OOP復習：整列

モデリングとシミュレーション特論

2019年度

只木進一

整列(sort)の方法

- n^2回の比較が必要なタイプ
  - bubble sort, selection sort, insertion sort
- $n \log n$回の比較が必要なタイプ
  - merge-sort, quick sort
- 実際に計測する

bubble sort

```
n : dのサイズ
for (i = n; i > 0; i − −) {
    for (j = 0; j < i − 1; j + +) {
        if (d_{j+1} < d_j) {
            d_{j+1}とd_jを入れ替え
        }
    }
}
```

二重ループで要素を比較

insertion sort

```
n : dのサイズ
for (i = 1; i < n; i++) {
    k = 要素iより小さい要素の位置
    if (k ≠ i) {
        位置iの前に要素kを挿入
    }
}
```

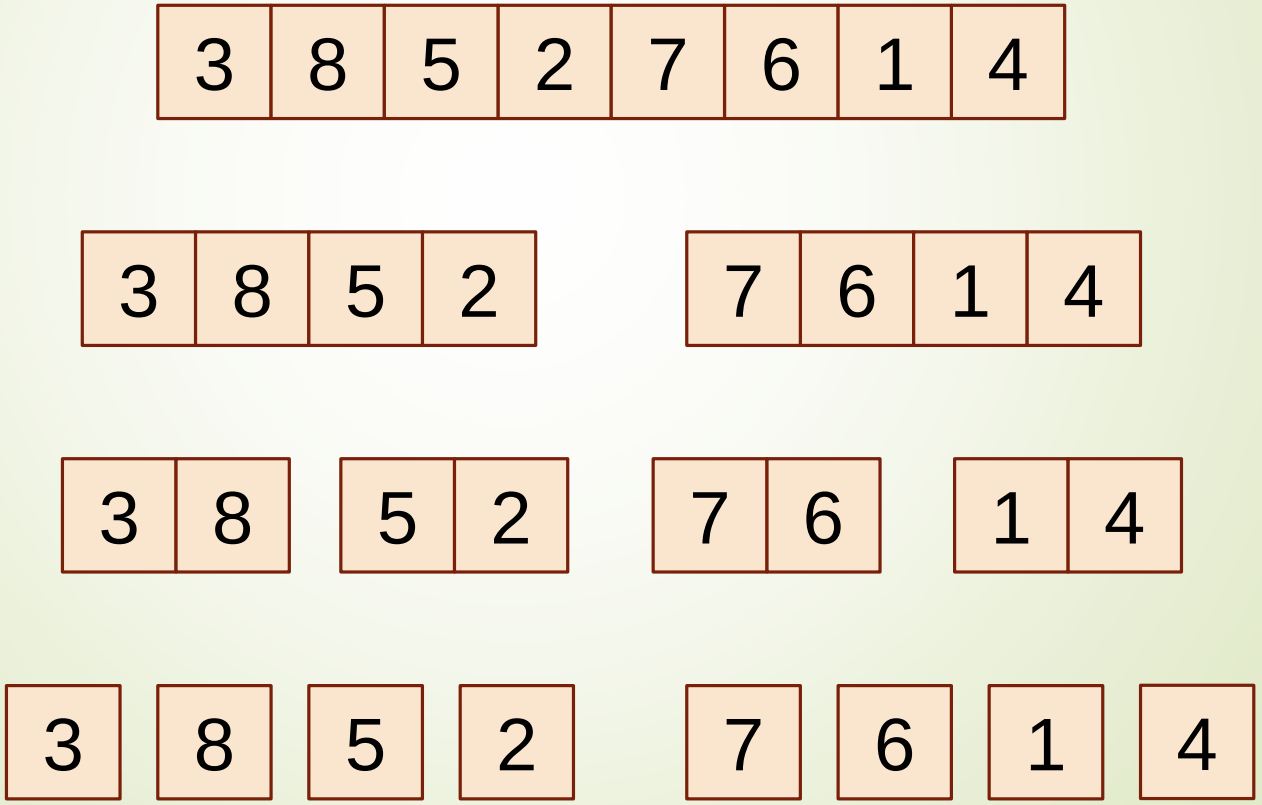
小さい要素を探すのが内側ループに相当

selection sort

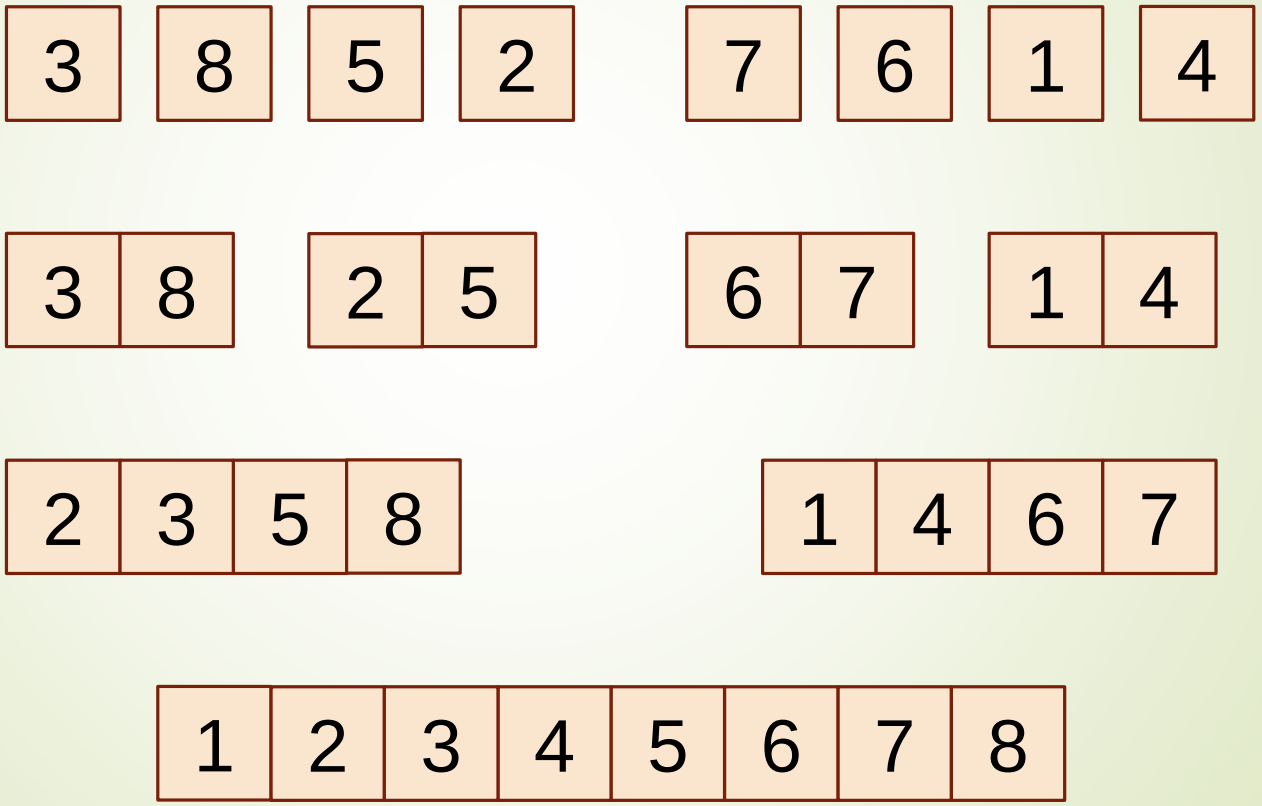
```
n : dのサイズ
for (i = 0; i < n − 1; i++) {
    m = 位置iから最後までで最小の要素の位置
    if (m ≠ i) {
        要素iと要素mを入れ替え
    }
}
```

