プログラミングの基礎Ⅱ

０．目次

２．１　コラッツ問題

自然数nから出発して、nが偶数ならば、2で割り、nが奇数ならば、3倍して1を足す操作を行う。この操作を繰り返すと最後に1になると予想されている。

問題１　自然数aの操作回数を求めよ。
問題２　自然数aからbまでのなかで、最大操作回数となる自然数を求めよ。

２．２　耐久数

正整数の各桁の数字を掛け、得られた結果についても同様の操作を繰り返す。すると、最後は１桁の数になる。
たとえば、77 -> 49 -> 36 -> 18 -> 8 より、操作回数は4となる。
この操作回数を耐久数という。

問題１　正整数ｎを複数個読み込み、それぞれの耐久数を求めよ。ただし、データの終わりは0とする。
問題２　正整数b(≧10)以下の耐久数を求めよ。

２．３　ピタゴラス数

正整数a,b,cについて、$a^2 + b^2 = c^2$（$1 \leqq a \leqq b \leqq c$）とする。
（このような正整数a,b,cをピタゴラス数という）

問題１　$c^2 \leqq n$を満たすピタゴラス数をすべて求めよ。

２．４　６１７４

４桁の正整数nにおいて、各桁の数字を大きい順に並べ替えた数から小さい順に並べ替えた数を引く。この操作を繰り返すと、同じ数字からなる４桁の数以外の正整数から始めると6174になることが知られている。
正整数をいくつか読込み、それぞれについて、6174になるまでの回数を求めよ。たとえば、1234→3087→8352→6174で３回となる。
ただし、データの最後は0とする。

２．プログラムの作成

２．１　コラッツ問題

自然数nから出発して、nが偶数ならば、2で割り、nが奇数ならば、3倍して1を足す操作を行う。この操作を繰り返すと最後に1になると予想されている。
たとえば、3→10→5→16→8→4→2→1。操作回数7回で1となる。

●自然数（1≦n≦100）の操作回数

n	回数	n	回数	n	回数	n	回数	n	回数
1	0	21	7	41	109	61	19	81	22
2	1	22	15	42	8	62	107	82	110
3	7	23	15	43	29	63	107	83	110
4	2	24	10	44	16	64	6	84	9
5	5	25	23	45	16	65	27	85	9
6	8	26	10	46	16	66	27	86	30
7	16	27	111	47	104	67	27	87	30
8	3	28	18	48	11	68	14	88	17
9	19	29	18	49	24	69	14	89	30
10	6	30	18	50	24	70	14	90	17
11	14	31	106	51	24	71	102	91	92
12	9	32	5	52	11	72	22	92	17
13	9	33	26	53	11	73	115	93	17
14	17	34	13	54	112	74	22	94	105
15	17	35	13	55	112	75	14	95	105
16	4	36	21	56	19	76	22	96	12
17	12	37	21	57	32	77	22	97	118
18	20	38	21	58	19	78	35	98	25
19	20	39	34	59	32	79	35	99	25
20	7	40	8	60	19	80	9	100	25

●n=97の場合

```
 97→292 →146 →73  →220 →110 →55  →166 →83  →250
125→376 →188 →94  →47  →142 →71  →214 →107 →322
161→484 →242 →121 →364 →182 →91  →274 →137 →412
206→103 →310 →155 →466 →233 →700 →350 →175 →526
263→790 →395 →1186→593 →1780→890 →445 →1336→668
334→167 →502 →251 →754 →377 →1132→566 →283 →850
425→1276→638 →319 →958 →479 →1438→719 →2158→1079
3238→1619→4858→2429→7288→3644→1822→911 →2734→1367
4102→2051→6154→3077→9232→4616→2308→1154→577 →1732
866→433 →1300→650 →325 →976 →488 →244 →122 →61
184→ 92 →46  →23  →70  →35  →106 →53  →160 →80
 40→ 20 →10  →5   →16  →8   →4   →2   →1
```

