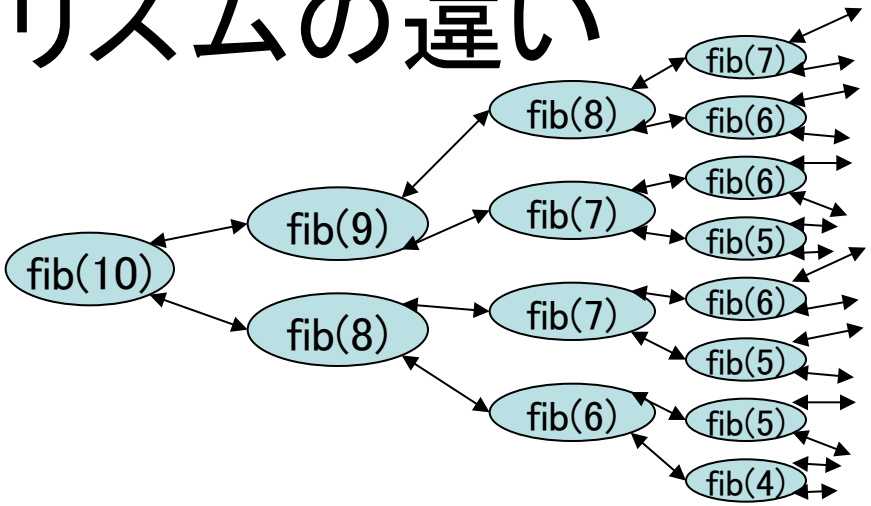
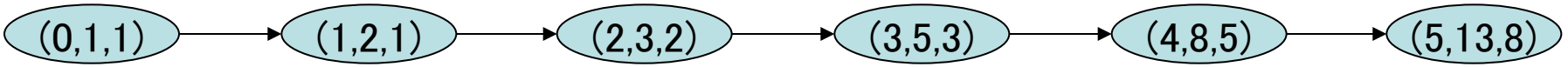
アルゴリズムと計算量

