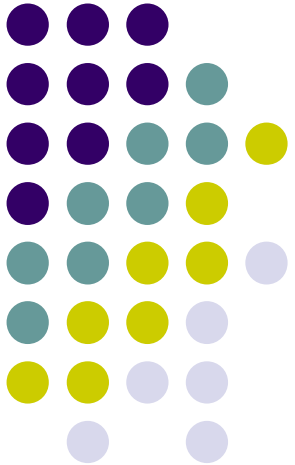
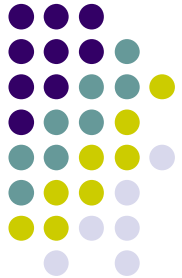
ヒープ

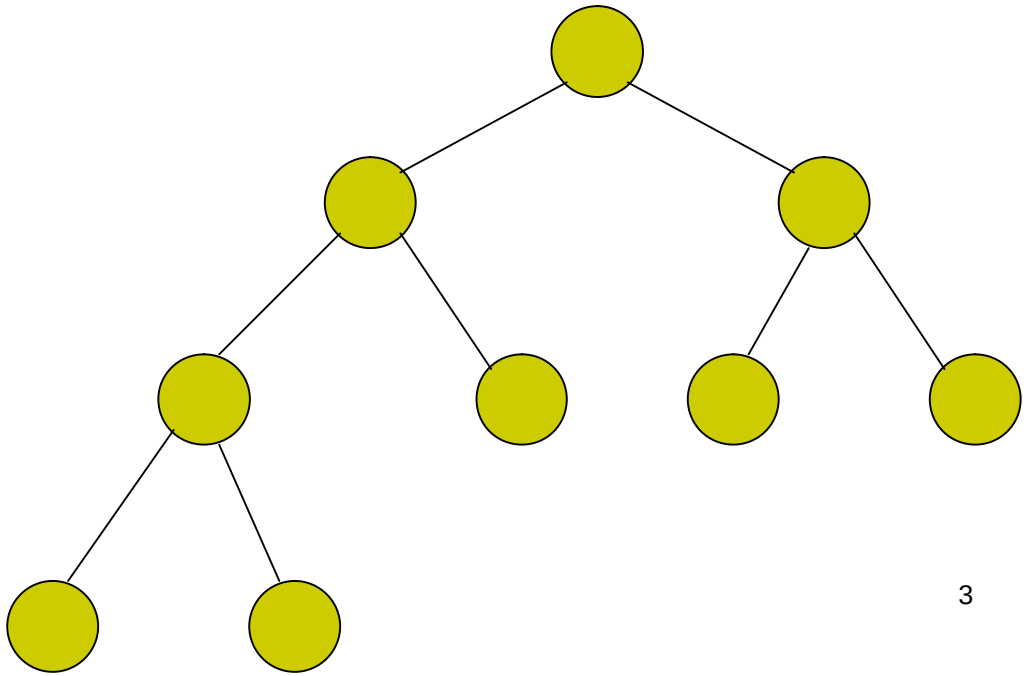


ヒープとは何か



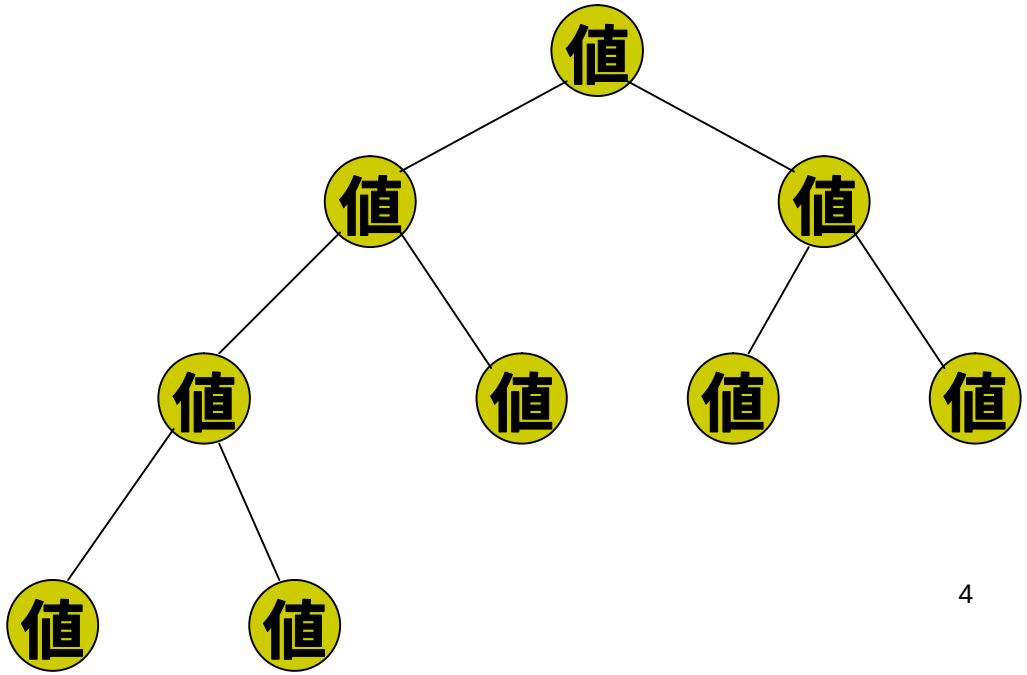
ヒープとは何か

- データ構造に二分木を採用している。



ヒープとは何か

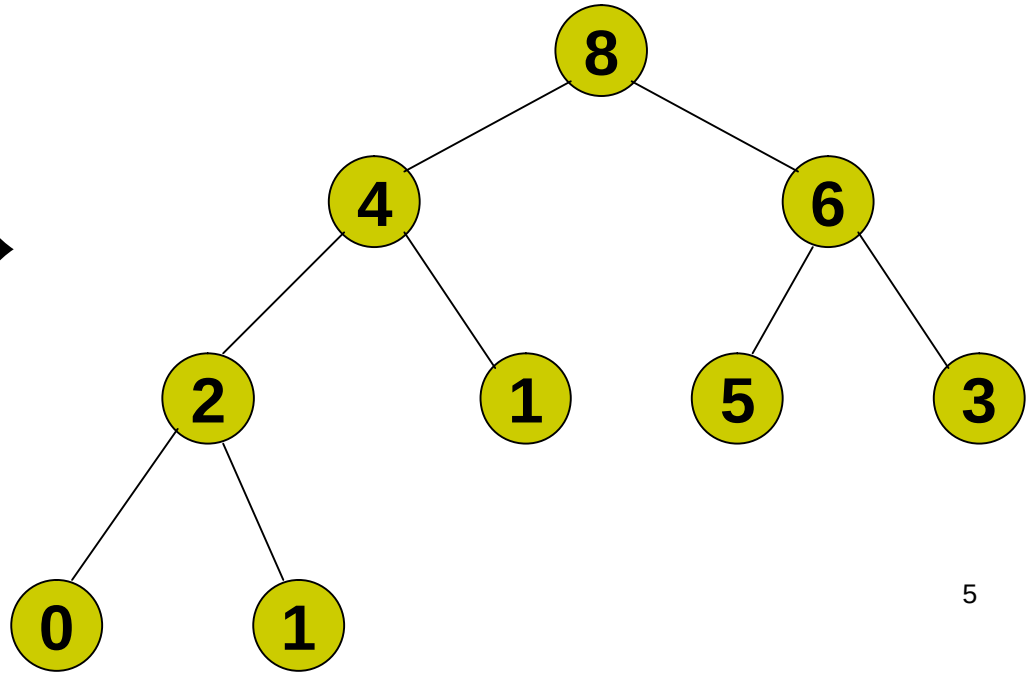
- データ構造に二分木を採用している。
- それぞれの頂点は値を持っている。



ヒープとは何か

- **データ構造に二分木を採用している。**
- **それぞれの頂点は値を持っている。**
- **親の値≧子の値** (Maximum Heap)

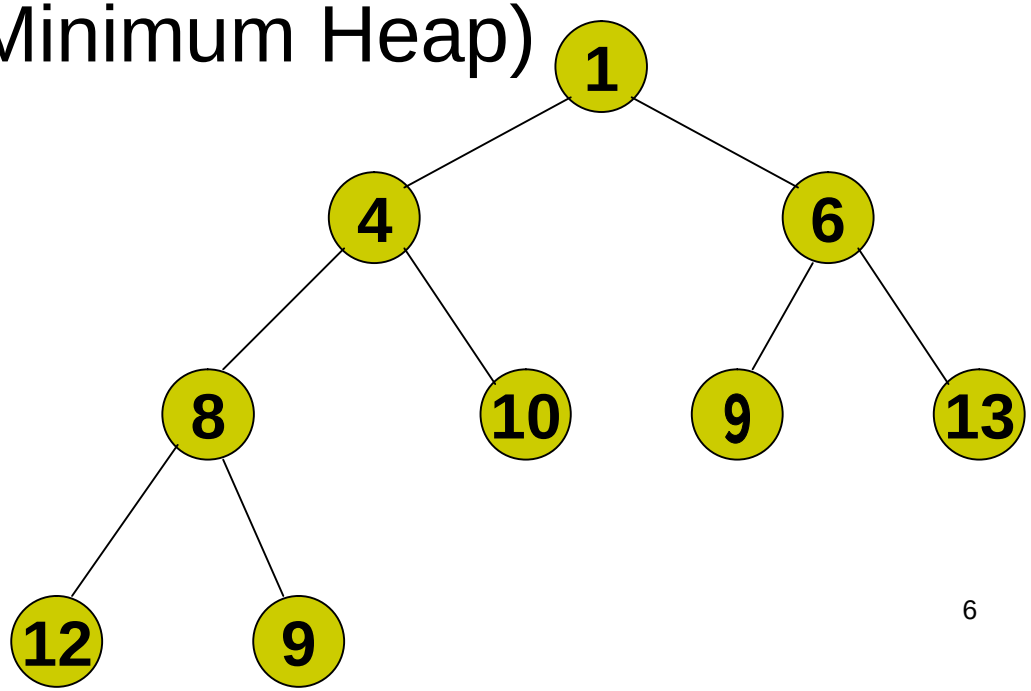
Maximum Heap→



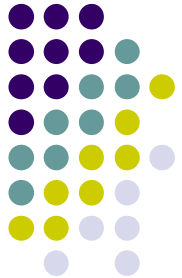
ヒープとは何か

- **データ構造に二分木を採用している。**
- **それぞれの頂点は値を持っている。**
- **親の値≧子の値** (Maximum Heap)
