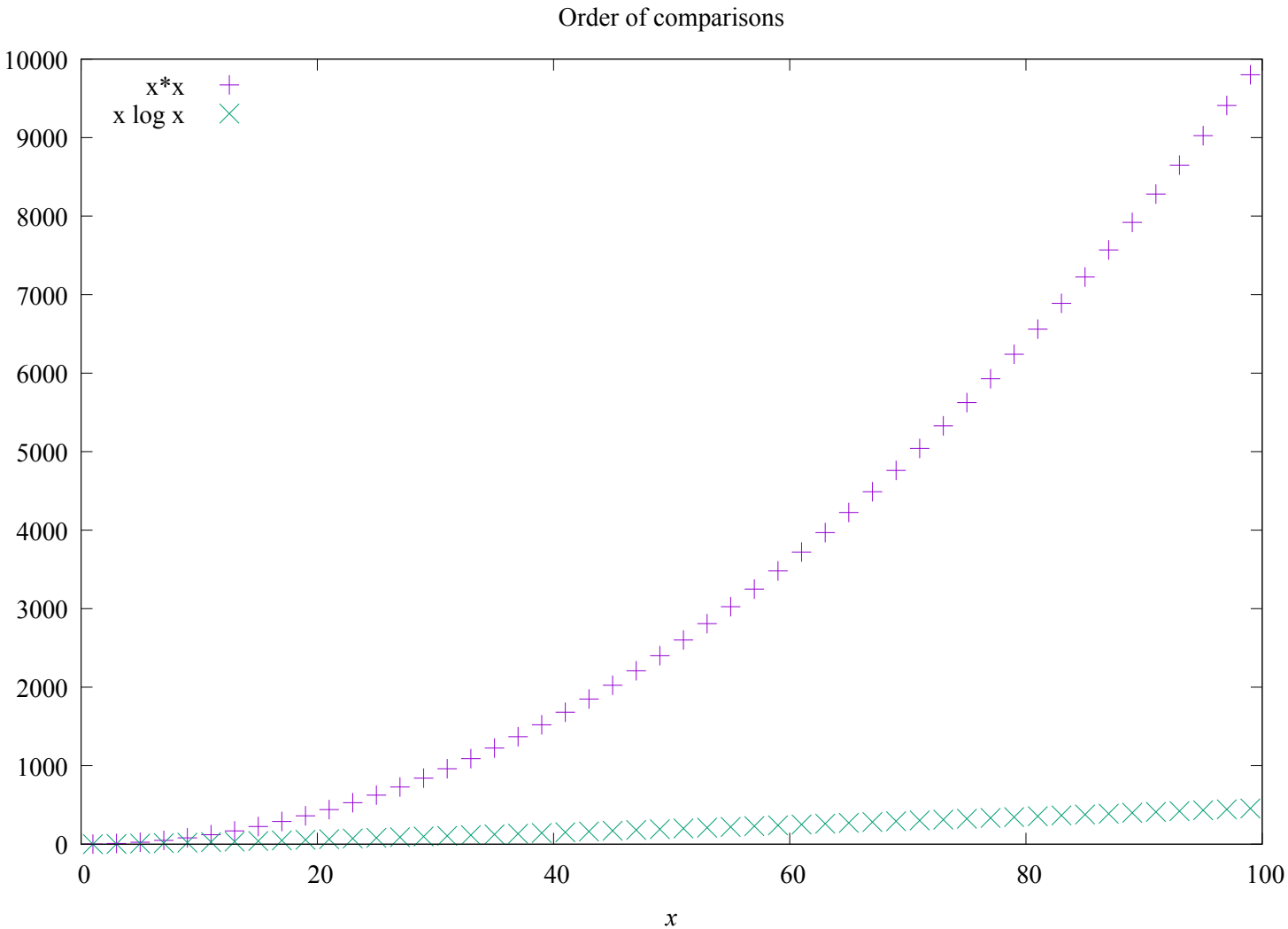
Review : Object Oriented Programming

