

基本的データ構造 と操作

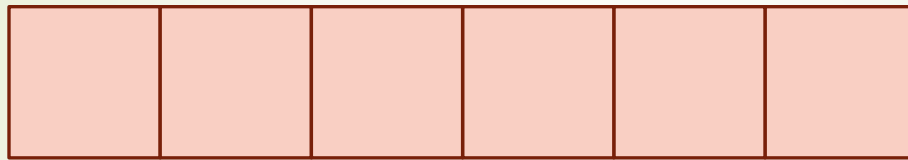
計算機アルゴリズム特論：2017年度
只木進一



この講義の目的

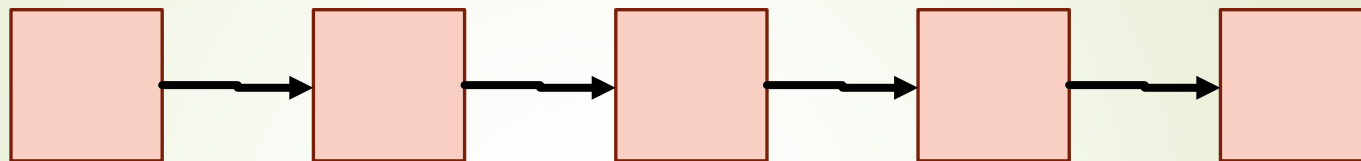
- データ構造とアルゴリズムの復習
 - 実装し、操作できる
- 計算量を評価する
- 複雑な問題のアルゴリズムとその計算量を評価する
- アルゴリズムの類型を知る
- 計算量が大きすぎる問題の近似的解法を理解する

基本的データ構造：配列 (arrays)




メモリ上に連続配置

基本的データ構造：リスト (lists)



データを「ポインタ」で結ぶ
Javaでは、Iteratorを使って順に手繰る




配列の利点と欠点

■ 利点

- ベクトル計算機などで高速計算可能

■ 欠点

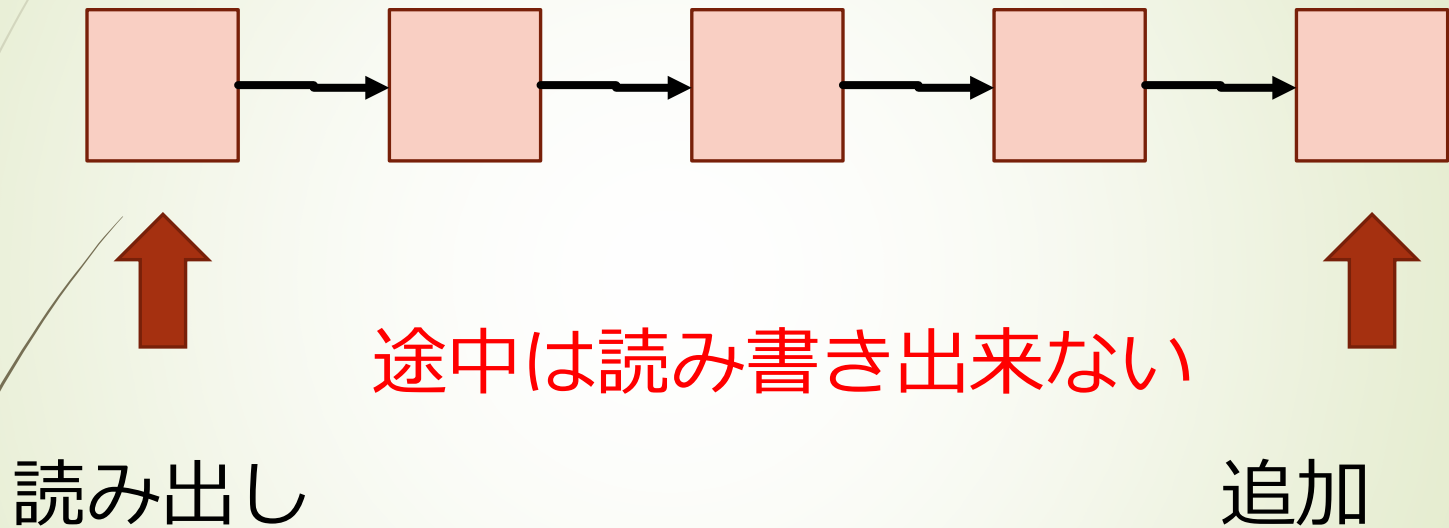
- サイズ変更が困難
- 途中のデータの抜き差しが困難
- クラステンプレートを使って配列を生成できない



配列とリスト

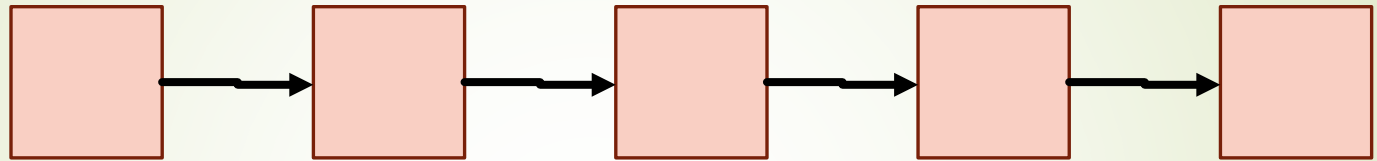
- 配列が向く場合
 - 大きさが固定
 - 内容の順番変更が無い
- リストが向く場合
 - 大きさが可変
 - 途中の要素の削除や途中への挿入が発生
 - クラステンプレートを使って配列を生成

リストに対する制約：待ち行列 (queues)



FIFO (First In First Out)

リストの制約：スタック (stacks)



読み出し
追加

途中は読み書き出来ない

FILO (First In Last Out)



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

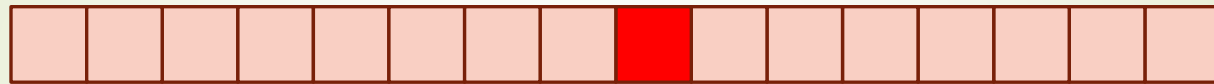
- 先頭から順に探索する
 - リストが長くなると比例して時間がかかる

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

二分探索

Binary Search

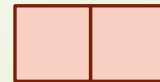
ここと比較




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
分割最大数は $\log_2 n$





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

アルゴリズムイメージ

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
 - ...

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



課題

- bubble sortを実装し、動作を確認しなさい。