ヒープ

０．目次

１．ヒープとは

２．ヒープ生成法

２．１　具体例
２．２　ヒープ生成法の手順
２．３　ヒープ作成・再帰版
２．４　ヒープ作成・非再帰版

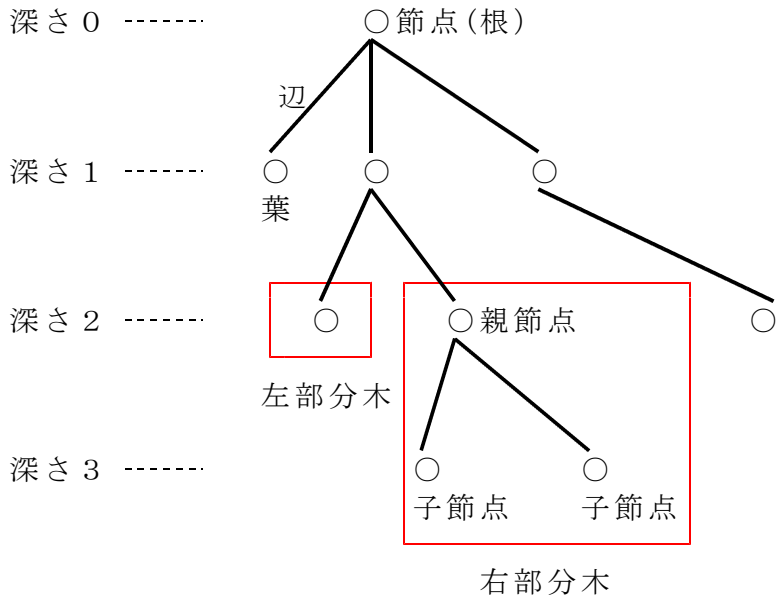
３．ヒープソート（昇順）・再帰版

４．問題

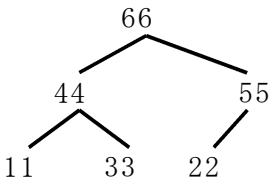
問題１
問題２　ヒープソート（降順）・再帰版

１．ヒープとは

ヒープは、完全２分木で、親節点のデータが子節点のデータよりも常に値が大きい性質を持つ。



完全木とは、葉の深さが高々１の差になっている木をいう。
２分木とは、どの節点も高々２個の子節点をもつ。
また、すべての深さで、データが左から詰められている。
たとえば、下図はヒープである。

