van Emde Boas Trees

rsy3244

July 8, 2019

Contents

Contents

前回の復習

- vEB 木を定義した.
- 二分木構造で各種操作の時間計算量は下表となった.
- 平方分割木によって二分木構造の木の高さを小さくしたが, 一部操作の最悪時間計算量が $O(\sqrt{u})$ と悪化
  - *summary*, *cluster* の線形探索がボトルネック

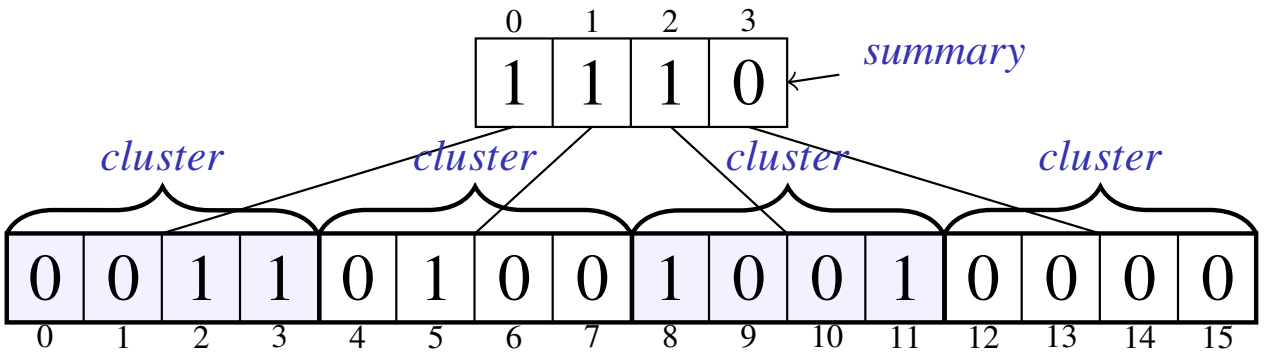
	二分木	平方分割木
MIN(V)	$\Theta(\log u)$	$O(\sqrt{u})$
MAX(V)	$\Theta(\log u)$	$O(\sqrt{u})$
MEMBER(V, x)	$O(1)$	$O(1)$
SUCCESSOR(V, x)	$O(\log u)$	$O(\sqrt{u})$
PREDECESSOR(V, x)	$O(\log u)$	$O(\sqrt{u})$
INSERT(V, x)	$O(\log u)$	$O(1)$
DELETE(V, x)	$O(\log u)$	$O(\sqrt{u})$

平方分割木 構造

平方分割木

平方分割木とは, 葉を $\sqrt{u}$ 個持ち, 高さが 2 の,
以下のような特徴を持つデータ構造である.

- 要素の全体集合の大きさ u を $u = 2^{2k}$(k は非負整数) とする.
- 配列を $\sqrt{u}$ 個に分割し, それぞれを *cluster* とする.
- *cluster* は大きさが $\sqrt{u}$ の配列であり, 対応する葉の値を持つ.
- 根は大きさ $\sqrt{u}$ の配列を持ち (これを *summary* とする) ,
  それぞれの要素は各 *cluster* の要素の論理和を保持する.

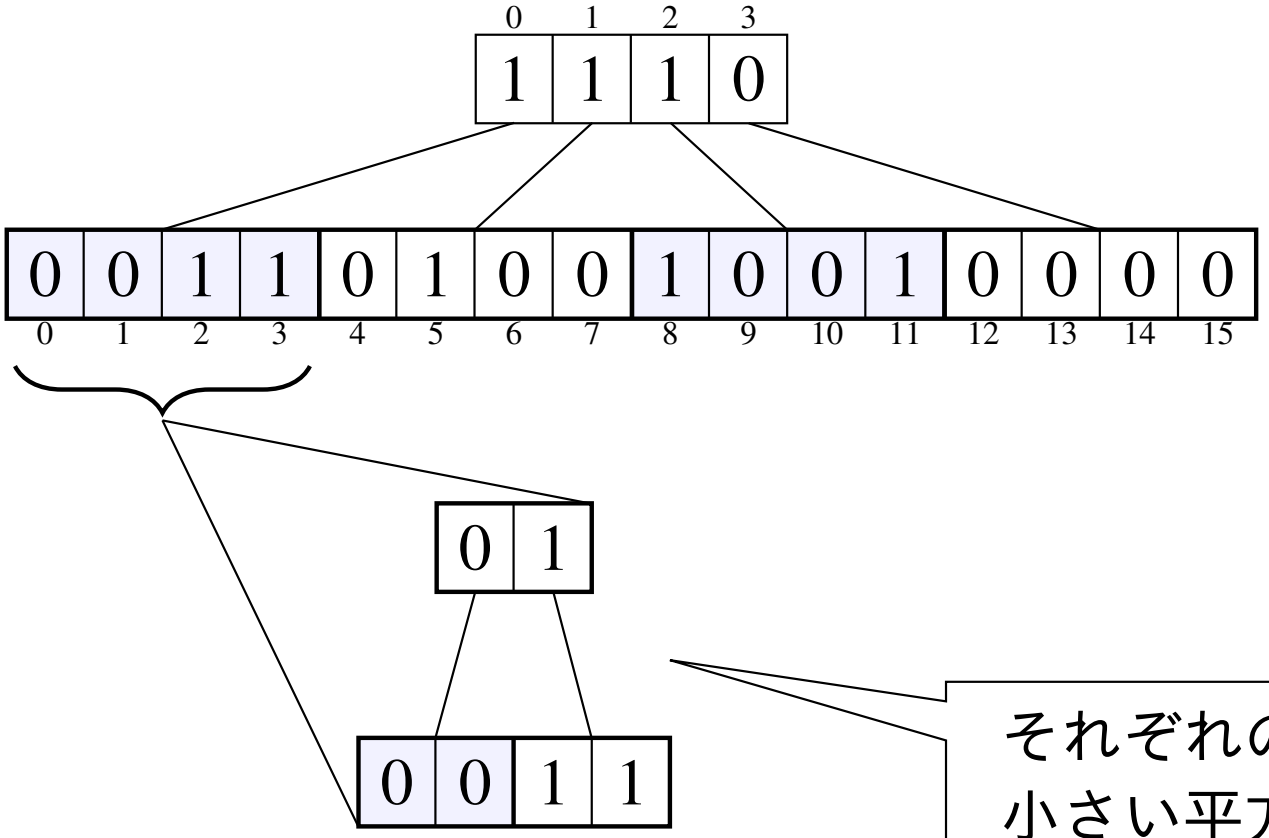


平方分割木 改善

平方分割木では, *sumary*, *cluster* が大きさ $\sqrt{u}$ の配列を持つ.

平方分割木 改善

平方分割木では, *sumary*, *cluster* が大きさ $\sqrt{u}$ の配列を持つ.
→ これらの配列を再帰的に管理できるデータ構造を考える.



それぞれの *cluster* を管理する
小さい平方分割木を考える

Contents

proto van Emde Boas structures

動的集合 v を保持する proto van Emde Boas structures を考える.

proto van Emde Boas structures

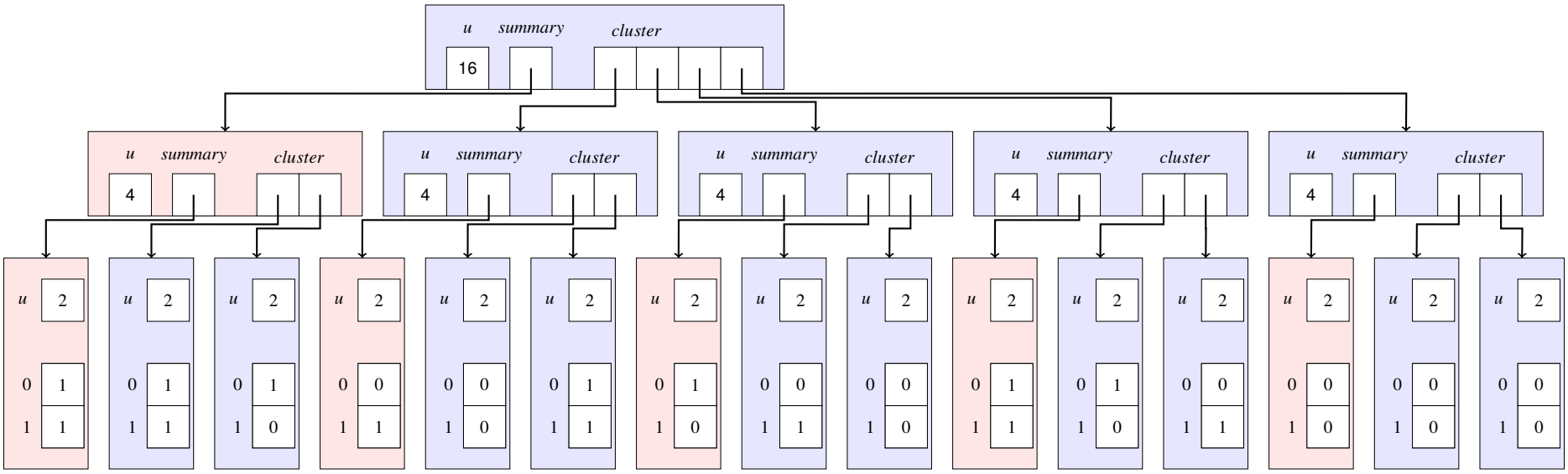
proto van Emde Boas structures (pvEB 構造) とは,
u の値と $summary$, $cluster$ に対応する
pvEB 構造へのポインタを持つデータ構造

- ポインタの先の pvEB 構造は $\sqrt{u}$ 個の要素を保持
- $u = 2$ の場合は, ポインタを持たず, $\{0, 1\}$ の要素を保持
- pvEB 構造間で対応する要素の値が異なるので,
  以下の式を用いてやり取りを行う.

$$\begin{aligned} \mathrm{high}(x) &= \lfloor \tfrac{x}{\sqrt{u}} \rfloor \\ \mathrm{low}(x) &= x \bmod \sqrt{u} \\ \mathrm{index}(x, y) &= x\sqrt{u} + y \end{aligned}$$

proto van Emde Boas structures

平方分割木の *summary*, *cluster* の配列を
$u = 4$ の pvEB 構造に置き換える

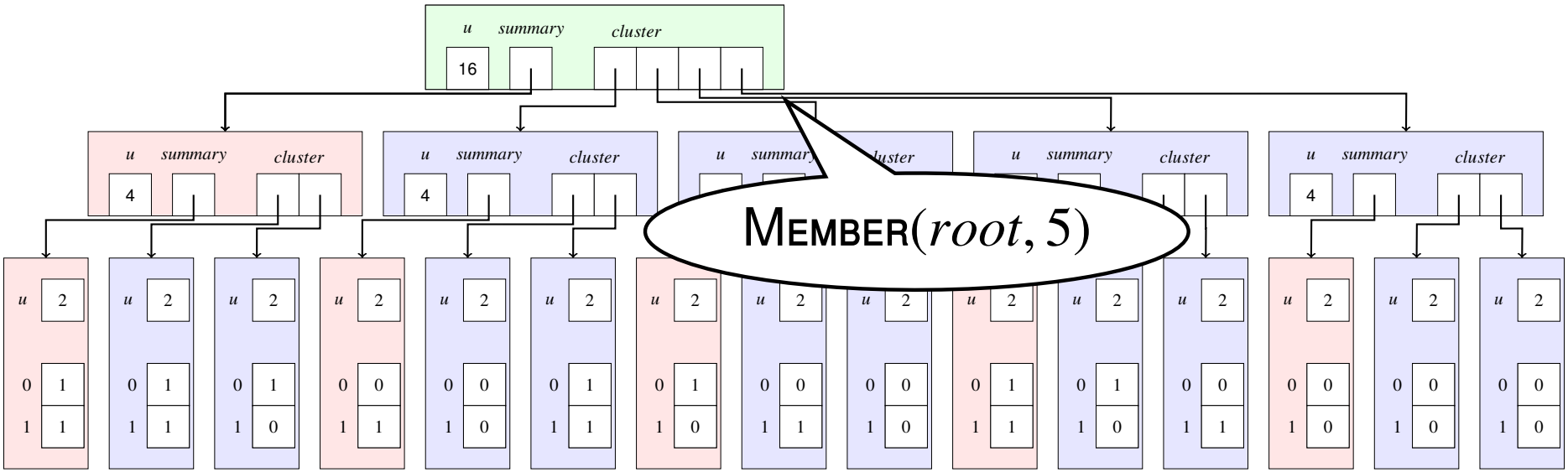


proto van Emde Boas structures

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

pvEB 構造 操作: MEMBER(V, x)

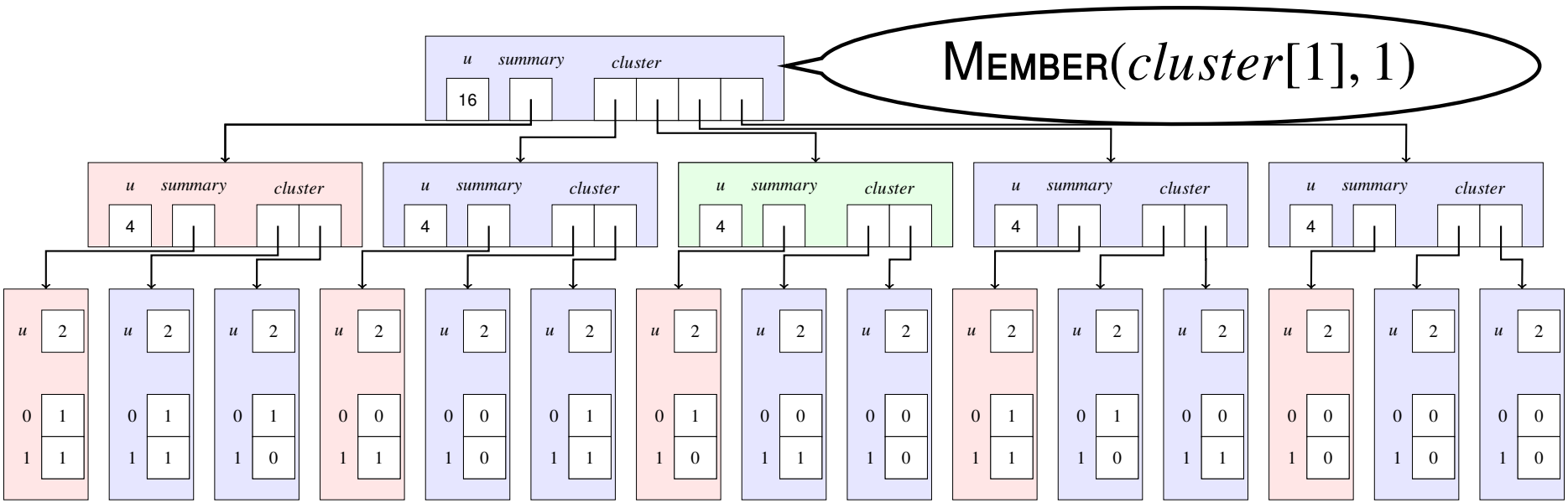
summary から対応する *cluster* が空か確認し,
空でないなら対応する *cluster* で MEMBER(V, x) を再帰的に呼び出す.



MEMBER$(V, 5), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: MEMBER(V, x)

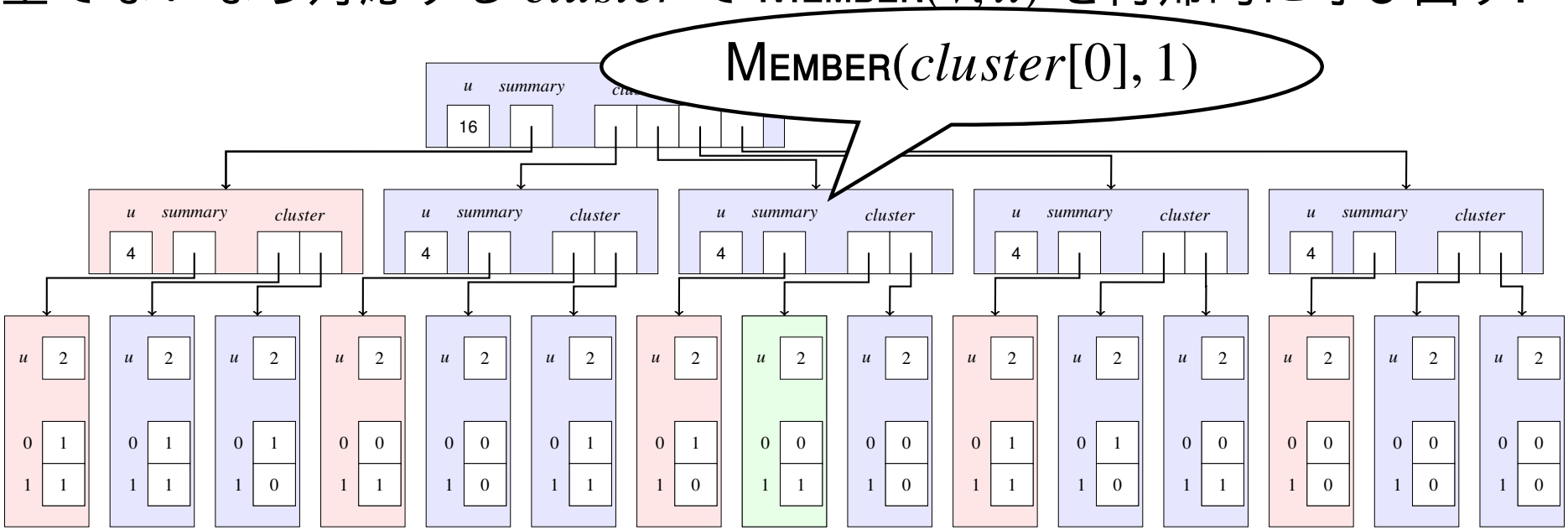
summary から対応する *cluster* が空か確認し,
空でないなら対応する *cluster* で MEMBER(V, x) を再帰的に呼び出す.



MEMBER$(V, 5), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: MEMBER(V, x)

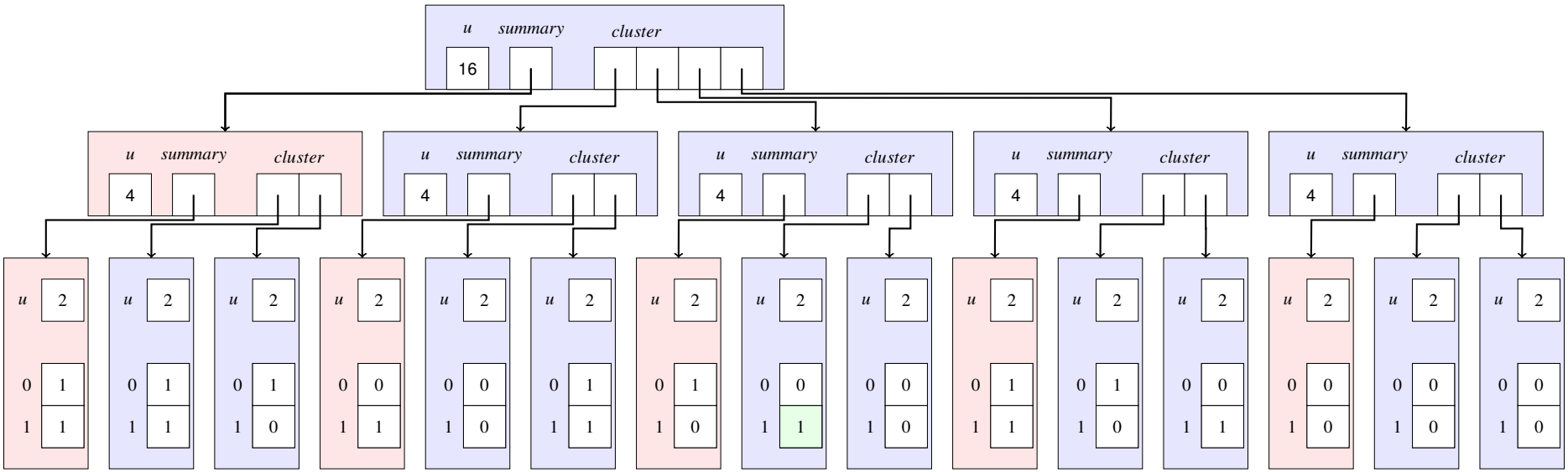
summary から対応する *cluster* が空か確認し,
空でないなら対応する *cluster* で MEMBER(V, x) を再帰的に呼び出す.



MEMBER$(V, 5), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: MEMBER(V, x)

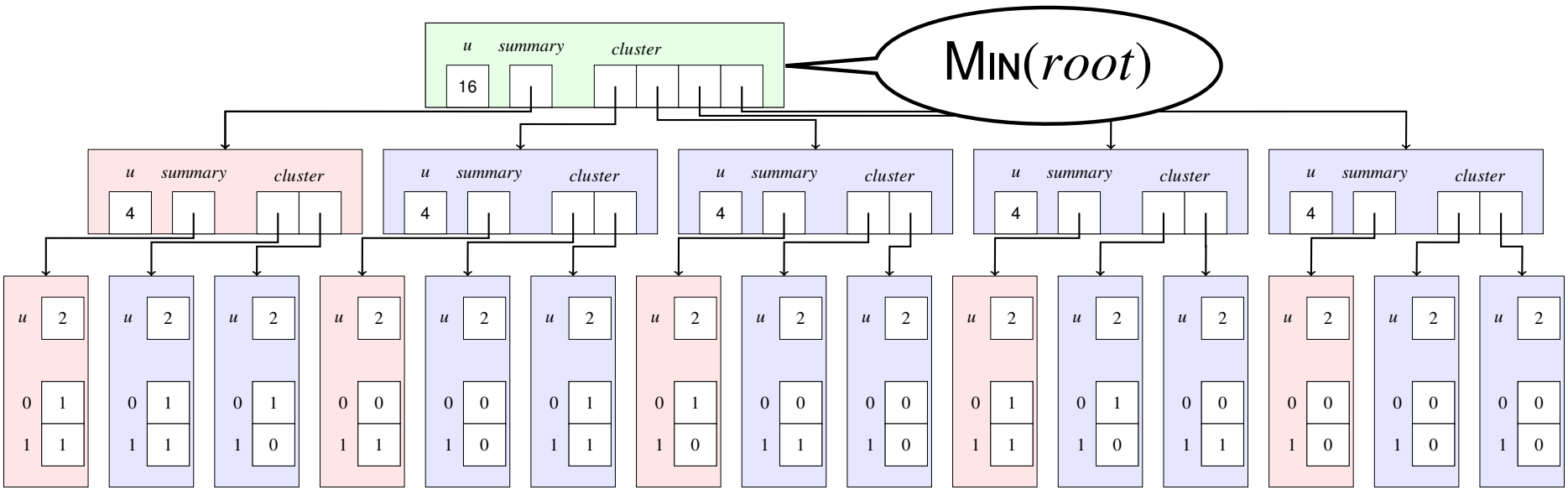
summary から対応する *cluster* が空か確認し,
空でないなら対応する *cluster* で MEMBER(V, x) を再帰的に呼び出す.



MEMBER$(V, 5), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: Min(V)

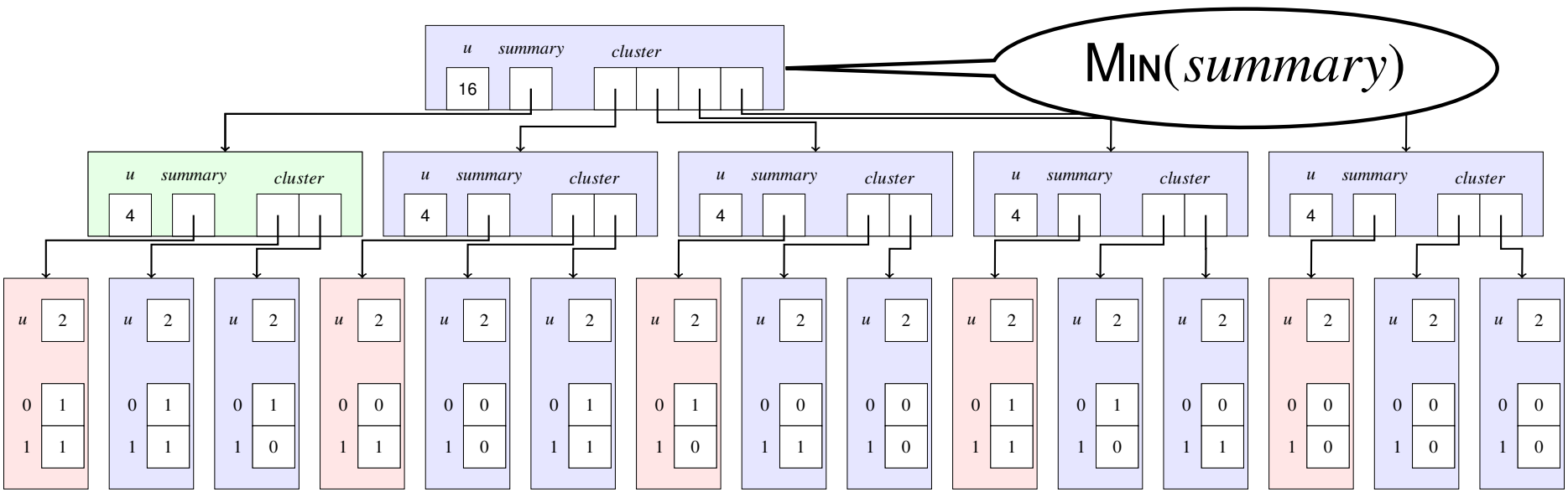
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

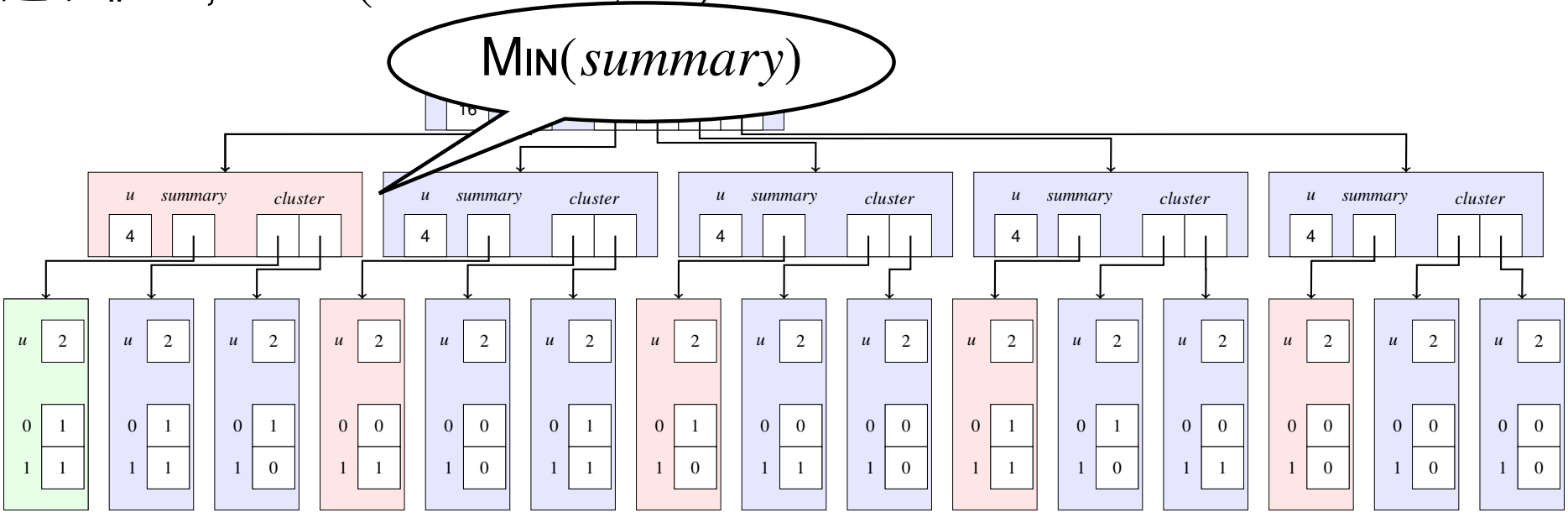
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

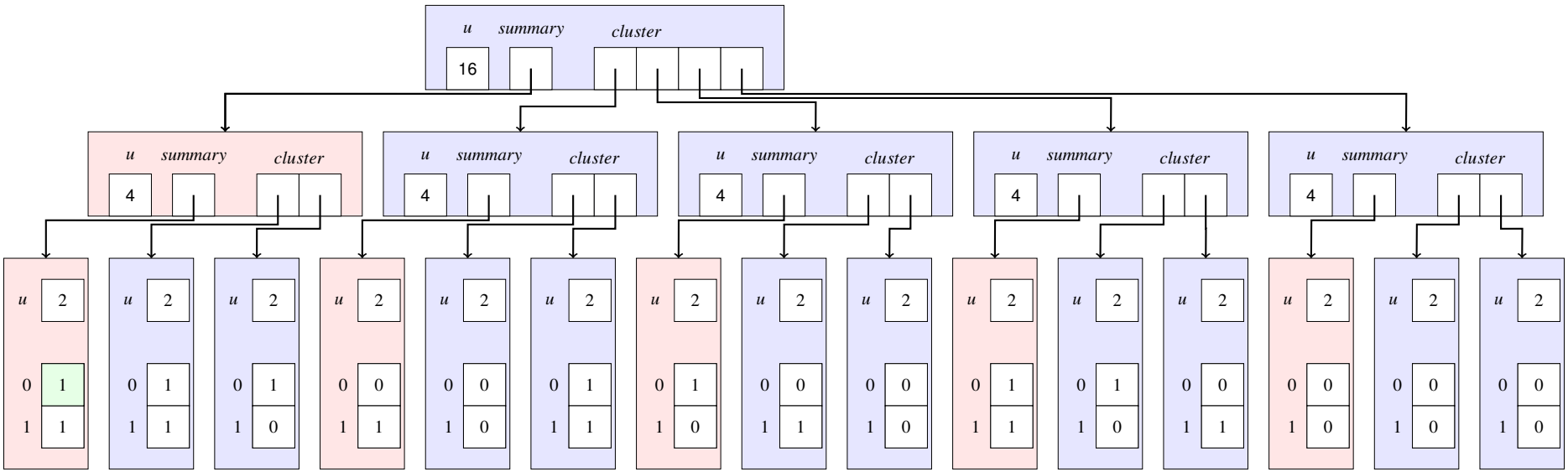
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