フィボナッチ数の計算: アルゴリズムの違い

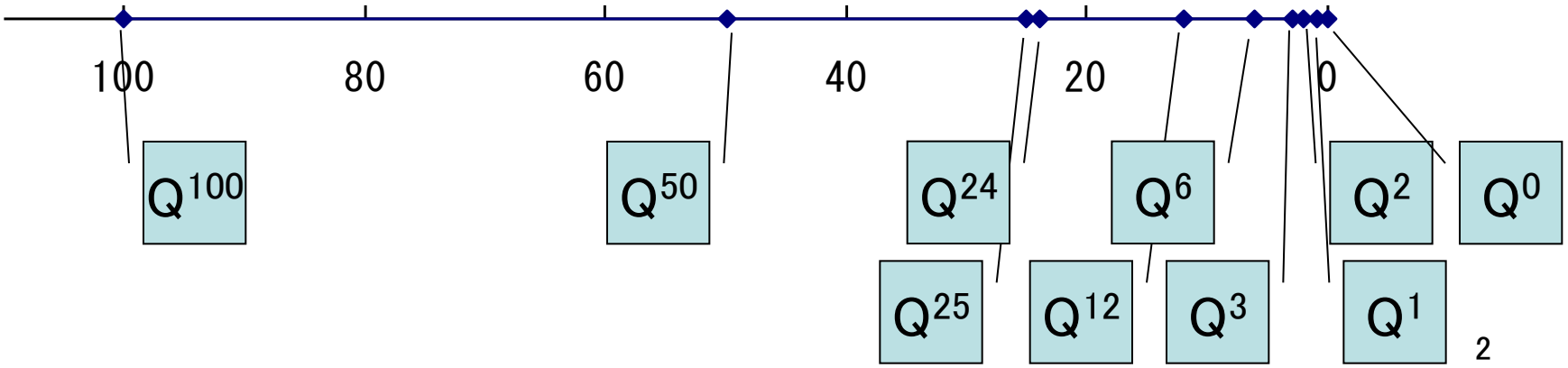
- 定義通り



- 数え上げ

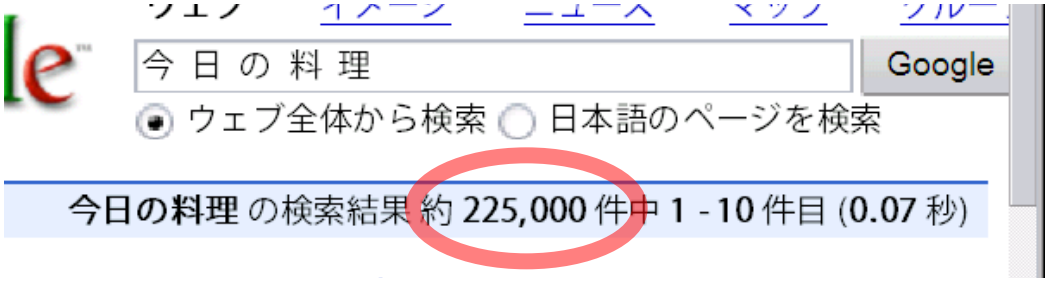


- 行列を使う



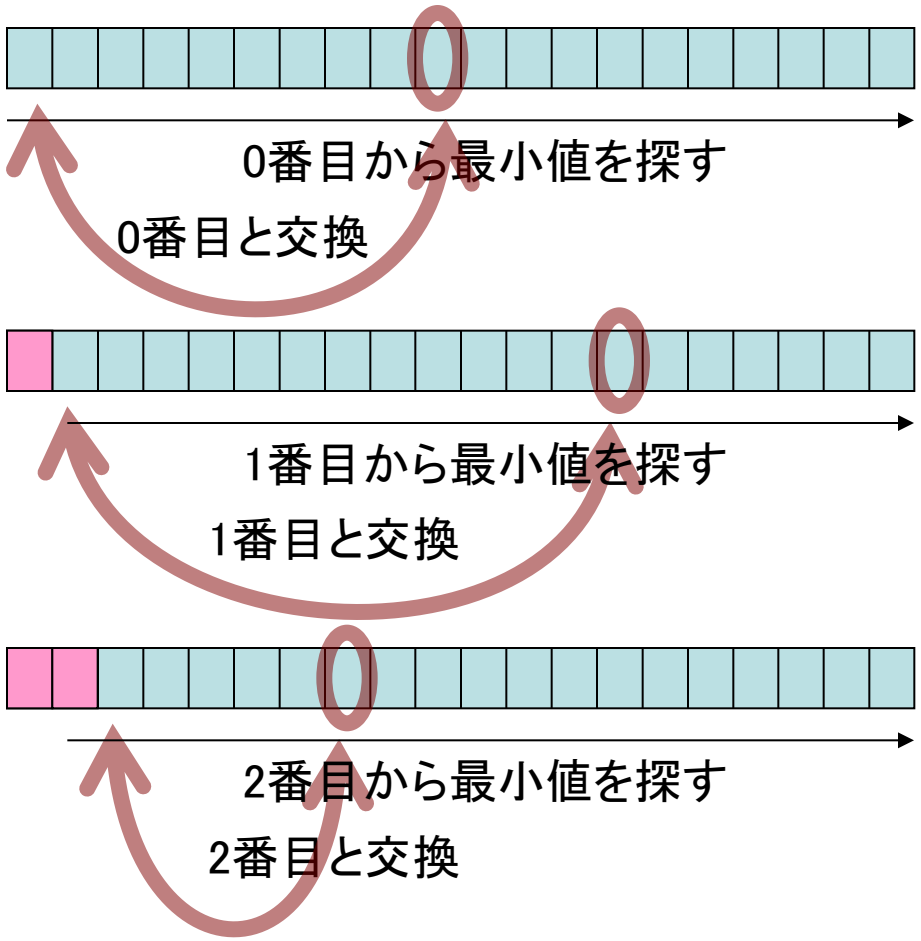
整列（sorting）

- 問題: n 個のデータを大きさ順に並べる
  (単純化): n 個の整数を小さい順に並べる
  - 色々な場面で使われる
  - データの「下ごしらえ」としても使われる
    例: 検索を高速化するために整列しておく
- データ数は、時として非常に大きくなる
  - 「全学生の氏名を学生証番号順に表示」
  - 「『今日の料理』という言葉を持つページを信頼度順に表示」



