アルゴリズムとデータ構造

コンピュータサイエンスコース
知能コンピューティングコース

第4回

スタック，キュー，ヒープ
ヒープソート

塩浦昭義
情報科学研究科 准教授

shioura@dais.is.tohoku.ac.jp
http://www.dais.is.tohoku.ac.jp/~shioura/teaching

データ構造とは？

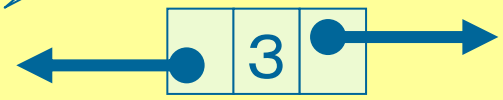
- アルゴリズムの中で，与えられた問題に関連するデータ集合を管理するための道具
- 良いデータ構造とは？
  - データ管理に必要な時間が短い
  - シンプル
  - 必要な領域計算量（記憶容量，領域量）が小さい

集合を管理する：双方向リストの利用

- 双方向リストを利用して集合を表現する
  - 必要な領域計算量は集合のサイズに等しい
  - 整数の追加，削除はO(1)時間で可能

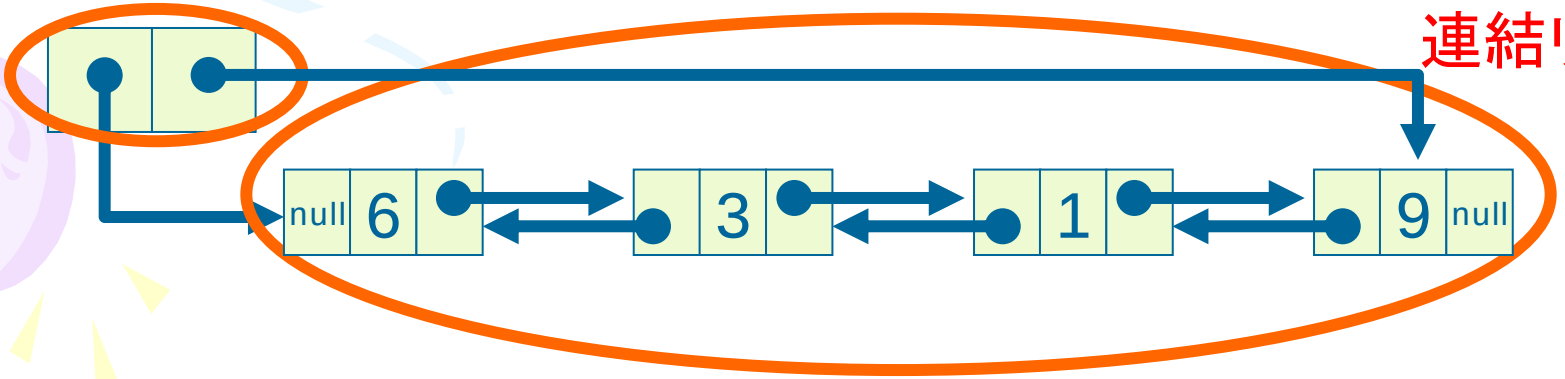
C言語では，構造体により実現可能

セルの構成：
要素（整数）
前のセルへのポインタ
次のセルへのポインタ



連結リストの最初と最後のセルへのポインタ

連結リストの本体

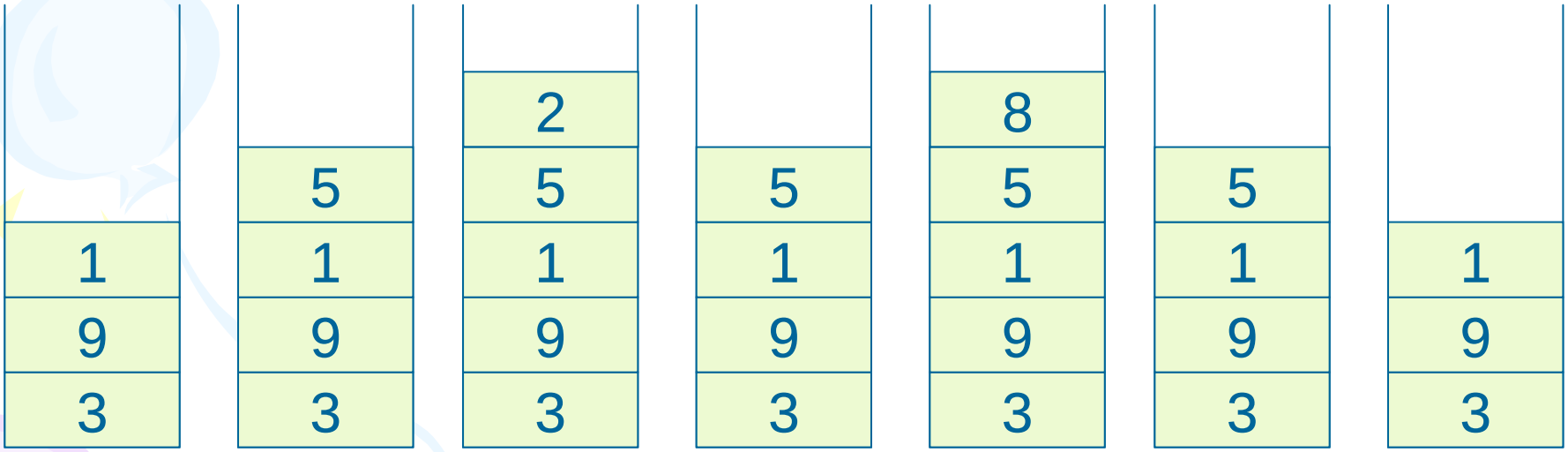


集合に対する特殊な演算とデータ構造

- 最も新しい要素を常に削除する
  - ➔ スタック
- 最も古い要素を常に削除する
  - ➔ 待ち行列（キュー）
- 最も小さい（大きい）要素を常に削除する
  - ➔ 優先度付き待ち行列，ヒープ

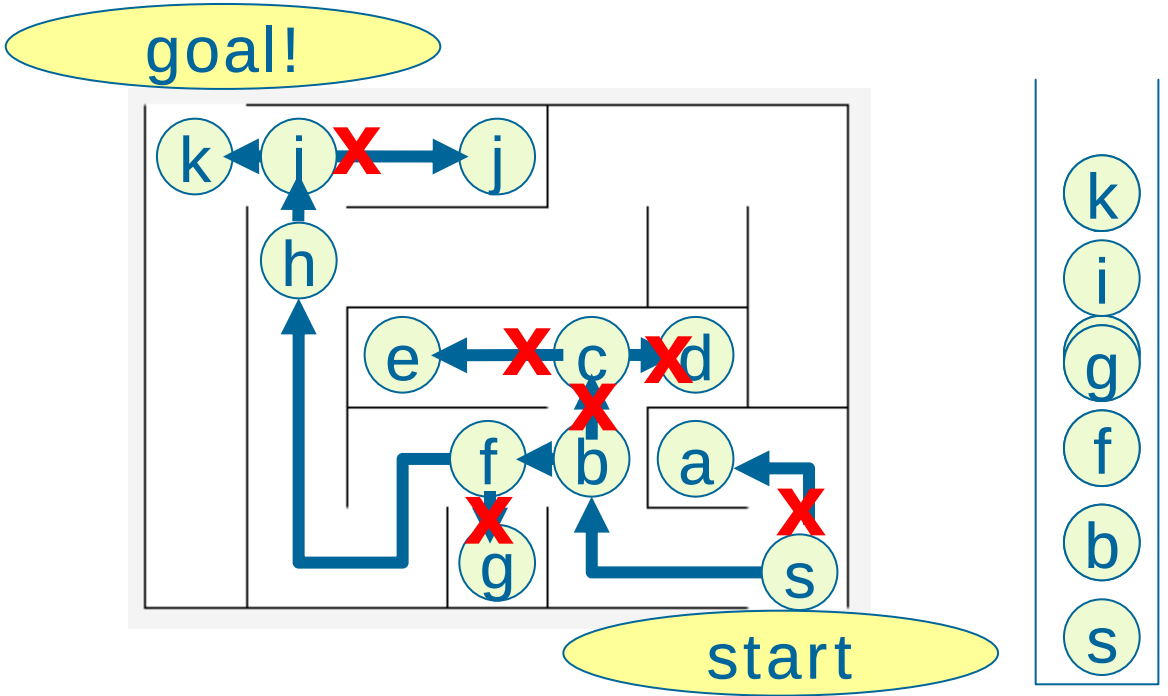
スタック

- 最も新しい要素を常に削除するときに使うデータ構造
- 次のような図で表現されることが多い



スタックとバックトラック

- スタックはバックトラック探索を行なうときによく使われる
- 例：迷路の探索：進んだ道を戻ったりしながらゴールを目指す
  - 可能性がある限り先に進む
    ➔ゴールへ到達する可能性がなくなったら戻る
  - これまで進んできた道を覚えるために，スタックを使う



