再帰関数Ⅱ

０．目次

６．２項係数

●再帰的定義

２項係数 c(n,r) は、つぎのように、定義される。

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

c(n,r)	0	1	2	3	4	5
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

関数 c(n,r)は、定義通りに書ける。

●プログラム(rf611.c)

```
1  /* << rf611.c >> */
2  #include <stdio.h>
3  int c(int n, int r);
4
5  int main() {
6    int n,r;
7
8    while( 1 ) {
9      /* 正整数n,rの読み込み。*/
10     scanf("%d%d",&n,&r);
11     if( (n < 0)||(r < 0)||(n < r) ) { break; };
12
13     printf("c(%d,%d) = %d ¥n",n,r,c(n,r));
14   }
15 }
16
17 /* ２項係数の計算。*/
18 int c(int n, int r) {
19   if( (r==0)||(n==r) ) {
20     return 1;
21   } else {
22     return c(n-1,r) + c(n-1,r-1);
23   }
24 }
```

実行結果

```
% cc rf611.c
% ./a.out
3 2
c(3,2) = 3
10 5
c(10,5) = 252
-1 -1
```

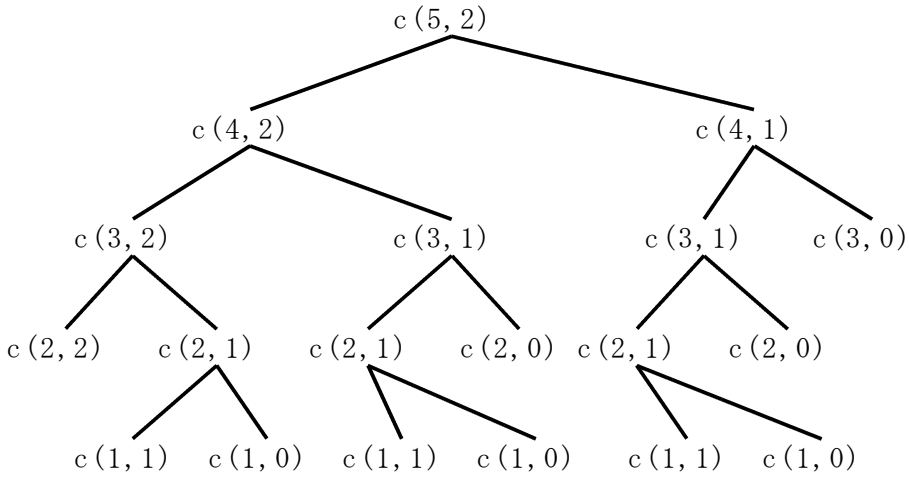
実行時間

```
% cc rf611.c
% ./a.out
28 14 -1 -1
c(28,14)=40116600
0.411u 0.000s 0:07.32 5.6%      0+0k 0+0io 0pf+0w
% ./a.out
30 15 -1 -1
c(30,15)=155117520
1.582u 0.001s 0:08.53 18.5%     0+0k 0+0io 0pf+0w
```

●プログラム(rf611.c)において、n,rが与えられたとき、関数c(n,r)が呼び出される回数

n	r	呼び出される回数
4	1	7
4	2	11
4	3	7
4	4	1

●関数c(n,r)の計算過程。n=5,r=2の場合。



（注意）同じ計算を繰り返すので大きなn,rに対して効率が悪くなる。

7．二分探索

データが配列dに昇順に並べられているとき行われる二分探索を再帰的に定義する。探索データをxとする。

●再帰的定義

search(d[i],…,d[j],x)

= "not found" (i>j)
= "found" (i≦j, x=d[(i+j)/2])
= search(d[i],…,d[(i+j)/2-1],x) (i≦j, x<d[(i+j)/2])
= search(d[(i+j)/2+1],…,d[j],x) (i≦j, x>d[(i+j)/2])

