KYOTO UNIVERSITY

アルゴリズムとデータ構造⑥
～ 探索問題（二分探索木） ～

鹿島久嗣

DEPARTMENT OF INTELLIGENCE SCIENCE AND TECHNOLOGY

探索問題：
データ集合から所望の要素を見つける

- 探索問題は、データ集合から特定のデータを見つける問題
  - データは「キー」と「内容」からなる
  - 与えられたキーに一致するキーをもったデータを見つけ、その内容を返す
- これは、**二分探索木**や**ハッシュ**等によって実現可能

今回の話題

二分探索木

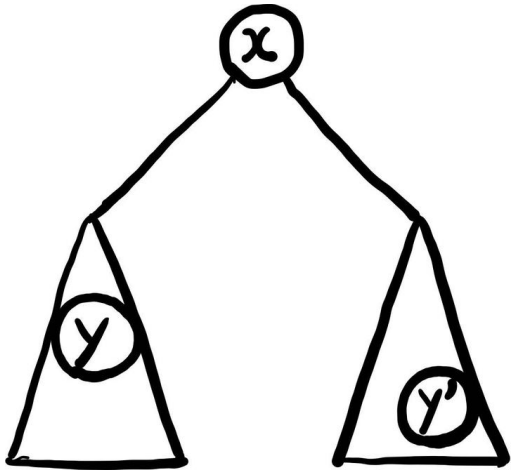
二分探索木：
データ集合から所望の要素を見つけるデータ構造

- 各節点が key, left（左の子）, right（右の子）, p（親） をそれぞれ最大１つもつ二分木
- キーには順序がつけられる；２つの節点x, yに対して
  $\mathrm{key}(x) = \mathrm{key}(y), \mathrm{key}(x) > \mathrm{key}(y), \mathrm{key}(x) < \mathrm{key}(y)$
  のいずれかが成り立つ
- キーは以下の条件を満たす：

