計算量

Computational Complexity

計算機アルゴリズム特論：2017年度

只木進一

計算量

- アルゴリズムの良さをはかる尺度
  - 計算時間、記憶領域、理解しやすさ、プログラムへの変換の容易さ、等々
- 時間計算量(Time Complexity)
  - 計算に要する時間
  - 講義ではこちらを主に議論
- 領域計算量(Space Complexity)
  - 計算に要する記憶領域

計算量

- 稼働させる計算機に依存
  - CPU、メモリ、ディスクに依存
  - 仮想的計算機で比較
- 入力サイズに依存
  - 入力データのサイズnの関数として評価
  - 漸近的計算量に着目：n→大のときの挙動

- 入力サイズが同じでも差
  - 探索データがリストの先頭なら速いが、末尾なら遅い
  - 平均値、最悪値（最も悪い場合の値）で議論
- どの部分に注目するか
  - 律速過程を見極める

例：多項式の計算

$$f(x) = a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

- 計算手順で演算回数が異なる
- 係数a_iと変数xを与えて、関数の値$f(x)$を求めるのに必要な計算
  - xのべきを求めるための乗算4回
  - 係数とxのべきとの乗算4回
  - 加算4回
  - 合計12回

```
double polynomial(double x, double a[]){
   int n=a.length;
   double v=a[0];
   double xx=x.:
    for ( int i=1; i<n; i++){
      v += xx * a[i];
      xx *= x;
    }
   return v;
}
```

- Hornerの方法

$$f(x) = \left(\left(\left(\left(a_4 x + a_3\right) x + a_2\right) x + a_1\right) x + a_0\right)$$

- 乗算4回＋加算4回

```
double polynomial(double x, double a[]){
    int n=a.length;
    double v=a[n-1];
    for ( int i=1; i<n; i++){
        v += v*x + a[i];
    }
    return v;
}
```

例：多項式の計算：一般化

- 律儀な方法
  - n次の項： 2n回の乗算
  - n回の加算

$$P(n)=\sum_{k=0}^{n}k+2n=\frac{n(n+1)}{2}+2n=\frac{n(n+5)}{2}$$

- Hornerの方法
  - 一つの括弧を閉じる：乗算と加算が各1回

$$P(n) = \sum_{k=0}^{n} 2 = 2n$$

例：逐次探索

10	15	30	37	50	70	81	91

- リストの長さN
- 要素中の一つを探索
  - 位置iにある場合
    - 存在確率$1/N$
    - 比較回数$i+1$

$$\frac{1}{N}\sum_{k=1}^{N} k = \frac{N+1}{2}$$

計算量の評価

- データのサイズnが大きいときに、計算量や計算時間が nに対してどのように増大するか
  - 定数倍にはあまり関心はない
    - 周辺の演算の影響
  - nに対して最も速く増大する項に関心がある
    - nが大きくなった時に最も重要

