第11回 ヒープ

アルゴリズムとデータ構造ならびに同演習

柏原　昭博

akihiro.kashihara@inf.uec.ac.jp

出席・レポートなど

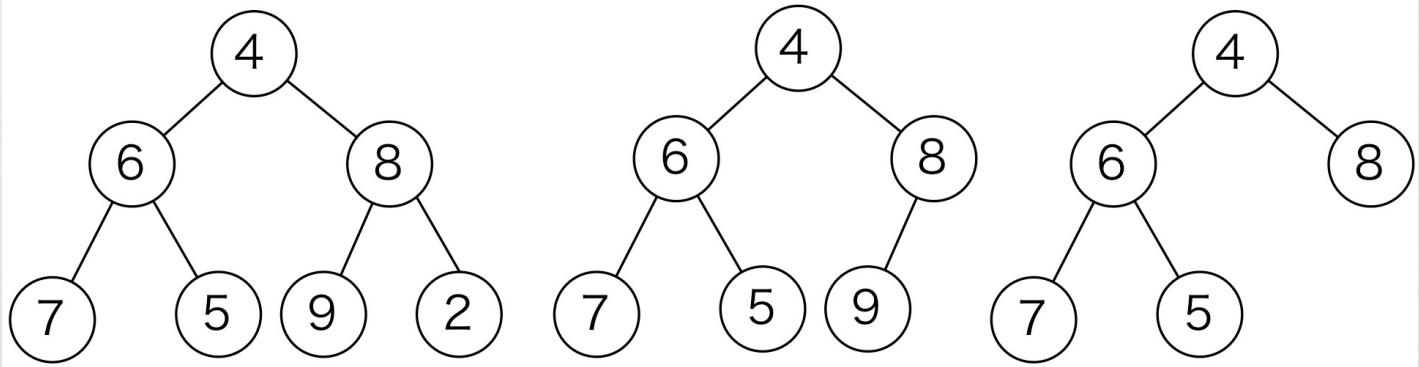
- 教材：スライド資料
- 出席：WebClassで登録
- レポート提出：WebClassに提出
  - cプログラム中に以下をコメントとして含める
    - 課題番号，提出日，学籍番号，氏名
    - 実行例
    - 感想

講義内容

- □　第11回目　ヒープ
- □　第12回目　ソート（整列）
- □　第13回目　クイックソート・その他のソート
- □　第14回目　文字列処理（基本）
- □　第15回目　文字列処理（配列処理・文字探索）

ヒープとは

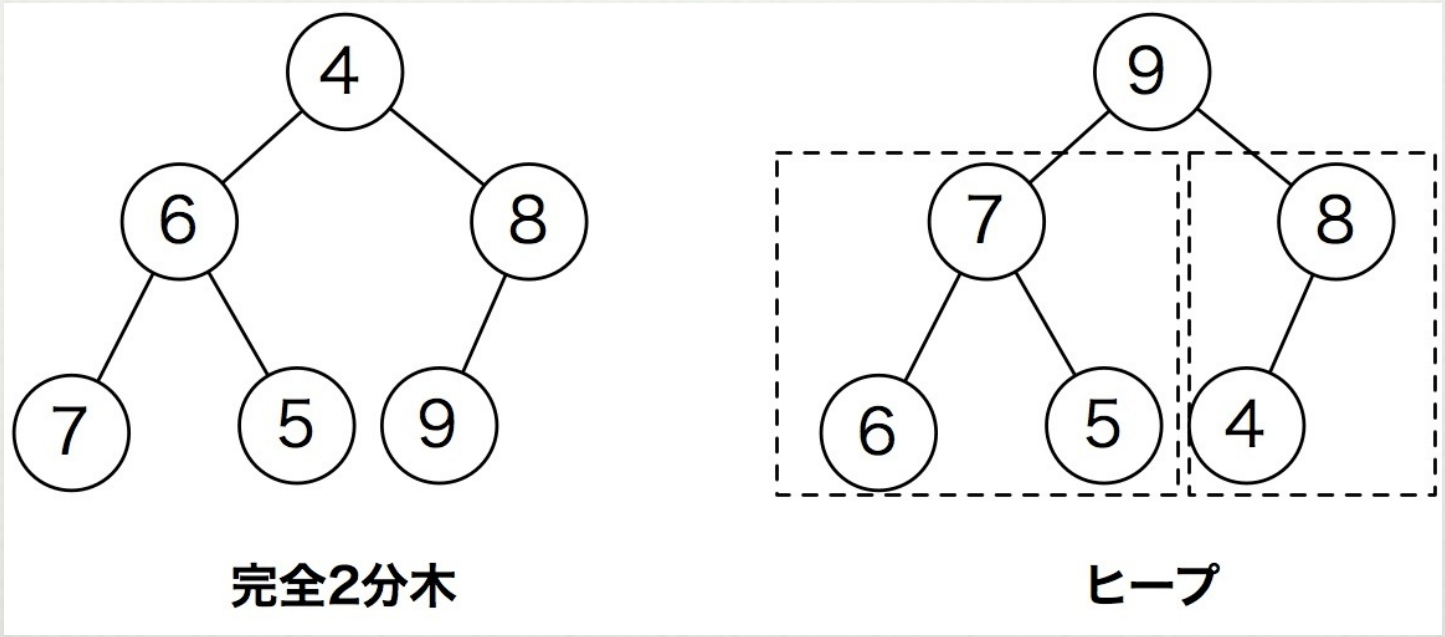
- **完全2分木**
  根から葉に至るすべてのノード（親節点）が２つの子を持つ2分木
  すべての葉は根から等しい距離（深さ）にある
  ※最後のノードは左の子のみでも可
  ※深さが１だけ異なる葉が存在し，最も深い葉は木全体の左側から詰められている



これらはすべて完全2分木

ヒープとは

- 親の値が子の値以上（あるいは以下）となっている**完全2分木**
  - 全ての親は必ず2つの子を持つ（最後の親は左の子のみでも可）
    - 部分木もヒープ
  - 兄弟の大小関係は任意