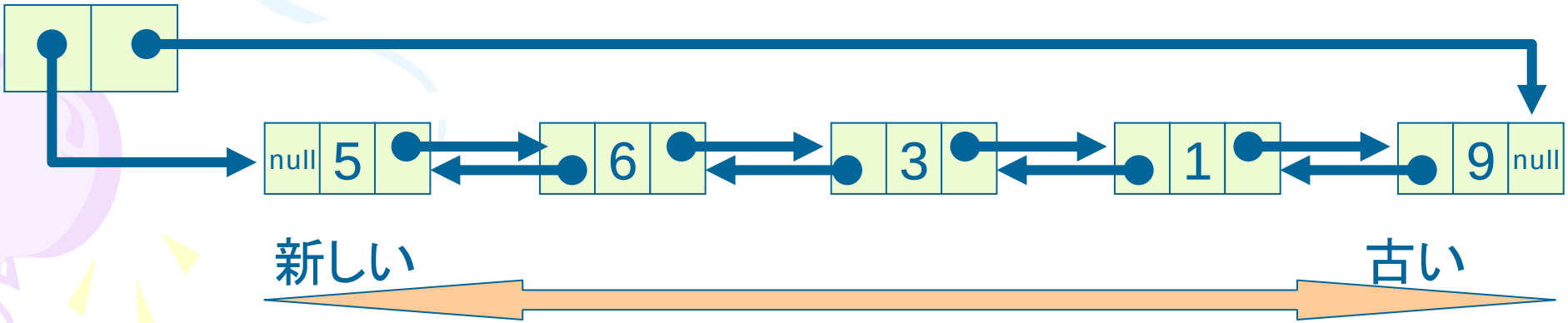
スタックの実現方法

- 双方向リストを使えばよい
- 要素を追加するとき，リストの先頭に追加

  ➔ リストの要素は，新しいものから古いものへと並ぶ
- 最も新しい要素を削除したい

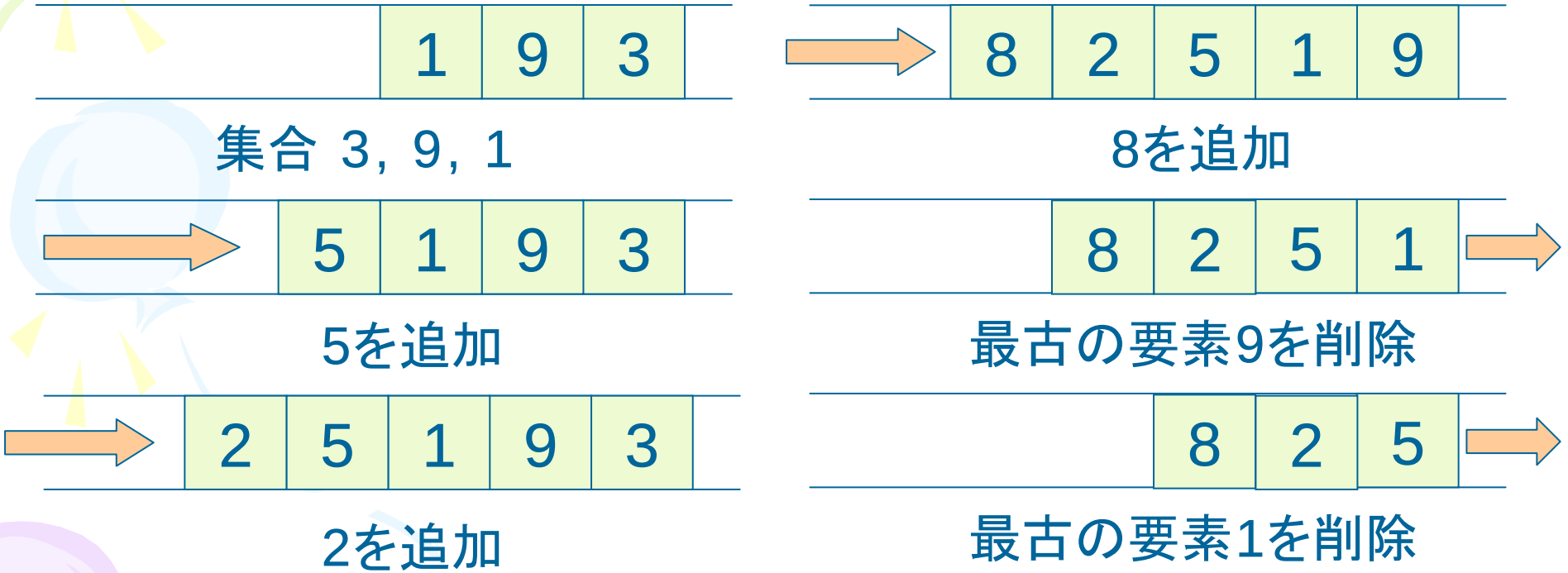
  ➔リストの先頭の要素を削除すればよい

∴ 追加も削除もO(1)時間でできる



待ち行列(キュー)