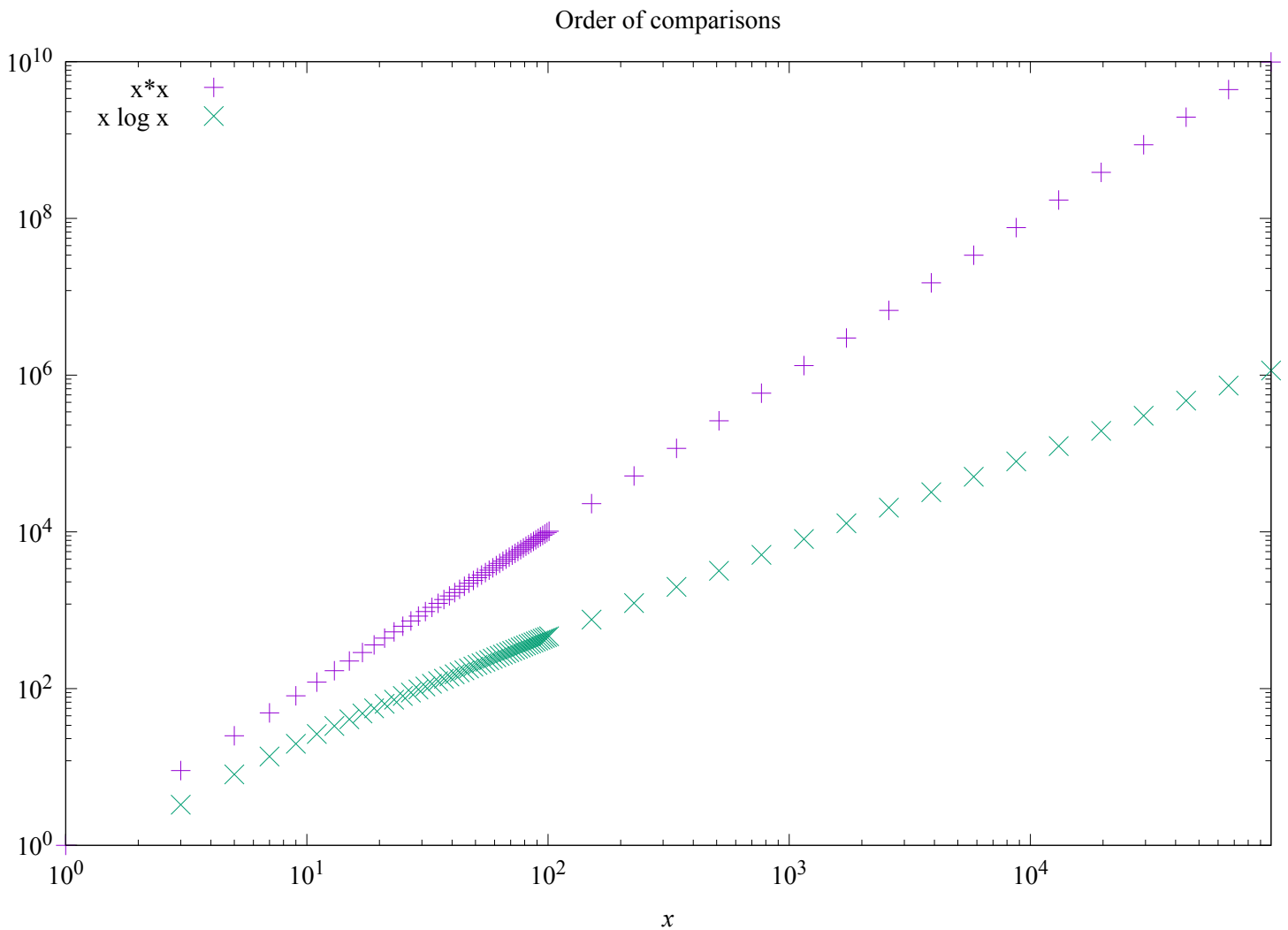
モデル化とシミュレーション特論
2021 年度前期
佐賀大学理工学研究科　只木進一