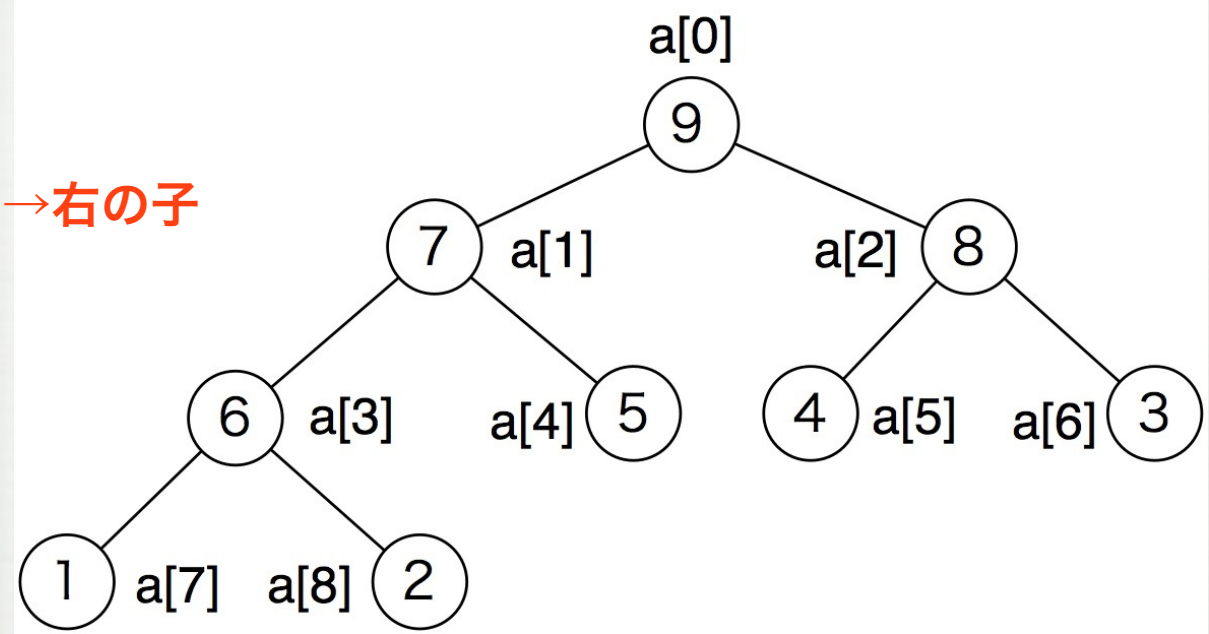
だけどヒープでない

Fig.1

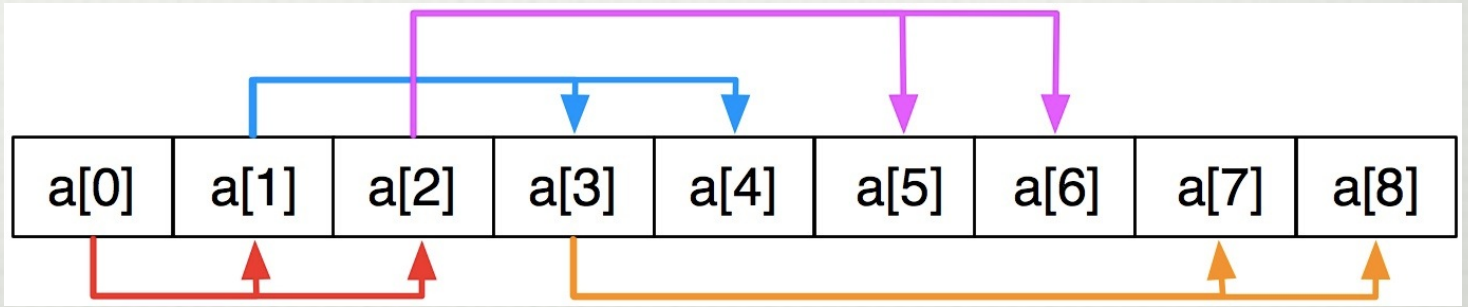
配列による表現

(a) ヒープ

親節点→左の子→右の子
の順につける



(b) 配列表現



ヒープ要素の参照

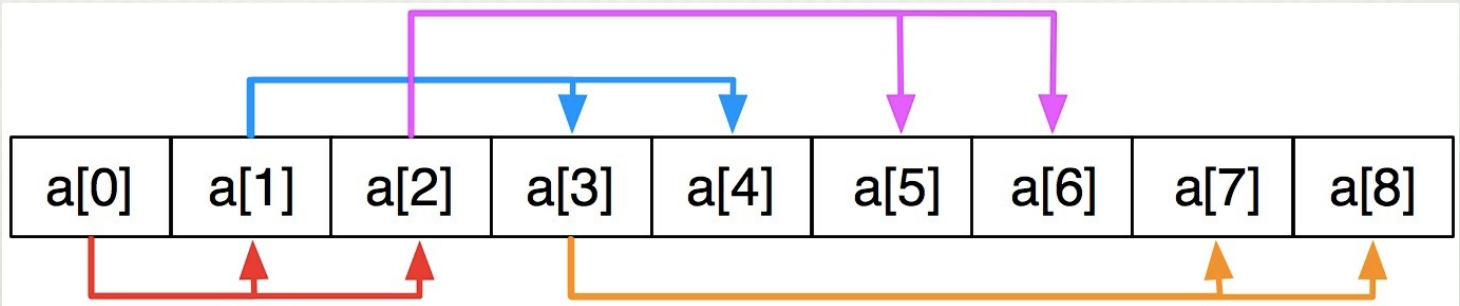
子の参照

a[i]の左の子：a[i*2+1] or a[(i+1)*2-1]

a[i]の右の子：a[i*2+2] or a[(i+1)*2]

親の参照

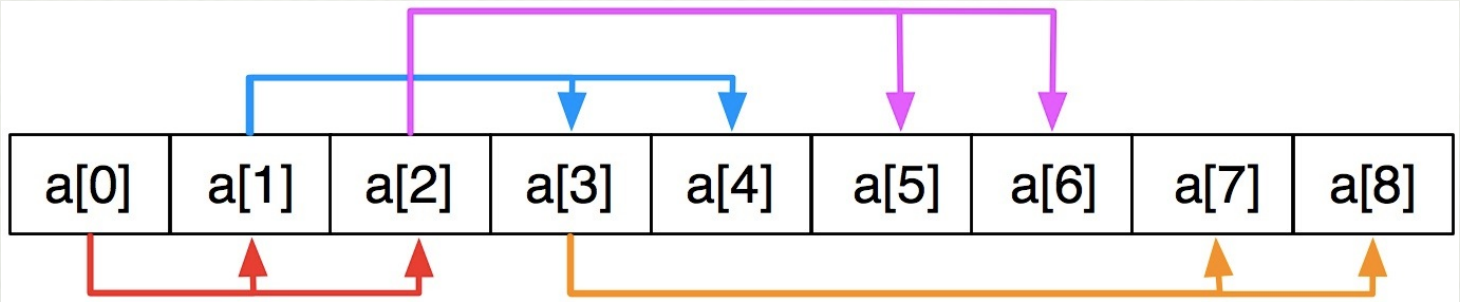
a[i]の親：a[(i-1)/2]



ヒープ要素の参照

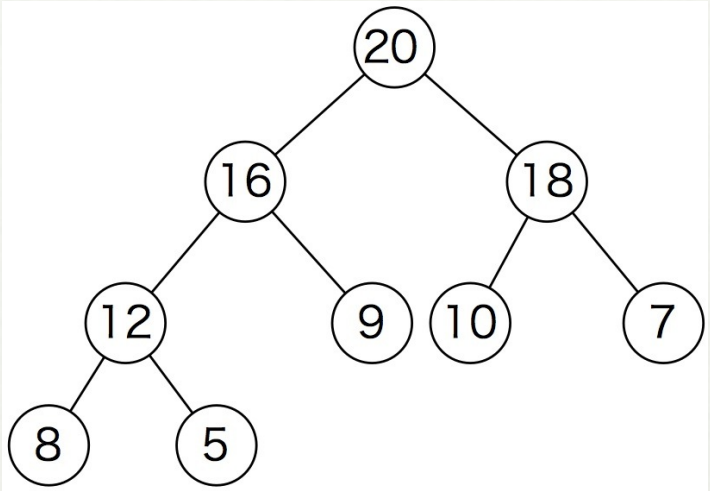
最後の親：a[(n-1)/2]

葉　　：a[(n-1)/2 +1]〜a[n]



練習問題11-1

1.以下のヒープについて答えなさい.



a. ヒープに含まれるノード（節点）数は：

b. 最後の親となる節点は：

c. 葉ノードの数は：

答え

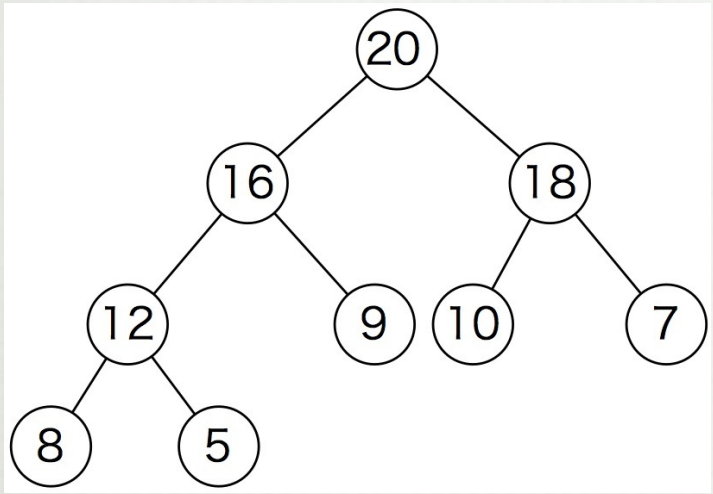
a. $2^3-1+2=9$個

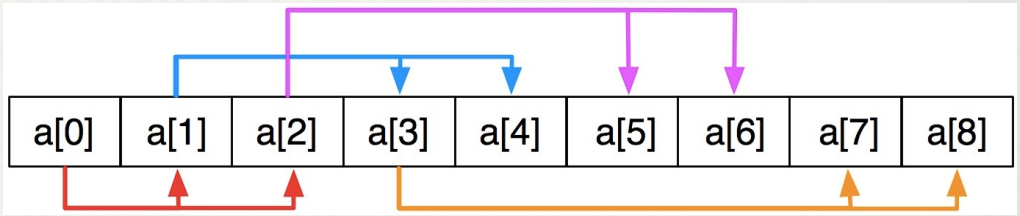
b. (8-1)/2=3→12

c. 4番目〜8番目(9,10,7,8,5): 5個

練習問題11-1

2.以下のヒープを配列を用いて表現してみよう





親→左の子→右の子
の順に入れていく

答え

20	16	18	12	9	10	7	8	5

練習問題11-1

3.ヒープを表す以下の配列について答えなさい.