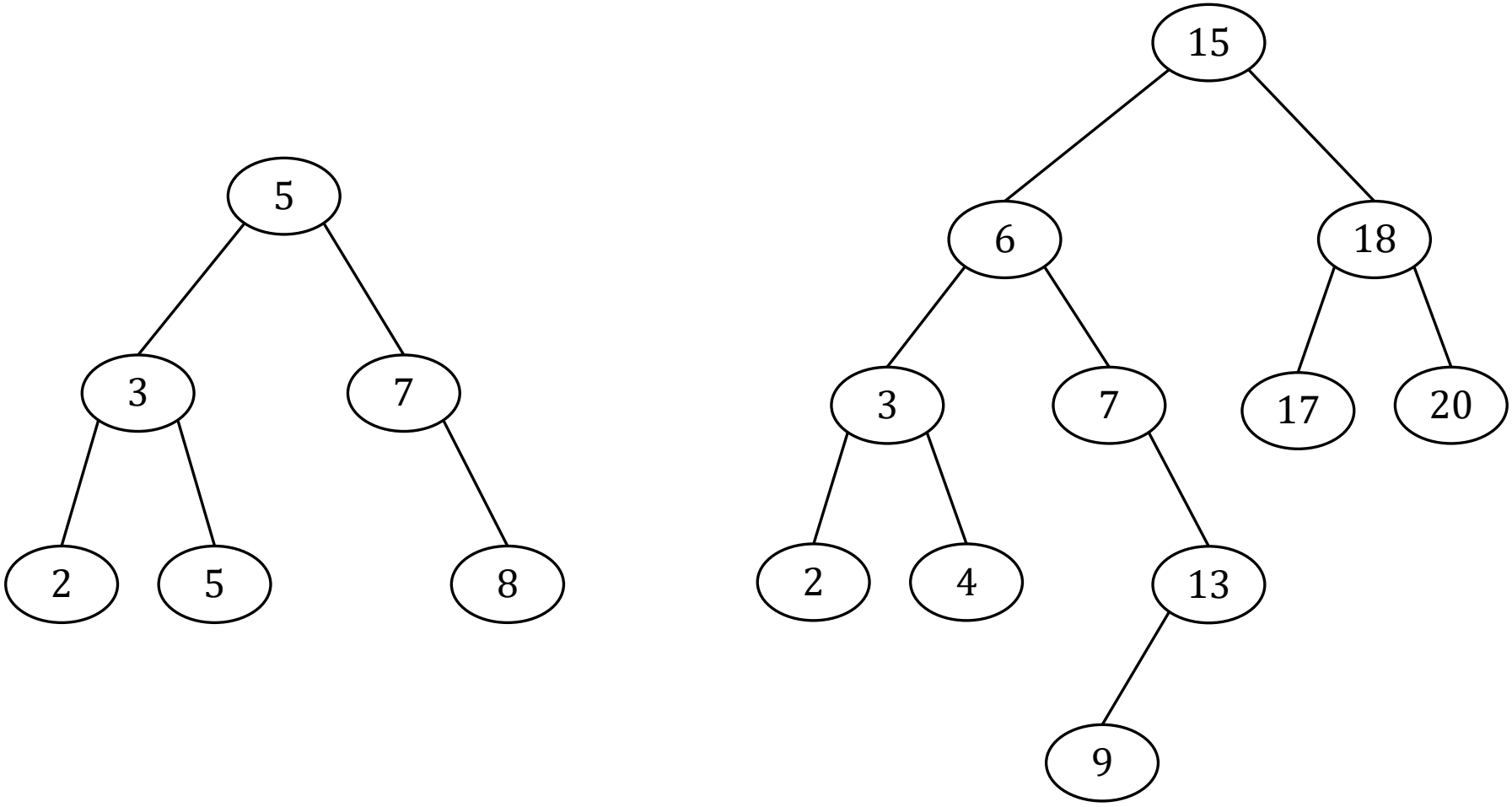
  $y \in x$の左の子を根とする部分木

  $y' \in x$の右の子を根とする部分木

  とすると $\mathrm{key}(y) \leqq \mathrm{key}(x) \leqq \mathrm{key}(y')$ が成立



二分探索木の例：
二分探索木の条件を満たすことを確認



二分探索木を用いた探索：木の高さに比例する時間で可能

- キーの満たす条件を用いて$O(h)$で発見（hは木の高さ）
- $\mathrm{SEARCH}(x, k)$：これを「x =根」で呼ぶ; kは探したいkey

  if $x = \mathrm{NULL}$ または $k = \mathrm{key}(x)$ then xを返す

  if $k < \mathrm{key}(x)$ then $\mathrm{SEARCH}(\mathrm{left}(x), k)$：左にあるはず

  if $k > \mathrm{key}(x)$ then $\mathrm{SEARCH}(\mathrm{right}(x), k)$ ：右にあるはず
- $\mathrm{SEARCH}(x, k)$：再帰を用いない場合の方法

  while $x \neq \mathrm{NULL}$ または $k \neq \mathrm{key}(x)$

  　　if $k < \mathrm{key}(x)$ then $x \leftarrow \mathrm{left}(x)$ else $x \leftarrow \mathrm{right}(x)$