オーダー記法

- 二つの関数$f(n)$と$g(n)$を考える

$$0 \leq \lim_{n \to \infty} \left| \frac{f(n)}{g(n)} \right| \leq c$$

このとき

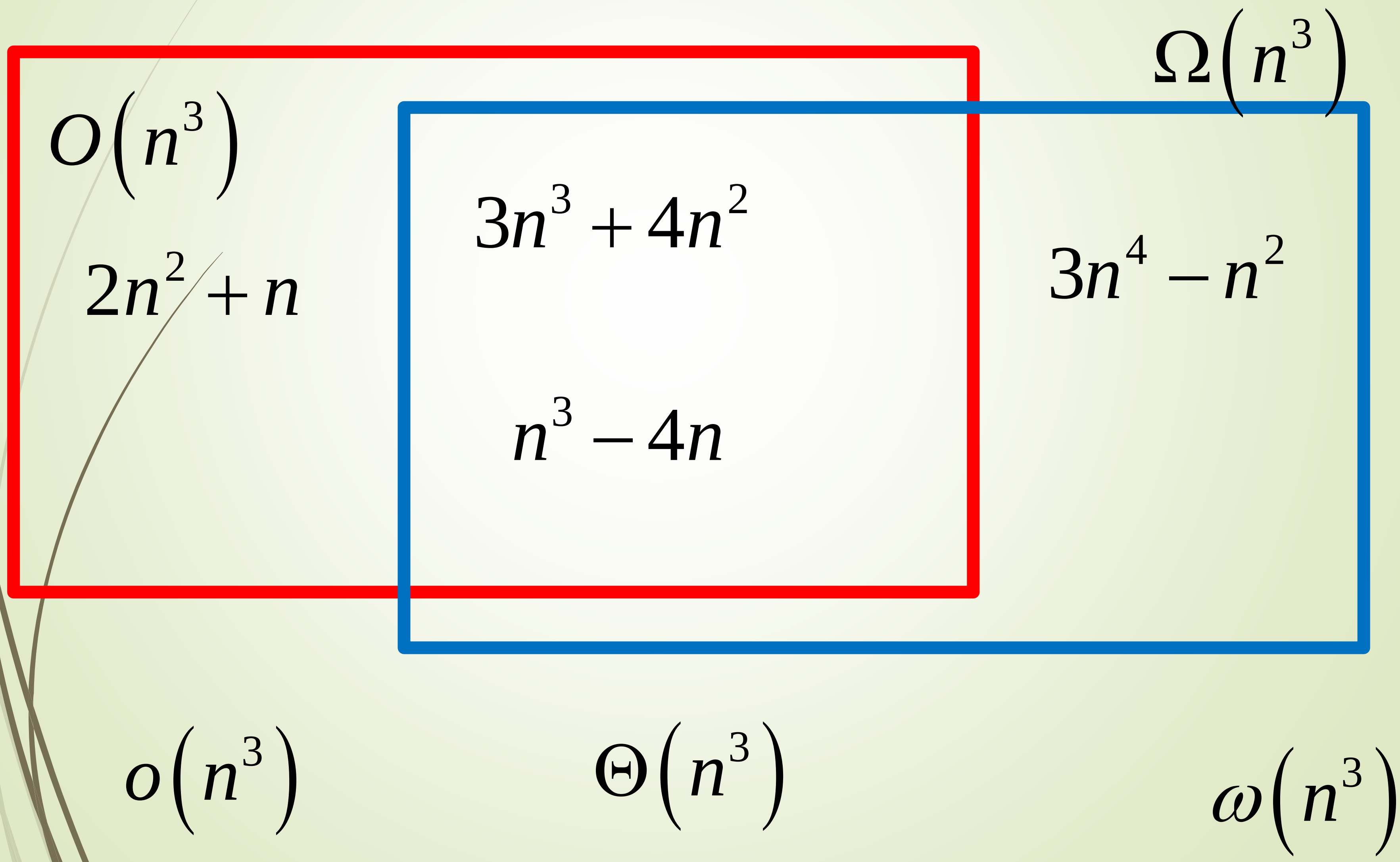
$$f(n) \in O(g(n))$$

- nが大きいとき、$f(n)$は$g(n)$より増加が速くない
  - 通常は、$g(n)$は簡単な関数形

他のオーダー記法

記法	定義
$f(n) \in O(g(n))$	$\lim_{n\to\infty} \left\lvert \frac{f(n)}{g(n)} \right\rvert \leq c$
$f(n) \in o(g(n))$	$\lim_{n\to\infty} \left\lvert \frac{f(n)}{g(n)} \right\rvert = 0$
$f(n) \in \Omega(g(n))$	$\lim_{n\to\infty} \left\lvert \frac{f(n)}{g(n)} \right\rvert \geq c$
$f(n) \in \omega(g(n))$	$\lim_{n\to\infty} \left\lvert \frac{f(n)}{g(n)} \right\rvert = \infty$
$f(n) \in \Theta(g(n))$	$f(n) = O\big(g(n)\big) \wedge$ $f(n) = \Omega(g(n))$

オーダーの関係



$\Omega\left(n^3\right)$
$O\left(n^3\right)$
$3n^3+4n^2$
$2n^2+n$
$3n^4-n^2$
n^3-4n
$o\left(n^3\right)$
$\Theta\left(n^3\right)$
$\omega\left(n^3\right)$

