Problem Set 3 Solutions

This problem set is due **at 11:59pm** on **Thursday, February 26, 2015.**

Exercise 3-1. Read CLRS, Section 20.3.

Exercise 3-2. Exercise 20-3.1.

Exercise 3-3. Exercise 20-3.2.

Problem 3-1. Variants on van Emde Boas [25 points]

For each of the following variants on the van Emde Boas data structures (presented in Lecture 4 and CLRS, Section 20.3), carefully describe what changes are needed to the pseudocode (from either lecture or the textbook), and analyze the costs of the vEB operations INSERT, DELETE, and SUCCESSOR, comparing them with the costs of the same operations for the original vEB structure.

(a) [7 points] Instead of dividing the structure into $u^{1/2}$ groups of $u^{1/2}$ numbers each, use $u^{1/3}$ groups of $u^{2/3}$ numbers each.

Solution: The pseudocode doesn't change, except for the different division into clusters. The operations MIN and MAX take constant time.
For MEMBER, we have the recurrence

$$T(u) = T(u^{2/3}) + c.$$

Previously, we had the recurrence $T(u) = T(u^{1/2}) + c$, which solved (see CLRS) to

$$T(u) = O(c \lg \lg u) = O(c \log_2 \log_2 u).$$

For the new recurrence, following the same solution method, we get

$$O(c \log_{3/2} \log_2 u),$$

which is the same order of magnitude, with a slightly larger constant.
For SUCCESSOR, PREDECESSOR, INSERT, and DELETE, we get the recurrence

$$T(u) = \max\{T(u^{1/3}), T(u^{2/3})\} + c = T(u^{2/3}) + c.$$

So we have the same analysis as for MEMBER, yielding

$$O(c \log_{3/2} \log_2 u) = O(\lg \lg u).$$

(b) [18 points] In addition to excluding the *minimum* element from lower-level vEB structures, also exclude the *maximum* element from lower-level vEB structures (and store it in the already existing max attribute). (Use the original division into $u^{1/2}$ groups of $u^{1/2}$ numbers here.)

Solution: We rewrite portions of the code in CLRS Section 20.3 that involve min and max. The order-of-magnitude of the complexity does not change.

The initialization of the empty data structure is as before. For MIN and MAX queries, the code is unchanged. Because we are treating MIN and MAX symmetrically, SUCCESSOR and PREDECESSOR are now symmetric with each other.

SUCCESSOR: Start with the code on p. 551. Lines 1-11 and 14-15 are unchanged. Lines 12-13 must be modified, however, to take into account the case where the successor may reside in no cluster at all; this is similar to lines 13-14 of the old predecessor code on p. 552. Thus, in place of the current lines 12-13, we write:

```
vEB-TREE-SUCCESSOR(V, x)
▷ This replaces lines 12-13 in the original code.
if succ-cluster = NIL
    if V.max ≠ NIL and x < V.max
        return V.max
    else return NIL
```

PREDECESSOR: Now predecessor is symmetric with successor. In fact, the predecessor code stays the same as the code on page 552.

INSERT: The modified code is as follows:

```
ONE-ELEMENT-TREE-INSERT(V, x)
▷ This should be called when V.min = V.max
if x > V.min
    V.max = x
else V.min = x
```

```
vEB-TREE-INSERT(V, x)
1  if V.min == NIL
2      vEB-EMPTY-TREE-INSERT(V,x)
3  elseif V.min == V.max
4      ONE-ELEMENT-TREE-INSERT(V,x)
5  else
6      if x < V.min
7          exchange x with V.min
8      elseif x > V.max
9          exchange x with V.max
10     if vEB-TREE-MINIMUM(V.cluster[high(x)]) == NIL
11         vEB-TREE-INSERT(V.summary,high(x))
12         vEB-EMPTY-TREE-INSERT(V.cluster[high(x)],low(x))
13     else
14         vEB-TREE-INSERT(V.cluster[high(x)],low(x))
```

DELETE: The modified code is as follows.

```
vEB-TREE-DELETE(V, x)
1  if V.min == V.max
2      V.min = NIL
3      V.max = NIL
4  elseif vEB-TREE-MINIMUM(V.summary) == NIL
5      if x = V.min
6          V.min = V.max
7      elseif x = V.max
8          V.max = V.min
9  else
10     if x == V.min
11         first-cluster = vEB-TREE-MINIMUM(V.summary)
12         x = index(first-cluster,
13            vEB-TREE-MINIMUM(V.cluster[first-cluster]))
14         V.min = x
15     elseif x == V.max
16         last-cluster = vEB-TREE-MAXIMUM(V.summary)
17         x = index(last-cluster,
18            vEB-TREE-MAXIMUM(V.cluster[last-cluster]))
19         V.max = x
20     vEB-TREE-DELETE(V.cluster[high(x)], low(x))
21     if vEB-TREE-MINIMUM(V.cluster[high(x)], low(x)) == NIL
22         vEB-TREE-DELETE(V.summary, high(x))
```

We explain the edits to the code in CLRS p. 554. Lines 1-3 stay the same since they are simply testing the 1-element special case for V. Now we add a new 2-element special case after line 3. Note that *summary* is empty because the structure contains just the *max* and *min* and neither appears in the clusters.

vEB-TREE-DELETE(V, x)
▷ Lines 1-3 are as in the book
1 **if** $V.min == V.max$
2 $\quad V.min =$ NIL
3 $\quad V.max =$ NIL
4 **elseif** vEB-TREE-MINIMUM$(V.summary) ==$ NIL
$\quad$ ▷ This deals with the case where the summary is empty
5 $\quad$ **if** $x = V.min$
6 $\quad\quad V.min = V.max$
7 $\quad$ **elseif** $x = V.max$
8 $\quad\quad V.max = V.min$

At this point in the execution, we know that the structure contains at least 3 elements. Thus, we don't need the base case, lines 4-8, since each base structure can contain at most two elements. Starting from line 9, a lot changes, since we are treating *min* and *max* symmetrically. So we can write:

vEB-TREE-DELETE(V, x)
8 ▷ Lines 4-8 from the original precede this.
9 **else**
10 $\quad$ **if** $x == V.min$
11 $\quad\quad$ ▷ The logic from lines 10-12 in the original goes here
12 $\quad$ **elseif** $x == V.max$
13 $\quad\quad$ ▷ The logic from lines 10-12 in the original goes here
14 $\quad\quad$ ▷ but with vEB-TREE-MAXIMUM$(V.summary)$, *last-cluster*,
15 $\quad\quad$ ▷ vEB-TREE-MAXIMUM$(V.cluster[last\text{-}cluster])$,
16 $\quad\quad$ ▷ and setting $V.max = x$ in the final line

For this code, note that the clusters of V cannot all be empty because V contains at least 3 elements. So the operations above on $V.summary$ actually return values.

The net effect of these lines is to reset x so it now refers to an element to be deleted from some cluster within V. The new x may be placed in *min* or *max*, if appropriate.

After this code, keep line 13 from the original Delete code, which deletes the element from its cluster, as well as lines 14-15, which delete the cluster from the summary if necessary. We omit lines 16-23, because they address the case where we are deleting the maximum element of V, which we have already handled.