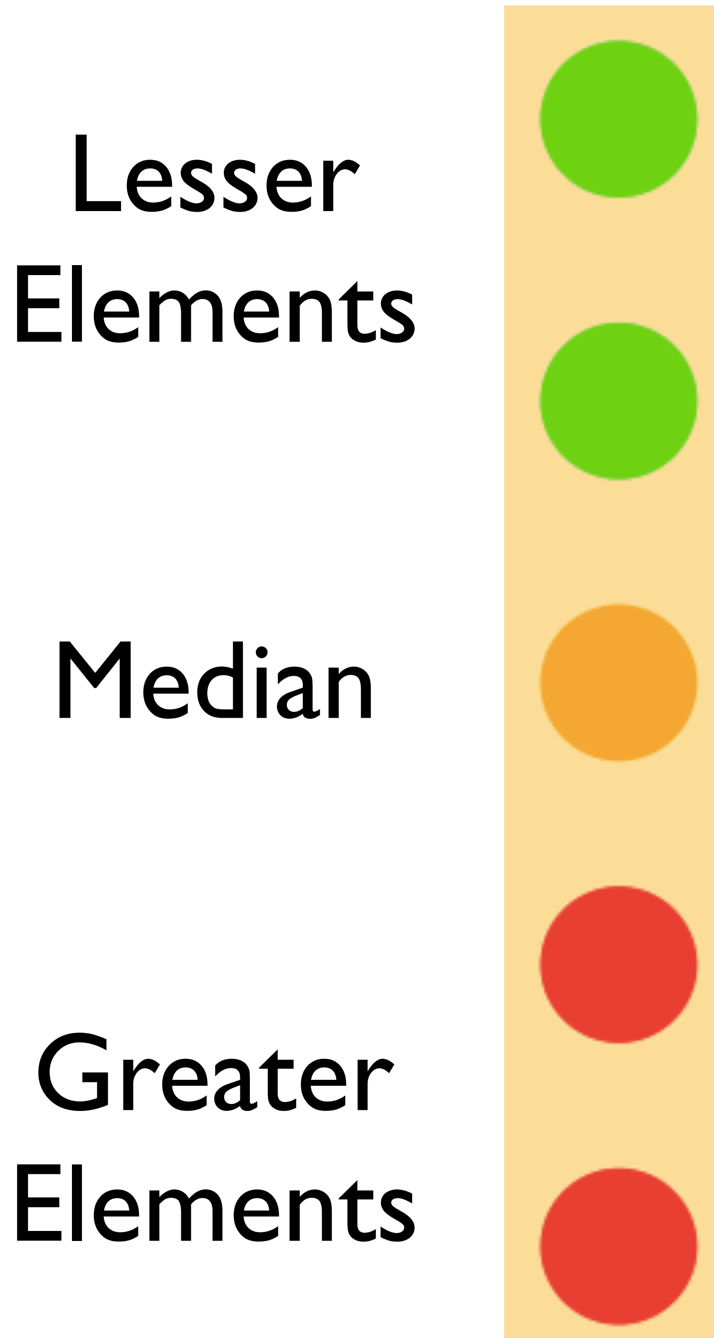
greater

$$T(n) = T(\lceil \frac{n}{5} \rceil) + T(\max(k-1, n-k)) + O(n)$$

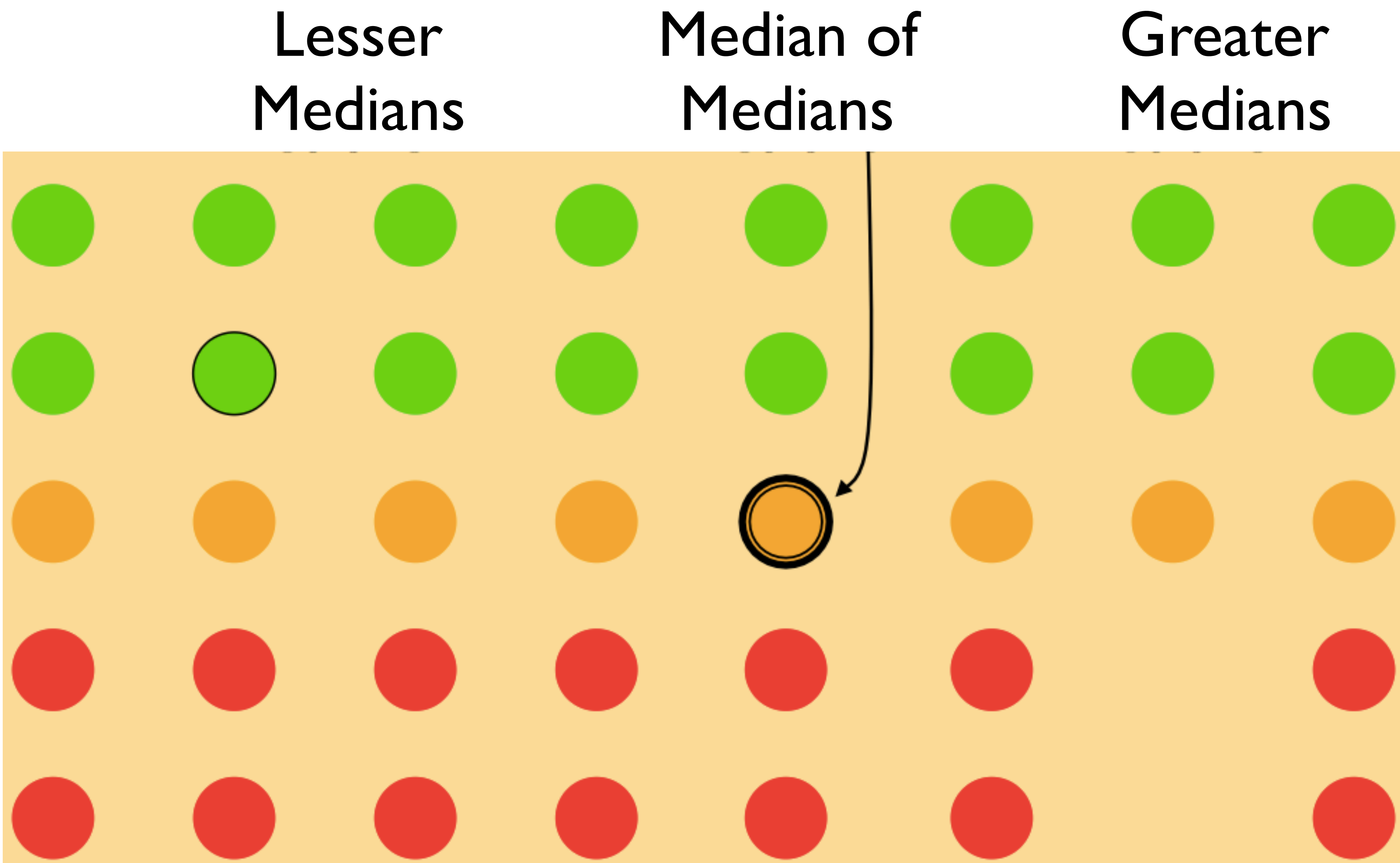
How to simplify?