- 最も古い要素を常に削除するときに使うデータ構造



ジョブの処理の際に利用

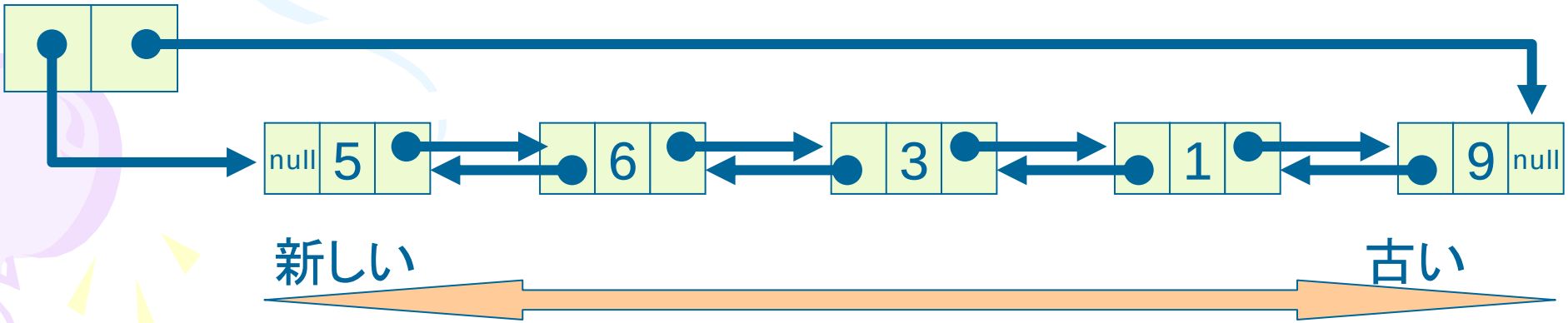
待ち行列の実現方法

- 双方向リストを使えばよい
- 要素を追加するとき，リストの先頭に追加

  ➔ リストの要素は，新しいものから古いものへと並ぶ
- 最も古い要素を削除したい

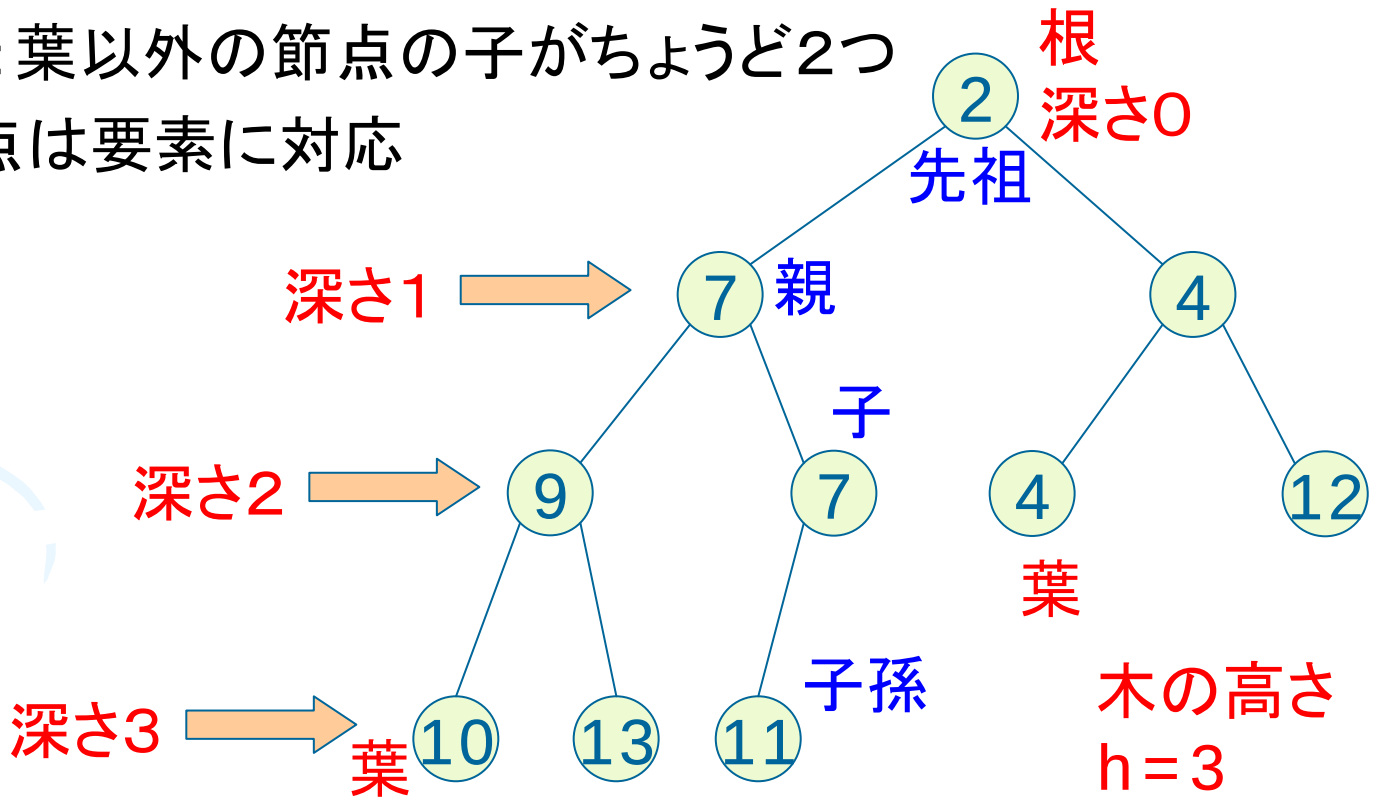
  ➔リストの最後尾の要素を削除すればよい

∴ 追加も削除もO(1)時間でできる



ヒープ

- 最も小さい（大きい）要素を常に削除するときに使うデータ構造
  - 新しい要素の追加，最小要素の削除は O(log n) 時間
- 2分木により表現される
  - 2分木：葉以外の各節点が高々2つの子をもつ根付き木
  - 完全2分木：葉以外の節点の子がちょうど2つ
- 2分木の各節点は要素に対応



