半順序完全２分木からの最小要素の削除

ヒープの先頭が，木の根であり最小要素.

(1) ヒープの末尾の要素を，ヒープの先頭に移し，
移動要素ｒとする（この時点では半順序は不成立）.

(2) 節点数 ｎ を１減らし，ヒープ末尾を１つ繰上げる.

(3) 繰り返す（より下に対し局所的な半順序の保証）.

(3-1) 移動要素ｒが葉にあれば(*)，(3) の終り.

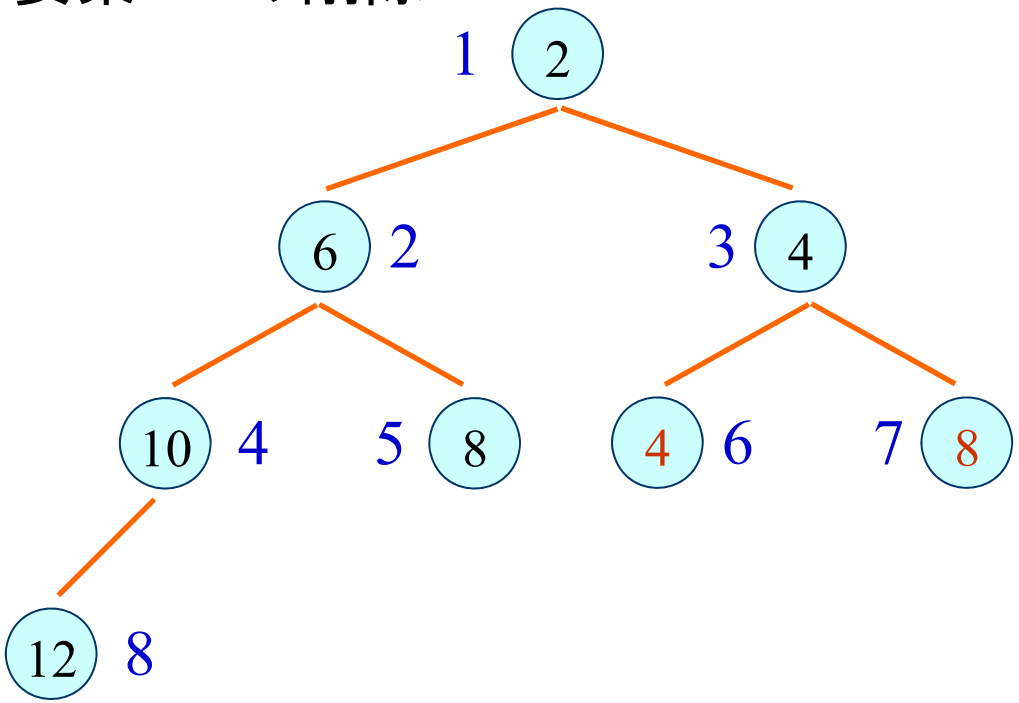
(3-2) 移動要素ｒが存在する節点が持つ，２つの子の要素のうち，値が小さい要素をｓとする.
子が１つの場合は，その要素をｓとする.

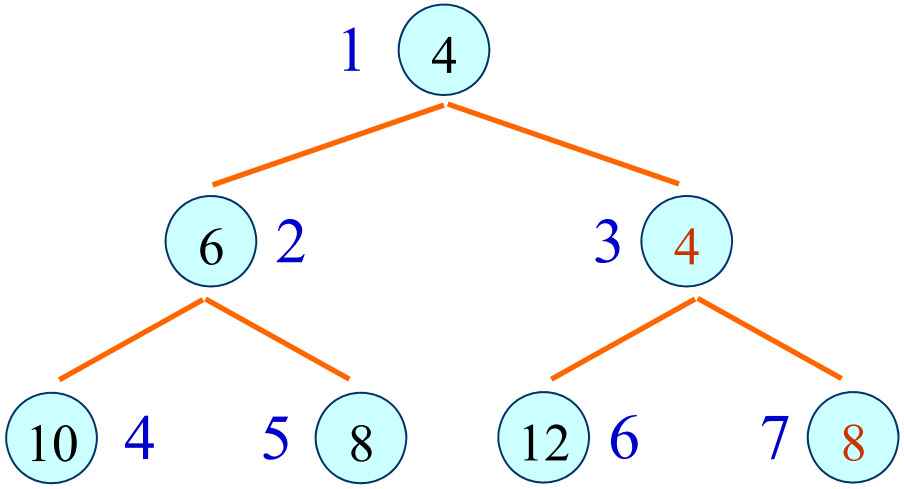
(3-3) 移動要素ｒとｓで値を比較し，移動要素の値が大きければ，
移動要素ｒとｓを入れ替える.
さもなければ，(3) の終り.