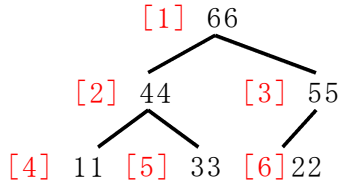


（注意）根のデータが最大値となる。

ヒープは配列で表現できる。

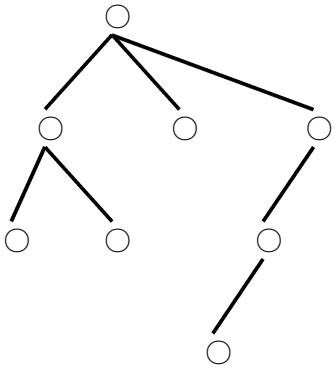
配列表現

i	1	2	3	4	5	6
a[i]	66	44	55	11	33	22

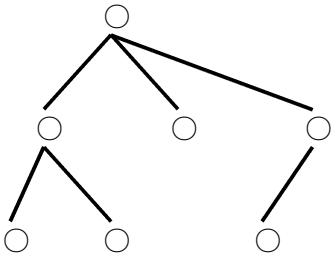


一般に、1≦i≦n/2に対して、a[i]≧a[2i]，a[i]≧a[2i+1] となる。

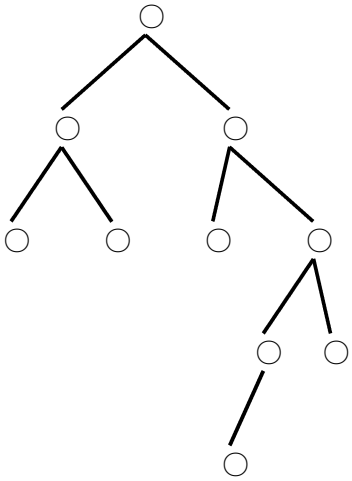
2 分木	×
完全木	×
木	○



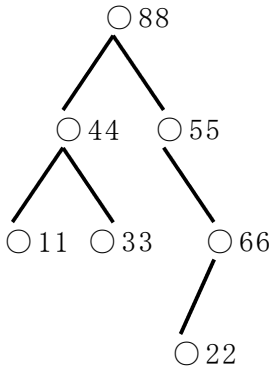
2 分木	×
完全木	○
木	○



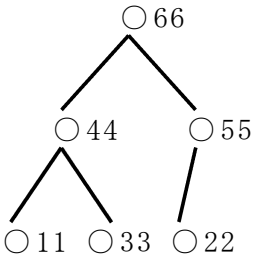
2 分木	○
完全木	×
木	○



ヒープ	×
2 分木	○
完全木	○
木	○



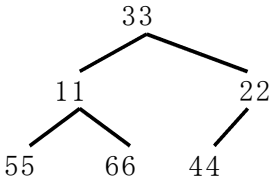
ヒープ	○
2 分木	○
完全木	○
木	○



２．ヒープ生成法

２．１　具体例

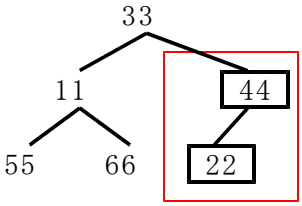
33,11,22,55,66,44を配列（a[1],a[2],…,a[6]）に読込み、ヒープを実現するには、(a),(b),(c),(d)の手順を行う。



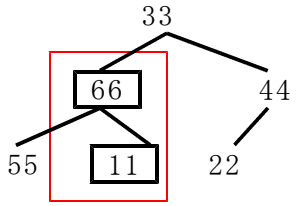
i	1	2	3	4	5	6
a[i]	33	11	22	55	66	44

a[4]からa[6]までは葉であることからそのままにして、a[3],a[2],a[1]の順に親節点と子節点の大小関係を調整していく。

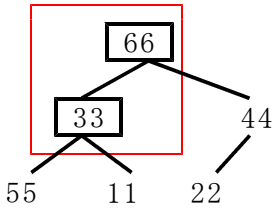
(a) a[3]に着目し22と44を交換



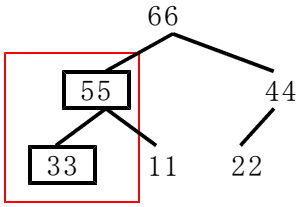
(b) a[2]に着目し11と66を交換



(c) a[1]に着目し33と66を交換



(d) (c)の結果、a[2]の調整が必要になり33と55を交換



i	1	2	3	4	5	6
a[i]	66	55	44	33	11	22

２．２　ヒープ生成法の手順

●手順

① n個のデータを配列a[1]からa[n]に格納する。

② a[n]からa[n/2+1]までは、葉であることからそのままにする。
a[n/2],a[n/2-1],…,a[1]の順に親節点と子節点の調整を行う。

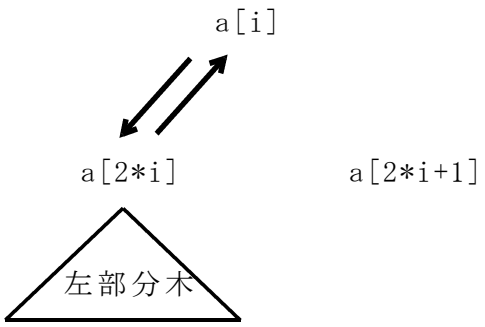
親節点と子節点の調整(a[i],a[2*i],a[2*i+1])は次の３つの場合がある。

（場合１）親節点(a[i])が最大の場合
a[i]>a[2*i], a[i]>a[2*i+1]

調整終了

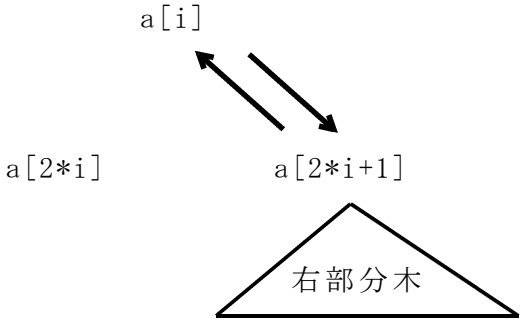
（場合２）子節点(a[2*i])が最大の場合
a[2*i]>a[2*i+1]、a[2*i]>a[i]

親節点(a[i])と子節点(a[2*i])を交換する。
左部分木について親節点と子節点の調整を続ける。



（場合３）子節点(a[2*i+1])が最大の場合
a[2*i+1]>a[2*i]、a[2*i+1]>a[i]