- **親の値≦子の値** (Minimum Heap)

Minimum Heap→



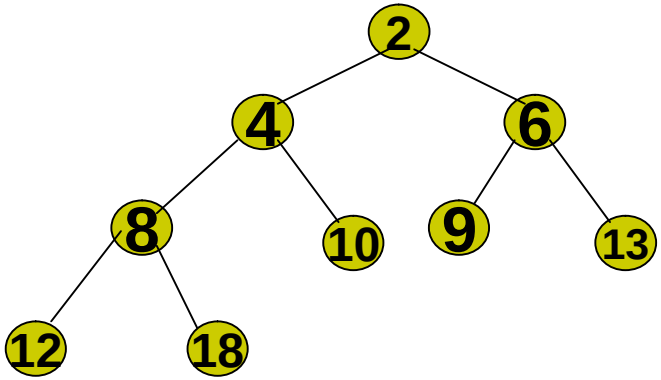
ヒープとは何か



- データ構造に二分木を採用しているが、
  マシンの中では一次元配列に値を保存している。

ヒープの頂点と配列の場所の対応

このヒープは



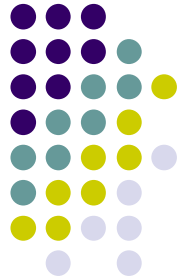
配列でこう表される

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
2	4	6	8	10	9	13	12	18

ヒープを使うことの意義

- **値の挿入、先頭の値の削除の計算量が少ない。**
  $\Theta(\log n)$

ヒープを使うことの意義



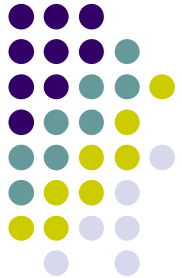
- 値の挿入、先頭の値の削除の計算量が少ない。
  $(\log n)$
- さらに、先頭の値は常に最大（もしくは、最小）

ヒープを使うことの意義

- 値の挿入、先頭の値の削除の計算量が少ない。
  (log n)
- さらに、先頭の値は常に最大 (もしくは、最小)

