配列処理Ⅱ

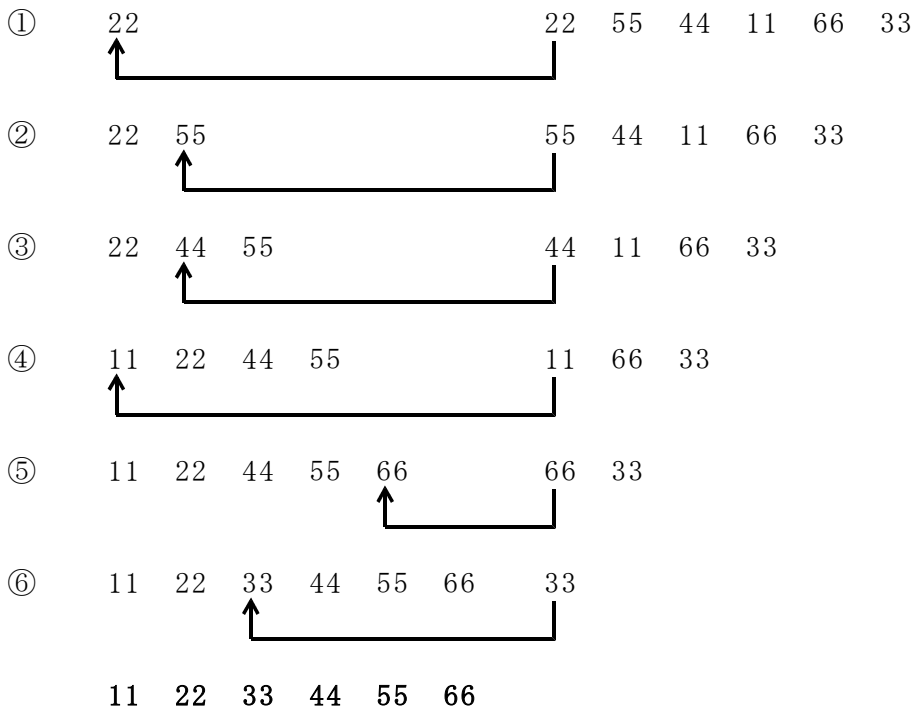
０．目次

６．ソート

６．１　挿入ソート

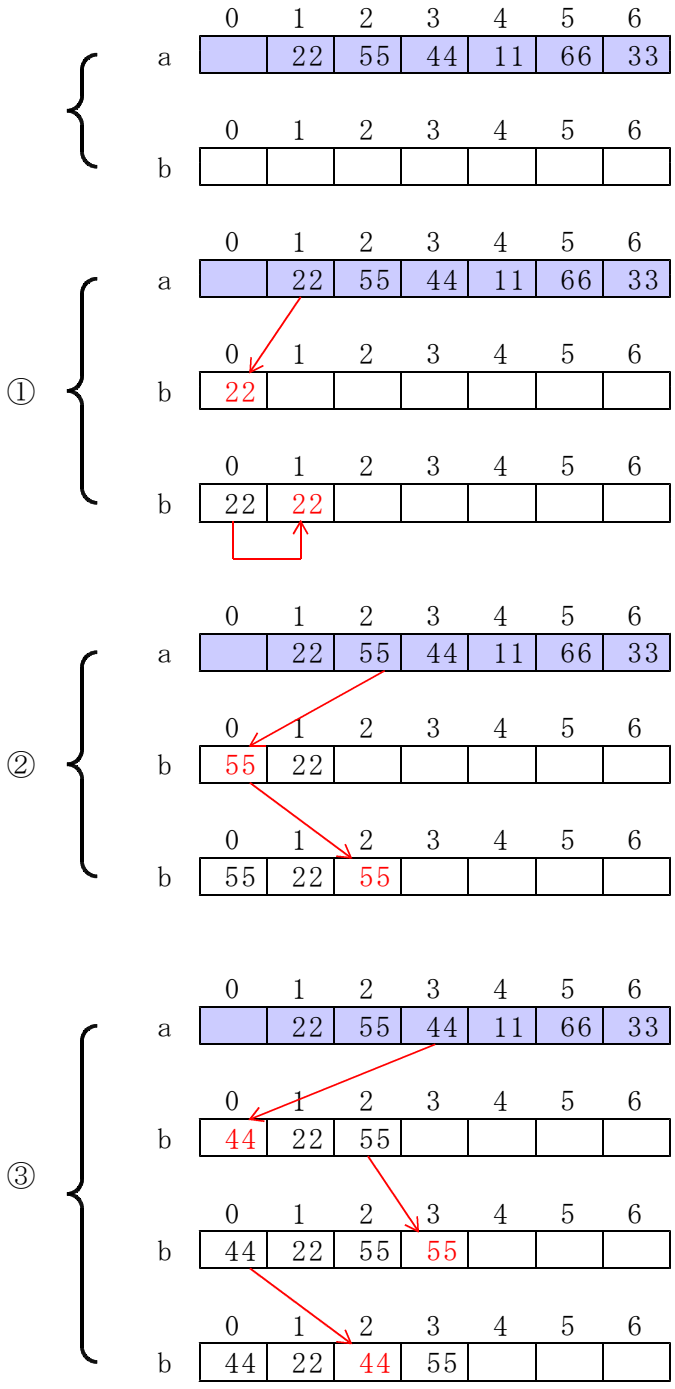
配列から要素を１つずつ取り出し、昇順に並べられている部分列の昇順を保存しながら適当な位置に挿入していく。