Order Statistics: Analysis

- One Group of 5 elements

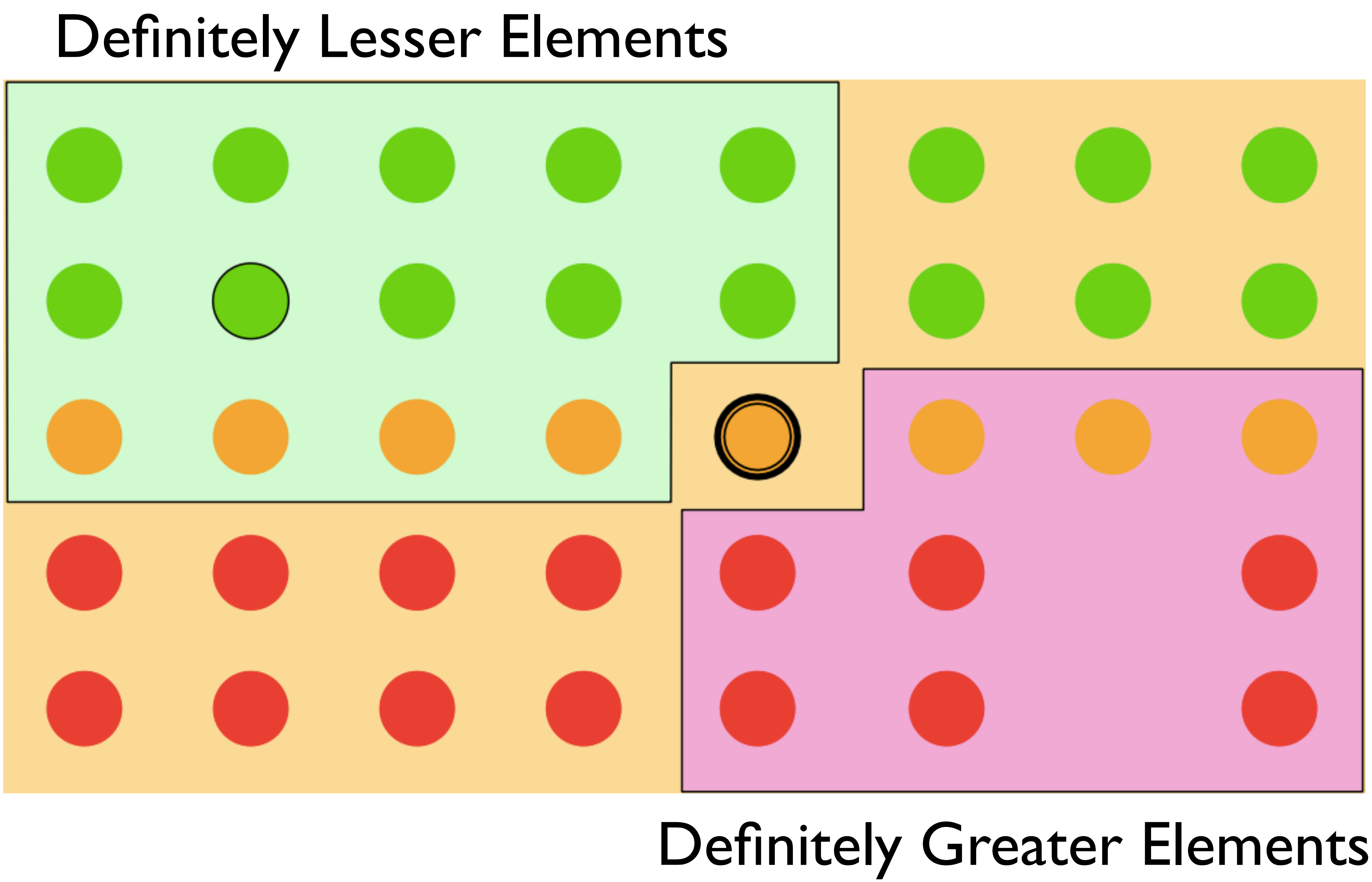


Order Statistics: Analysis

