２項係数計算法の開発

０．目次

２項係数の計算法をいろいろ考察する。

１．開発方法１（階乗を用いる方法）

２項係数 $c(n,r)=\dfrac{n!}{(n-r)!r!}=\dfrac{n(n-1)\cdots(n-r+1)}{r!}$ の定義を使って計算する。

この定義にしたがって計算すると（すなわち、n!と(n-r)!とr!を求め、その後、n!/((n-r)!r!)を求める）、階乗の計算の部分で大きな値が現れ、コンピュータの計算精度の関係からオーバーフローという現象が起こる。この現象が起こると計算を続けることが不可能になる。

オーバーフローが発生するので、狭い範囲しか計算できない欠点がある。

n	n!	n	n!
1	1	11	39916800
2	2	12	479001600
3	6	13	6227020800
4	24	14	87178291200
5	120	15	1307674368000
6	720	16	20922789888000
7	5040	17	355687428096000
8	40320	18	6402373705728000
9	362880	19	121645100408832000
10	3628800	20	2432902008176640000

●プログラム１(bi111.c)

```
/* << bi111.c >> */
#include <stdio.h>

int main() {
  int c,    /* ２項係数の値。*/
      i,
      n,r; /* nとr。*/

  while( 1 ) {
    /* n,rの読み込み。*/
    scanf("%d%d",&n,&r);
    if( (n < 0)||(r < 0)||(n < r) ) { break; };

    /* ２項係数の計算。*/
    c = 1;
    for( i=n; i>=n-r+1; i-- ) {
      c = c*i;
    }
    for( i=1; i<=r; i++ ) {
      c = c/i;
    }
```

```
22
23      /* 結果の出力。*/
24      printf("c(%d,%d) = %d ¥n",n,r,c);
25    }
26  }
```

実行結果

```
% cc bi111.c
% ./a.out
10 5
c(10,5) = 252
12 6
c(12,6) = 924
14 7
c(14,7) = 3432
16 8
c(16,8) = 12870
18 9
c(18,9) = 1276
-1 -1
```

正解は48620

２項係数の定義を変形して、次の関係式を使う。

関係式：c(n,r) = (n!/(r!(n-r)!)
= {n(n-1)(n-2)…(n-r+1)}/(1…r)
= (n/1){(n-1)/2}{(n-2)/3}…{(n-r+1)/r}

計算の手順は、つぎのようになる。

（１）nを求め、Xとする。

（２）Xにn-1を掛け、その後2で割り結果を新たにXとする。
連続する２つの正整数の積は2の倍数になるので、2で割り切れる。

（３）Xにn-2を掛け、その後3で割り結果を新たにXとする。
連続する３つの正整数の積は3の倍数になるので、3で割り切れる。

（４）あと同様の操作をn-3からn-r+1まで繰り返す。

この結果、Xの値が求める２項係数c(n,r)となる。

この手順で計算すると、割算をしたとき、常に割り切れる。
この方法は、早めに割り切れる場合は割っておくので、オーバーフローの現象の発生を避けることはできないが、遅らせることができる。

●プログラム２(bi121.c)

```
 1  /* << bi121.c >> */
 2  #include <stdio.h>
 3
 4  int main() {
 5    int c,    /* ２項係数の値。*/
 6        i,
 7        n,r; /* nとr。*/
 8
 9    while( 1 ) {
10      /* n,rの読み込み。*/
11      scanf("%d%d",&n,&r);
12      if( (n < 0)||(r < 0)||(n < r) ) { break; };
13
14      /* ２項係数の計算。*/
15      c = 1;
16      for( i=1; i<=r; i++ ) {
17        c = c*(n-i+1)/i;
18      }
19
20      /* 結果の出力。*/
21      printf("c(%d,%d) = %d ¥n",n,r,c);
22    }
23  }
```

実行結果

```
% cc bi121.c
% ./a.out
24 12
c(24,12) = 2704156
26 13
c(26,13) = 10400600
28 14
c(28,14) = 40116600
30 15
c(30,15) = -131213633
-1 -1
```