二分探索関数search(i,j)のプログラムは次のように書ける。

●プログラム(rf711.c)

```
1  /* << rf711.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 9999 /* データの個数の最大値。*/
5  int d[N+1], /* データを格納する配列。*/
6      x;      /* 探索データ。*/
7  void search(int i, int j);
8
9  int main() {
10   int i,
11   n;      /* データの個数。*/
12
13   /* データの個数nとデータの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { d[i] = i*10; }
17
18   /* 二分探索。*/
19   while( 1 ) {
20     /* 探索データxの読み込み。*/
21     scanf("%d",&x);
22     if( x <= 0 ) { break; }
23     search(1,n);
24   }
25 }
26
27 /* 再帰的定義 */
28 void search(int i, int j) {
29   int m;
30   if( i > j ) {
31     printf("%dはみつかりません¥n",x); return;
```

```
32    }
33    m = (i+j)/2;
34    if( x == d[m] ) {
35      printf("%dが見つかりました¥n",x); return;
36    }
37    if( x <= d[m] ) {
38      search(i,m-1);
39    } else {
40      search(m+1,j);
41    }
42    return;
43  }
```

実行結果

```
% cc rf711.c
% ./a.out
10 30
30が見つかりました
55
55は見つかりません
0
```

●プログラム(rf711.c)において、100個のデータ 10,20,…,1000が与えられたとき、探索データxを探索するため関数search(i,j)が呼び出される回数

x	呼び出される回数
100	6
500	1
123	7
345	7

●プログラム(rf711.c)の動作。10個のデータ 10,20,…,100。

x=30の場合。

search(1,10)

1	2	3	4	5	6	7	8	9	10
10	20	30	40	50	60	70	80	90	100

↓ m=(1+10)/2=5

search(1,4)

1	2	3	4
10	20	30	40

↓ m=(1+4)/2=2

search(3,4)

3	4
30	40

↓ m=(3+4)/2=3

見つかった

3	4
30	40

x=55の場合。

search(1,10)

1	2	3	4	5	6	7	8	9	10
10	20	30	40	50	60	70	80	90	100

↓ m=(1+10)/2=5

search(6,10)

6	7	8	9	10
60	70	80	90	100

↓ m=(6+10)/2=8

search(6,7)

6	7
60	70

↓ m=(6+7)/2=6

search(6,5)
見つからない

1	2	3	4	5	6	7	8	9	10
10	20	30	40	50	60	70	80	90	100

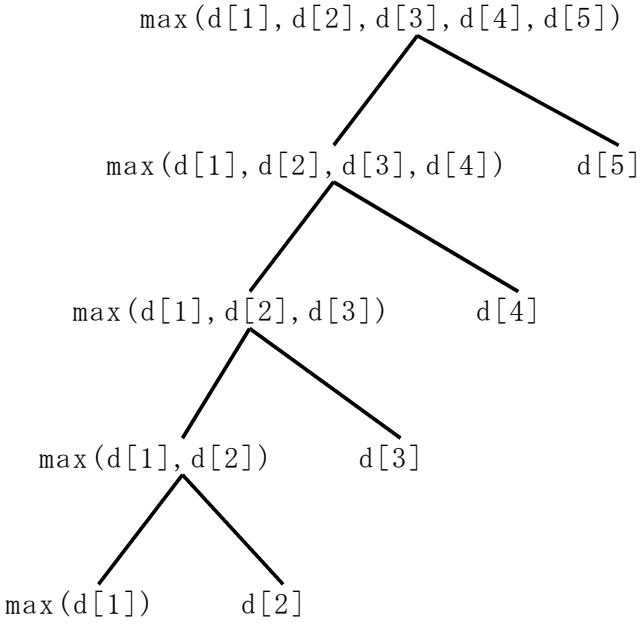
８．最大値探索

ｎ個のデータ中、最大のものを見つける方法を再帰的に考える。

●再帰的定義１

```
max(d[1],…,d[n]) = max(max(d[1],…,d[n-1]),d[n])    (n>1)
                 = d[1]                            (n=1)
```