heap[i](i=0～8)

9	8	7	5	6	4	3	2	1

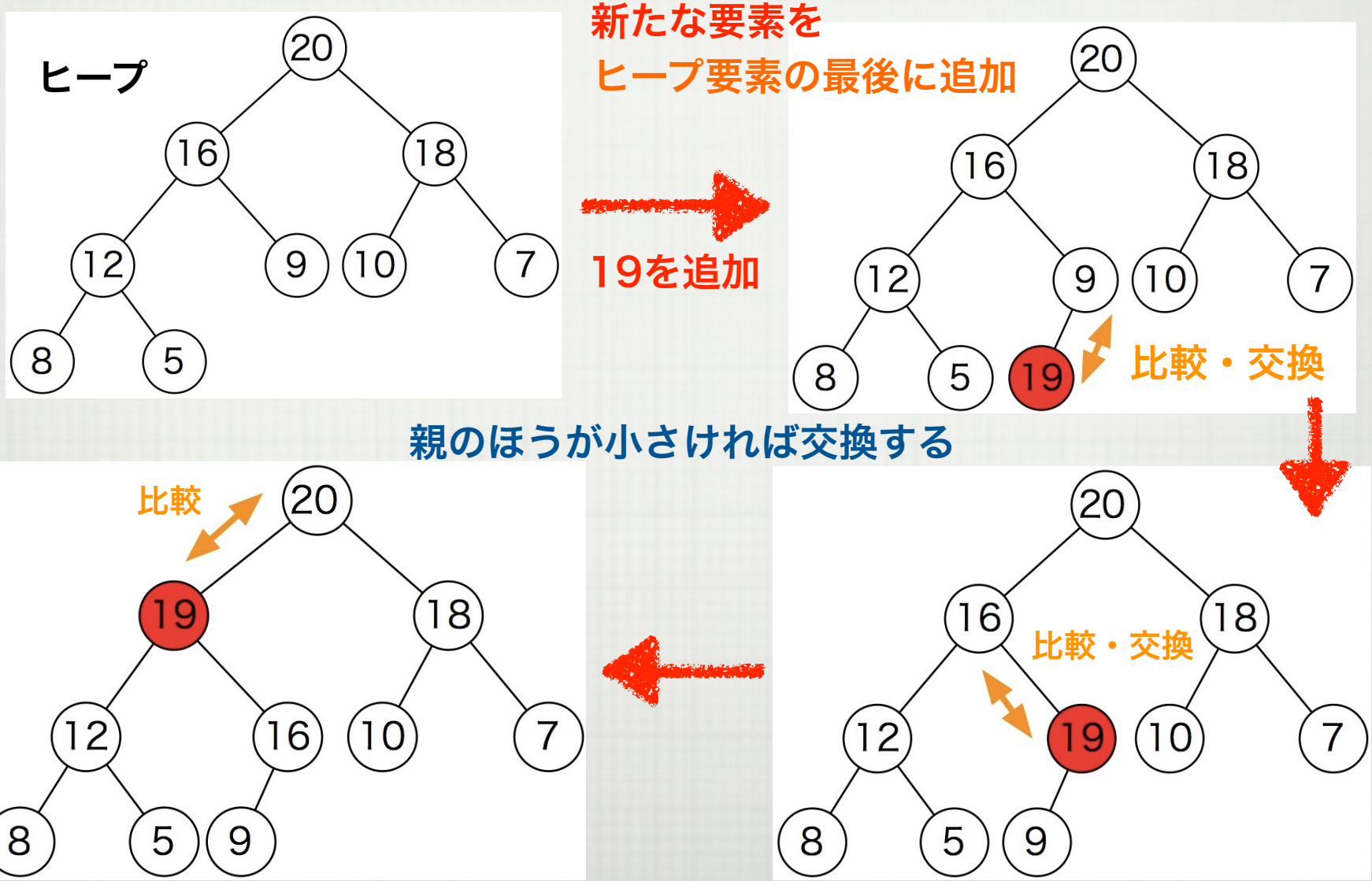
a. 要素heap[2](=7)の左の子は：
b. 要素heap[3](=5)の右の子は：
c. 要素heap[6](=3)の親は：
d. 最後の親は？

答え
heap[2*2+1=5]：4
heap[3*2+2=8]：1
heap[(6-1)/2=2]：7
heap[(8-1)/2=3]：5

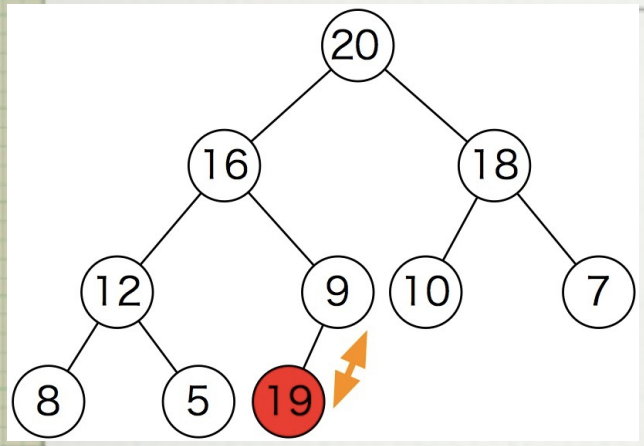
ヒープの処理

- □　ヒープへの要素追加（要素の上方移動）
- □　完全2分木からヒープの作成（親の下方移動）
- □　ヒープから根の削除・ヒープの再構成

ヒープ作成：新たな要素の追加
上方移動による方法



ヒープ作成：新たな要素の追加
上方移動による方法[配列]



ヒープ（初期状態）

a[0]								a[8]
20	16	18	12	9	10	7	8	5

2.親と比較 a[4]　　1.末尾に追加 a[9]

20	16	18	12	9	10	7	8	5	19

a[1]　4.親と比較　a[4]　3.値を交換　a[9]

20	16	18	12	19	10	7	8	5	9

5.値を交換

a[0]　a[1]　a[4]

20	19	18	12	16	10	7	8	5	9

6.根と比較

20	19	18	12	16	10	7	8	5	9

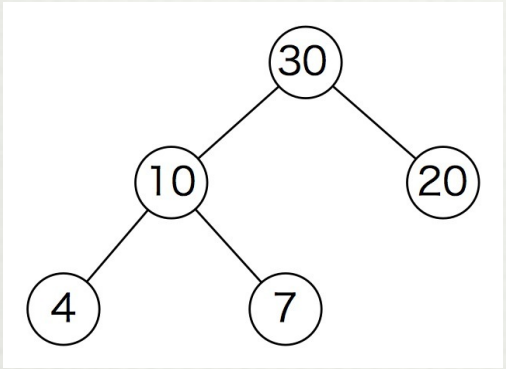
ヒープの作成：上方移動

Algo-Heap1

- 新しい要素をヒープに追加
  - 新しい要素を配列の最後に格納
  - その要素の親の値と比較
    - 親の値が小さければ交換
    - 親の値が大きいか等しい場合は終了
  - 交換した要素を子として，その親に対して同じことを繰り返す.
    （親がなくなるまで）

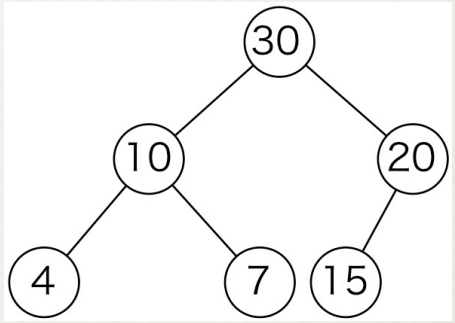
練習問題11-2

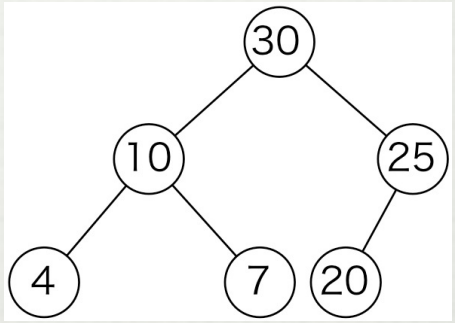
1.次のヒープに新しい要素xを，上方移動で追加する過程をシミュレーションしなさい.