  end while; xを返す

二分探索木からソート済み配列を取り出す：中順での巡回による要素列挙

- 二分探索木から、全てのキーを整列された順で出力できる

INORDER(x)：中順での巡回（これを x ＝ 根 で呼ぶ）

if xが葉 then key(x)を出力

else

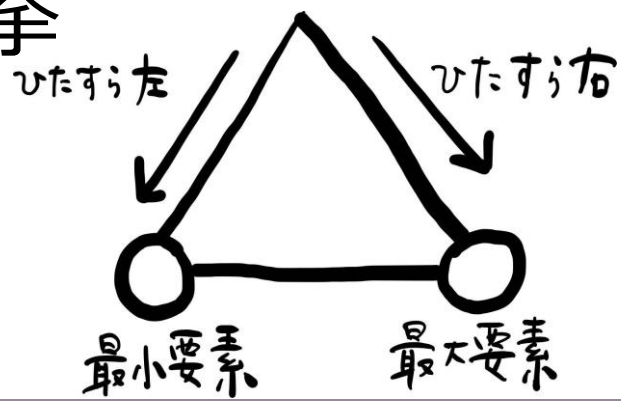
この順序が重要

① INORDER(left(x))：x以下の要素が列挙される

② key(x)を出力

③ INORDER(right(x))：x以上が列挙

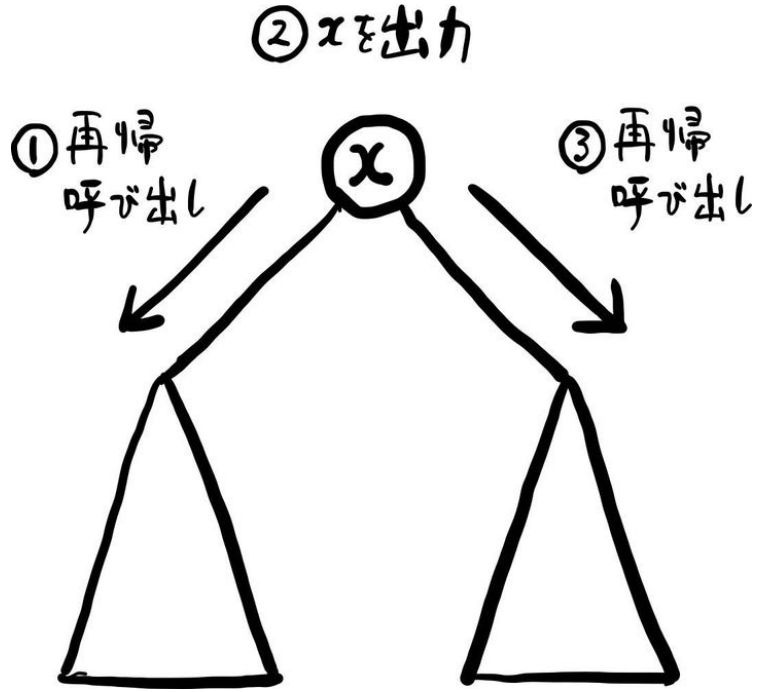
end if



- なお、最小（最大）の要素の発見は left （right）をたどることでO(h)で可能

前順・後順での巡回：要素出力のタイミングによって異なる巡回順になる

- PREORDER(x)：前順での巡回
  - ② key(x) を出力　（要素を出力するタイミングに注意）
  - ① PREORDER(left(x))
  - ③ PREORDER(right(x))
- POSTORDER(x)：後順での巡回
  - ① POSTORDER(left(x))
  - ③ POSTORDER(right(x))
  - ② key(x)を出力
- 出力の位置に注意（中順は①→②→③）



次節点・前節点：次に小さい（大きい）要素を取り出す

- 次節点(successor)：中順で次の節点（＝次に小さい）
- 前節点(predecessor)：中順でひとつ前の節点
- SUCCESSOR(x)：次節点の発見

if $\mathrm{right}(x) \neq \mathrm{NULL}$ then $\mathrm{MININUM}(\mathrm{right}(x))$