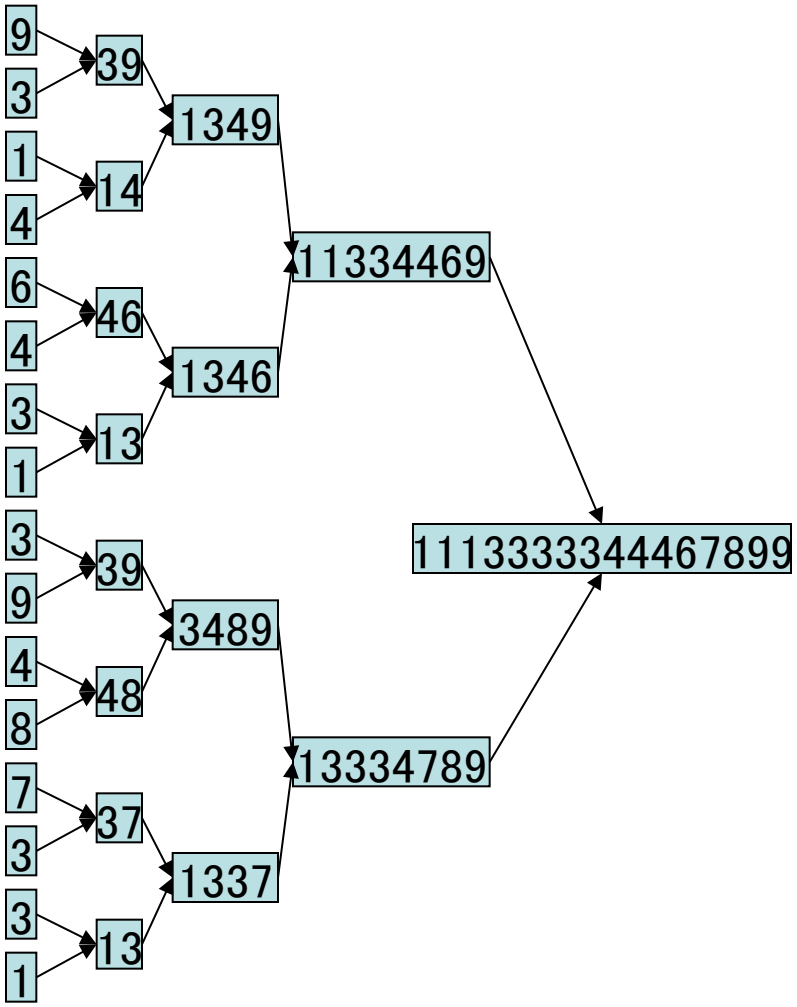
単純整列法

- 方法:
  - i=0,1,2,…について
    列の中でi番目に
    小さい数を見つけ、
    それをi番目へ移動
  - 「i番目に小さい数」は
    列のi番目以降の
    最小値



併合整列法

- 前提:
  - 整列済みの列が沢山ある
- 方法:
  - 2つの列を順序よくくっつけて1つにする(併合)
  - 列が1つになるまで繰り返す



課題: アルゴリズムによる速度差の実測

- 平方根・整列の複数のアルゴリズムに対応したプログラムを作り、速度の違いを調べる
  - 平方根の場合: x/δの大きさによって時間がどう変化するかをグラフを描いてみる。1秒間に何回計算できるかで比べよ。
  - 整列の場合: ランダムな列を作り、それを整列させてみよ。列の長さで時間がどう変化するかをグラフを描いてみる。
- グラフから、実行時間を予測する式を推定せよ

```
load("./randoms.rb ")    # randoms(id,size,max)
load"./bench.rb")        # run(function_name, x, v)
load("./simplesort.rb") # simplesort(a)
load("./mergesort.rb")  # mergesort(a)

def compare_sort(max, step)
  for i in 1..(max/step)
    x=i*step
    a=randoms(i,x,1)
    run("simplesort", x, a)
    a=randoms(i,x,1)
    run("mergesort", x, a)
  end
end
```

進捗状況の確認

1. Fibonacci数の計算と整列のすべての練習を実施した
2. Fibonacci数の計算と整列の半分くらいを実施した
3. Fibonacci数の計算だけ、すべてを実施した
4. Fibonacci数の計算の一部を実施した
5. まだ書いていない

進捗状況の確認

1. simplesort も mergesort も動いたら、投票してください。

   - 配列 a の（先頭を0 番目としたときの）i 番目以降で、最小値が出現する番号を返す min_index(a,i) を追加して、単純整列法を完成させなさい。
   - 省略部分を補って関数 merge を完成させなさい。
   - 終わったら compare_sort(5000, 250) を実行する。

投票したら、練習5.17に挑戦してみよう。

計算量

- これまで見てきたように、アルゴリズムによってプログラムの実行時間は大きく変わり得る
- プログラムを作らずに、良いアルゴリズムを選ぶにはどうすればよいか?