問題１　自然数aの操作回数を求めよ。

●考え方

定義通り。

●プログラム(a211.c)

```
/* << a211.c >> */
#include <stdio.h>
#include <stdlib.h>
#define INTMAX 2147483647 /* 最大正整数。*/

int main() {
  int a,      /* 自然数a。*/
      count,  /* 操作回数。*/
      t;

  /* 自然数aの読み込み。*/
  scanf("%d",&a);

  /* 初期設定。*/
  count = 0;

  /* 自然数aの操作回数countを求める。*/
  t = a;
  while( t != 1 ) {
    count++;
    if( t%2 == 0 ) {
      t = t/2;
    } else {
      /* オーバーフロー処理。*/
      if( t > (INTMAX-1)/3 ) {
        printf("オーバーフロー¥n"); exit(1);
      }
      t = 3*t+1;
    }
  }

  /* 結果出力。*/
  printf("a=%d count=%d¥n",a,count);
}
```

実行結果

```
% cc a211.c
% ./a.out
101
a=101 count=25
% ./a.out
715827882
a=715827882 count=32
% ./a.out
715827883
オーバーフロー
```

問題２　自然数aからbまでのなかで、最大操作回数となる自然数を求めよ。

●考え方１

定義通り。

●プログラム(a212a.c)

```
/* << a212a.c >> */
#include <stdio.h>
#include <stdlib.h>
#define INTMAX 2147483647 /* 最大正整数。*/

int main() {
  int a,b,    /* 自然数a,b。*/
      count,  /* 操作回数。*/
      m,      /* 最大操作回数となる自然数。*/
      max,    /* 最大操作回数。*/
      n,      /* 自然数。*/
      t;

  /* 自然数a,bの読み込み。*/
  scanf("%d%d",&a,&b);

  /* 初期設定。*/
  max = 0;

  /* 自然数a以上b以下の最大操作回数を求める。*/
  for( n=a; n<=b; n++ ) {
    /* 自然数nの操作回数countを求める。*/
    t = n;
    count = 0;
    while( t != 1 ) {
      count++;
      if( t%2 == 0 ) {
        t = t/2;
      } else {
      /* オーバーフロー処理。*/
      if( t > (INTMAX-1)/3 ) {
        printf("%d オーバーフロー¥n",t); exit(1);
      }
        t = 3*t+1;
      }
    }

    /* 最大操作回数の更新。mは最大操作回数となる自然数。*/
    if( count > max ) {
      max = count; m = n;
    }
```

```
42    }
43
44    /* 結果出力。*/
45    printf("%d～%dにおいて、%dで最大操作回数%d¥n",a,b,m,max);
46  }
```

実行結果

```
% cc a212a.c
% ./a.out
1 100
1～100において、97で最大操作回数118
% ./a.out
1 1000
1～1000において、871で最大操作回数178
% ./a.out
1 10000
1～10000において、6171で最大操作回数261
% ./a.out
1 715827882
827370449 オーバーフロー
```

実行時間

```
% cc a212a.c
% time ./a.out
1 100000
1～100000において、77031で最大操作回数350
0.068u 0.002s 0:03.79 1.5%      0+0k 0+0io 0pf+0w
```

●考え方２

考え方１を改良して、すでに得られた操作回数を配列に保存し、他の自然数の操作回数を求めるときに利用する。

●プログラム(a212b.c)

```
/* << a212b.c >> */
#include <stdio.h>
#include <stdlib.h>
#define INTMAX 2147483647 /* 最大正整数。*/
#define MAX 1000          /* 配列の最大要素数。*/

int main() {
  int a,b,       /* 自然数a,b。*/
      c[MAX+1], /* 1≦n≦MAXについて操作回数を保存する配列。*/
      count,     /* 操作回数。*/
      m,         /* 最大操作回数となる自然数。*/
      max,       /* 最大操作回数。*/
      n,         /* 自然数。*/
      t;

  /* 自然数a,bの読み込み。*/
  scanf("%d%d",&a,&b);

  /* 初期設定。*/
  for( n=1; n<=MAX; n++ ) { c[n] = 0; }
  max = 0;

  /* 自然数a以上b以下の最大操作回数を求める。*/
  for( n=a; n<=b; n++ ) {
    /* 自然数nの操作回数countを求める。*/
    t = n; count = 0;
    while( t != 1 ) {
      count++;
      if( t%2 == 0 ) {
        t = t/2;
      } else {
      /* オーバーフロー処理。*/
      if( t > (INTMAX-1)/3 ) {
        printf("%d オーバーフロー¥n",t); exit(1);
      }
        t = 3*t+1;
      }

      /* すでに得られた結果を利用する。*/
      if( t <=MAX ) {
        if( c[t] > 0 ) {
          count = count + c[t]; break;
        }
```

```
44        }
45      }
46      if( n <= MAX ) { c[n] = count; }
47
48      /* 最大操作回数の更新。*/
49      if( count > max ) {
50        max = count; m = n;
51      }
52    }
53
54    /* 結果出力。*/
55    printf("%d～%dにおいて、%dで最大操作回数%d¥n",a,b,m,max);
56  }
```

