アルゴリズム入門
第7回

千葉　滋

東京大学
情報理工学系研究科
創造情報学専攻

1

?

- メリーさんのひつじ
  メェメェひつじ
  メリーさんのひつじ
  まっしろね

  どこでもついていく
  めぇめぇ
  ついていく
  どこでもついていく
  かわいいわね

  あるとき学校へ
  学校へ学校へ
  あるとき学校へ
  ついてきた

  生徒が笑った
  アハハアハハ
  生徒が笑った
  それを見て

  先生はかんかんに
  おこっておこって
  先生はおこって
  追い出した

  メリーさんは困って
  困って困って
  メリーさんはしくしく
  泣き出した

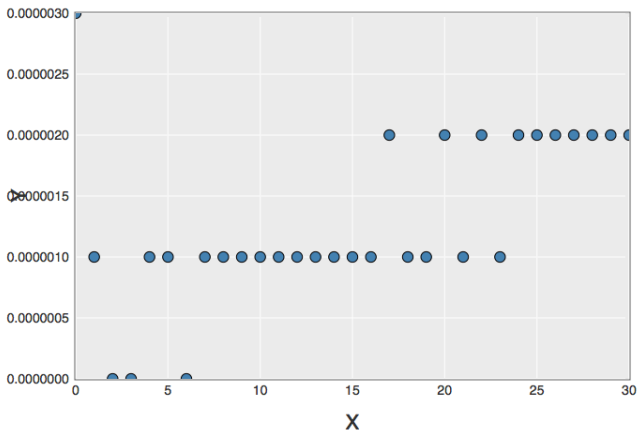
2

感想より

- 前回の再帰地獄とは打って変わって不気味なくらいの簡単さです。罠か何かでしょうか。
- これくらいの課題がちょうどいいです。今日は安心してお昼ご飯が食べられます。