→**計算量**

  - アルゴリズムが答えを出すのに必要とする資源の見積り
  - 資源 = 計算回数や記憶容量
  - 通常はおおまかな式(オーダー, O)で見積り比較する

    ←それでも充分

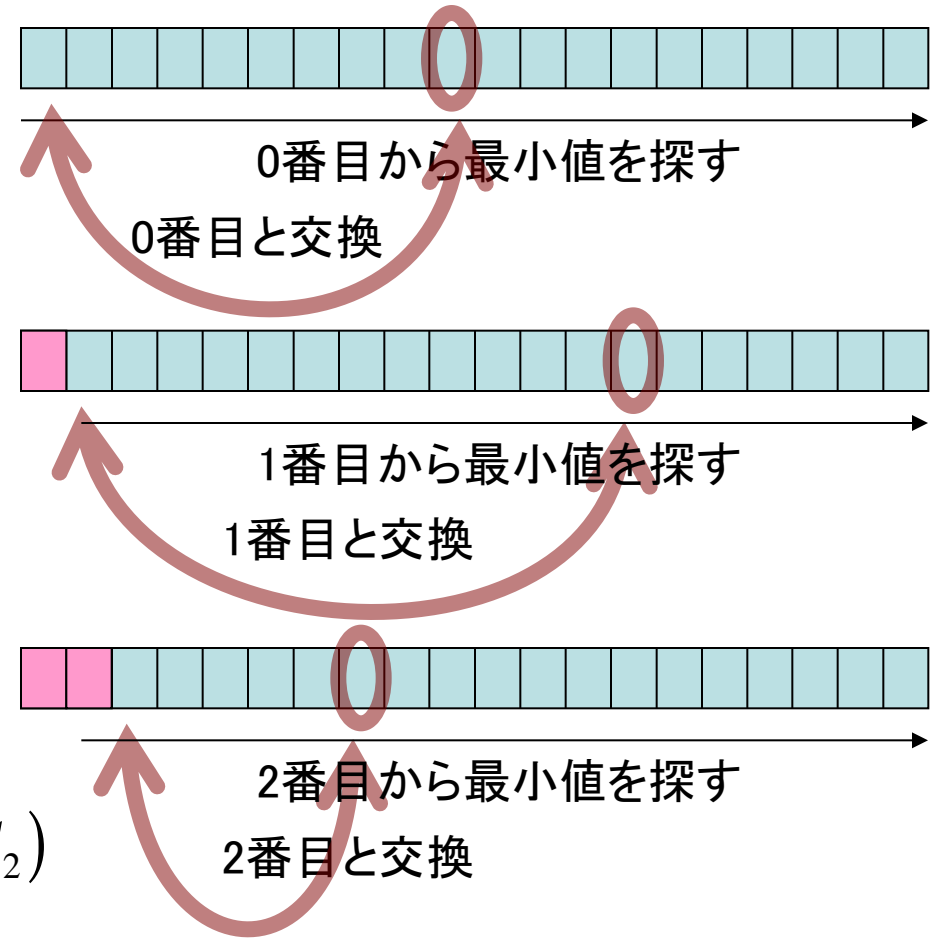
計算量の求め方

1. 各演算、関数呼び出し、判断、分岐をそれぞれ「1回の計算」とする
2. 入力の引数 x に対する計算回数の式を求める
3. 定数を消し、主要な項だけを残す（計算量のオーダー）

※3を前提にすると1, 2は厳密でなくともよくなる

単純整列法

- 計算回数:
  - ➢i=0のとき，n-1回比較
  - ➢i=1のとき，n-2回比較
  - ➢i=n-1のとき，0回比較



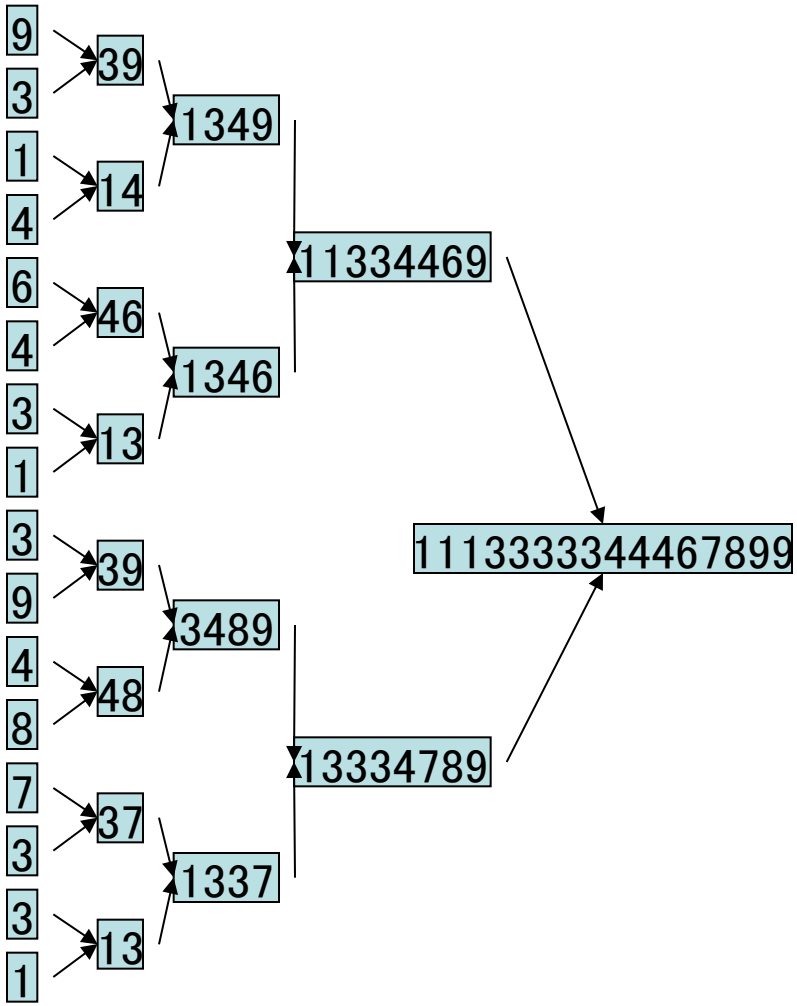
$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n \quad \left(={}_nC_2\right)$$