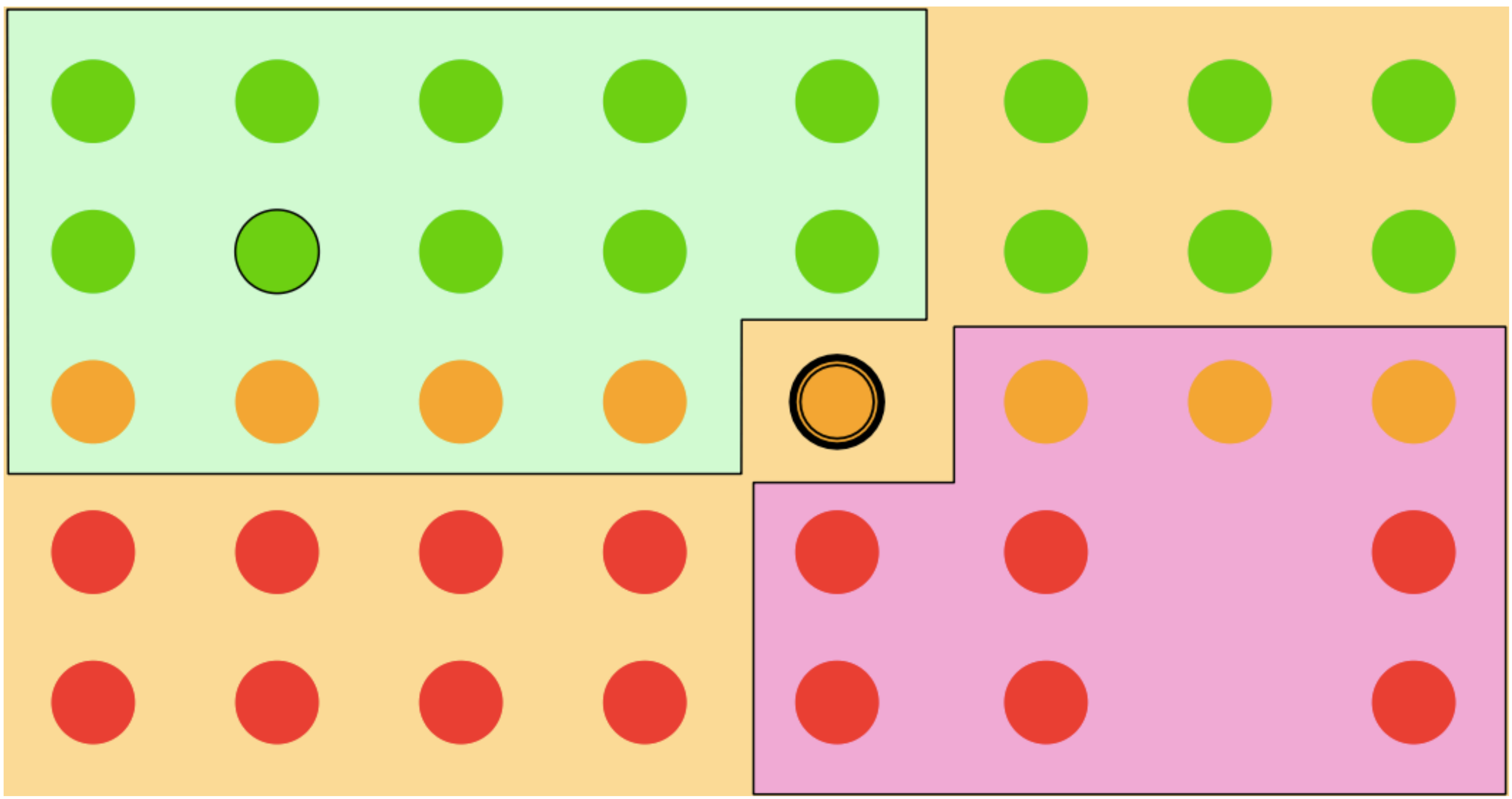


All groups of 5 elements. (At most one smaller group.)