ヒープの条件

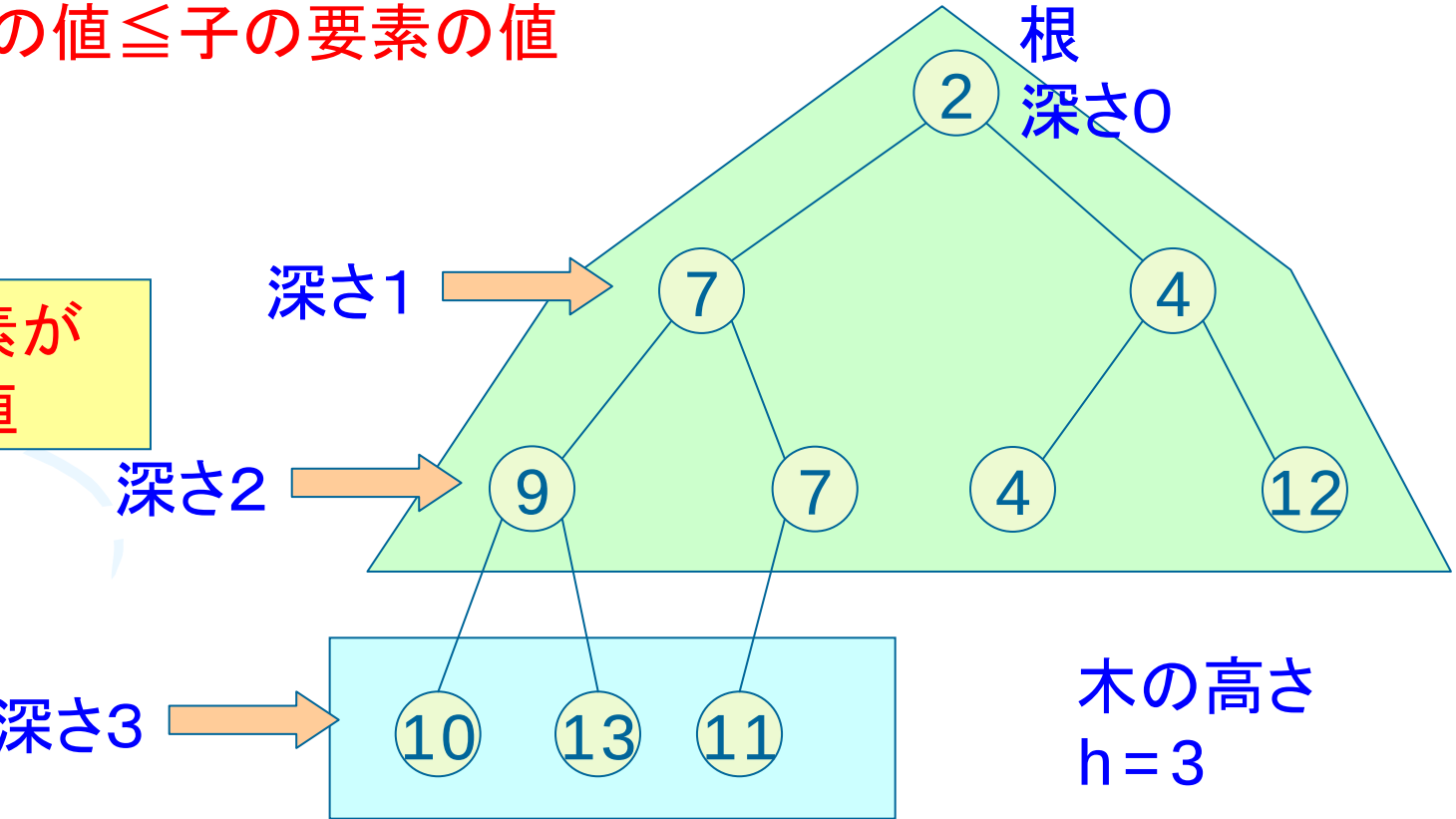
木の高さ
$= \lfloor \log_2 n \rfloor$

- 次の条件を満たす2分木をヒープと呼ぶ

(1)木の高さが h ➔ 深さ h-1 までは完全2分木
深さ h では，左側に葉が詰められている

(2)親の要素の値≦子の要素の値

根の要素が
最小値



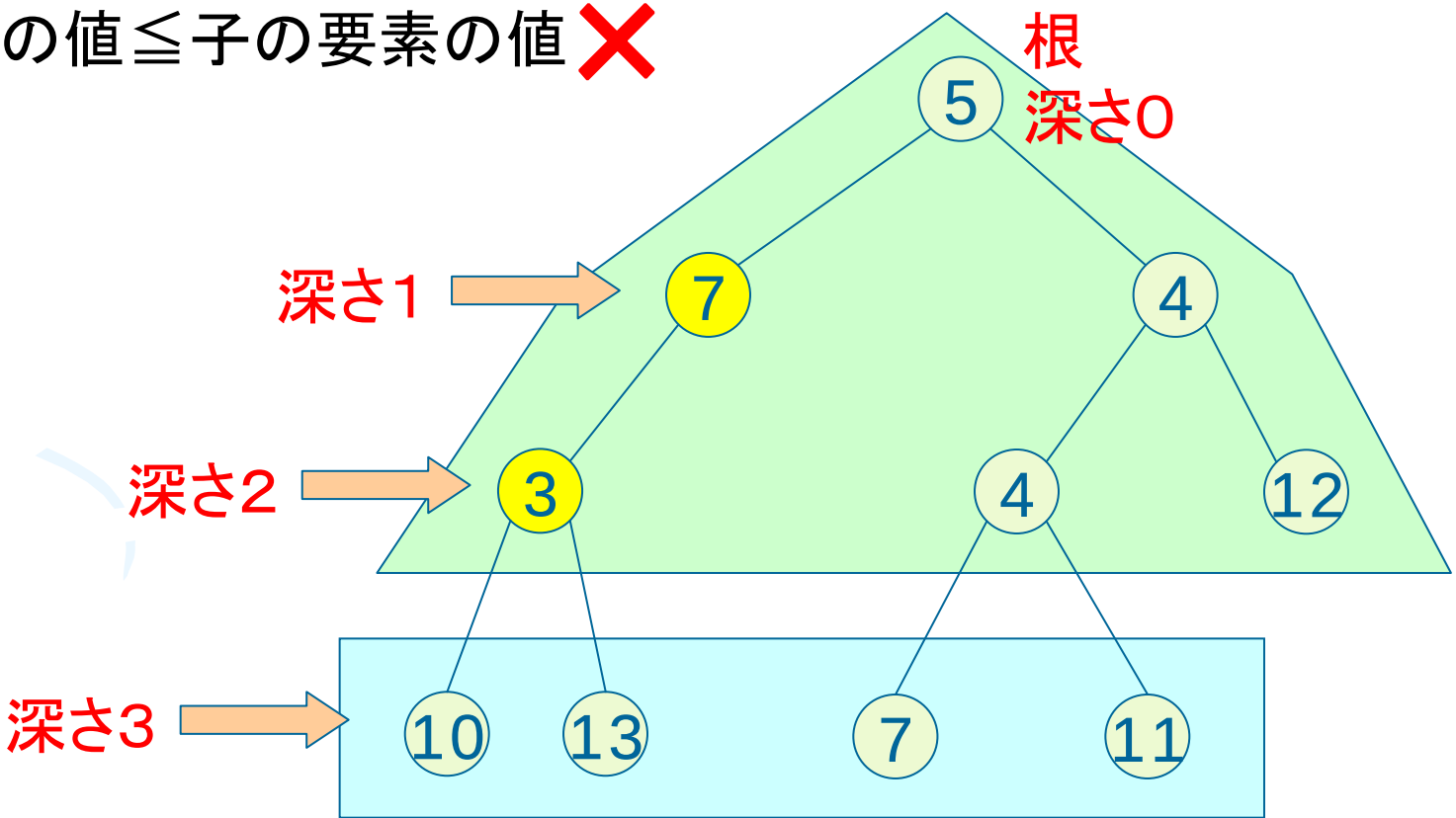
ヒープではない2分木の例

- 次の条件を満たす2分木をヒープと呼ぶ

(1)木の高さが h ➔ 深さ h-1 までは完全2分木 ✗

深さ h では，左側に葉が詰められている ✗

(2)親の要素の値≦子の要素の値 ✗



ヒープの実現方法