実行結果

```
% cc a212b.c
% ./a.out
100 1000
100～1000において、871で最大操作回数178
% ./a.out
1000 10000
1000～10000において、6171で最大操作回数261
% ./a.out
10000 100000
10000～100000において、77031で最大操作回数350
```

実行時間

```
% cc a212a.c
% time ./a.out
1 100000
1～100000において、77031で最大操作回数350
0.025u 0.000s 0:04.17 0.4%      0+0k 0+0io 0pf+0w
```

２．２　耐久数

正整数の各桁の数字を掛け、得られた結果についても同様の操作を繰り返す。すると、最後は１桁の数になる。
たとえば、77 -> 49 -> 36 -> 18 -> 8 より、操作回数は4となる。
この操作回数を**耐久数**という。

```
77  -->  7×7 = 49
49  -->  4×9 = 36
36  -->  3×6 = 18
18  -->  1×8 =  8
```

問題１　正整数aを複数個読み込み、それぞれの耐久数を求めよ。
ただし、データの終わりは0とする。

●プログラム(a221.c)

```
1  /* << a221.c >> */
2  #include <stdio.h>
3
4  int main() {
5    int a,      /* 正整数a。*/
6        d,      /* 正整数nの各桁の数字。*/
7        m,      /* 各桁の積。*/
8        count; /* 操作回数。*/
9
10   while( 1 ) {
11     /* 正整数aの読み込み。*/
12     scanf("%d",&a);
13     if( a <= 0 ) { break; }
14
15     /* 初期設定。*/
16     count = 0;
17
18     /* 耐久数の計算。*/
19     printf("%10d の",a);
20     while( a > 9 ) {
21       /* 整数aの各桁の積mを求める。*/
22       m = 1;
23       while( a > 0 ) {
24         d = a%10;
25         m = m*d;
26         a = a/10;
27       }
28       count++;
29       a = m;
30     }
31
32     /* 結果出力。*/
33     printf("耐久数 : %d ¥n",count);
34   }
35 }
```

実行結果

```
% cc a221.c
% ./a.out
10
        10 の耐久数：1
25
        25 の耐久数：2
39
        39 の耐久数：3
77
        77 の耐久数：4
679
       679 の耐久数：5
6788
      6788 の耐久数：6
68889
     68889 の耐久数：7
2677889
   2677889 の耐久数：8
26888999
  26888999 の耐久数：9
0
```

（考察）

正整数ｎの各桁の数字を掛け、得られた結果をｍとすると、ｎ＞ｍ　が成り立つ。したがって、耐久数を求める手順は、必ず終了する。

・２桁の場合。

$$10a + b - ab = a(10 - b) + b > 0$$

・３桁の場合。

$$\begin{aligned} 100a + 10b + c - abc &= 10(10a + b) + c - abc \\ &> 10(ab) + c - abc \\ &= ab(10 - c) + c \\ &> 0 \end{aligned}$$

・４桁の場合。

$$\begin{aligned} 1000a + 100b + 10c + d - abcd &= 10(100a + 10b + c) + d - abcd \\ &> 10(abc) + d - abcd \\ &= abc(10 - d) + d \\ &> 0 \end{aligned}$$

k桁の場合も同様に示すことができる。

問題２　正整数b以下の耐久数をすべて求めよ。

正整数nの各桁の数字の積をmとし、その耐久数をs[m]とする。
考察から、正整数nの耐久数s[n]を求める場合、正整数mの耐久数s[m]から、

```
s[n] = s[m] + 1
```