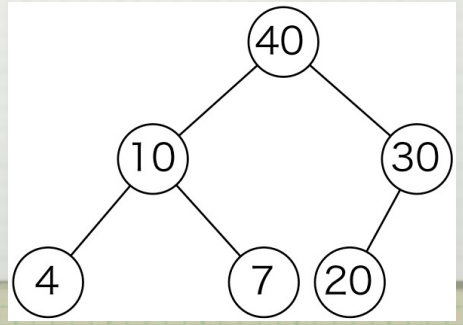


x=15の場合
x=25の場合
x=40の場合

答え

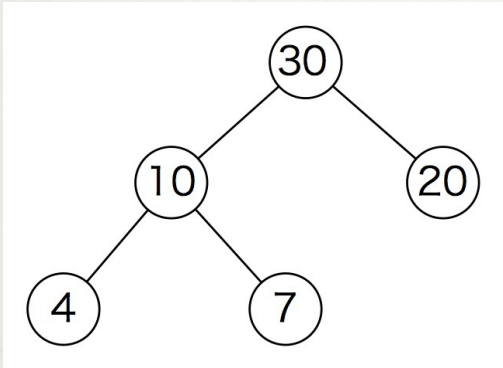




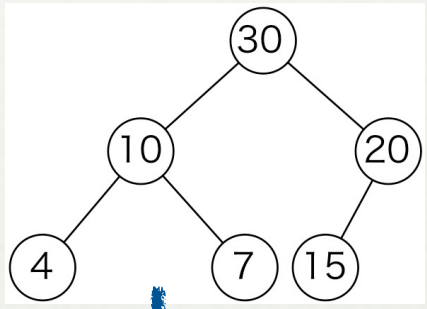


練習問題11-2

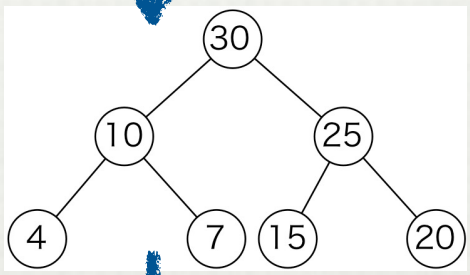
2. 次のヒープに新しい要素３つを，上方移動で順に追加する過程をシミュレーションしなさい.

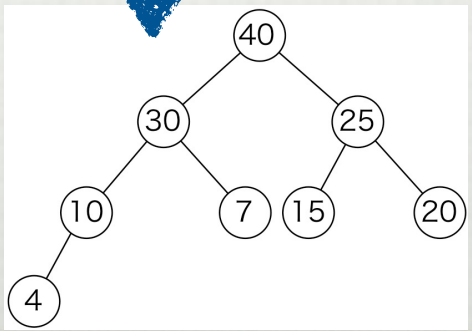


答え



x=15,25,40を順に追加





演習11-1

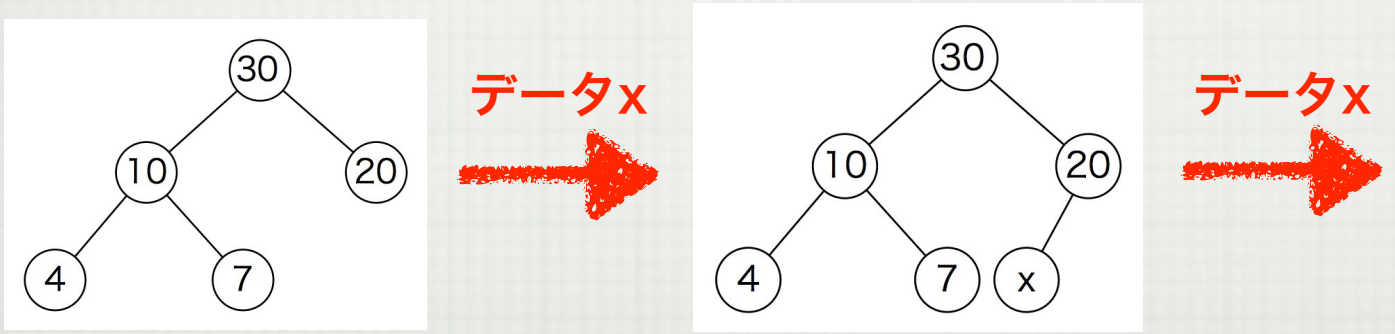
- 図に示すようなヒープに新しいデータxを追加するプログラム（sample11-1.c）を実行して，ヒープへの追加の様子を理解しよう.

x=15, 25, 40

```c
#include <stdio.h>
#define N 50
int main (void)
{
   int a[N];//the maxmum size of array is fixed at N
   int n;//heap size
   int i, parent, child, tmp;
   a[0]=30;//ヒープ
   a[1]=10;
   a[2]=20;
   a[3]=4;
   a[4]=7;
   n=5;//ヒープの最後
   printf("input data: ");//input data
   scanf("%d", &a[n]);//ヒープの最後に追加
   child=n;//追加ノードを子ノード
   parent=(child-1)/2;//その親ノード
   while (child>0 && a[parent]<a[child]) {//根まで比較
    tmp=a[parent];//交換
    a[parent]=a[child];
    a[child]=tmp;
    child=parent;//親ノードを子ノードに
    parent=(child-1)/2;//その親ノード
   }
   for (i=0;i<=n;i++)
    printf("a[%d]=%d  ",i,a[i]);
   printf("\n");
   printf("input data: ");//input data
   return 1;
}
```

演習11-2

- sample11-1.c を参考に，図のようなヒープに新しいデータxを順次追加するプログラム（sample11-2.c）を作成しよう．
  なお，入力が^d(EOF)の時に終了するものとする．



```c
#include <stdio.h>
#define N 50
int main (void)
{
   int a[N];//the maxmum size of array is fixed at N
   int n;//array size
   int i, parent, child, tmp;
   a[0]=30;//ヒープ
   a[1]=10;
   a[2]=20;
   a[3]=4;
   a[4]=7;
   n=5;
   printf("input data: ");//input data
   while (scanf("%d", &a[n])!=EOF) {//ヒープの最後に追加
    child=n;//追加ノードを子ノード
    parent=(child-1)/2;//その親ノード
    while (child>0 && a[parent]<a[child]) {//根まで比較
        tmp=a[parent];//交換
        a[parent]=a[child];
        a[child]=tmp;
        child=parent;//親ノードを子ノードに
        parent=(child-1)/2;//その親ノード
    }
    n++;//ヒープの最後のノード更新
    for (i=0;i<n;i++)
        printf("a[%d]=%d  ",i,a[i]);
    printf("\n");
    printf("input data: ");//input data
   }
   return 1;
}
```

課題11-1

- Algo-Heap1は空のヒープからもはじめることができる．そこで，sample11-2.cを参考に，データを一つずつ入力して入力ごとにヒープを構成していくプログラム（ex11-1.c）を書きなさい．ただし，入力するデータ数はたかだか50とする．

課題11-2

- ex11-1.cを参考に，親の値が子の値以下となるヒープを構成するようにプログラム(ex11-2.c)を修正しなさい.