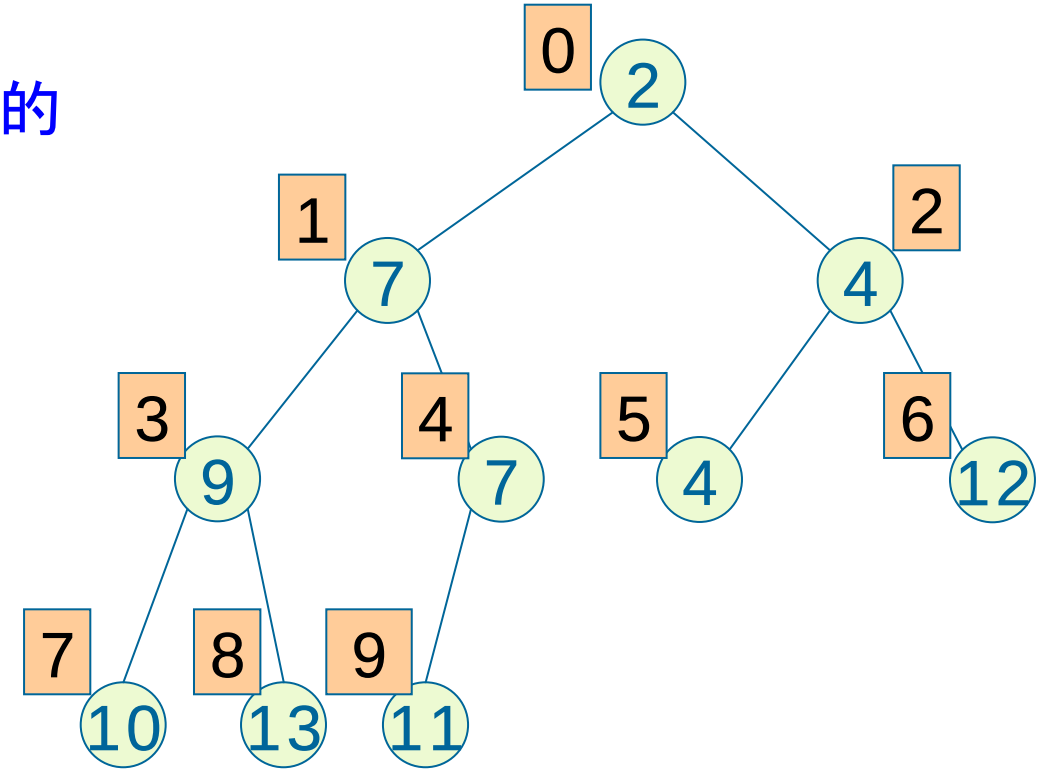
- ヒープは2分木

➔ポインタと構造体を使って実現可能

- ヒープは配列を使っても実現可能←こちらが一般的

0	1	2	3	4	5	6	7	8	9	10
2	7	4	9	7	4	12	10	13	11	-

根から葉に向かって
上から下に，左から右に
番号を付ける
➔この順番で配列に
入れる



ヒープの配列による実現

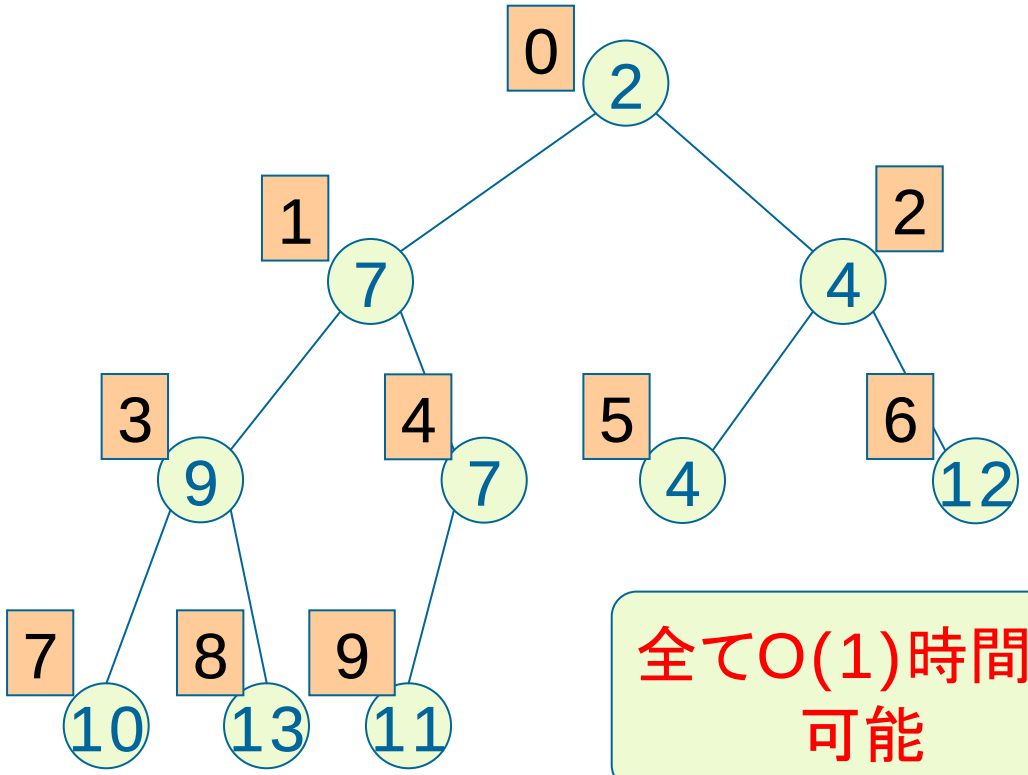
根と最後の要素は
簡単に見つかる:
A[0] は根
A[n-1]は最後の要素
(n: 要素の数)

与えられた要素A[k]の
子が簡単に見つかる:
左の子 A[2k+1]
右の子 A[2k+2]