正解は155117520

２．開発方法２（配列を用いる方法）

２．１　開発方法２．１（２次元配列を用いる方法）

２項係数 c(n,r) は、つぎのようにも定義される。

$$
\begin{aligned}
c(n,r) &= c(n-1,r) + c(n-1,r-1) && (n \geqq 2, 1 \leqq r \leqq n-1)\\
&= 1 && (n \geqq 0,\ r=0\ \ )\\
&= 1 && (n \geqq 1,\ r=n\ \ )
\end{aligned}
$$

２次元配列cを使って計算する。

n＼r	0	r-1	r	3
n-1				
n				

●計算例

２項係数c(5,3)について、計算過程を示す。

（１）

n＼r	0	1	2	3
1	1	1		
2				
3				
4				
5				

（２）

n＼r	0	1	2	3
1	1	1		
2	1	2	1	
3				
4				
5				

（３）

n＼r	0	1	2	3
1	1	1		
2	1	2	1	
3	1	3	3	1
4				
5				

（４）

n＼r	0	1	2	3
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5				

（５）

n＼r	0	1	2	3
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

この結果、c(5,3)=10を得る。

●プログラム１(bi211.c)

```
 1  /* << bi211.c >> */
 2  #include <stdio.h>
 3  #define N 99 /* nの最大値。*/
 4
 5  int main() {
 6    int c[N+1][N+1], /* ２項係数を保存する配列。*/
 7        i,j,jmax,
 8        n,r;          /* nとr。*/
 9
10    while( 1 ) {
11      /* n,rの読み込み。*/
12      scanf("%d%d",&n,&r);
13      if( (n < 0)||(r < 0)||(n < r) ) { break; };
14
15      /* 初期設定。*/
16      for( i=0; i<=n; i++ ) { c[i][0] = 1; }
17      for( i=1; i<=r; i++ ) { c[i][i] = 1; }
18
19      /* ２項係数の計算。*/
20      for( i=2; i<=n; i++ ) {
21        jmax = i-1;
22        if( r < jmax ) { jmax = r; }
23        for( j=1; j<=jmax; j++ ) {
24          c[i][j] = c[i-1][j] + c[i-1][j-1];
25        }
26      }
27
28      /* 結果の出力。*/
29      printf("c(%d,%d) = %d ¥n",n,r,c[n][r]);
30    }
31  }
```

実行結果

```
% cc bi211.c
% ./a.out
28 14
c(28,14) = 40116600
30 15
c(30,15) = 155117520
32 16
c(32,16) = 601080390
34 17
c(34,17) = -1961361076
-1 -1
```