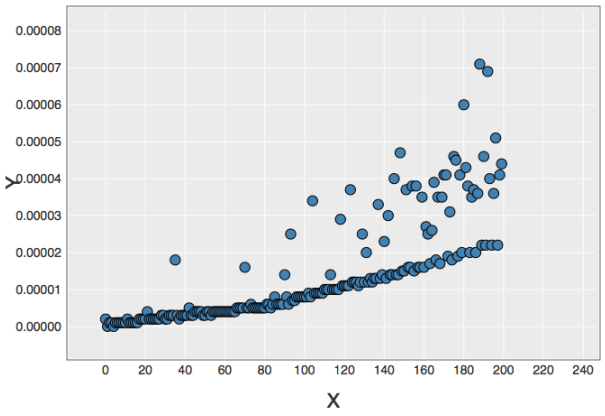
質問

- fib1のグラフはなんでこんな形になるんですか？



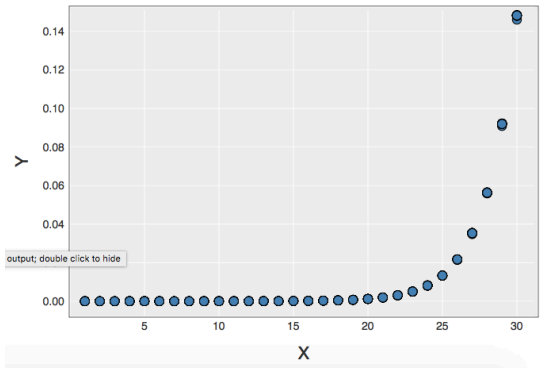
runfibl のグラフ

- 空気よめ？



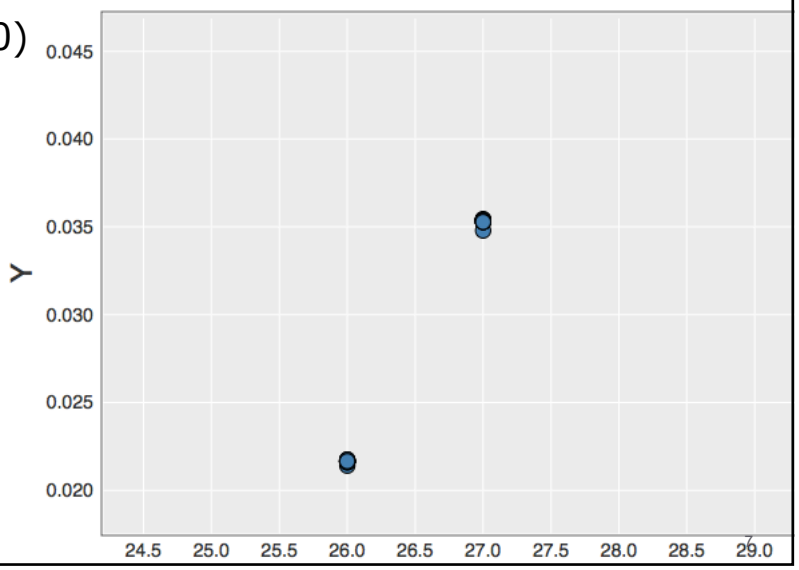
runfib のグラフ

- fib(n)



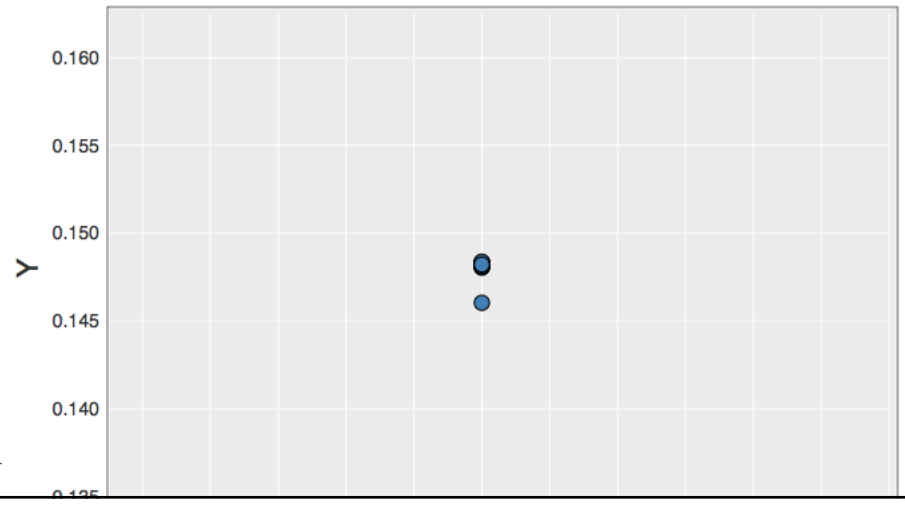
fib(26), fib(27)

- fib(20)



fib(30)

- fib(n)



課題

- 問題1: 前回の課題を自動化する
  - runfib(k) を k を変えながら連続して何回も計算する
  - 関数 manyfibs(m, n) を定義せよ
    - fib(m) から fib(n) まで k を 1 ずつ増やしながら実行し、
      k を要素とする配列と実行時間を要素とする配列を計算する。
      例えば manyfibs(1, 3) なら計算結果は

      [[1, 2, 3]
        [fib(1)の計算時間 , fib(2)の計算時間 , fib(3)の計算時間]]

      となる。

配列の足し算

- 配列は足し算が可能
  - 二つの配列の連結を意味する
  - [1, 2] + [3, 4] の計算結果は [1, 2, 3, 4]
  - [1] + [3, 4] の計算結果は [1, 3, 4]
  - [1, 2] + [3] の計算結果は [1, 2, 3]
  - [] は要素の数が0個の配列
  - [] + [1, 2] の計算結果は [1, 2]
  - [1, 2] + [] の計算結果は [1, 2]

[1] は要素の数が1個の配列の意味。[3] なら0番目の要素が3である配列。配列は集合のようなものなので、要素数が1個や0個の場合もありうる。

manyfibs(k) の定義

- 略とあるところを埋めよ
  - def manyfibs(m, n)
    manyfibs2([[], []], m, n)
    end

    def manyfibs2(results, m, n)
      if m > n
        略
      else
        r = [results[0] + 略,
              results[1] + [runfib(m)]]
        manyfibs2(r, 略 , 略)
      end
    end