で求められる。ひとつずつ、耐久数を求めるより、効率がよい。

●プログラム(a222.c)

```
/* << a222.c>> */
#include <stdio.h>
#include <stdlib.h>
#define N 9999 /* 耐久数を保存する配列要素の最大値。*/

int main() {
  int b,     /* 正整数b。*/
      d,     /* 正整数の桁の数字。*/
      m,     /* 正整数の各桁の積。*/
      n,     /* 正整数n。*/
      t,
      s[N+1]; /* s[n]：正整数nの耐久数。*/

  /* 正整数bの読み込み。*/
  scanf("%d",&b);
  if( (b <= 0)||(b > N) ) { exit(0); }

  /* 初期設定。*/
  for( n=0; n<=9; n++ ) { s[n] = 0; }

  /* 耐久数の計算。*/
  for( n=10; n<=b; n++ ) {
    t = n;
    m = 1;

    /* 整数tの各桁の積mを求める。*/
    while( t > 0 ) {
      d = t%10;
      m = m*d;
      t = t/10;
    }
    s[n] = s[m] + 1;
  }

  /* 結果出力。*/
  for( n=1; n<=b; n++ ) {
    printf("%5d:%2d",n,s[n]);
    if( n%8 == 0 ) { printf("¥n"); }
  }
  printf("¥n");
}
```

実行結果

```
% cc a222.c
% ./a.out
200
  1: 0   2: 0   3: 0   4: 0   5: 0   6: 0   7: 0   8: 0
  9: 0  10: 1  11: 1  12: 1  13: 1  14: 1  15: 1  16: 1
 17: 1  18: 1  19: 1  20: 1  21: 1  22: 1  23: 1  24: 1
 25: 2  26: 2  27: 2  28: 2  29: 2  30: 1  31: 1  32: 1
 33: 1  34: 2  35: 2  36: 2  37: 2  38: 2  39: 3  40: 1
 41: 1  42: 1  43: 2  44: 2  45: 2  46: 2  47: 3  48: 2
 49: 3  50: 1  51: 1  52: 2  53: 2  54: 2  55: 3  56: 2
 57: 3  58: 2  59: 3  60: 1  61: 1  62: 2  63: 2  64: 2
 65: 2  66: 3  67: 2  68: 3  69: 3  70: 1  71: 1  72: 2
 73: 2  74: 3  75: 3  76: 2  77: 4  78: 3  79: 3  80: 1
 81: 1  82: 2  83: 2  84: 2  85: 2  86: 3  87: 3  88: 3
 89: 3  90: 1  91: 1  92: 2  93: 3  94: 3  95: 3  96: 3
 97: 3  98: 3  99: 2 100: 1 101: 1 102: 1 103: 1 104: 1
105: 1 106: 1 107: 1 108: 1 109: 1 110: 1 111: 1 112: 1
113: 1 114: 1 115: 1 116: 1 117: 1 118: 1 119: 1 120: 1
121: 1 122: 1 123: 1 124: 1 125: 2 126: 2 127: 2 128: 2
129: 2 130: 1 131: 1 132: 1 133: 1 134: 2 135: 2 136: 2
137: 2 138: 2 139: 3 140: 1 141: 1 142: 1 143: 2 144: 2
145: 2 146: 2 147: 3 148: 2 149: 3 150: 1 151: 1 152: 2
153: 2 154: 2 155: 3 156: 2 157: 3 158: 2 159: 3 160: 1
161: 1 162: 2 163: 2 164: 2 165: 2 166: 3 167: 2 168: 3
169: 3 170: 1 171: 1 172: 2 173: 2 174: 3 175: 3 176: 2
177: 4 178: 3 179: 3 180: 1 181: 1 182: 2 183: 2 184: 2
185: 2 186: 3 187: 3 188: 3 189: 3 190: 1 191: 1 192: 2
193: 3 194: 3 195: 3 196: 3 197: 3 198: 3 199: 2 200: 1
```