●考え方

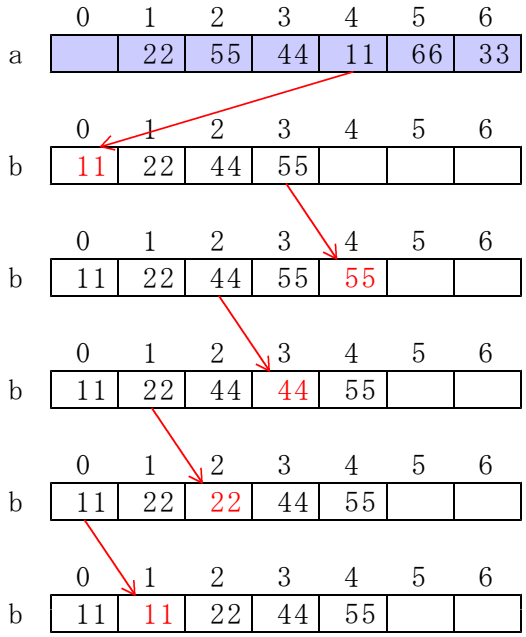


● 2つの配列を使う処理

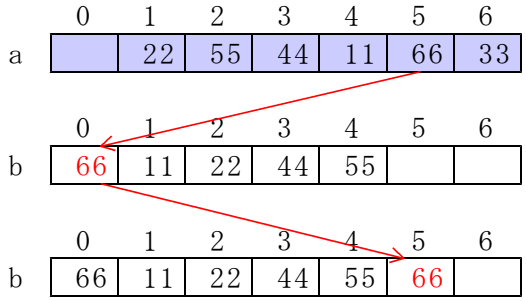
配列a：入力用、配列b：出力用



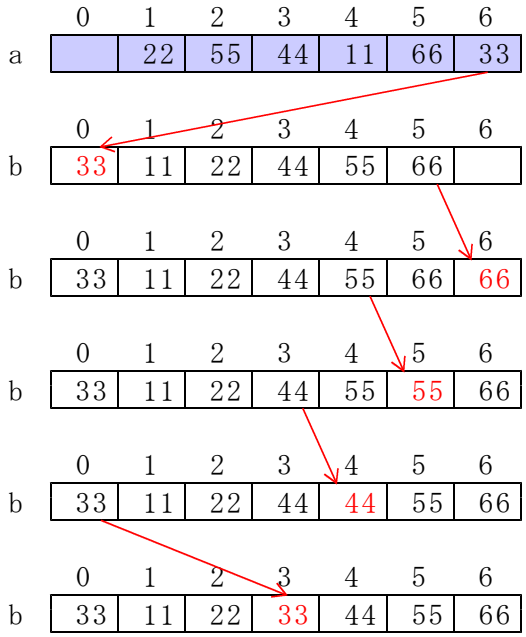
④


0 1 2 3 4 5 6
a 22 55 44 11 66 33
0 1 2 3 4 5 6
b 11 22 44 55
0 1 2 3 4 5 6
b 11 22 44 55 55
0 1 2 3 4 5 6
b 11 22 44 44 55
0 1 2 3 4 5 6
b 11 22 22 44 55
0 1 2 3 4 5 6
b 11 11 22 44 55

⑤


0 1 2 3 4 5 6
a 22 55 44 11 66 33
0 1 2 3 4 5 6
b 66 11 22 44 55
0 1 2 3 4 5 6
b 66 11 22 44 55 66

⑥


0 1 2 3 4 5 6
a 22 55 44 11 66 33
0 1 2 3 4 5 6
b 33 11 22 44 55 66
0 1 2 3 4 5 6
b 33 11 22 44 55 66 66
0 1 2 3 4 5 6
b 33 11 22 44 55 55 66
0 1 2 3 4 5 6
b 33 11 22 44 44 55 66
0 1 2 3 4 5 6
b 33 11 22 33 44 55 66

●プログラム(ar611.c)