例：多項式

- 律儀な方法：
- ：$O(n)$

Insertion Sort
基本的考え方

- 大きさ10の配列の先頭5個は整列済みとする

B	E	H	M	T	D	K	X	R	L

- 6番目の要素”D”を適切な位置に挿入する
  - 順に左に移動し、より小さい要素の手前に挿入

B	D	E	H	M	T	K	X	R	L

Insertion Sort
基本的考え方:詳細

- 配列d
- 0番から$i-1$番までは整列済み
- $j=i$からjを$d_j<d_{j-1}$である限り一つずつ下げ、d_jとd_{j-1}を入れ替える

Insertion Sort
アルゴリズム

for ($i = 1; i < n; i++$) {

  for ($j = i; j > 0$ && $d\_j < d\_\{j-1\}$; $j--$){

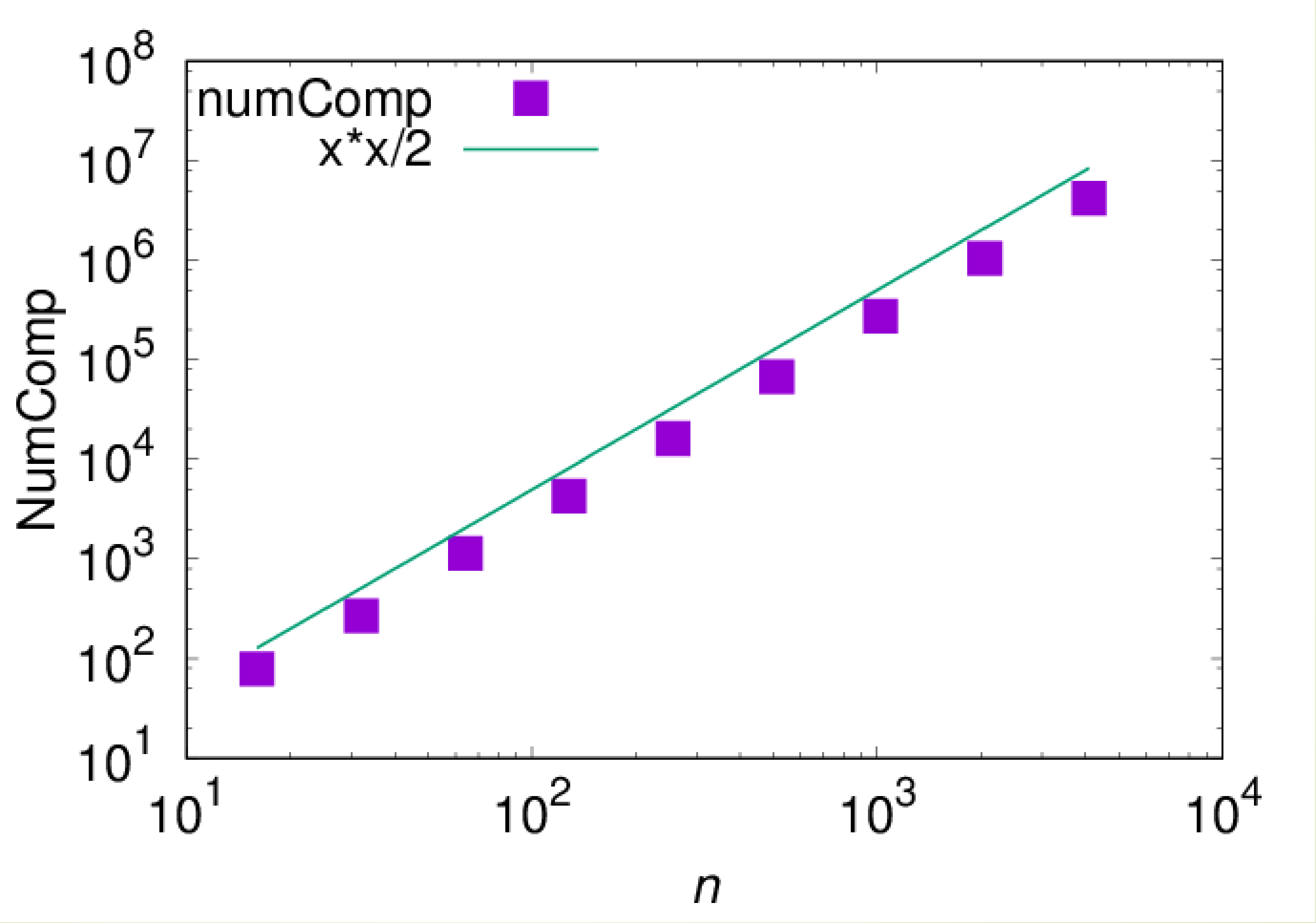
    exch ($j, j-1$)

  }

}

- 内側のループの要素入替回数（最大）
  - 1回目：1
  - 2回目：2
  - 最後：$n-1$
- 要素入替回数（最大） $O(n^2)$

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



numComp
x*x/2
NumComp
n
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8