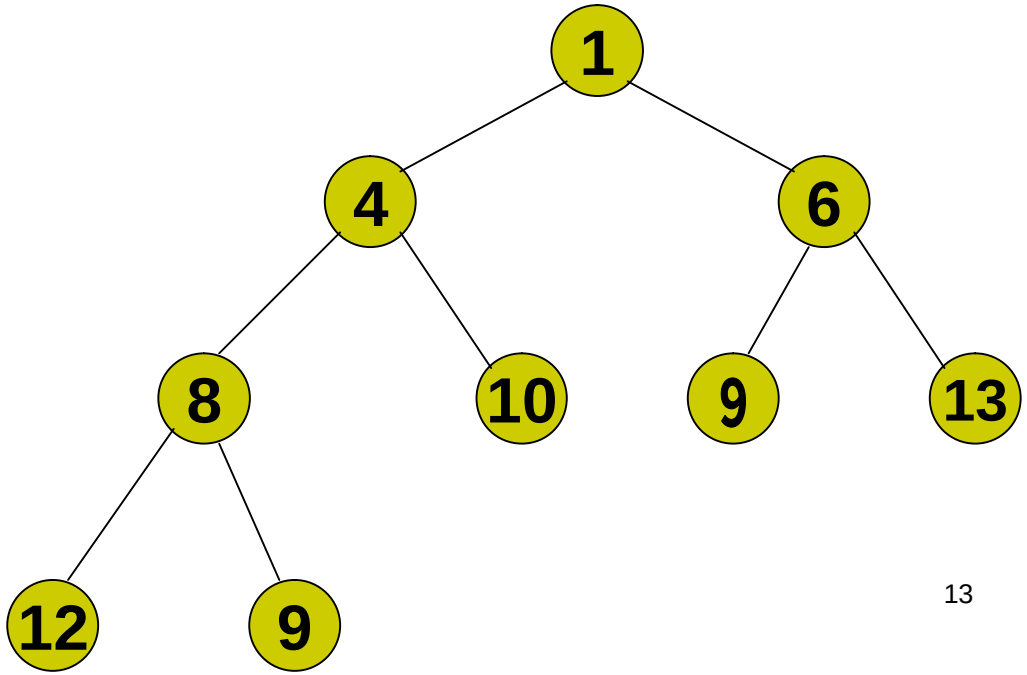
➔リアルタイムで値の挿入や削除を頻繁に行い、
かつ最大 (最小) の値が何なのかを知りたいとき、
ヒープを使うとよい。
(プリンターのジョブ管理,OSのメモリ管理等)

値の挿入



値の挿入

- このヒープに値3を挿入してみる。

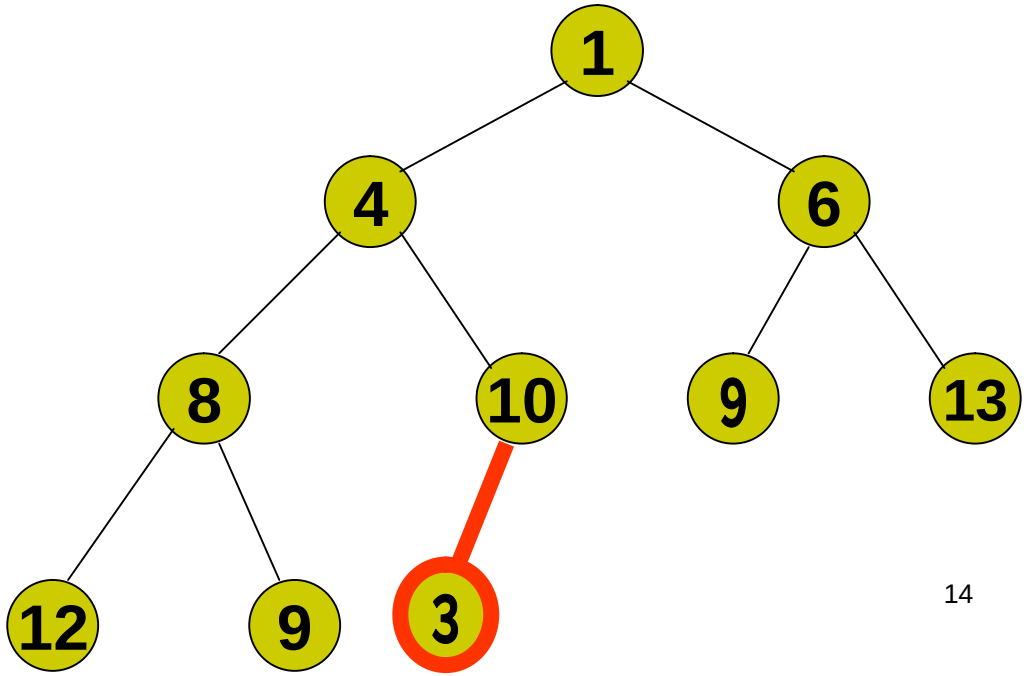


値の挿入

- STEP1

  A[10] = 3;

  ➔ **配列の先に3が追加される。**



値の挿入

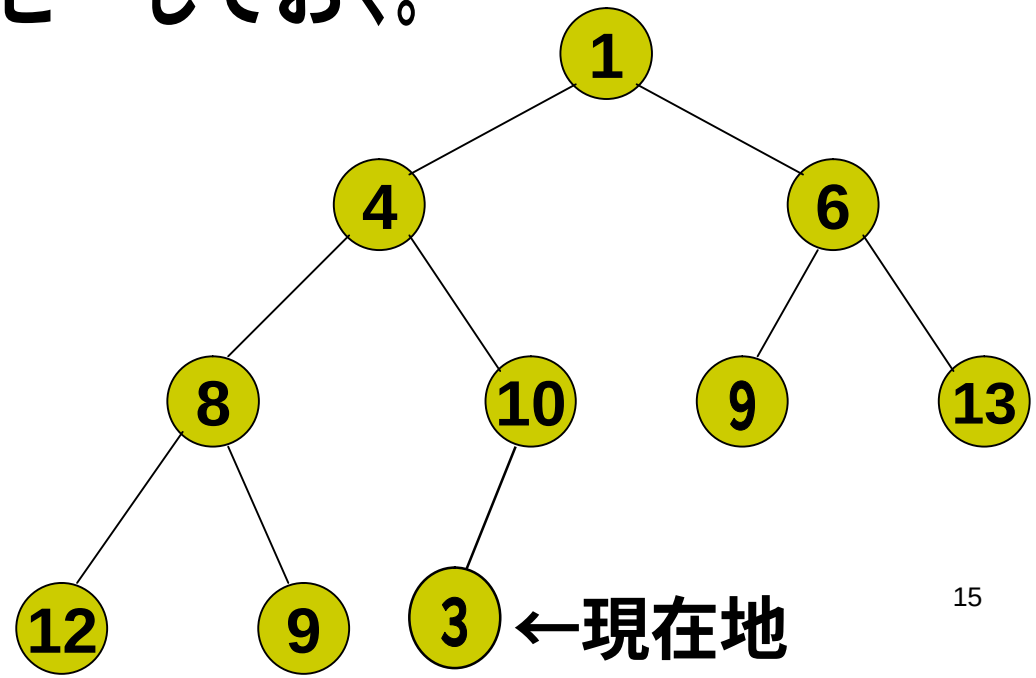
- STEP２

  v = A[10] ;