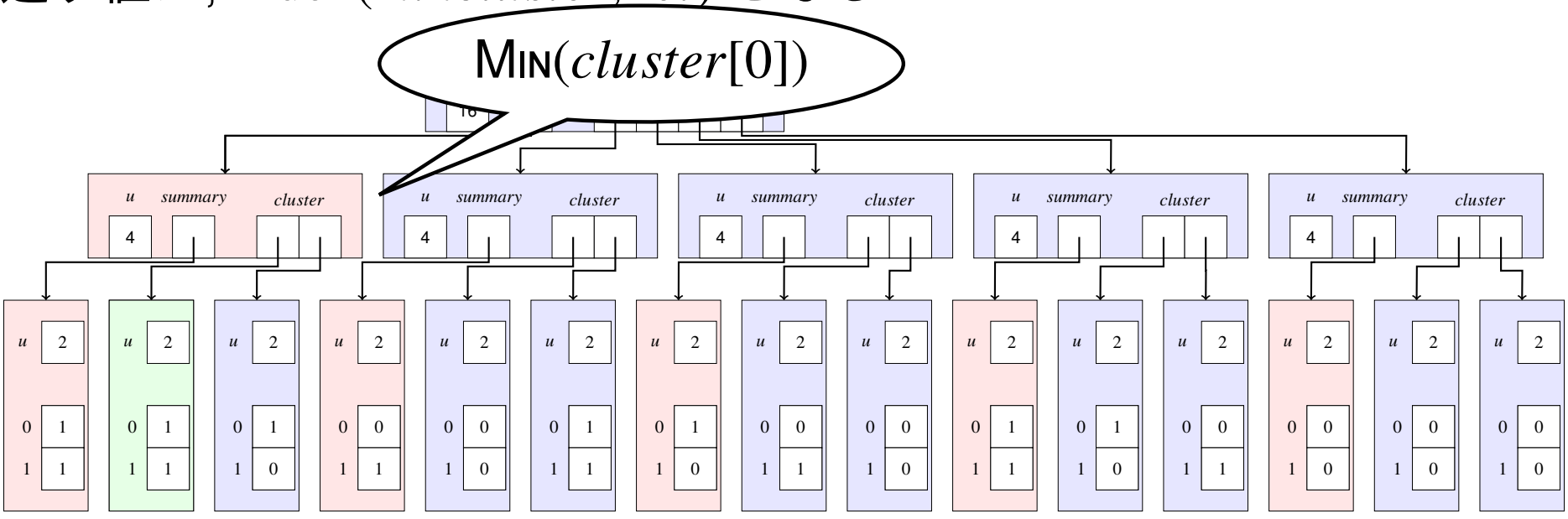
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

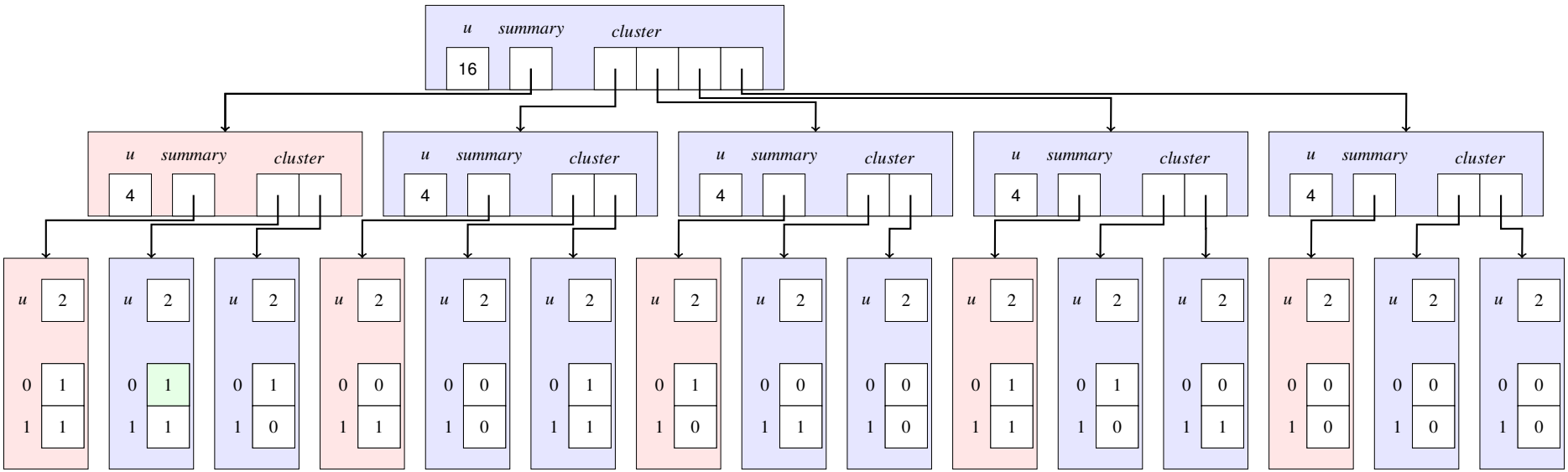
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

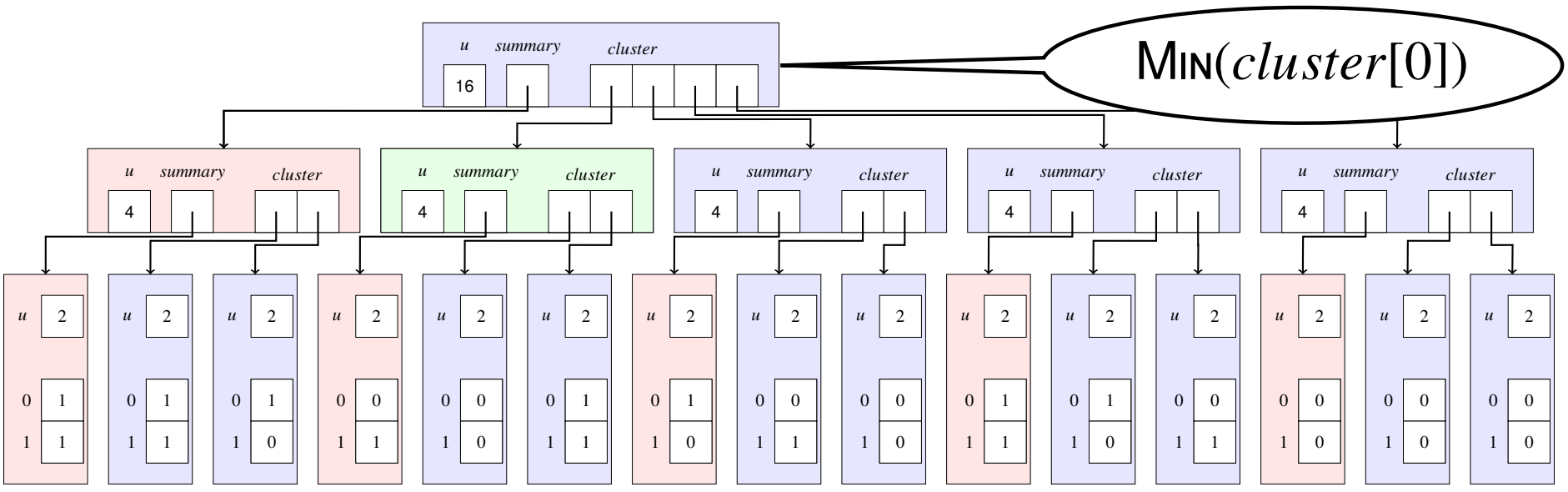
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

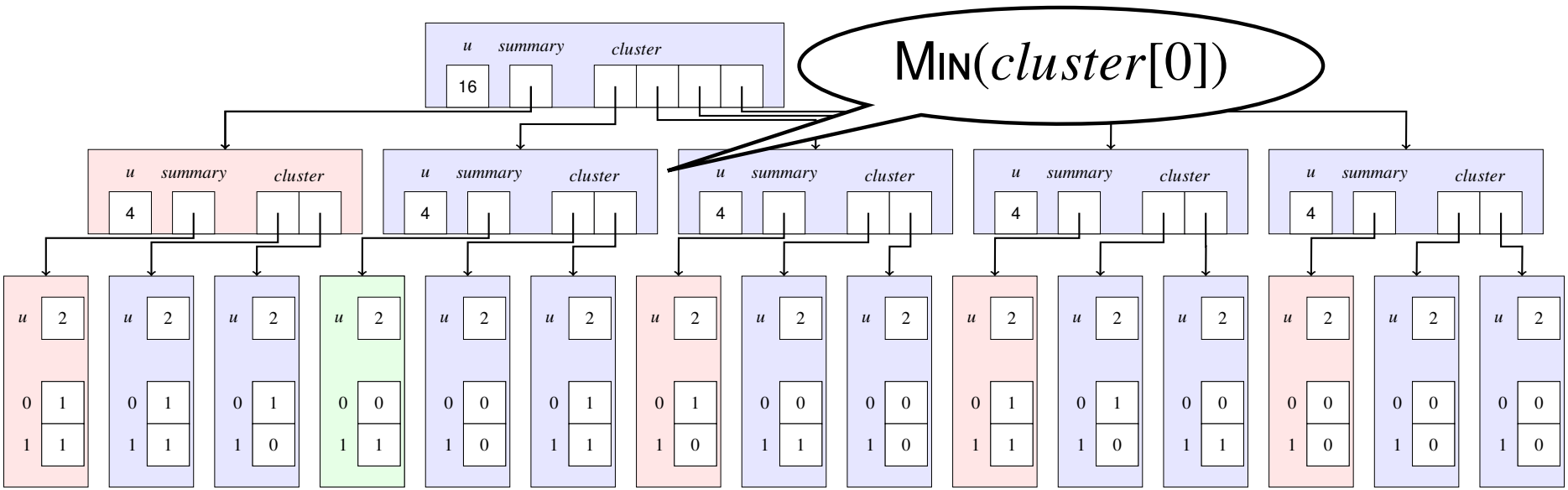
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

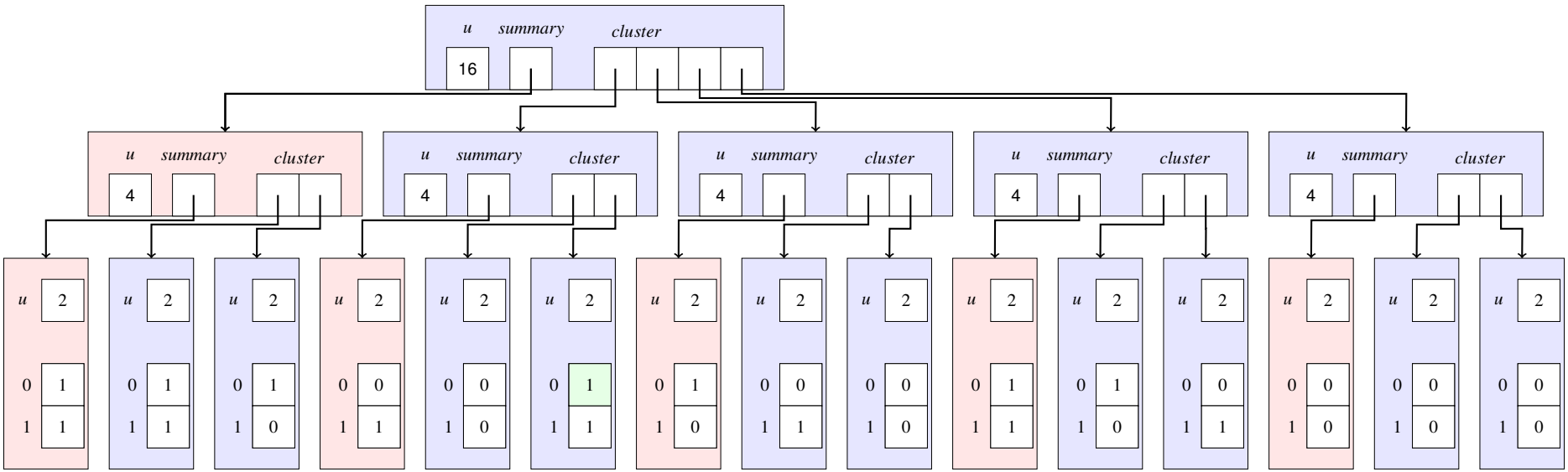
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Min(V)

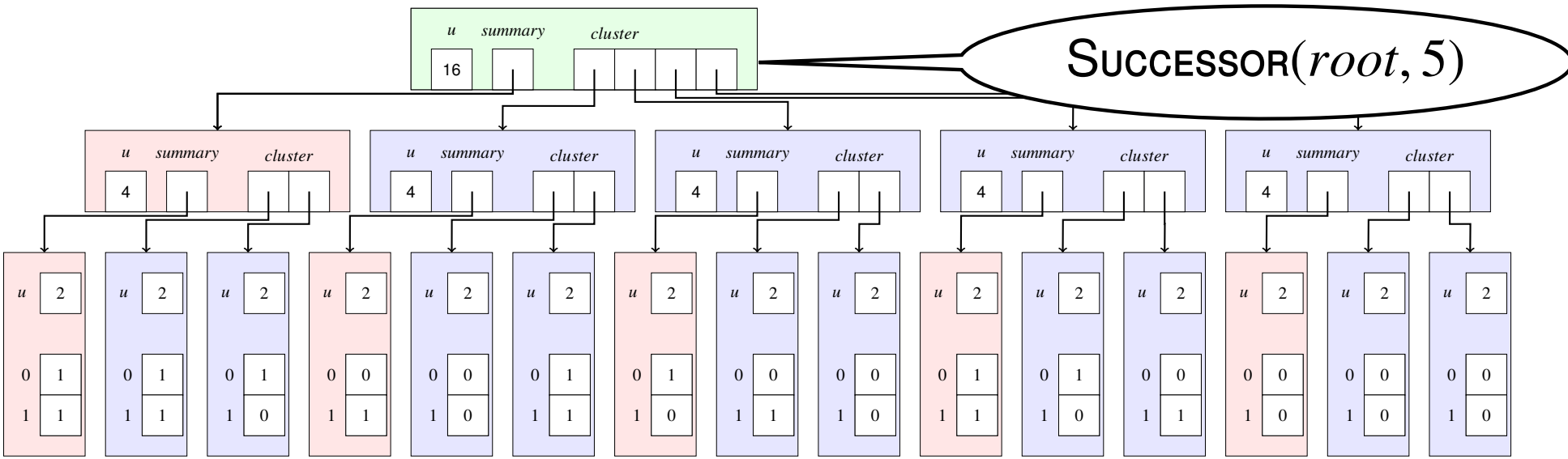
summary に対して Min($V.summary$) を呼び出し,
返り値を *mincluster* とすると,
Min($V.cluster[mincluster]$) を再帰的に呼び出す.
再帰関数の返り値 *ret* はその pvEB 構造に対応した値なので,
返す値は, index($mincluster, ret$) となる.



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

pvEB 構造 操作: Successor(V, x)

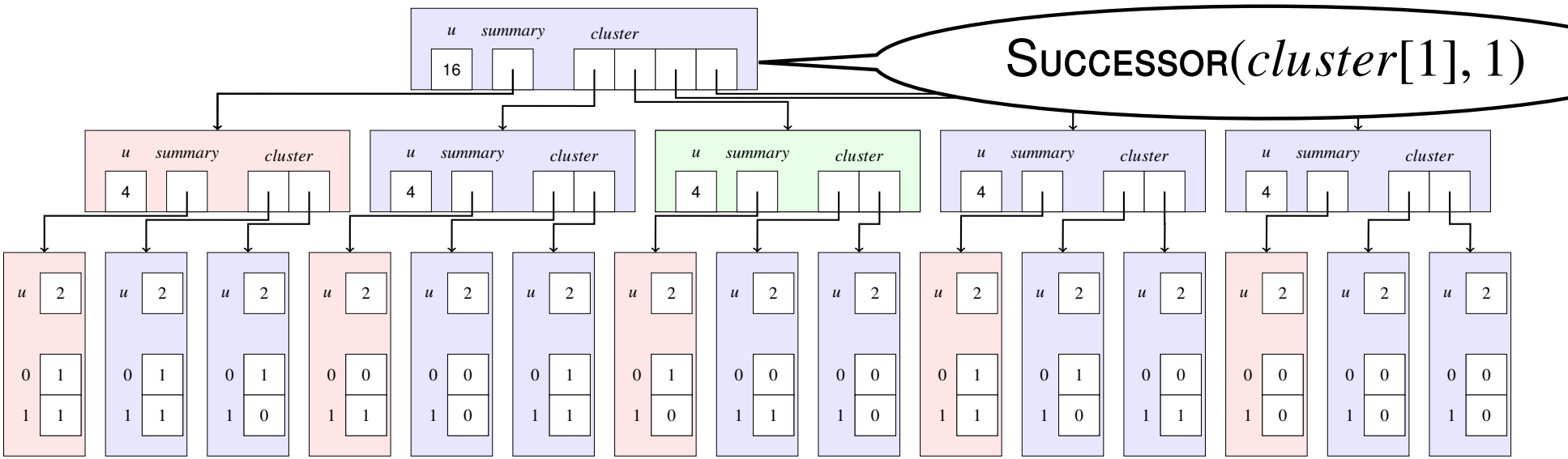
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

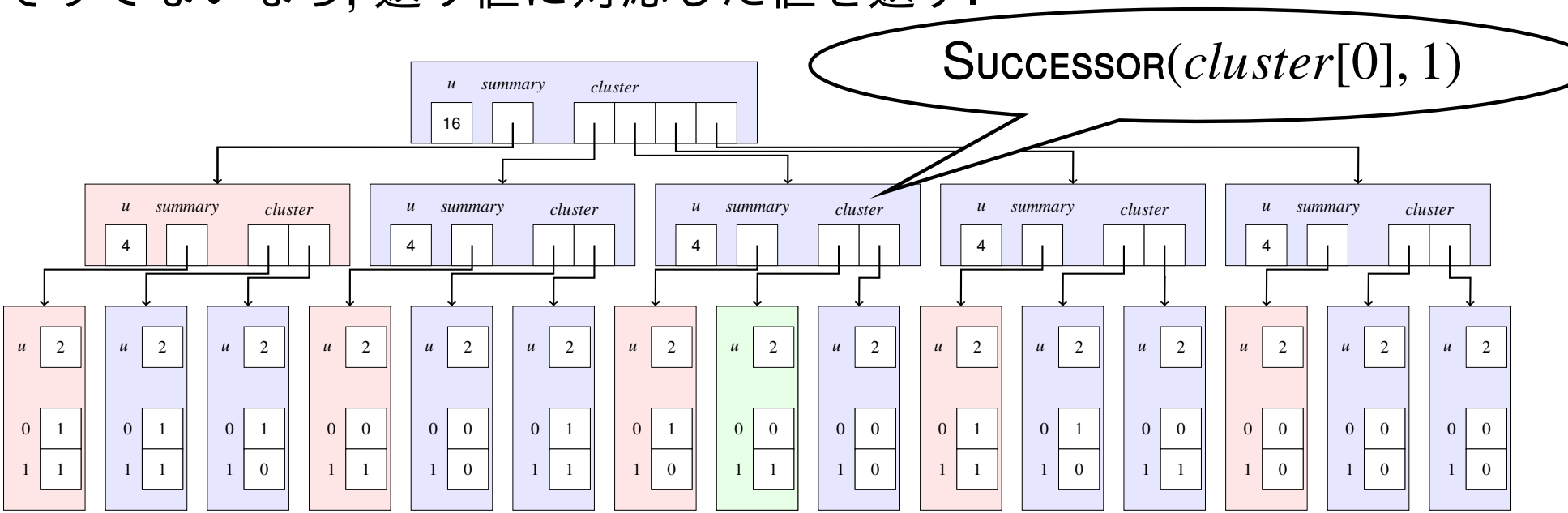
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

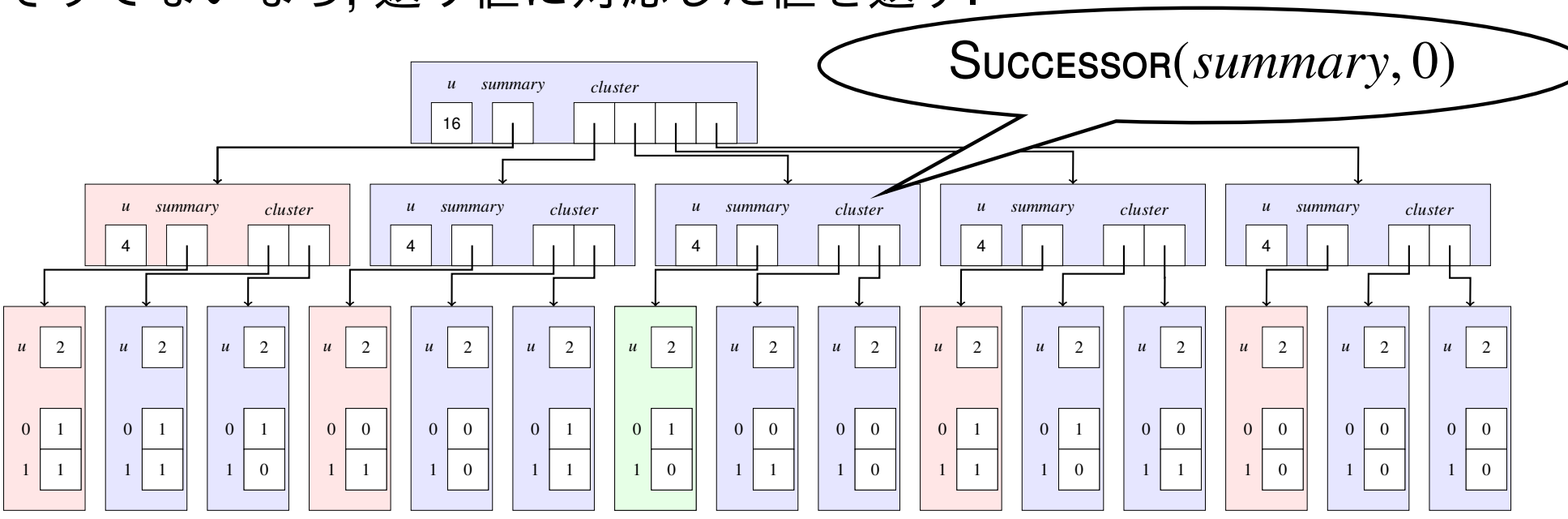
Successor$(cluster[\text{high}(x)], \text{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

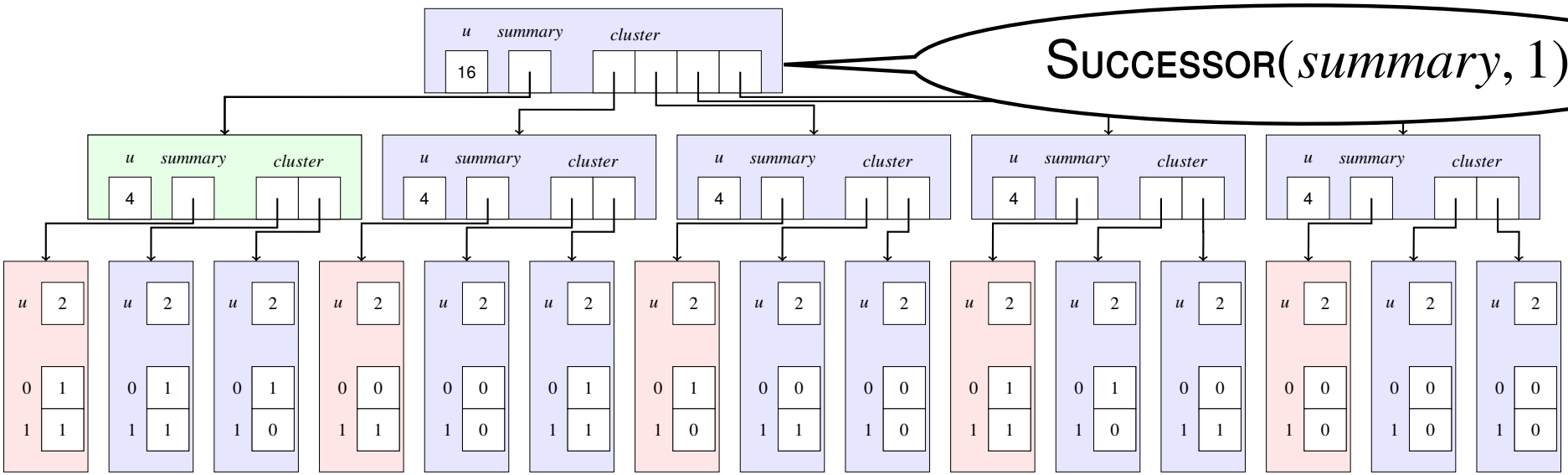
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

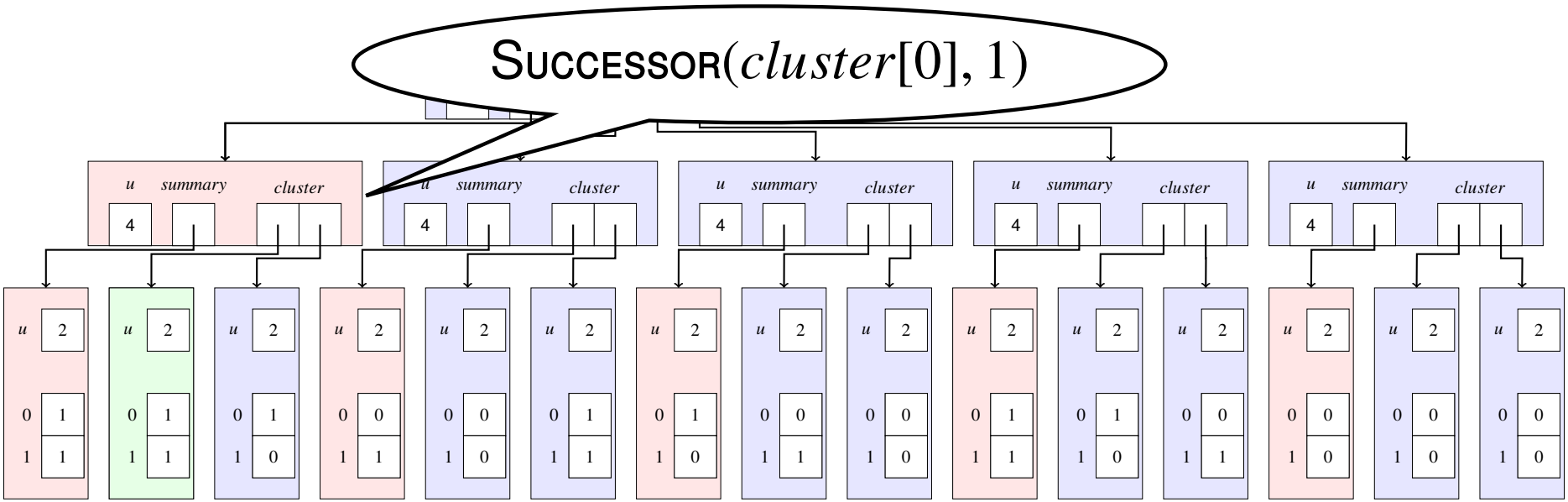
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