$$\mathbf{O}(n^2)$$

併合整列法

- 計算回数:
  - 第1段，約n回コピー(比較)
  - 第2段，約n回コピー(比較)
  - 第log n 段，
    約n回コピー(比較)

$$O(n \log n)$$



計算量を求める: 整列アルゴリズム

問題: k以下の整数の長さnの列を整列

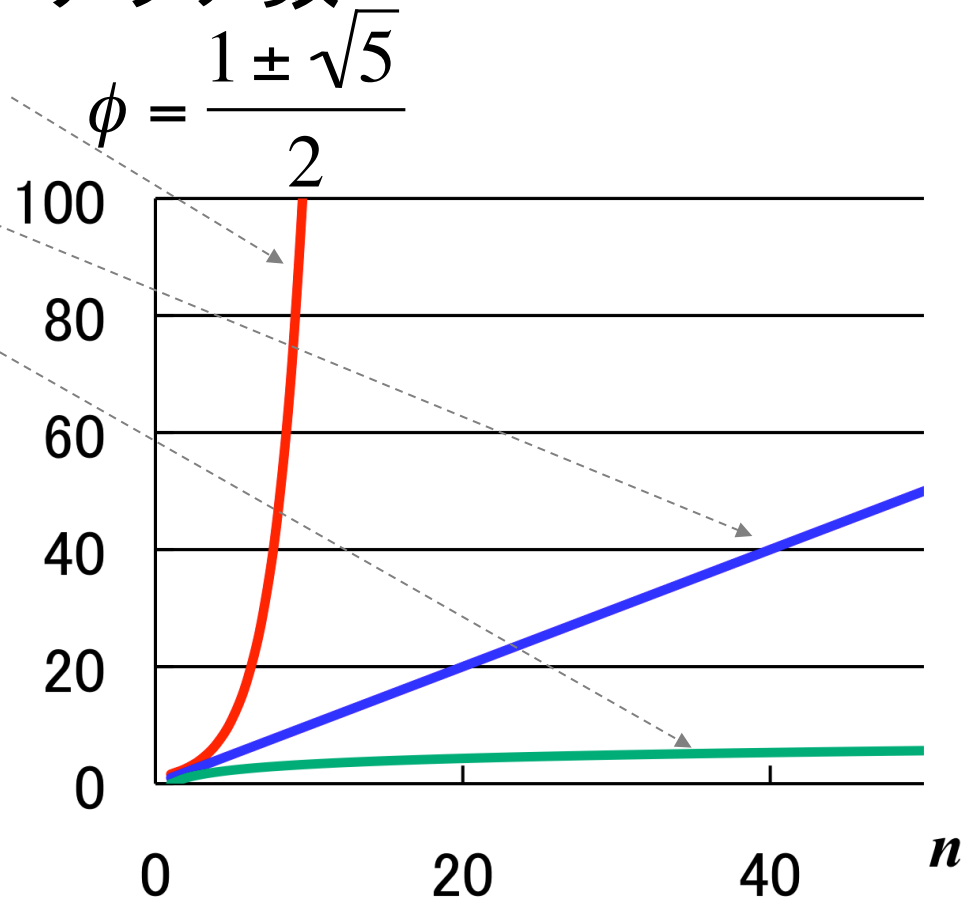
- 単純整列法 — $O(n^2)$
- 併合整列法 — $O(n \log n)$

n	10	100	1,000	10,000	100,000	1.0E+06	1.0E+07	1.0E+08	1.0E+09
n^2	100	10,000	1.0E+06	1.0E+08	1.0E+10	1.0E+12	1.0E+14	1.0E+16	1.0E+18
$n\log n$	33	664	9,966	132,877	1.7E+06	2.0E+07	2.3E+08	2.7E+09	3.0E+10

