Introduction to Algorithms

Chapter 20: van Emde Boas 木

@ohken

2020/03/05

データ構造の主要な操作

- Search (またはMember)
  - 要素xのポインタ(含まない場合NIL)を返す。
    (Memberは含むかどうかの真理値を返す)
- Successor/Predecessor
  - 要素xより大きな(resp. 小さな)要素のうち最小(resp. 最大)の要素を返す
- Minimum/Maximum
- Insert
- Delete

導入

- 今までのデータ構造ではInsert, Delete, Minimumのいずれかは(ならし/最悪で)$\Omega(\log n)$かかる
  - 二分ヒープでは、全部そう
  - 赤黒木も、全部そう
  - Fibonacci ヒープではDeleteが$\Omega(\log n)$(他は$O(1)$)
- 上の操作の計算量をすべて$o(\log n)$ にするのは(無条件では)不可能
  - できると仮定すると、n個の整数を順にInsertして、n回Extract-Min(= Minimum + Delete)すれば$o(n \log n)$でソートが可能になる.
    比較ソートの計算量の下限に矛盾.

$O(n)$でのソート

例えば「n個の数が$\{1, 2, 3, \cdots, n\}$の元」と知っていれば$O(n)$でソートできる(8章).

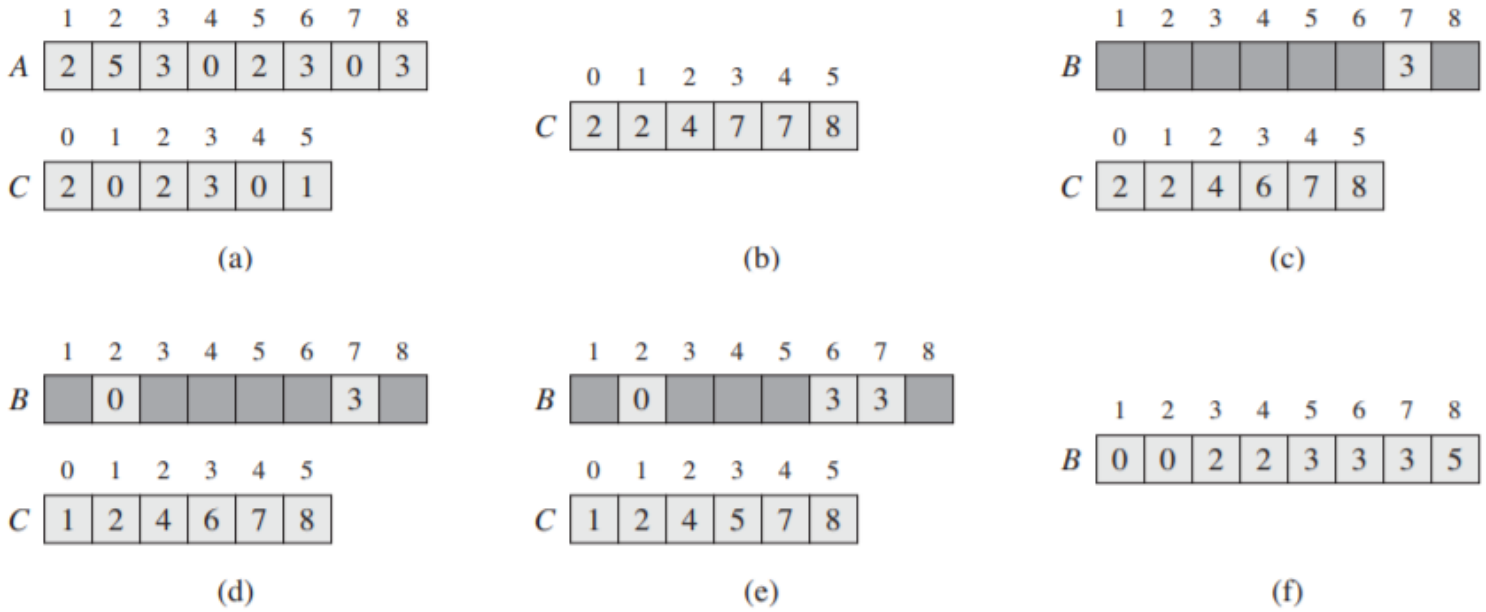


Figure 8.2 The operation of COUNTING-SORT on an input array $A[1..8]$, where each element of A is a nonnegative integer no larger than $k = 5$. **(a)** The array A and the auxiliary array C after line 5. **(b)** The array C after line 8. **(c)–(e)** The output array B and the auxiliary array C after one, two, and three iterations of the loop in lines 10–12, respectively. Only the lightly shaded elements of array B have been filled in. **(f)** The final sorted output array B.

問. 適当な仮定のもとで全ての主要な操作を$o(\log n)$で行えるか？

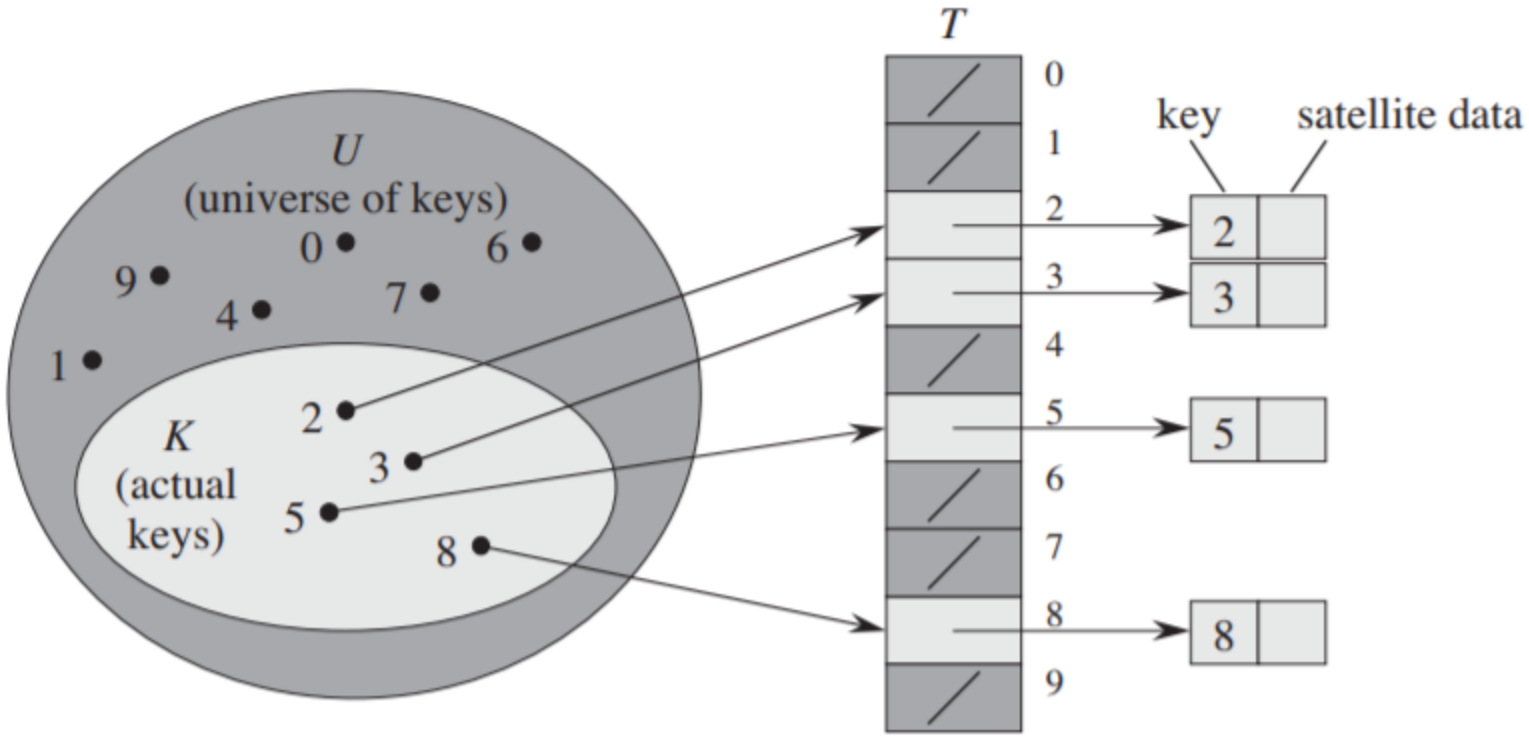
答. できる. そう、van Emde Boas木ならね.