$y \leftarrow \mathrm{parent}(x)$

右の子がいるなら
その右部分木の最小要素が次接点

while $y \neq \mathrm{NULL}$ かつ $x = \mathrm{right}(y)$

　　$x \leftarrow y; y \leftarrow \mathrm{parent}(x)$

自分が親の右の子である限り
上にあがっていく

（親は自分より先に挙げられるはずなので
親は次節点ではない）

end while

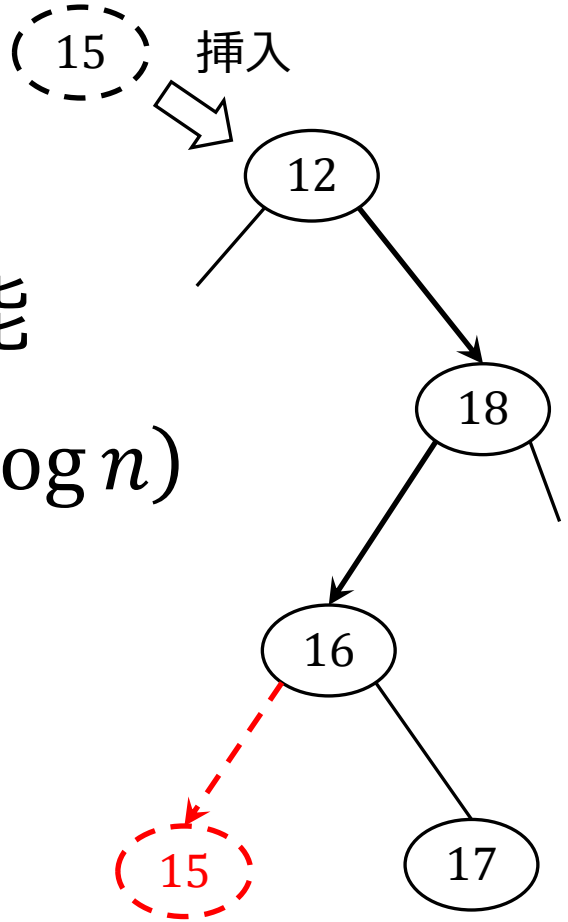
xを返す

自分が親の左の子なら
親が次節点のはず

（中順では、親は自分の次に挙げられる）

二分探索木への新たな要素の挿入：挿入は、実際に探索してみることで$O(h)$で実行可能

- 探索と同様にkeyの比較で辿っていき、該当する節点がなくなった時にそこに入れる
- 高さhの木では$O(h)$時間かかる
- これを繰り返して二分探索木を構成可能
  - ランダムな順で挿入すれば平均高さ$O(\log n)$
    - $\approx 1.39 \log n$
  - 最悪の場合には、高さn



二分探索木からの要素削除：次節点（か前節点）で置き換える

- ３つの場合に分けて考える

1. 削除する節点zが葉のときは単に削除
2. zの子が1つの場合： zを削除して子をその位置に移動
3. zの子が2つの場合：

SUCCESSORの最初の場合

   I. zの次節点yを見つける（yはzの右の子孫の最小要素）
      別に前接点でもいい
   II. yを削除して、zの位置にyを入れる

上記2

   - yの子は高々１個(右の子)なのでyの削除は容易
   - 子孫との大小関係が保たれていることに注意

$\mathrm{key}(z) \leqq \mathrm{key}(y)$なので、$z$の左の子孫と$y$との大小関係はOK
zの右の子孫のなかでyは最小なので、こちらもOK