・n=5の場合。



最大値関数max(i)：d[1]からd[i]までの中から最大の要素を返す。
最大値を求めるプログラムは次のように書ける。

●プログラム(rf811.c)

```
/* << rf811.c >> */
#include <stdio.h>
#include <stdlib.h>
#define N 9999 /* データの個数の最大値。*/
int d[N+1], /* データを格納する配列。*/
int max(int i);

int main() {
  int i,
      n; /* データの個数。*/

  /* データの個数とデータを入力。*/
  scanf("%d",&n);
  if( (n <= 0)||(n > N) ) { exit(0); }
  for( i=1; i<=n; i++ ) { scanf("%d",&d[i]); }

  /* データの出力。*/
  for( i=1; i<=n; i++ ) {
    printf("%5d ",d[i]);
    if( i%10==0 ) { printf("¥n"); }
  }
  printf("¥n最大値：%d ¥n",max(n));
}

/* d[1]からd[i]までの中から最大の要素を返す。*/
int max(int i) {
  int z,z1,z2;
  if( i == 1 ) {
    z = d[1];
  } else {
    z1 = max(i-1);
    z2 = d[i];
    if( z1 > z2 ) {
      z = z1;
    } else {
      z = z2;
    }
  }
  return z;
}
```

実行結果

```
% cc rf811.c
% ./a.out
9
44 55 88 33 99 22 11 66 77
   44    55    88    33    99    22    11    66    77
最大値：99
```

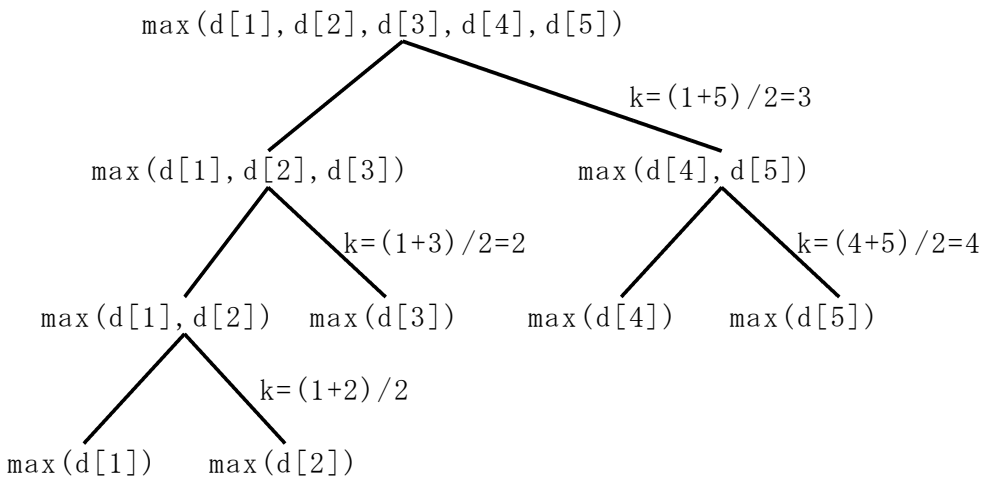
●再帰的定義２

max(d[i],…,d[j])

= max(max(d[i],…,d[k]),max(d[k+1],…,d[j]))　(i<j, kは適当な値。i≦k<j)

= d[i]　(i=j)

・n=5,k=(i+j)/2の場合。



最大値関数max(i,j)：d[i]からd[j]までの中から最大の要素を返す。
最大値を求めるプログラムは次のように書ける。

●プログラム(rf821.c)

```
/* << rf821.c >> */
#include <stdio.h>
#include <stdlib.h>
#define N 9999 /* データの個数の最大値。*/
int d[N+1], /* データを格納する配列。*/
int max(int i, int j);

int main() {
  int i,
      n; /* データの個数。*/

