Problem Set 03

Mostafa Touny

Contents

Exercises

Ex. 3.1

Done.

Ex. 3.2

Skipped; I don't understand the problem.

Ex. 2

Skipped; I don't understand the problem.

Problems

Prob. 3.1

psuedo-code changes; compare new complexity with old one

a

Psuedo-code. No Changes at all. Note the sequence $2^9 \to 2^3 \to 2^1$, up to the base case of $u = 2$ as before, starting with total data of size $2^9 = 512$.

Complexity. Similarly $\mathcal{O}(\lg \lg u)$. We follow the same reasoning on the master method but on the case of a cluster size $u^{1/3}$. We gain $\lg_b a = \lg_3 1 = 0$, or more accurately $\lg_b a = \lg_{4/3} 1 = 0$, whereby $\lceil m/3 \rceil \leq 3m/4$. Thus, Reaching exactly the same complexity.

b

Psuedo-code Changes

`vEB-TREE-MIN` No Changes.

`vEB-TREE-MAX` No Changes.

`vEB-TREE-MEMBER(V, x)` No Changes.

`vEB-TREE-SUCCESSOR(V, x)` No Changes.

`vEB-TREE-PREDOCESSOR(V, x)` Symmetric, Skipped.

`vEB-EMPTY-TREE-INSERT(V, x)` No Changes.

Intuitively, We apply the same trick of swapping `V.min`. Complexity is the same, as we only re-ordered a constant-complexity code block.

```
vEB-TREE-INSERT(V, x)
  if V.min == NIL
    vEB-EMPTY-TREE-INSERT(V, x)
  else
    if x < V.min
      exchange x with V.min
    if x > V.max
      exchange x with V.max
    if V.u > 2
```

```
      if vEB-TREE-MINIMUM(V.cluster[high(x)] == NIL
        vEB-TREE-INSERT(V.summary, high(x))
        vEB-EMPTY-TREE-INSERT(V.cluster[high(x)], low.x)
    else
      vEB-TREE-INSERT(V.cluster[high(x)], low.x)
```

Intuitively, We apply the same trick of updating `V.min` and assigning `x` to a new value before the delete operation. In this case, there's no need to check for `summary` as `V.max` will be already set to either `V.min` or the suitable `index` value. Complexity is the same for exactly the same reasoning

```
vEB-TREE-DELETE(V, x)
.
..
else
  if x == V.min
    first-cluster = vEB-TREE-MINIMUM(V.summary)
    x = index(first-cluster, vEB-TREE-MINIMUM(V.cluster[first-cluster]))
    V.min = x
  elseif x == V.max
    last-cluster = vEB-TREE-MAXIMUM(V.summary)
    x = index(high(x), vEB-TREE-MAXIMUM(V.cluster[high(x)]))
    V.max = x

  vEB-TREE-DELETE(V.cluster[high(x)], low(x))

  if vEB-TREE-MINIMUM(V.cluster[high(x)]) == NIL
    vEB-TREE-DELETE(V.summary, high.x)
```

Algorithm's correctness are not proved. We relied only on our intuition. Not comfortable the new modification is simpler and yet offers the same complexity; Why not illustrated in this way by the author?