- $O(\log \log n)$で行うことができる
  - 正確には、n個の整数が0以上u未満で重複なしという仮定で$O(\log \log u)$

8.1 準備的なアプローチ

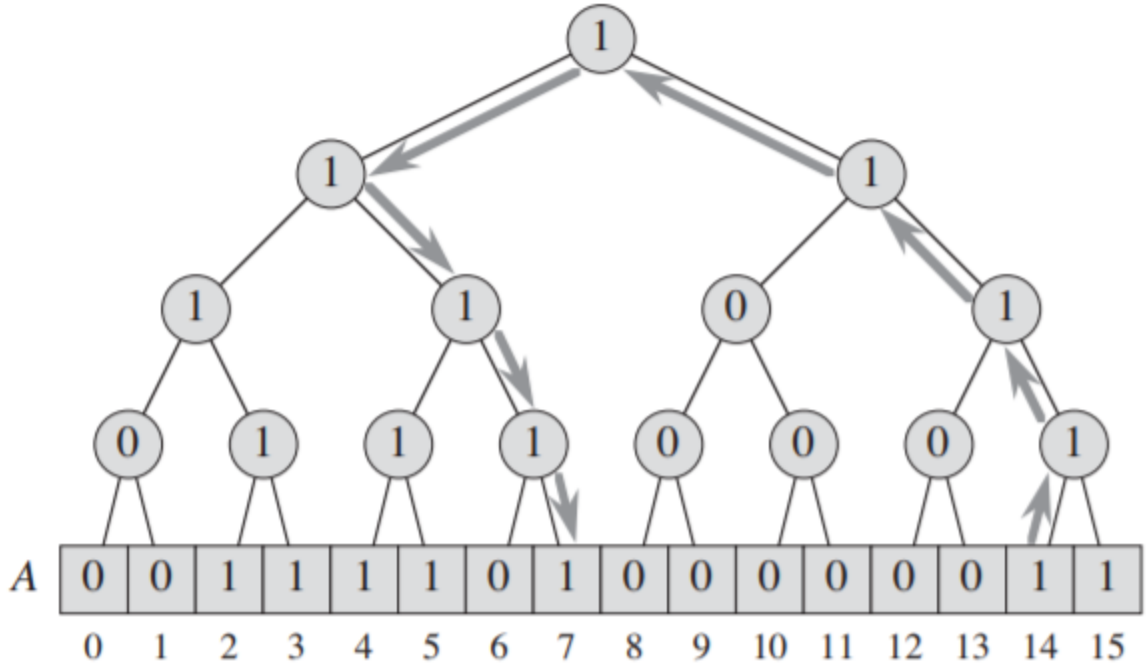
1. (安直に)重複なしの配列をハッシュで扱う

- U: とりえる値の集合(= $\{0, 1, 2, \cdots, u-1\}$)
- 二値配列$A[0..u-1]$で集合を保持 → Minimum, Successor等が$\Theta(u)$