親節点(a[i])と子節点(a[2*i+1])を交換する。
右部分木について親節点と子節点の調整を続ける。



まとめると、

```
if( a[2*i] > a[2*i+1] ) {
  if( a[2*i] > a[i] ) {
    /* 場合２。*/
    w=a[i]; a[i]=a[2*i]; a[2*i]=w; /* 左部分木で交換。*/
    /* つぎの調整に進む。*/
  } else {
    /* 場合１。*/
    /* 調整停止。*/
  }
} else {
  if( a[2*i+1] > a[i] ) {
    /* 場合３。*/
    w=a[i]; a[i]=a[2*i+1]; a[2*i+1]=w; /* 右部分木で交換。*/
    /* つぎの調整に進む。*/
  } else {
    /* 場合１。*/
    /* 調整停止。*/
  }
}
```

となる。整理すると、

```
if( a[2*i] < a[2*i+1] ) {
  k = 2*i+1;
} else {
  k = 2*i;
}
if( a[i] > a[k] ) { /* 場合１。*/ break; }
/* 場合２，３。*/
w = a[i]; a[i] = a[k]; a[k] = w;
/* つぎの調整に進む。*/
```

となる。

２．３　ヒープ作成・再帰版

●プログラム（hp231.c）

```
1  /* << hp231.c >> */
2  #include <stdio.h>
3  #define N 100 /* nの最大値。*/
4  int a[N+1];   /* データを保存する配列。*/
5  void change(int i, int j);
6
7  int main() {
8    int i,
9        n; /* データの個数。*/
10
11   while( 1 ) {
12     /* データの個数nの読み込み。*/
13     scanf("%d",&n);
14     if( (n <= 0)||(n > N) ) { break; }
15
16     /* 数値データの入力と表示。*/
17     for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
18     printf("ヒープ前：");
19     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
20     printf("¥n");
21
22     /* ヒープ化 */
23     for( i=n/2; i>=1; i-- ) {
24       change(i,n);
25     }
26
27     /* 結果出力 */
28     printf("ヒープ後：");
29     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
30     printf("¥n");
31   }
32 }
33
34 /* a[i]からa[j]までヒープ化する。*/
35 void change(int i, int j) {
36   int k,w;
37
38   /* 子節点がない場合。*/
39   if( 2*i > j ) { return; }
40
41   /* 子節点がひとつの場合。*/
42   if( 2*i == j ) {
43     if( a[i] < a[2*i] ) {
44       w = a[i]; a[i] = a[2*i]; a[2*i] = w;
45     }
```

```
46      return;
47    }
48
49    /* 子節点が２つの場合。*/
50    if( a[2*i] < a[2*i+1] ) {
51      k = 2*i+1;
52    } else {
53      k = 2*i;
54    }
55    /* 場合１。*/
56    if( a[i] > a[k] ) { return; }
57    /* 場合２，３。*/
58    w = a[i]; a[i] = a[k]; a[k] = w;
59    change(k,j);
60  }
```

出力結果

```
% cc hp231.c
% ./a.out
8
88 77 66 55 44 33 22 11
ヒープ前：  88 77 66 55 44 33 22 11
ヒープ後：  88 77 66 55 44 33 22 11
8
11 22 33 44 55 66 77 88
ヒープ前：  11 22 33 44 55 66 77 88
ヒープ後：  88 55 77 44 11 66 33 22
9
55 66 44 77 33 88 22 99 11
ヒープ前：  55 66 44 77 33 88 22 99 11
ヒープ後：  99 77 88 66 33 44 22 55 11
9
11 22 33 44 55 99 88 77 66
ヒープ前：  11 22 33 44 55 99 88 77 66
ヒープ後：  99 77 88 66 55 33 11 44 22
0
```

