最大値探索法の開発

０．目次

n個のデータの中から最大値を求めるプログラム開発過程と発展問題の開発過程を考察する。

１．開発過程１

目標１：4個のデータの最大値を求める。

●プログラム(m111.c)

```
1  /* << m111.c >> */
2  #include <stdio.h>
3
4  int main() {
5    int a,b,c,d, /* 入力データ。*/
6        max;     /* 最大値。*/
7
8    /* 入力データa,b,c,dの読み込み。*/
9    scanf("%d%d%d%d",&a,&b,&c,&d);
10
11   /* 探索前データの表示。*/
12   printf("探索前：");
13   printf("a=%d b=%d c=%d d=%d¥n",a,b,c,d);
14
15   /* 最大値探索。*/
16   max = a;
17   if( b > max ) { max = b; }
18   if( c > max ) { max = c; }
19   if( d > max ) { max = d; }
20
21   /* 探索後データの表示。*/
22   printf("探索後：");
23   printf("a=%d b=%d c=%d d=%d¥n",a,b,c,d);
24   printf("最大値：%d¥n",max);
25 }
```

実行結果

```
% cc m111.c
% ./a.out
22 44 11 33
探索前：a=22 b=44 c=11 d=33
探索後：a=22 b=44 c=11 d=33
最大値：44
```

（考察）問題のサイズが小さい場合で、処理のコードを書いてみると、繰り返し処理ができる部分等が見えてくる。

目標２：4個のデータの最大値を求める。

改良：多数のデータに対応するため、配列を使う。

●プログラム(m121.c)

```
1  /* << m121.c >> */
2  #include <stdio.h>
3
4  int main() {
5    int a[5], /* 入力データ。*/
6        i,
7        n,    /* データの個数。*/
8        max;  /* 最大値。*/
9
10   /* データの読み込み。*/
11   n = 4;
12   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
13
14   /* 探索前データの表示。*/
15   printf("探索前：");
16   for( i=1; i<=4; i++ ) { printf("%3d",a[i]); }
17   printf("¥n");
18
19   /* 最大値探索。*/
20   max = a[1];
21   if( a[2] > max ) { max = a[2]; }
22   if( a[3] > max ) { max = a[3]; }
23   if( a[4] > max ) { max = a[4]; }
24
25   /* 探索後データの表示。*/
26   printf("探索後：");
27   for( i=1; i<=4; i++ ) { printf("%3d",a[i]); }
28   printf("¥n");
29   printf("最大値：%d¥n",max);
30 }
```

実行結果

```
% cc m121.c
% ./a.out
33 44 11 22
探索前：  33 44 11 22
探索後：  33 44 11 22
最大値：44
```

目標３：n個のデータの最大値を求める。

改良：コードを簡潔に記述するため、for文を使う。

●プログラム(m131.c)

```
 1  /* << m131.c >> */
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #define N 999 /* データ個数の最大値。*/
 5
 6  int main() {
 7    int a[N+1], /* 入力データ。*/
 8        i,
 9        n,      /* データの個数。*/
10        max;    /* 最大値。*/
11
12    /* データの個数nとデータの読み込み。*/
13    scanf("%d",&n);
14    if( (n <= 0)||(n > N) ) { exit(0); }
15    for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
16
17    /* 探索前データの表示。*/
18    printf("探索前：");
19    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
20    printf("¥n");
21
22    /* 最大値探索。*/
23    max = a[1];
24    for( i=2; i<=n; i++ ) {
25      if( a[i] > max ) { max = a[i]; }
26    }
27
28    /* 探索後データの表示。*/
29    printf("探索後：");
30    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
31    printf("¥n");
32    printf("最大値：%d¥n",max);
33  }
```

実行結果

```
% cc m131.c
% ./a.out
8
66 44 22 88 33 11 55 77
探索前：  66 44 22 88 33 11 55 77
探索後：  66 44 22 88 33 11 55 77
最大値：88
```