Order Statistics: Analysis



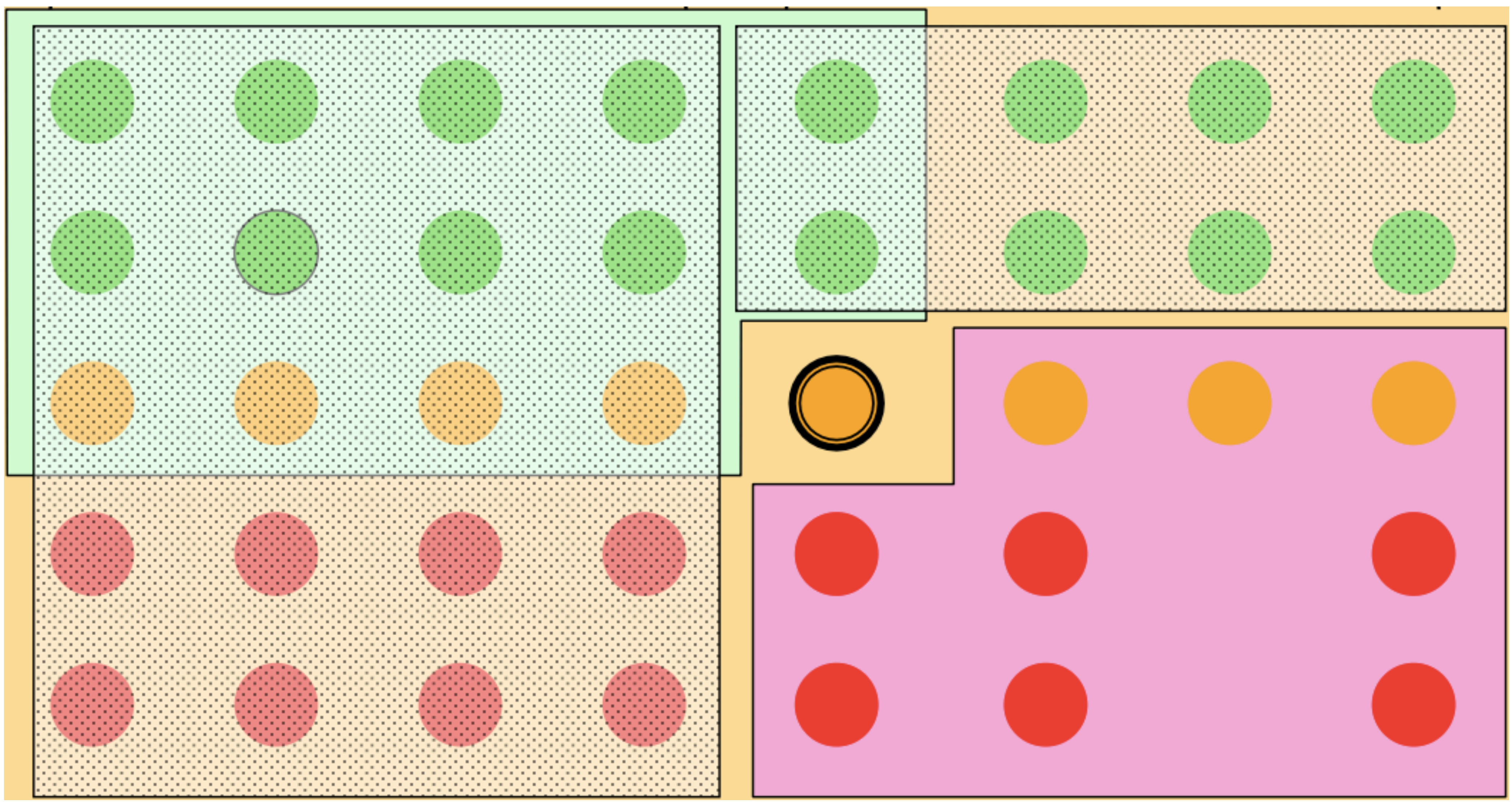
Order Statistics: Analysis 1