２．４　ヒープ作成・非再帰版

●プログラム（hp241.c）

```
1  /* << hp241.c >> */
2  #include <stdio.h>
3  #define N 100 /* nの最大値。*/
4  int a[N+1];   /* データを保存する配列。*/
5  void change(int i, int j);
6
7  int main() {
8    int i,
9        n; /* データの個数。*/
10
11   while( 1 ) {
12     /* データの個数nの読み込み。*/
13     scanf("%d",&n);
14     if( (n <= 0)||(n > N) ) { break; }
15
16     /* 数値データの入力。*/
17     for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
18     printf("ヒープ前：");
19     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
20     printf("¥n");
21
22     /* ヒープ化 */
23     for( i=n/2; i>=1; i-- ) {
24       change(i,n);
25     }
26
27     /* 結果出力 */
28     printf("ヒープ後：");
29     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
30     printf("¥n");
31   }
32 }
33
34 /* a[i]からa[j]までヒープ化する。*/
35 void change(int i, int j) {
36   int k,w;
37
38   while( 1 ) {
39     /* 子節点がない場合。*/
40     if( 2*i > j ) { break; }
41
42     /* 子節点がひとつの場合。*/
43     if( 2*i == j ) {
44       if( a[i] < a[2*i] ) {
45         w = a[i]; a[i] = a[2*i]; a[2*i] = w;
```

```
46        }
47        break;
48      }
49
50      /* 子節点が２つの場合。*/
51      if( a[2*i] < a[2*i+1] ) {
52        k = 2*i+1;
53      } else {
54        k = 2*i;
55      }
56      /* 場合１。*/
57      if( a[i] > a[k] ) { break; }
58      /* 場合２，３。*/
59      w = a[i]; a[i] = a[k]; a[k] = w;
60      i = k;
61    }
62  }
```

出力結果

```
% cc hp241.c
% ./a.out
8
88 77 66 55 44 33 22 11
ヒープ前： 88 77 66 55 44 33 22 11
ヒープ後： 88 77 66 55 44 33 22 11
8
11 22 33 44 55 66 77 88
ヒープ前： 11 22 33 44 55 66 77 88
ヒープ後： 88 55 77 44 11 66 33 22
9
22 44 66 88 11 33 55 77 99
ヒープ前： 22 44 66 88 11 33 55 77 99
ヒープ後： 99 88 66 77 11 33 55 22 44
9
11 66 22 77 33 88 44 99 55
ヒープ前： 11 66 22 77 33 88 44 99 55
ヒープ後： 99 77 88 66 33 22 44 11 55
0
```