  ➔A[10]はこの後上書きされるので

  変数vに値をコピーしておく。

v 3



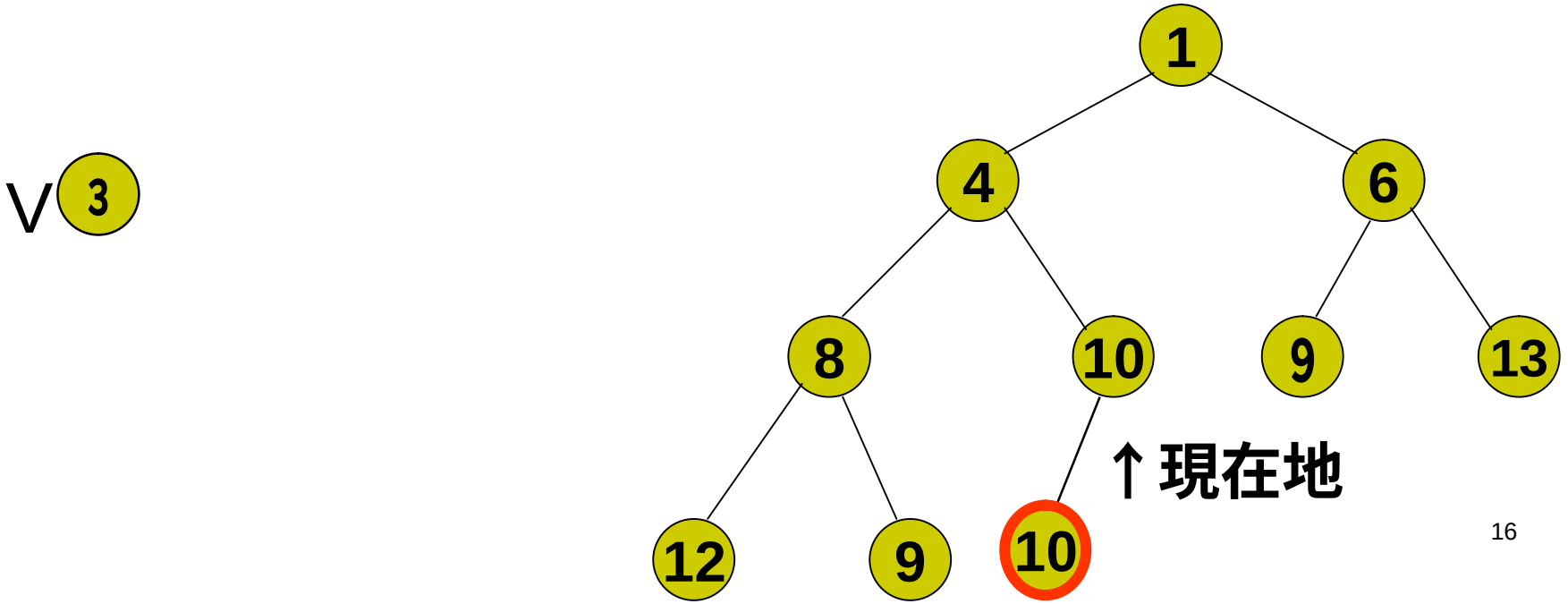
値の挿入

- STEP3

  値vとA[5]を比較する。

  ➔v<A[5]であるので子の値を親の値で上書き。

  現在地を親の場所に移す。



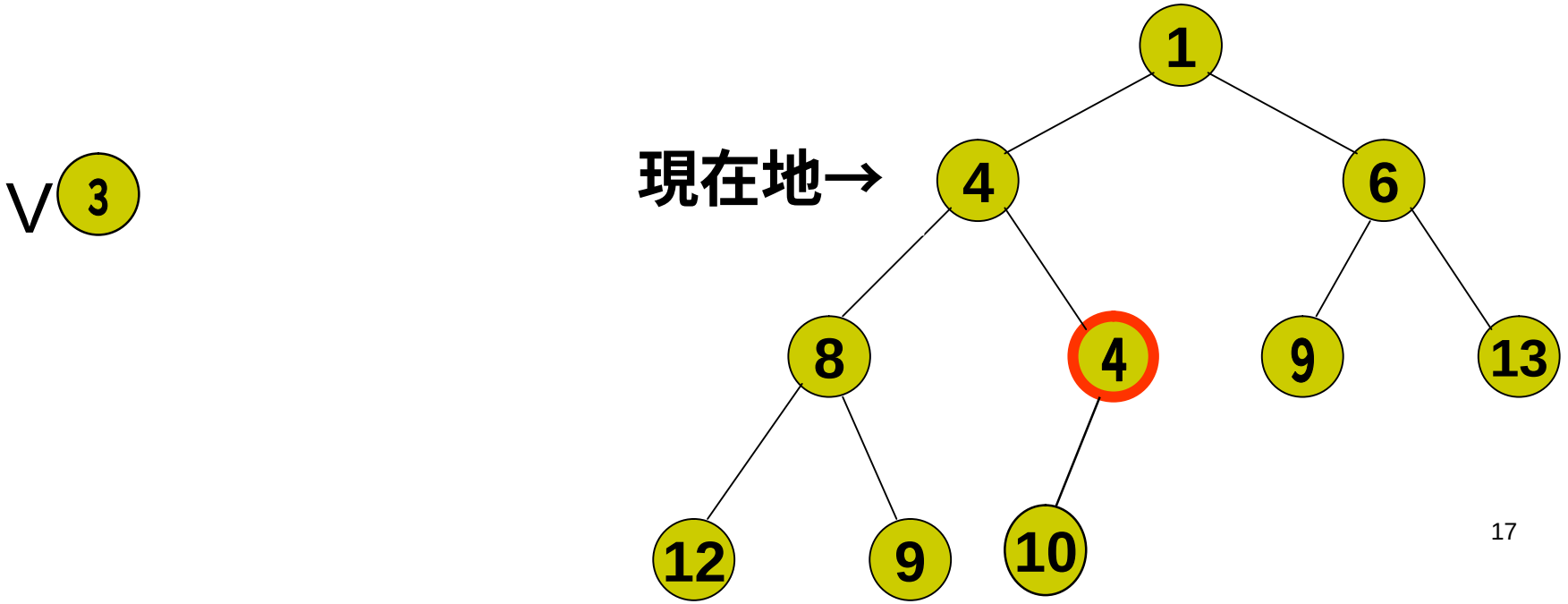
値の挿入

- STEP4

  値vとA[2]を比較する。

  ➔v<A[2]であるので子の値を親の値で上書き。

  現在地を親の場所に移す。