与えられた要素の
親が簡単に見つかる:
$A[k]$ の親は
$A[\lfloor (k-1)/2 \rfloor]$

根
親
子
最後の要素

0	1	2	3	4	5	6	7	8	9	10
2	7	4	9	7	4	12	10	13	11	-



全てO(1)時間で
可能

最小の要素を削除する

- 根の要素が最小値

（1）根の節点を削除

➔ヒープでなくなる➔木の修正

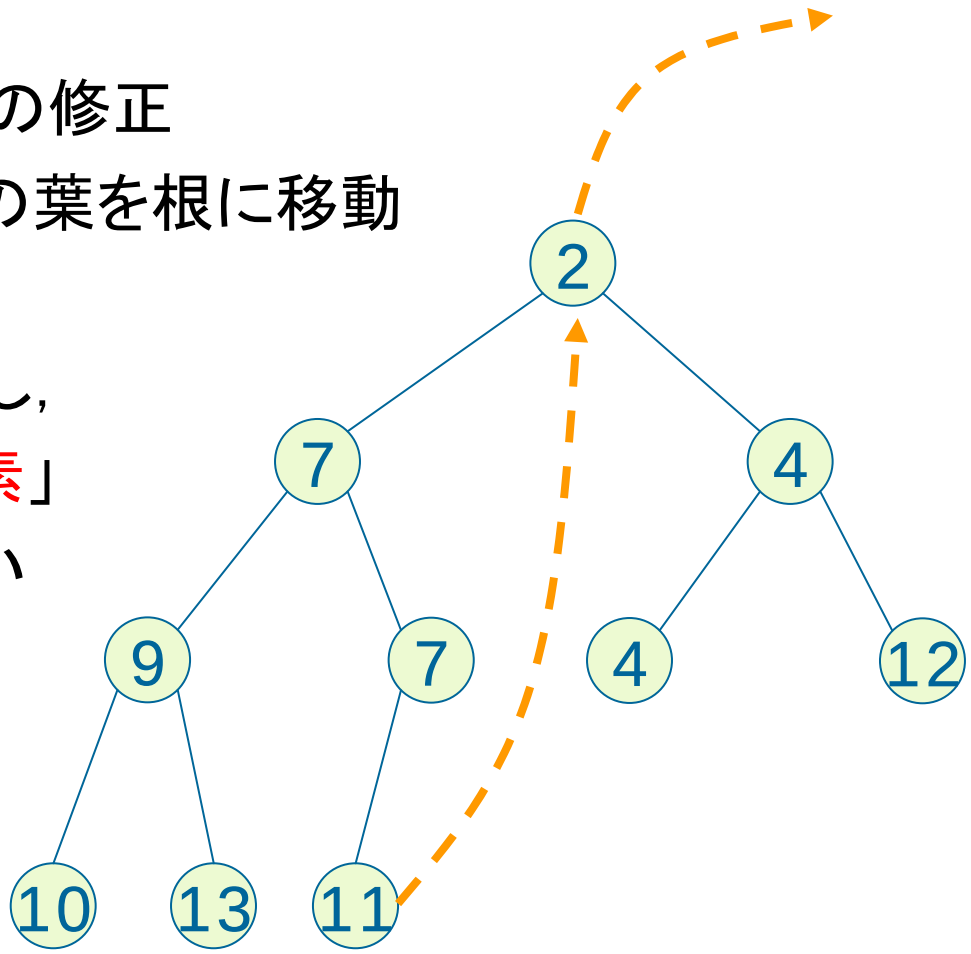
（2）一番深いレベルの右側の葉を根に移動

➔2分木になった

しかし，新しい根に対し，

「親の要素≦子の要素」

の条件は成り立たない



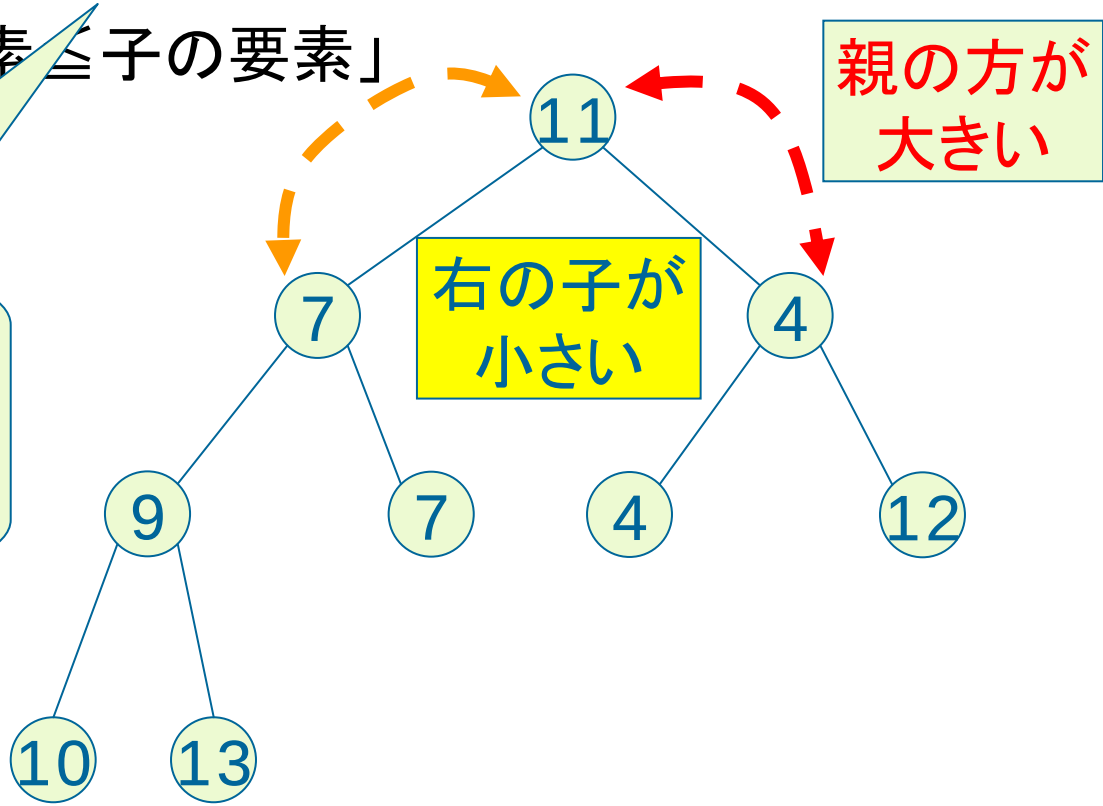
最小の要素を削除する

（3）新しい根の要素を子と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換

– 2つの子のうち，要素の小さい方の子と親を交換

➔ 交換後は「親の要素≦子の要素」が成り立つ

交換する子の選び方を間違えると，条件が成り立たない

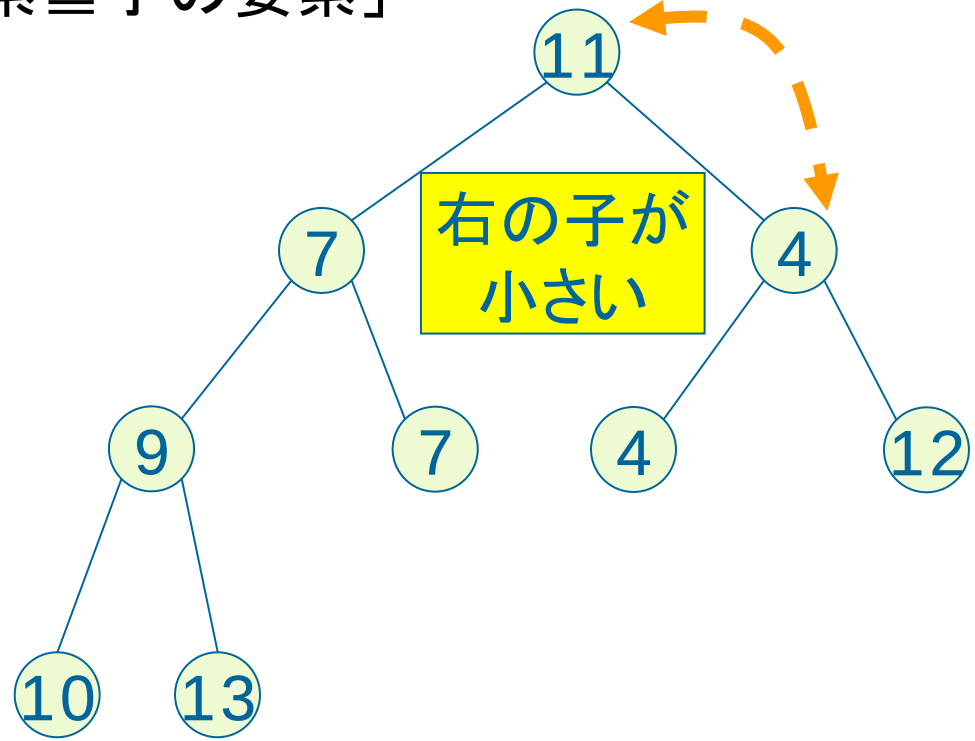


最小の要素を削除する

（3）新しい根の要素を子と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換