２．３　ピタゴラス数

正整数a,b,cについて、$a^2 + b^2 = c^2$（$1 \leqq a \leqq b \leqq c$）とする。
（このような正整数a,b,cを**ピタゴラス数**という）
a,bが互いに素なピタゴラス数を既約なピタゴラス数という。規約なピタゴラス数をいくつか示す。

a	b	c	a	b	c	a	b	c
3	4	5	19	180	181	252	275	373
5	12	13	57	176	185	135	352	377
8	15	17	95	168	193	189	340	389
7	24	25	28	195	197	228	325	397
20	21	29	84	187	205	40	399	401
12	35	37	21	220	221	120	391	409
9	40	41	60	221	229	29	420	421
28	45	53	105	208	233	87	416	425
11	60	61	120	209	241	145	408	433
16	63	65	32	255	257	84	437	445
48	55	73	23	264	265	280	351	449
13	84	85	69	260	269	168	425	457
39	80	89	115	252	277	261	380	461
65	72	97	160	231	281	31	480	481
20	99	101	161	240	289	44	483	485
60	91	109	68	285	293	132	475	493
15	112	113	136	273	305			
44	117	125	25	312	313			
88	105	137	75	308	317			
17	144	145	36	323	325			
51	140	149	175	288	337			
85	132	157	180	299	349			
119	120	169	225	272	353			
52	165	173	27	364	365			

$c^2 \leqq n$を満たすピタゴラス数の個数を示す。

n	ピタゴラス数の個数
10^2	2
10^3	11
10^4	52
10^5	220
10^6	881
10^7	3371
10^8	12471
10^9	45251
10^{10}	161436
10^{11}	568423
10^{12}	1980642
10^{13}	6842852

