アルゴリズム論

2010年度

三つの問題を Python で解く[3]

分割統治法 divide and conquer

(1) 与えられた問題をいくつかの小問題に分割し (divide)

(2) 各小問題の解を求め (conquer)

(3) 得られた小問題の解を用いて元の問題の解を得る

… という操作を再帰的に繰り返して，アルゴリズムを作成する手法

２分探索法 binary search

- ソーティングされているリスト中に指定された値(キー)があるかどうかを調べる
- 見つかればその位置(インデックス)を戻し，なければ特別な値（-1）を戻す
- 基本は，リストの中央値とキーを比較
  - 同じなら見つかった
  - 中央値とキーの大小関係から，次はリストの前半または後半だけを調べればよい
- 最悪でも $\lceil \log_2 n \rceil$ 回の比較で終了する
- 見つからない場合も $\lceil \log_2 n \rceil$ 回の比較で終了

binarySearch(A, key)

```
def binarySearch(A, key):
    """ A[m] = key となる m を戻す。なければ -1 """
    left, right = (0, len(A))    # 探索範囲 left <= m < right
    while left < right:
        middle = (left + right) / 2       # 探索範囲の中央値
        if A[middle] == key:     # key を発見！
            return middle
        if A[middle] < key:      # key は右半分にあるはず
            left = middle + 1
        else:                    # key は左半分にあるはず
            right = middle
    return -1                    # 発見できず
```

binarySR(A, key)

```
def binarySR(A, key):
    """ ２分探索：再帰版 """
    return binarySRAux(A, 0, len(A), key)

def binarySRAux(A, left, right, key):
    """ binarySR の補助関数 (auxiliary) """
    if left >= right:   # 発見できず
        return -1
    middle = (left + right) / 2
    if A[middle] == key:
        return middle
    if A[middle] < key:
        return binarySRAux(A, middle + 1, right, key)
    else:
        return binarySRAux(A, left, middle, key)
```

グラフ彩色問題

- グラフ G の頂点を k 色で彩色できるとき，G は k-彩色可能であるという
- グラフ G の彩色数とは，G を彩色できる最小の色数
- 関数 colorable(G, k) は，グラフ G が k-彩色可能なら true，k-彩色不可能なら false を返すとする

判定問題と最適化問題

- グラフ彩色判定問題
  - 入力：グラフ G および正整数 k
  - 性質： G は k–彩色可能
- グラフ彩色最適化問題
  - 入力：グラフ G
  - 出力： G の彩色数
- 判定問題を解く関数 colorable(G, k) があると仮定する

☆ colorable(G, k) を利用して，最適化問題を解く関数 chromatic(G) を作るには …

彩色問題の性質

- 彩色数を χ とすると $1 \le \chi \le |V|$
- $k-$彩色可能なら $(k+1)-$彩色可能

k	1	2	$\cdots$	$\chi-1$	χ	$\chi+1$	$\cdots$	$\lvert V \rvert$
T or F	F	F	F	F	T	T	T	T

$\min \le \chi \le \max$ であるとき

$$k-\text{彩色可能} \Rightarrow \min \le \chi \le k$$

$$k-\text{彩色不可能} \Rightarrow k+1 \le \chi \le \max$$

課題 b1：グラフ彩色問題

- グラフ彩色判定問題を解く関数 colorable(G, k) を利用して，グラフ彩色最適化問題を解く関数 chromatic(G) を書け。
- グラフ情報は G = [頂点数, 彩色数] であり，colorable(G, k) は次のように定義しておく。

```
def colorable(G, k):
    return (G[1] <= k)
```

- ただし，次のプログラムは不可。

chromatic(G)

```
def chromatic(G):
    """ これはルール違反 """
    return G[1]    # G[1] に G の彩色数が書かれている

def chromatic(G):
    """ 最悪の場合 colorable を頂点の数だけ呼び出す """
    for k in range(1, G[0]):  # G[0] はグラフの頂点数
        if colorable(G, k):
            return k
    return G[0]

def chromatic(G):    # 課題 b1 が要求するプログラム
    """ 最悪でも colorable を ⌈log2 n⌉ 回呼び出して終わる """
      .....
      .....
```

対数 logarithm

正の実数 $a \neq 1$ をとると, 任意の正の実数 x に対し
$x = a^p$ を満たす実数 p が唯一定まる。
この p を $p = \log_a x$ と書き, a を底とする x の対数という。

［常用対数/ブリッグスの対数］common logarithm
$a = 10$ であり, $\mathrm{Log}\, x$ と書くことがある

［自然対数/ネイピアの対数］natural logarithm
$a = e$ であり, $\ln x$ と書くことがある

［二進対数］binary logarithm
$a = 2$ であり, $\lg x$ と書くことがある

単に $\log x$ とあれば, 前後の文脈や扱われている分野から判断する

離散対数 [1]