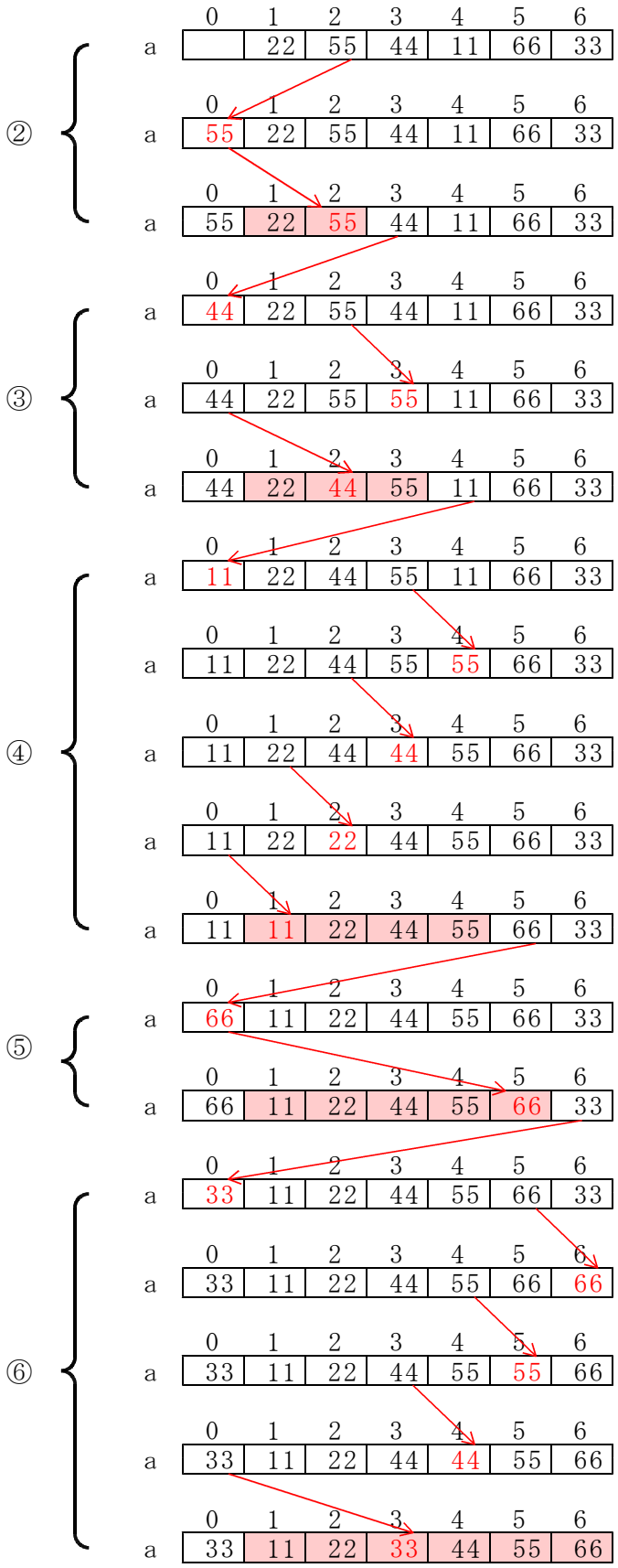
```
/* << ar611.c >> */
/* 挿入ソート */
#include <stdio.h>
#include <stdlib.h>
#define N 10000 /* データの最大個数。*/

int main() {
  int a[N+1], /* 入力用配列。*/
      b[N+1], /* 出力用配列。*/
      i,j,
      n;      /* データの個数。*/

  /* データの個数nと配列a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]); }

  /* ソート前配列の表示。*/
  printf("ソート前 : ");
  for( i=1; i<=n; i++ ) { printf("%4d ",a[i]); }
  printf("¥n");

  /* 挿入処理。*/
  for( i=1; i<=n; i++ ) {
    b[0] = a[i];
    j = i - 1;
    while( b[j] > b[0] ) {
      b[j+1] = b[j];
      j--;
    }
    b[j+1] = b[0];
  }

  /* ソート後配列の表示。*/
  printf("ソート後 : ");
  for( i=1; i<=n; i++ ) { printf("%4d ",b[i]); }
  printf("¥n");
}
```