1. Various sort methods
2. Sample Programs
3. Review sorting in the viewpoint of OOP

Various sort methods

- Sort method with n^2 comparisons
  - bubble sort, selection sort, insertion sort
- Sort method with $n \log n$ comparisons
  - merge sort, quick sort
- $n^2 \gg n \log n$ for $n \gg 1$
- Observe the number of comparisons for various sorting methods


Order of comparisons
x*x
x log x
10000
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
x


Order of comparisons
x*x
x log x
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
10^{0}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
x

Bubble sort

Algorithm 1 Bubble sort

n is the size of the array d
for $i = n$; $i > 0$; $i--$ **do**
  for $j = 0$; $j < i - 1$; $j++$ **do**
    if $d_{j+1} < d_j$ **then**
      swap d_{j+1} with d_j
    end if
  end for
end for

Attention: two nested loops require $O(n^2)$ comparisons.

selection sort

Algorithm 2 selection sort

n is the size of the array d
for $i = 0$; $i < n - 1$; $i++$ **do**
    m is the position of the smallest element between i and the last
    if $m \neq i$ **then**
        swap the element at i with that at m
    end if
end for

Attention : searching the smallest element is the inner loop and requires $O(n)$ comparisons.

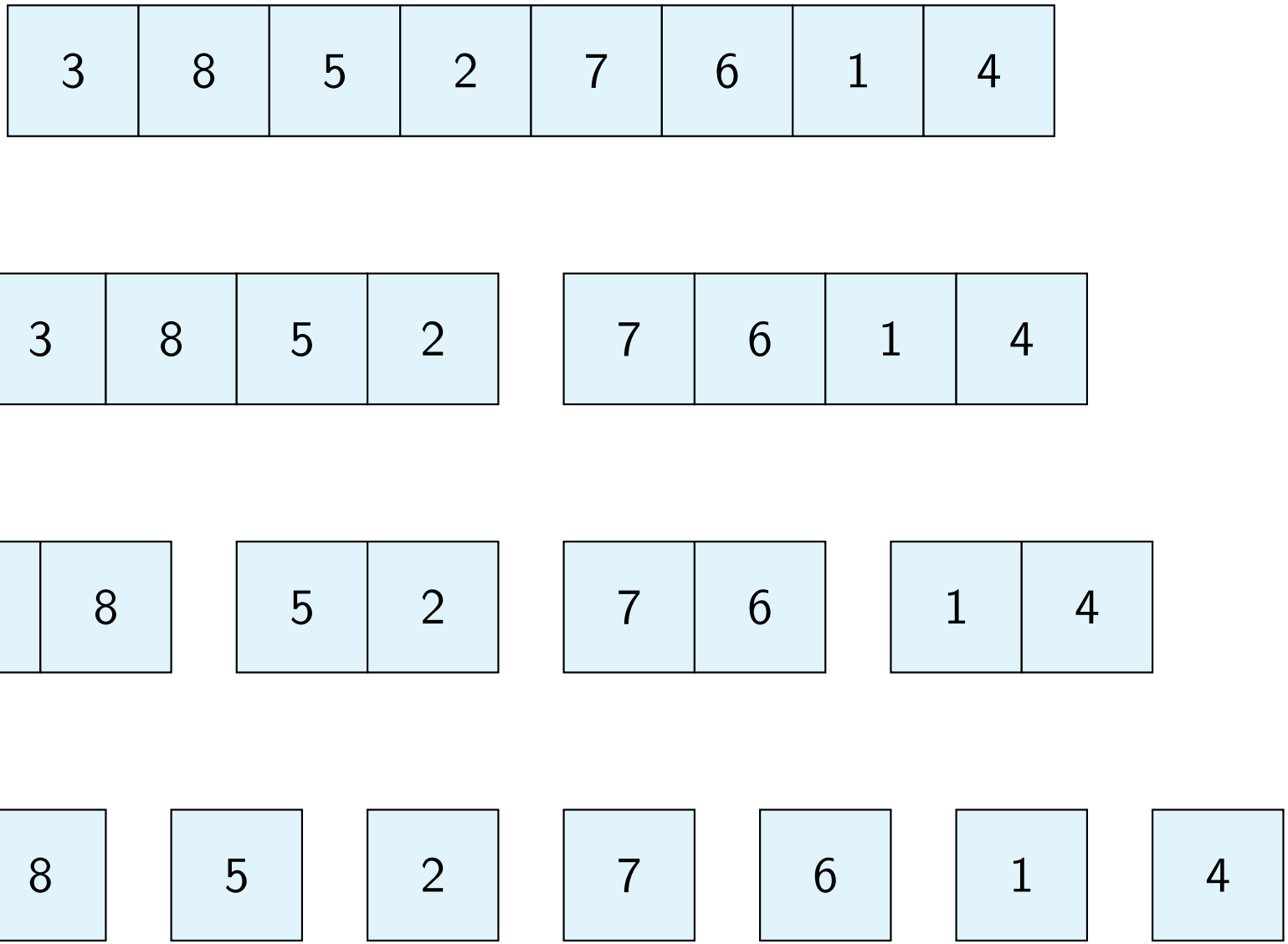
insertion sort

Algorithm 3 insertion sort

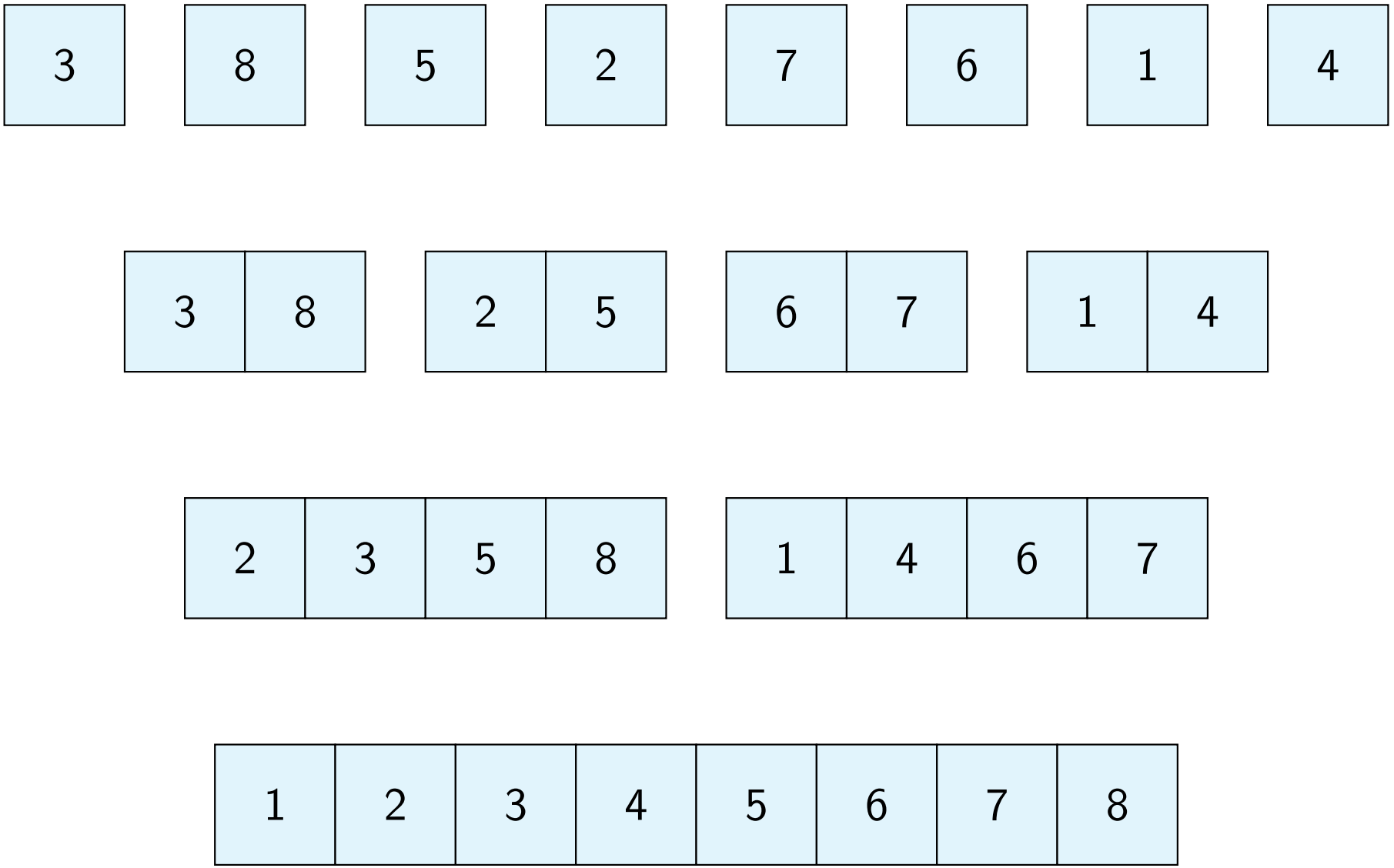
n is the size of the array d
for $i = 0$; $i < n$; $i++$ **do**
    m is the index of the smallest element between i and the last
    if $m \neq i$ **then**
        insert element at m into i
    end if
end for