２．２　開発方法２．２（１次元配列を用いる方法）

改良：２次元配列を使わず、少ない記憶場所（１次元配列）で計算できる。

$(1+x)^n$を展開すると、x^rの係数に２項係数$c(n,r)$が現れる関係を利用する。

$$(1+x)^n = c(n,0) + c(n,1)x^1 + \cdots + c(n,r)x^r + \cdots + c(n,n)x^n$$

$(1+x)^n$の係数$c(n,r)$を求めるのに、$(1+x)^1$から始めて$(1+x)^i$を計算し、
最後に$(1+x)^n$を求める。

$$(1+x)^i = (1+x)^{i-1}(1+x) \quad (1 \leqq i \leqq n)$$

２項係数$c(6,3)$の計算過程を示す。

・$(1+x)^0$から$(1+x)^1$を求める。
$(1+x)^1 = (1+x)^0(1+x)$

1	0	0	0
1	1	0	0

・$(1+x)^1$から$(1+x)^2$を求める。
$(1+x)^2 = (1+x)^1(1+x)$

1	1	0	0
1	2	1	0

・$(1+x)^2$から$(1+x)^3$を求める。
$(1+x)^3 = (1+x)^2(1+x)$

1	2	1	0
1	3	3	1

・$(1+x)^3$から$(1+x)^4$を求める。
$(1+x)^4 = (1+x)^3(1+x)$

1	3	3	1
1	4	6	4

・$(1+x)^4$から$(1+x)^5$を求める。
$(1+x)^5 = (1+x)^4(1+x)$

1	4	6	4
1	5	10	10

・$(1+x)^5$から$(1+x)^6$を求める。
$(1+x)^6 = (1+x)^5(1+x)$

1	5	10	10
1	6	15	20

この結果、$c(6,3)=20$を得る。

●プログラム２(bi221.c)

```
1  /* << bi221.c >> */
2  #include <stdio.h>
3  #define N 99 /* nの最大値。*/
4
5  int main() {
6    int c[N+1], /* ２項係数を保存する配列。*/
7        i,j,
8        n,r;    /* nとr。*/
9
10   while( 1 ) {
11     /* n,rの読み込み。*/
12     scanf("%d%d",&n,&r);
13     if( (n < 0)||(r < 0)||(n < r) ) { break; };
14
15     /* 初期設定。*/
16     c[0] = 1;
17     for( j=1; j<=r; j++ ) { c[j] = 0; }
18
19     /* ２項係数の計算。*/
20     for( i=1; i<=n; i++ ) {
21       for( j=r; j>=1; j-- ) {
22         c[j] = c[j] + c[j-1];
23       }
24     }
25
26     /* 結果の出力。*/
27     printf("c(%d,%d) = %d ¥n",n,r,c[r]);
28   }
29 }
```

実行結果

```
% cc bi221.c
% ./a.out
28 14
c(28,14) = 40116600
30 15
c(30,15) = 155117520
32 16
c(32,16) = 601080390
34 17
c(34,17) = -1961361076
-1 -1
```

３．開発方法３（指数と仮数で表現する方法）

数値を仮数部と指数部に分け仮数部が０以上１未満になるように調整しながら計算を進めると、近似値の計算ができる。

計算式は、

$$c(n,r) = (n/1)\{(n-1)/2\}\{(n-2)/3\}\cdots\{(n-r+1)/r\}$$

を使う。途中の計算は仮数部と指数部に分けて行う。仮数部は、0以上1以下とする。したがって、計算精度の関係から大きいn,rに対してc(n,r)を正確には求められないが、近似値を手軽に計算できる。c(10,5)を例にして、考え方を示す。

$$C(10,5) = (10/1)\times(9/2)\times(8/3)\times(7/4)\times(6/5)$$

$$= \underline{(0.1\times10^{2})}\times(9/2)\times(8/3)\times(7/4)\times(6/5)$$

$$= \underline{(0.45\times10^{2})}\times(8/3)\times(7/4)\times(6/5)$$

$$= \underline{(0.12\times10^{3})}\times(7/4)\times(6/5)$$

$$= \underline{(0.21\times10^{3})}\times(6/5)$$

$$= \underline{(0.252\times10^{3})}$$

●プログラム１(bi311.c)

```
/* << bi311.c >> */
#include <stdio.h>

int main() {
  int i,
      m,    /* 指数部。*/
      n,r; /* nとr。*/
  float f; /* 仮数部。*/

  while( 1 ) {
    /* n,rの読み込み。*/
    scanf("%d%d",&n,&r);
    if( (n < 0)||(n < r)||(r < 0) ) { break; }

    /* 近似値の計算。*/
    f = 1; m = 0;
    for( i=1; i<=r; i++ ) {
      f = f*(n - i + 1)/i;
      while( f >= 1.0 ) {
        f = f/10.0; m++;
      }
```

```
22      }
23
24      /* 結果の出力。*/
25      printf("c(%d,%d) = %8.5f * 10 ** %d¥n",n,r,f,m);
26    }
27  }
```