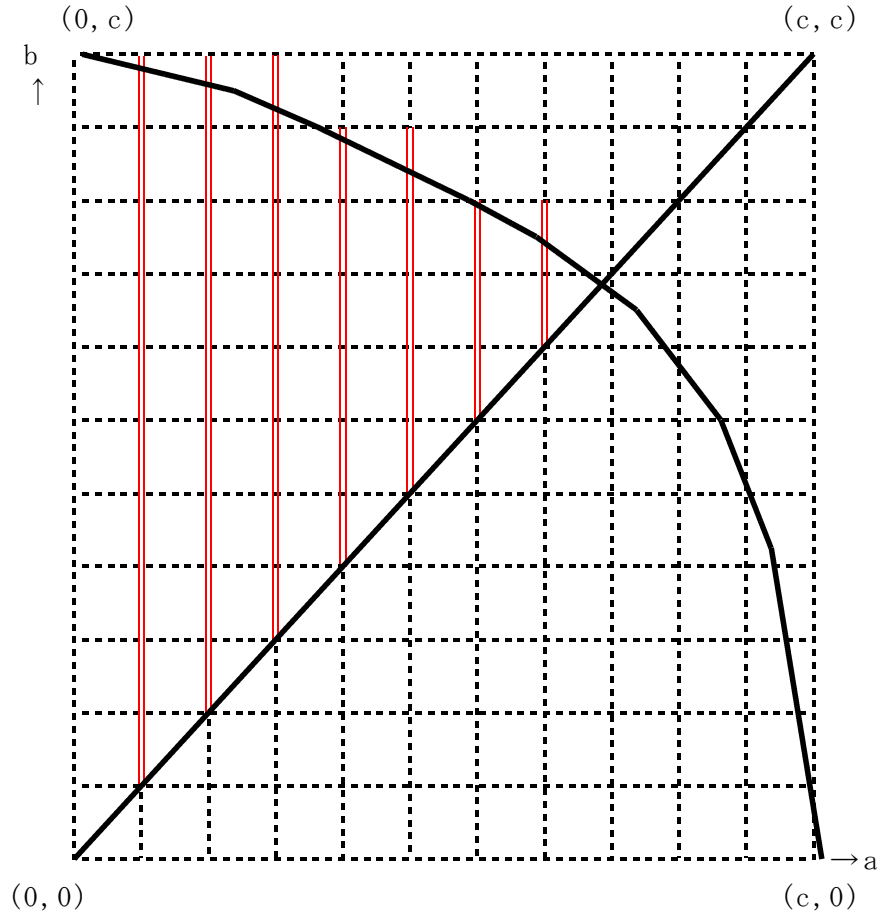
問題１　$c^2 \leqq n$を満たすピタゴラス数をすべて求めよ。

●考え方１

整数a,bを少しずつ大きくして、$a^2 + b^2 = c^2$が成り立つかどうか調べていく。$1 \leqq a \leqq b \leqq c$より、$2a^2 \leqq c^2$ が成り立つ。

●プログラム(a231a.c)

```
1  /* << a231a.c >> */
2  #include <stdio.h>
3
4  int main () {
5    int a,b,c,
6        count, /* ピタゴラス数の個数。*/
7        n,     /* nの値。*/
8        w;
9
10   /* 整数nの読み込み。*/
11   scanf("%d",&n);
12
13   /* 初期設定。*/
14   count = 0;
15
16   /* ピタゴラス数を求める。*/
17   c = 1;
18   while( c*c <= n ) {
19     for( a=1; 2*a*a<=c*c; a++ ) {
20       for( b=a; b<=c; b++ ) {
21         w = a*a + b*b - c*c;
22         if( w > 0 ) {
23           break;
24         }
25         if( w == 0 ) {
26           count++; printf("%d: %d %d %d¥n",count,a,b,c);
27         }
28       }
29     }
30     c++;
31   }
32
33   /* 結果出力。*/
34   printf("%d以下のピタゴラス数は%d個¥n",n,count);
35 }
```



実行結果

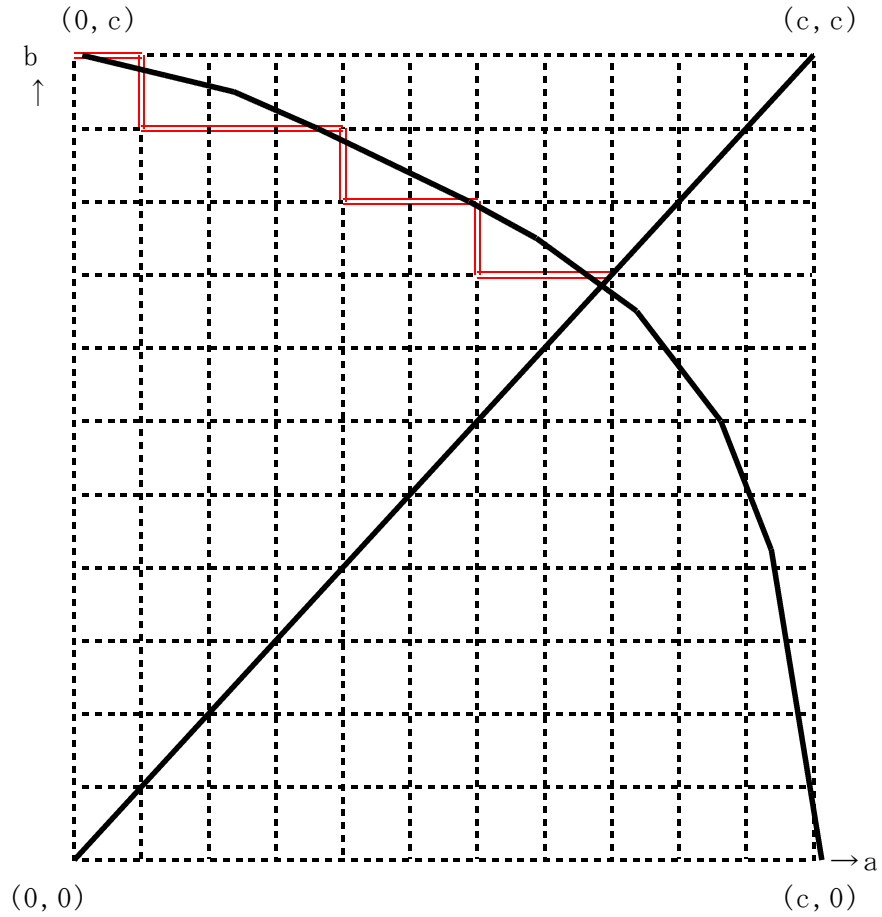
```
% cc a231a.c
% ./a.out
1000
1: 3 4 5
2: 6 8 10
3: 5 12 13
4: 9 12 15
5: 8 15 17
6: 12 16 20
7: 7 24 25
8: 15 20 25
9: 10 24 26
10: 20 21 29
11: 18 24 30
1000以下のピタゴラス数は11個
```