実行結果

```
% cc ar611.c
% ./a.out
9
33 55 77 22 99 88 11 66 44
ソート前 :   33   55   77   22   99   88   11   66   44
ソート後 :   11   22   33   44   55   66   77   88   99
```

● 1 つの配列を使う処理

配列a：入力用、出力用



●プログラム(ar612.c)

```
/* << ar612.c >> */
/* 挿入ソート */
#include <stdio.h>
#include <stdlib.h>
#define N 100000 /* データの最大個数。*/

int main() {
  int a[N+1], /* 入出力用配列。*/
      i,j,
      n;      /* データの個数。*/

  /* データの個数nと配列a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]); }

  /* ソート前、配列の表示。*/
  printf("ソート前 : ");
  for( i=1; i<=n; i++ ) { printf("%4d ",a[i]); }
  printf("¥n");

  /* 挿入処理。*/
  for( i=2; i<=n; i++ ) {
    a[0] = a[i];
    j = i - 1;
    while( a[j] > a[0] ) {
      a[j+1] = a[j];
      j--;
    }
    a[j+1] = a[0];
  }

  /* ソート後、配列の表示。*/
  printf("ソート後 : ");
  for( i=1; i<=n; i++ ) { printf("%4d ",a[i]); }
  printf("¥n");
}
```

実行結果

```
% cc ar612.c
% ./a.out
9
33 55 77 22 99 88 11 66 44
ソート前：   33   55   77   22   99   88   11   66   44
ソート後：   11   22   33   44   55   66   77   88   99
```

実行時間（出力の時間を除く）

配列a[i](1≦i≦n)が、a[i] = (11*i)%1024 と定義されている場合

```
% cc ar612.c
% ./a.out
10000
n=10000
0.073u 0.001s 0:02.62 2.6%      0+0k 0+0io 0pf+0w
% ./a.out
100000
n=100000
7.287u 0.004s 0:11.62 62.6%     0+0k 0+8io 0pf+0w
```