($k = 100$の場合)

計算量を求める: フィボナッチ数

- 問題: n番目のフィボナッチ数
  - 定義通り — $\mathbf{O}(\varphi^n)$
  - 数え上げ — $\mathbf{O}(n)$
  - 行列 — $\mathbf{O}(\log n)$



課題: 計算量と実測値の対応

- 紹介したアルゴリズムの計算量のオーダーを求めてみよ
- プログラムの実行時間の実測値と対応がとれているかを確認せよ
- 対応がとれない場合は、理由を考えよ

情報科学

数値計算

数値計算とは？

Mathematica を
実行してみよう

- 代数演算
  - 等式を一定の規則のもとに変形して解析的に求める
- 数値計算
  - 反復計算によって”近似的に”解を求める
  - 解析的に求めることが困難な方程式を解く場合に用いる

例） 物理学、化学、天文学などにおける数値積分や微分方程式の解法など

数値計算トピック

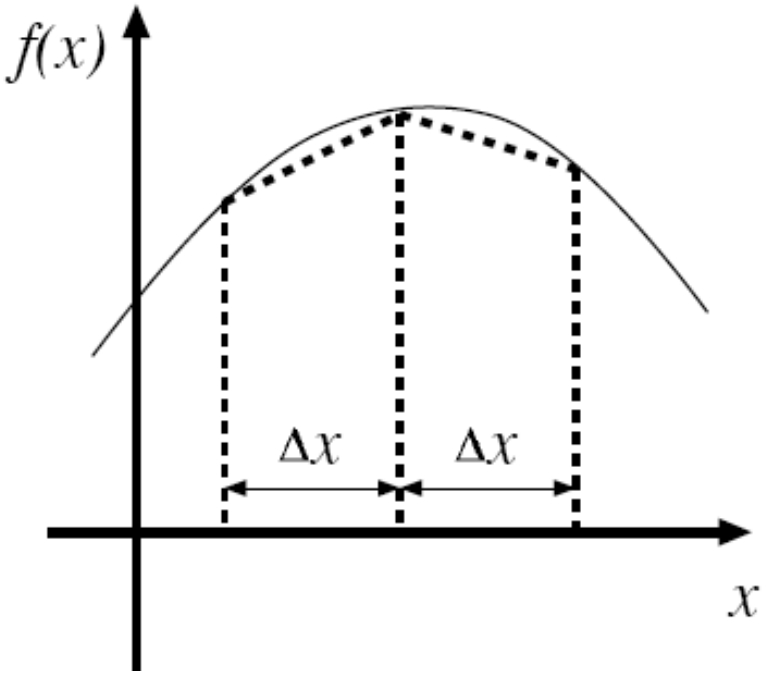
- 数値積分
  - モンテカルロ法を含む。
- 数値計算における誤差
- 連立方程式

数値積分

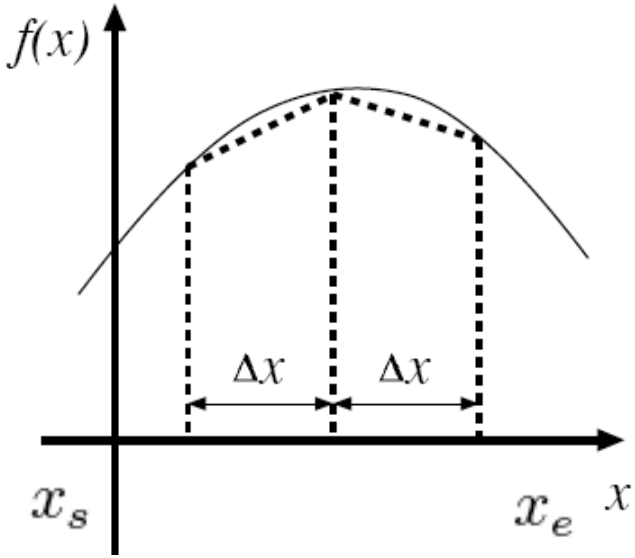
1. 台形公式
2. シンプソン公式
3. モンテカルロ法

台形公式

- 積分を区分線形(piecewise linear)で近似する方法
- 全積分区間をn等分し各部分区間 Δx
  の積分を台形の面積として計算し総和をとる



台形公式（続き）



ある関数 $f(x)$ の x_s から x_e までの積分の近似値を求めるには、

各部分区間 $\Delta x = \dfrac{x_e - x_s}{n}$

における台形の総和で近似する

$$\int_{x_s}^{x_e} f(x)dx \approx \sum_{i=0}^{n-1} \frac{1}{2}\{f(x_s + i\Delta x) + f(x_s + (i+1)\Delta x)\}\Delta x$$