実行時間（出力時間は除く）

```
% cc a231a.c
% time ./a.out
1000000
1000000以下のピタゴラス数は881個
0.640u 0.000s 0:03.84 16.6%     0+0k 0+8io 0pf+0w
% time ./a.out
10000000
10000000以下のピタゴラス数は3371個
19.965u 0.001s 0:22.96 86.9%     0+0k 0+0io 0pf+0w
```

●考え方２

$c^2 \leqq n$を満たすピタゴラス数（$a^2 + b^2 = c^2$ a,b,c（$a \leqq b \leqq c$）は正整数）の求め方を考察する。
cの値が与えられたとする。ピタゴラス数は格子点(0,0)と格子点(c,c)を結ぶ対角線より上側の4分円$a^2+b^2=c^2$（$0 \leqq a \leqq c, 0 \leqq b \leqq c$）上に存在する。平面の格子点(a,b)を4分円に沿って見落としがないようにたどっていくと、効率よくピタゴラス数をさがすことができる。二重線に注目してほしい。



●プログラム(a231b.c)

```
1  /* << a231b.c >> */
2  #include <stdio.h>
3
4  int main () {
5    int a,b,c,
6        count, /* ピタゴラス数の個数。*/
7        n,     /* nの値。*/
8        w;
9
10   /* 整数nの読み込み。*/
11   scanf("%d",&n);
12
13   /* 初期設定。*/
14   count = 0;
15
16   /* ピタゴラス数を求める。*/
17   c = 1;
18   while( c*c <= n ) {
19     a = 1;
20     b = c;
21     while( a <= b ) {
22       w = a*a + b*b - c*c;
23       if( w > 0 ) {
24          b--; continue;
25       }
26       if( w == 0 ) {
27         count++; printf("%d: %d %d %d¥n",count,a,b,c);
28       }
29       a++;
30     }
31      c++;
32   }
33
34   /* 結果出力。*/
35   printf("%d以下のピタゴラス数は%d個¥n",n,count);
36 }
```

実行結果

```
% cc a231b.c
% ./a.out
1000
1: 3 4 5
2: 6 8 10
3: 5 12 13
4: 9 12 15
5: 8 15 17
6: 12 16 20
7: 7 24 25
8: 15 20 25
9: 10 24 26
10: 20 21 29
11: 18 24 30
1000以下のピタゴラス数は11個
```

実行時間（出力時間は除く）

```
% cc a231b.c
% time ./a.out
1000000
1000000以下のピタゴラス数は881個
0.002u 0.000s 0:02.76 0.0%      0+0k 0+0io 0pf+0w
isemba@hcs.ibaraki.ac.jp[63]:!!
% time ./a.out
10000000
10000000以下のピタゴラス数は3371個
0.030u 0.000s 0:03.07 0.9%      0+0k 0+0io 0pf+0w
isemba@hcs.ibaraki.ac.jp[64]:!!
% time ./a.out
100000000
100000000以下のピタゴラス数は12471個
0.269u 0.000s 0:03.22 8.0%      0+0k 0+0io 0pf+0w
isemba@hcs.ibaraki.ac.jp[65]:!!
% time ./a.out
1000000000
1000000000以下のピタゴラス数は45251個
2.686u 0.002s 0:05.73 46.7%     0+0k 0+0io 0pf+0w
```