６．２　シェルソート

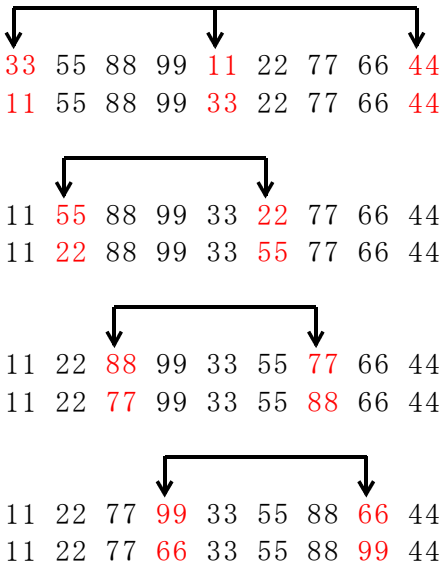
挿入ソートの一種で、要素の間隔が一定のものについてあらかじめ整列させておく。ある程度整列された状態で、最後に間隔１で整列させる。

（手順１）h[1]番目ごとの要素で整列する。
（手順２）h[2]番目ごとの要素で整列する。
・・・
（手順ｍ）h[m]番目ごとの要素で整列する。

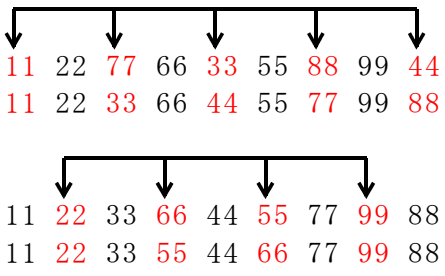
ただし、h[1]>h[2]>…>h[m](=1)

●例　データ数：9　データ：33 55 88 99 11 22 77 66 44

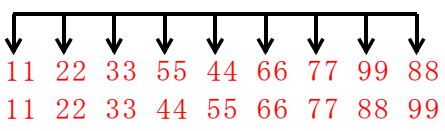
（手順１）h[1]=9/2=4



（手順２）h[2]=9/4=2



（手順３）h[3]=9/8=1



●プログラム(ar621.c)

```
1  /* << ar621.c >> */
2  /* シェルソート */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define N 100000 /* データの最大個数。*/
6
7  int main() {
8    int a[N+1], /* データ保存用配列。*/
9    h,i,j,k,
10   n,          /* データの個数。*/
11   w;
12
13   /* データの個数nと配列aの読み込み。*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* ソート前、配列の表示。*/
19   printf("ソート前 : ");
20   for( i=1; i<=n; i++ ) { printf("%d ",a[i]); };
21   printf("¥n");
22
23   /* シェルソート。 h=n/2,n/4,n/8,...,1 とする。*/
24   /* 間隔hを小さくしていく。*/
25   h = n/2;
26   while( h > 0 ) {
27     for( i=1; i<=h; i++ ) {
28       /* 各グループiに付いて挿入法を適用する。*/
29       for( j=i+h; j<=n; j=j+h ) {
30         w = a[j];
31         k = j - h;
32         while( k > 0 ) {
33           if( a[k] <= w ) { break; }
34           a[k+h] = a[k];
35           k = k - h;
36         }
37         a[k+h] = w;
38       }
39     }
40     h = h/2;
41   }
42
43   /* ソート後、配列の表示。*/
44   printf("ソート後 : ");
45   for( i=1; i<=n; i++ ) { printf("%d ",a[i]); };
46   printf("¥n");
47 }
```

実行結果

```
% cc ar621.c
% ./a.out
9
33 55 88 99 11 22 77 66 44
ソート前：33 55 88 99 11 22 77 66 44
ソート後：11 22 33 44 55 66 77 88 99
```

実行時間（出力の時間を除く）

配列a[i](1≦i≦n)が、a[i] = (11*i)%1024 と定義されている場合

```
% cc ar621.c
% ./a.out
10000
n=10000
0.002u 0.001s 0:03.03 0.0%      0+0k 0+0io 0pf+0w
% ./a.out
100000
n=100000
0.023u 0.000s 0:03.56 0.5%      0+0k 0+0io 0pf+0w
```