– 2つの子のうち，要素の小さい方の子と親を交換

➔ 交換後は「親の要素≦子の要素」が成り立つ



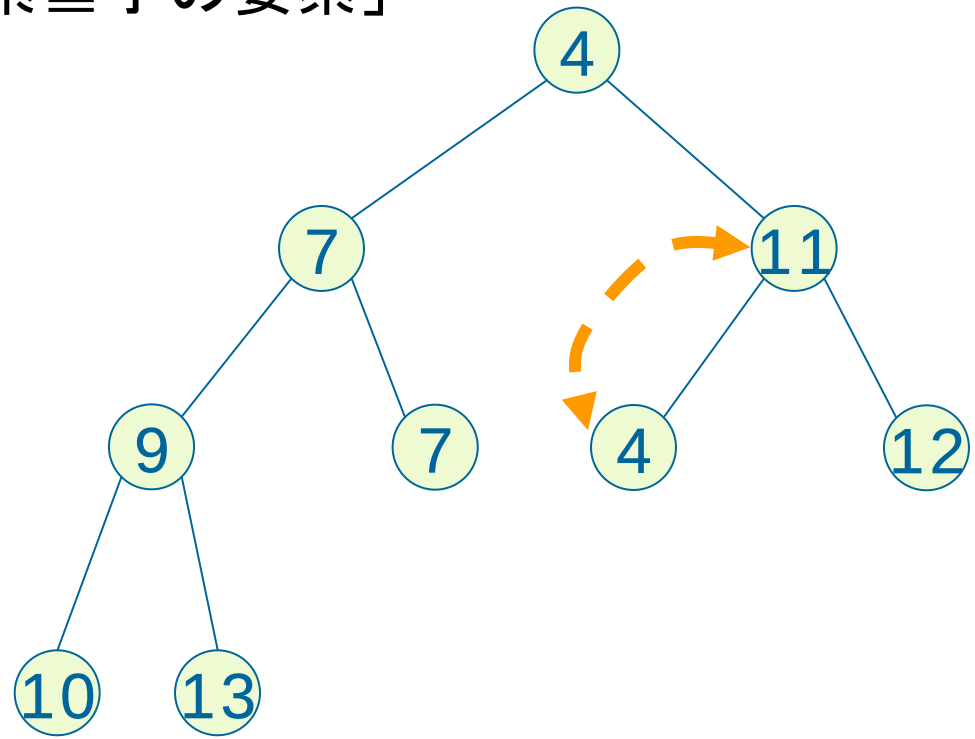
最小の要素を削除する

（3）新しい根の要素を子と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換

– 2つの子のうち，要素の小さい方の子と親を交換

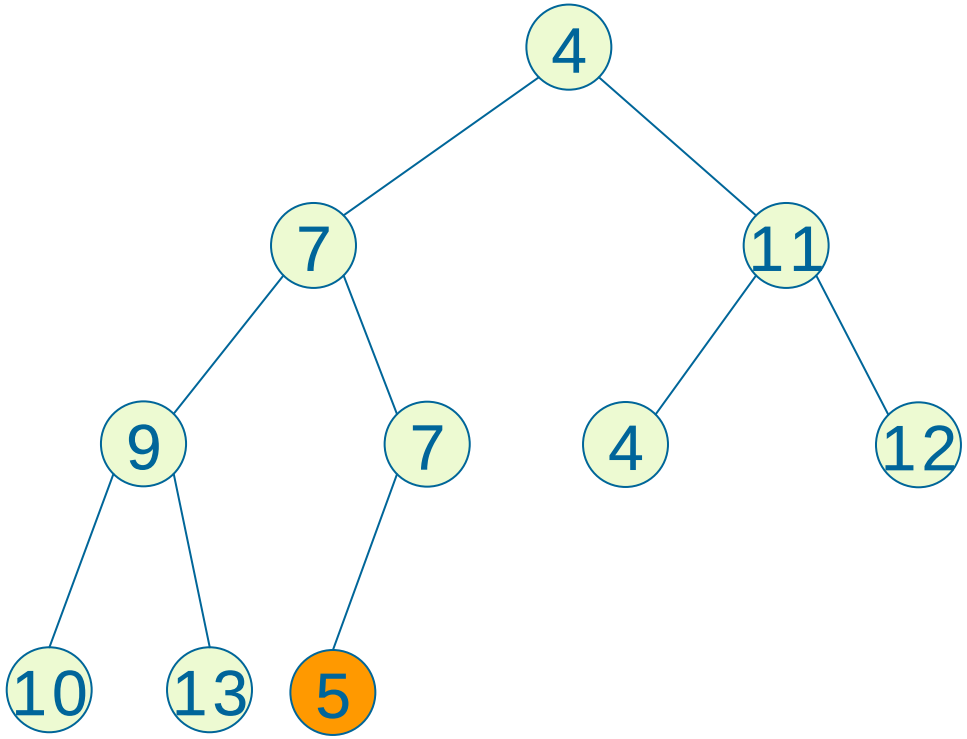
➔ 交換後は「親の要素≦子の要素」が成り立つ

親子の入れ替えの回数
≦ 木の高さ
➔ 時間計算量は
$O(\log n)$



新しい要素を追加する

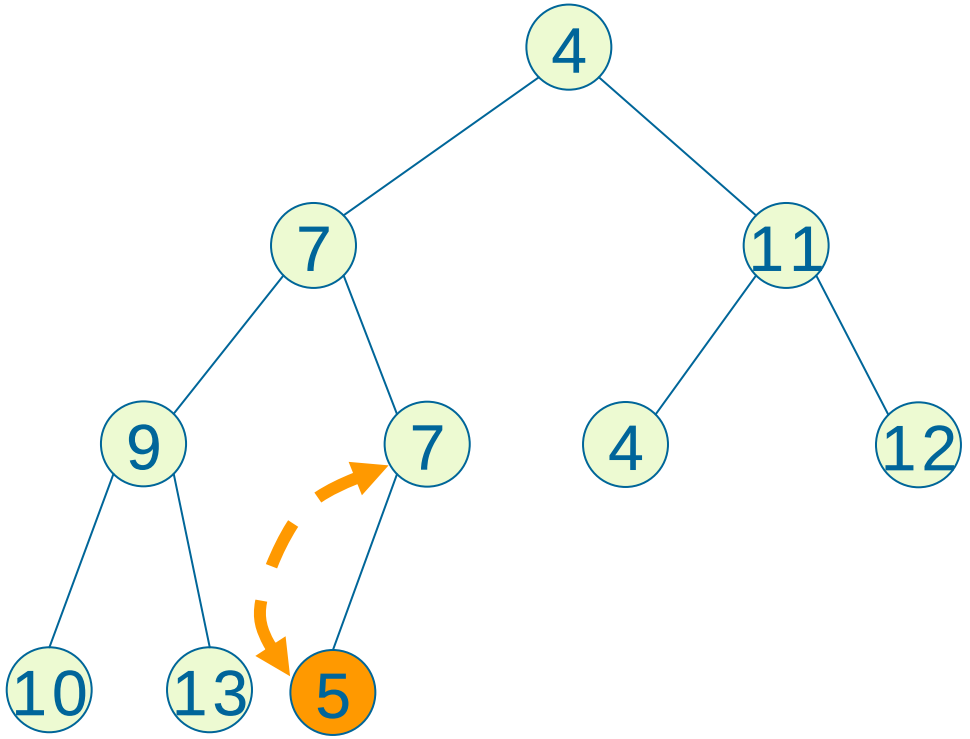
（1）一番深いレベルの最も右側の葉の隣（空きがない場合は次のレベルの一番左）に新しい要素を挿入



新しい要素を追加する

（1）一番深いレベルの最も右側の葉の隣（空きがない場合は次のレベルの一番左）に新しい要素を挿入

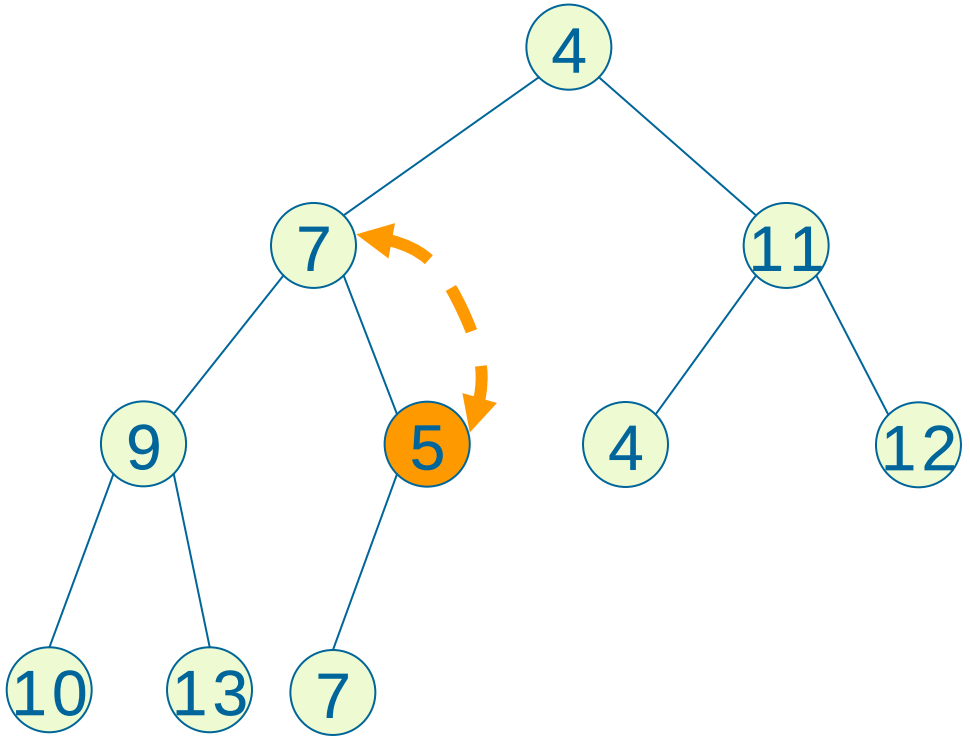
（2）新しい要素を親と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換



