

3. スタック, キュー, 木

- スタックとは, 後に入れたデータを先に取り出すデータ構造である. この性質を LIFO (last-in first-out) と呼ぶ. 次の操作を行うことができる.
 - push(x): スタックの一番上に x を挿入する.
 - pop: スタックの一番上の要素を返し, それをスタックから削除する.
- キュー (待ち行列) とは, 先に入れたデータを先に取り出すデータ構造である. この性質を FIFO (first-in first-out) と呼ぶ. 次の操作を行うことができる.
 - enqueue(x): キューの最後尾に x を挿入する.
 - dequeue: キューの先頭の要素を返し, それをキューから削除する.
- 特別な頂点が 1 つ定まっている木を根付き木といい, この特別な頂点を根という. 隣接する 2 頂点 P, Q に対し, P の方が根に近いとき, P を Q の親, Q を P の子という. また, 頂点 Q から根までの道の中に頂点 P があるとき, P を Q の先祖, Q を P の子孫という. 子を持たない頂点を葉という. 頂点から根までの道の長さをその頂点の深さという. 頂点の深さの最大値を根付き木の高さ^{*1}という.
- 各頂点の子の数が n 以下であるような根付き木を n 分木という.
- 2 分木においては, 子が左側か右側かを常に指定し, 左の子, 右の子という. また, 左の子, 右の子を根とする部分木をそれぞれ左部分木, 右部分木という.
- 2 分木において, 深さが最大でない頂点はすべて埋まっており, 深さが最大の頂点は左詰めになっているものを, (広義の) 完全 2 分木という.
- 2 分木 T において, どの 2 頂点の値も比較可能であり, 簡単のため, どの 2 頂点も同じ値を持たないものとする. T が 2 分探索木であるとは, 各頂点の値が, 左部分木のどの頂点の値よりも大きく, 右部分木のどの頂点の値よりも小さいことである. 次の操作を行うことができる.
 - min (max): T に含まれる値の最小値 (最大値) を求める.
 - search(x): x が T に含まれるかどうか判定する.
 - insert(x): x を T に挿入する.
 - delete(x): x を T から削除する.

最小値 (最大値) を求めるには最も左 (右) にある葉を探せばよい.

x が T に含まれるか判定するには, まず x を根と比較する. 一致すれば x は T に含まれる. x の方が小さいときは左部分木に対して, x の方が大きいときは右部分

*1 頂点の深さの最大値に 1 加えたものを高さと呼ぶこともある.

木に対して同じ操作を繰り返す。葉に到達しても一致しないならば、 x は T に含まれない。

データを挿入する際には正しい位置を同様に探し、葉を付け加えればよい。

データの削除は次のように行われる。まず、削除する頂点が葉である場合は、それを取り除けばよい。削除する頂点が1個の子を持つ場合、上下の頂点をつなげばよい。削除する頂点が2個の子を持つ場合、左部分木の頂点で値が最大のもの（または右部分木の頂点で値が最小のもの）を削除する頂点の位置に移し、空いた頂点の上下の頂点をつなげばよい。

一般に、2分探索木はデータの探索が高速に可能である（平均で $O(\log n)$ 回の比較でよい）。しかし、データの挿入・削除の順序によっては木の高さが大きくなり、効率が非常に悪くなる。

問題

3-1. 空のスタックに対し、次の操作を行ったときの過程を図示せよ。

push(5) → push(6) → push(1) → push(3) → pop →
push(9) → pop → push(8) → pop → pop

3-2. 空のキューに対し、次の操作を行ったときの過程を図示せよ。

enqueue(5) → enqueue(6) → enqueue(1) → enqueue(3) → dequeue →
enqueue(9) → dequeue → enqueue(8) → dequeue → dequeue

3-3. 頂点を7個持つ2分木で、高さが最大のものと最小のものを1個ずつ図示せよ。

3-4. 値 47, 15, 84, 45, 28, 20, 57 を持つ7個の頂点を持つ2分探索木で、完全2分木であるものを図示せよ。

3-5. 空の2分探索木に対し、次の操作を行ったときの過程を図示せよ。

insert(66) → insert(87) → insert(25) → insert(24) →
insert(39) → delete(66) → insert(47) → insert(83)

3-6. 空の2分探索木に対し、次の操作を行ったときの過程を図示せよ。

insert(58) → insert(10) → insert(41) → insert(87) →
insert(61) → insert(73) → insert(39) → delete(58)