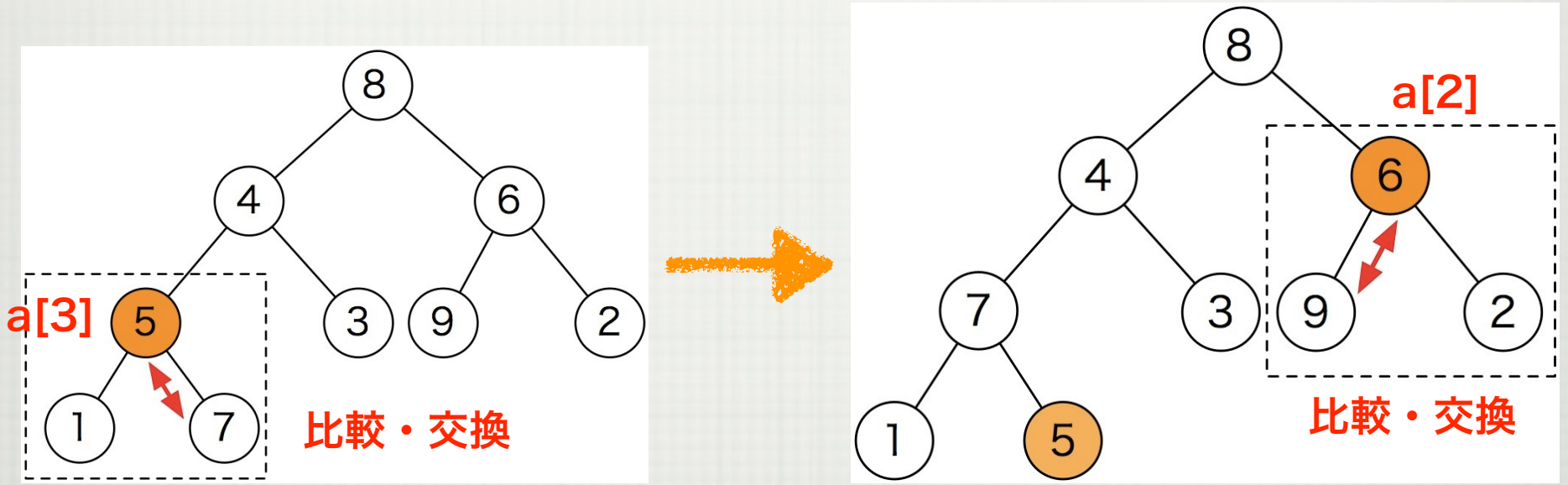
ヒープの処理

- □ ヒープへの要素追加（要素の上方移動）
- □ 完全2分木からヒープの作成（親の下方移動）
- □ ヒープから根の削除・ヒープの再構成

ヒープの作成：下方移動（1）

- あらかじめ与えられた完全2分木からのヒープ作成

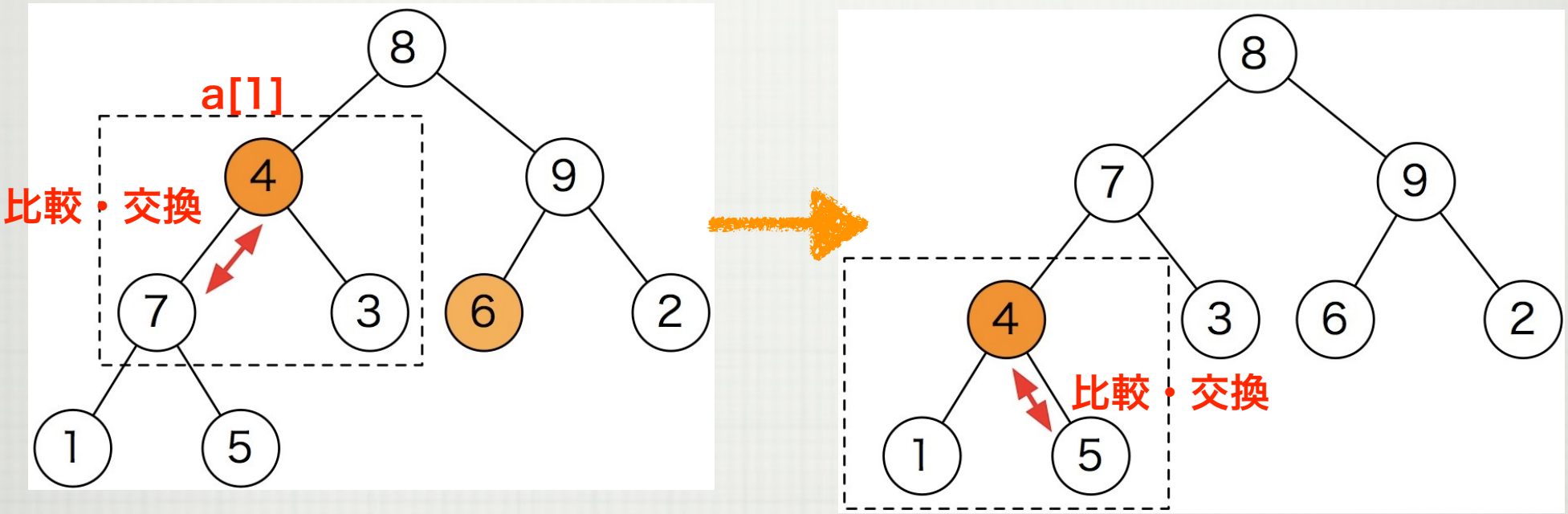
子を持つ最後の親から開始



子と親を「比較・交換」することにより，親を下方移動

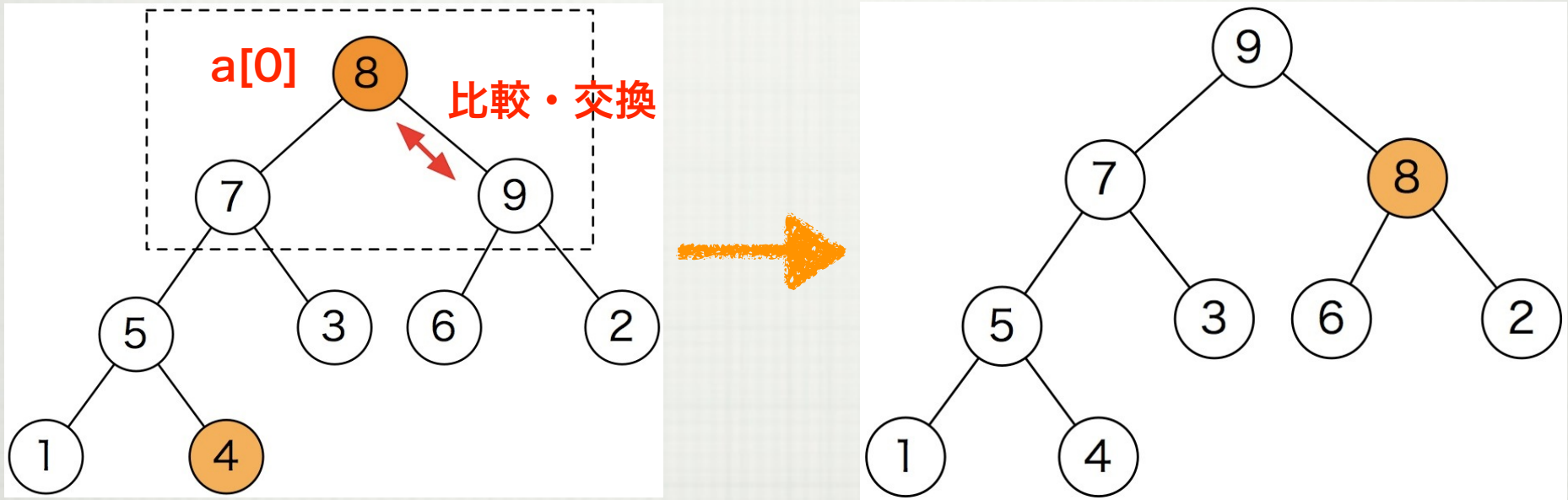
親の値が子の値（大きい方の子）より小さい場合，交換

ヒープの作成：下方移動（2）



部分木がある限り「比較・交換」を繰り返し，親を下方移動

ヒープの作成：下方移動（3）



根（ルート）まで繰り返す

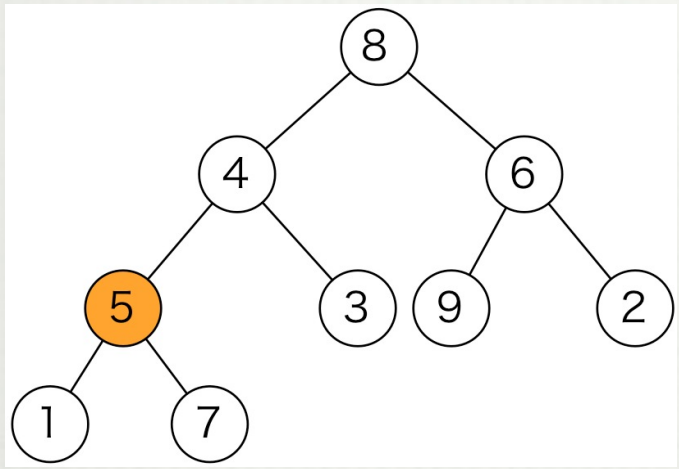
ヒープの作成：下方移動

Algo-Heap2

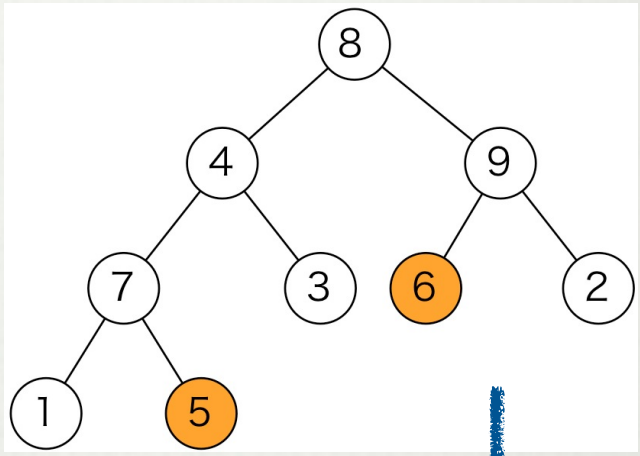
- 全データを2分木に割り当てる
- 子を持つ最後の親から始め，部分木ごとにルートまで以下を繰り返す
  - 親の値が子より小さければ，2つの子のうち大きい方と交換
  - 下方移動した要素を親として，下流の部分木に同じ処理を繰り返す.