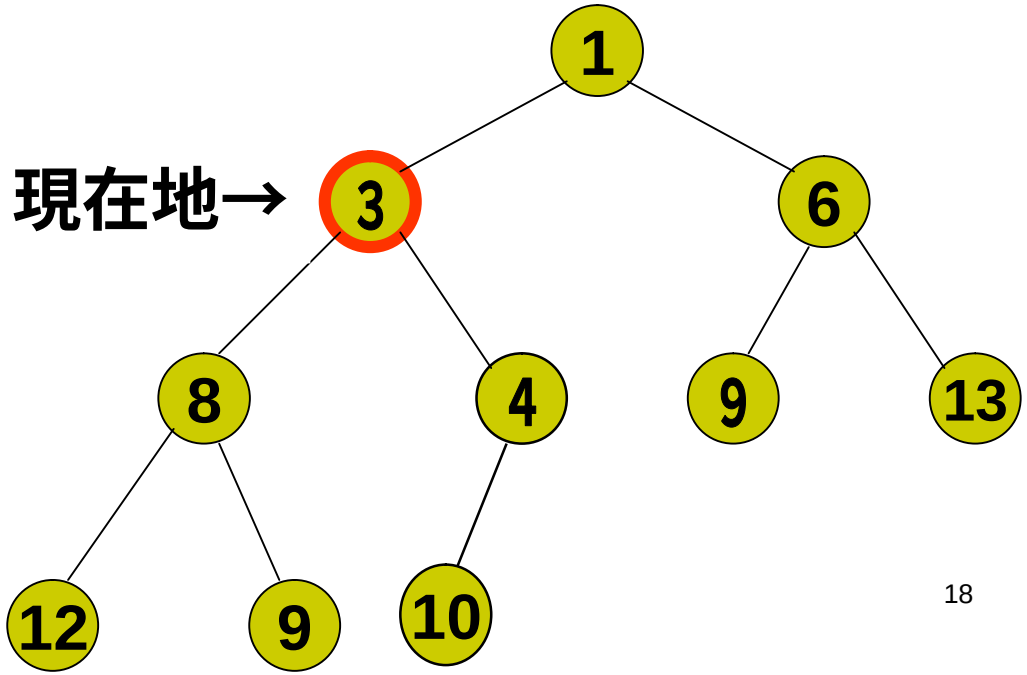
値の挿入

- STEP5

  値vとA[1]を比較する。

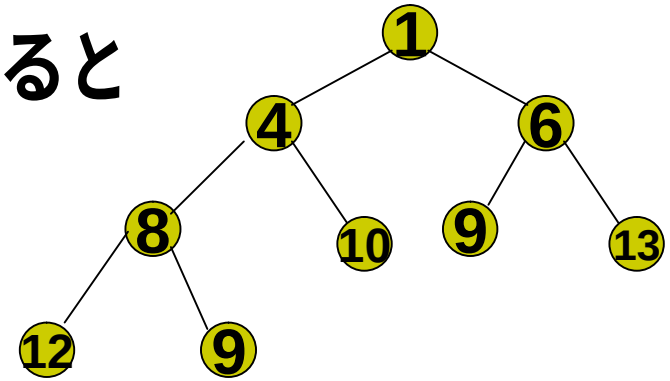
  ➔v>A[2]であるので現在地の値をvで上書き。

  これで終了。

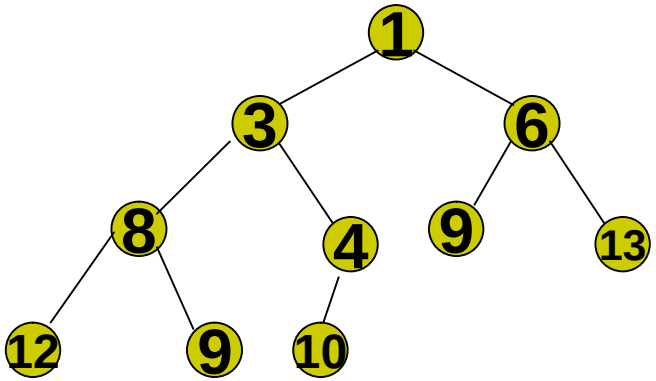


値の挿入

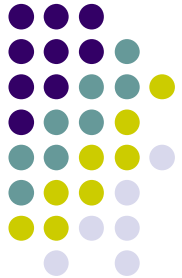
このヒープに3を挿入すると



こうなる→

