Merge Sort

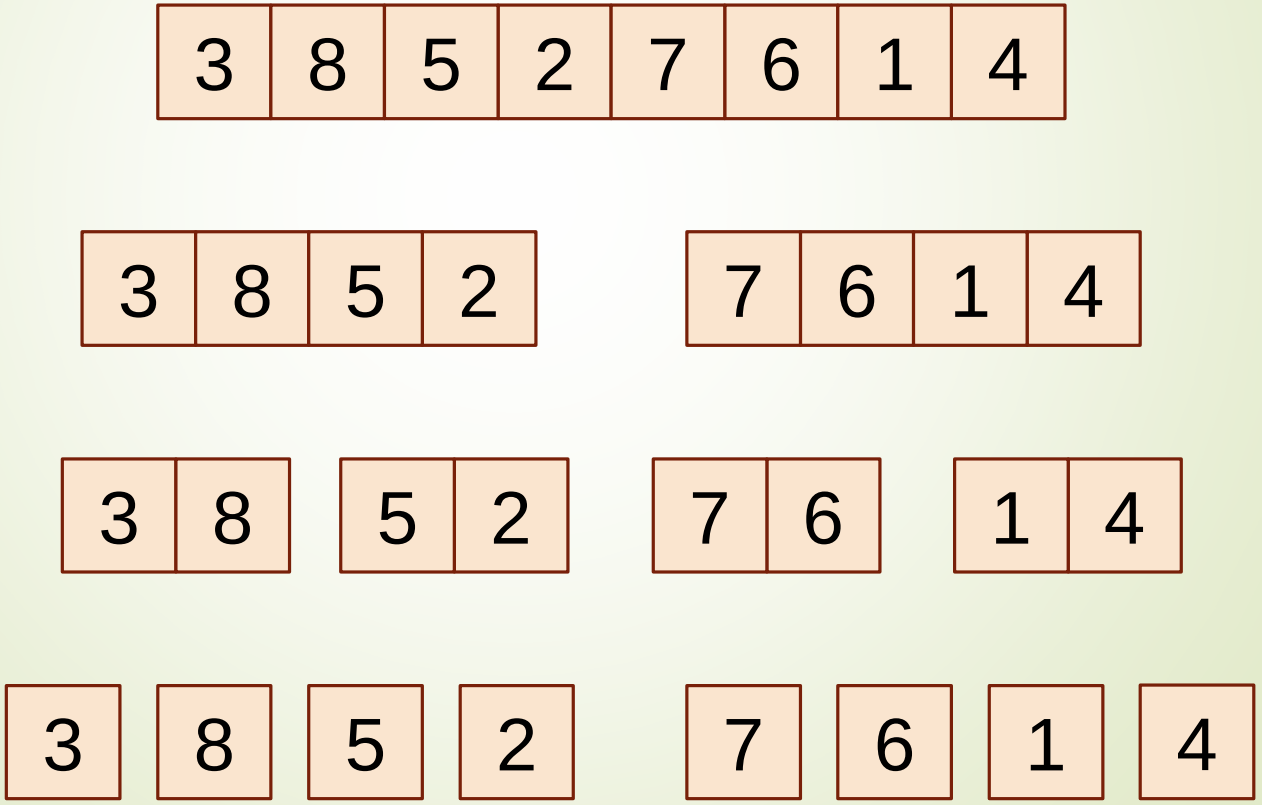
計算機アルゴリズム特論：2015年度

只木進一

Merge Sortの基本的考え方

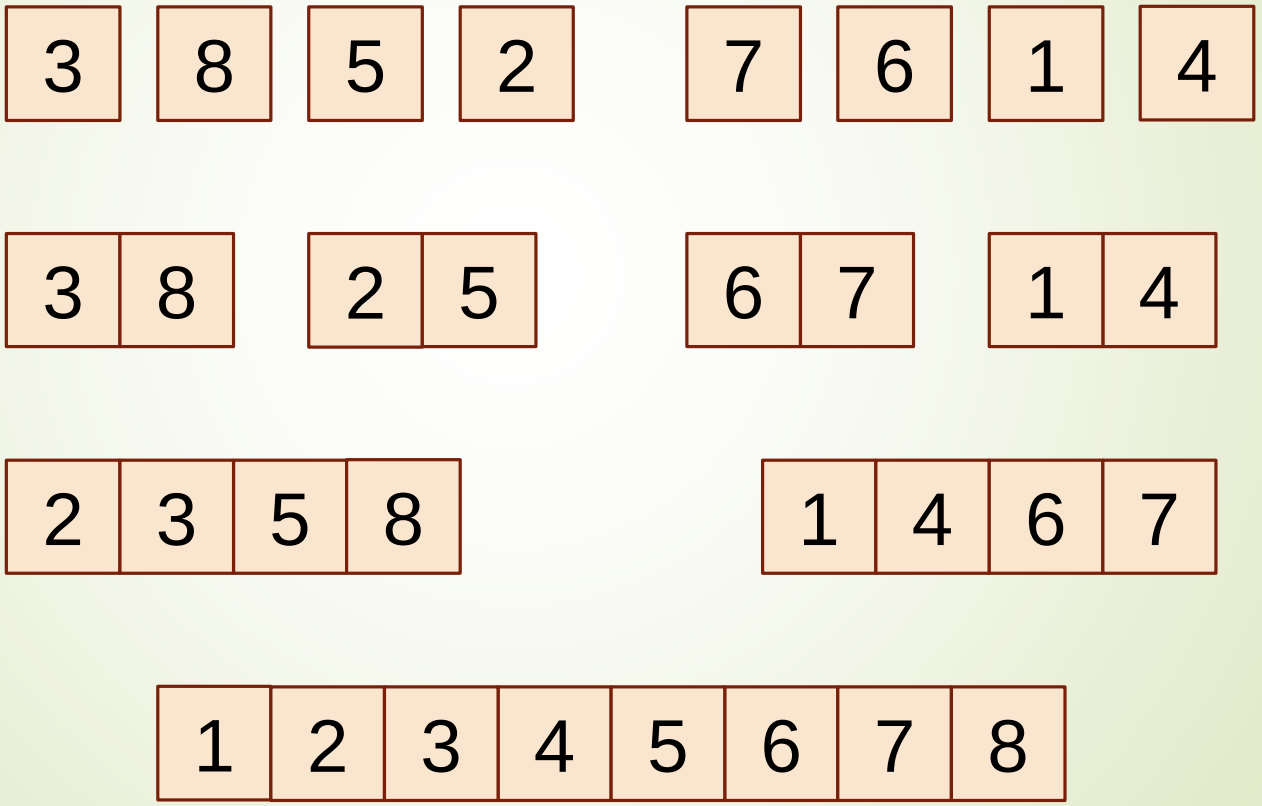
- 規模の小さな問題は容易に解ける
  - 長さ1のリストは、sort不要
- すでに解かれた部分問題から、解を得るのは容易な場合がある
  - sort済みの二つのリストから、sortされた一つのリストを作るのは容易