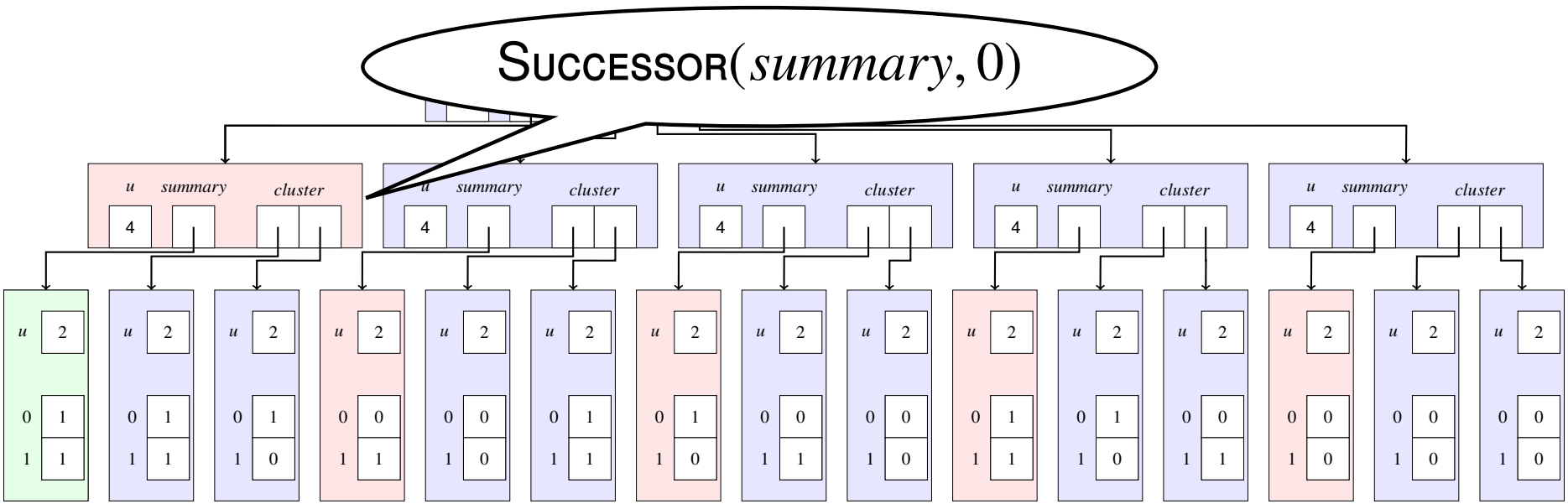
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

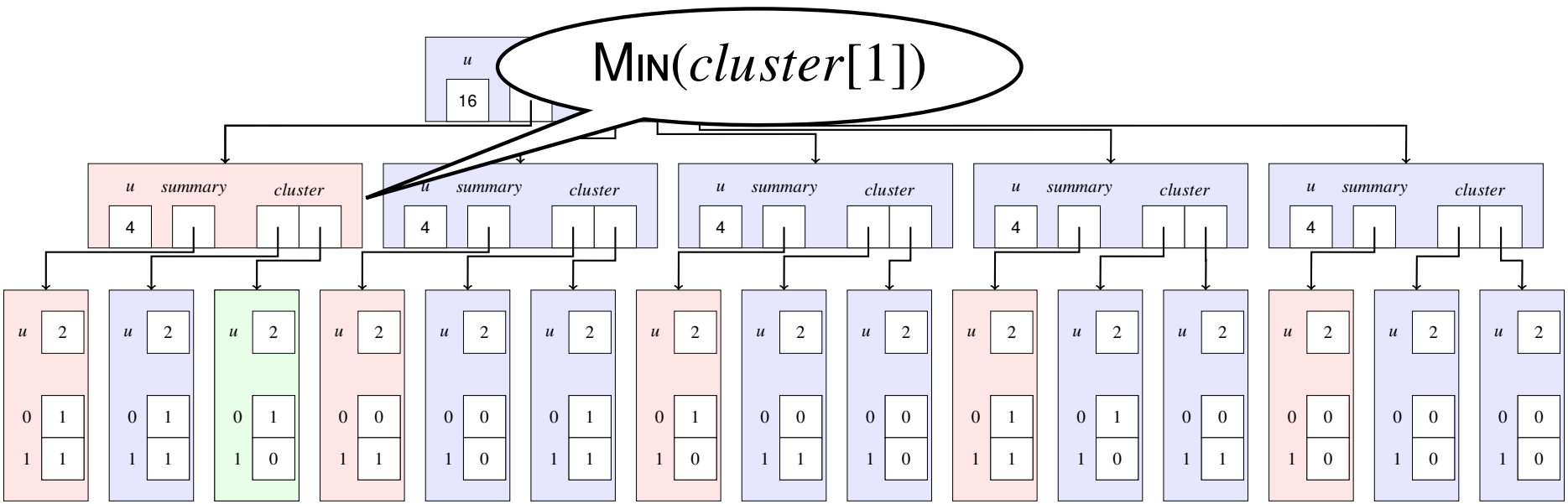
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

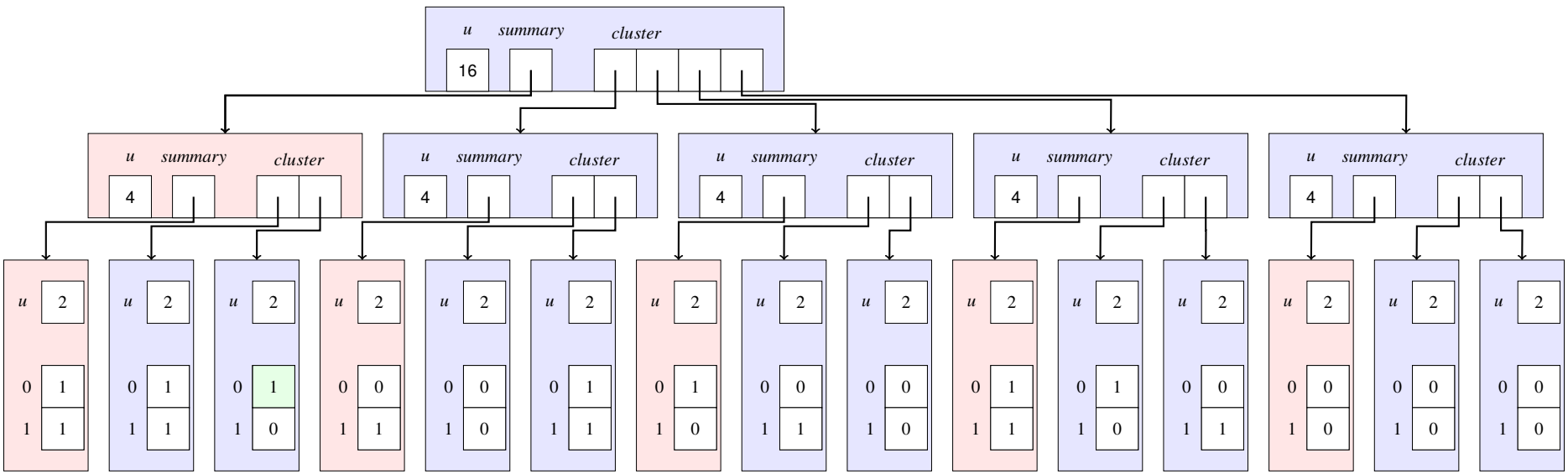
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

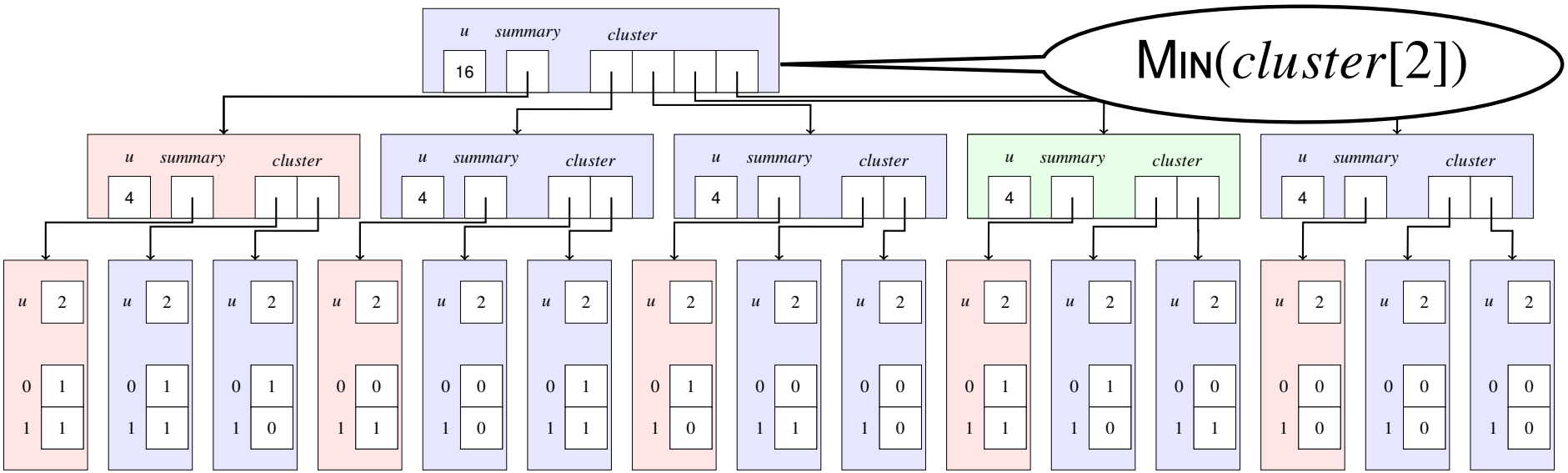
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

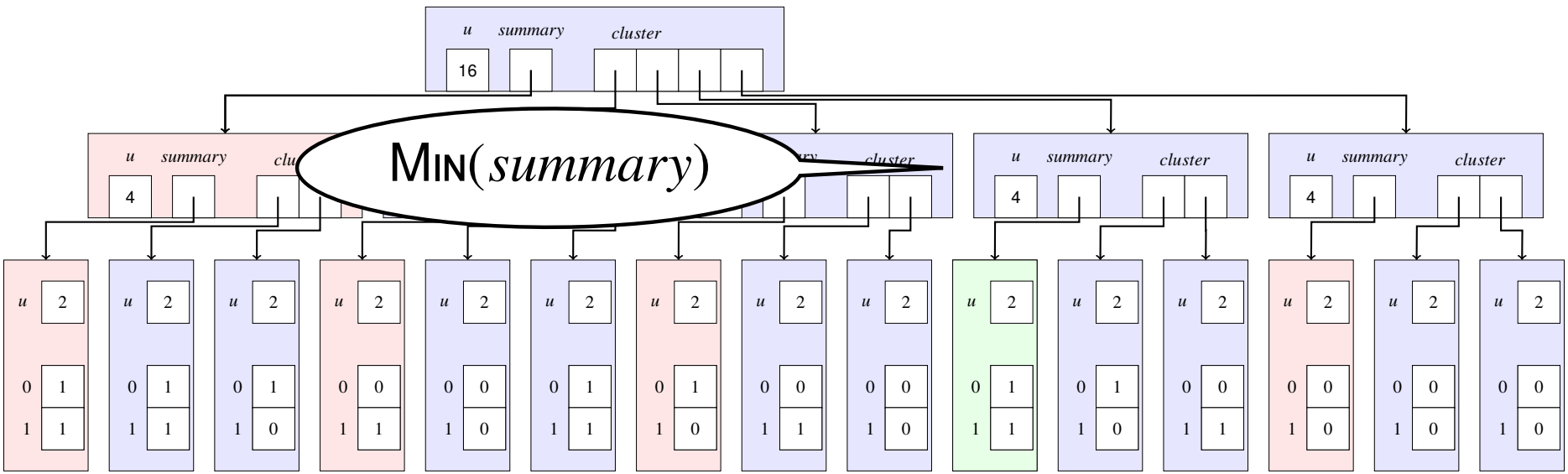
Successor$(cluster[\text{high}(x)], \text{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

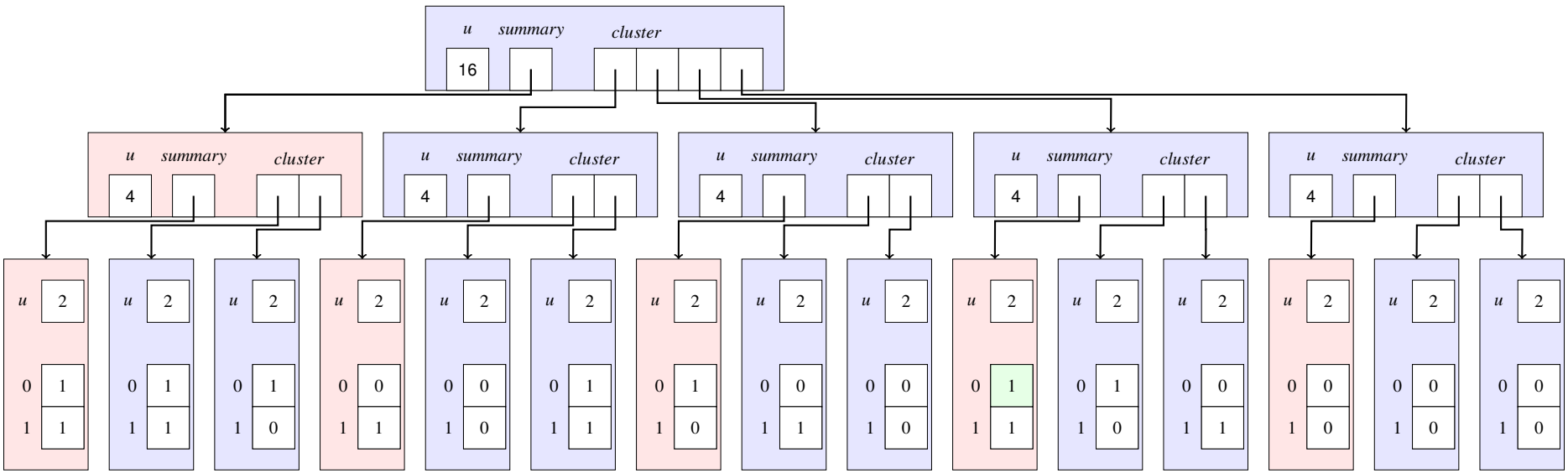
Successor$(cluster[\text{high}(x)], \text{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

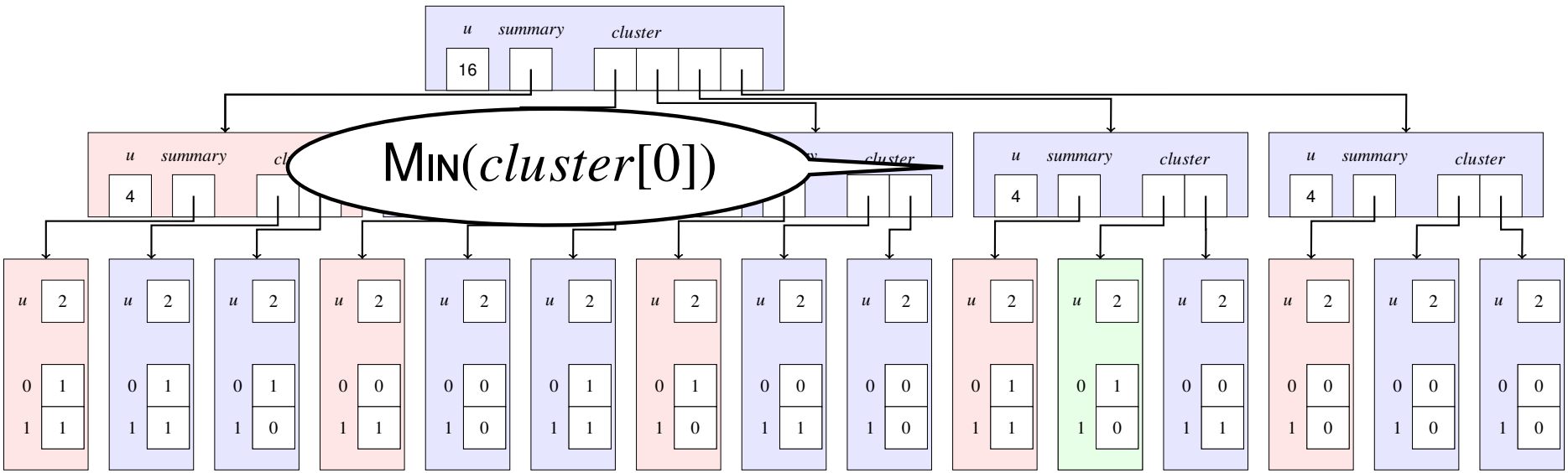
Successor$(cluster[\text{high}(x)], \text{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

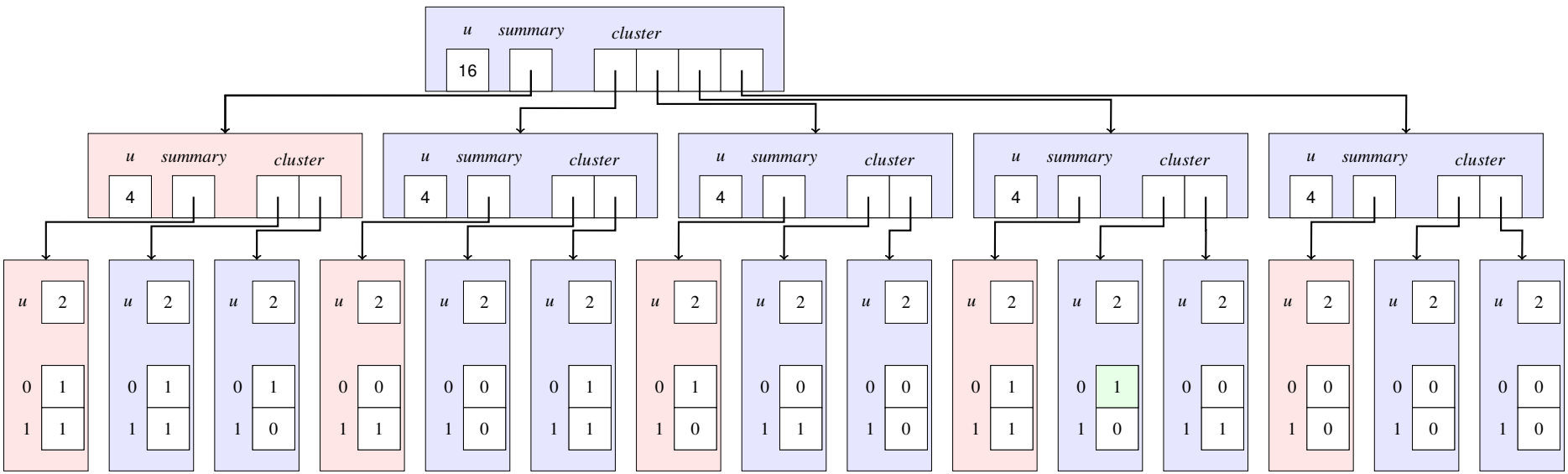
Successor$(cluster[\text{high}(x)], \text{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: Successor(V, x)

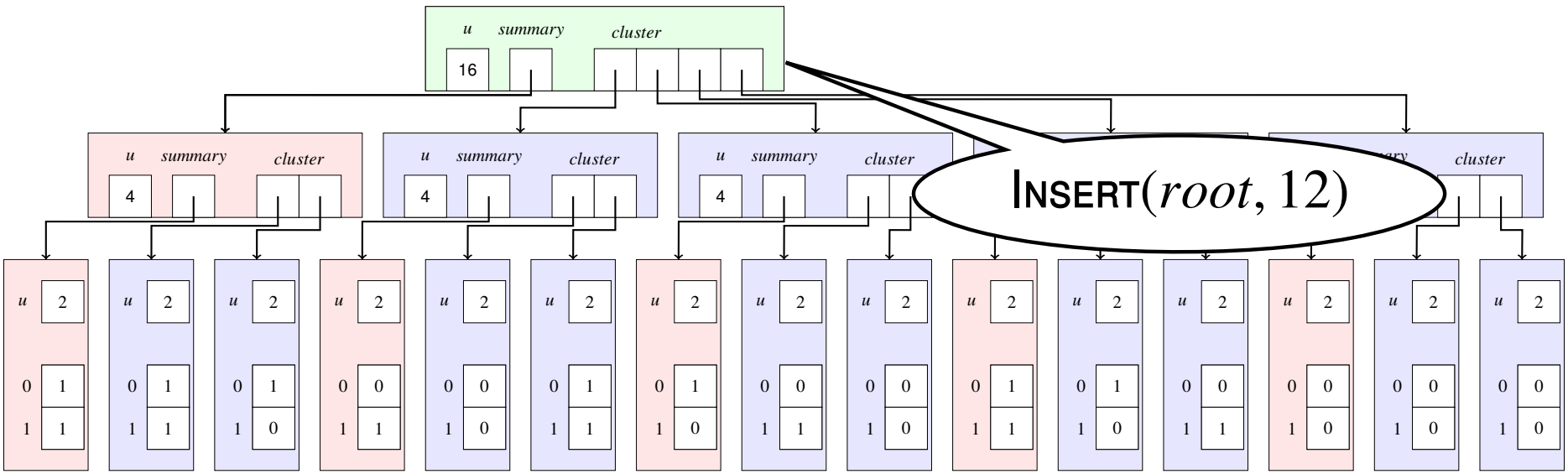
Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を呼び出し,
返り値が NIL なら, Successor$(summary, high(x))$ で,
次に要素を持つ $cluster$ を求め, その最小値を返す.
そうでないなら, 返り値に対応した値を返す.



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

pvEB 構造 操作: INSERT(V, x)

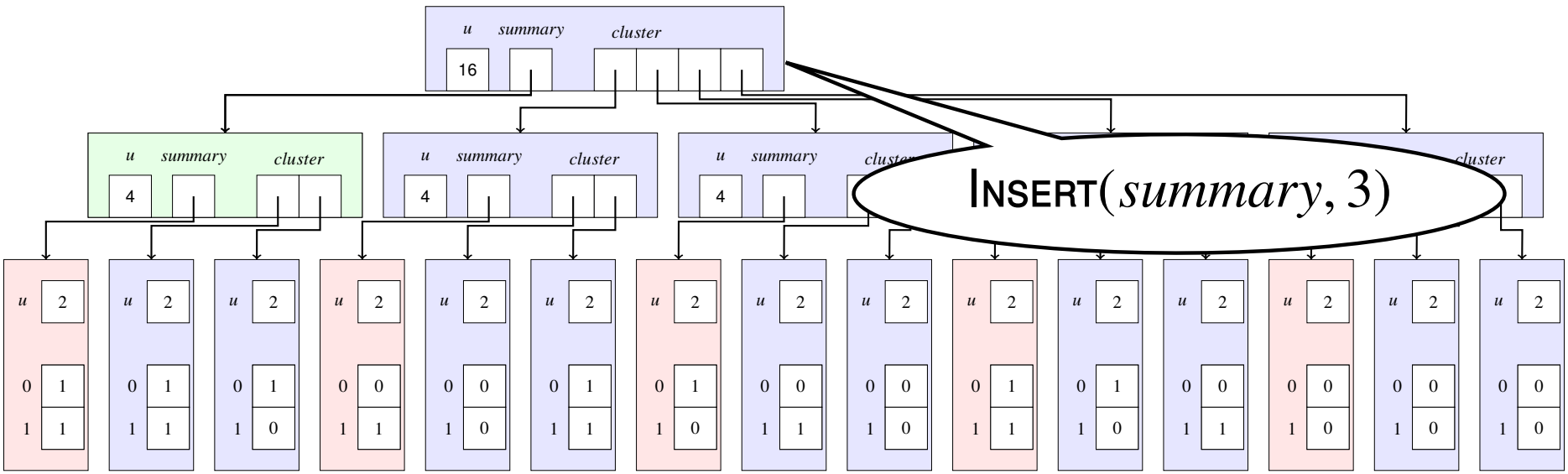
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

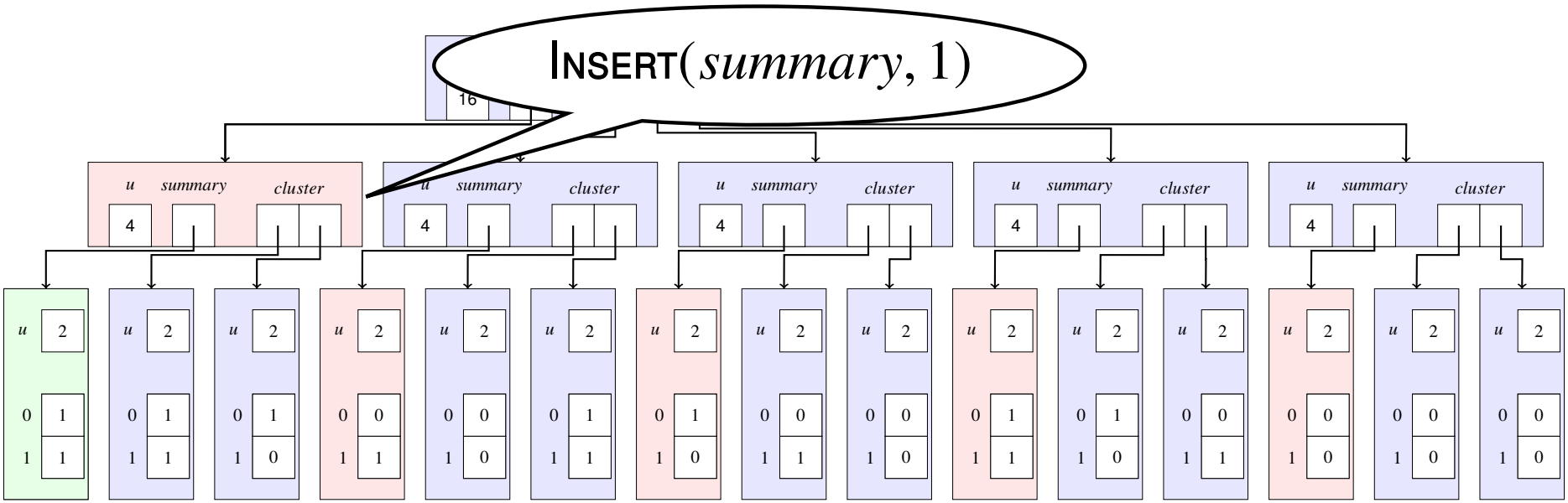
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

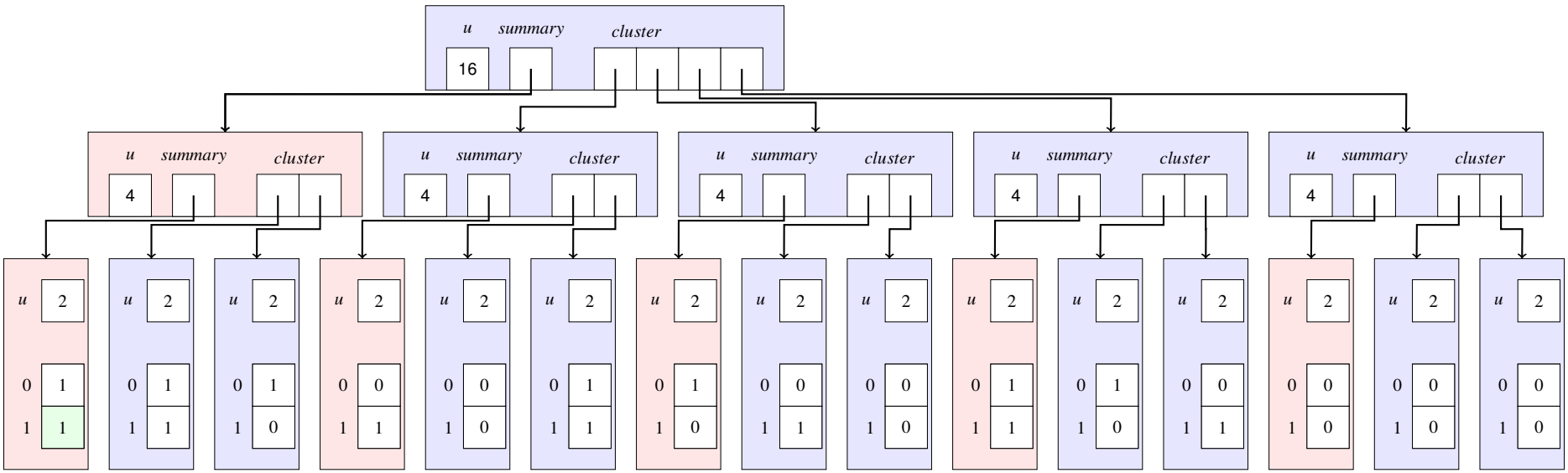
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

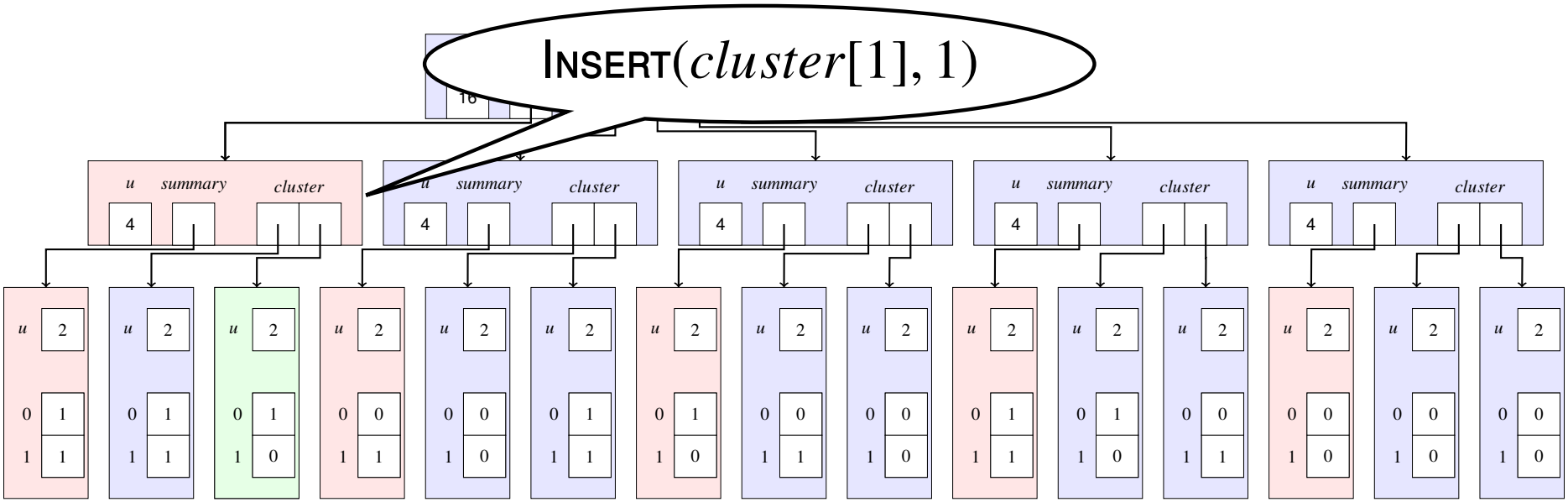
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