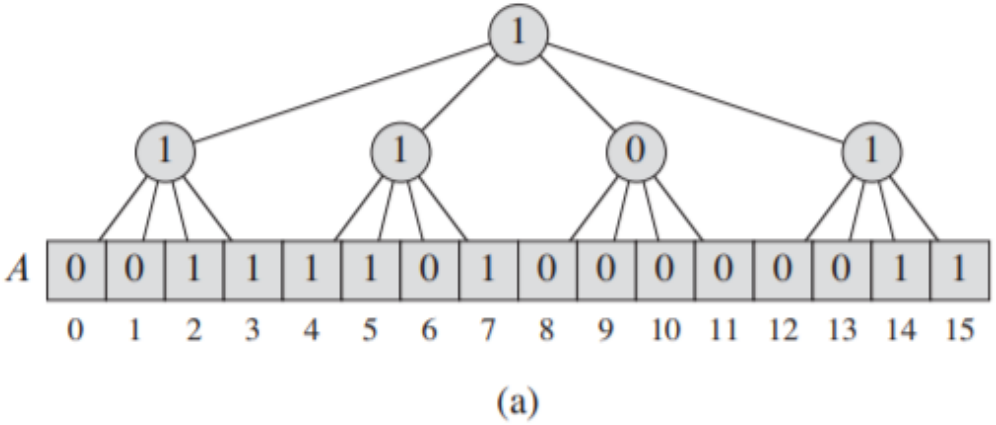
2. 二分木をつけてみる

- 親ノード$=\max${子ノード}
- Memberが$O(1)$, それ以外は$O(\log u)$

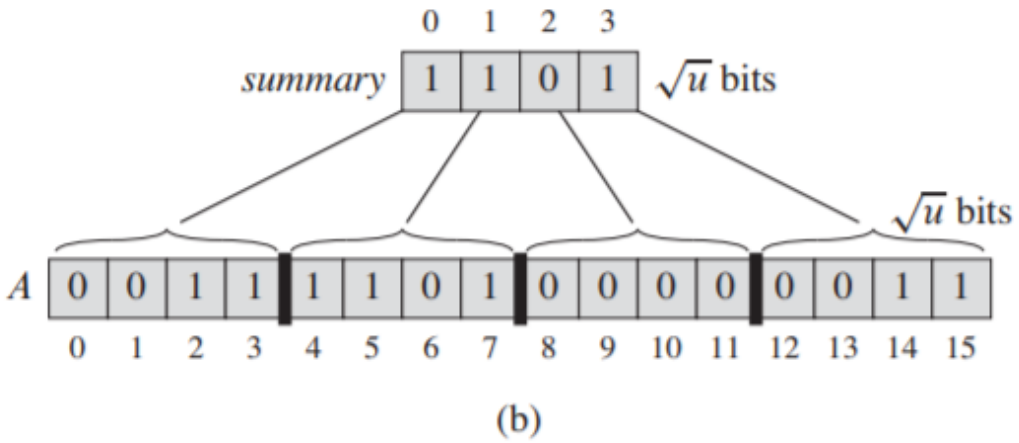


3. $\sqrt{u}$分木にしてみる = 木の高さを 2 にする

- 簡単のため$u = 2^{2k}$の形
- 二値配列$summary[0..\sqrt{u}-1]$
  - i成分は$\sqrt{u}$個の部分二値配列(**クラスタ**)$A[i\sqrt{u}..(i+1)\sqrt{u}-1]$の$\max$
- Member, Insertが$O(1)$, それ以外は$O(\sqrt{u})$
- $u^{1/k}$分木にしても同様($\sqrt{u}$が$u^{1/k}$になるだけ)



(a)



(b)

観察

- 3のアプローチではクラスタ全体を調べる部分がボトルネックになる
  - 先っぽのクラスタの要素数が少ない方がよさそう
- しかし二分木にすると高さ$O(\log u)$がボトルネックに...