Merge Sort
リストの分離



- 簡単のため、リスト長を$n = 2^{\mathrm{m}}$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：$n \log_2 n$

Merge Sort
リストの結合



- 結合の回数は$\log_2 n = m$
- 各階層でのリストへの追加はn回
- 結合時の工数：$n \log_2 n$

Merge Sortアルゴリズム 再帰での記述

```
List<T> mergeSort(L<T> list){
  if(list.size()==1){return list;}
  L<T> l1 = list の前半;
  L<T> l2 = list の後半;

  List<T> l1out = mergeSort(l1);
  List<T> l2out = mergesort(l2);
  return mergeList(l1out,l2out);
}
```

二つの整列済みリストの結合

```
private List<T> mergeList(List<T> a, List<T> b) {
      List<T> c = Collections.synchronizedList(
            new ArrayList<T>());
      while ((!a.isEmpty()) || (!b.isEmpty())) {
         T t;
         if (!a.isEmpty()) {
            if (!b.isEmpty()) {
               if (a.get(0).compareTo(b.get(0)) < 0) {
                  t = a.remove(0);
               } else {t = b.remove(0); }
            } else {t = a.remove(0); }
         } else {t = b.remove(0); }
         c.add(t);
      }
}
```

AbstractSort.java

```java
package sort;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Sortの抽象クラス
 * @author tadaki
 * @param <T>
 */
public abstract class AbstractSort<T extends Comparable<T>> {

    protected int numStep;
    protected List<T> list;

    /**
     * Sortを実行する抽象メソッド
     * @return
     */
    abstract public List<T> sort();

    /**
     * コンストラクタ
     * 配列でのデータ入力をリストとして保存
     * @param d
     */
    public AbstractSort(T[] d) {
        list = Collections.synchronizedList(new ArrayList<>());
        for (int i = 0; i < d.length; i++) {
            list.add(d[i]);
        }
    }

    /**
     * 処理回数を返す
     * @return
     */
    public int getNumStep() {
        return numStep;
    }

    /**
     * 結果の表示
     * @param <T>
     * @param l
```

AbstractSort.java

```
     */
    static public <T> void printList(List<T> l){
        for(int i=0;i<l.size();i++){
            System.out.print(l.get(i).toString());
            System.out.print(" ");
        }
        System.out.println();
    }
}
```

MergeSort.java

```java
package sort;

import java.util.List;
import myLib.utils.Utils;

/**
 * Merge Sort
 *
 * @author tadaki
 * @param <T>
 */
public class MergeSort<T extends Comparable<T>> extends AbstractSort<T> {

    public MergeSort(T[] d) {
        super(d);
    }

    @Override
    public List<T> sort() {
        list = sortSub(list);
        return list;
    }

    /**
     * 処理の再帰呼び出し
     *
     * @param tList
     * @return
     */
    private List<T> sortSub(List<T> tList) {
        int kk = tList.size();
        if (kk == 1) {//長さが1の場合には、そのまま返す
            return tList;
        }
        //リストを分割
        int k = kk / 2;
        List<T> aIn = Utils.createList();
        for (int i = 0; i < k; i++) {
            aIn.add(tList.get(i));
        }
        List<T> bIn = Utils.createList();
        for (int i = k; i < kk; i++) {
            bIn.add(tList.get(i));
        }
        //分割したリストを使って、再帰呼び出し
        List<T> a = sortSub(aIn);
```

```
        List<T> b = sortSub(bIn);
        //二つのリストをmerge
        return mergeList(a, b);
    }

    /**
     * 二つのリストのmerge
     *
     * @param a
     * @param b
     * @return
     */
    private List<T> mergeList(List<T> a, List<T> b) {
        List<T> c = Utils.createList();
        while ((!a.isEmpty()) || (!b.isEmpty())) {
            T t;
            if (!a.isEmpty()) {
                if (!b.isEmpty()) {
                    if (a.get(0).compareTo(b.get(0)) < 0) {
                        t = a.remove(0);
                    } else {
                        t = b.remove(0);
                    }
                } else {
                    t = a.remove(0);
                }
            } else {
                t = b.remove(0);
            }
            c.add(t);
        }
        return c;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Integer data[] = {3, 6, 2, 9, 1, 6, 2, 8};
        MergeSort<Integer> sort = new MergeSort<>(data);
        List<Integer> list = sort.sort();
        AbstractSort.printList(list);

    }

}
```