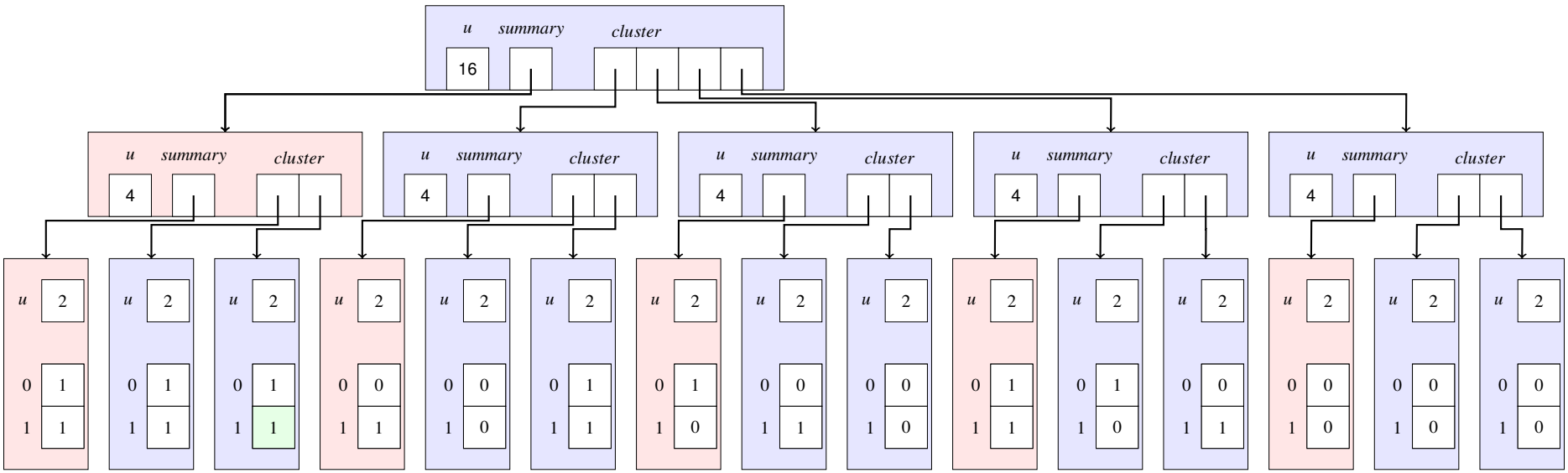
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

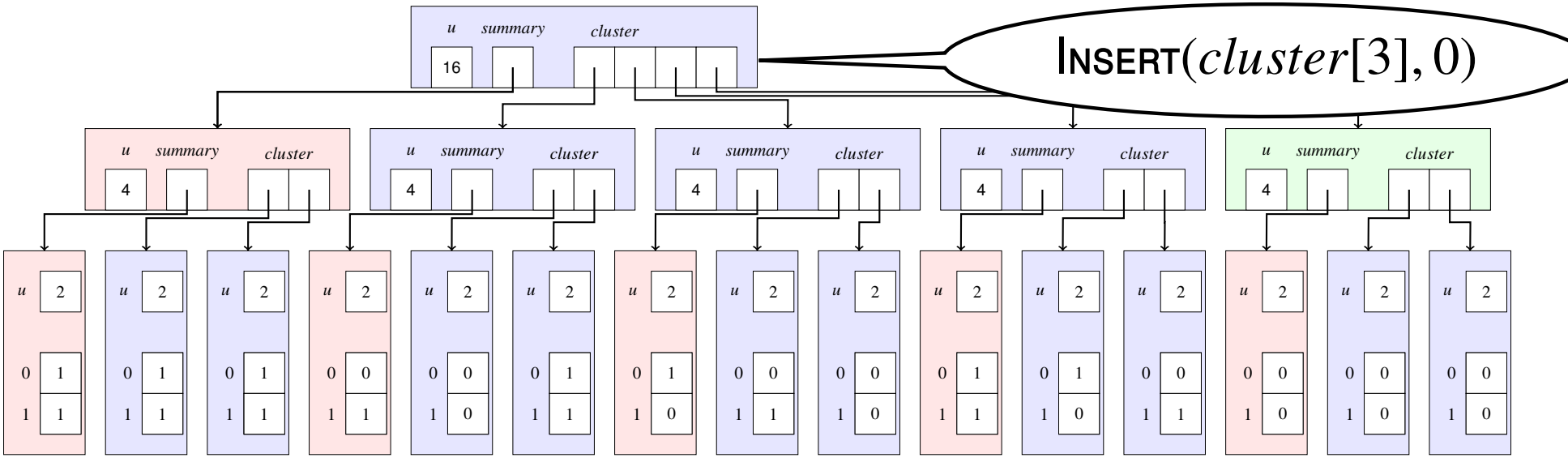
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

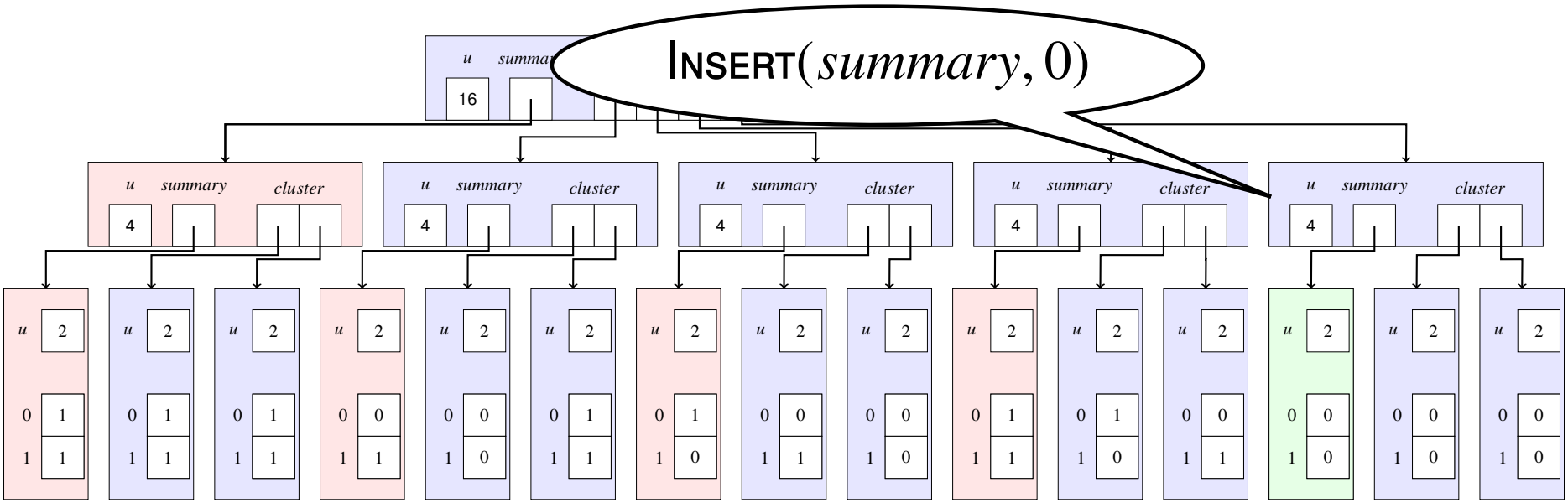
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

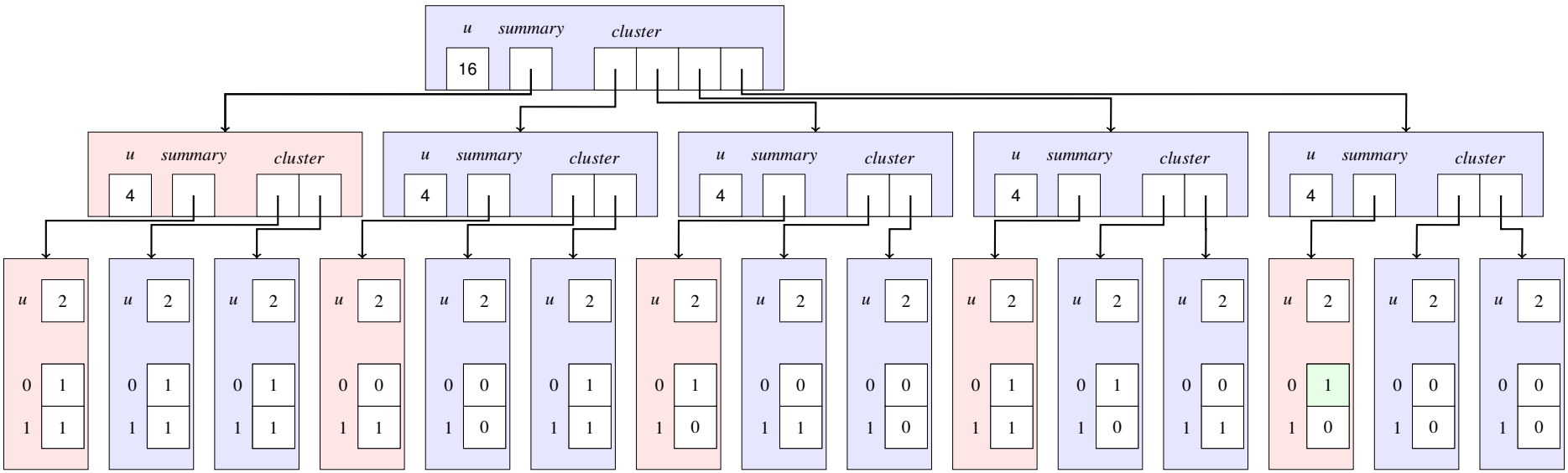
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

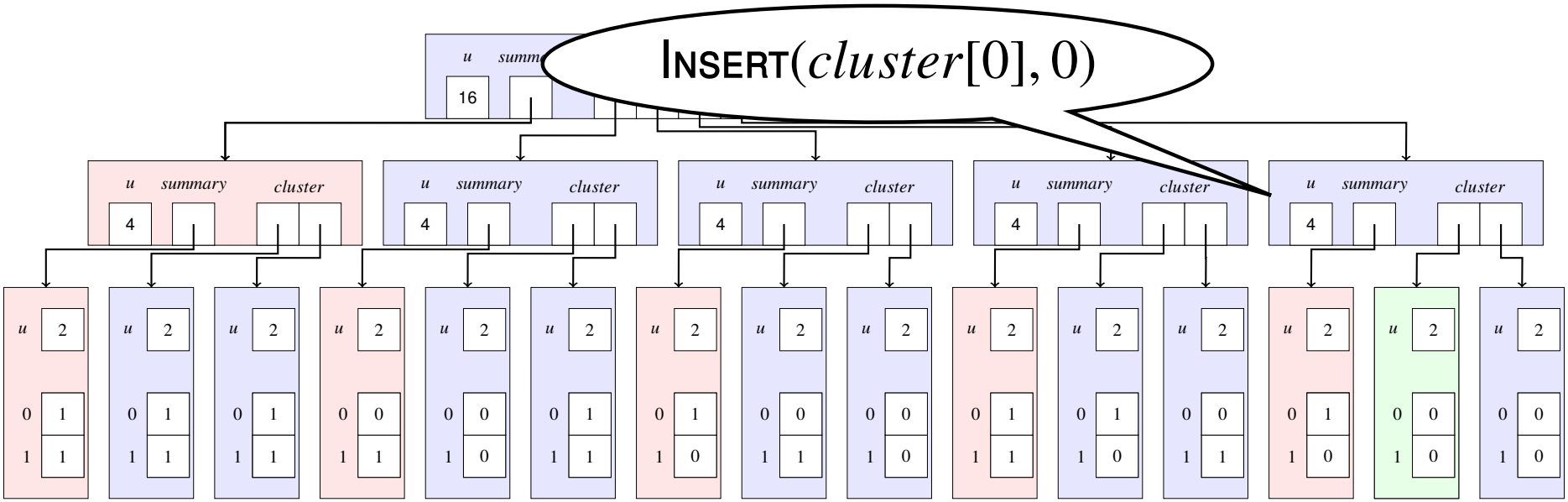
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

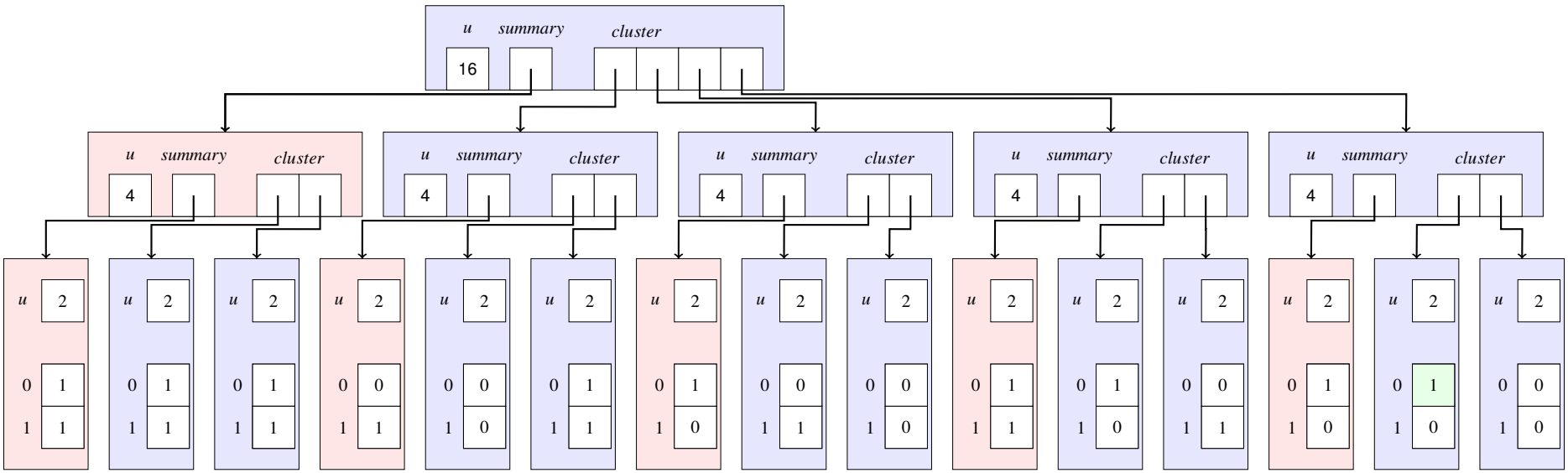
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: INSERT(V, x)

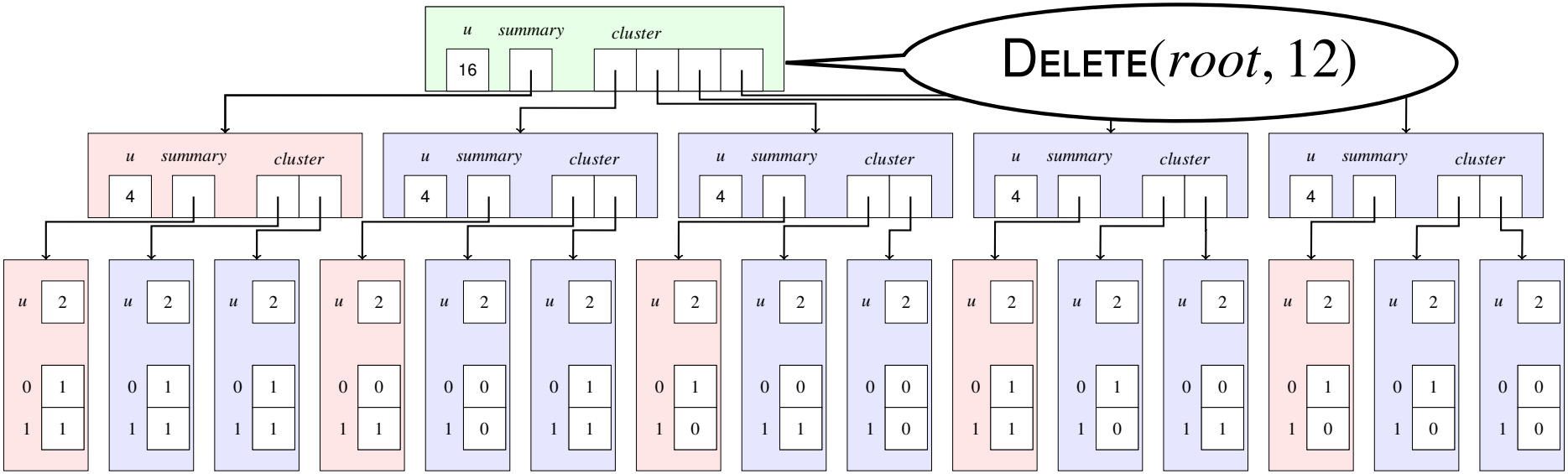
INSERT$(summary, high(x))$, INSERT$(cluster[high(x)], low(x))$ を,
再帰的に呼び出す.
$u = 2$ の pvEB 構造に到達したら, 対応する要素を更新する.



INSERT$(V, 12), V = \{2, 3, 5, 8, 11\}$

pvEB 構造 操作: DELETE(V, x)

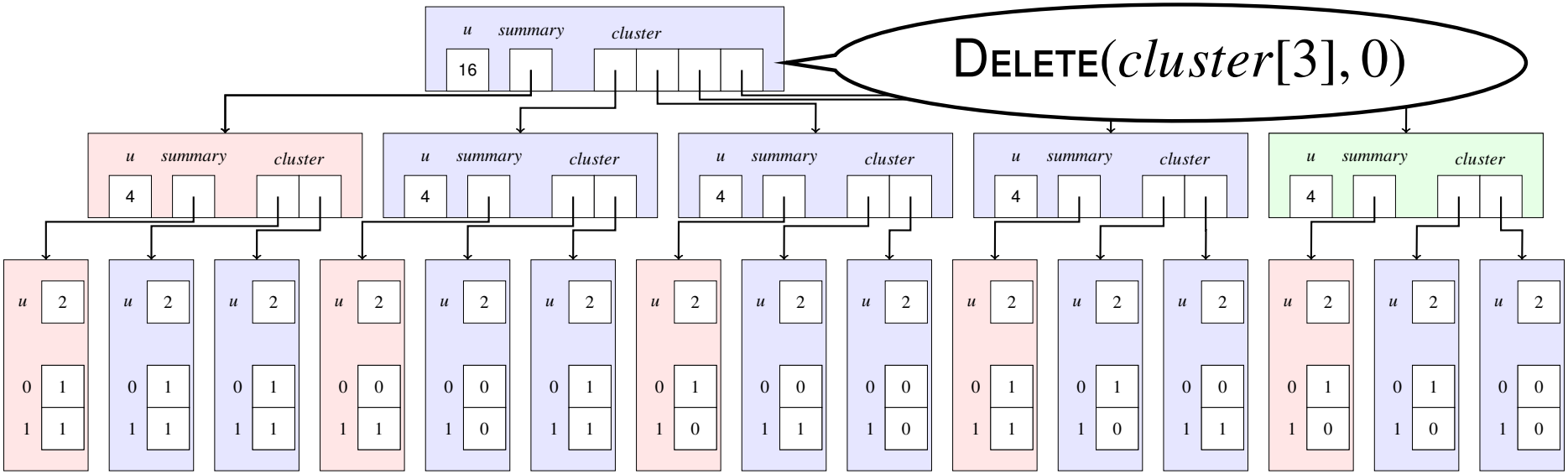
DELETE($cluster[high(x)], low(x)$) により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE($summary, high(x)$) により,
$summary$ を更新する.



DELETE($V, 12$), $V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

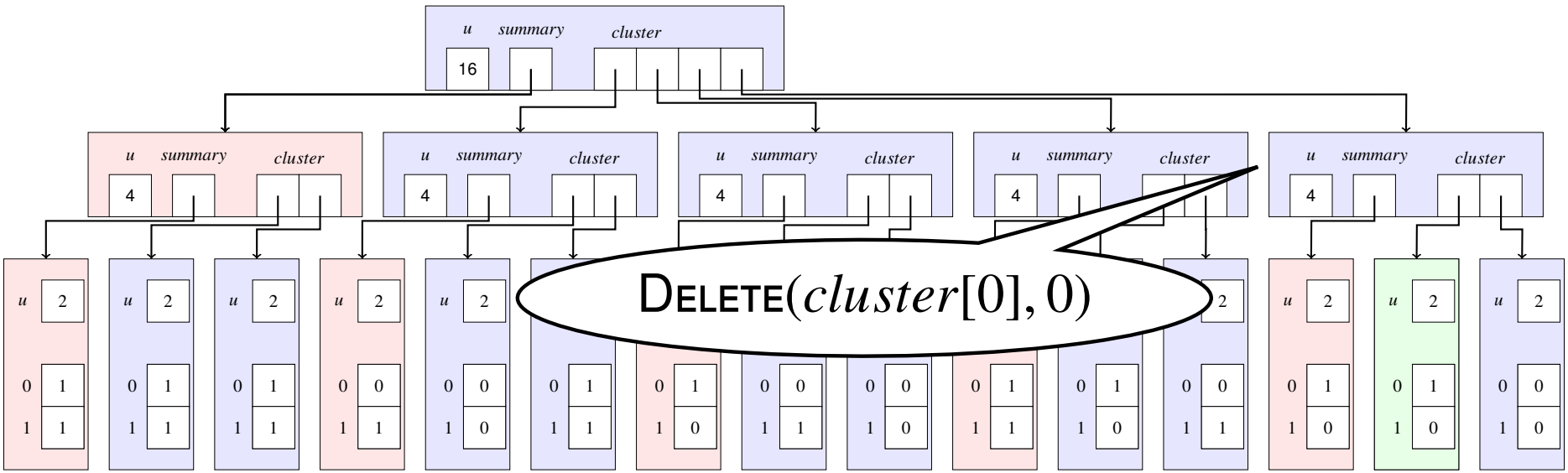
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

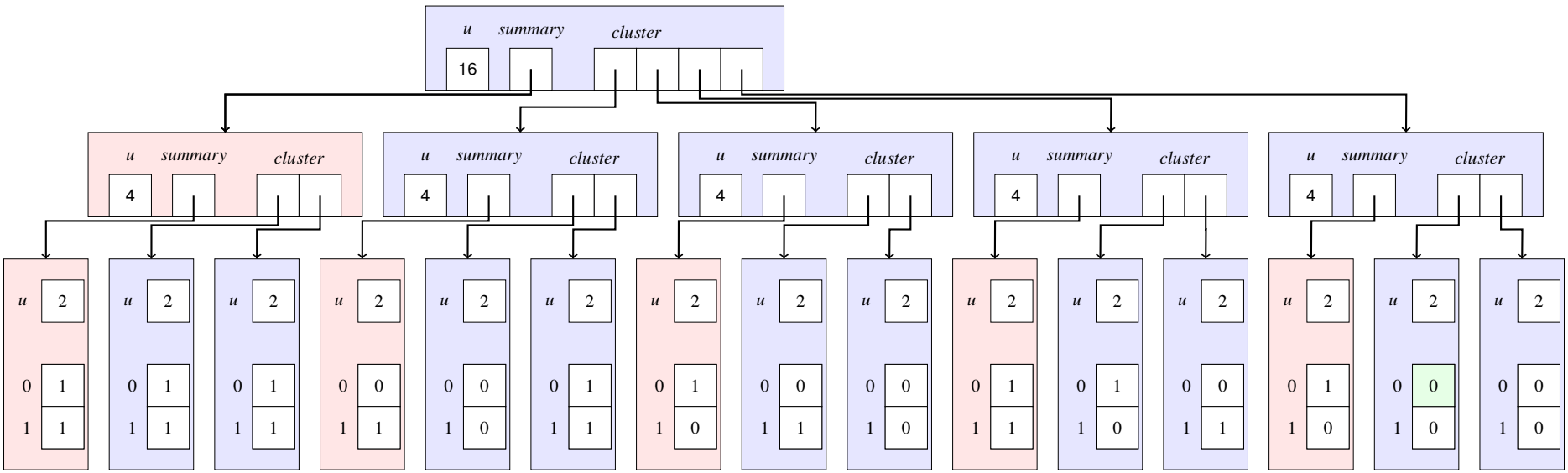
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

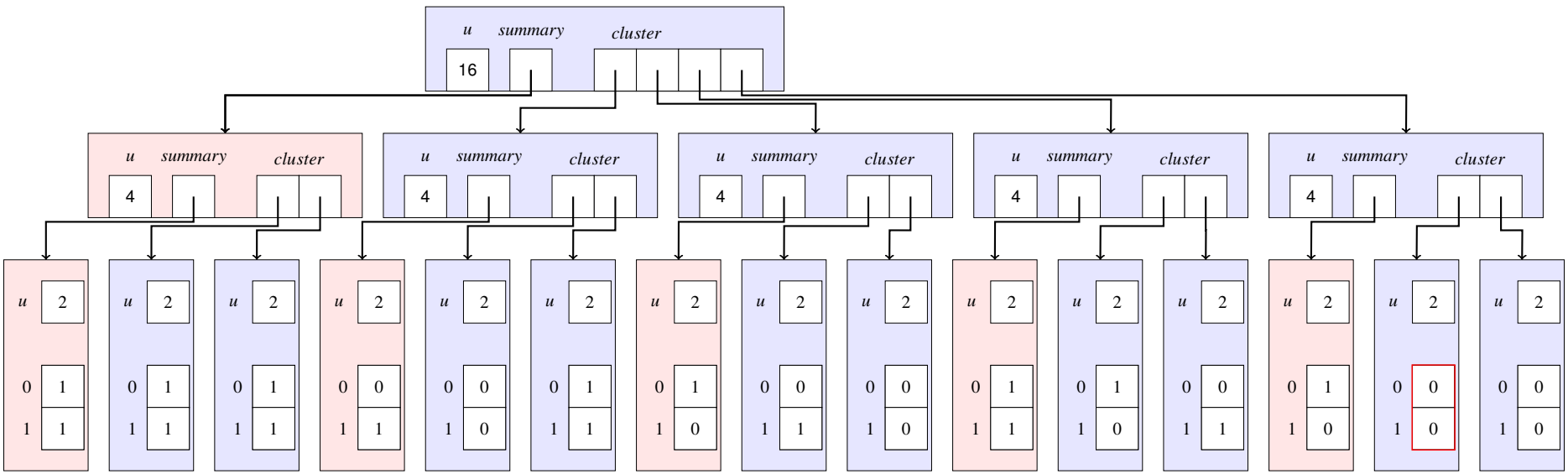
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

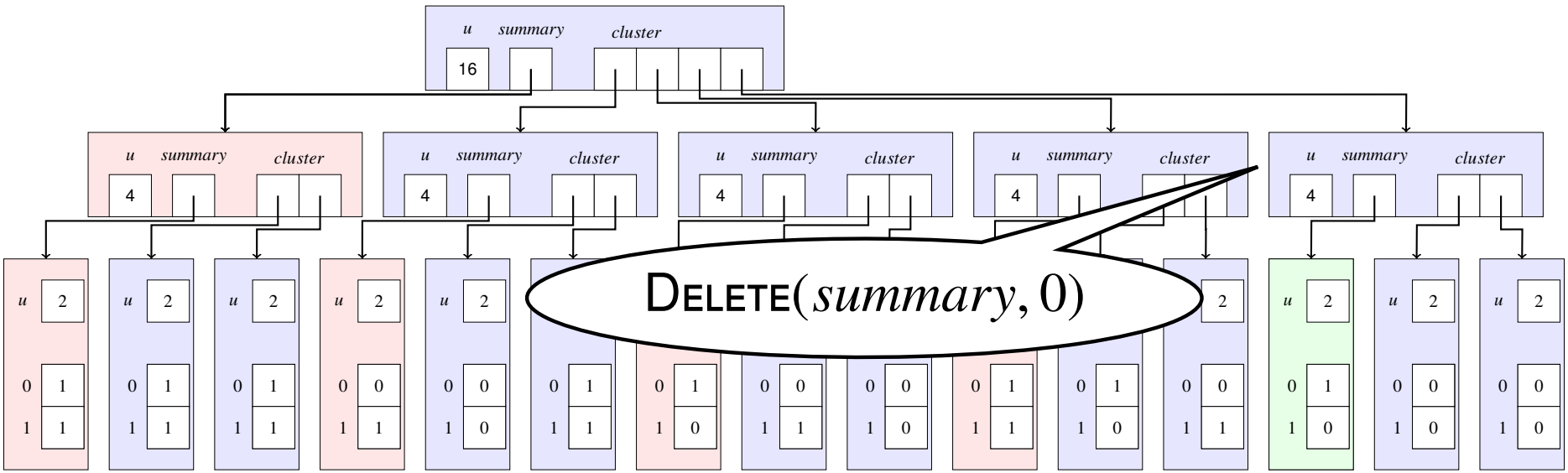
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