```

```
12    /* データの個数nとデータを入力。*/
13    scanf("%d",&n);
14    if( (n <= 0)||(n > N) ) { exit(0); }
15    for( i=1; i<=n; i++ ) { scanf("%d",&d[i]); }
16
17    /* データの出力。*/
18    for( i=1; i<=n; i++ ) {
19      printf("%5d ",d[i]); if( i%10==0 ) { printf("¥n"); }
20    }
21    printf("¥n");
22    printf("最大値：%d ¥n",max(1,n));
23  }
24
25  /* d[i]からd[j]までの中から最大の要素を返す。*/
26  int max(int i, int j) {
27    int k,z,z1,z2;
28    if( i == j ) {
29      z = d[i];
30    } else {
31      k = (i+j)/2;
32      z1 = max(i,k);
33      z2 = max(k+1,j);
34      if( z1 > z2 ) {
35        z = z1;
36      } else {
37        z = z2;
38      }
39    }
40    return z;
41  }
```

実行結果

```
1  % cc rf821.c
2  % ./a.out
3  9
4  44 55 88 33 99 22 11 66 77
5     44    55    88    33    99    22    11    66    77
6  最大値：99
```

９．集合{1,2,…,n}上の部分集合生成

集合{1,2,…,n}上の部分集合を示す。

n	部分集合 B(n)
1	{},{1}
2	{},{1},{2},{1,2}
3	{},{1},{2},{3},{1,2},{1,3},{2,3},{1,2,3}
4	{},{1},{2},{3},{4},{1,2},{1,3},{1,4},{2,3},{2,4} {3,4},{1,2,3},{1,2,4},{1,3,4},{2,3,4},{1,2,3,4}

●集合{1,2,…,n}上の部分集合の個数。

集合{1,2,…,n}上の部分集合の個数f(n)とおくと、要素1を含む部分集合と要素1を含まない部分集合に分類できる。前者はf(n-1)個の部分集合、後者もf(n-1)個の部分集合を含む。このことから、漸化式

$$f(n) = f(n-1) + f(n-1)\ (n \geqq 2),\ f(1)=2$$

を得る。この漸化式を解くと、

$$f(n) = 2^n (n \geqq 1)$$

となる。

●表現法１

記号0,1からなる長さnの記号列a(1)a(2)…a(n)で部分集合を表す。
ただし、a(i)=0で、要素iが部分集合に含まれない、a(i)=1で、要素iが部分集合に含まれることに対応させる。
記号0,1からなる長さnの記号列a(1)a(2)…a(n)の集合をA(n)とする。

n	部分集合 A(n)
1	0,1
2	00,01,10,11
3	000,001,010,011,100,101,110,111
4	0000,0001,0010,0011,0100,0101,0110,0111, 1000,1001,1010,1011,1100,1101,1110,1111

いくつかの文字列の並び$S=\{s_1, s_2, \cdots, s_p\}$, $T=\{t_1, t_2, \cdots, t_q\}$が与えられたとき、aS, S+Tを次のように定義する。

$$aS = \{as_1, as_2, \cdots, as_p\}$$
$$S+T = \{s_1, s_2, \cdots, s_p, t_1, t_2, \cdots, t_q\}$$