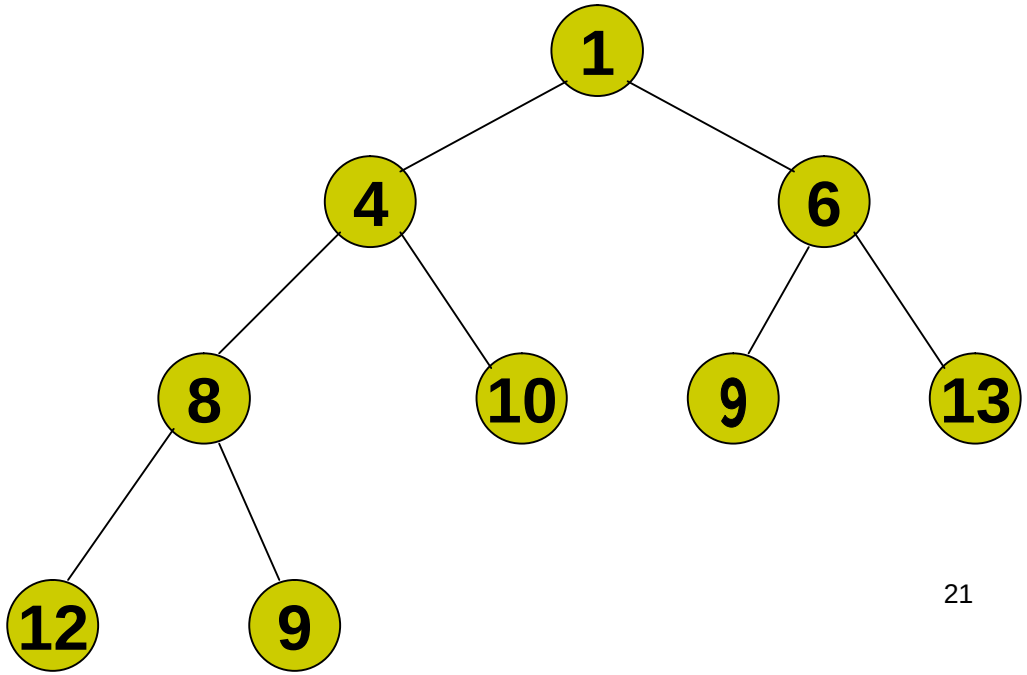


先頭の値の削除



先頭の値の削除

- このヒープの先頭の値を削除する。



先頭の値の削除

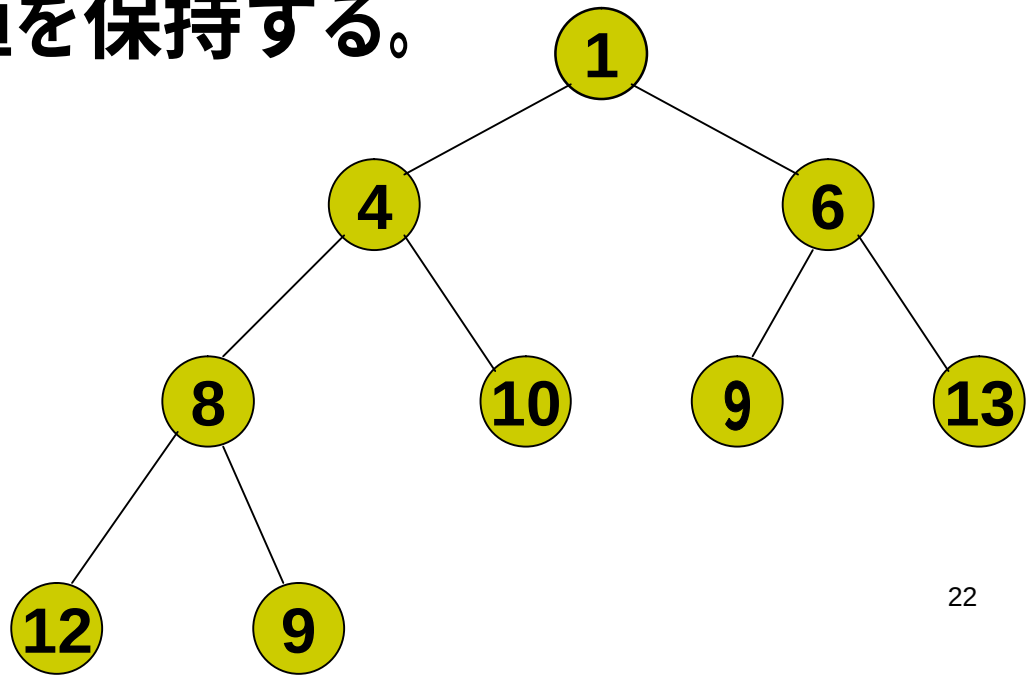
- STEP1

  ret=A[1];

  ➔A[1]**は上書きされるので**

  変数ret**に値を保持する。**

ret 1



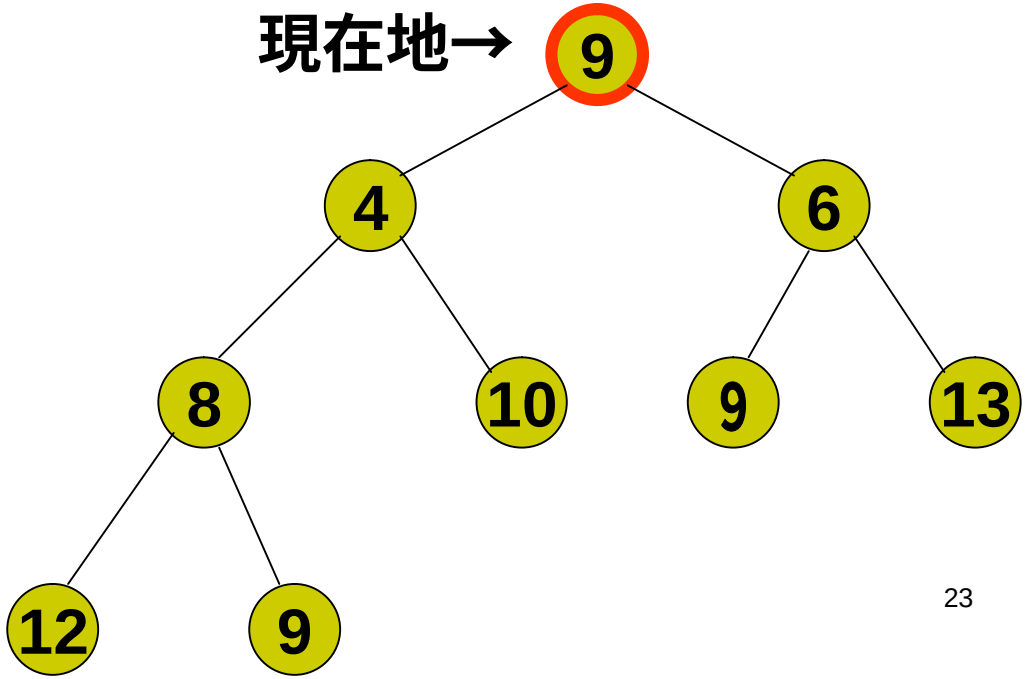
先頭の値の削除

- STEP2

  A[1]=A[n];