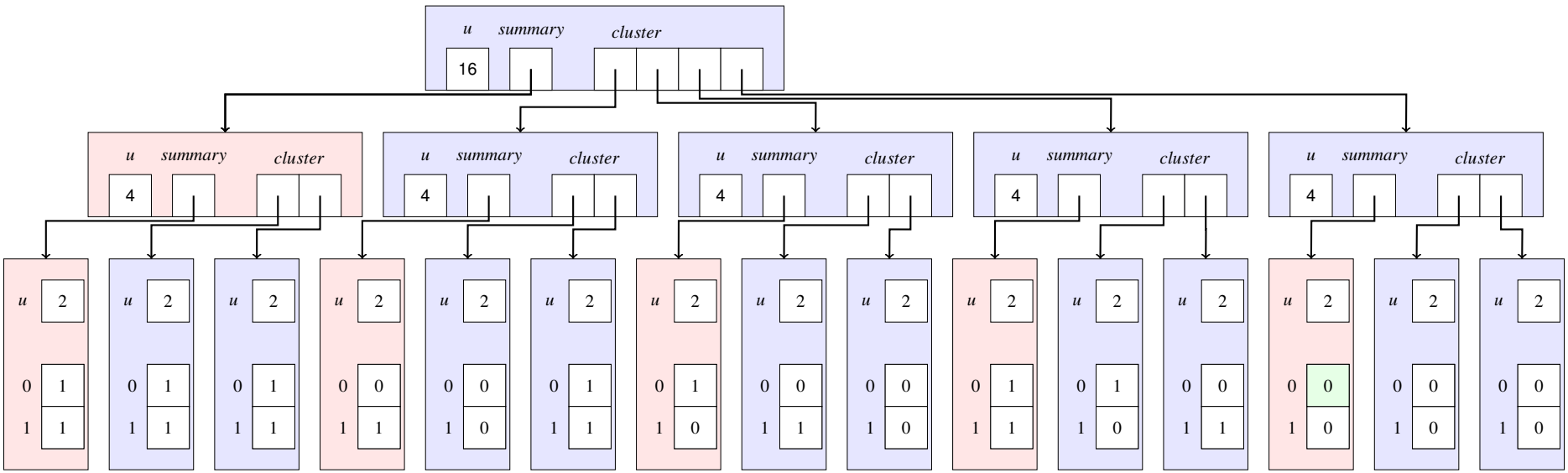
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

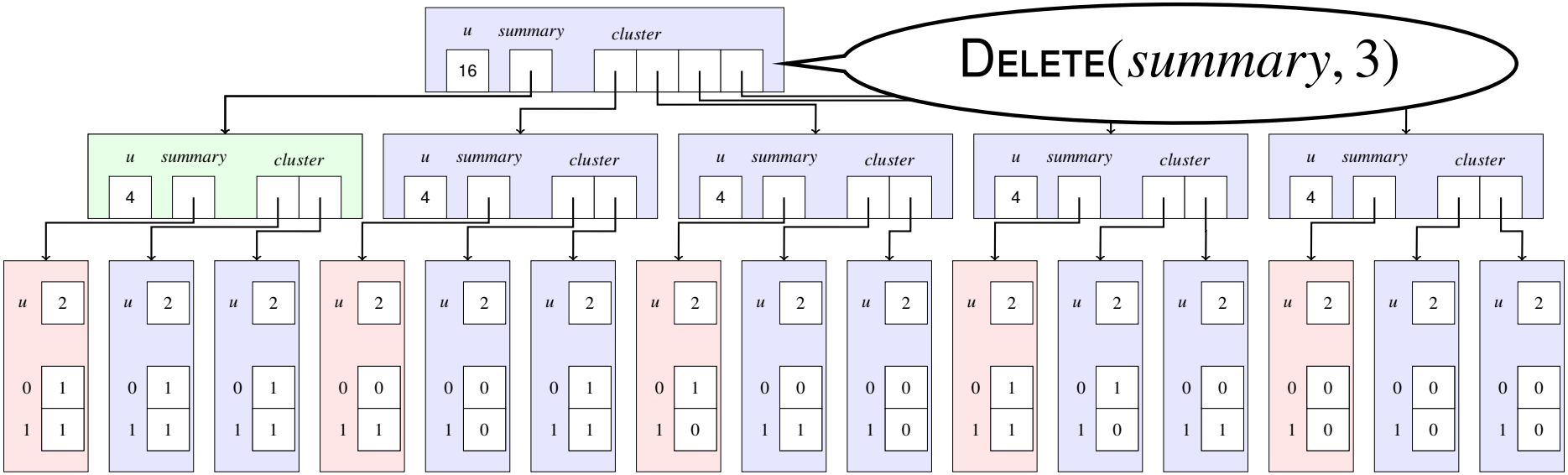
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

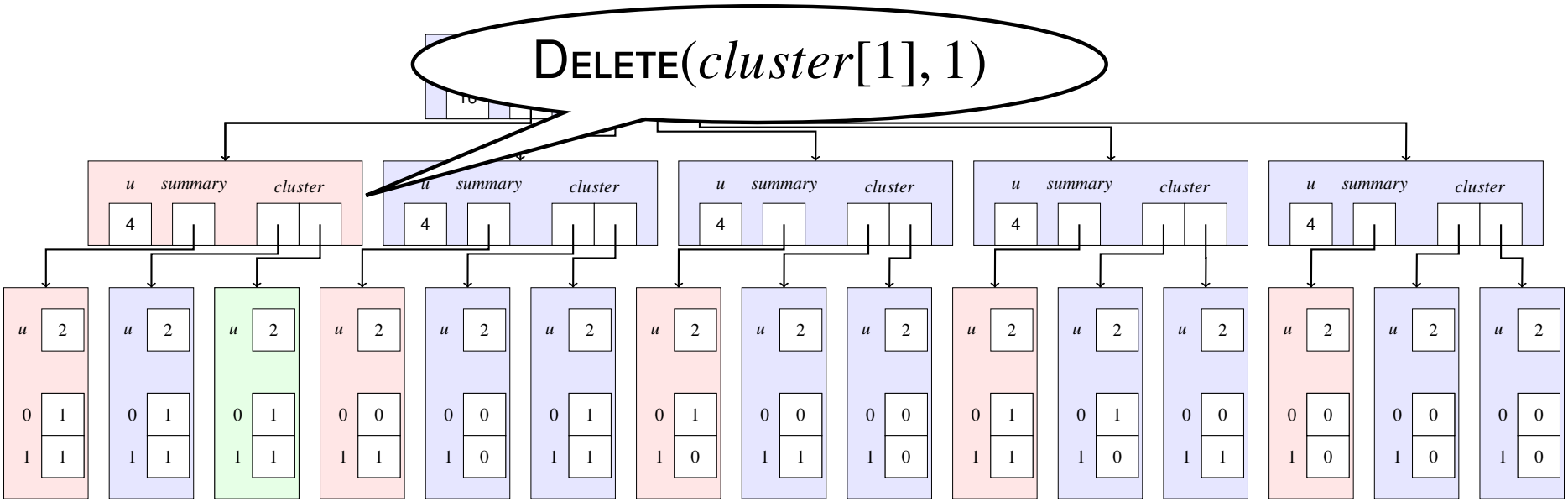
DELETE($cluster[high(x)], low(x)$) により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE($summary, high(x)$) により, $summary$ を更新する.



DELETE($V, 12$), $V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

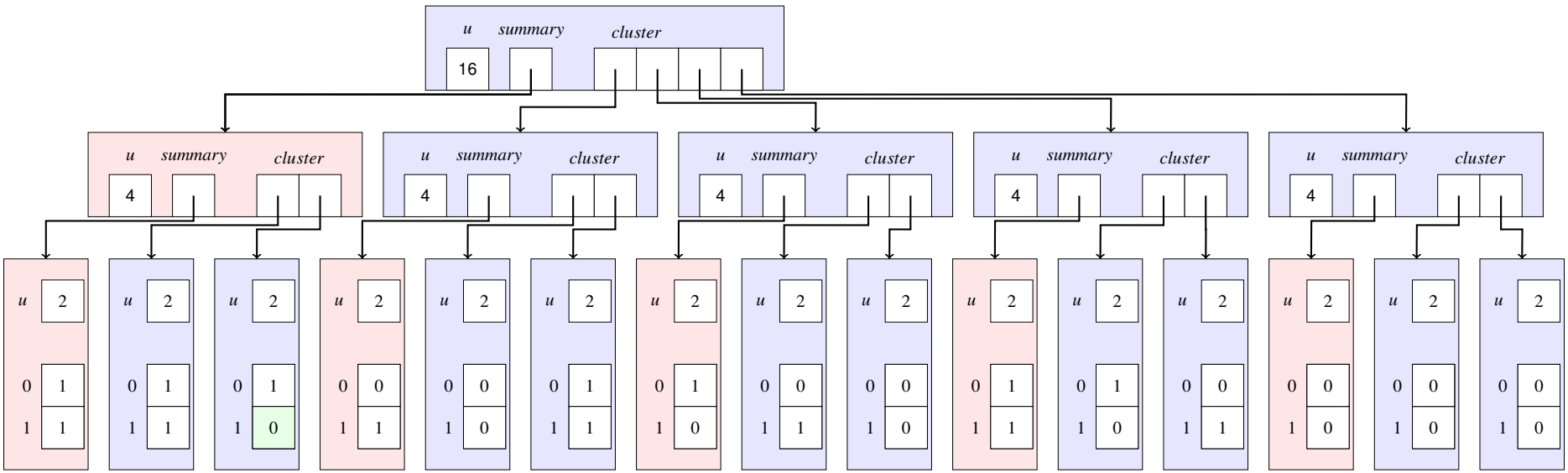
DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 操作: DELETE(V, x)

DELETE$(cluster[high(x)], low(x))$ により, 対応する $cluster$ を更新する.
更新後, $cluster$ に要素がない場合, DELETE$(summary, high(x))$ により, $summary$ を更新する.



DELETE$(V, 12), V = \{2, 3, 5, 8, 11, 12\}$

pvEB 構造 計算量 (1/5)

空間計算量

pvEB 構造は自身の u について, $\sqrt{u}+1$ 個のポインタを持つ.
よって, 全体の空間計算量を $\mathrm{S}(u)$ とすると,

$$\mathrm{S}(u) = (\sqrt{u}+1)\mathrm{S}(\sqrt{u}) + \Theta(\sqrt{u})$$

となる. よって, 以下が成り立つ.

$$\mathrm{S}(u) \leq \left(\prod_{i=1}^{\log\log u}\left(u^{2^{-i}}+1\right)\right)\mathrm{S}(2) + \sum_{i=1}^{\log\log u}\left(\sum_{j=1}^{i}\left(u^{2^{-j}}+1\right)\Theta\left(u^{2^{-i}}\right)\right)$$

上式より,

$$\mathrm{S}(u) = O(u\log\log u)$$

実際には, $\mathrm{S}(u) = O(u)$ となるそうですが, わかりませんでした.

pvEB 構造 計算量 (2/5)

MEMBER(V, x)

MEMBER(V, x) は, 根に相当する pvEB 構造から葉まで探索するので, 時間計算量は $O(\log \log u)$

MIN(V)

MIN(V) は, 時間計算量を $\mathrm{T}(u)$ とすると, 以下のように導出できる.

$$
\begin{aligned}
\mathrm{T}(u) &= 2\mathrm{T}(\sqrt{u}) + O(1) \\
\mathrm{S}(m) &:= \mathrm{T}(2^m) \text{ とすると,} \\
\mathrm{S}(m) &= 2\mathrm{S}\left(\frac{m}{2}\right) + O(1) \\
\mathrm{S}(m) &= \Theta(m) \\
\mathrm{T}(u) &= \mathrm{T}(2^m) = \mathrm{S}(m) = \Theta(m) = \Theta(\log u)
\end{aligned}
$$

pvEB 構造 計算量 (3/5)

Successor(V, x)

Successor(V, x) は, 1 回の処理で, Successor($summary, high(x)$), Successor($cluster[high(x)], low(x)$) を呼び出す.
また, Min(V') も呼び出すので時間計算量 $\mathrm{T}(u)$ は以下の式となる.
Min(V) と同様に導出する.

$$
\begin{aligned}
\mathrm{T}(u) &= 2\mathrm{T}(\sqrt{u}) + \Theta(\log u) \\
\mathrm{S}(m) &= 2\mathrm{S}\left(\frac{m}{2}\right) + \Theta(m) \\
\mathrm{S}(m) &= O(m \log m) \\
\mathrm{T}(u) &= \mathrm{S}(m) = O(m \log m) = O(\log u \log\log u)
\end{aligned}
$$

pvEB 構造 計算量 (4/5)

INSERT(V, x)

INSERT(V, x) は *summary*, *cluster* それぞれに再帰的に処理を行うので，
MIN(V) と同様に時間計算量は $\Theta(\log u)$ となる．

pvEB 構造 計算量 (5/5)

DELETE(V, x)

DELETE(V, x) は, $cluster$ で更新後,
$cluster$ に要素がないか確認する必要がある.
$cluster$ は $\sqrt{u}$ 個の要素を持ち,
それぞれに MEMBER($cluster[\mathrm{high}(x)], i$) で確認するので,
時間計算量は $O(\sqrt{u} \log \log u)$ となる.

pvEB 構造 欠点

- MIN(V) は, *summary*, *cluster* 両方に対し再帰的に処理
- SUCCESSOR(V, x) は, 上に加え MIN(V, x) を処理
- INSERT(V, x) は, *summary*, *cluster* 両方を再帰的に更新
- DELETE(V, x) は, *cluster* の線形処理がボトルネック

pvEB 構造 欠点

- MIN(V) は, *summary*, *cluster* 両方に対し再帰的に処理
  → 再帰呼び出しは 1 回まで
- SUCCESSOR(V, x) は, 上に加え MIN(V, x) を処理
  → 再帰呼び出しは 1 回まで
- INSERT(V, x) は, *summary*, *cluster* 両方を再帰的に更新

- DELETE(V, x) は, *cluster* の線形処理がボトルネック

pvEB 構造 欠点

- MIN(V) は, *summary*, *cluster* 両方に対し再帰的に処理
  → 再帰呼び出しは 1 回まで
- SUCCESSOR(V, x) は, 上に加え MIN(V, x) を処理
  → 再帰呼び出しは 1 回まで
- INSERT(V, x) は, *summary*, *cluster* 両方を再帰的に更新
  → 更新処理を最低限に
- DELETE(V, x) は, *cluster* の線形処理がボトルネック

pvEB 構造 欠点

- MIN(V) は, *summary*, *cluster* 両方に対し再帰的に処理
  → 再帰呼び出しは 1 回まで
- SUCCESSOR(V, x) は, 上に加え MIN(V, x) を処理
  → 再帰呼び出しは 1 回まで
- INSERT(V, x) は, *summary*, *cluster* 両方を再帰的に更新
  → 更新処理を最低限に
- DELETE(V, x) は, *cluster* の線形処理がボトルネック
  → 要素がないかを定数時間で判定

Contents

van Emde Boas tree

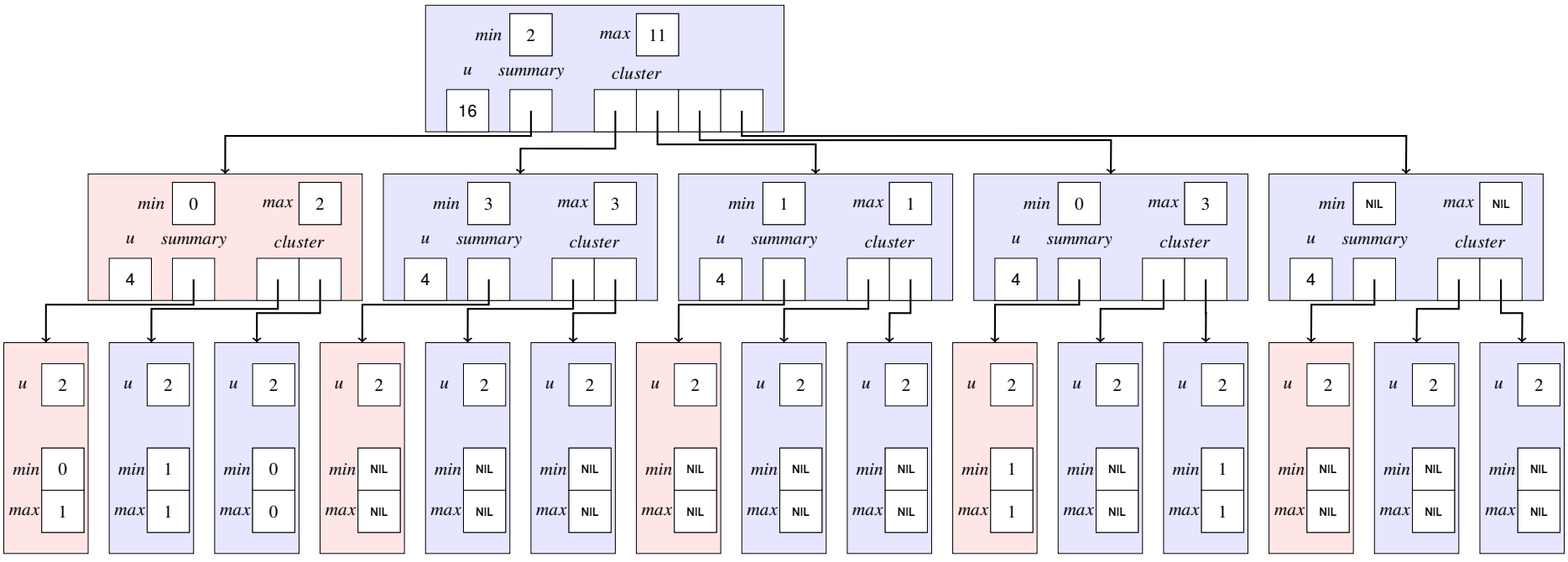
pvEB 構造に最小値, 最大値を加えた van Emde Boas tree を考える.

van Emde Boas tree

van Emde Boas tree (vEB 木) は,
van Emde Boas node (vEB ノード) で構成されるデータ構造

- 各 vEB ノードは pvEB 構造と同様の変数に加え,
  そのノードが持つ要素の最小値, 最大値を保持
- $u = 2$ の場合は, ポインタを持たず, 最小値, 最大値のみを保持
- 最小値の要素は, 子のノードに持たせない

van Emde Boas tree

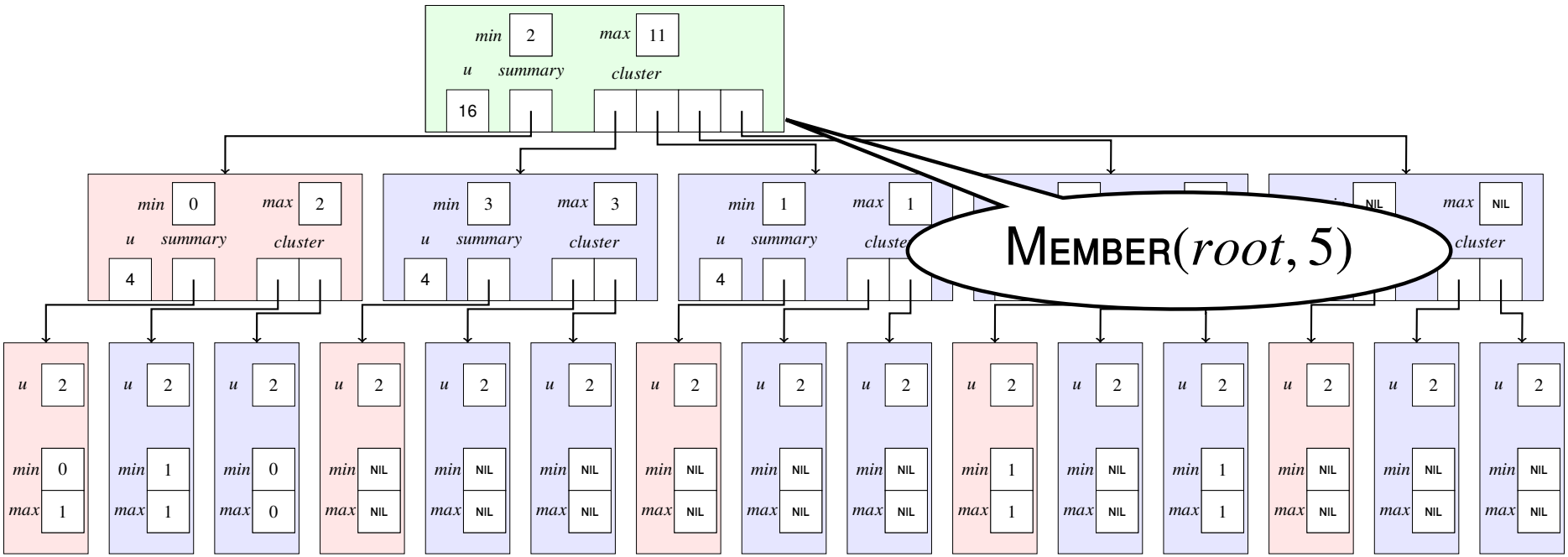


van Emde Boas tree

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

vEB 木 操作: MEMBER(V, x)

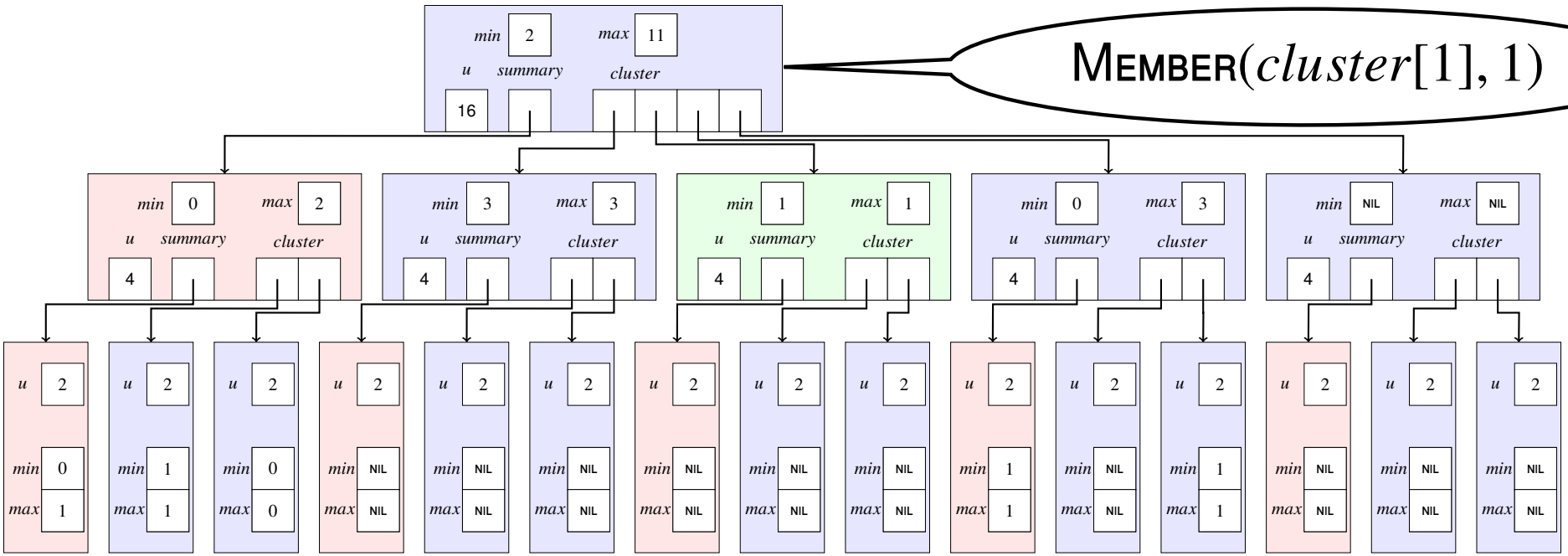
min, max が x ならば TRUE を返す.
そうでないならば, MEMBER($cluster$[high(x)], low(x)]) を返す.



MEMBER($V, 5$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: MEMBER(V, x)

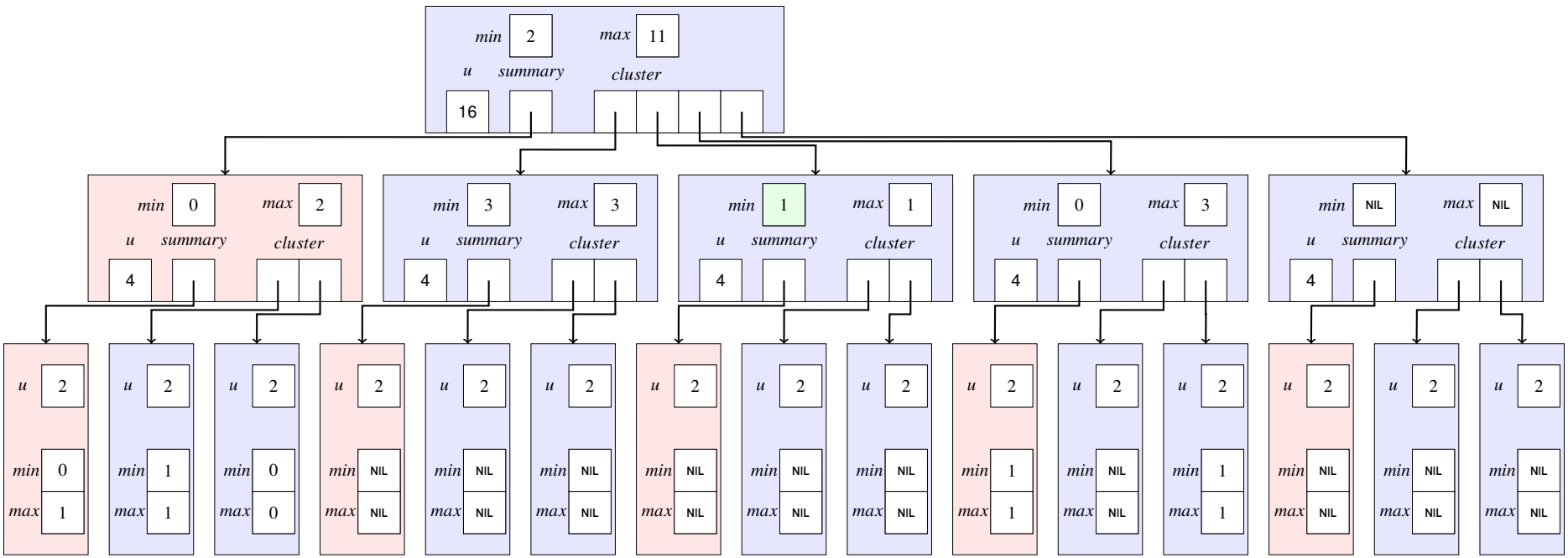
min, max が x ならば TRUE を返す.
そうでないならば, MEMBER($cluster$[high(x)], low(x)]) を返す.



MEMBER($V, 5$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: MEMBER(V, x)

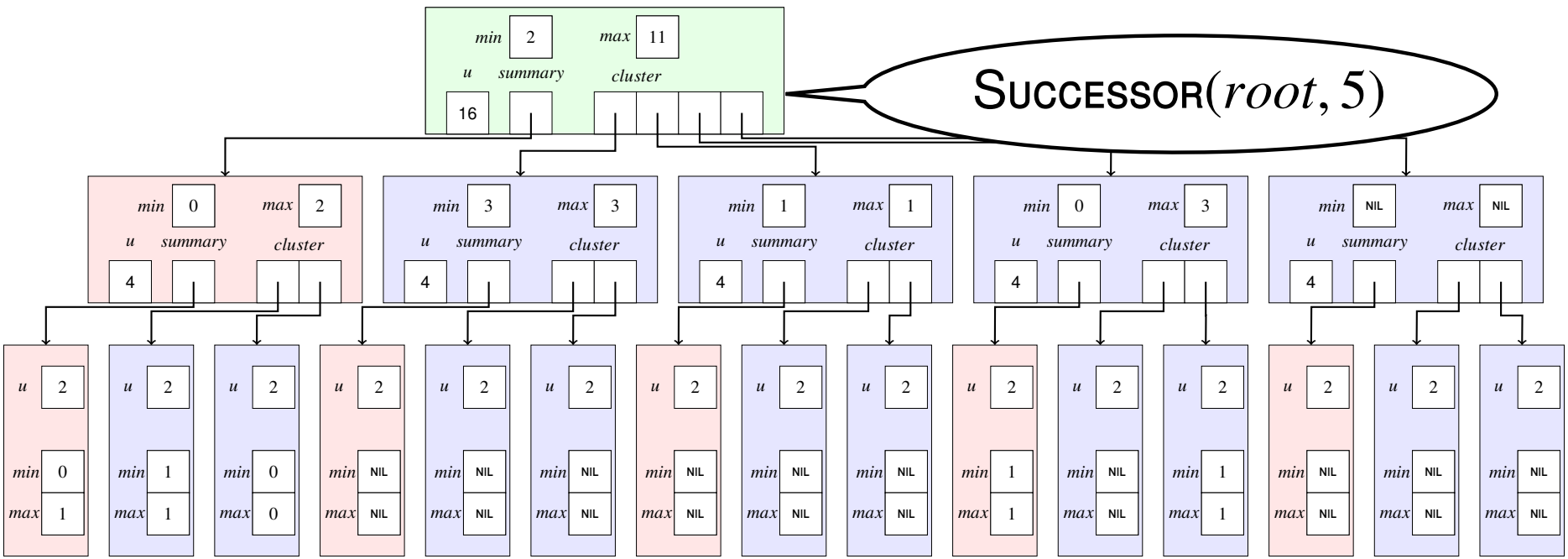
min, max が x ならば TRUE を返す.
そうでないならば, MEMBER($cluster$[high(x)], low(x)]) を返す.