目標４：n個のデータの最大値を求める。

改良：プログラムをわかりやすくするため、関数を使う。

●プログラム(m141.c)

```
/* << m141.c >> */
#include <stdio.h>
#include <stdlib.h>
#define N 999 /* データ個数の最大値。*/
void max(int x[], int n);

int main() {
  int a[N+1], /* 入力データ。*/
      i,
      n;      /* データの個数。*/

  /* データの個数nとデータの読み込み。*/
  scanf("%d",&n);
  if( (n <= 0)||(n > N) ) { exit(0); }
  for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }

  /* 探索前データの表示。*/
  printf("探索前：");
  for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
  printf("¥n");

  /* 最大値探索。最大値をa[1]に保存する。*/
  max(a,n);

  /* 探索後データの表示。*/
  printf("探索後：");
  for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
  printf("¥n");
  printf("最大値：%d¥n",a[1]);
}

/* 関数max。*/
void max(int x[], int n) {
  int i,w;
  for( i=2; i<=n; i++ ) {
    if( x[i] > x[1] ) {
      w = x[1];
      x[1] = x[i];
      x[i] = w;
    }
  }
}
```

実行結果

```
% cc m141.c
% ./a.out
8
33 77 55 44 11 88 22 66
探索前 :  33 77 55 44 11 88 22 66
探索後 :  88 33 55 44 11 77 22 66
最大値 : 88
```

目標５：n個のデータの最大値を求める。

改良：処理を速くするため、最大値の位置のみを求め、データの移動は最後に１回行う。移動するデータ量が多いときに有効。

●プログラム(m151.c)

```
 1  /* << m151.c >> */
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  #define N 999 /* データ個数の最大値。*/
 5  void max(int x[], int n);
 6
 7  int main() {
 8    int a[N+1], /* 入力データ。*/
 9        i,
10        n;      /* データの個数。*/
11
12    /* データの個数nとデータの読み込み。*/
13    scanf("%d",&n);
14    if( (n <= 0)||(n > N) ) { exit(0); }
15    for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
16
17    /* 探索前データの表示。*/
18    printf("探索前 : ");
19    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
20    printf("¥n");
21
22    /* 最大値探索。最大値をa[1]に保存する。*/
23    max(a,n);
24
25    /* 探索後データの表示。*/
26    printf("探索後 : ");
27    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
28    printf("¥n");
29    printf("最大値 : %d¥n",a[1]);
30  }
31
```

```
32 /* 関数max。*/
33 void max(int x[], int n) {
34   int i,
35       j, /* 最大値が保存されている配列x内の位置。*/
36       w;
37   j = 1;
38   for( i=2; i<=n; i++ ) {
39     if( x[i] > x[j] ) {
40       /* 最大値の位置を保存する。*/
41       j = i;
42     }
43   }
44   /* データの移動。*/
45   w = x[1]; x[1] = x[j]; x[j] = w;
46 }
```

実行結果

```
% cc m151.c
% ./a.out
8
44 66 33 77 88 55 11 22
探索前： 44 66 33 77 88 55 11 22
探索後： 88 66 33 77 44 55 11 22
最大値：88
```

２．開発過程２

問題を一般化していく。

目標１：n個のデータから、最大値とその次の値を求める。
方法：a[1]に1番目のデータを格納、a[2]に2番目のデータを格納する。

●プログラム(m211.c)

```
1  /* << m211.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 999 /* データ個数の最大値。*/
5  void max(int x[], int n);
6
7  int main() {
8    int a[N+1], /* 入力データ。*/
9        i,
10       n;       /* データの個数。*/
11
12   /* データの個数nとデータの読み込み。*/
13   scanf("%d",&n);
14   if( (n <= 0)||(n > N) ) { exit(0); }
15   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
16
17   /* 探索前データの表示。*/
18   printf("探索前：");
19   for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
20   printf("¥n");
21
22   /* n個のデータから、最大値とその次の値を探索。*/
23   max(a,n);
24
25   /* 探索後データの表示。*/
26   printf("探索後：");
27   for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
28   printf("¥n");
29 }
30
31 /* 関数max。*/
32 void max(int x[], int n) {
33   int i,j,w;
34
35   /* １番目のデータを求め、x[1]に格納。*/
36   j = 1;
37   for( i=2; i<=n; i++ ) {
38     if( x[i] > x[j] ) { j = i; }
```

```
39    }
40    w = x[1];
41    x[1] = x[j];
42    x[j] = w;
43
44    /* ２番目のデータを求め、x[2]に格納。*/
45    j = 2;
46    for( i=3; i<=n; i++ ) {
47      if( x[i] > x[j] ) { j = i; }
48    }
49    w = x[2];
50    x[2] = x[j];
51    x[j] = w;
52  }
```