実行結果

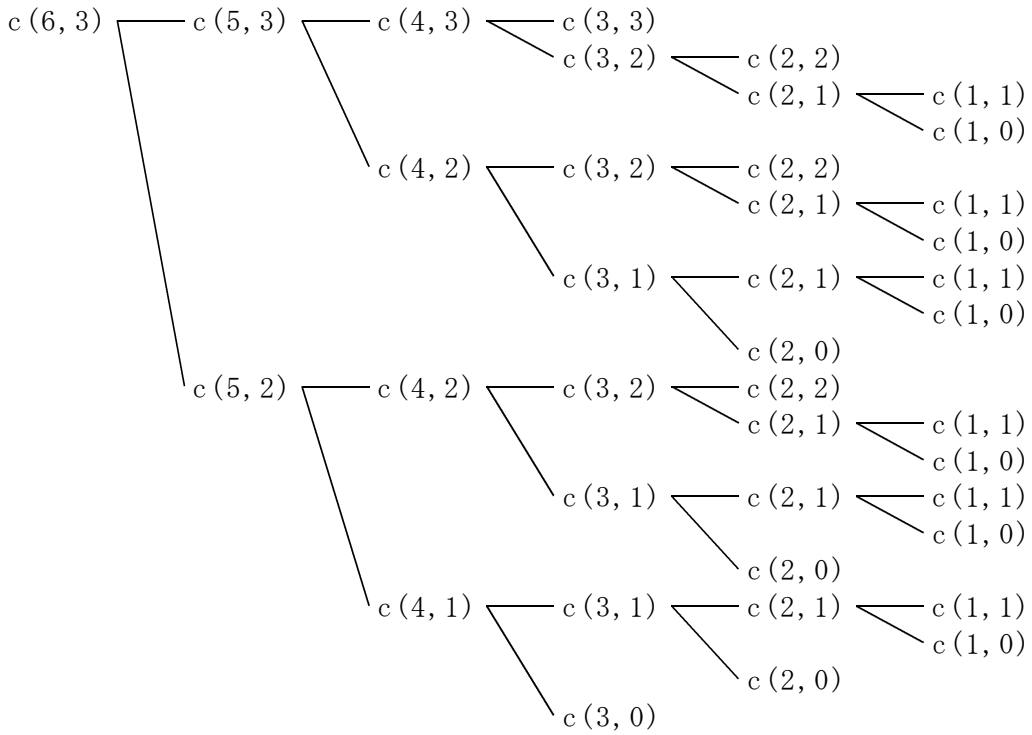
```
% cc bi311.c
% ./a.out
10 5
c(10,5) =  0.25200 * 10 ** 3
100 50
c(100,50) =  0.10089 * 10 ** 30
1000 500
c(1000,500) =  0.27029 * 10 ** 300
10000 5000
c(10000,5000) =  0.15918 * 10 ** 3009
-1 -1
```

４．開発方法４（再帰関数を用いる方法）

２項係数 c(n,r) の定義を利用し、再帰関数を使って計算する。

$$
\begin{aligned}
c(n,r) &= c(n-1,r) + c(n-1,r-1) && (n \geqq 2, 1 \leqq r \leqq n-1)\\
&= 1 && (n \geqq 0,\ r=0\ \ )\\
&= 1 && (n \geqq 1,\ r=n\ \ )
\end{aligned}
$$

n,rが大きいときに、同じ計算を繰り返すことになるので、計算時間がかかる。



●プログラム１(bi411.c)

```
1  /* << bi411.c >> */
2  #include <stdio.h>
3  double c(int n, int r);
4
5  int main() {
6    int n,r;
7
8    while( 1 ) {
9      /* n,rの読み込み。*/
10     scanf("%d%d",&n,&r);
11     if( (n < 0)||(n < r)||(r < 0) ) { break; }
12
13     /* 結果の出力。*/
14     printf("c(%d,%d) = %16.0f¥n",n,r,c(n,r));
15   }
```

```
16 }
17
18 /* 再帰関数。*/
19 double c(int n, int r) {
20   if( (r == 0)||(r == n) ) {
21     return 1;
22   } else {
23     return c(n-1,r)+c(n-1,r-1);
24   }
25 }
```