MEMBER($V, 5$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: Successor(V, x)

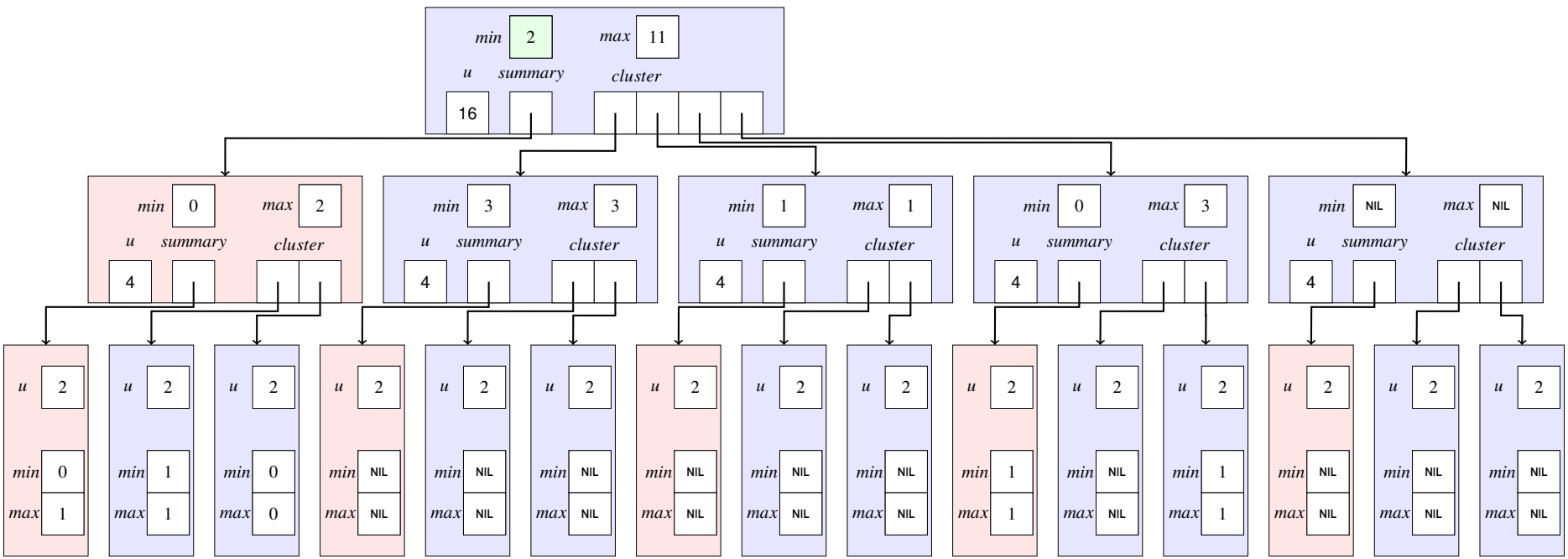
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

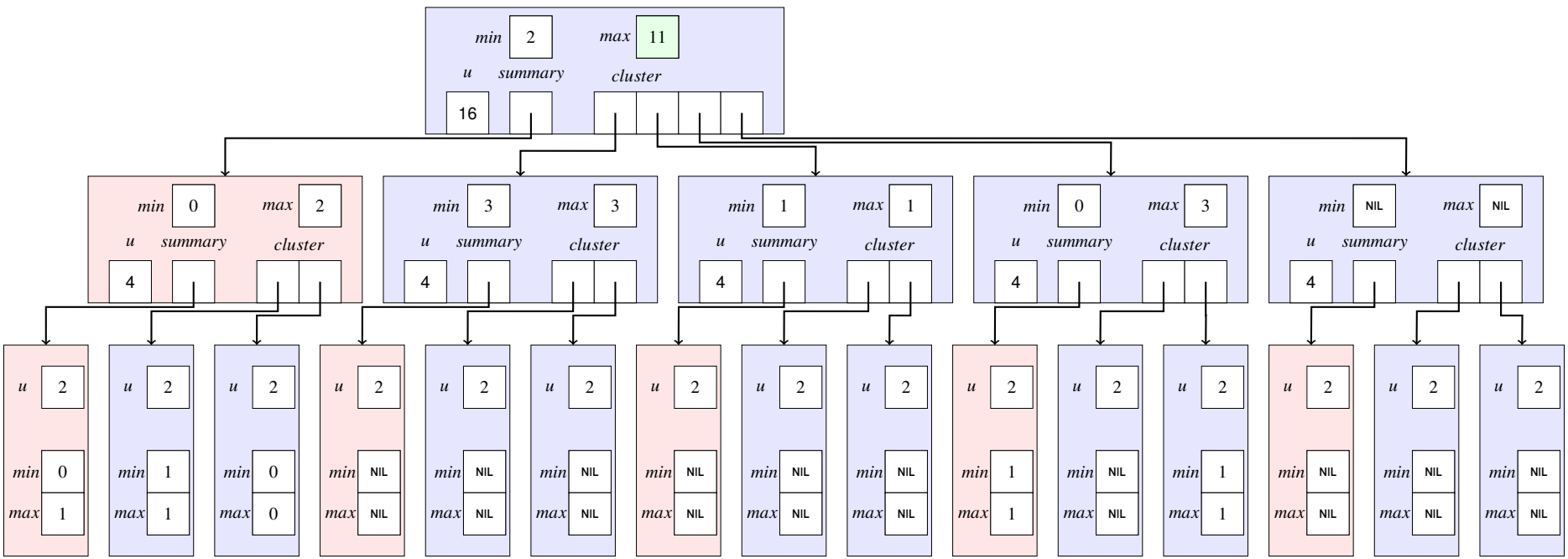
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

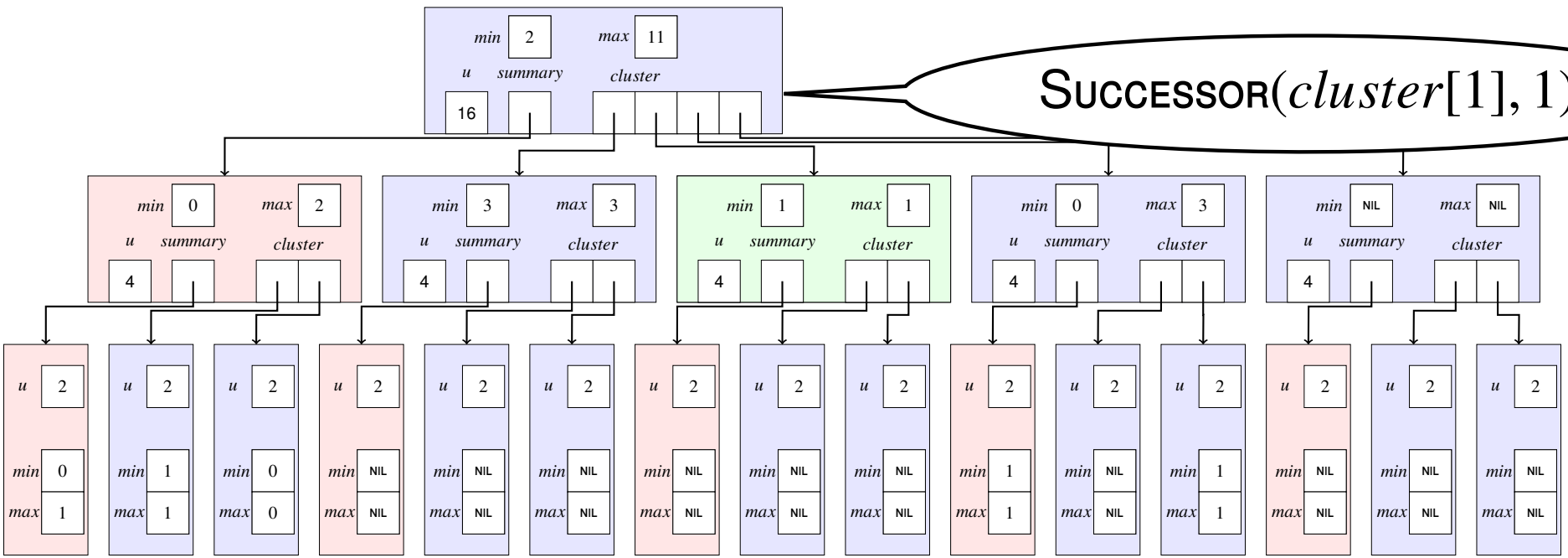
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

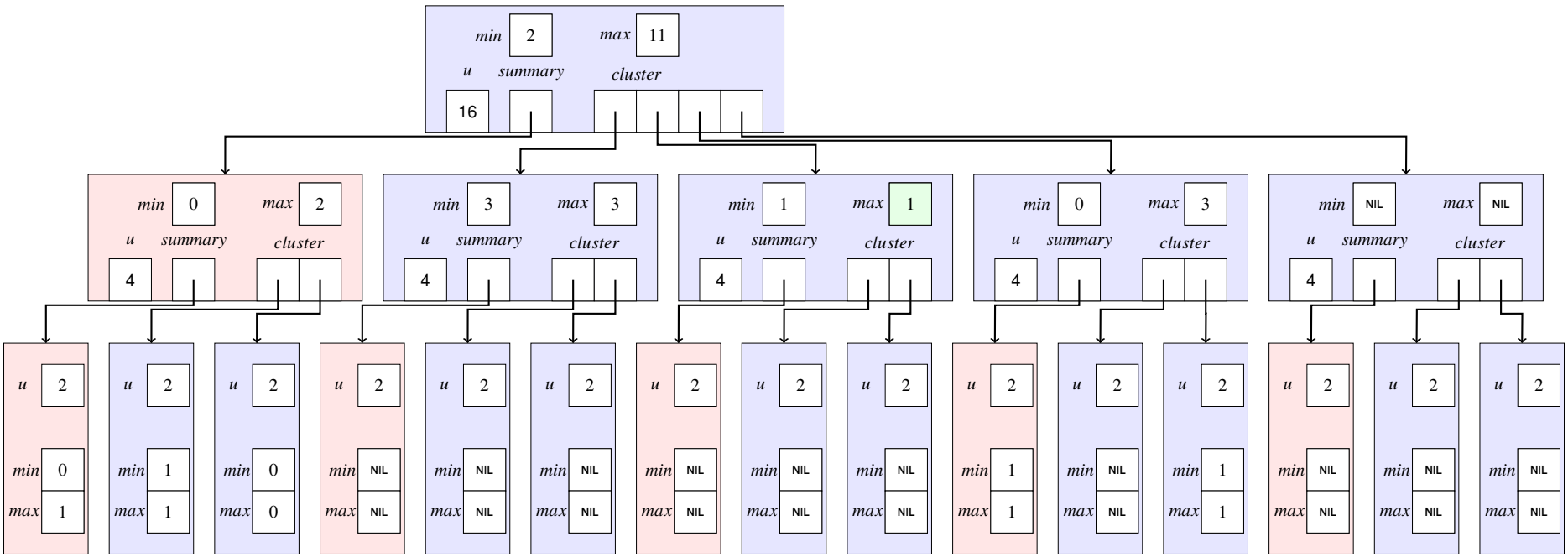
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

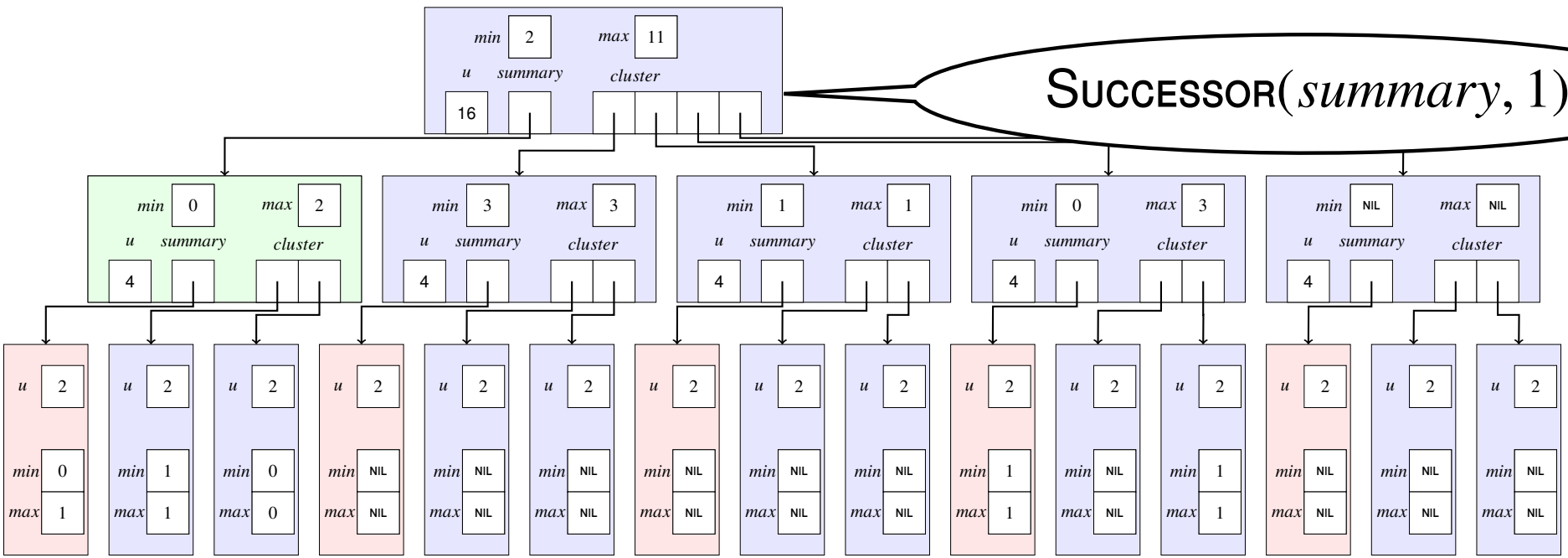
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

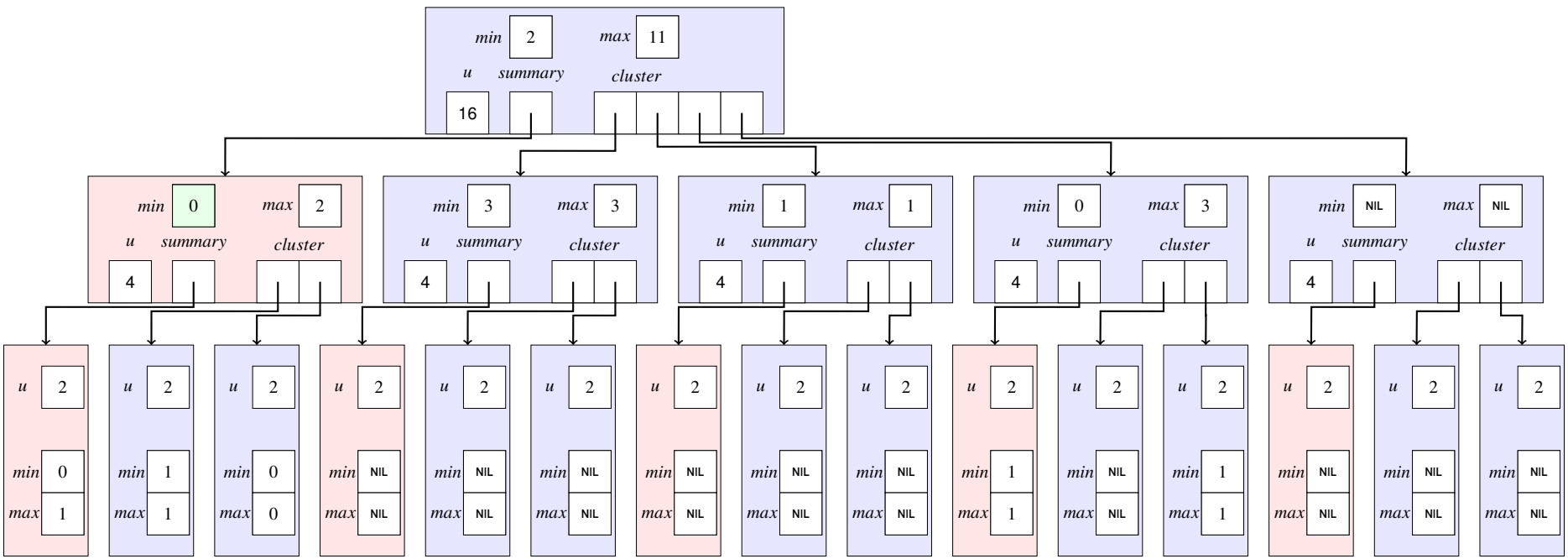
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

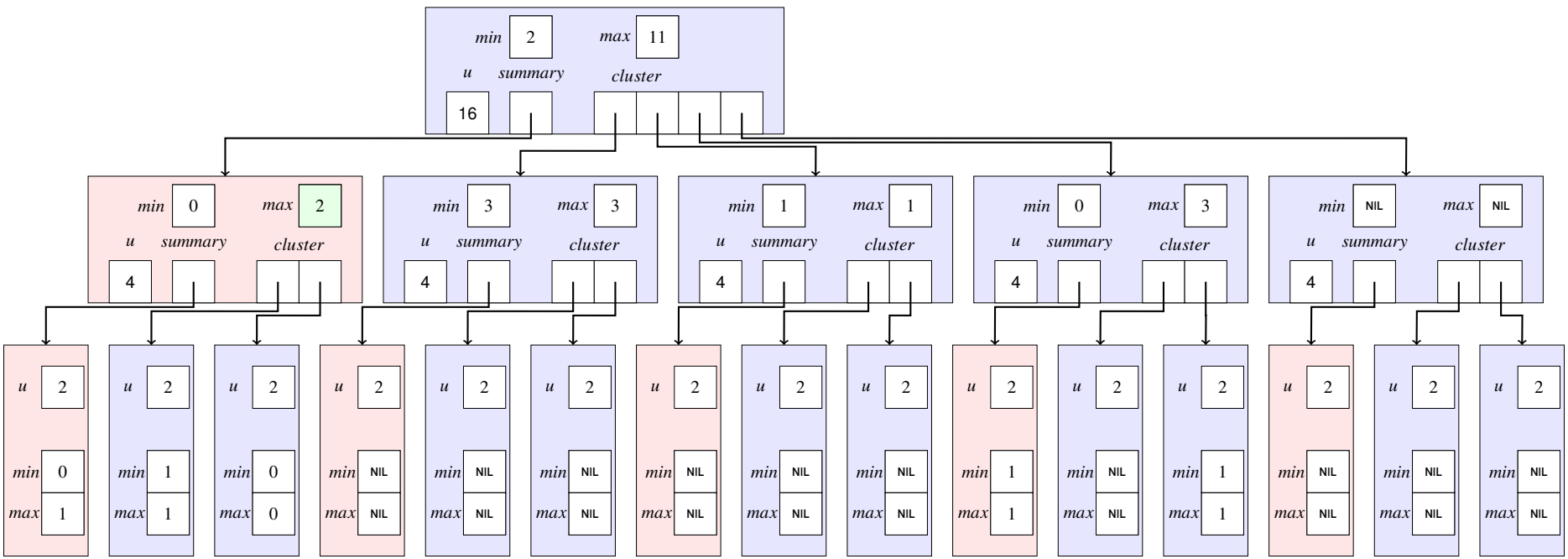
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

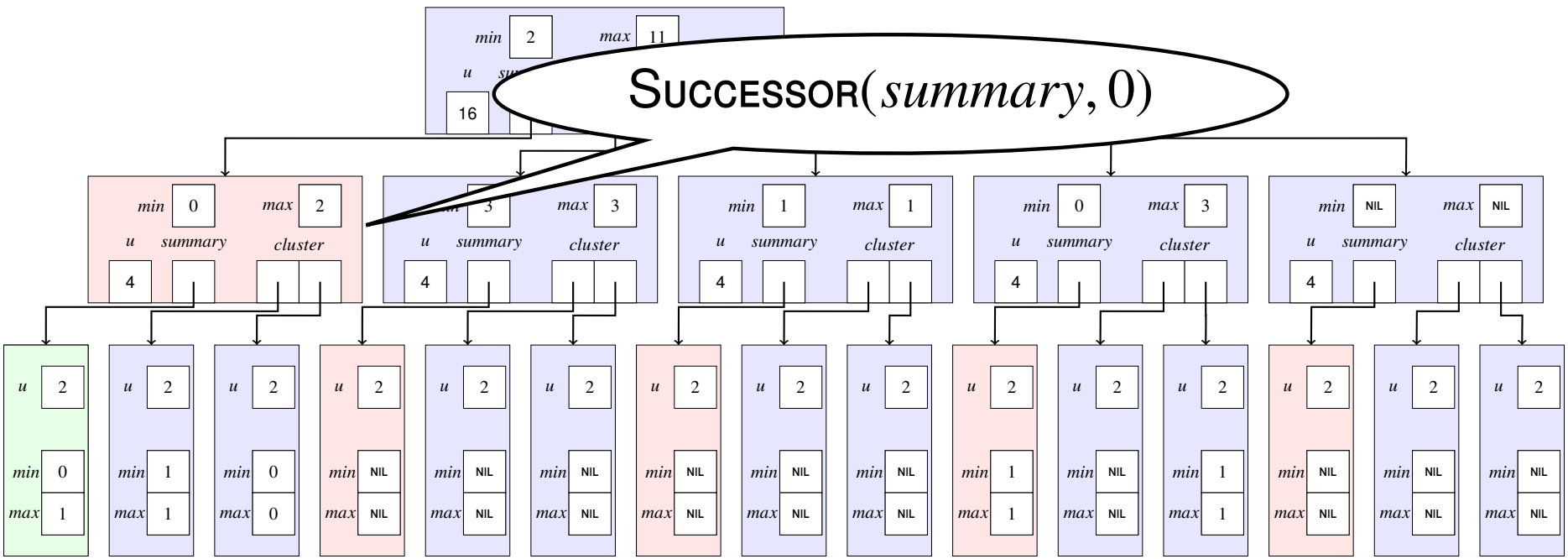
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

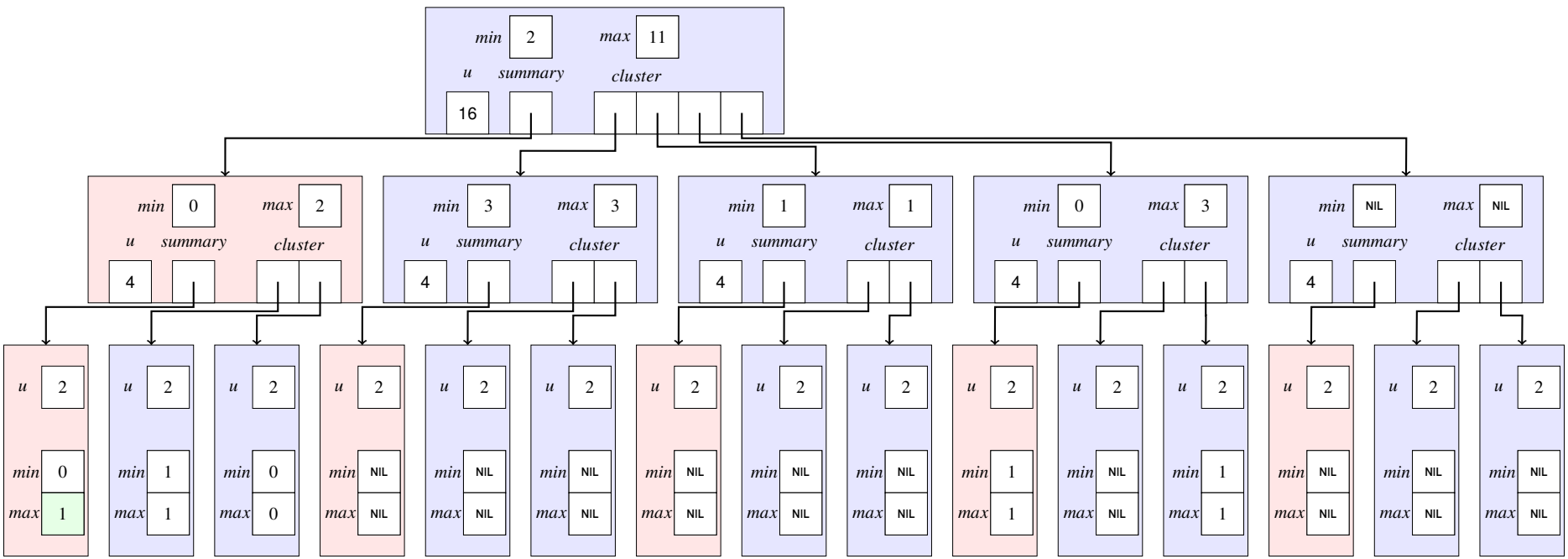
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

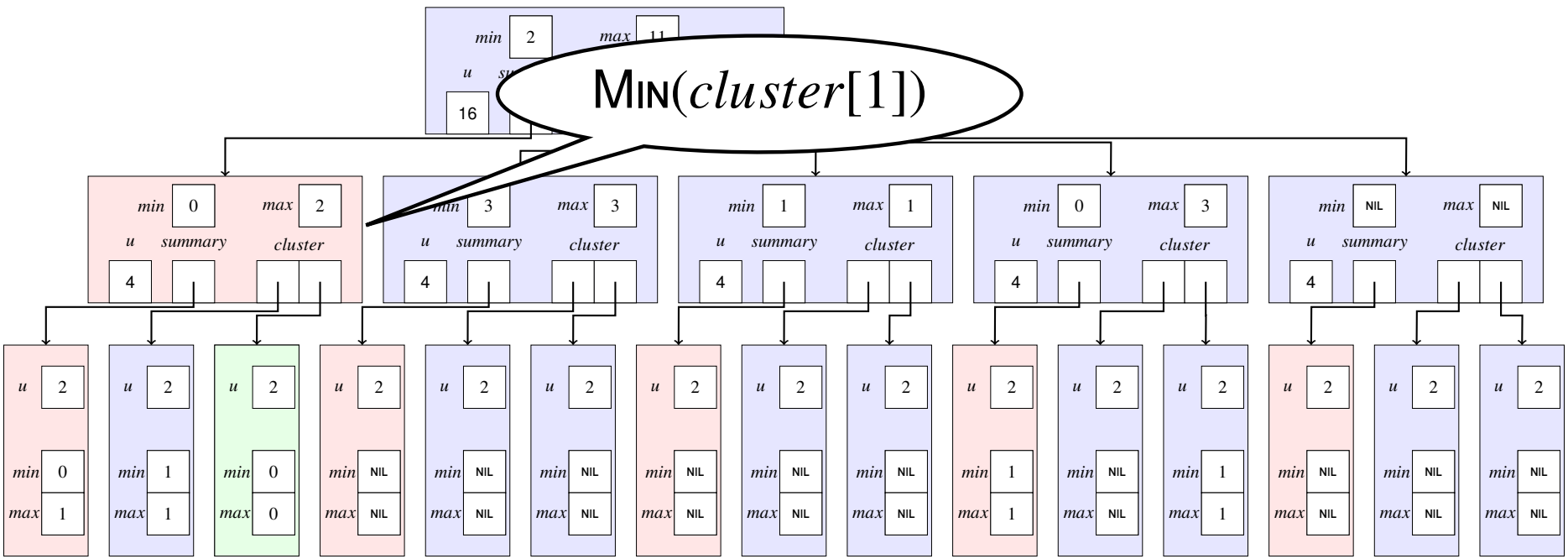
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

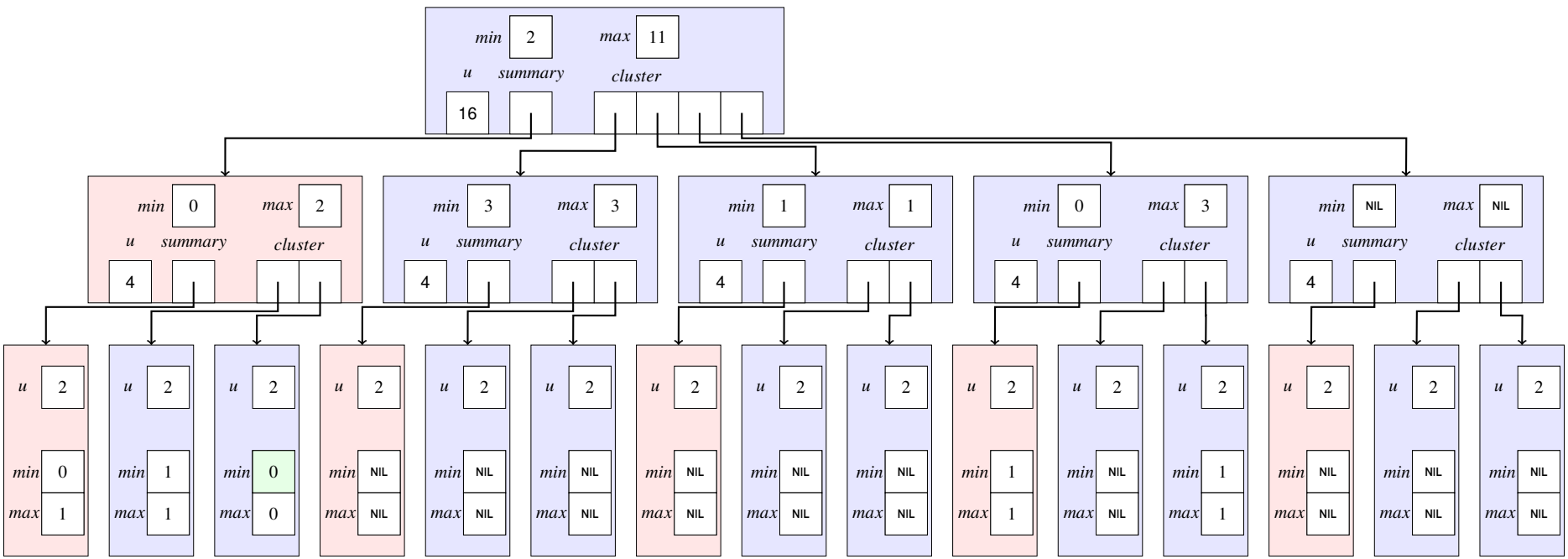
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

