van Emde Boas Trees

光吉　健汰

北海道大学工学部 情報エレクトロニクス学科 情報理工学コース 4 年
情報知識ネットワーク研究室

June 12, 2019

Contents

1. 自己紹介
2. van Emde Boas tree とは
3. binary-tree

Contents

1. 自己紹介
2. van Emde Boas tree とは
3. binary-tree

こんにちは

Contents

1 自己紹介

2 van Emde Boas tree とは

3 binary-tree

van Emde Boas tree とは (1/2)

van Emde Boas tree (以下 vEB 木) は動的集合を扱うデータ構造

動的集合

動的集合とは, 集合に対して後述の各種操作が行えるようなデータ構造

- 今回扱う要素は非負整数とする
- vEB 木が保持しうる要素の集合を全体集合 U とし, その大きさを u とする
- vEB 木が現在保持している集合を V とし, その大きさを n とする

van Emde Boas tree とは (2/2)

van Emde Boas tree (以下 vEB 木) は動的集合を扱うデータ構造

操作

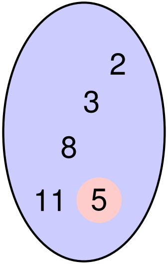
$\textsc{Member}(V, x)$	V に x が存在するかを返す
$\textsc{Min}(V)$	V の要素の最小値を返す
$\textsc{Max}(V)$	V の要素の最大値を返す
$\textsc{Successor}(V, x)$	V の x より大きい最小の要素を返す
$\textsc{Predecessor}(V, x)$	V の x より小さい最大の要素を返す
$\textsc{Insert}(V, x)$	V に x を挿入する
$\textsc{Delete}(V, x)$	V から x を削除する

これらの操作が最悪時間計算量 $O(\log \log u)$ で実行可能

操作 MEMBER(V, x)

- MEMBER(V, x) は, x が集合に存在するかを真偽値で返す.

Figure: $V = \{2, 3, 5, 8, 11\}$



- MEMBER($V, 5$) $=$ TRUE
- MEMBER($V, 6$) $=$ FALSE

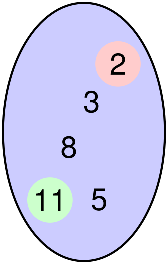
各関数の引数

関数に渡す引数は要素として取りうる値のみとする

操作 Min(V), Max(V)

- Min(V) は, 集合の要素の最小値を返す.
- Max(V) は, 集合の要素の最大値を返す.

Figure: $V = \{2, 3, 5, 8, 11\}$

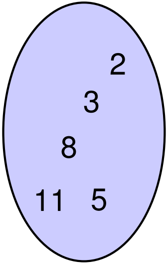


- Min(V) $= 2$
- Max(V) $= 11$

操作 $\textsc{Successor}(V, x)$, $\textsc{Predecessor}(V, x)$

- $\textsc{Successor}(V, x)$ は, 集合の要素から x より大きい最小の値を返す.
- $\textsc{Predecessor}(V, x)$ は, 集合の要素から x より小さい最大の値を返す.

Figure: $V = \{2, 3, 5, 8, 11\}$



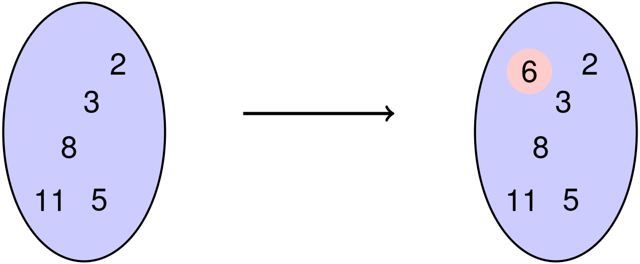
- $\textsc{Successor}(V, 2) = 3$
- $\textsc{Predecessor}(V, 7) = 5$

操作 Insert(V, x)

- Insert(V, x) は, 集合に x を挿入する.

Figure: $V = \{2, 3, 5, 8, 11\}$

Figure: $V = \{2, 3, 5, \mathbf{6}, 8, 11\}$

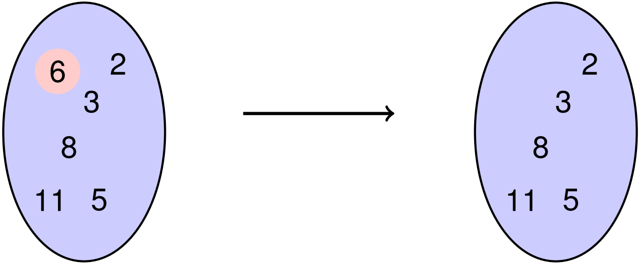


操作 Delete(V, x)

- Delete(V, x) は, 集合 V から x を削除する.

Figure: $V = \{2, 3, 5, \mathbf{6}, 8, 11\}$

Figure: $V = \{2, 3, 5, 8, 11\}$



Contents

1 自己紹介

2 van Emde Boas tree とは

3 binary-tree

直接アドレス法 (1/2)

空間計算量 $O(u)$ で動的集合を保持する手段として, 直接アドレス法を考える.

直接アドレス法

要素の値を配列の添字として利用し, データを保持するテクニック

今回考えているデータ構造では付属データは持たないので,
各配列の要素には要素を保持しているかを bit で格納する.

0	0	1	1	0	1	0	0	1	0	0	1	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure: $V = \{2, 3, 5, 8, 11\}$

直接アドレス法 (2/2)

空間計算量 $O(u)$ で動的集合を保持する手段として, 直接アドレス法を考える.