小さい要素を探すのが内側ループに相当

Merge Sort
リストの分離



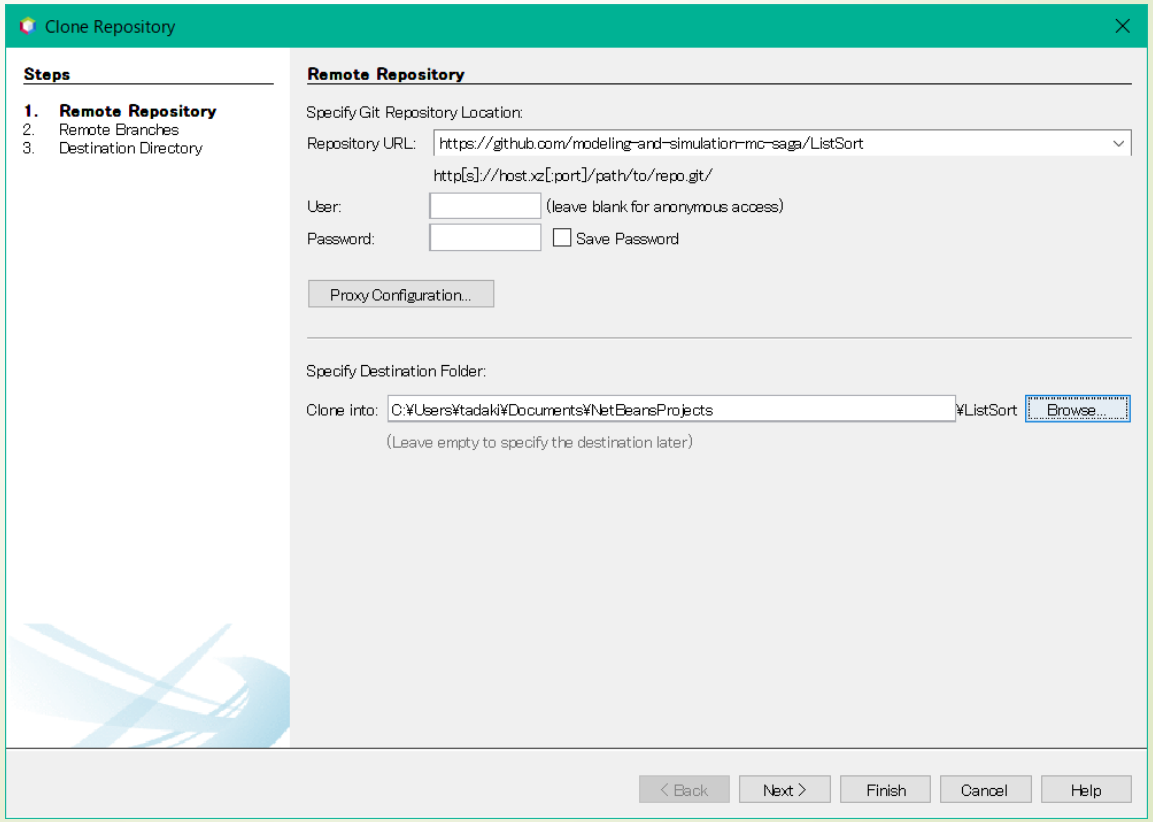
Merge Sort
リストの結合



- 簡単のため、リスト長を$n = 2^m$とする。
- 分割の回数は$\log_2 n = m$
- 各階層でのリストへの追加はn回。
- 分割時の工数：$n \log_2 n$

サンプルプログラムの取得

- “Teams”->”Clone”



- 二つのリポジトリからサンプルコードを取得
  - https://github.com/modeling-and-simulation-mc-saga/ListSort
  - https://github.com/modeling-and-simulation-mc-saga/MyLib