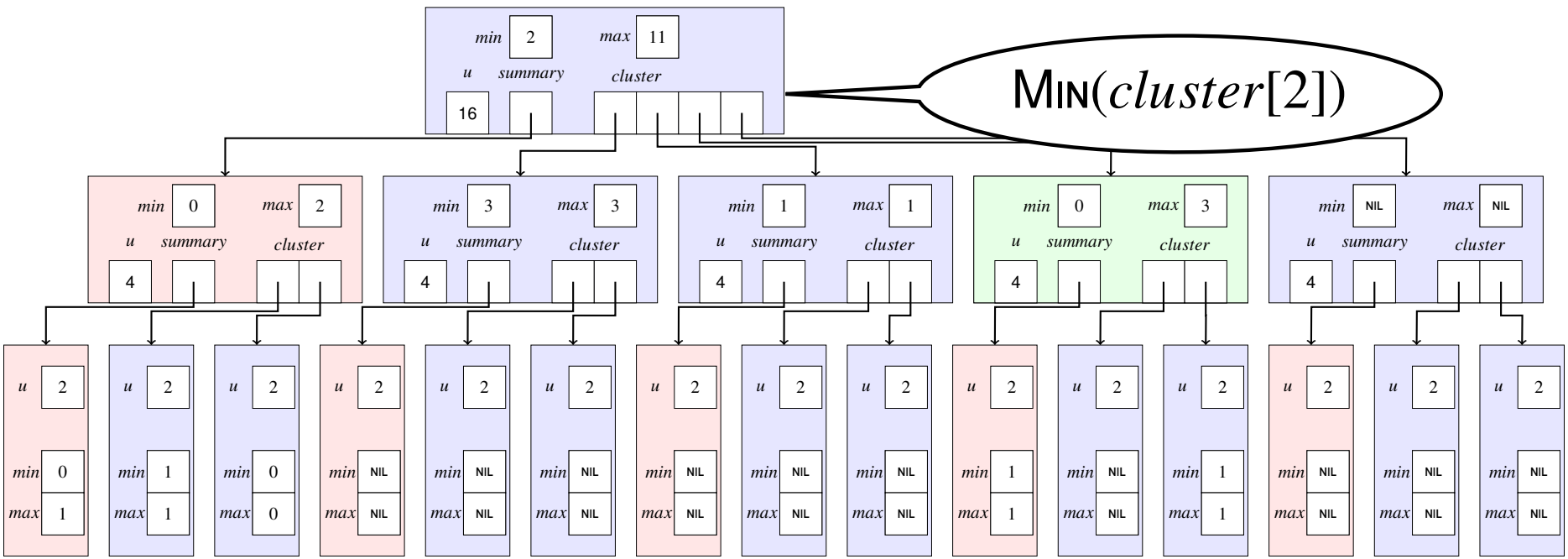
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

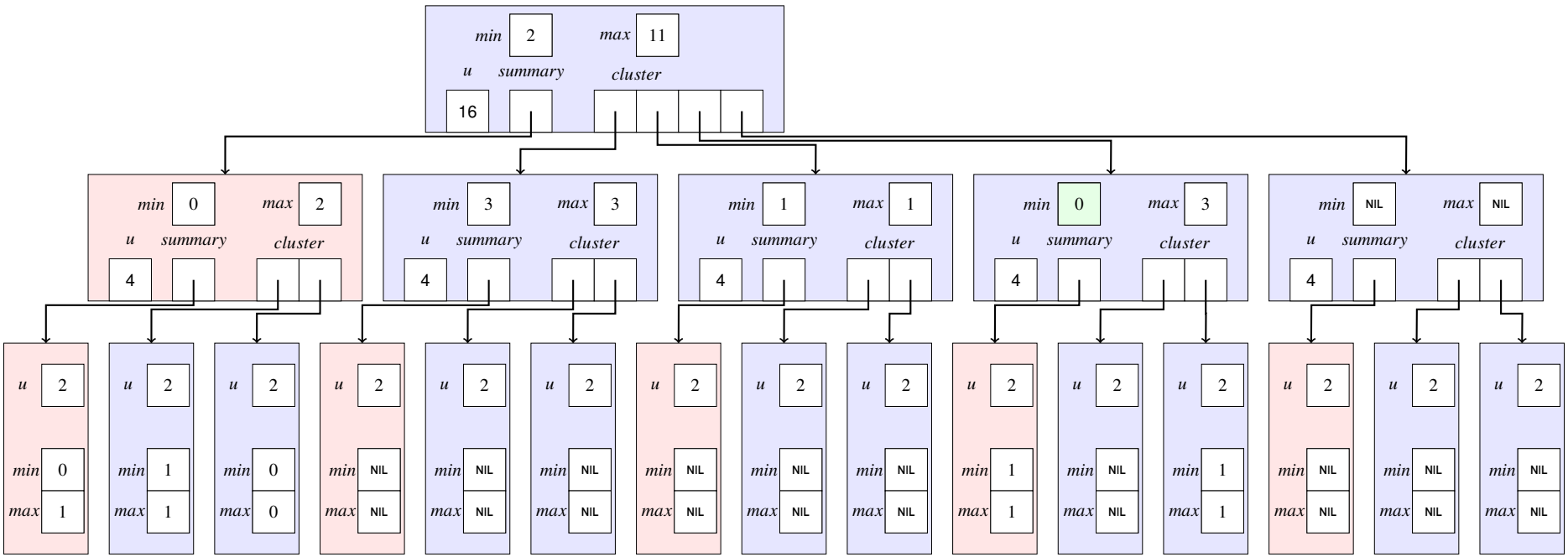
min が x 以上ならば, min を返す.
対応する $cluster$ の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ $cluster$ を取得し,
その $cluster$ の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: Successor(V, x)

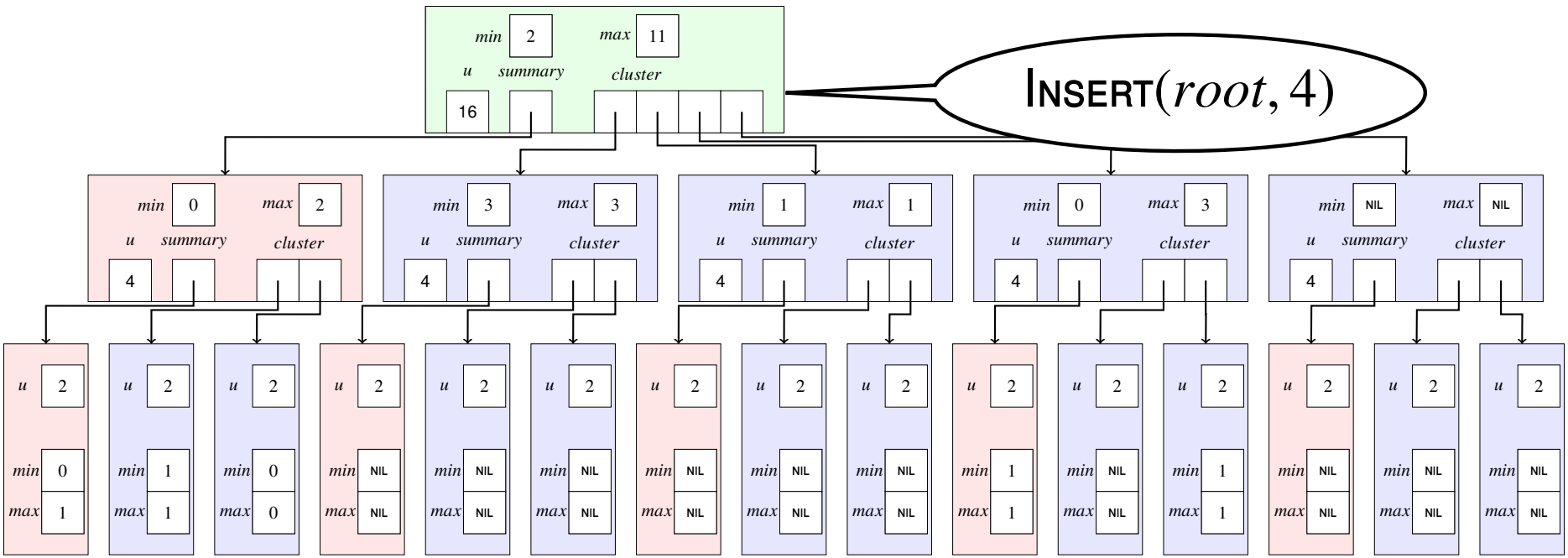
min が x 以上ならば, *min* を返す.
対応する *cluster* の最大値が $\mathrm{low}(x)$ 以下ならば,
Successor$(summary, \mathrm{high}(x))$ で次の要素を持つ *cluster* を取得し,
その *cluster* の最小値を返す.
そうでなければ, Successor$(cluster[\mathrm{high}(x)], \mathrm{low}(x))$ を返す.



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

vEB 木 操作: INSERT(V, x)

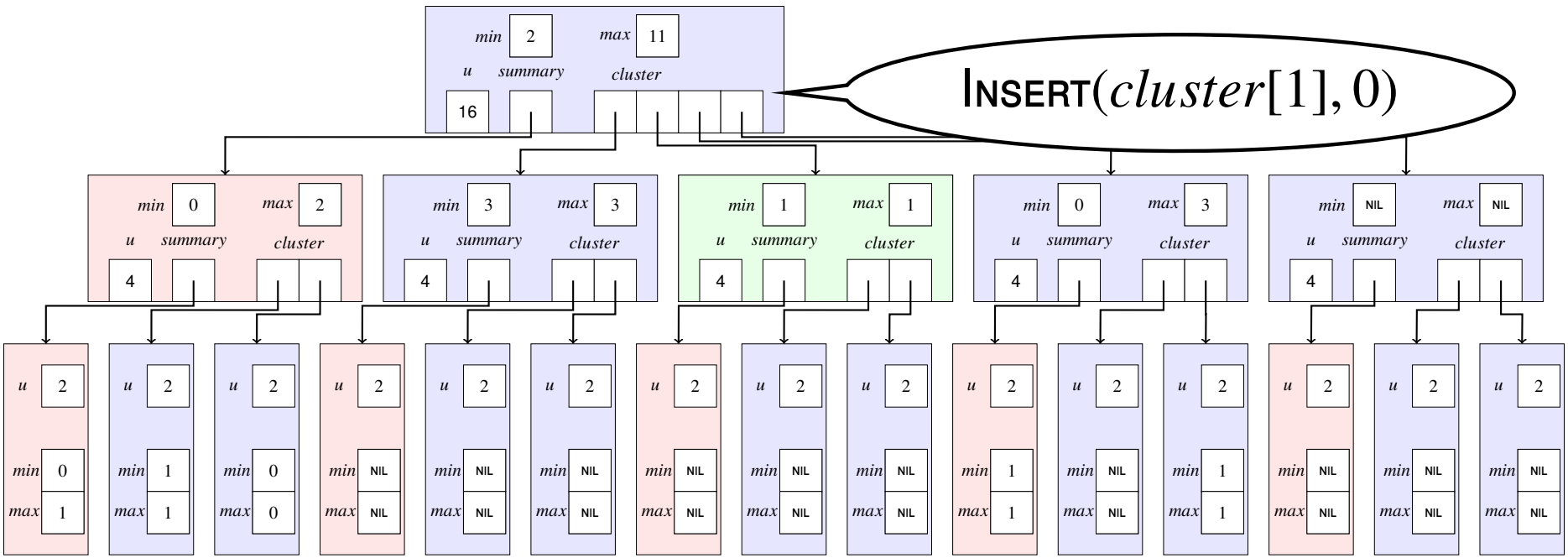
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

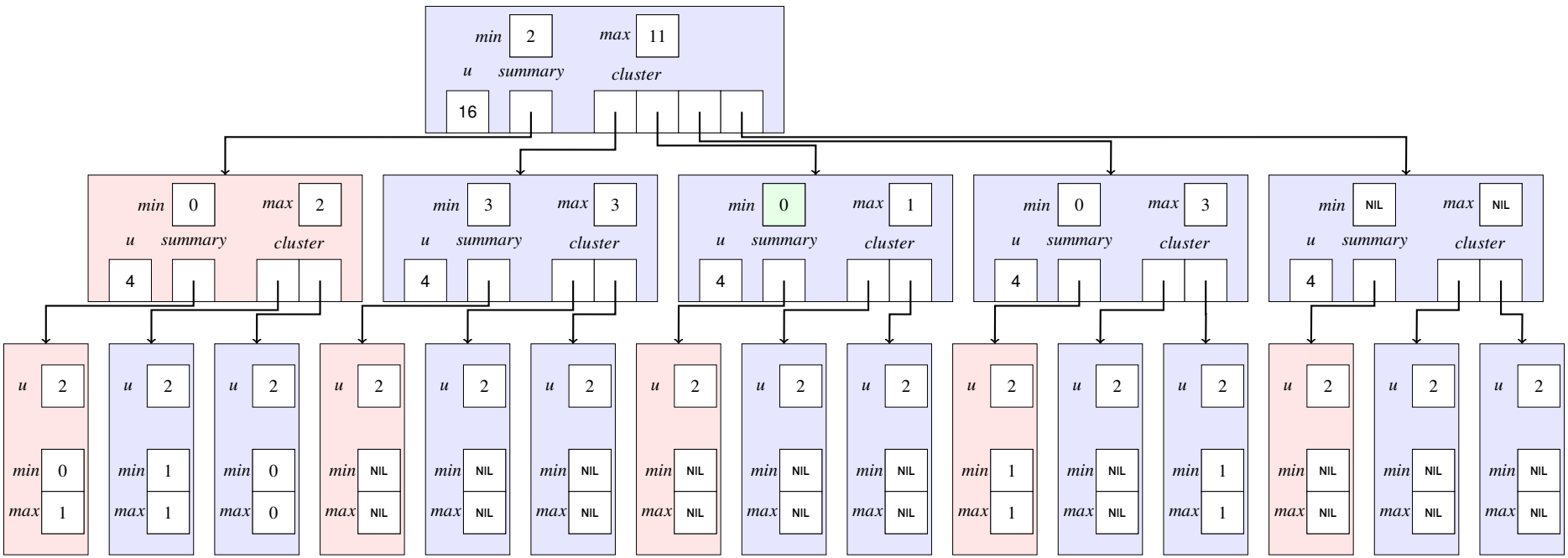
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

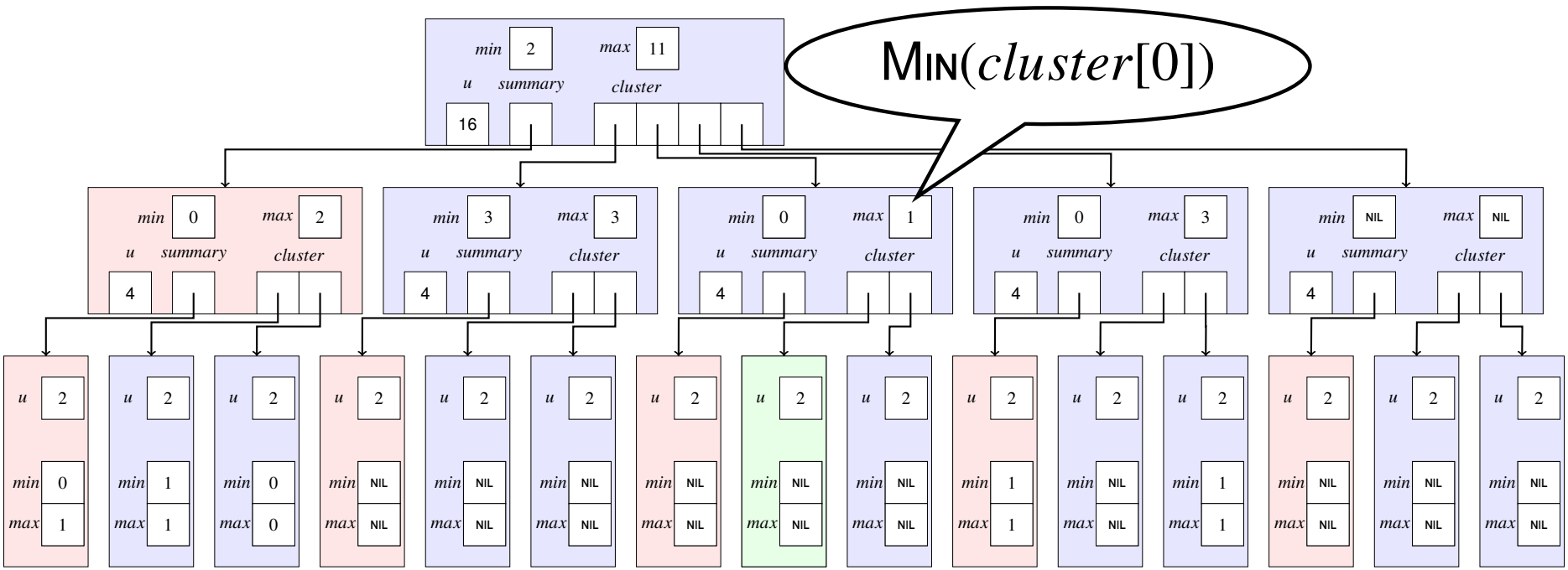
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT$(summary, \mathrm{high}(x))$ を行う.
INSERT$(cluser[\mathrm{high}(x)], \mathrm{low}(x))$ を実行し終了する.



INSERT$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

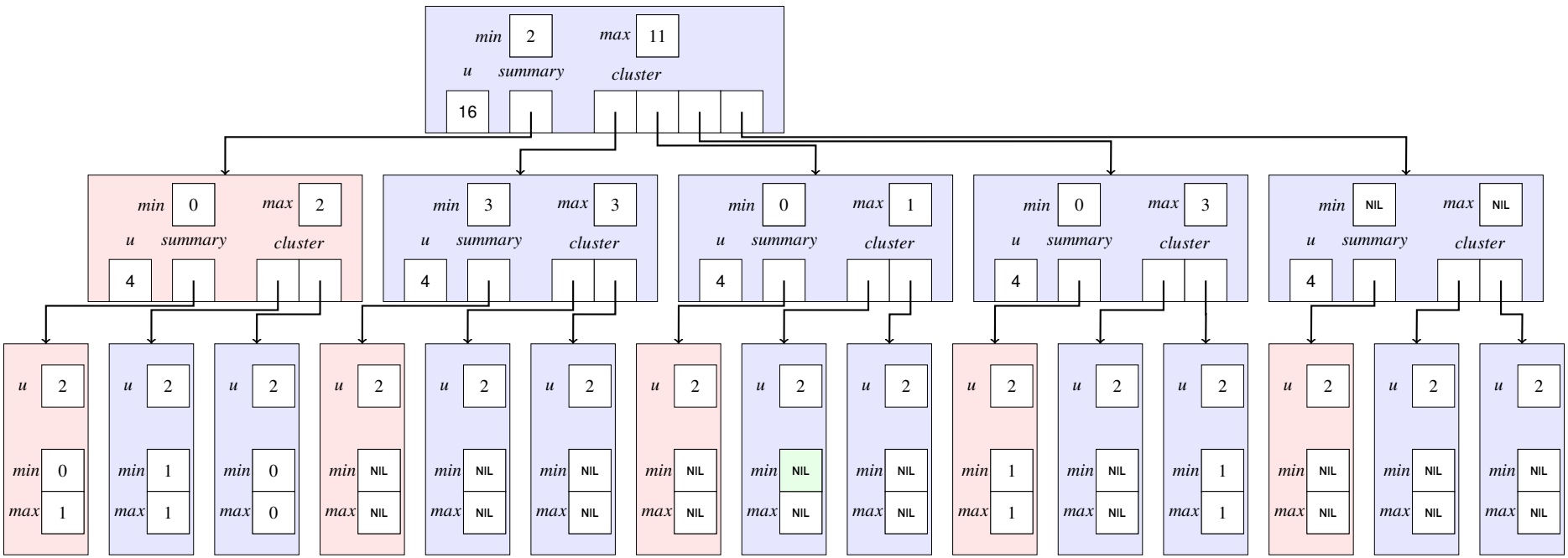
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

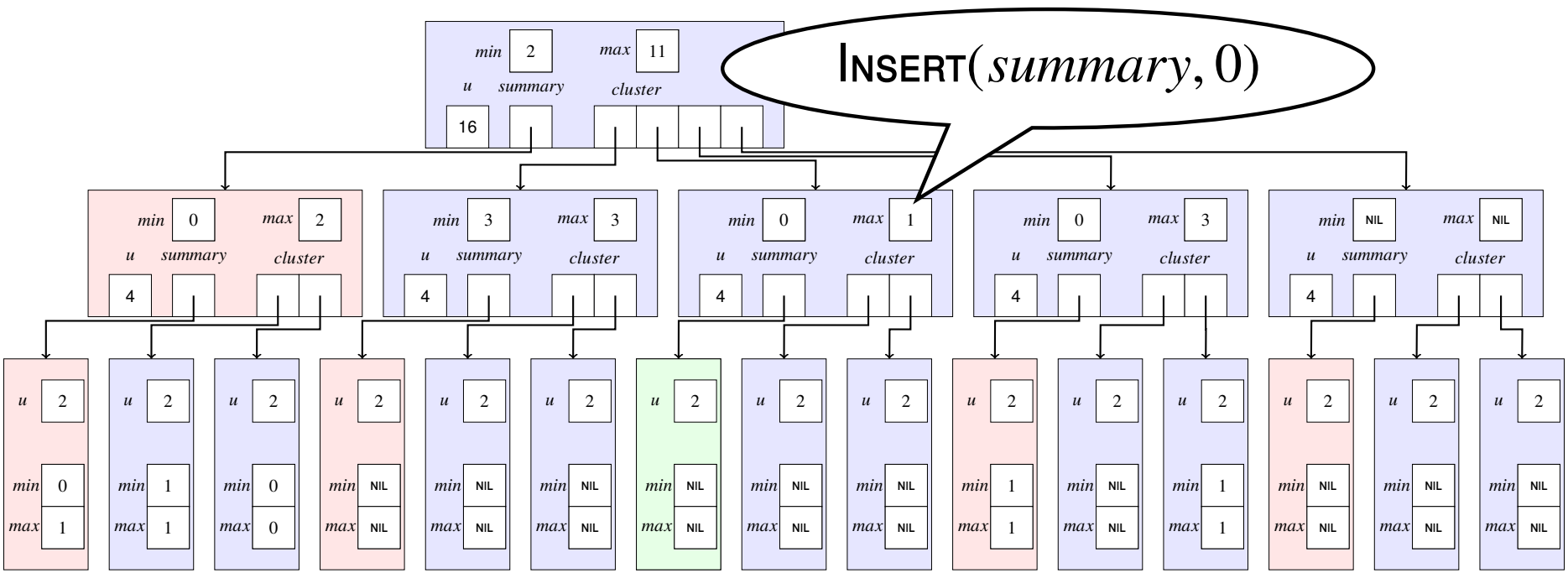
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

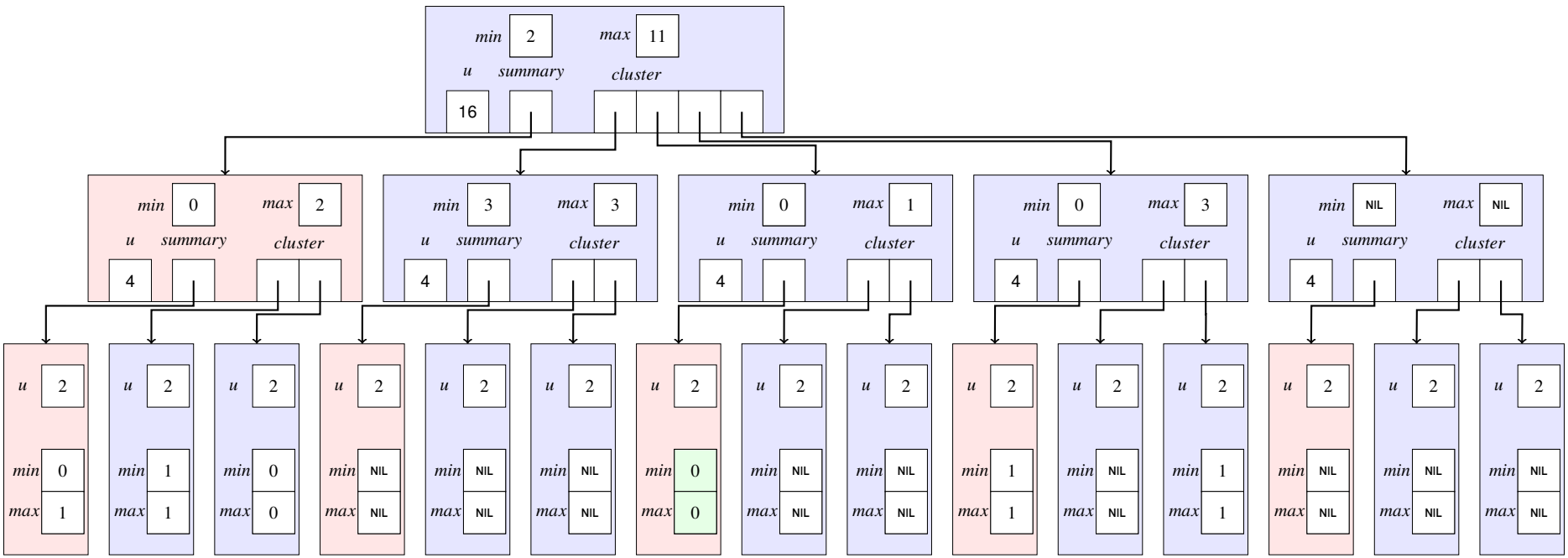
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

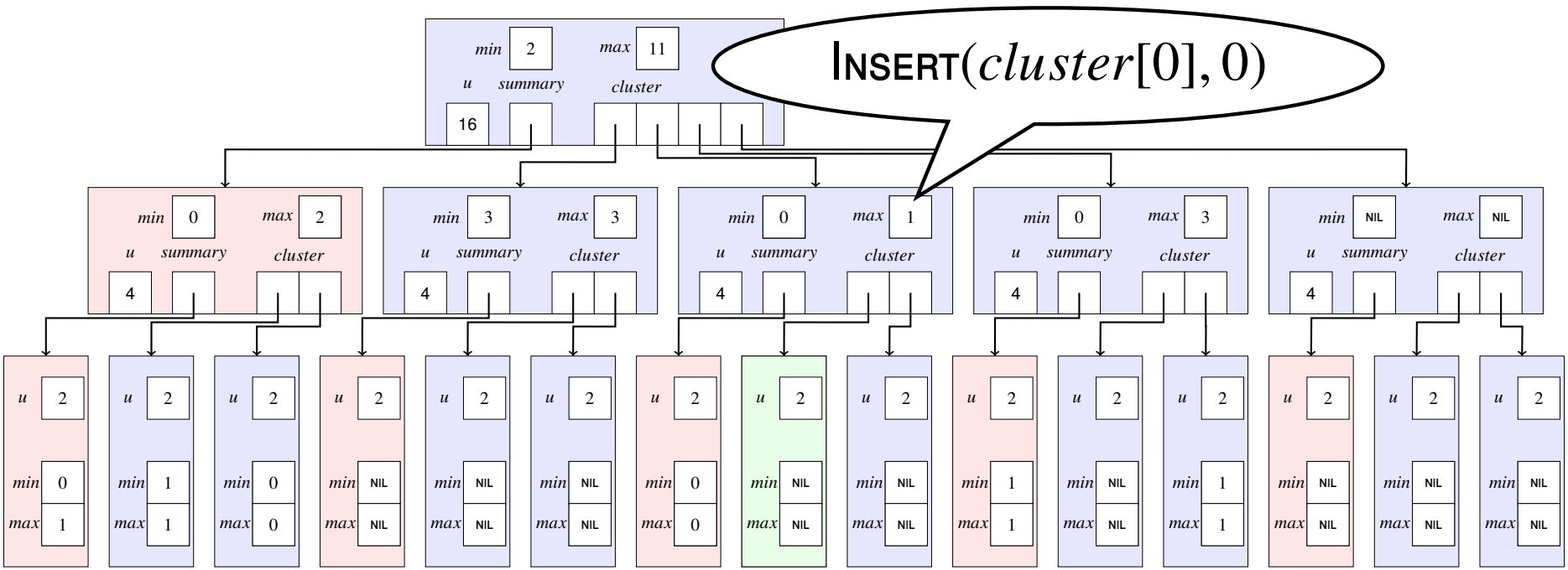
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT$(summary, \mathrm{high}(x))$ を行う.
INSERT$(cluser[\mathrm{high}(x)], \mathrm{low}(x))$ を実行し終了する.