(*) 葉であることは，（n/2（切捨て）＋１）から ｎ で判断

.

最小要素 ２ の削除





先頭 ⇨

末尾 ⇨

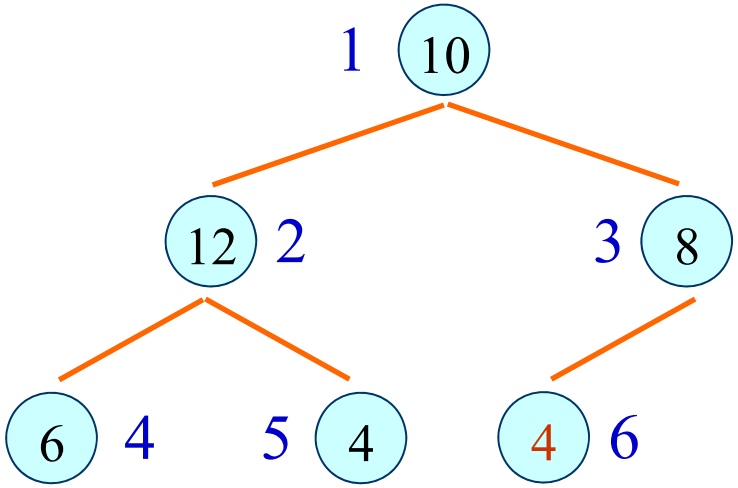
1	4
2	6
3	4
4	10
5	8
6	12
7	8
8	12
9	
10	

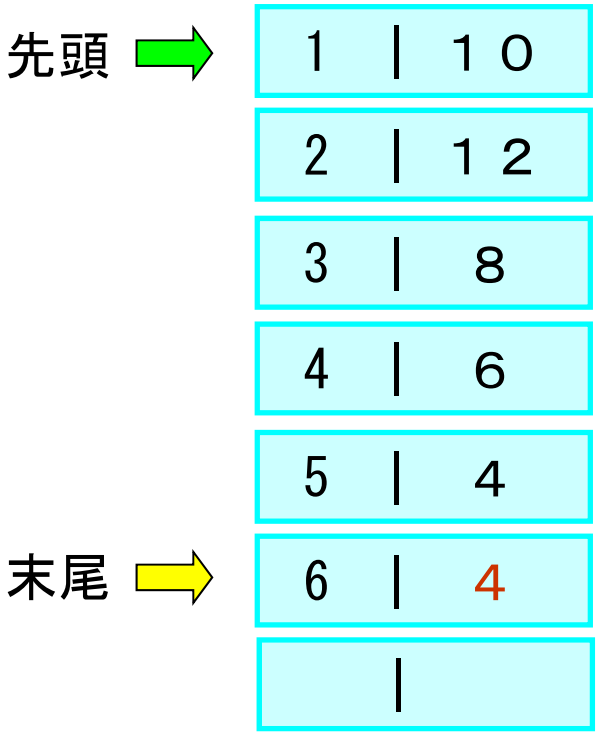
半順序完全２分木の性質は，失われない．

完全２分木の，半順序木への変換

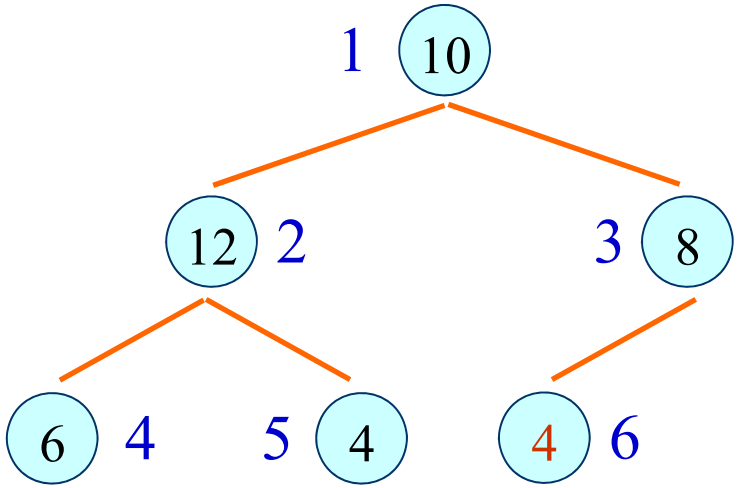
節点数を n とする.

(1) i を，n／2 （切捨て）から 1 まで，
繰り返す.
(1-1) node[i] を移動要素 r とし，
最小要素の削除の (3) を
行う.

