Lingfu Zhang February 22, 2015

6.046 Problem 3-1

Collaborators: *none*

(a) Based on CRLS, all changes we need to make here are redefine 'high(x)', 'low(x)' and index(x, y) as:

high(x) $= \lfloor x/2^{\lfloor (\lg u)/3 \rfloor} \rfloor$
low(x) $= x \bmod 2^{\lfloor (\lg u)/3 \rfloor}$
index(x, y) $= x \cdot 2^{\lfloor (\lg u)/3 \rfloor} + y$
and everything else is exactly the same.

Now let's consider the consider the costs of INSERT, DELETE and SUCCESSOR.
For INSERT, when the cluster x belongs to doesn't exist, we do vEB-TREE-INSERT($V.summary$, high(x)), which takes $T(2^{\lfloor (\lg u)/3 \rfloor})$ time; when the cluster x belongs to exists before, we do vEB-TREE-INSERT($V.cluster$[high(x)], low(x)), which takes $T(2^{\lceil (2\lg u)/3 \rceil})$ time. All other operations take constant time. Then we have:

$$T(u) \leq T(2^{\lceil (2\lg u)/3 \rceil}) + O(1)$$

or

$$T(2^m) \leq T(2^{\lceil 2m/3 \rceil}) + O(1) \leq T(2^{3m/4}) + O(1)$$

for all $m \geq 3$. Let $S(m) = T(2^m)$, we can get $T(u) = T(2^m) = S(m) = O(\lg m) = O(\lg \lg m)$.

For vEB-TREE-DELETE(V, x), if the cluster containing x isn't empty after vEB-TREE-DELETE($V.cluster$[high(x)], low(x)), we needn't do vEB-TREE-DELETE($V.summary$, high(x)); otherwise, $V.cluster$[high(x)] only contains x before the deletion, and vEB-TREE-DELETE($V.cluster$[high(x)], low(x)) takes constant time. So the costs is the same as INSERT.

For vEB-TREE-SUCCESSOR(V, x), we also either do vEB-TREE-SUCCESSOR($V.cluster$[high(x)], low(x)) or vEB-TREE-SUCCESSOR($V.summary$, high(x)), and some other operations with constant time. Thus the total cost is still the same as INSERT.

Although the costs is still $O(\lg \lg u)$, we may state that the costs is higher than before, as we now have $S(m) \leq S(3m/4) + O(1)$ instead of $S(m) \leq S(2m/3) + O(1)$.

(b) All the following changes are based on the pseudocode in CRLS.

For vEB-TREE-INSERT(V, x), first, check whether V is empty: if so, do vEB-EMPTY-TREE-INSERT(V, x), which keeps the same; otherwise, if $x < V.min$, swap x with $V.min$; if $x > V.max$, swap x with $V.max$. Now if $V.u > 2$, we insert x to lower level vEB structures. This is the same as line 5-9 in CRLS.

All changes we make here takes constant time. We now check whether $x > V.max$ at the beginning instead of in the end, and make swap if necessary. The analysis of costs should be the same, and total costs is still $\lg \lg u$.

For vEB-TREE-DELETE(V, x), if $V.min == V.max$, there is only one element in V, let $V.min = NIL$, $V.max = NIL$. Else, if $V.u == 2$, let $V.min = 1$ if $x == 0$ and $V.max = 0$ if $x == 1$. Now consider $V.u > 2$. If $x == V.min$, we find the next smallest value in V, and let both x and $V.min$ be that value. If $x == V.max$, we find the next largest value in V, and let both x and $V.max$ be that value. Now we delete x from lower level vEB structures and delete the cluster in $V.summary$ if it becomes empty. This is the same as line 13-15 in CLRS.

All changes made here takes constant time in every recursive call, and we deal with the situation where $x = V.max$ differently. The analysis of costs is basicly the same as before. The total costs is still $\lg \lg u$.

For vEB-TREE-SUCCESSOR(V, x), first we deal with the condition where $V.u = 2$, and this is line 1-4 in CLRS. Then if $V.min$ is not NIL and $x < V.min$, return $V.min$. Then try to find successor in $V.cluster[\text{high}(x)]$, which is the same as line 7-10 in CLRS. If $V.cluster[\text{high}(x)]$ is empty or $\text{low}(x)$ is not less than the maximum value in the cluster, find the successor of $\text{high}(x)$ in $V.summary$. If such a successor exists, return the minimum value of the cluster; otherwise, compare x with $V.max$: if $x < V.max$, return $V.max$; otherwise (including $V.max$ is NIL), return NIL.

The only change to this is to compare x with $V.max$ if all attemps to find a successor before fails. This takes constant costs, and the total costs of SUCCESSOR is still $\lg \lg u$.