- Member(V, x), Insert(V, x), Delete(V, x) の処理は, 配列のランダムアクセスが定数時間であるので, 時間計算量 $O(1)$.
- Successor(V, x) の処理は, x の次の値から bit が立っている要素まで
  最悪 $\Theta(u)$ 回探索する必要があるので, 時間計算量 $\Theta(u)$.
  - Predecessor(V, x), Min(V), Max(V) も同様の処理を行うため, 時間計算量 $\Theta(u)$.

0	0	1	1	0	1	0	0	1	0	0	1	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Figure: $V = \{2, 3, 5, 8, 11\}$

二分木構造 (1/2)

$Successor(V, x)$ で必要な探索回数を小さくするために,
配列の各要素を葉とした二分木構造を考える.

二分木

二分木とは, 葉ではない頂点の子が常に 2 個であるような木

木 閉路を持たない, 連結なグラフ

根 木の頂点のうちの 1 つに定義する

子 ある頂点について, 隣接する頂点のうち根から遠いもの

親 ある頂点について, 隣接する頂点のうち根に近いもの

葉 木の頂点のうち, 子を持たないもの

二分木構造 (2/2)

$Successor(V, x)$ で必要な探索回数を小さくするために，
配列の各要素を葉とした二分木構造を考える．

葉ではない各頂点には，子の値の論理和を格納する．
そのため，ある頂点の bit が立っている場合，
その頂点の下にある葉の少なくとも 1 頂点は bit が立っている．

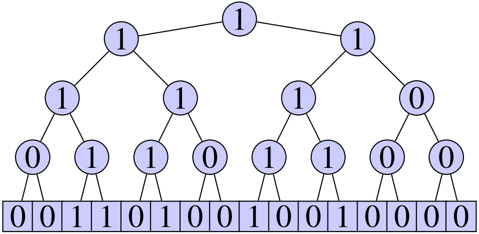


Figure: $V = \{2, 3, 5, 8, 11\}$

二分木構造 操作 Min(V), Max(V)

- Min(V) は, 根から左の子が 1 であれば左の子へ,
  そうでなければ右の子へ降りる操作を再帰的に行う.
  - Max(V) も同様に処理する.

Algorithm 1 Min(V)

Require:

$root$ 二分木の根
$v.left$ 頂点 v の左の子 (右は $v.right$)
$v.parent$ 頂点 v の親
$v.value$ 頂点 v の値

function Min(V)
  $v \leftarrow root$
  if $v.value = 0$ **then**
    return nil
  while $v.left \neq$ nil **do**
    if $v.left.value = 1$ **then**
      $v \leftarrow v.left$
    else
      $v \leftarrow v.right$
end function

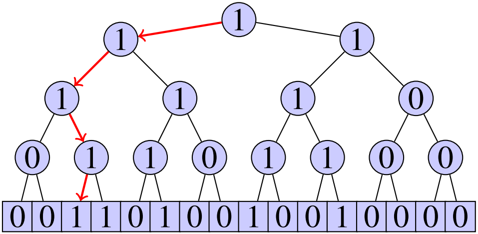


Figure: Min(V), $V = \{2, 3, 5, 8, 11\}$

二分木構造 操作 Successor(V, x), Predecessor(V, x) (1/2)

- Successor(V, x) は, x の親から, 自身が直前の頂点が左の子かつ右の子の bit が立っている頂点になるまで親を辿り, 右の子の最小値を返す.

Algorithm 2 Successor(V, x)

$prev \leftarrow x$ の頂点
$v \leftarrow prev.parent$
while $v \neq root$ **do**
  if $v.value = 1$ *and*
$v.left = prev$ *and* $v.right.value = 1$ **then**
    return
Min($v.right$ を根とした二分木)
  else
    $prev = v$
    $v = v.parent$
return NIL

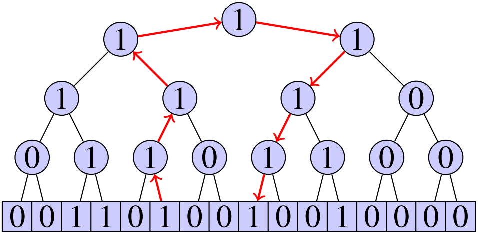


Figure:
Successor$(V, 5)$, $V = \{2, 3, 5, 8, 11\}$

二分木構造 操作 SUCCESSOR(V, x), PREDECESSOR(V, x) (2/2)

- SUCCESSOR(V, x) は, x の親から, 自身が直前の頂点が左の子かつ右の子の bit が立っている頂点になるまで親を辿り, 右の子の最小値を返す.

Algorithm 3 SUCCESSOR(V, x)

$prev \leftarrow x$ の頂点
$v \leftarrow prev.parent$
while $v \neq root$ **do**
  if $v.value = 1$ *and* $v.left = prev$ *and* $v.right.value = 1$ **then**
    return MIN($v.right$ を根とした二分木)
  else
    $prev = v$
    $v = v.parent$
return NIL

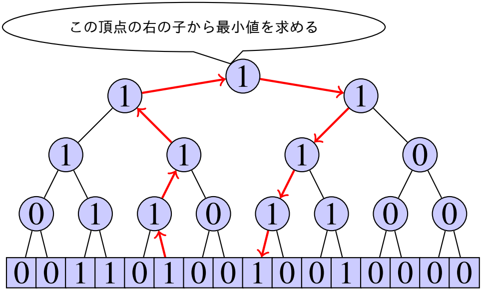


Figure: SUCCESSOR$(V, 5)$, $V = \{2, 3, 5, 8, 11\}$