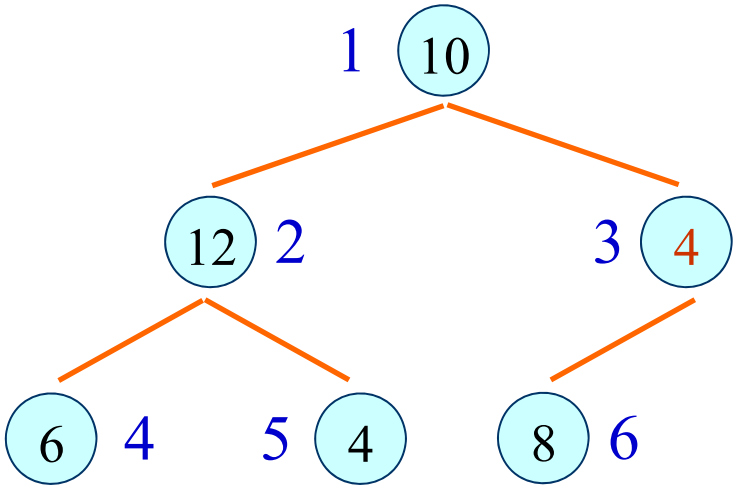




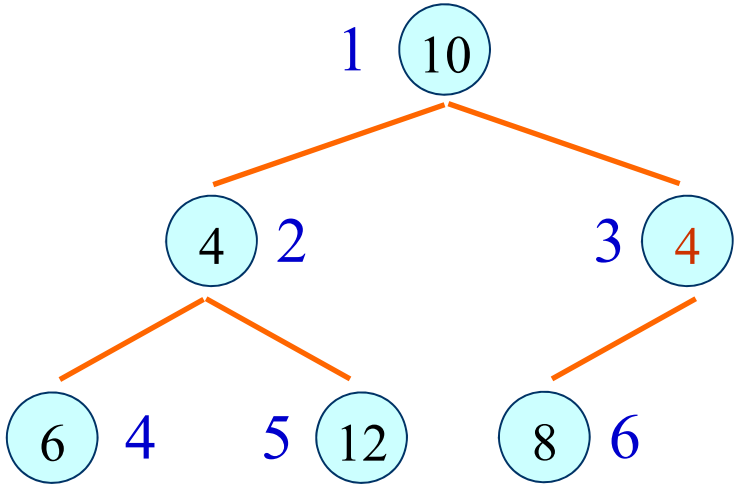
初期状態



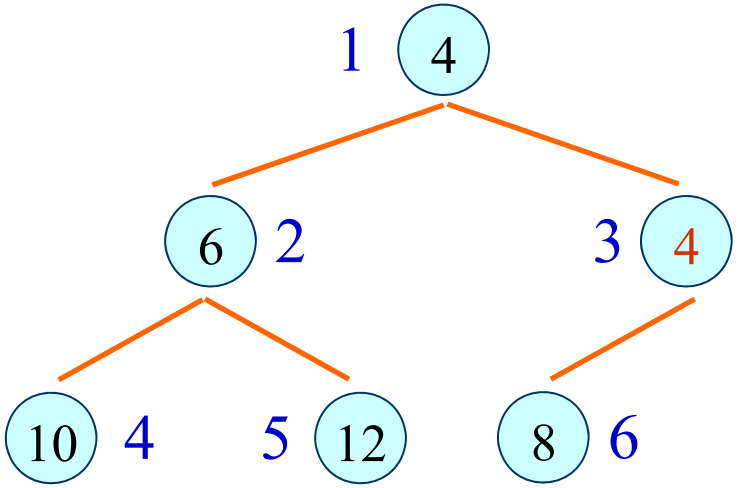
１回目終了時　ｉ＝３



２回目終了時　ｉ＝２



３回目終了時　ｉ＝１



半順序完全２分木からの特定要素 node[i] の削除

節点数を ｎ とする.

(1) ヒープの末尾の要素 node[n] を，節点 ｉ の位置に置く.
(2) 節点数 ｎ を１減らし，ヒープ末尾を１つ繰上げる.
(3) 現在，節点 ｉにある要素（末尾にあった要素）を追加要素ｐとして，要素の追加の（3）を行う（より上に対し局所的な半順序の保証）.
(4) 現在，節点 ｉ にある要素を移動要素ｒとして，最小要素の削除の（3）を行う（より下に対し局所的な半順序の保証）.