６．３　分配ソート

データが整数２桁というように限定されているとき、それぞれの出現回数を調べることにより、並べ替えができる。

●例　正整数１桁の場合。

整列前：４ ４ １ １ ９ １ ８ ８ ４ ５ ４ ６ ８ ３ ６

出現回数を調べる。

値	出現回数
1	3
2	0
3	1
4	5
5	1
6	2
7	0
8	3
9	1

整列後：１ １ １ ３ ４ ４ ４ ４ ４ ５ ６ ６ ８ ８ ８ ９

となる。

●プログラム(ar631.c)

```
1  /* << ar631.c >> */
2  /* 分配ソート */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #define N 100000 /* データの最大個数。*/
6
7  int main() {
8    int a[N+1], /* 入力データ用配列。*/
9        d[9999], /* 出現回数を保存する配列（4桁の整数）。*/
10       i,j,k,
11       n;       /* データの個数。*/
12
13   /* データの個数nと配列aの読み込み（正整数２桁）*/
14   scanf("%d",&n);
15   if( (n <= 0)||(n > N) ) { exit(0); }
16   for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
17
18   /* ソート前、配列の表示。*/
19   printf("ソート前：");
20   for( i=1; i<=n; i++ ) { printf("%d ",a[i]); }
21   printf("¥n");
```

```
22
23    /* 分配ソート。*/
24    for( j=1; j<=9999; j++ ) {
25      d[j] = 0;
26    }
27    for( i=1; i<=n; i++ ) {
28      d[a[i]]++;
29    }
30
31    /* ソート後、配列の表示。*/
32    printf("ソート後：");
33    for( j=1; j<=9999; j++ ) {
34      for( k=1; k<=d[j]; k++ ) { printf("%d ",j); }
35    }
36    printf("¥n");
37  }
```

実行結果

```
% cc ar631.c
% ./a.out
13
44 22 55 33 11 88 99 66 55 77 44 55 33
整列前：44 22 55 33 11 88 99 66 55 77 44 55 33
整列後：11 22 33 33 44 44 55 55 55 66 77 88 99
```

実行時間（出力の時間を除く）

配列a[i](1≦i≦n)が、a[i] = (11*i)%1024 と定義されている場合

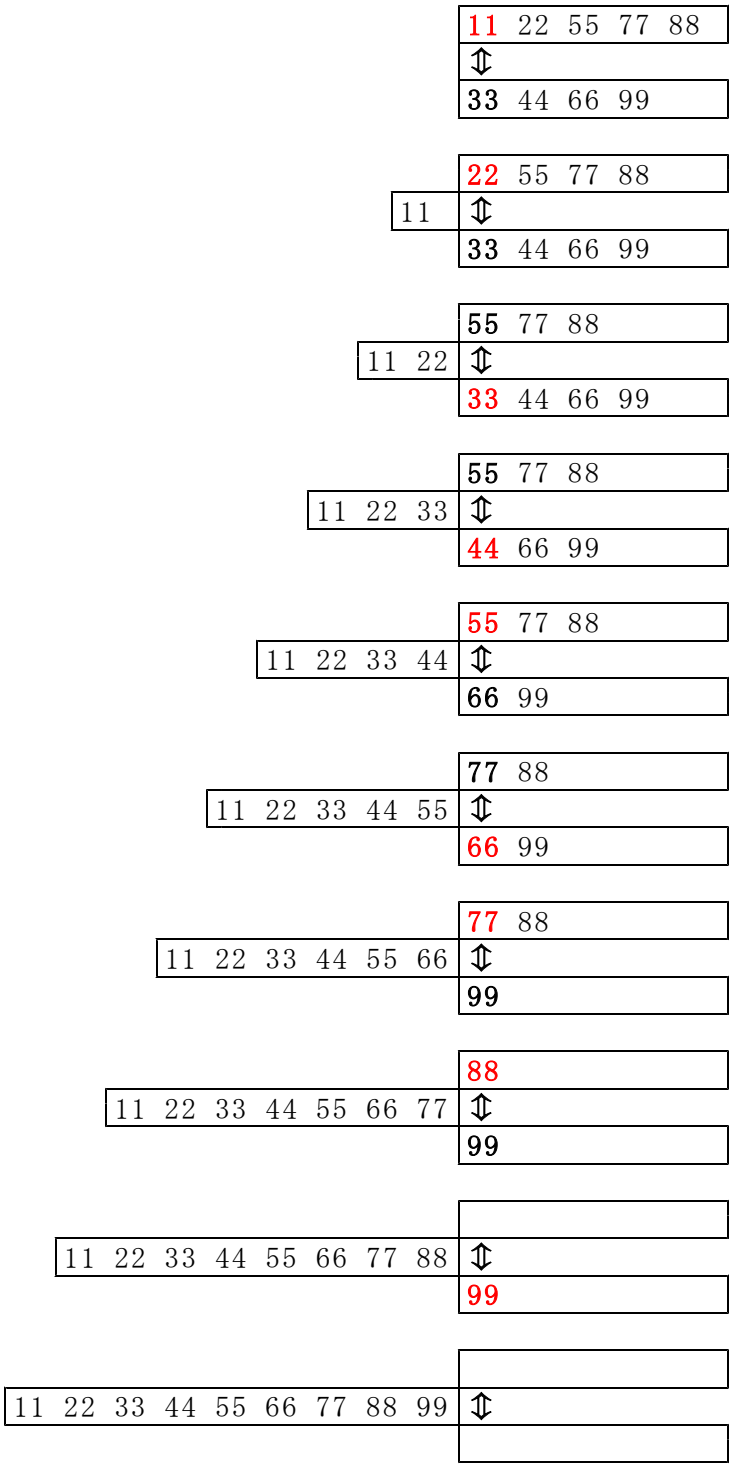
```
% cc ar631.c
% ./a.out
10000
n=10000
0.000u 0.000s 0:02.55 0.0%      0+0k 0+8io 0pf+0w
% ./a.out
100000
n=100000
0.001u 0.000s 0:03.51 0.0%      0+0k 0+0io 0pf+0w
```