    # runfib や fib の定義は前回と同じ

[runfib(m)] は
runfib(m) の計算結果を 1 個の
要素とする配列、の意味

課題

- 問題 2
  - 行列アルゴリズムでフィボナッチ数を計算する fib1 関数を書け。
- 問題 3
  - 線形時間で行列のかけ算をするアルゴリズムでフィボナッチ数を計算する fib2 関数を書き、実行時間を fib1 と比べよ。

行列を用いたアルゴリズム

- **(fib(*k*+1), fib(*k*))**をベクトルだと思うと

$$\begin{pmatrix} \mathrm{fib}(k+1) \\ \mathrm{fib}(k) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \mathrm{fib}(k) \\ \mathrm{fib}(k-1) \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k \begin{pmatrix} \mathrm{fib}(1) \\ \mathrm{fib}(0) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

という関係がある

- 教科書5.3章参照

Q とする

行列を用いたアルゴリズム

- したがって

$$Q^k = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

とすると fib(k) = c + d

動的計画法

- 要するにこれと同じ考え方

${}_nC_{n-2}$ の計算なら (n>2)

- 動的計画法（メモ化）
  - 表全体を記録する必要はない
    の直前の行の値だけ必要

n \ k	0	1	2	3	4	5	6
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1

行列を用いたアルゴリズム

- fib(k)の計算量 = $\boldsymbol{Q}^{k}$ の計算量

- 約 $\log_2 \boldsymbol{k}$ 回の行列積で計算できる
  - 1回あたりの計算: 2×2行列の乗算ほか
- 計算量 = $\boldsymbol{O}(\log \boldsymbol{n})$

$$Q^n = \begin{cases} E & (n = 0) \\ (Q^{n/2})^2 & (n \text{ は } 2 \text{ 以上の偶数}) \\ Q \times Q^{n-1} & (n \text{ は奇数}) \end{cases}$$

Q^k の計算量

- $O(\log_2 k)$ となる
- $Q^k = (Q^{k/2})^2 = ((Q^{k/4})^2)^2 = (\cdots (((Q^2)^2)^2) \cdots)^2$
  - つまり k が偶数なら、２乗を $\log_2 k$ 回繰り返せば Q^k が計算できる
  - 行列積１回分を単位計算量と考えると、Q^k の計算量は $\log_2 k$ に比例、つまり対数関数的にしか増加しない

行列の表現

- 今回は（他の表現方法もありうるが）２次元配列で表現する。
- 例えば

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$ [[1, 1], [1, 0]] と表現する

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$ [[1, 2], [3, 4]] と表現する

[[1,3], [2,4]] でもよいが、今回は上のようにする

行列積

- 関数 matmult

```
def matmult(a, b)
  [ [ a[0][0]*b[0][0]+a[0][1]*b[1][0], 略 ],
    [ 略, 略 ] ]
end
```

略とあるところを埋めよ

問題2: fib1 のプログラムを完成させよ

- matsquare は行列の 2 乗（平方）を計算
  matpower は行列 a の n 乗を計算

```
# k > 0 とする
def fib1(k)
  q = [[1, 1],[1, 0]]
  m = matpower(q, k)
  m[略][略]+m[略][略]
end

def matsquare(a)
  matmult(a, a)
end
```

```
def matpower(a, n)
  if n == 1
    a
  elsif n % 2 == 0
    略
  else
    略
  end
end
```

n%2はnを2で割った余り。つまり n % 2 == 0 は偶数だったら、の意味

ヒント:
$(Q^{n/2})^2$ を計算すればよい。
matpower を再帰的に使うことになる。
matsquare も使う。

奇数だったらこちらを計算

問題3: 単純な行列の掛算によるプログラム

- n 乗を n 回のかけ算で計算
  - 線形時間になる (n に比例した時間がかかる)

以下を完成させよ。使ってよい関数は matmult だけ。

```
def fib2(k)
  q = [[1, 1],[1, 0]]
  m = matpower2(q, k)
  m[略][略]+m[略][略]
end
```

```
def matpower2(a, n)
  result = [[1, 0], [0, 1]]
  for i in 1..n
    result = 略
  end
  result
end
```