処理の手間

追加，削除とも，処理の手間は，木の深さに比例する．

完全半順序２分木の要素追加，（最小も含めた）要素削除の計算量は，
Ｏ（ｌｏｇ_2 （ｎ））

完全２分木の半順序木への変換は，
Ｏ（ｎ ｌｏｇ_2 （ｎ））

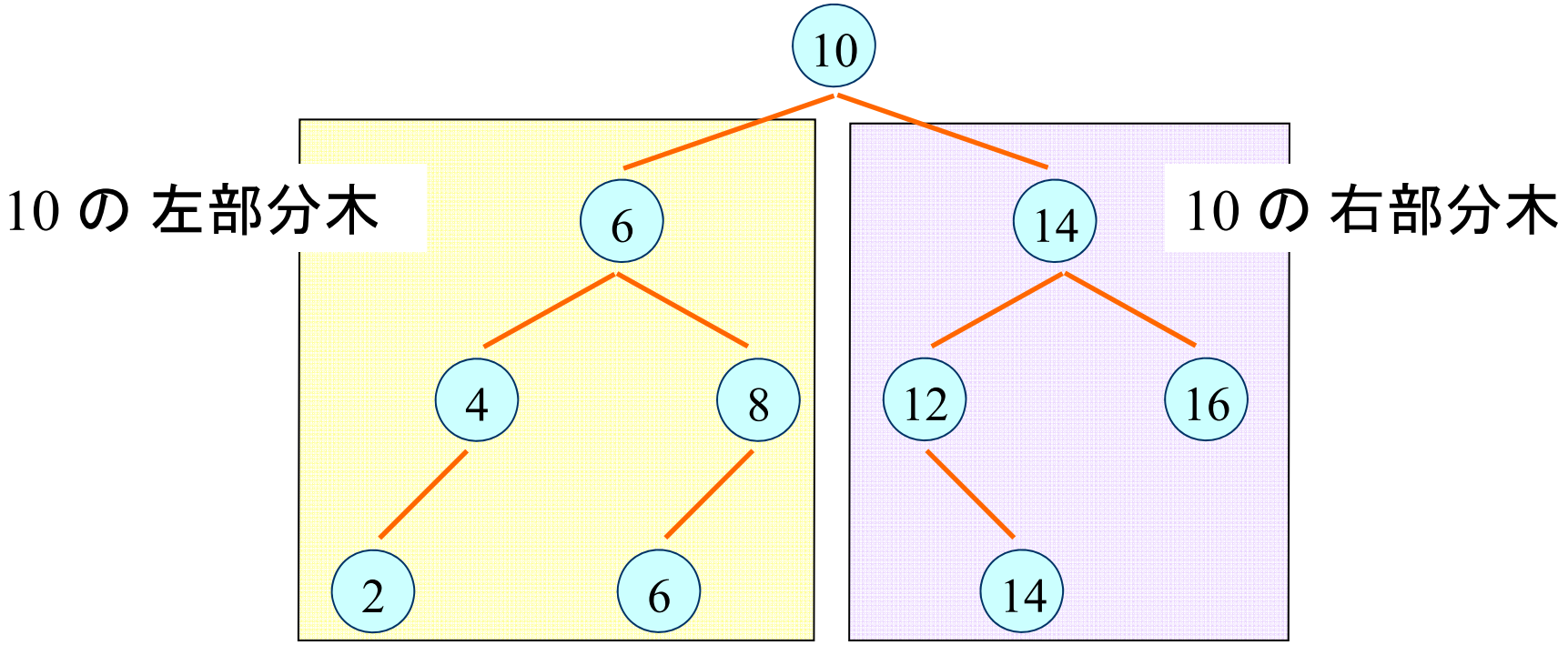
その他の木

２分探索木

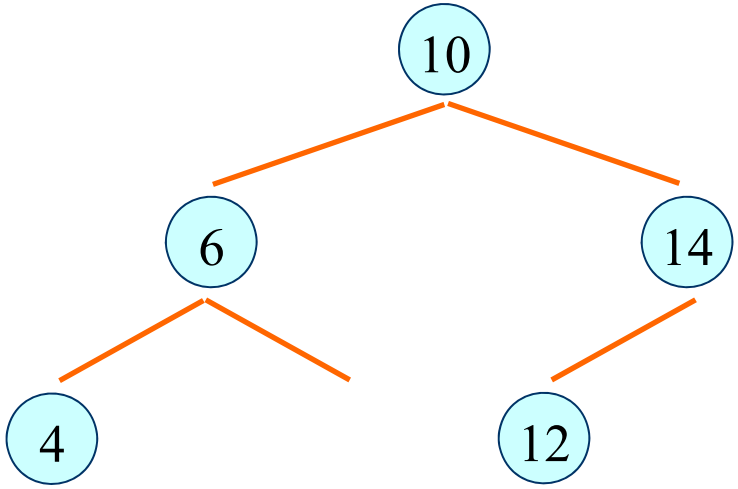
２分木の節点ｉにある要素の値ｋｉに対し，

左部分木にある要素の値はｋｉより小さい．

右部分木にある要素の値はｋｉ以上である．



効率的な２分探索木
高さＯ（ｌｏｇ_2 ｎ）



非効率的な２分探索木
高さＯ（ｎ）