

基本的データ構造 と操作：2

計算機アルゴリズム特論：2015年度
只木進一



探索アルゴリズム

Search Algorithms

■ 問題設定

- あるクラス T には、大小関係、等号関係が定義されている
- リスト L には、 T の要素が小さい順に格納されている
- T のインスタンス t と等号の成り立つ要素を L から探す

逐次探索

Sequential Search

- 先頭から順に探索する
 - リストが長くなると比例して時間がかかる

```
T target;  
List<T> list;  
for(T t:list){  
    if (t.compareTo(target)==0){  
        return;  
    }  
}
```

二分探索

Binary Search

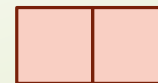
ここと比較




対象が前半と後半
のいずれに含まれ
るかを判断する。



リストが長く
なると、その
対数で時間が
かかる





```
List<T> list;
T target;
while (!list.isEmpty()) {
    int k = list.size() / 2;
    if (list.get(k).compareTo(t) == 0) {
        return true;
    }
    if (list.get(k).compareTo(t) > 0) {
        list = list.subList(0, k);
    } else {
        int kk = list.size();
        list = list.subList(k, kk);
    }
}
```

整列アルゴリズム

Bubble Sort

3	8	5	2	7	6	1	4
---	---	---	---	---	---	---	---

i番目とi+1番目を比較して正順に

3	5	2	7	6	1	4	8
---	---	---	---	---	---	---	---


3	2	5	6	1	4	7	8
---	---	---	---	---	---	---	---

2	3	5	1	4	6	7	8
---	---	---	---	---	---	---	---

2	3	1	4	5	6	7	8
---	---	---	---	---	---	---	---

つづく



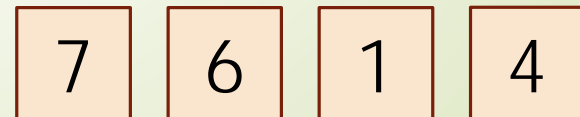
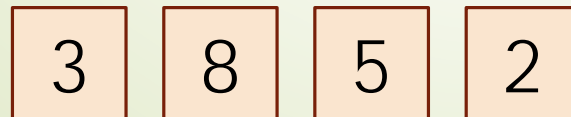
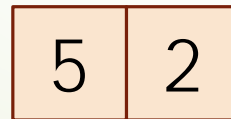
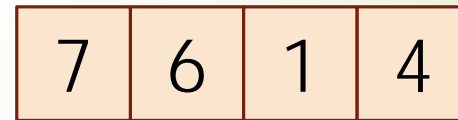
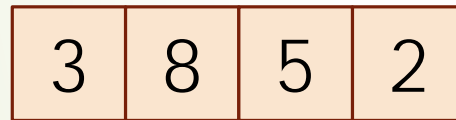
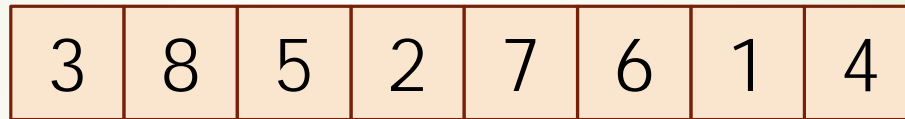
- 
- リストの長さを n とする
 - 一回目の比較は $n - 1$ 回
 - 二回目の比較は $n - 2$ 回
 - ...

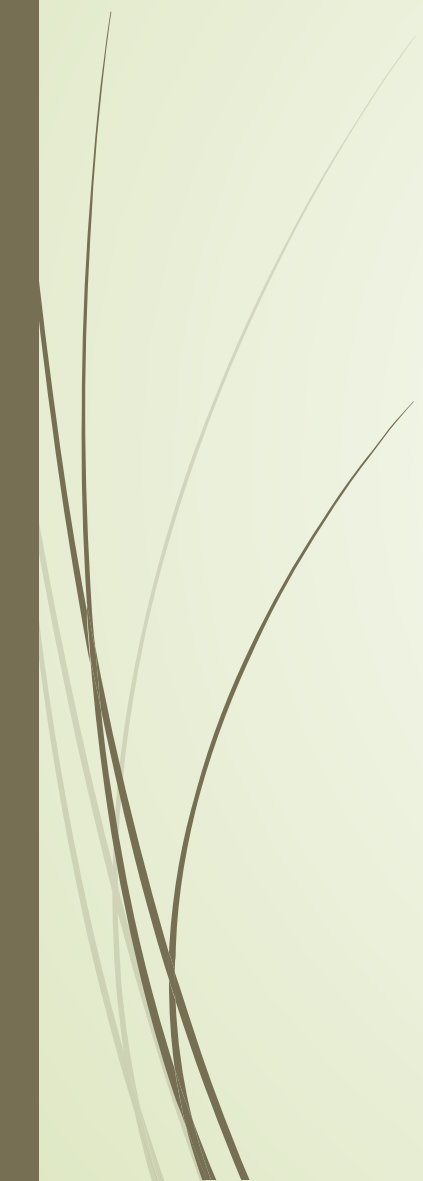

$$\sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

整列アルゴリズム

Merge Sort

リストを分離





3	8	5	2
---	---	---	---

7	6	1	4
---	---	---	---


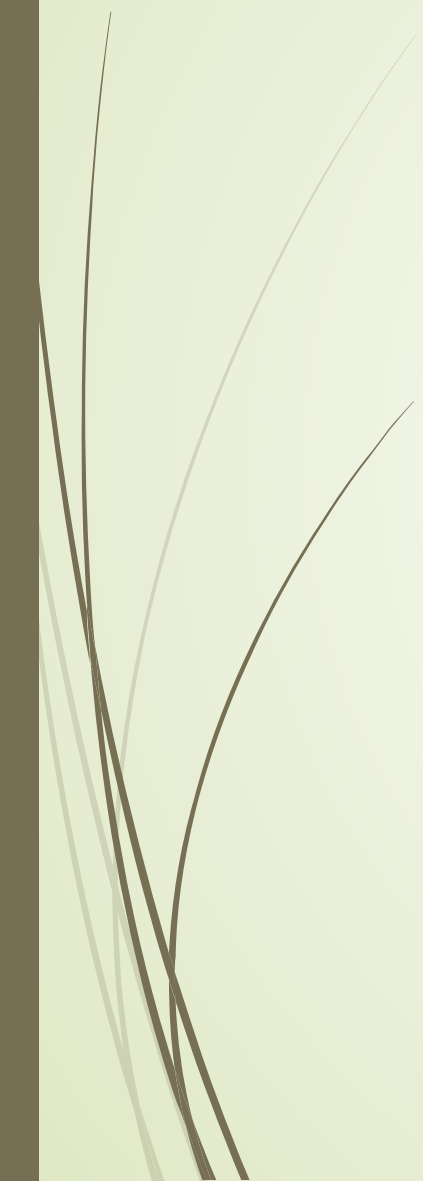
3	8	2	5
---	---	---	---

6	7	1	4
---	---	---	---

2	3	5	8
---	---	---	---

1	4	6	7
---	---	---	---

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

- 
- 
- アルゴリズムは
 - 正しくなければならない
 - 効率を考えなければならない
 - しかし、トリッキーなアルゴリズムは保守性が下がる
 - ライブラリ化の必要性