→ 先細りの木構造にして高さとクラスタの大きさを同時に小さくしよう！

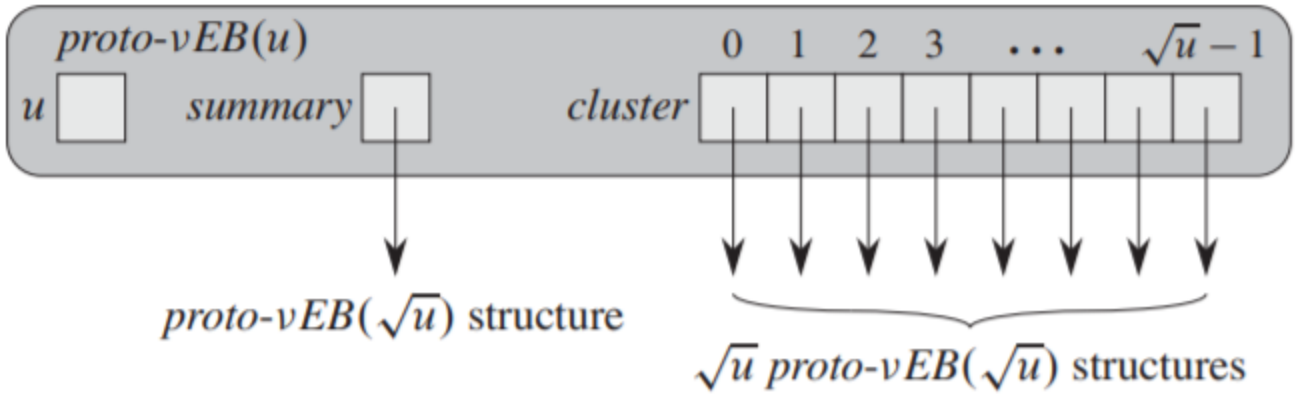
20.2 再帰構造

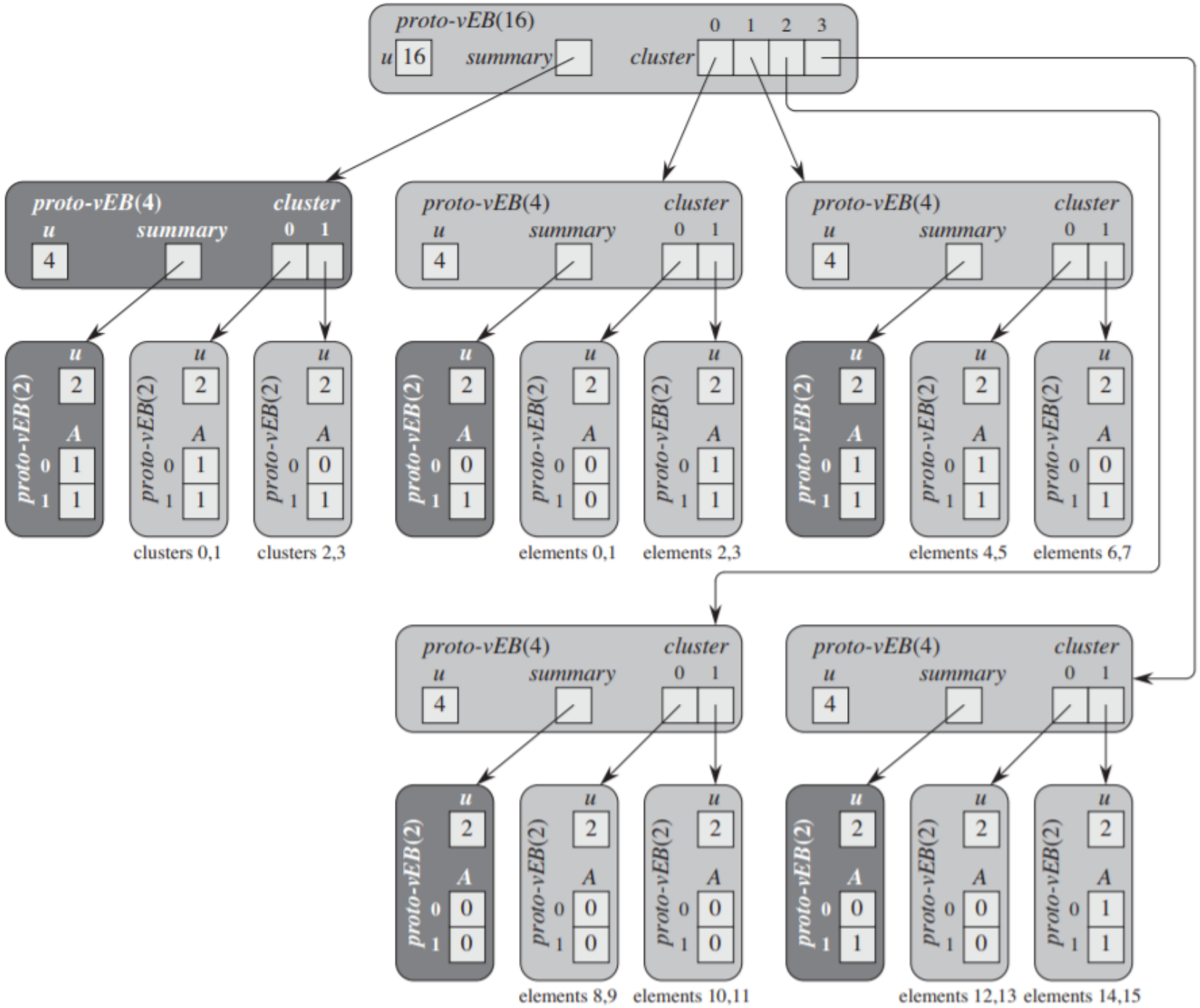
- 多分木を再帰的なデータ構造として構成したい(**前回間違っていたので訂正**)
  - $\sqrt{u} = u^{1/2}$個の子ノードを持つ構造を作る.
  - 子ノードは同様に$u^{1/4}$個の子ノードを持ち、
  - さらにそれらはそれぞれ$u^{1/8}$個の子ノードを持ち、...
  - 子ノードの数が2になるまで続ける.
- 簡単のため$u = 2^{2^k}$ の形を仮定
  - $\sqrt{u} = 2^{2^{k-1}}$、続けて平方根をとれる

