中間テスト

第8回 11月28日（水）1限

物理系校舎 3階 313（講義室6）

KYOTO UNIVERSITY

アルゴリズムとデータ構造⑥
～ 探索問題（２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、データの集合から所望のデータを見つけてくる
  - データは「キー」と「（データの）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る
- 2分探索木やハッシュ等によって実現可能

２分探索木

２分探索木：
データ集合から所望の要素を見つける

- ２分探索木：
  - 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：xの左側の子を根とする部分木に含まれる頂点
    - y'：xの右側の子を根とする部分木に含まれる頂点
    - $\mathrm{key}(y) \leqq \mathrm{key}(x) \leqq \mathrm{key}(y')$

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
- SEARCH(x, k)：これを「x =根」で呼ぶ; kは探すkey

  if x = NULL または k = key(x) then xを返す

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

  if $k >$ key(x) then SEARCH(right$(x), k)$ ：右にあるはず
- SEARCH(x, k)：再帰なしの方法

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

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

  end while; xを返す

２分探索木からソート済み配列を取り出す：中順での要素列挙

- ２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

　INORDER(left(x))：必ずx以下

　key(x)を出力

　INORDER(right(x))：必ずx以上

end if

- 最小（最大）の要素の発見であれば、left（right）をたどることでO(h)で発見可能

前順・後順での巡回：その他の巡回

- PREORDER(x)：前順での巡回

  key(x)を出力

  PREORDER(left(x))

  PREORDER(right(x))

- POSTORDER(x)：後順での巡回

  POSTORDER(left(x))

  POSTORDER(right(x))

  key(x)を出力

- 出力の位置に注意

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 right(x) $\neq$ NULL then MININUM(right(x))

右の子がいるなら
その右部分木の
最小要素

$y \leftarrow$ parent(x)

自分が親の左の子なら親が
次節点

while $y \neq$ NULL かつ $x =$ right(y)

$\quad x \leftarrow y$; $y \leftarrow$ parent(x)

自分が親の右の子である限
り上がっていく

end while

xを返す

二分探索木への挿入と削除：
挿入

- 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について$O(h)$

- これを繰り返して２分探索木を構成可能
  - ランダムな順で挿入すれば平均的に高さ$O(\log n)$

二分探索木への挿入と削除：削除

- 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：
   1. zの次節点yを見つける
   2. yを削除して、zの位置にyを入れる
      - yの子はたかだか１個（右の子）

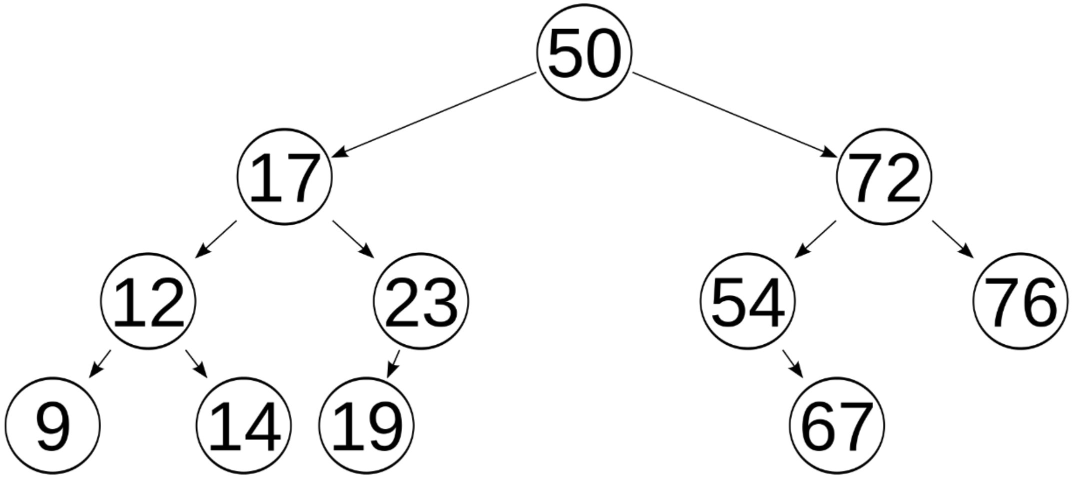
平衡木

平衡木：
バランスのとれた２分探索木

- ２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（無い場合には葉までの長さ）
- ２分探索木が完全２分木に近い場合には$O(\log n)$だが、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２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２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2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2分木のなかで最も高いもの

  ⇔ 高さhの2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

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

  - つまり $h = \dfrac{\log n}{\log\left(\frac{1+\sqrt{5}}{2}\right)} - 3 \approx 1.44 \log n$ （最良ケースの1.44倍）

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