Attention : searching the smallest element is the inner loop and requires $O(n)$ comparisons.

Merge sort: dividing list into the smallest size



Merge sort: merging elements



merge sort

Algorithm 4 merge sort

n: the size of the array d
$k_{\text{left}} = 0,\ k_{\text{right}} = n$
procedure $\textsc{sortSub}(k_{\text{left}}, k_{\text{right}})$
    $k_{\text{middle}} = (k_{\text{left}} + k_{\text{right}})/2$ (truncate to integer)
    $\textsc{sortSub}(k_{\text{left}}, k_{\text{middle}})$
    $\textsc{sortSub}(k_{\text{middle}}, k_{\text{right}})$
    Combining two sorted lists
end procedure

- Merging operations in horizontal direction needs $O(n)$ comparisons
- Number of layers in vertical direction is $O(\log n)$

quick sort

Algorithm 5 quick sort

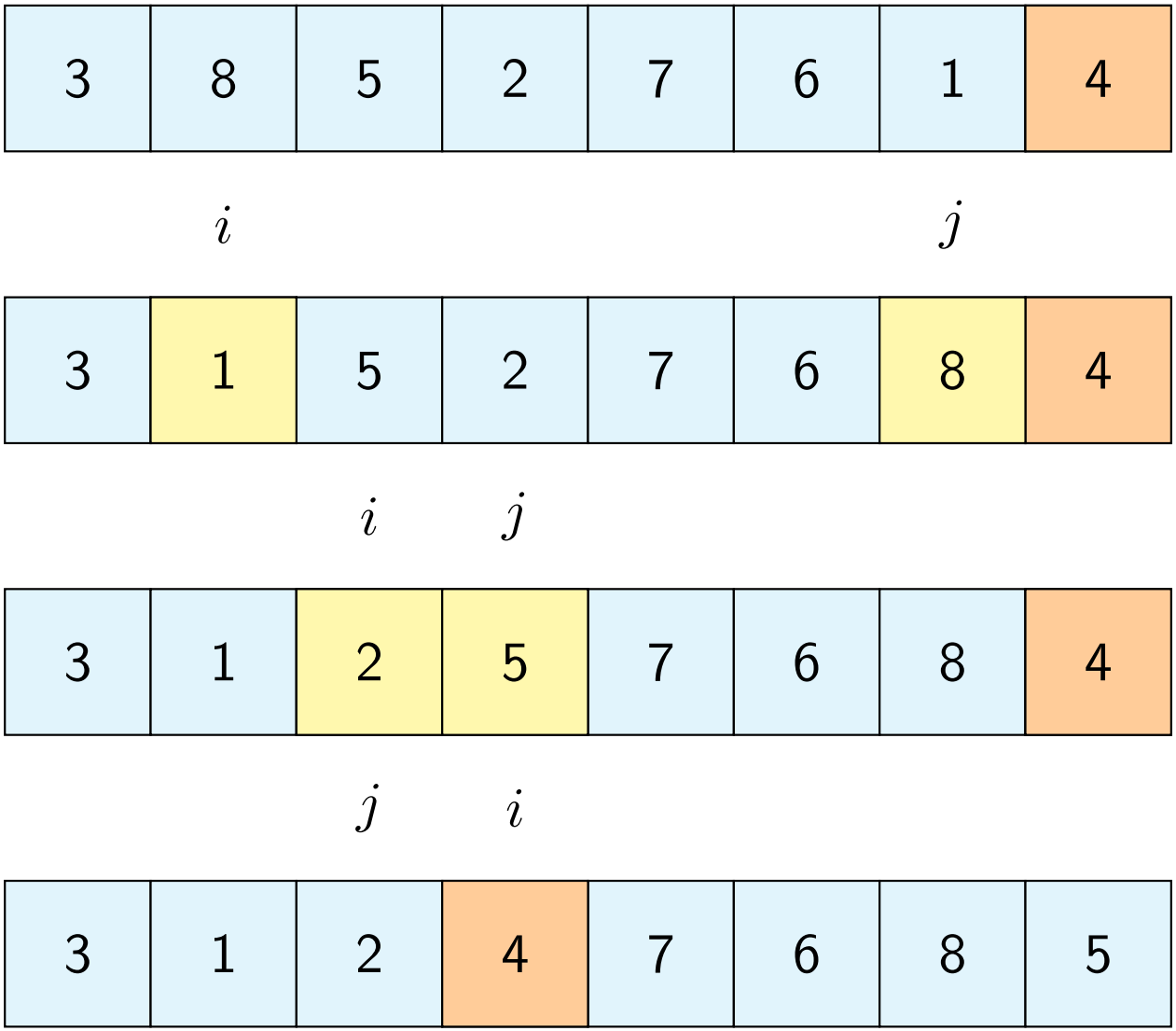
n: the size of the array d
$k_{\mathsf{left}} = 0, k_{\mathsf{right}} = n$
procedure $\textsc{sortSub}(k_{\mathsf{left}}, k_{\mathsf{right}})$
    $k_{\mathsf{middle}} = \textsc{partition}(k_{\mathsf{left}}, k_{\mathsf{right}})$
    $\textsc{sortSub}(k_{\mathsf{left}}, k_{\mathsf{middle}})$
    $\textsc{sortSub}(k_{\mathsf{middle}}, k_{\mathsf{right}})$
end procedure