  ➔ヒープの一番後ろの値を先頭に入れる。

ret 1



先頭の値の削除

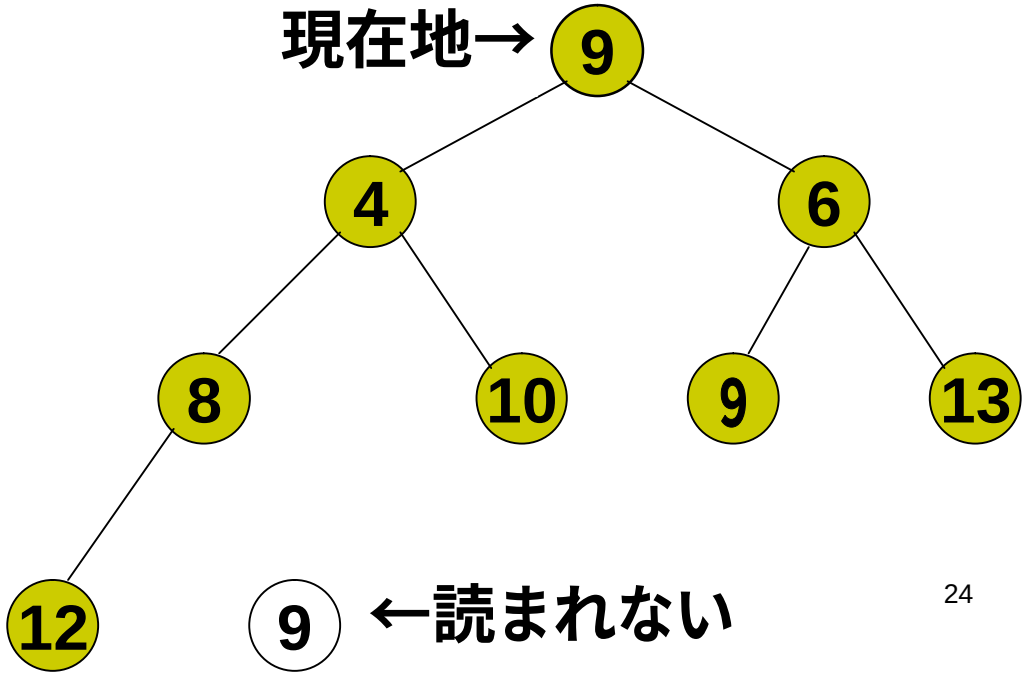
- STEP３

  n--;

  ➔ヒープの要素数を一個減らす。

  ヒープの一番後ろの値は今後読み込まれない。

ret 1



先頭の値の削除

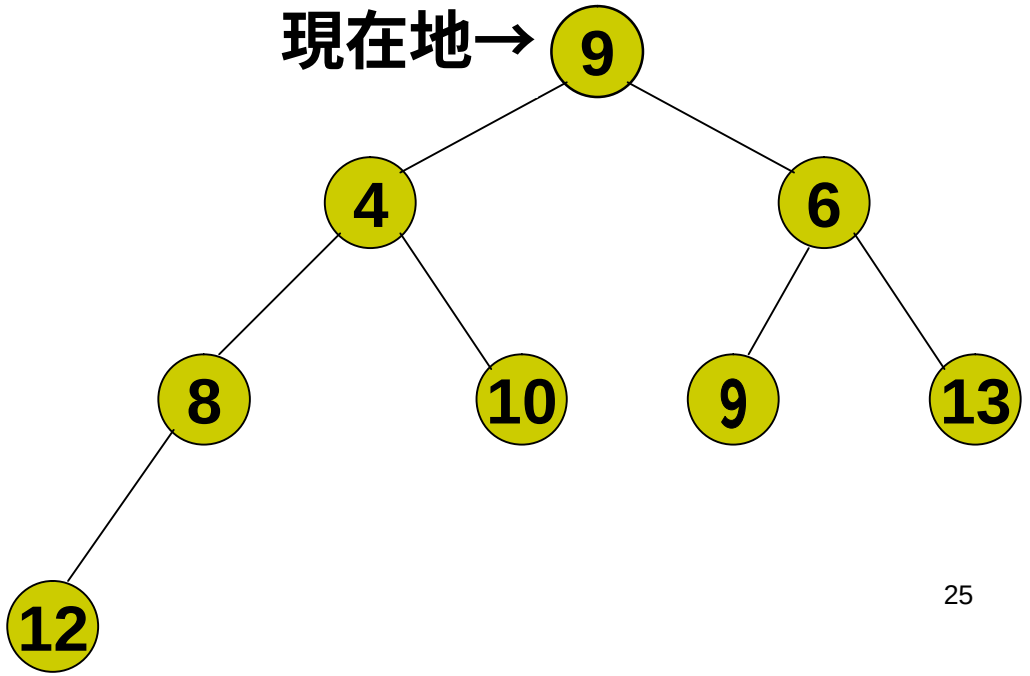
- STEP４

  v=A[1];