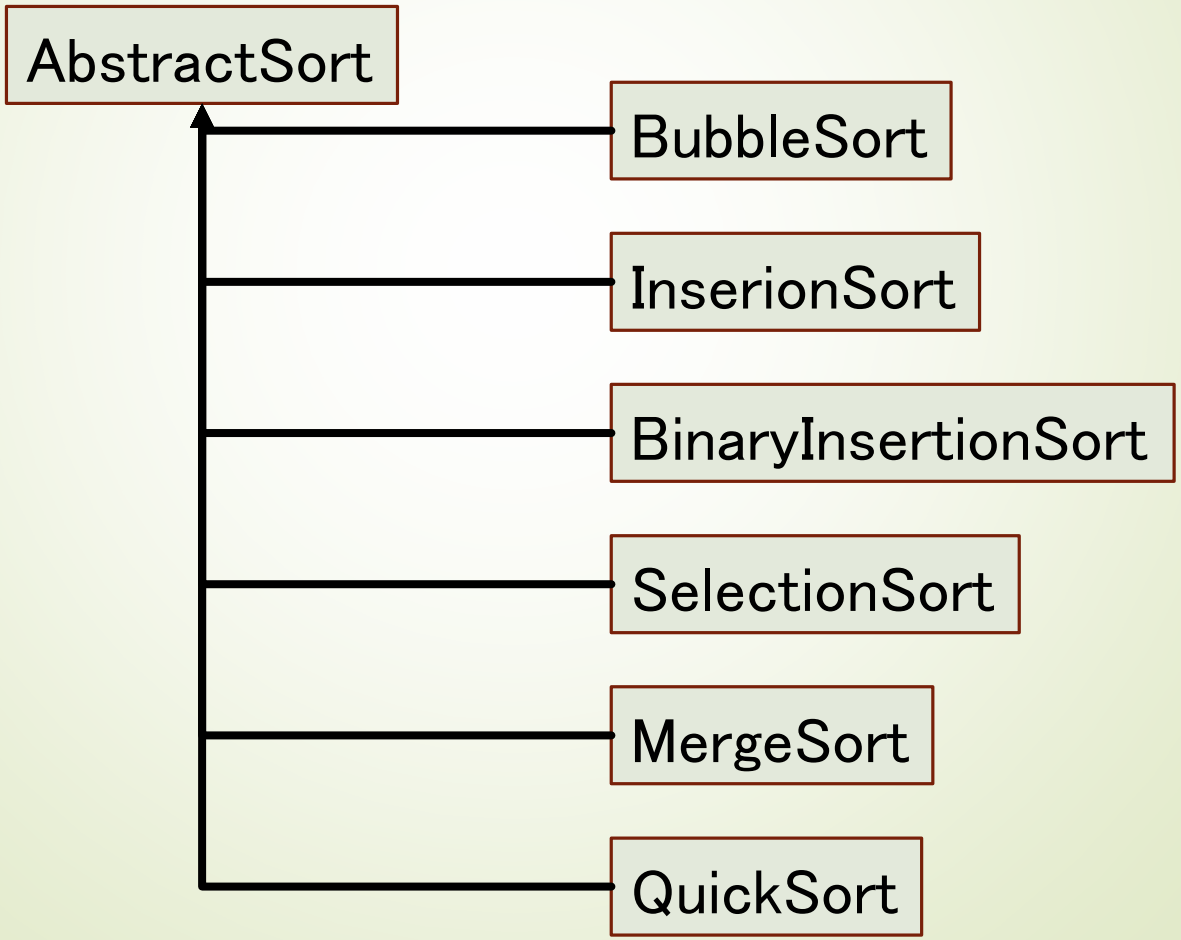
Object Oriented Programming

- 例として整列プログラムを考える
- 何ができるべきか
  - 整列対象は、要素間の大小関係が分かれば十分
  - 整列作業に最小限必要な機能
    - 比較、入れ替え

Object Oriented Programming

- 整列対象
  - 大小関係が定義できれば良い
    - Comparableを利用
    - 他はどうなっていても関係無い
  - public class Data implements Comparable<Data>
    - compareTo()を実装

- 整列操作
  - public abstract class AbstractSort<T extends Comparable<T>>
  - 具体的な整列法は、すべてAbstractSortのインスタンスに見える
  - 整列操作に共通的なメソッド
  - 比較回数を計測できる


AbstractSort
BubbleSort
InserionSort
BinaryInsertionSort
SelectionSort
MergeSort
QuickSort

シミュレーション

- 要素数をnとすると
  - 泡立ち法などは、比較がn^2回
  - QuickSortなどは、比較が$n \log n$
- 実際に計測