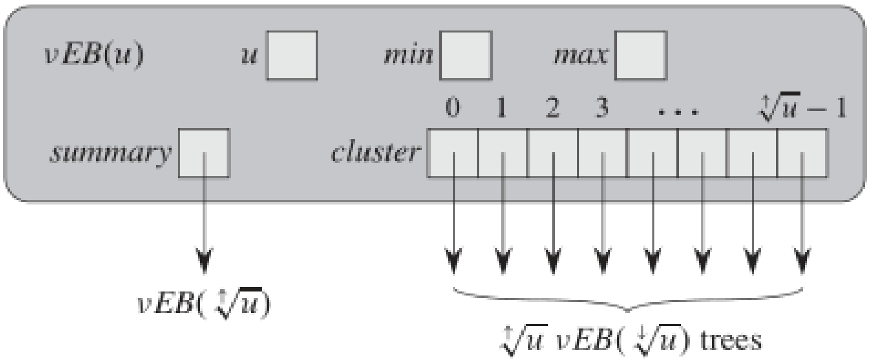
3. VAN EMDE BOAS TREE

3.1 Introduction

The van Emde Boas tree modifies proto-vEB structure. We denote a vEB tree with a universe size of u as vEB(u) and, unless u is equal the base size of 2, the attribute summary points to vEB($\sqrt{u}$) tree and the array cluster[0,1....$\sqrt{u}$-1] points to $\sqrt{u}$ vEB($\sqrt{u}$). The following figure shows the basic node structure of vEB tree.



The attributes of node is as follows:

clusters: There are $\sqrt{u}$ clusters C0, . . . , C$\sqrt{u}$−1 , where each cluster Cc is a van-Emde-Boas node and stores a set S 0 c ⊆ [$\sqrt{u}$] over the reduced universe [$\sqrt{u}$].

summary: The summary stores the clusters Cc with at least one element by the identifier c, i.e. c ∈ summary iff |Cc| ≥ 1. The summary itself is a van-Emde-Boas node over the reduced universe [$\sqrt{u}$].

min/max: min and max are respectively the minimum and maximum element of the set S. They will never be stored in any of the clusters below the node u and are the base case for the recursion. As long as |S| ≤ 2 the node will not initialize the clusters or the summary.

3.2 Opeartions

min(x) / max(x): We store the minimum and maximum in the attributes min and max, two of operations are one-liners, taking constant time.

- *Time Complexity* : O(1)

search(x): The procedure search(x) has a recursive call. We check directly whether x equals the minimum or maximum element. Since a vEB tree doesn't store bits as a proto-vEB structure does, we design search(x) to return *true* or *false* rather than (1,0).

- *Time Complexity* : O(loglogu)

successor(x) and predecessor(x): With help of summary and max, min we have to make only one recursive call for searching successor or predecessor for any key. This reduces the time complexitiy from O(logu) to O(loglogu). If the element doesn't have successor or predecessor than it returns -1.

- *Time Complexity* : O(loglogu)

insert(x): When we insert an element, either the cluster that it goes into already has another element or it does not. If the cluster already has another element, then the cluster number is already in the summary, and so we do not need to make that recursive call. If the cluster does not already have another element, then the element being inserted becomes the only element in the cluster, and we do not need to recurse to insert an element into an empty vEB tree

- *Time Complexity* : O(loglogu)

delete(x): With help of min, max and summary we search the key to be deleted in only one recursive call. While recursion we handle different cases.

- *Time Complexity* : O(loglogu)

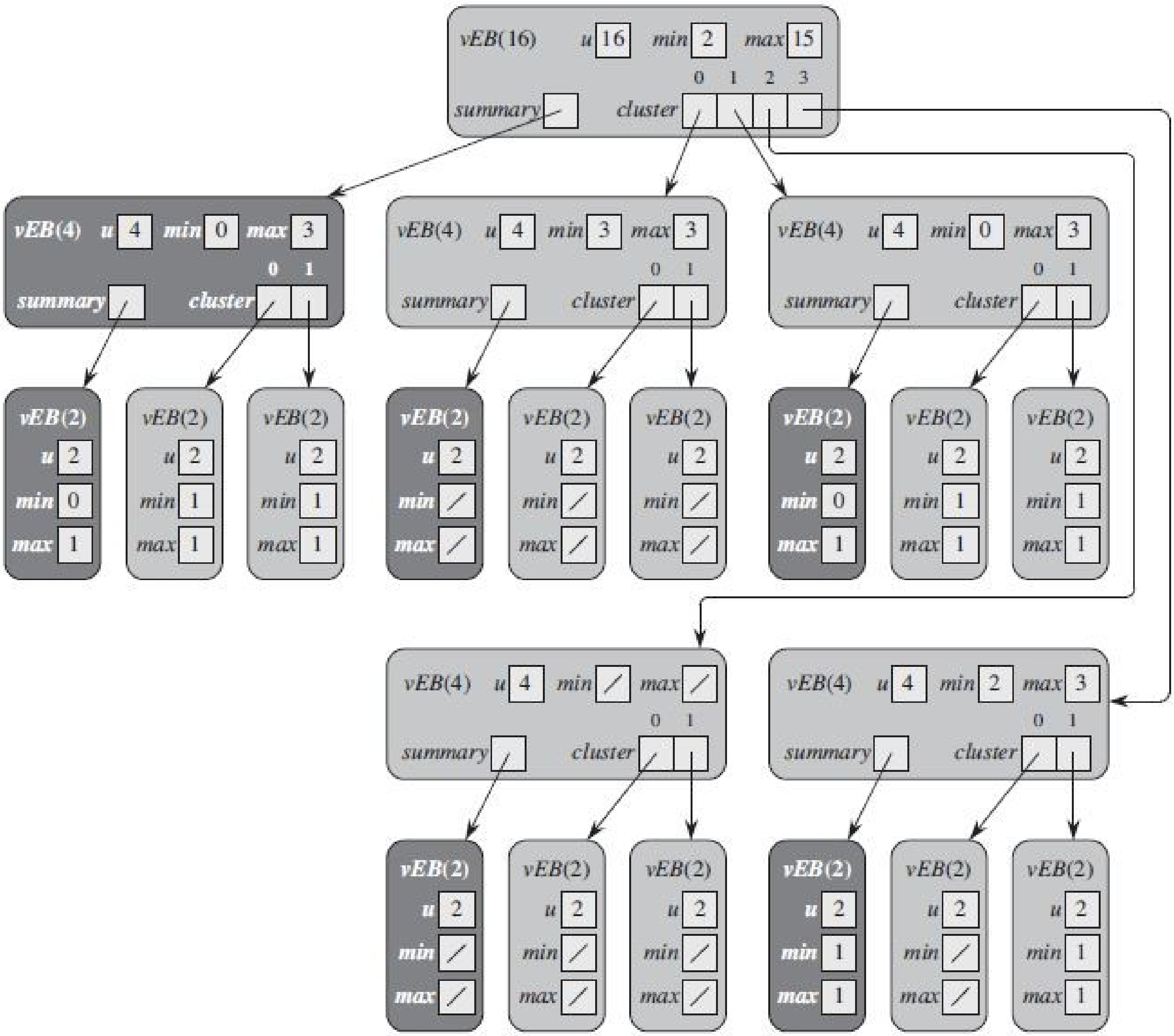


Figure 3: shows vEB tree of size 16 containing elements [2,3,4,5,7,14,15]