３．ヒープソート（昇順）・再帰版

ヒープを使って効率よくデータを昇順に並べ替えることができる。

（手順１）配列a[1],…,a[n]にデータを読み込む。

（手順２）配列a[1],…,a[n]をヒープにする。i=nとする。

（手順３）a[1]には最大値が保存されるので、a[i]と交換する。

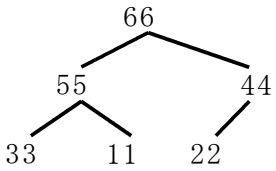
（手順４）配列a[1],…,a[i-1]をヒープにする。iを1減らす。

（手順５）iが1になるまで手順３，４を繰り返す。
最終的に、a[1]≦a[2]≦…≦a[n]と昇順のデータが得られる。

（手順１） 33,22,11,55,66,44を配列（a[1],a[2],…,a[6]）に読込む。

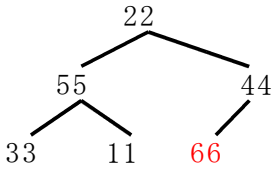
i	1	2	3	4	5	6
a[i]	33	22	11	55	66	44

（手順２） ヒープを実現する。



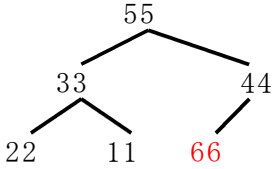
i	1	2	3	4	5	6
a[i]	66	55	44	33	11	22

（手順３） a[1]には最大値が保存されるので、a[6]と交換する。



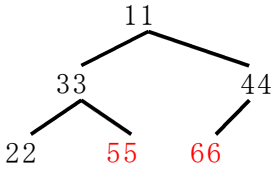
i	1	2	3	4	5	6
a[i]	22	55	44	33	11	66

（手順４） 配列a[1],…,a[5]をヒープにする。



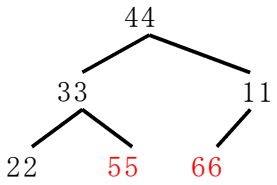
i	1	2	3	4	5	6
a[i]	55	33	44	22	11	66

（手順３） a[1]には最大値が保存されるので、a[5]と交換する。



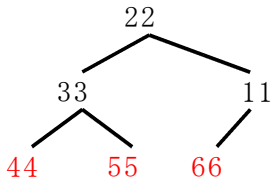
i	1	2	3	4	5	6
a[i]	11	33	44	22	55	66

（手順４） 配列a[1],…,a[4]をヒープにする。



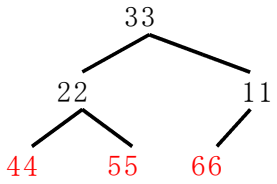
i	1	2	3	4	5	6
a[i]	44	33	11	22	55	66

（手順３） a[1]には最大値が保存されるので、a[4]と交換する。



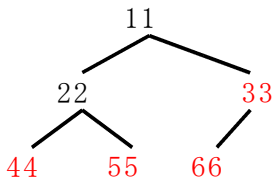
i	1	2	3	4	5	6
a[i]	22	33	11	44	55	66

（手順４） 配列a[1],…,a[3]をヒープにする。



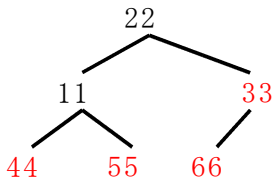
i	1	2	3	4	5	6
a[i]	33	22	11	44	55	66

（手順３） a[1]には最大値が保存されるので、a[3]と交換する。



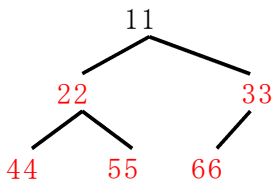
i	1	2	3	4	5	6
a[i]	11	22	33	44	55	66

（手順４） 配列a[1],a[2]をヒープにする。



i	1	2	3	4	5	6
a[i]	22	11	33	44	55	66

（手順３） a[1]には最大値が保存されるので、a[2]と交換する。



i	1	2	3	4	5	6
a[i]	11	22	33	44	55	66

●プログラム（hp311.c）

```
1  /* << hp311.c >> */
2  /* 昇順にソート。*/
3  #include <stdio.h>
4  #define N 100 /* nの最大値。*/
5  int a[N+1];   /* データを保存する配列。*/
6  void change(int i, int j);
7
8  int main() {
9    int i,
10       n, /* データの個数。*/
11       w;
12
13   while( 1 ) {
14     /* 手順１：データの個数nの読み込み。*/
15     scanf("%d",&n);
16     if( (n <= 0)||(n > N) ) { break; }
17
18     /* 数値データの入力と表示。*/
19     for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
20     printf("ソート前：");
21     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
22     printf("¥n");
23
24     /* 手順２：ヒープ生成。*/
25     for( i=n/2; i>=1; i-- ) {
26       change(i,n);
27     }
28
29     /* ソート */
30     for( i=n; i>=2; i-- ) {
31       /* 手順３：a[1]とa[i]を交換。*/
32       w = a[1]; a[1] = a[i]; a[i] = w;
33       /* 手順４：a[1]からa[i-1]をヒープにする。*/
34       change(1,i-1);
35     }
36
37     /* 結果出力。*/
38     printf("ソート後：");
39     for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
40     printf("¥n");
41   }
42 }
43
44 /* a[i]からa[j]までヒープ化する。*/
45 void change(int i, int j) {
46   int k,w;
47
48   /* 子節点がない場合。*/
```

```
49    if( 2*i > j ) { return; }
50
51    /* 子節点がひとつの場合。*/
52    if( 2*i == j ) {
53      if( a[i] < a[2*i] ) {
54        w = a[i]; a[i] = a[2*i]; a[2*i] = w;
55      }
56      return;
57    }
58
59    /* 子節点が２つの場合。*/
60    if( a[2*i] < a[2*i+1] ) {
61      k = 2*i+1;
62    } else {
63      k = 2*i;
64    }
65    /* 場合１。*/
66    if( a[i] > a[k] ) { return; }
67    /* 場合２，３。*/
68    w = a[i]; a[i] = a[k]; a[k] = w;
69    change(k,j);
70  }
```

出力結果

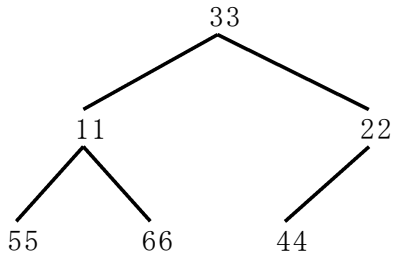
```
% cc hp311.c
% a.out
9
11 22 33 44 55 66 77 88 99
ソート前：  11 22 33 44 55 66 77 88 99
ソート後：  11 22 33 44 55 66 77 88 99
9
99 88 77 66 55 44 33 22 11
ソート前：  99 88 77 66 55 44 33 22 11
ソート後：  11 22 33 44 55 66 77 88 99
9
55 66 44 77 33 88 22 99 11
ソート前：  55 66 44 77 33 88 22 99 11
ソート後：  11 22 33 44 55 66 77 88 99
0
```