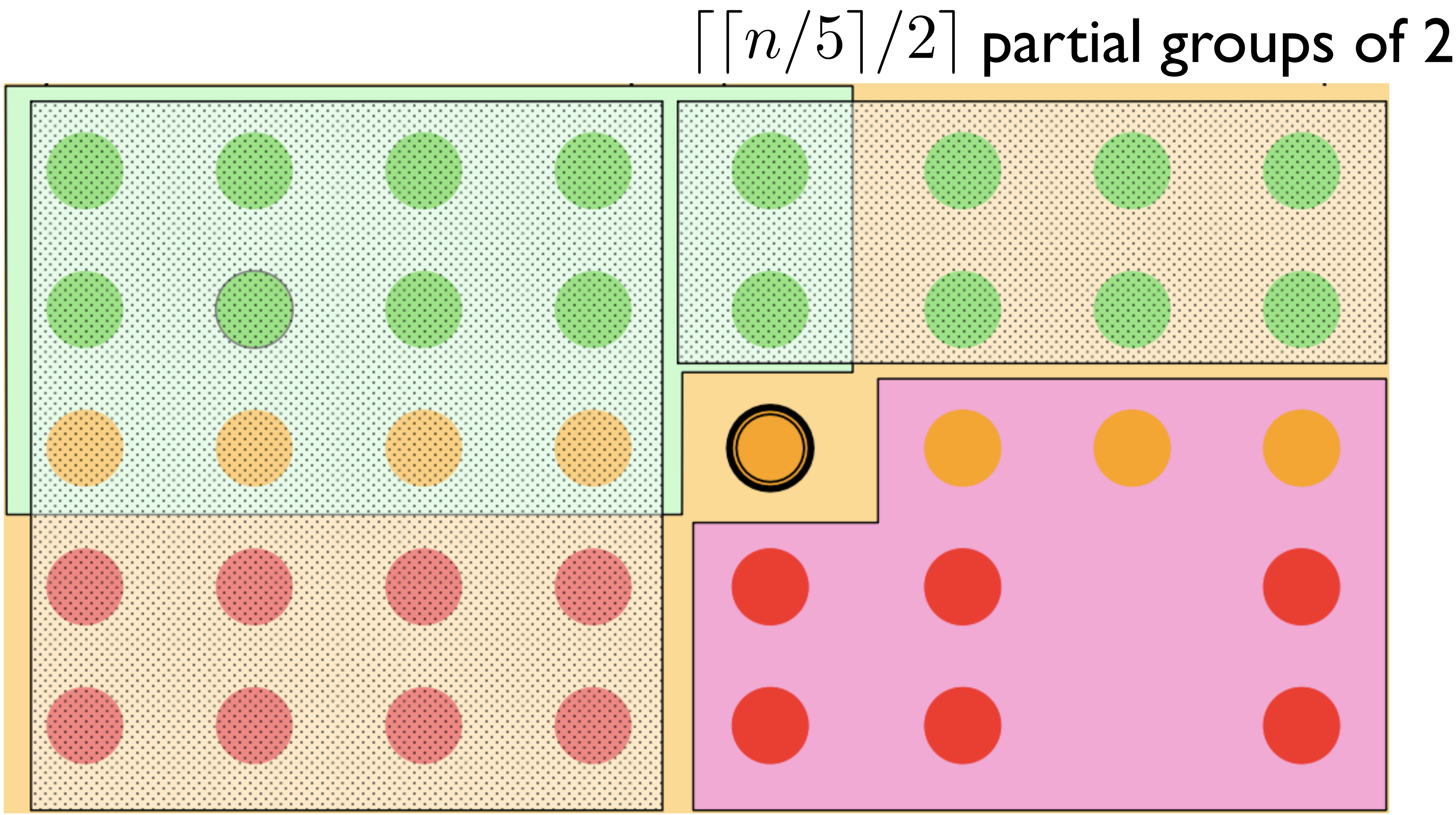
Must recur on all elements outside one of these boxes.
How many?