実行結果

```
% cc m211.c
% ./a.out
8
44 33 55 22 88 66 11 77
探索前：  44 33 55 22 88 66 11 77
探索後：  88 77 55 22 44 66 11 33
```

目標２：n個のデータの上位からm番目まで求める。

方法：a[1]に1番目のデータを格納、a[2]に2番目のデータを格納、…、
a[m]にm番目のデータを格納する。

●プログラム(m221.c)

```
1  /* << m221.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 999 /* データ個数の最大値。*/
5  void max(int x[], int n, int m);
6
7  int main() {
8    int a[N+1], /* 入力データ。*/
9        i,
10       m,      /* 上位からm番目。*/
11       n;      /* データの個数。*/
12
13   /* データの個数nとm番目とデータの読み込み。*/
14   scanf("%d%d",&n,&m);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
```

```
17
18    /* 探索前データの表示。*/
19    printf("n=%d m=%d¥n",n,m);
20    printf("探索前：");
21    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
22    printf("¥n");
23
24    /* n個のデータの上位からm番目まで探索。*/
25    max(a,n,m);
26
27    /* 探索後データの表示。*/
28    printf("探索後：");
29    for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
30    printf("¥n");
31  }
32
33  /* 関数max。*/
34  void max(int x[], int n, int m) {
35    int i,j,k,w;
36
37    /* k番目のデータを求め、x[k]に格納。*/
38    for( k=1; k<=m; k++ ) {
39      j = k;
40      for( i=j+1; i<=n; i++ ) {
41        if( x[i] > x[j] ) { j = i; }
42      }
43      w = x[k];
44      x[k] = x[j];
45      x[j] = w;
46    }
47  }
```

実行結果

```
% cc m221.c
% ./a.out
8 3
33 88 55 77 11 22 44 66
n=8 m=3
探索前： 33 88 55 77 11 22 44 66
探索後： 88 77 66 33 11 22 44 55
```