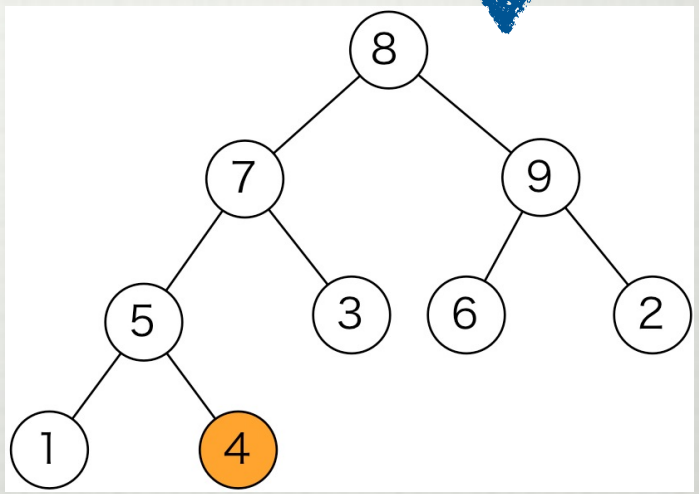
練習問題11-3

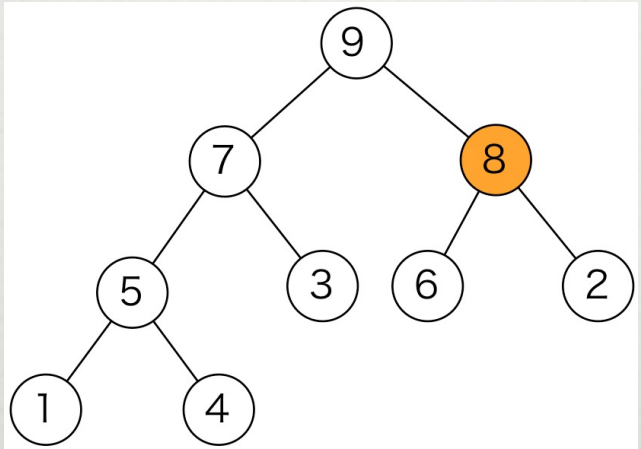
次の二分木から下方移動によってヒープを作る過程をシミュレーションしなさい.



答え

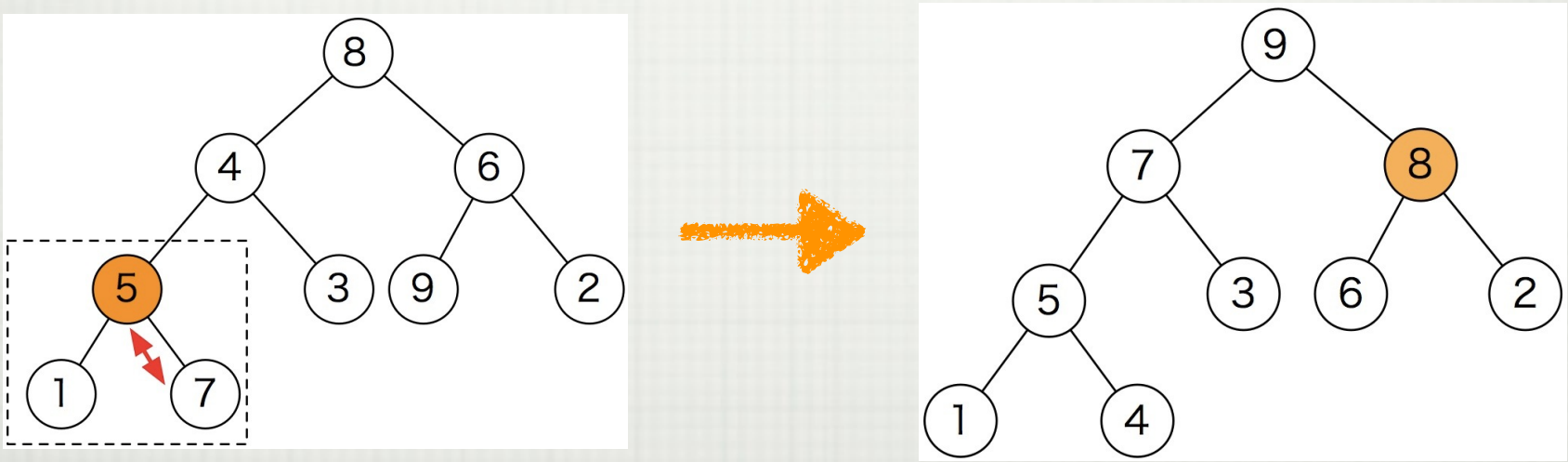






演習11-3

- 次の2分木から下方移動によってヒープを作成するプログラム（sample11-3.c）を作成して，ヒープが順次作られていく様子を理解しよう.



```c
#define N 50
int main (void)
{
   int a[N];//the maxmum size of array is fixed at N
   int n;//array size
   int i, j, parent, child, tmp;
   a[0]=8;//2分木
   a[1]=4;
   a[2]=6;
   a[3]=5;
   a[4]=3;
   a[5]=9;
   a[6]=2;
   a[7]=1;
   a[8]=7;
   n=8;
   for (j=0;j<=n;j++) printf("a[%d]=%d  ",j,a[j]);
   printf("\n");
   for (i=(n-1)/2;i>=0;i--){//最後の部分木から根を親とする部分木まで
    parent=i;
    child=parent*2+1;//左の子ノード
    while (child<=n) {//下方移動
        if (child<n && a[child]<a[child+1])//値が大きい子ノードを決める
            child++;
        if (a[parent]<a[child]) {//親ノードが小さい場合交換
            tmp=a[parent]; a[parent]=a[child];a[child]=tmp;
        }
        parent=child;//次の親ノード設定
        child=parent*2+1;//その左の子ノード設定 }
    for (j=0;j<=n;j++)
        printf("a[%d]=%d  ",j,a[j]);
    printf("\n");
   }
   return 1;
```