具体例から説明する　　$S = \{1, 2, 3, 4, 5, 6\}$ とする

S において, $a \bullet b = ab \bmod 7 =$ (a ∗ b) % 7 とする [7 を法とする乗算]

代数系 $\langle S, \bullet \rangle$ は, 1 を単位元とする群である

$\forall x, y, z \in S : (x \bullet y) \bullet z = x \bullet (y \bullet z)$

$\forall x \in S : 1 \bullet x = x \bullet 1 = x$

$1 \bullet 1 = 1 \Rightarrow 1^{-1} = 1, \quad 2 \bullet 4 = 1 \Rightarrow 2^{-1} = 4,$

$3 \bullet 5 = 1 \Rightarrow 3^{-1} = 5, \quad 4 \bullet 2 = 1 \Rightarrow 4^{-1} = 2,$

$5 \bullet 3 = 1 \Rightarrow 5^{-1} = 3, \quad 6 \bullet 6 = 1 \Rightarrow 6^{-1} = 6$

•	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

代数系 $\langle S, \bullet \rangle$ は可換群, かつ, 巡回群

離散対数 [2]

$\bullet$	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	4	6	1	3	5
3	3	6	2	5	1	4
4	4	1	5	2	6	3
5	5	3	1	6	4	2
6	6	5	4	3	2	1

べき乗：$a^0=1$ [単位元], $a^k=a\bullet a^{k-1}$ $[k\geq 1]$

$3^0=1,\ 3^1=3,\ 3^2=2,\ 3^3=6,\ 3^4=4,$
$3^5=5,\ 3^6=1,\ 3^7=3^1,\ 3^8=3^2,\cdots$

S のすべての要素は,
3^k $[0\leq k<6]$ で重複なく表現できる

$\Rightarrow S$ は, 3 で生成される位数 6 の巡回群である

離散対数問題：
生成元 g で生成される巡回群 G において,
$a\in G$ であるとき, $g^k=a$ となる k を求めよ

$\log_3 1=0,\ \log_3 2=2,$
$\log_3 3=1,\ \log_3 4=4,$
$\log_3 5=5,\ \log_3 6=3$

このような k を $\log_g a$ と書く
$0\leq \log_g a<|G|$ の範囲で一意的に定まる

離散対数 [3]

林卓也, 高木剛; “離散対数問題解読世界記録更新への道ー676ビットの解読ー”, 情報処理, Vol.51, No.9, pp.1181-1188, Sep. 2010.

2009年12月9日：有限体 $GF(3^{6\times 71})$（位数：676ビット）上の離散対数問題の計算に成功した。この結果は, 05年9月にフランスのグループによって達成された613ビットの解読記録を約60ビット更新した。

ところで… 676ビットは10進数で何桁？

正の整数 k を n 進数で表すと桁数は $\lfloor \log_n k \rfloor + 1$ である

$$\lfloor \log_{10} 2^{676} \rfloor + 1 = \lfloor 676 \log_{10} 2 \rfloor + 1$$

$$= \lfloor 676 \times 0.30102999\cdots \rfloor + 1 = \lfloor 203.4962770\cdots \rfloor + 1 = 204$$

離散対数 [4]

離散対数問題：
生成元 g で生成される巡回群 G において,
$a \in G$ であるとき, $g^k = a$ となる k を求めよ

g^k の計算（N を法とする）にどんな工夫ができますか？

```
# normal version
ans = 1
for i in range(k):
    ans = (ans * g) % N
```

```
# binary version
ans = 1
while k != 0:
    if k % 2 == 1:
        ans = (ans * g) % N
    k /= 2
    g = (g * g) % N
```

$$g^5 = g^4 \times g^1$$
$$g^{10} = g^8 \times g^2$$
$$g^{11} = g^8 \times g^2 \times g^1$$

課題 b2：べき乗計算

- N を法とする正整数の乗算において，g の k 乗を計算する２種類の関数 powBinary(g, k, N) と powNormal(g, k, N) をまず作る。さらに，これらを times 回計算してその計算時間を表示する。
  [PC にできるだけ他の負荷をかけないで計測]
- さまざまな値に対して両者の計算時間を計測してその結果をグラフや表にまとめ，powBinary の優位性をアピールせよ。
  - g を固定して k を変えるとどうなるか。逆は？
  - 計算時間に N は影響しない？
  - powNormal の方が早い場合もあるだろう

課題 00 のキーワード

- GA：遺伝(的)アルゴリズム
- GP：遺伝的プログラミング
- 近似アルゴリズム
- 並列コンピューティング
- (normalized-) least mean-squares: (N-)LMS
- Dijkstra 法：最短経路探索手法
- 通信プロトコル
- アルゴリズムのアイデア・経緯・歴史・証明