３．開発過程３

与えられた問題の条件を緩くすると(「１番目からm番目まで」→「上位のm個」)、新たな解法が見えてくる。

目標１：n個のデータから上位のm個を求める。
準備：n個のデータ中、適当な値z以上の要素の集まりと、z以下の要素の集まりに分割する。

●考え方

12個のデータ：77 22 66 44 55 44 22 44 11 44 11 22　で考察する。

ほぼ中央(6番目)のデータ(44)に着目して、44以上のデータを配列の先頭に、44以下のデータを配列の末尾に分割する。

(手順０)範囲を1番目から12番目までとする。

77 22 66 44 55 44 22 44 11 44 11 22

(手順１)1番目から44以下のデータが見つかるまで右端方向に探索する。

77 22 66 44 55 44 22 44 11 44 11 22

(手順２)12番目から44以上のデータが見つかるまで左端方向に探索する。

77 22 66 44 55 44 22 44 11 44 11 22

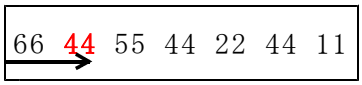
(手順３)見つかった両データを交換する。

77 44 66 44 55 44 22 44 11 22 11 22

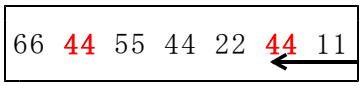
(手順４)範囲を3番目から9番目までとする。

77 44 66 44 55 44 22 44 11 22 11 22

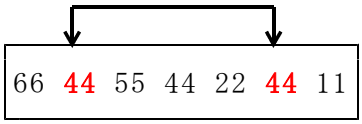
（手順１）3番目から44以下のデータが見つかるまで右端方向に探索する。



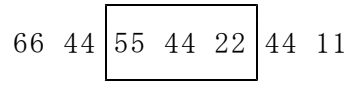
（手順２）9番目から44以上のデータが見つかるまで左端方向に探索する。



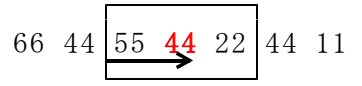
（手順３）見つかった両データを交換する。



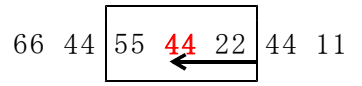
（手順４）範囲を5番目から7番目までとする。



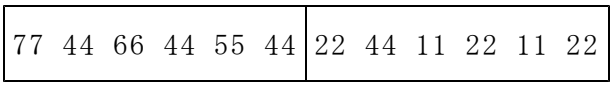
（手順１）5番目から44以下のデータが見つかるまで右端方向に探索する。



（手順２）7番目から44以上のデータが見つかるまで左端方向に探索する。



（手順３）見つかった両データが同じ位置なので終了する。



1番目から6番目まで44以上、7番目から12番目まで44以下。

●プログラム(m311.c)

```
1  /* << m311.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #define N 999 /* データ個数の最大値。*/
5  int div(int x[], int left, int right);
6
7  int main() {
8    int a[N+1], /* 入力データ。*/
9        i,
10       n,       /* データの個数。*/
11       p,
12       z;       /* ほぼ中央の位置のデータ。*/
13
14   /* データの個数nとm番目とデータの読み込み。*/
15   scanf("%d",&n);
16   if( (n <= 0)||(n > N) ) { exit(0); }
17   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
18
19   /* 分割前データの表示。*/
20   printf("分割前 : ");
21   for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
22   printf("¥n");
23
24   /* 分割。ほぼ中央の位置のデータzを用いて分割し、
25            左半分の右端の位置pを求める。*/
26   z = a[(1+n)/2];
27   p = div(a,1,n);
28
29   /* 分割後データの表示。*/
30   printf("分割後(%d以上) : ",z);
31   for( i=1; i<=p; i++ ) { printf("%3d",a[i]); } printf("¥n");
32   printf("分割後(%d以下) : ",z);
33   for( i=1; i<=p; i++ ) { printf("   "); }
34   for( i=p+1; i<=n; i++ ) { printf("%3d",a[i]); } printf("¥n");
35 }
36
37 /* 関数 : x[left]～x[right]の中で、x[left]～x[k]がx[(left+right)/2]
38          以上となる位置kを返す。*/
39 int div(int x[], int left, int right) {
40   int b,i,j,w;
41
42   /* 手順０。*/
43   i = left; j = right; /* x[i]～x[j]が探索範囲。*/
44   b = x[(i+j)/2];
45   while( 1 ) {
46
47     /* 手順１。*/
48     while( x[i] > b ) { i = i+1; }
```

```
49
50      /* 手順２。*/
51      while( x[j] < b ) { j = j-1; }
52      if( i >= j ) { break; }
53
54      /* 手順３。*/
55      w = x[i]; x[i] = x[j]; x[j] = w;
56
57      /* 手順４。*/
58      i++; j--;
59    }
60
61    /* i≠j の場合、x[n0]～x[i-1]がb以上である。
62       i=j の場合、x[n0]～x[i]がb以上である。*/
63    if( i == j ) { return i; } else { return i-1; }
64  }
```

実行結果

```
% cc m311.c
% ./a.out
1 11
分割前： 11
分割後(11以上)： 11
分割後(11以下)：
% ./a.out
2 11 11
分割前： 11 11
分割後(11以上)： 11
分割後(11以下)：    11
% ./a.out
2 11 22
分割前： 11 22
分割後(11以上)： 22
分割後(11以下)：    11
% ./a.out
2 22 11
分割前： 22 11
分割後(22以上)： 22
分割後(22以下)：    11
% ./a.out
6 11 22 22 33 33 33
分割前： 11 22 22 33 33 33
分割後(22以上)： 33 33 33
分割後(22以下)：          22 22 11
```