演習11-4

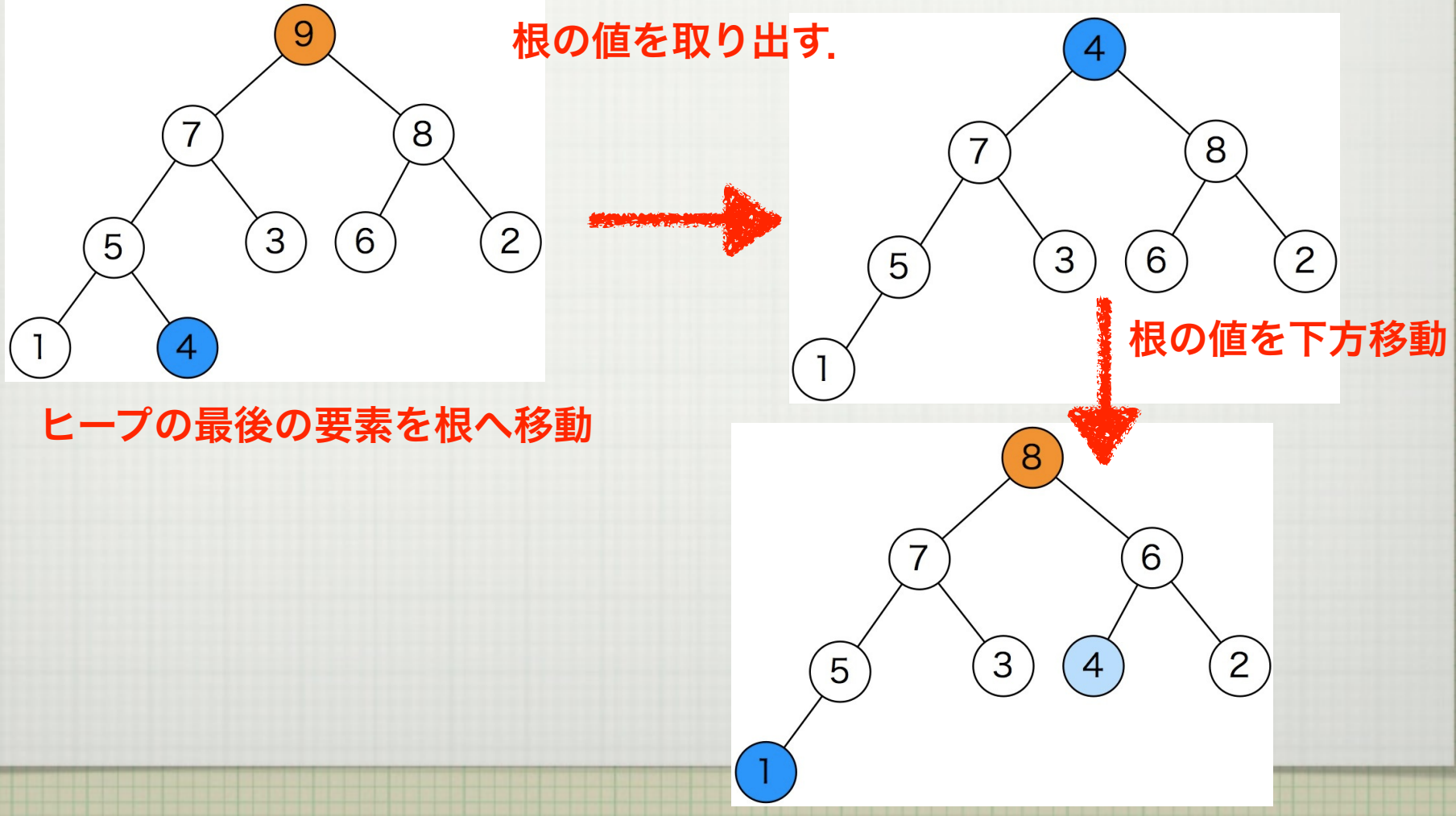
- sample11-3.cを参考に，親の値が子の値以下となるヒープを構成するプログラムを作りなさい(sample11-4.c).

ヒープの処理

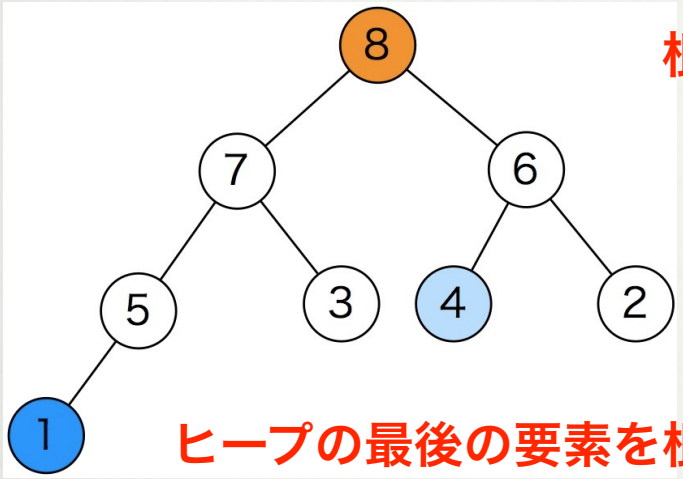
- ヒープへの要素追加（要素の上方移動）
- 完全2分木からヒープの作成（親の下方移動）
- **ヒープから根の削除・ヒープの再構成**

ヒープから根の削除

- 根（最大要素）を削除して，ヒープを再構成
- これを繰り返せば，要素を整列（ソート）できる

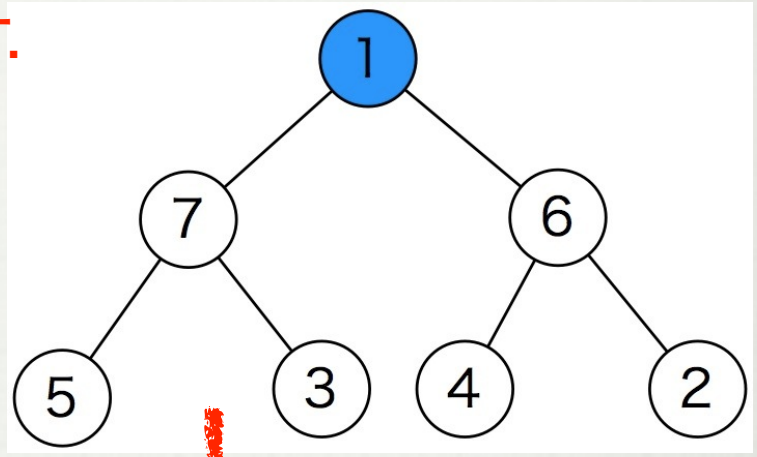


ヒープから根の削除（２）

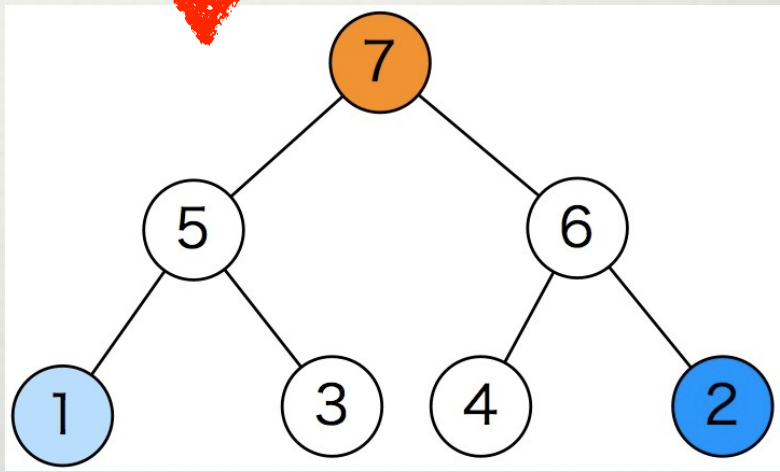


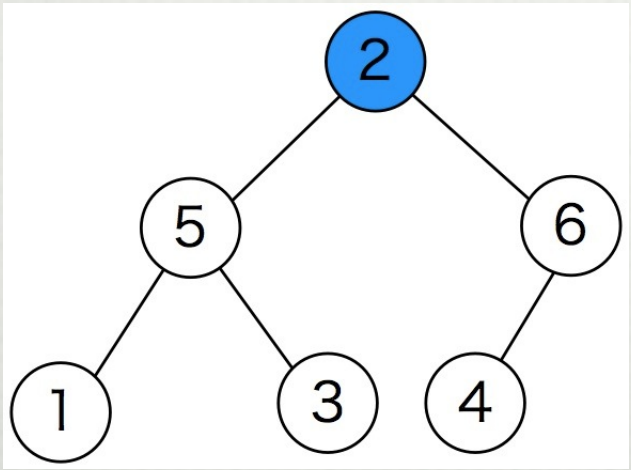
根の値を取り出す.