$$= \Delta x \left\{ \frac{f(x_s) + f(x_e)}{2} + \sum_{i=1}^{n-1} f(x_s + i\Delta x) \right\}$$

```
def f(x)
    x/((x+1.0)*(x+2.0))
end

def trapezoid(xs,xe,n)
    deltax = (xe-xs)*1.0/n
    sum = (f(xs)+f(xe))/2.0
    for i in 1..(n-1)
        sum = sum + f(xs+i*deltax)
    end
    deltax * sum
end
```

trapezoid.rb

練習

- 以下の積分を台形公式で近似して円周率を求めよ。分割回数nを引数として円周率を返す関数を定義せよ。

$$\pi = 4\int_0^1 \sqrt{1-x^2}\,dx$$

trapezoid.rb の f を定義しなおし、

```
def pi(n)
    4.0 * trapezoid(0.0, 1.0, n)
end
```

進捗状況の確認

1. π の計算ができた時点で投票してください。
2. 値がおかしい。
3. プログラムが動かない。

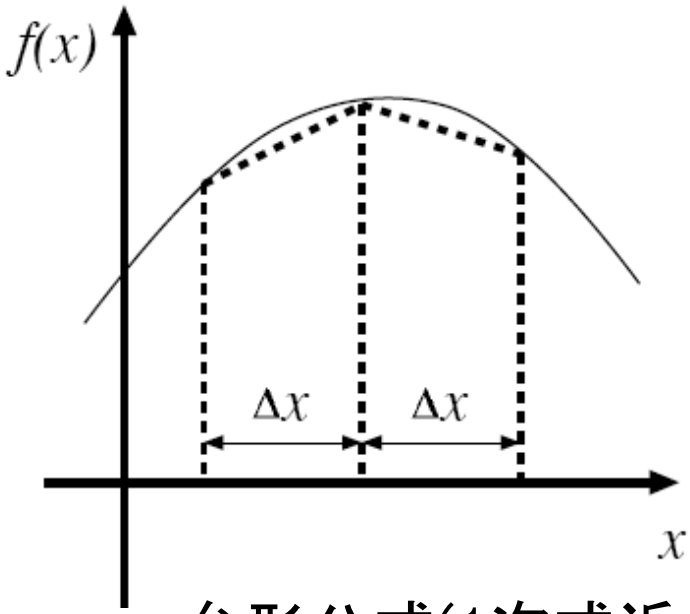
$$\pi = 4\int_0^1 \sqrt{1-x^2}\,dx$$

正しく動いたら、n の値を変えて、真の値に近づく様子を観察しよう。

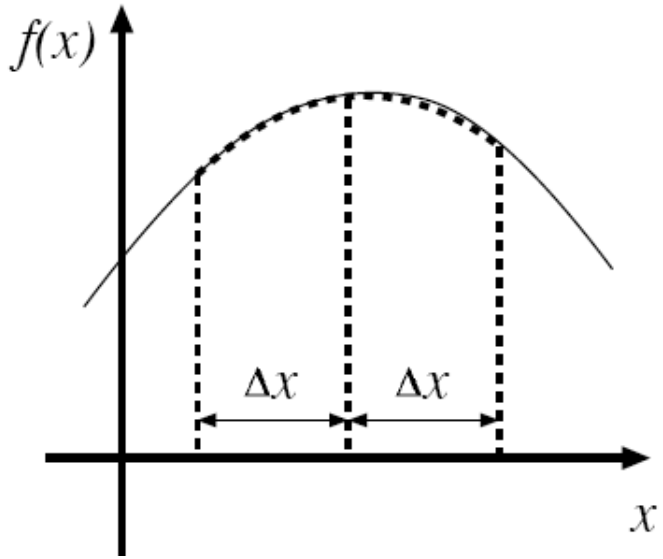
終わったら次のシンプソン公式（練習6.2）。

シンプソン公式

- 積分を1次式ではなく2次式で近似する方法



台形公式(1次式近似)



シンプソン公式(2次式近似)