  ➔ A[1]は上書きされるので変数vに値を保持する。

ret 1

v 9



先頭の値の削除

- STEP5

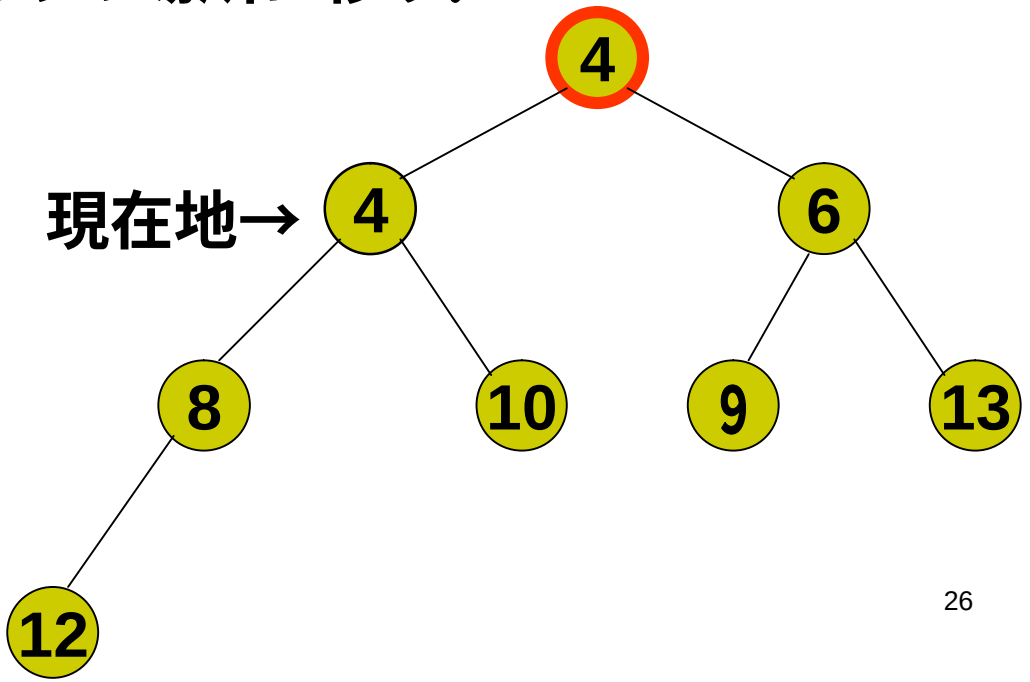
  値A[2]とA[3]を比較する。

  ➔現在地の値を小さいほうの値で上書き。

  現在地を小さいほうの場所に移す。

ret 1

v 9



先頭の値の削除

- STEP6

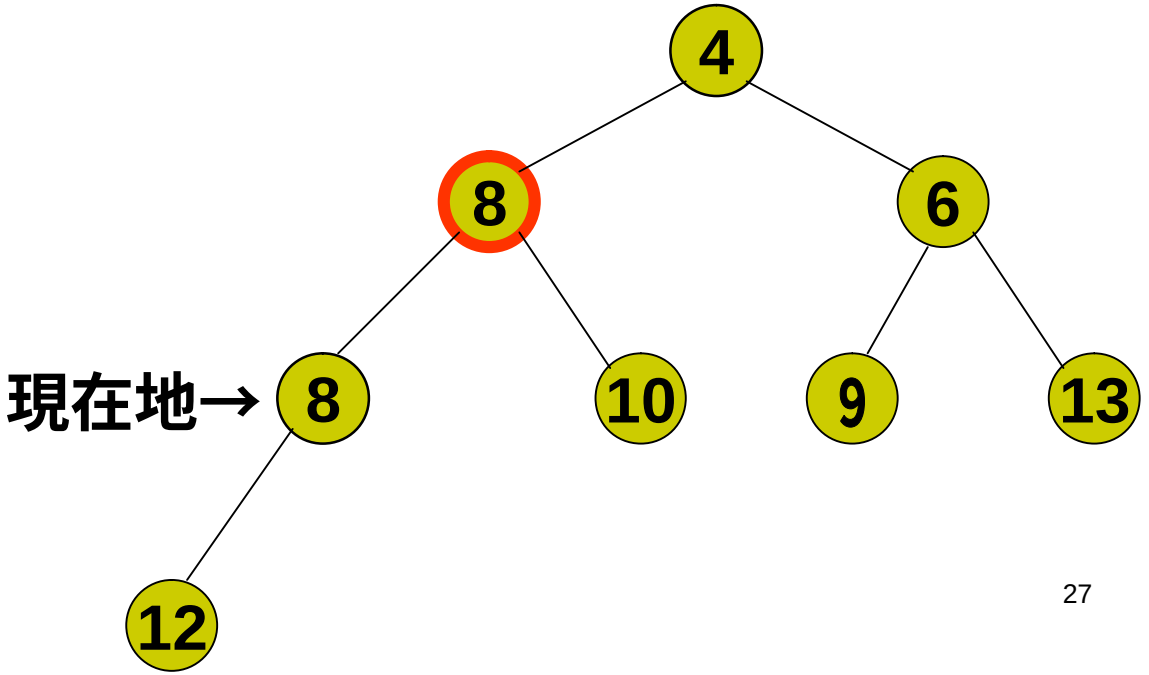
  値A[4]とA[5]を比較する。

  ➔現在地の値を小さいほうの値で上書き。

  現在地を小さいほうの場所に移す。

ret 1

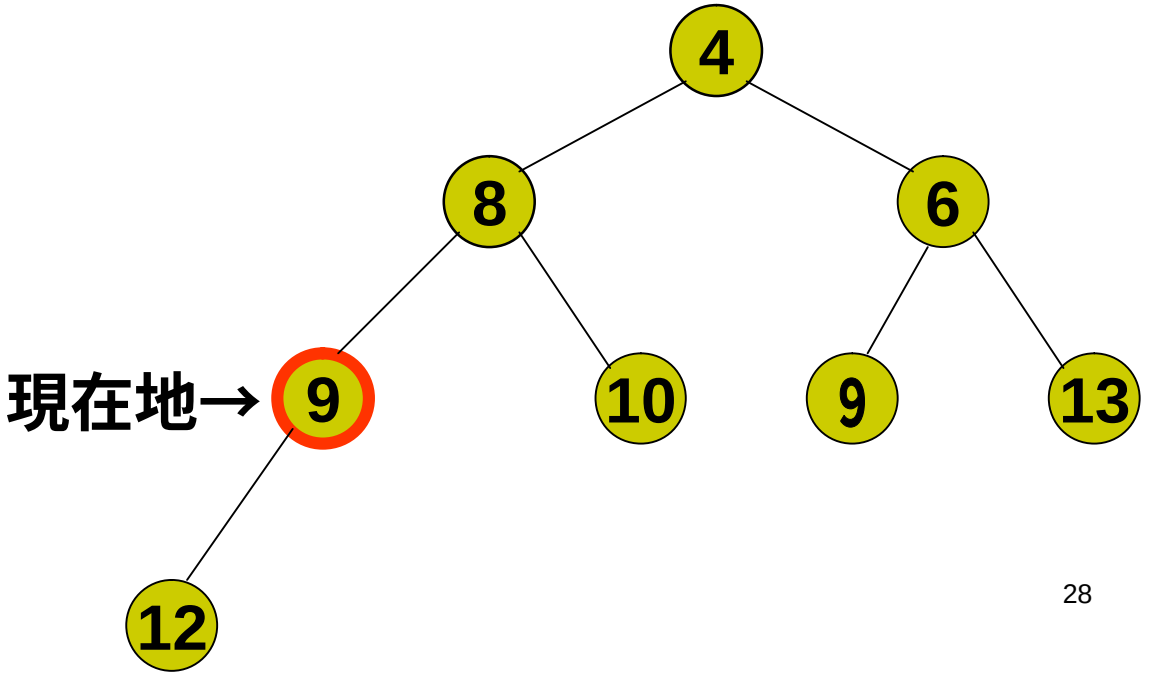
v 9



先頭の値の削除

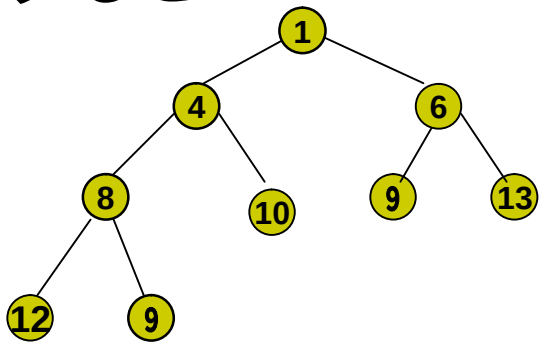
- STEP7
  vと子の値を比較する。
  ➔現在地の値をvの値で上書き。
  これで終了。

ret 1
v 9



先頭の値の削除

このヒープの先頭の値を削除すると



こうなる→