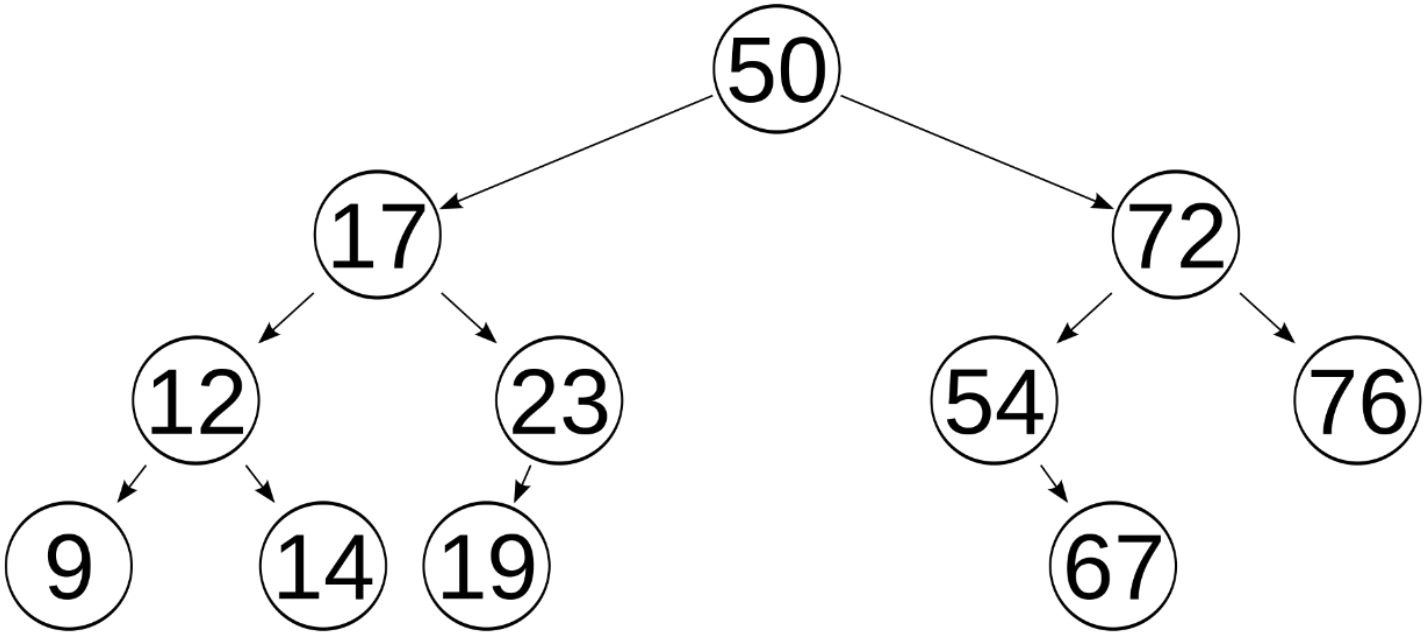
平衡木

平衡木：
バランスのとれた二分探索木

- 二分探索木をもちいた探索のコストは、根から所望の節点までの道のりの長さ
  （探索対象が見つからない場合には葉までの長さ）
- 二分探索木が完全二分木に近い場合には$O(\log n)$
  しかし、バランスが悪いとコストがかかる場合がある
- 平衡木：木の高さ（根から葉までの道のりの長さ）が常に$O(\log n)$であるような探索木
  - AVL木、赤黒木、スプレー木、B木、…

AVL木：
バランスのとれた二分探索木

- どの節点についても、右の部分木と左の部分木の高さの差が最大1であるような二分探索木



https://ja.wikipedia.org/wiki/AVL%E6%9C%A8#/media/File:AVLtreef.svg

AVL木の性能：最悪ケースで$O(\log n)$

- 2分木のなかで最も低いものは完全二分木（$\log n$）
- いっぽう、もっとも高いものが最悪ケース
  - 頂点数nをもつ二分木のなかで最も高いもの

  ⇔ 高さhの二分木のうち、もっとも頂点数が少ないもの
  - 高さhのAVL木の最小の頂点数をN_hとすると

$$N_h = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{h+3} - \left(\frac{1-\sqrt{5}}{2}\right)^{h+3}\right) - 1 \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{h+3} \quad (h\text{が大のとき})$$

$\left|\frac{1-\sqrt{5}}{2}\right| < 1$

  - つまり $h = \frac{\log n}{\log\left(\frac{1+\sqrt{5}}{2}\right)} - 3 \approx 1.44 \log n$

（最良ケースの1.44倍）

$\approx \frac{1}{1.44}$

補足：
なるべくバランスの悪いAVL木をつくる

- 高さhのAVL木の最小の頂点数をN_hとすると

$$N_h = N_{h-1} + N_{h-2} + 1$$

- $f_h = N_h + 1$とすれば $f_h = f_{h-1} + f_{h-2}$
  （フィボナッチ数列）

- フィボナッチ数列の解：

$$f_h = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^{h+3} - \left(\frac{1-\sqrt{5}}{2}\right)^{h+3}\right)$$