7．マージ

２つのソートされたリストをソートされた１つのリストにまとめる。
a[1]≦a[2]≦a[3]≦a[4]≦a[5]≦a[6]≦a[7]≦a[8]≦a[9] と仮定する。

マージ前：a[1]，a[2]，a[5]，a[7]，a[8]
　　a[3]，a[4]，a[6]，a[9]

マージ後：a[1]，a[2]，a[3]，a[4]，a[5]，a[6]，a[7]，a[8]，a[9]



●プログラム(ar711.c)

```
/* << ar711.c >> */
/* マージ */
#include <stdio.h>
#include <stdlib.h>
#define N 1000 /* データの最大個数。*/

int main() {
  int a[N+1],b[N+1], /* マージ前の配列。*/
      c[2*N+1],      /* マージ後の配列。*/
      i,j,k,
      na,            /* 配列aのデータの個数。*/
      nb,            /* 配列bのデータの個数。*/
      nc,            /* 配列cのデータの個数。*/
      x;

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

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

  /* 配列a,bの先頭要素を比較して小さいものを取り出す。
     ncはマージ後のデータの個数。*/

  /* 初期設定。*/
  i = 1; j = 1; nc = 0;

  /* マージ処理。*/
  while( 1 ) {
    /* 配列aが空になったときの処理。*/
    if( i > na ) {
      while( j <= nb ) {
        nc++; c[nc] = b[j]; j++;
      }
      break;
    }

    /* 配列bが空になったときの処理。*/
    if( j > nb ) {
      while( i <= na ) {
        nc++; c[nc] = a[i]; i++;
      }
      break;
    }
```

```
49
50      /* その他の場合。*/
51      if( a[i] <= b[j] ) {
52        nc++; c[nc] = a[i]; i++;
53      } else {
54        nc++; c[nc] = b[j]; j++;
55      }
56    }
57
58    /* 結果（配列c）の表示。*/
59    for( k=1; k<=nc; k++ ) { printf("%d ",c[k]); }
60    printf("¥n");
61  }
```

実行結果

```
% cc ar711.c
% ./a.out
5
11 22 55 77 88
4
33 44 66 99
11 22 33 44 55 66 77 88 99
```