実行結果

```
% cc bi411.c
% ./a.out
5 0
c(5,0) =                    1
5 1
c(5,1) =                    5
5 2
c(5,2) =                   10
-1 -1
```

実行時間

```
実行時間測定
32 16 -1 -1
c(32,16) =          601080390
6.301u 0.002s 0:12.02 52.4%     0+0k 0+0io 0pf+0w
```

簡単に計算できる部分を計算し、再帰呼び出しを減らし実行時間を短縮する。

●プログラム２(bi412.c)

```
1  /* << bi412.c >> */
2  #include <stdio.h>
3  double c(int n, int r);
4
5  int main() {
6    int n,r;
7
8    while( 1 ) {
9      /* n,rの読み込み。*/
10     scanf("%d%d",&n,&r);
11     if( (n < 0)||(n < r)||(r < 0) ) { break; }
12
13     /* 結果の出力。*/
14     printf("c(%d,%d) = %16.0f¥n",n,r,c(n,r));
```

```
15    }
16  }
17
18  /* 再帰関数。*/
19  double c(int n, int r) {
20    if( (r == 0)||(r == n) ) {
21      return 1;
22    } else if ( (r == 1)||(r == n-1) ) {
23      return n;
24    } else {
25      return c(n-1,r)+c(n-1,r-1);
26    }
27  }
```

実行結果

```
% cc bi412.c
% ./a.out
5 1
c(5,1) =                    5
5 2
c(5,2) =                   10
-1 -1
```

実行時間

```
実行時間測定
% time ./a.out
32 16 -1 -1
c(32,16) =          601080390
2.106u 0.000s 0:08.69 24.1%      0+0k 0+0io 0pf+0w
```

一定の範囲について、計算値を配列に保存しておき、呼び出されたときその値を使い、再計算を行わないで時間を節約する。

●プログラム３(bi413.c)

```
1   /* << bi413.c >> */
2   #include <stdio.h>
3   int z[9][9]; /* ２項係数c(i,j)(0≦i≦8,0≦j≦8)の値を
4                   保存する配列。*/
5   double c(int n, int r);
6
7   int main() {
8     int i,j,n,r;
9
10    while( 1 ) {
11      /* n,rの読み込み。*/
```

```
12      scanf("%d%d",&n,&r);
13      if( (n < 0)||(n < r)||(r < 0) ) { break; }
14
15      /* 初期設定。*/
16      for( i=0; i<9; i++ ) {
17        for( j=0; j<9; j++ ) { z[i][j] = 0; }
18      }
19
20      /* 結果の出力。*/
21      printf("c(%d,%d) = %16.0f¥n",n,r,c(n,r));
22    }
23  }
24
25  /* 再帰関数。*/
26  double c(int n, int r) {
27    int w;
28
29    /* c(n,r)(0≦n≦8,0≦r≦8)で、計算済みの場合は,戻る。*/
30    if( (n < 9)&&(r < 9 )&&(z[n][r] > 0) ) {
31      return z[n][r];
32    }
33
34    if( (r == 0)||(r == n) ) {
35      /* 未計算の場合は、配列に保存。*/
36      if( (n < 9)&&(r < 9 ) ) {
37        z[n][r] = 1;
38      }
39      return 1;
40    } else {
41      w = c(n-1,r)+c(n-1,r-1);
42      /* 未計算の場合は、配列に保存。*/
43      if( (n < 9)&&(r < 9 ) ) {
44        z[n][r] = w;
45      }
46      return w;
47    }
48  }
```

実行結果

```
% cc bi413.c
% ./a.out
5 1
c(5,1) =                    5
12 6
c(12,6) =                 924
-1 -1
```

実行時間

```
実行時間測定
% time ./a.out
32 16 -1 -1
c(32,16) =         601080390
0.211u 0.001s 0:06.97 3.0%      0+0k 0+0io 0pf+0w
```