２．４　６１７４

４桁の正整数nにおいて、各桁の数字を大きい順に並べ替えた数から小さい順に並べ替えた数を引く。この操作を繰り返すと、同じ数字からなる４桁の数以外の正整数から始めると6174になることが知られている。
正整数をいくつか読込み、それぞれについて、6174になるまでの回数を求めよ。ただし、データの最後は0とする。

たとえば、1234→3087→8352→6174で３回となる。

```
1234    4321 - 1234 = 3087
3087    8730 -  378 = 8352
8352    8532 - 2358 = 6174
6174    7641 - 1467 = 6174
```

●考え方１

正整数nを各桁に分解し、数字の出現頻度を配列d[*]に保存する。
配列d[*]を使って、最大数と最小数を求める。

●プログラム(a241a.c)

```
/* << a241a.c >> */
#include <stdio.h>

int main() {
  int count, /* 操作回数。*/
      d[10], /* d[i]は数字iの出現回数を意味する。*/
      i,j,
      k,     /* 桁数。*/
      max,   /* 最大数。*/
      min,   /* 最小数。*/
      n,     /* ４桁の正整数。*/
      t;

  while( 1 ) {
    /* 4桁正整数の読み込み。*/
    scanf("%d",&n);
    if( n <= 0 ) { break; }

    /* 初期設定。*/
    count = 0;

    /* 処理操作。*/
    t = n;
    while( t != 6174 ) {
      /* 整数nを各桁に分解する。kは桁数。d[i]は数字iがd[i]回
         現れたことを意味する。*/
      for( i=0; i<=9; i++ ) { d[i] = 0; }
```

```
28        k = 0;
29        while( t > 0 ) {
30          k++;
31          d[t%10]++;
32          t = t/10;
33        }
34
35        /* 数字0を見落とさない処理。*/
36        while( k < 4 ) { k++; d[0]++; }
37
38        /* 最大数maxを求める。*/
39        max = 0;
40
41
42
43
44
45
46        /* 最小数minを求める。*/
47        min = 0;
48
49
50
51
52
53          count++;
54          t = max - min;
55      }
56
57      /* 結果出力。*/
58      printf("%dの操作回数は%d¥n",n,count);
59    }
60  }
```

実行結果

```
% cc a241a.c
% ./a.out
1234
1234の操作回数は3
1112
1112の操作回数は5
0
```

●考え方２

正整数nを各桁に分解し、数字を昇順に並べ替え、最大数と最小数を求める。

●プログラム(a241b.c)

```
/* << a241b.c >> */
#include <stdio.h>

int main() {
  int count,  /* 操作回数。*/
      d[5],   /* 正整数の各桁を保存する配列。*/
      i,j,
      k,      /* 桁数。*/
      max,    /* 最大数。*/
      min,    /* 最小数。*/
      n,      /* ４桁の正整数。*/
      t;

  while( 1 ) {
    /* 4桁正整数の読み込み。*/
    scanf("%d",&n);
    if( n <= 0 ) { break; }

    /* 初期設定。*/
    count = 0;

    /* 処理操作。*/
    t = n;
    while( t != 6174 ) {
      /* 整数nを各桁に分解し、配列d[*]に保存。kは桁数。*/
      k = 0;
      while( t > 0 ) {
        k++;
        d[k] = t%10;
        t = t/10;
      }

      /* 数字0を見落とさない処理。*/
      while( k < 4 ) { k++; d[k] = 0; }

      /* 配列d[*]の要素を昇順に並べる。*/
      for( i=k; i>1; i-- ) {
        for( j=1; j<i; j++ ) {
          if( d[j] > d[j+1] ) {
            t = d[j]; d[j] = d[j+1]; d[j+1] = t;
          }
        }
      }

```

```
45          /* 最大数を求める。*/
46          max = 0;
47
48
49
50
51          /* 最小数を求める。*/
52          min = 0;
53
54
55
56          count++;
57          t = max - min;
58        }
59
60        /* 結果出力。*/
61        printf("%dの操作回数は%d¥n",n,count);
62      }
63  }
```

実行結果

```
% cc a241b.c
% ./a.out
1234
1234の操作回数は3
1123
1112の操作回数は5
0
```