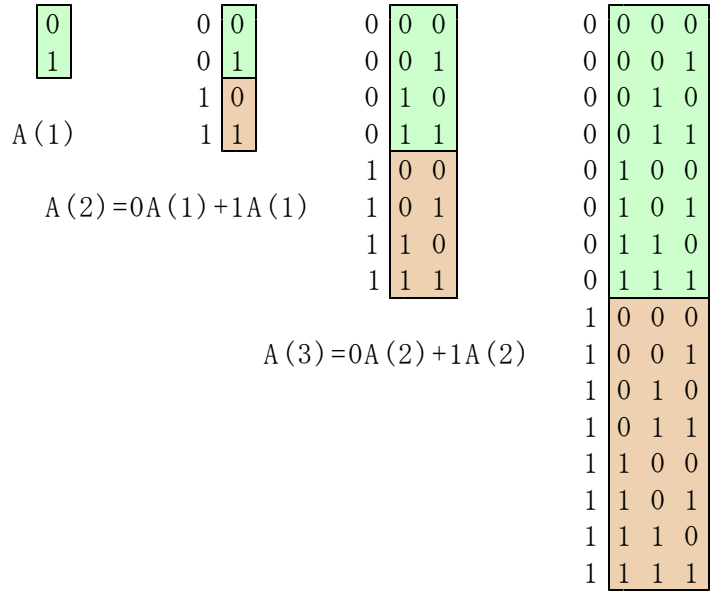
このとき、A(n)は、次のようにも書ける。

$$\begin{cases} A(1) = \{0,1\} \\ A(n) = 0A(n-1) + 1A(n-1) \quad (n \geqq 2) \end{cases}$$

$$\begin{aligned} A(2) &= 0A(1) + 1A(1) \\ &= 0\{0,1\} + 1\{0,1\} \\ &= \{00,01,10,11\} \end{aligned}$$

$$\begin{aligned} A(3) &= 0A(2) + 1A(2) \\ &= 0\{00,01,10,11\} + 1\{00,01,10,11\} \\ &= \{000,001,010,011,100,101,110,111\} \end{aligned}$$

$$\begin{aligned} A(4) &= 0A(3) + 1A(3) \\ &= 0\{000,001,010,011,100,101,110,111\} \\ &+ 1\{000,001,010,011,100,101,110,111\} \\ &= \{0000,0001,0010,0011,0100,0101,0110,0111, \\ &\quad 1000,1001,1010,1011,1100,1101,1110,1111\} \end{aligned}$$



$$\begin{cases} A(1) = \{0,1\} \\ A(n) = 0A(n-1) + 1A(n-1) \quad (n \geqq 2) \end{cases}$$

上記の表現に基づくプログラムは次のように書ける。

●プログラム(rf911.c)

```
1  /* << rf911.c >> */
2  #include <stdio.h>
3  #define N 9 /* 要素数の最大値。*/
4  int a[N+1], /* 部分集合を表現する配列。*/
5      count,  /* 部分集合の個数。*/
6      n;      /* 集合の要素数。*/
7  void sub(int k);
8
9  int main() {
10   while( 1 ) {
11     /* 要素数nの読み込み。*/
12     scanf("%d",&n);
13     if( (n <= 0)||(n > N) ) { break; }
14
15     /* 初期設定。*/
16     count = 0;
17
18     /* 部分集合の生成。*/
19     sub(1);
20     printf("n=%d count=%d¥n",n,count);
21   }
22 }
23
24 /* k番目からn番目までの要素の部分集合を生成する。*/
25 void sub(int k) {
26   int i;
27   if( k > n ) {
28     count++; printf("%d: ",count);
29     for( i=1; i<k; i++ ) {
30       printf("%d ",a[i]);
31     }
32     printf("¥n");
33   } else {
34     /* k番目の要素を含まない部分集合を生成。*/
35     a[k] = 0; sub(k+1);
36     /* k番目の要素を含む部分集合を生成。*/
37     a[k] = 1; sub(k+1);
38   }
39 }
```

実行結果

```
% cc rf911.c
% ./a.out
3
1: 0 0 0
2: 0 0 1
3: 0 1 0
4: 0 1 1
5: 1 0 0
6: 1 0 1
7: 1 1 0
8: 1 1 1
n=3 count=8
0
```

●プログラム(rf1011.c)において、nが与えられたとき、関数subが呼び出される回数

n	呼び出される回数
3	15
4	31
5	63
6	127

●木の表現１

部分集合A(n)を木で表現する。a(i)=0は、深さiで要素iが集合に含まれないことを意味し、a(i)=1は、深さiで要素iが集合に含まれることを意味する。
記号0の子を0,1と並べ、記号1の子も0,1と並べる。