ヒープの最後の要素を根へ移動

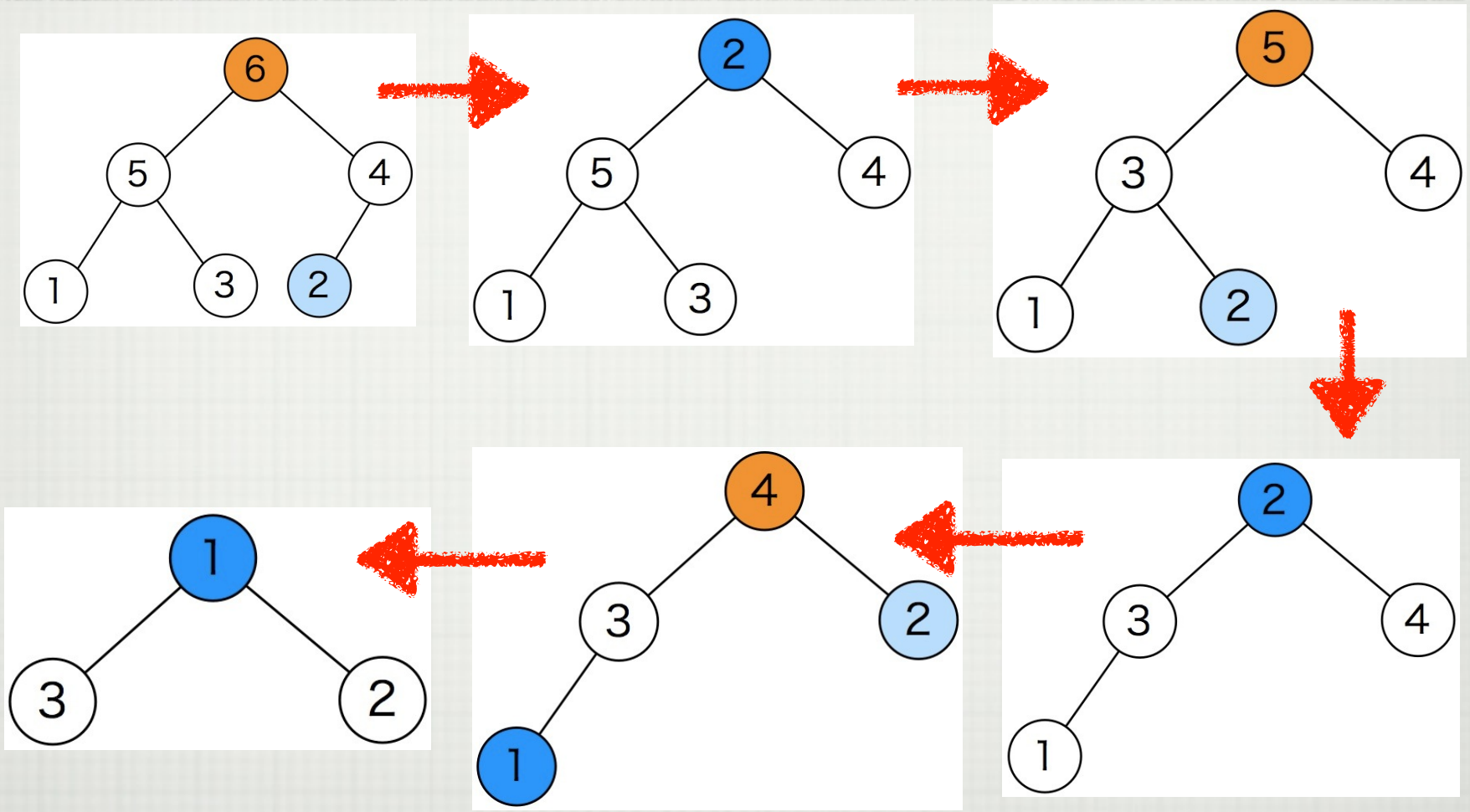


根の値を下方移動

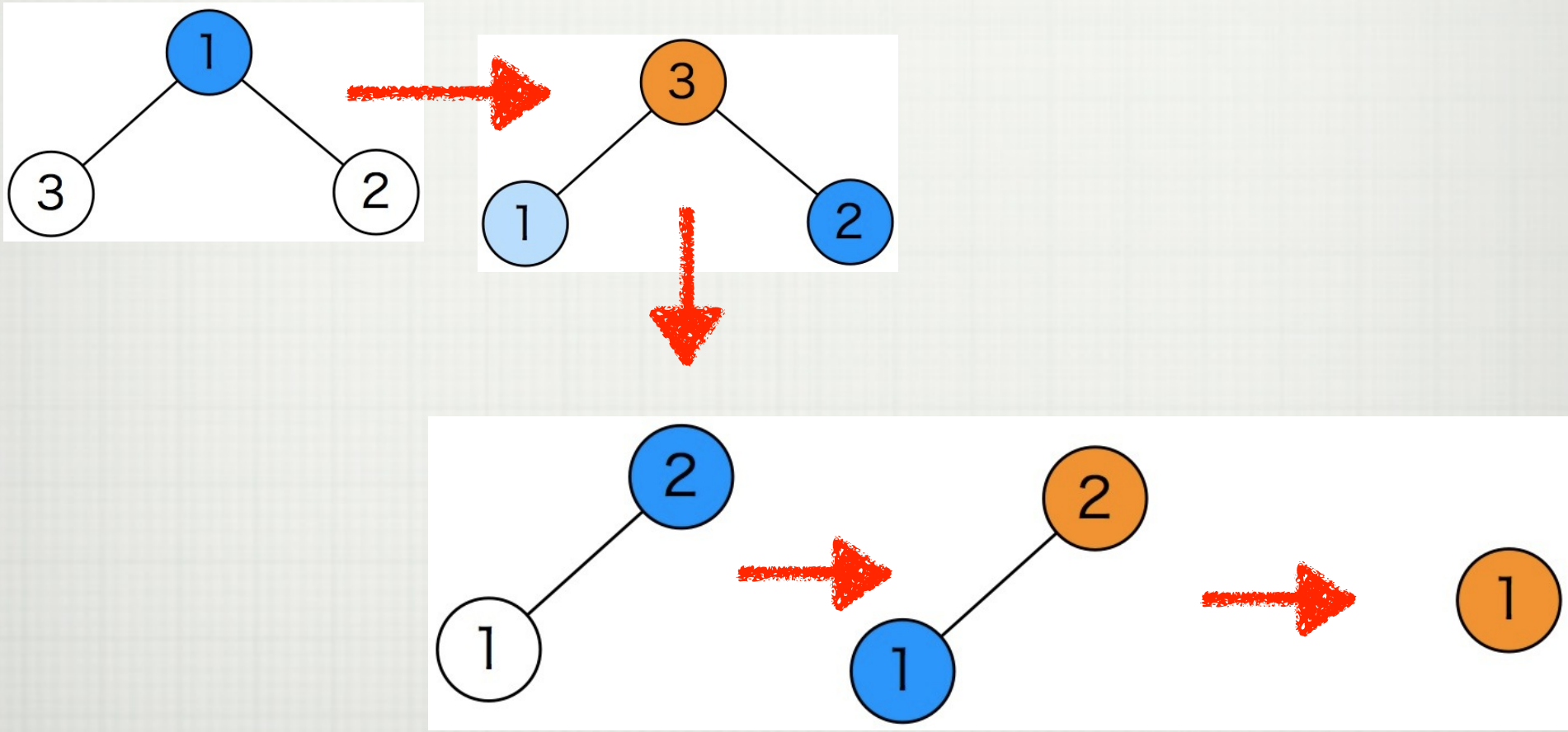




ヒープから根の削除（３）



ヒープから根の削除（4）



ヒープから根の削除

Algo-Heap3

- 最後の要素を根に移動
  - 根の左部分木，右部分木はヒープとなっている
- 根の値を下方移動すれば良い
  - 根の値が，子の値より小さい場合，大きい方の子の値と交換
  - 子の値のほうが小さくなるか，葉に到達するまで繰り返す

演習11-5

- sample11-3.cと同様にあらかじめ与えた2分木から下方移動によってヒープを構成し，出来上がったヒープから順に根を除きヒープを再構成するプログラム（sample11-5.c）を作成しなさい.
なお，ヒープが根の値だけとなるまで繰り返すこと.

ヒント：下方移動する関数shiftdownを作る

```
shiftdown (int parent, int n, int heap[])
//parent:下方移動させる親ノードのインデックス
//n: 下方移動させるヒープの最後の要素
//heap[]: ヒープを表す配列
```

```
void shiftdown (int parent, int n, int heap[])
{
  int child;
  int tmp;

  child=parent*2+1;
  while (child<=n) {
   if (child<n && heap[child]<heap[child+1])
      child++;
   if (heap[parent]<heap[child]) {
      tmp=heap[parent]; heap[parent]=heap[child];heap[child]=tmp;
   }
   parent=child;
   child=parent*2+1;
  }
}
```

課題11-3

- データを一つずつ入力して，上方移動によって入力ごとにヒープを構成する．そして，次に出来上がったヒープから順に根を取り除きヒープを再構成するプログラムを書きなさい．なお，ヒープが根の値だけとなるまで繰り返すこと． ただし，入力するデータ数はたかだか50とする．

レポート提出

- □　課題11-1～11-3
- □　提出期限：7月8日　0:00（7月7日まで）
- □　提出先：Webclass