４．問題

問題１

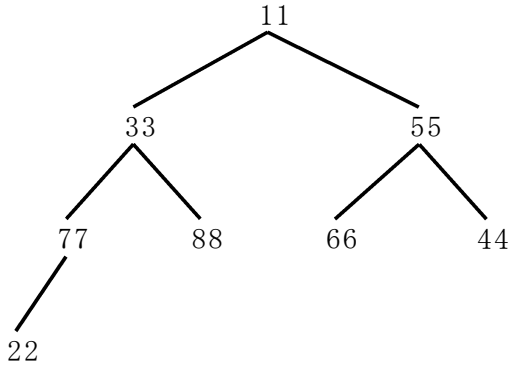
データを配列に読込み、ヒープを作成せよ。

（１）入力データ：33 11 22 55 66 44



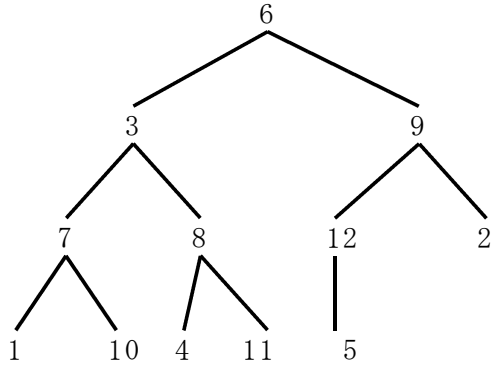
①

（２）入力データ：11 33 55 77 88 66 44 22



②

（３）入力データ：6 3 9 7 8 12 2 1 10 4 11 5



③

問題２　ヒープソート（降順）・再帰版

ヒープを使って、降順にソートするプログラム（再帰版）を書け。

●プログラム（hp412.c）

```
/* << hp412.c >> */
/* 降順にソート。*/
#include <stdio.h>
#define N 100 /* nの最大値。*/
int a[N+1];   /* データを保存する配列。*/
void change(int i, int j);

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

  while( 1 ) {
    /* 手順１：データの個数nの読み込み。*/
    scanf("%d",&n);
    if( (n <= 0)||(n > N) ) { break; }

    /* 数値データの入力と表示。*/
    for( i=1; i<=n; i++ ) { scanf("%d",&a[i]); }
    printf("ソート前：");
    for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
    printf("¥n");

    /* 手順２：ヒープ生成。*/
    for( i=n/2; i>=1; i-- ) {
      change(i,n);
    }

    /* ソート */
    for( i=n; i>=2; i-- ) {
      /* 手順３：a[1]とa[i]を交換。*/
      w = a[1]; a[1] = a[i]; a[i] = w;
      /* 手順４：a[1]からa[i-1]をヒープにする。*/
      change(1,i-1);
    }

    /* 結果出力。*/
    printf("ソート後：");
    for( i=1; i<=n; i++ ) { printf(" %d",a[i]); }
    printf("¥n");
  }
}
```

```
44 /* a[i]からa[j]までヒープ化する。*/
45 void change(int i, int j) {
46   int k,w;
47
48   /* 子節点がない場合。*/
49   if( 2*i > j ) { return; }
50
51   /* 子節点がひとつの場合。*/
52   if( 2*i == j ) {
53     if( ④                    ) {
54       w = a[i]; a[i] = a[2*i]; a[2*i] = w;
55     }
56     return;
57   }
58
59   /* 子節点が２つの場合。*/
60   if( ⑤                      ) {
61     k = 2*i+1;
62   } else {
63     k = 2*i;
64   }
65   if( ⑥              ) { return; }
66   w = a[i]; a[i] = a[k]; a[k] = w;
67   change(k,j);
68 }
```

出力結果

```
% cc hp412.c
% ./a.out
9
77 88 99 66 55 44 11 22 33
ソート前： 77 88 99 66 55 44 11 22 33
ソート後： 99 88 77 66 55 44 33 22 11
9
11 99 22 88 33  77 44  66 55
ソート前： 11 99 22 88 33 77 44 66 55
ソート後： 99 88 77 66 55 44 33 22 11
9
11 22 33 44 55 66 77 88 99
ソート前： 11 22 33 44 55 66 77 88 99
ソート後： 99 88 77 66 55 44 33 22 11
9
99 88 77 66 55 44 33 22 11
ソート前： 99 88 77 66 55 44 33 22 11
ソート後： 99 88 77 66 55 44 33 22 11
0
```