van Emde Boas木のプロトタイプ$proto\text{-}vEB(u)$を構成しよう

proto-vEBの定義

- $proto\text{-}vEB(2) = A[0..1]$: 長さ2の二値配列
- $u = 2^{2^k}, k \geq 1$に対して$proto\text{-}vEB(u)$は次からなる:
  - $cluster[0..\sqrt{u}-1]$: 長さ$\sqrt{u}$の$proto\text{-}vEB(\sqrt{u})$のポインタの配列
  - $summary$: $proto\text{-}vEB(\sqrt{u})$のポインタ




proto-vEB(16)
u 16
summary
cluster 0 1 2 3
proto-vEB(4)
u 4
summary
cluster 0 1
proto-vEB(2)
u 2
A 0 1 1
clusters 0,1
clusters 2,3
elements 0,1
elements 2,3
elements 4,5
elements 6,7
elements 8,9
elements 10,11
elements 12,13
elements 14,15

$O(\log \log n)$のトリック

以降でどのようにして計算量$O(\log \log u)$が出てくるかを先に示しておく。

- イメージ的には、木の高さが$\log \log u = k$。クラスタ数も深さについて指数的に減るため、各操作はクラスタ内をほぼ定数回探索しながら再帰して実現できる

この計算量を式に落とすとたいてい次の形になる:

$$T(u) = T(\sqrt{u}) + O(1)$$

$S(k) := T(2^{2^k})$と置きなおすと

$$T(u) = S(k) = S(k-1) + O(1) = \cdots = O(k) = O(\log \log u)$$

となる.

関数の準備

- $proto\text{-}vEB(u)$は次の関数を備えている:
  - $\mathrm{high}(x) = \lfloor x/\sqrt{u} \rfloor$
  - $\mathrm{low}(x) = x \mod \sqrt{u}$
  - $\mathrm{index}(h, l) = h\sqrt{u} + l$

整数xを二進表現したとき

- $\mathrm{high}(x)$は上半分のbit列
- $\mathrm{low}(x)$は下半分のbit列

を表す。特に$x = \mathrm{index}(\mathrm{high}(x), \mathrm{low}(x))$が成り立つ。

$proto\text{-}vEB$ での操作

Member(V, x)

```
if V.u == 2
    return V.A[x]
else
    return Member(V.cluster[high(x)], low(x))
```

注意：Memberにはsummaryは不要.