quick sort : continued

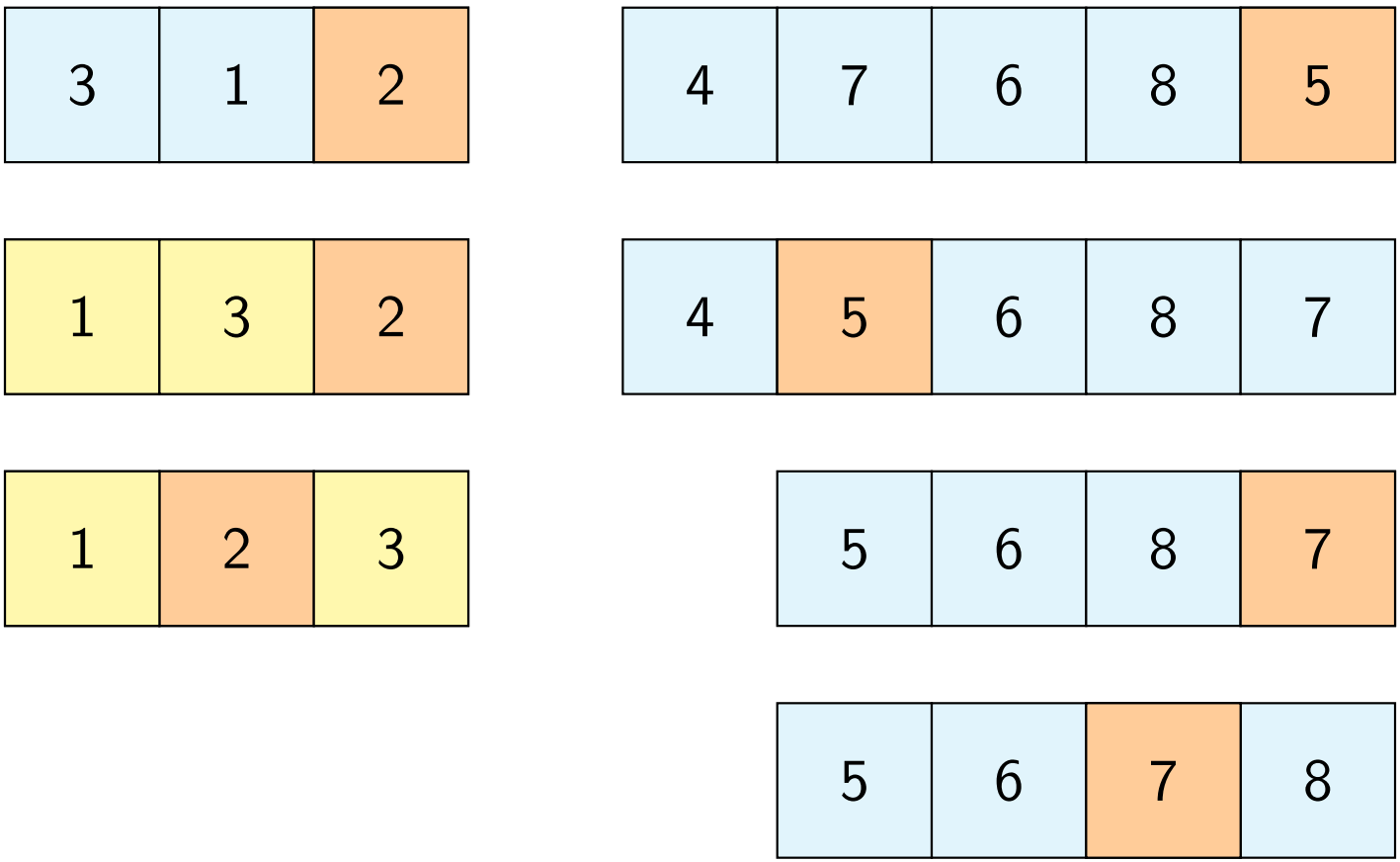
Algorithm 6 partition

procedure PARTITION(k, ℓ)
$v = d_{\ell-1}$
$i = k, j = \ell - 1$
while $i < j$ **do**
Search an element greater than or equal v from the left. Its position is i.
Search an element less than or equal v from the right. Its position is j.
if $i < j$ **then**
Swap d_i with d_j
end if
end while
Swap d_i with $d_{\ell-1}$
Return i
end procedure