INSERT$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

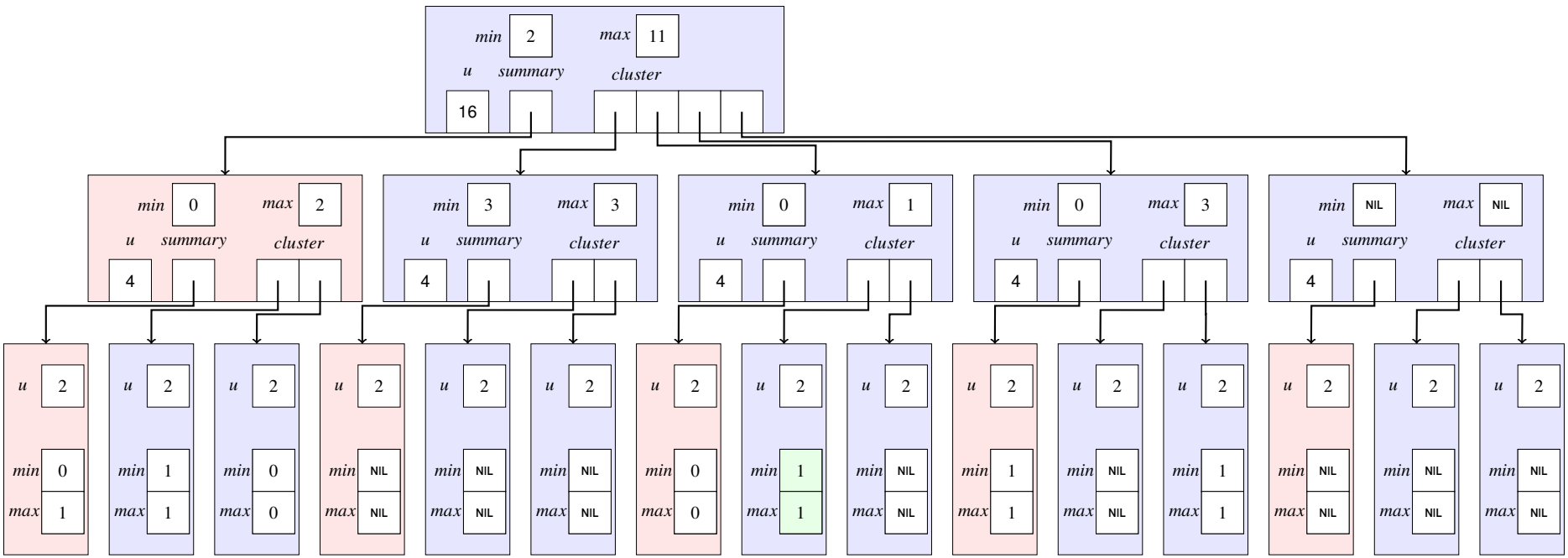
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT$(summary, \mathrm{high}(x))$ を行う.
INSERT$(cluser[\mathrm{high}(x)], \mathrm{low}(x))$ を実行し終了する.



INSERT$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: INSERT(V, x)

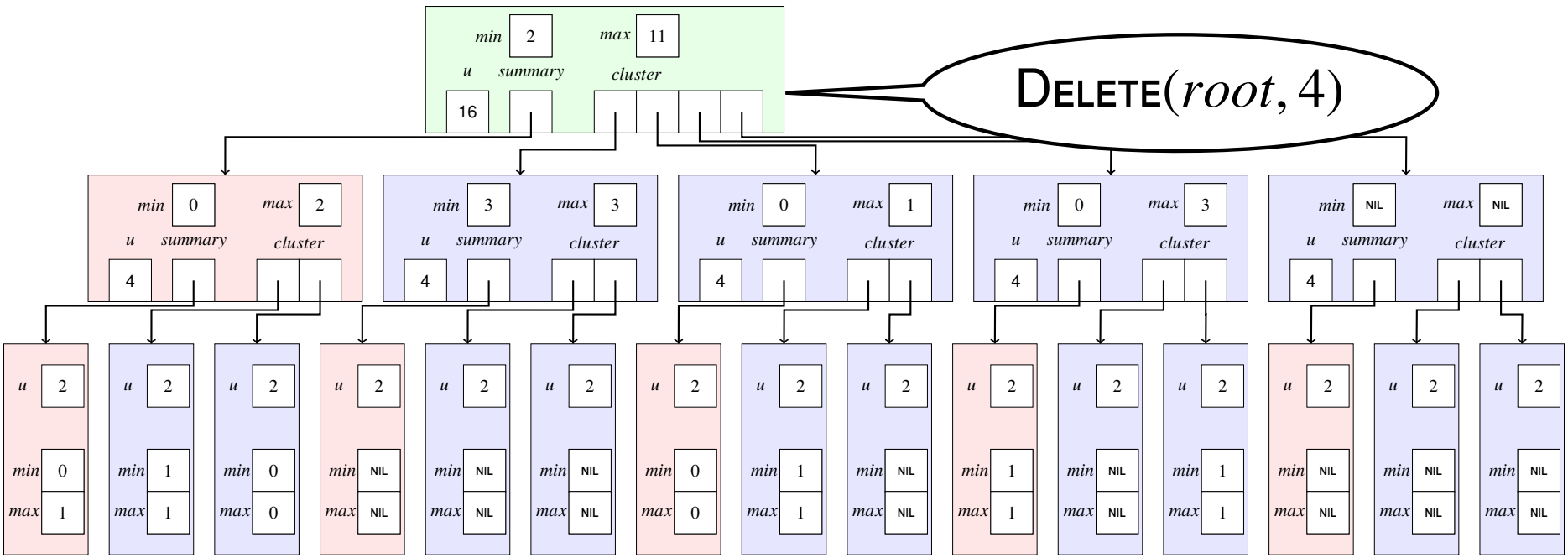
min が NIL の場合, min と max を x に更新して終了する.
x が min より小さければ x と min を交換し,
今後 min であった値について更新する.
対応する $cluster$ の最小値が NIL の場合, INSERT($summary$, high(x)) を行う.
INSERT($cluser$[high(x)], low(x)) を実行し終了する.



INSERT($V, 4$), $V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: Delete(V, x)

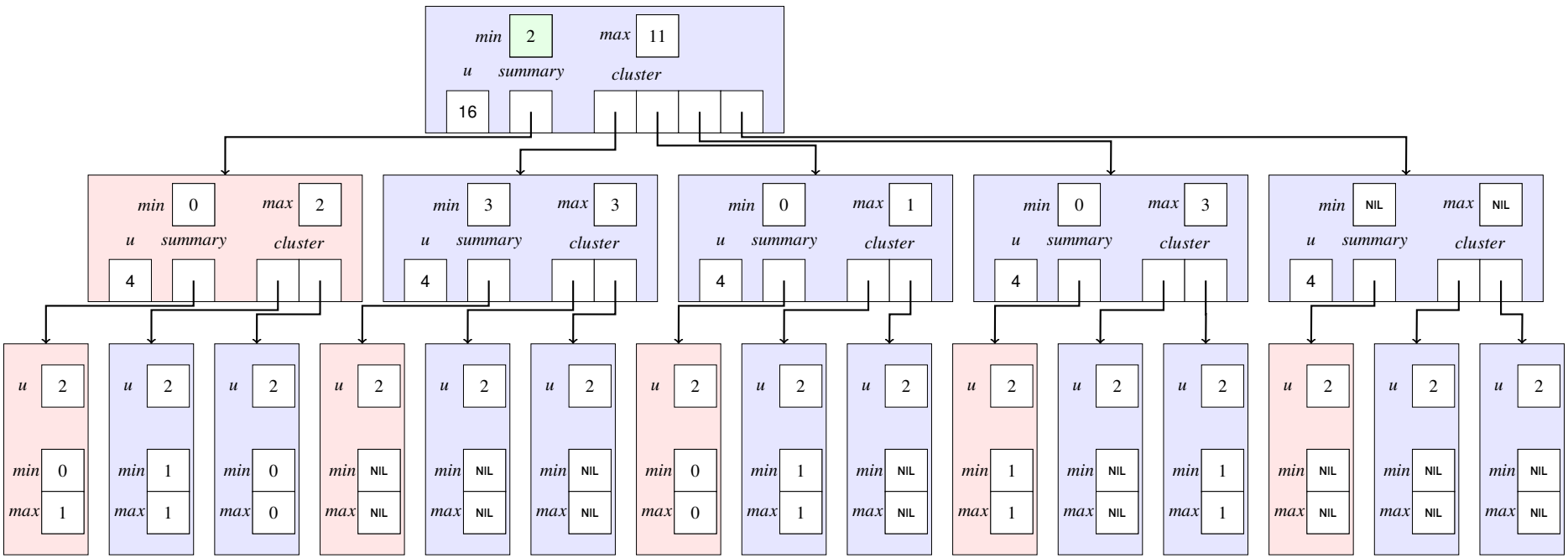
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
Delete$(summary, \mathrm{high}(x))$ を処理する.



Delete$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: DELETE(V, x)

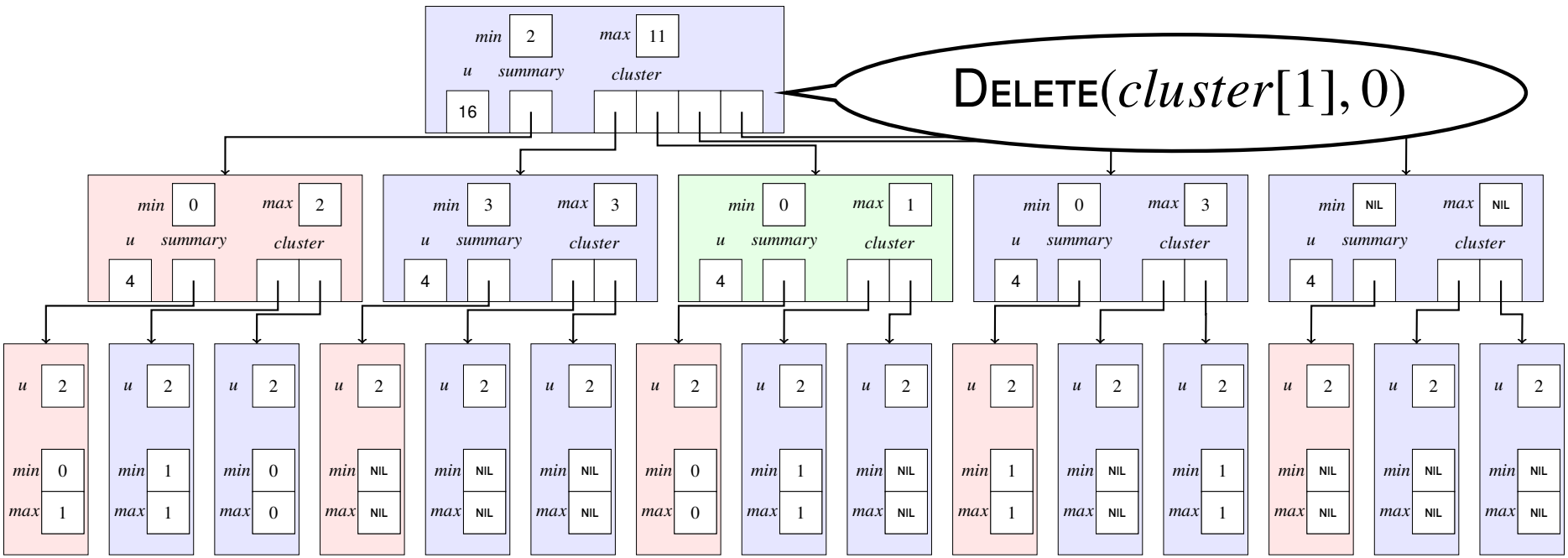
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
DELETE$(summary, \mathrm{high}(x))$ を処理する.



DELETE$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: Delete(V, x)

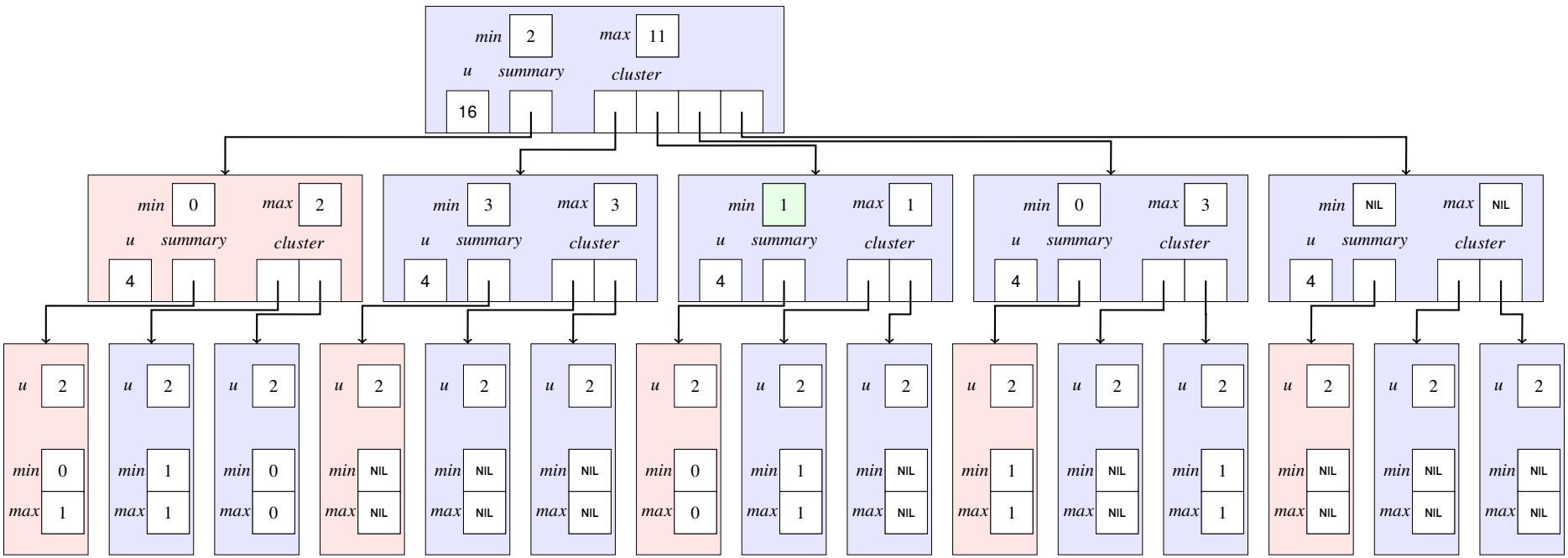
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
Delete$(summary, \mathrm{high}(x))$ を処理する.



Delete$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: Delete(V, x)

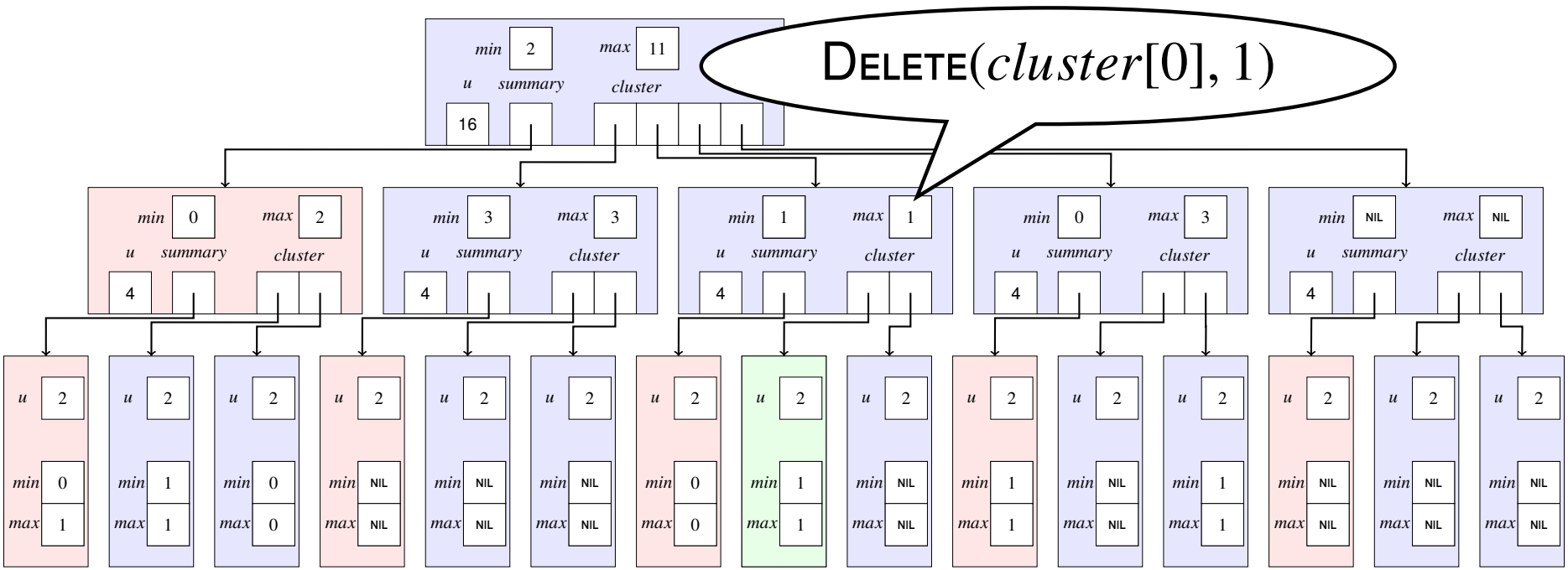
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
Delete$(summary, \mathrm{high}(x))$ を処理する.



Delete$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: DELETE(V, x)

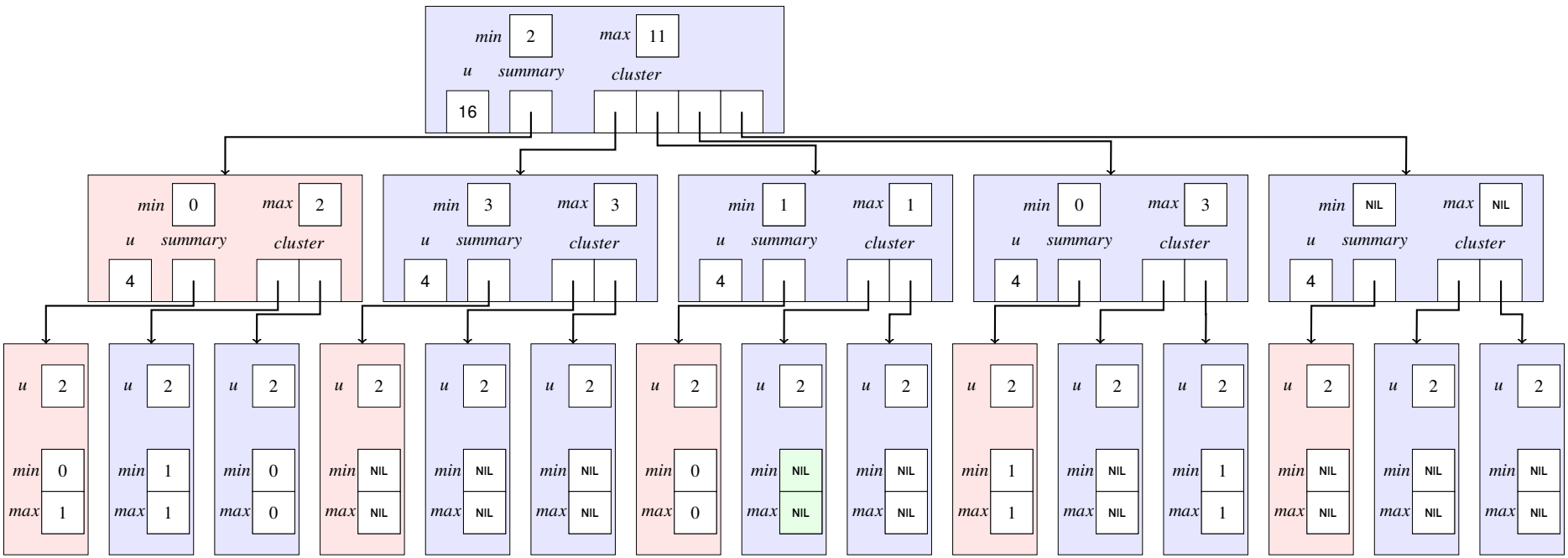
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
DELETE$(summary, \mathrm{high}(x))$ を処理する.



DELETE$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: Delete(V, x)

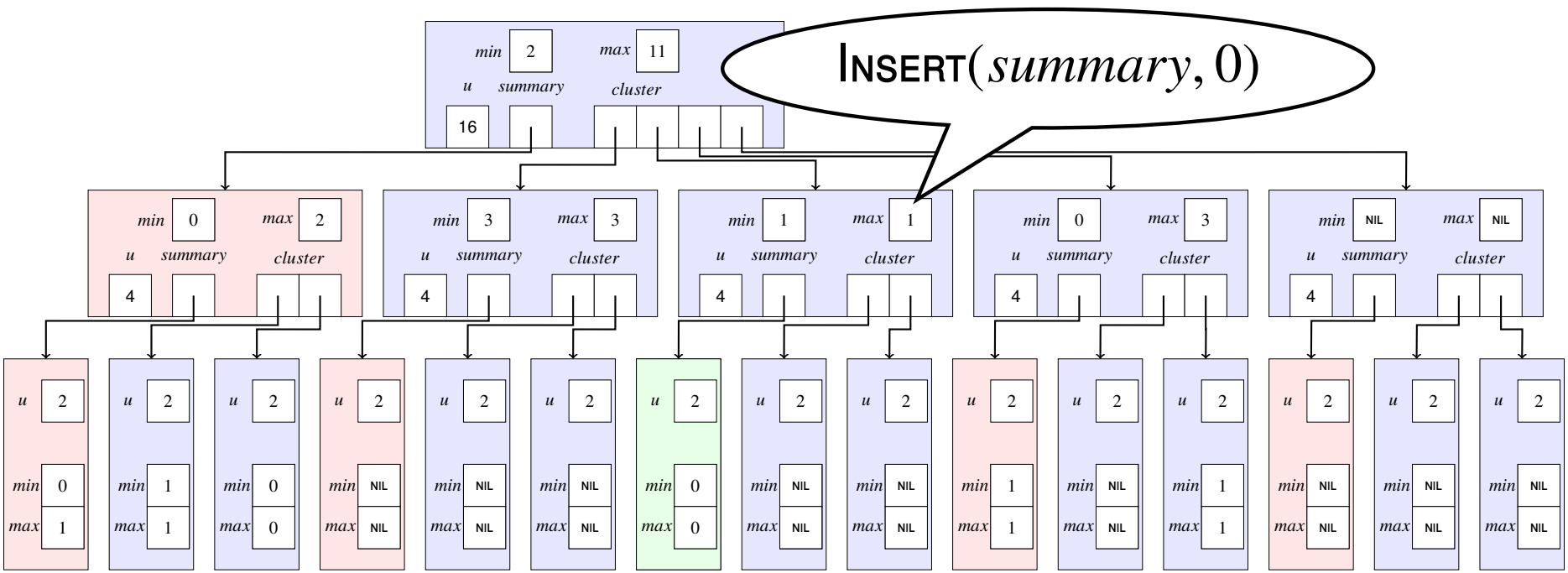
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
Delete$(summary, \mathrm{high}(x))$ を処理する.



Delete$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: DELETE(V, x)

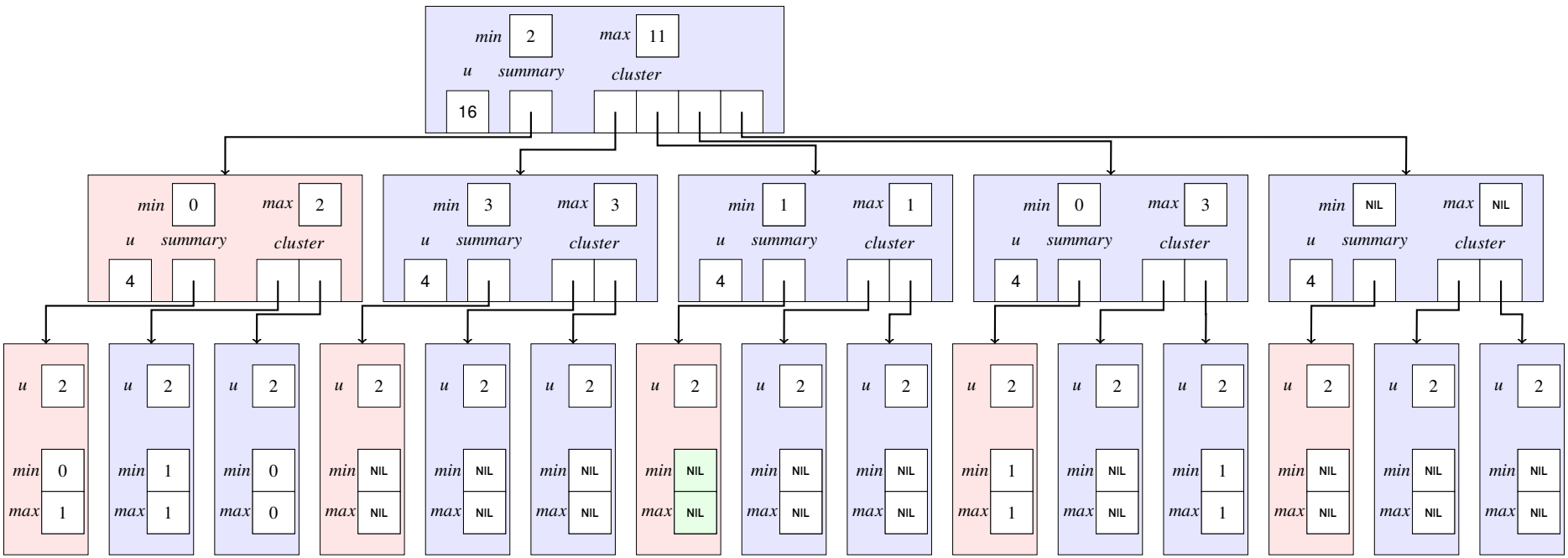
min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
DELETE$(summary, \mathrm{high}(x))$ を処理する.



DELETE$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 操作: DELETE(V, x)

min と max が x の場合, どちらも NIL にして終了する.
$u = 2$ の場合, min, max を他方の値に更新する.
$x = min$ の場合, x,min を $cluster$ 全体の最小値にし,
以降更新された x の値で処理する.
対応する $cluster$ に対し削除を行い, その $cluster$ の min が NIL の場合,
DELETE$(summary, \mathrm{high}(x))$ を処理する.



DELETE$(V, 4), V = \{2, 3, 5, 8, 11\}$

vEB 木 計算量 (1/3)

Member(V, x)

関数内での再帰呼び出しは 1 回以下なので, 時間計算量は
$O(\log \log u)$

Min(V)

各 vEB ノードは min, max の値を保持しているので,
時間計算量は $O(1)$

Successor(V, x)

関数内では, 再帰呼び出しは常に 1 回以下となる.
また, Min(V) の呼び出しを行うが, これは定数時間で処理可能なので,
Successor(V, x) の時間計算量は $O(\log \log u)$

vEB 木 計算量 (2/3)

INSERT(V, x)

INSERT(V, x) は *summary*, *cluster* それぞれに
再帰呼び出しをすることがあるが,
summary に対して挿入処理を行うとき, *cluster* への挿入処理は,
その *cluster* の *min*, *max* を更新しか行わず,
定数時間で処理可能である.
よって, INSERT(V, x) の時間計算量は $O(\log \log u)$

vEB 木 計算量 (3/3)

DELETE(V, x)

DELETE(V, x) も, INSERT(V, x) と同様に,
summary, *cluster* に対し再帰呼び出しをすることがあるが,
summary に対して, 削除を行うとき, *cluster* への削除処理は,
その *cluster* の *min*, *max* を NIL に置き換えるだけである.
よって, DELETE(V, x) の時間計算量は $O(\log \log u)$

まとめ

- 平方分割木から, 再帰的な構造である pvEB 構造を考えた.
- pvEB 構造の各操作の時間計算量は, 以下の通りとなった.

操作	時間計算量	操作	時間計算量
MEMBER(V, x)	$O(\log \log u)$	SUCCESSOR(V, x)	$O(\log u \log \log u)$
MIN(V, x)	$O(\log u)$	PREDECESSOR(V, x)	$O(\log u \log \log u)$
MAX(V, x)	$O(\log u)$	INSERT(V, x)	$\Theta(\log u)$
		DELETE(V, x)	$O(\sqrt{u} \log \log u)$

- pvEB 構造に min, max の変数を持たせた vEB 木を考えた.
- MIN(V, x), MAX(V, x) が定数時間で処理可能となったことから, 他の操作の時間計算量は $O(\log \log u)$ を達成した.