新しい要素を追加する

（1）一番深いレベルの最も右側の葉の隣（空きがない場合は次のレベルの一番左）に新しい要素を挿入

（2）新しい要素を親と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換

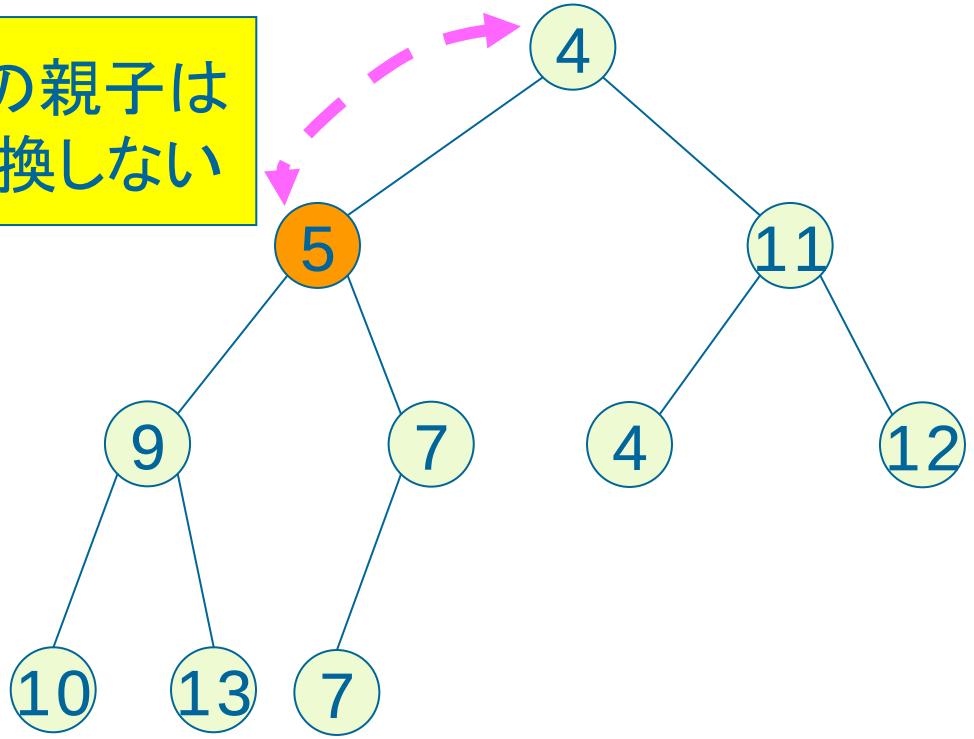


新しい要素を追加する

（1）一番深いレベルの最も右側の葉の隣（空きがない場合は次のレベルの一番左）に新しい要素を挿入

（2）新しい要素を親と比較し，「親の要素＞子の要素」が成り立つときは，繰り返し親子を交換

この親子は
交換しない



ステップ（2）が
行なわれる度に
新しい要素は
一つ上に上がる
➔反復回数≦木の高さ
➔時間計算量は$O(\log n)$

ヒープソート

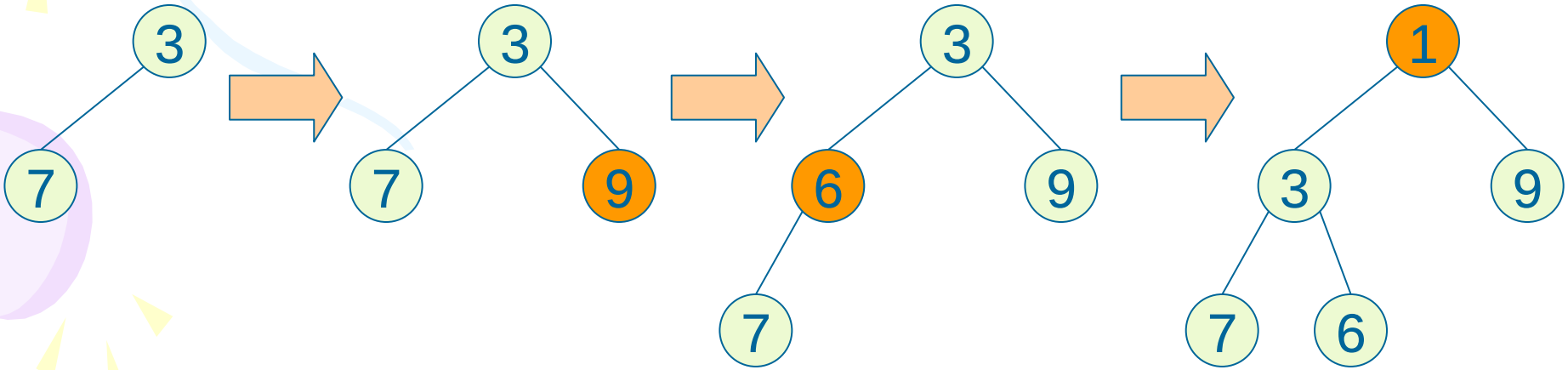
- ヒープを使った整列アルゴリズム
- アルゴリズムの手順

  (1)与えられた数の集合を順番にヒープに追加(n 回)

  (2)ヒープから最小要素を次々に削除，並べる(n 回)

  ➔ O(n log n) 時間

7，3，9，6，1 を順に追加



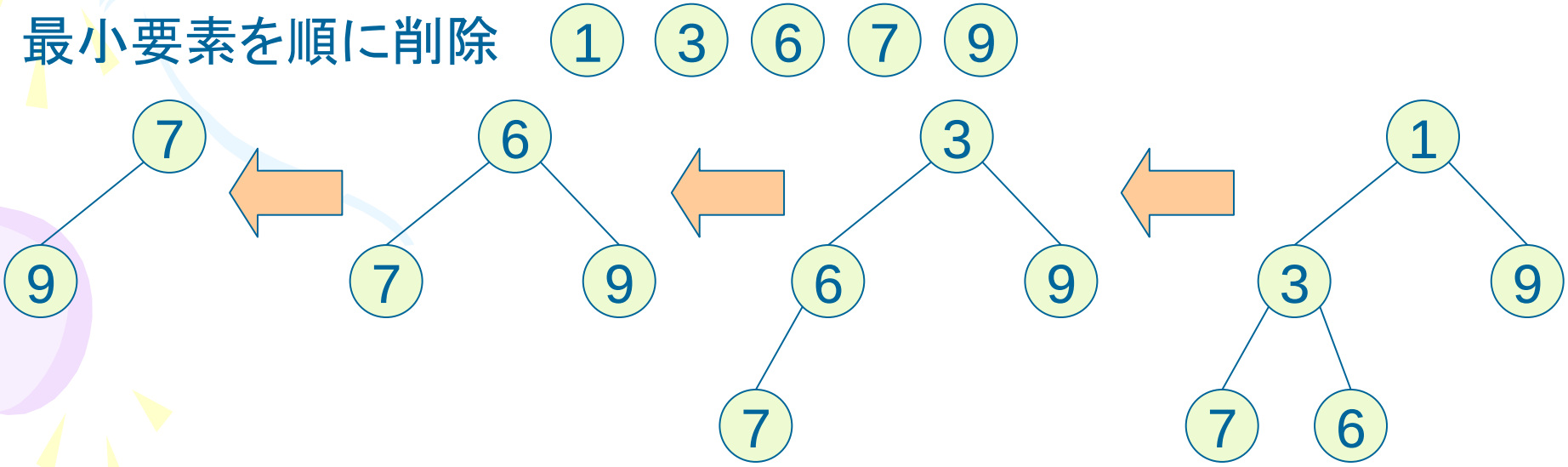
ヒープソート

- ヒープを使った整列アルゴリズム
- アルゴリズムの手順

  (1)与えられた数の集合を順番にヒープに追加(n 回)

  (2)ヒープから最小要素を次々に削除，並べる(n 回)

  ➔ O(n log n) 時間



レポート(プログラム作成)

- これまで授業で紹介した整列アルゴリズムのうち，2つ以上を選んでプログラミングせよ.
- 提出方法：電子メールにファイルを添付して塩浦のアドレスに送付
  - メールの件名は「アルゴリズムとデータ構造第4回レポート」
  - 大学のメールアカウントより提出すること
- レポートをメールにて送信する際，添付するファイルはソースファイルのみ(***.c，***.cpp など).
  実行形式(a.out，***.exeなど)は決して送らないこと！！
- メールには，以下のことを書くこと
  - 学籍番号及び氏名
  - 作成したプログラムに関する簡単な説明(どのアルゴリズムをプログラミングしたか，クイックソートの場合，どのような軸選択法を使ったか，など)
- 締切：5月21日(木)午後7時まで

レポート（紙で提出）

（1）数の集合{15，13，9，5，12，8，7，4，0，6，2，1}に対するヒープを二分木の形で書きなさい

（2）（1）で求めたヒープに対して，最小要素を削除するときの動作を，図を使って詳しく説明せよ．

（3）（2）で求めたヒープに対して，要素10を追加するときの動作を，図を使って詳しく説明せよ．

（4）スタックは連結リスト（双方向ではないリスト）を使っても実現できる．連結リストを用いた場合，追加と削除がO（1）時間で行えることを（図を使って）説明せよ．

締切：５月１５日（木）午前９時頃