目標２：n個のデータ中、上位のm個を求める。

方法：a[1]～a[m]に1番目からm番目のデータを格納する。
必ずしも、a[1]≧a[2]≧…≧a[m]とならないことに注意。

●考え方

n=12 m=5 とする。
探索前： 11 22 33 44 55 33 22 44 55 44 11 22

与えられたデータから大きい順に5個の要素を選ぶ必要がある。
ほぼ中央(6番目)のデータ(33)に着目して、33以上のデータと33以下のデータのグループに分割する。

分割前 | 11 22 33 44 55 33 22 44 55 44 11 22 |

分割後 | 44 55 44 44 55 33 | 22 33 22 11 11 22 |

33以上のグループの要素が6個となり、5個より多いので、
33以上のグループから大きい順に5個の要素を選ぶ。
そこで、33以上のグループのほぼ中央(3番目)のデータ(44)に着目し、
44以上のデータと44以下のデータのグループに分割する。

分割前 | 44 55 44 44 55 33 |

分割後 | 55 55 44 | 44 44 33 |

44以上のグループの要素が3個となり、5個より少ないので、
44以下のグループから大きい順に2個の要素を選ぶ。
そこで、44以下のグループの先頭から2番目のデータ(44)に着目し、
44以上のデータと44以下のデータのグループに分割する。

分割前 | 44 44 33 |

分割後 | 44 | 44 33 |

44以上のグループの要素が1個となり、2個より少ないので、
44以下のグループから大きい順に1個の要素を選ぶ。
そこで、44以下のグループの先頭から1番目のデータ(44)に着目し、
44以上のデータと44以下のデータのグループに分割する。

分割前 | 44 | 33 |

分割後 | 44 | 33 |

44以上のグループの要素が1個となり、ちょうど5個選ばれた。

●プログラム(m321.c)

```
/* << m321.c >> */
#include <stdio.h>
#include <stdlib.h>
#define N 999 /* データ個数の最大値。*/
int div(int x[], int left, int right);

int main() {
  int a[N+1], /* 入力データ。*/
      i,
      k0,k1,k2,
      m,
      n,       /* データの個数。*/
      p;

  /* データ個数とm番目とデータの読み込み。*/
  scanf("%d%d",&n,&m);
  if( (n <= 0)||(n > N)||(n < m) ) { exit(0); }
  for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }

  /* 分割前データの表示。*/
  printf("分割前 : ");
  for( i=1; i<=n; i++ ) { printf("%3d",a[i]); }
  printf("¥n");

  /* 大きいものからm個を選ぶ。*/
  k0 = 1; k1 = n; k2 = m;
  while ( 1 ) {
    p = div(a,k0,k1);
    if( p-k0+1 == k2 ) { break; }
    if( p-k0+1 < k2 ) {
      k2 = k2 - (p-k0+1);
      k0 = p+1;
    } else {
      k1 = p-1;
    }
```

```
36    }
37
38    /* 選択後データの表示。*/
39    printf("選択後(m=%d):",m);
40    for( i=1; i<=m; i++ ) { printf("%3d",a[i]); }
41    printf(" --- ");
42    for( i=m+1; i<=n; i++ ) { printf("%3d",a[i]); } printf("¥n");
43  }
44
45  /* 関数:x[left]～x[right]の中で、x[left]～x[k]がx[(left+right)/2]
46              以上となる位置kを返す。*/
47  int div(int x[], int left, int right) {
48
49  <<   省略   >>
50
51  }
```

実行結果

```
% cc m321.cc
% ./a.out
6 1 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=1):  33 ---  33 33 22 22 11

% ./a.out
6 2 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=2):  33 33 ---  33 22 22 11

% ./a.out
6 3 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=3):  33 33 33 ---  22 22 11

% ./a.out
6 4 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=4):  33 33 33 22 ---  22 11

% ./a.out
6 5 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=5):  33 33 33 22 22 ---  11

% ./a.out
6 6 11 22 22 33 33 33
選択前:  11 22 22 33 33 33
選択後(m=6):  33 33 33 22 22 11 ---
```