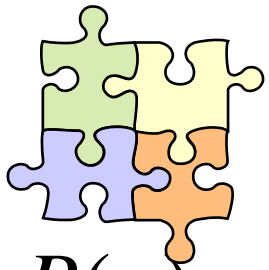
$$\frac{1}{2}\{f(x)+f(x+\Delta x)\}\Delta x$$

$$\Delta x = \frac{x_e - x_s}{n}$$

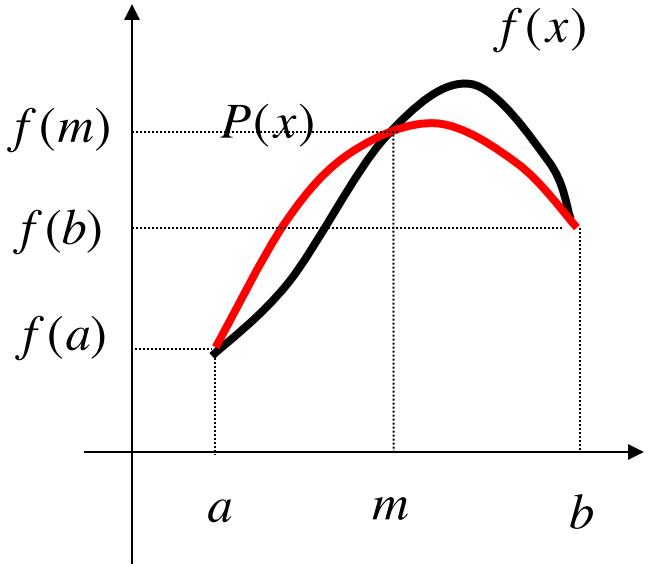
$$\frac{1}{3}\{f(x)+4f(x+\Delta x)+f(x+2\Delta x)\}\Delta x$$

$$\Delta x = \frac{x_e - x_s}{2n}$$

ラグランジュ補間



- ある関数$f(x)$を3点a,m,bで補間する2次関数$P(x)$



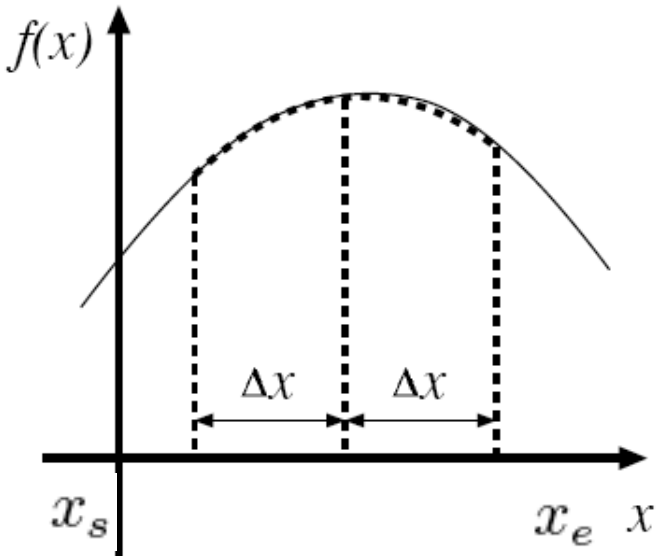
$$P(x)=\frac{(x-b)(x-m)}{(a-b)(a-m)}f(a)+\frac{(x-a)(x-b)}{(m-a)(m-b)}f(m)+\frac{(x-a)(x-m)}{(b-a)(b-m)}f(b)$$

$$m=\frac{(b+a)}{2} \Rightarrow \int_a^b P(x)dx=\frac{(b-a)}{6}[f(a)+4f(m)+f(b)]$$

$a \to x$
$b \to x+2\Delta x$
$m \to x+\Delta x$

$$\Rightarrow = \frac{1}{3}\{f(x)+4f(x+\Delta x)+f(x+2\Delta x)\}\Delta x$$

シンプソン公式（続き）



部分区間 $\Delta x = \dfrac{x_e - x_s}{2n}$ とし

区間 $2\Delta x$ の部分の積分近似値は

$$\frac{1}{3}\left\{f(x) + 4f(x+\Delta x) + f(x+2\Delta x)\right\}\Delta x$$

となるため、x_s から x_e までの

積分近似値の総和は

$$\begin{aligned}\int_{x_s}^{x_e} f(x)dx &\approx \sum_{i=0}^{n-1}\frac{1}{3}\left\{f(x_s+2i\Delta x) + 4f(x_s+(2i+1)\Delta x) + f(x_s+(2i+2)\Delta x)\right\}\Delta x \\ &= \frac{\Delta x}{3}\left\{f(x_s) + 4f(x_s+\Delta x) + f(x_e) + \sum_{i=1}^{n-1}\left(2f(x_s+2i\Delta x) + 4f(x_s+(2i+1)\Delta x)\right)\right\}\end{aligned}$$

練習

- 以下の積分をシンプソン公式で近似して円周率を求めよ。n を引数として円周率を返す関数を定義せよ。

$$\pi = 4\int_0^1 \sqrt{1-x^2}\,dx$$

trapezoid.rb (simpson.rb) の f を定義しなおし、

```
def pi(n)
    4.0 * trapezoid または simpson(0.0, 1.0, n)
end
```

1. シンプソン公式で π の計算ができたら投票してください。