計算量は

$$T(u) = T(\sqrt{u}) + O(1) = O(\log \log u)$$

Minimum(V)

```
if V.u == 2
    if V.A[0] == 1
        return 0
    else if V.A[1] == 1
        return 1
    else
        return NIL
else
    mincluster = Minimum(V.summary)
    if mincluster = NIL          // 全てのビットが落ちていたらNIL
        return NIL
    else                         // どこか立っていたらそこでのMinimumをとる
        offset = Minimum(V.cluster[mincluster])
        return index(mincluster, offset)
```

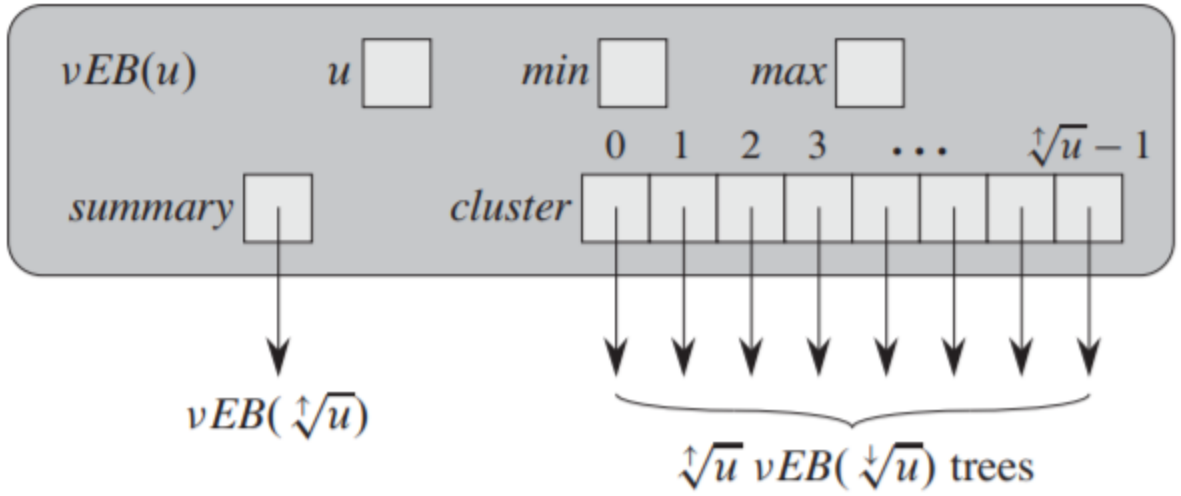
最悪計算量は

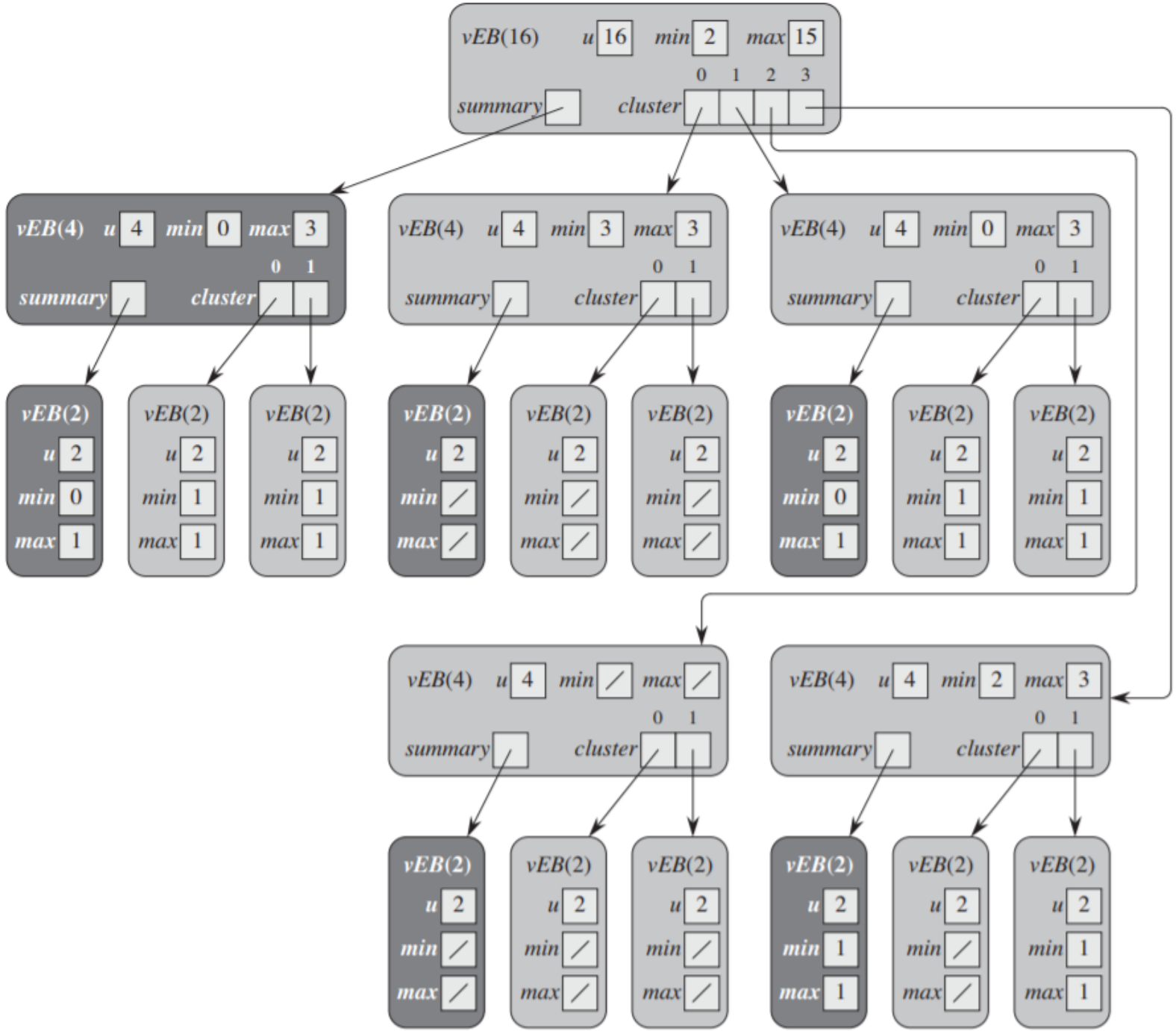
$$T(u) = 2T(\sqrt{u}) + O(1)$$

$$T(2^{2^k}) = 2T(2^{2^{k-1}}) + O(1) = O(2^k) = O(\log u)$$

Van Emde Boas 木の定義

- $u = 2^k$の形でつかえるようにする
  - $\sqrt[\uparrow]{u} = 2^{\lceil k/2 \rceil}$
  - $\sqrt[\downarrow]{u} = 2^{\lfloor k/2 \rfloor}$
- 最小値と最大値も保持しておく




vEB(16) u 16 min 2 max 15
0 1 2 3
summary cluster
vEB(4) u 4 min 0 max 3
0 1
summary cluster
vEB(4) u 4 min 3 max 3
0 1
summary cluster
vEB(4) u 4 min 0 max 3
0 1
summary cluster
vEB(2) u 2 min 0 max 1
vEB(2) u 2 min 1 max 1
vEB(2) u 2 min 1 max 1
vEB(2) u 2 min / max /
vEB(2) u 2 min / max /
vEB(2) u 2 min / max /
vEB(2) u 2 min 0 max 1
vEB(2) u 2 min 1 max 1
vEB(2) u 2 min 1 max 1
vEB(4) u 4 min / max /
0 1
summary cluster
vEB(4) u 4 min 2 max 3
0 1
summary cluster
vEB(2) u 2 min / max /
vEB(2) u 2 min / max /
vEB(2) u 2 min / max /
vEB(2) u 2 min 1 max 1
vEB(2) u 2 min / max /
vEB(2) u 2 min 1 max 1