Example: quick sort

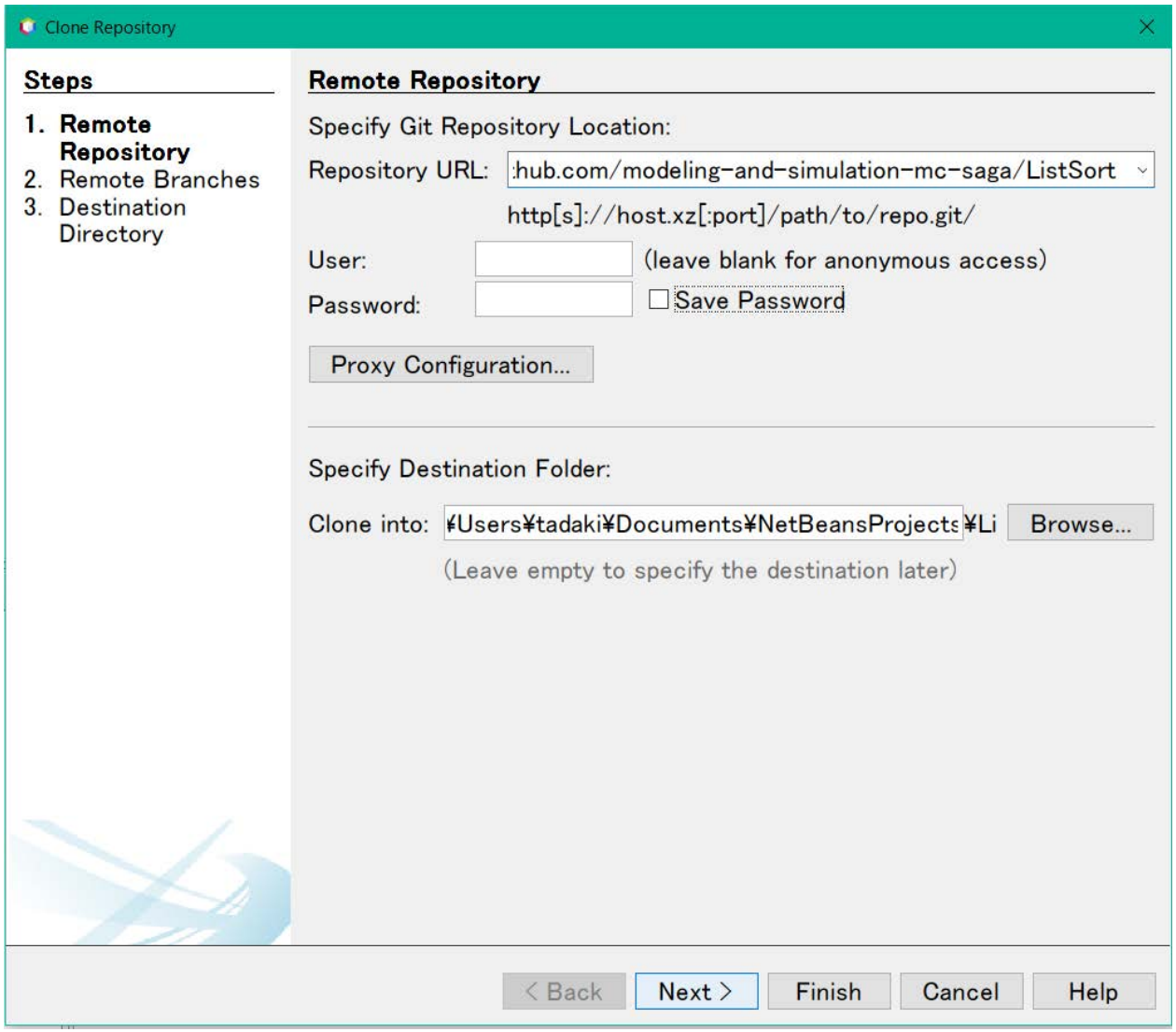


Example: quick sort, continued



Get Sample Programs by NetBeans

"Teams" → "Git" → "Clone"



Get sample programs by Git command

- Obtain Git from `https://git-scm.com/downloads`
- Use command

  `git clone repository`

Repository

- https://github.com/modeling-and-simulation-mc-saga/ListSort
- https://github.com/modeling-and-simulation-mc-saga/MyLib

Review sorting in the viewpoint of OOP

- Minimum common functions for sorting
  - Target objects are required to have large-and-small relationship
  - Minimum functions for sorting : compare and swap
- `Comparable` interface for target objects

AbstractSort

- Sort objects implementing `Comparable` interface
- Not implement concrete sorting process
- Implement common methods required for sorting
- Function for counting comparisons
- Derived classes
  BubbleSort, InsertionSort, SelectionSort, MergeSort, QuickSort

Simulation results

- n: the number of elements
  - bubble sort and etc. need $O\left(n^2\right)$ comparisons
  - merge sort and etc. need $O\left(n \log n\right)$ comparisons