Order Statistics: Analysis 1

$\lfloor \lceil n/5 \rceil / 2 \rfloor$ full groups of 5

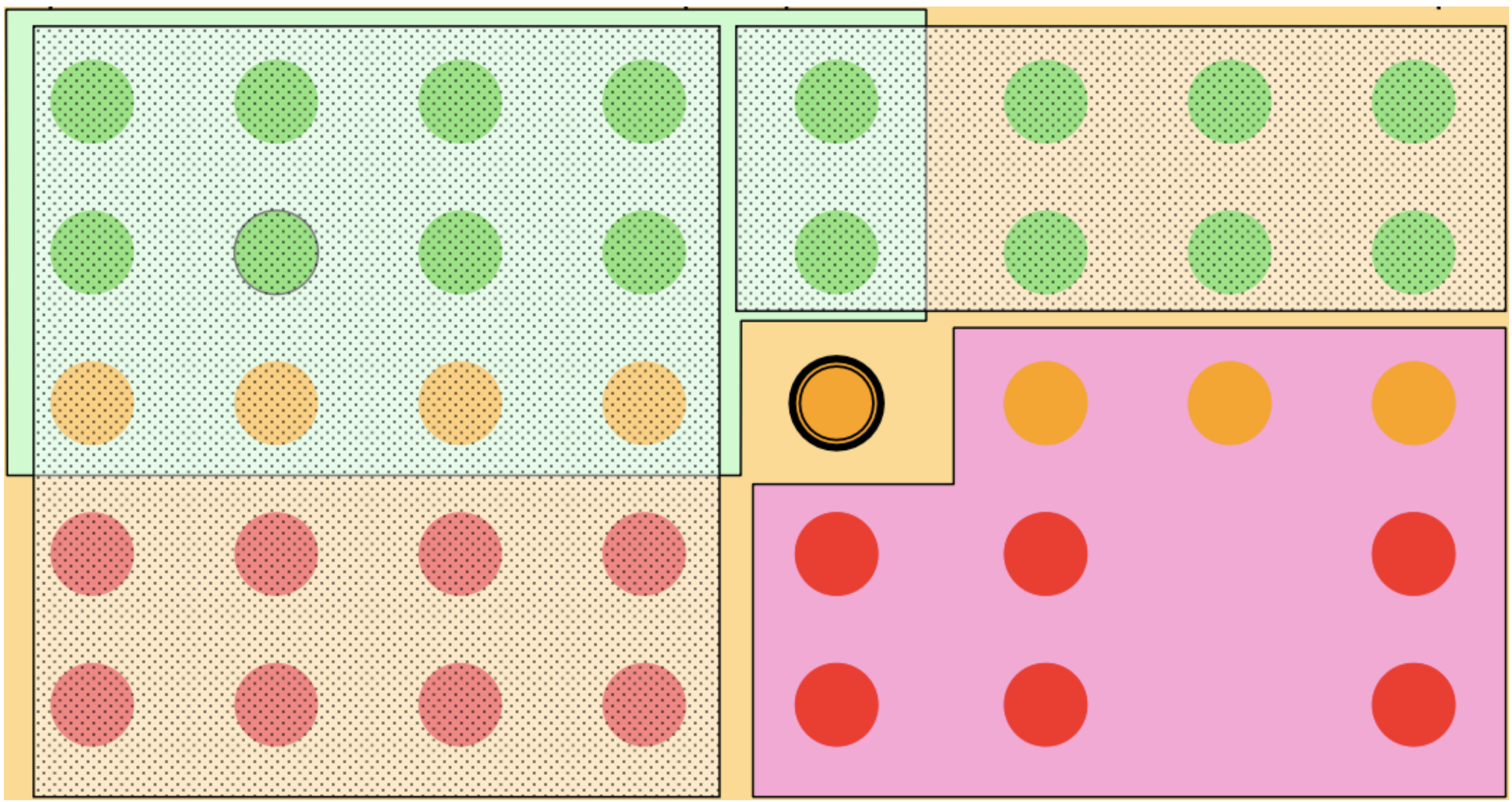


Order Statistics: Analysis 1



Order Statistics: Analysis 1

$\lfloor \lceil n/5 \rceil /2 \rfloor$ full groups of 5 $\lceil \lceil n/5 \rceil /2 \rceil$ partial groups of 2



Count elements outside smaller box. $5\lfloor \lceil n/5 \rceil /2 \rfloor + 2\lceil \lceil n/5 \rceil /2 \rceil \leq 7n/10 + 6$

Order Statistics: Analysis

$$T(n) = T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + O(n)$$

A very unusual recurrence. How to solve?

Order Statistics: Analysis

Substitution: Prove $T(n) \leq cn$

$$
\begin{aligned}
T(n) &\leq c\lceil \frac{n}{5} \rceil + c(\frac{7n}{10} + 6) + dn \\
&\leq c(\frac{n}{5} + 1) + c(\frac{7n}{10} + 6) + dn \\
&= \frac{9}{10}cn + 7c + dn \\
&= cn - (cn/10 - 7c - dn) \\
&\leq cn
\end{aligned}
$$

when choose c, d such that $cn/10 - 7c - dn \geq 0$

Order Statistics

Why groups of 5?

Order Statistics

Why groups of 5?

Sum of two recurrence sizes must be < 1.
Grouping by 5 is smallest size that works.

Thank You!

