600.363/463 Algorithms - Fall 2013

Solution to Assignment 6

(30 points)

I (10 points) 21-1 Off-line minimum

a The values in the extracted array are 4, 3, 2, 6, 8, 1.

b Note that each key is inserted only once. Since the loop starts from the smallest value of $i = 1$, for each $i$, if it is in some $K_j$, which means it is inserted by $I_j$, then before $I_j$ the dynamic set $T$ does not contain $i$, and after $I_j$ it is inserted into $T$, therefore in the the EXTRACT-MIN after $I_j$, $i$ is the smallest in $T$, so it must be extracted out; if $i$ is not in any key set, it will be skipped. Hence extracted[j] contains the value in $T$ which the j-th EXTRACT-MIN returns.

c Using disjoint-set data structure, we can construct an efficient implementation of the algorithm. Initially create disjoint-sets for the subsequences $I_1, ... I_{m+1}$ and place the representative of each set in a linked list in sorted order. Additionally, label each representative with its subsequence number. Then line 2 is implemented by FIND-SET operation; in line 5 the next set is obtained from the root as the next set in the linked list; line 6 is implemented by UNION operation.

Since the OFF-LINE-MINIMUM can be implemented by a sequence of disjoint-set operations, the running time for OFF-LINE-MINIMUM is $O(m \alpha(n))$ (or $O(m \log^* n)$).

II (10 points) 21-2 Depth determination

a If we use disjoint-set data structure, MAKE-TREE takes $\Theta(1)$ time; GRAFT is basically a union operation, thus it takes $\Theta(1)$ time; the cost of FIND-DEPTH depends on the depth of the given node. For a sequence of $m$ operations, the depth of a node is $O(m)$, thus for the worst case $T(n) = mO(m) = O(m^2)$.

Wlog let $k = m/3$ be an integer, considering a sequence of operations with $k + 1$ MAKE-TREES creating $k + 1$ single-node trees, $k$ GRAFTs forming a single path, and $k - 1$ FIND-DEPTH for the leaf node, then the running time of the $m$ operations is $T(n) = (k + 1) * \Theta(1) + k\Theta(1) + (k - 1) * k = \Omega(m^2)$.

Hence the worst case running time is $\Theta(m^2)$.

b MAKE-TREE can be implemented by creating a disjoint set with a single node $v$. $d[v]$ is set to be 0 inside MAKE-TREE.

c According to the definition of $d[v]$ that the sum of the pseudo-distances along the path from $v$ to root of its set $S_i$ equals to the depth of $v$ in $T_i$, FIND-DEPTH can be implemented by modifying FIND-SET in such a way: assume the path is composed of $v_0, v_1, \ldots, v_k$ where $v_k$ is the root, for every node $v_i$ along the path, update $d[v_i] = \sum_{j=i}^{k} d[v_j]$, i.e., with path
compression, whenever the parent pointer of a node changes, the pseudodistance is updated by the sum of its ancestor’s pseudodistances.

d Let the path from \( v \) to root of the tree is \( v = v_0, v_1, v_2, \cdots, v_k = w \), where \( w \) is the root. If \( \text{rank}(r) < \text{rank}(w) \), using UNION operations to make \( r \)’s parent pointer point to \( w \), and updating \( d[r] \) by \( d[r] + \sum_{i=0}^{k-1} d[v_i] \); If \( \text{rank}(r) \geq \text{rank}(w) \), using UNION operations to make \( w \)’s parent pointer point to \( r \), updating \( d[r] \) by \( d[r] + \sum_{i=1}^{k-1} d[v_i] \) and updating \( d[w] \) by \( d[w] - d[r] \). Note that the updating operation does not require extra cost in UNION.

e Since the sequence of \( m \) MAKE-TREE, FIND-DEPTH and GRAFT operations can be implemented by a sequence of \( m \) disjoint-set operations, the running time is \( O(m\alpha(n)) \)(or \( O(m\log^* n) \)).

III (10 points)

1 Let \( T(1) = T(2) = 1. \) Assume \( T(n) = c^n \). Since \( T(n) = 2T(n-1) + 3T(n-2) \), for \( n > 2 \), we have

\[
c^2 = 2c^{n-1} + 3c^{n-2}
\]

Solving this equation we get \( c_1 = 3 \) and \( c_2 = -1 \).

Let \( T(n) = a3^n + b(-1)^n \), then by the initial values:

\[
\begin{align*}
T(1) &= 3a - b = 1 \\
T(2) &= 9a + b = 1
\end{align*}
\]

Solving this equation we get \( a = 1/6 \) and \( b = -1/2 \).

Therefore,

\[
T(n) = \frac{1}{6}3^n - \frac{1}{2}(-1)^n = O(3^n).
\]

2 Intuitively, since \( 2 \cdot \frac{1}{3} + \frac{1}{4} + \frac{1}{12} = 1 \), claim that \( T(n) \leq cn \log n \), then prove by induction:

\[
T(n) \leq 2T(n/3) + T(n/4) + T(n/12) + n
\]

\[
\leq 2c\frac{n}{3} \log \frac{n}{3} + c\frac{n}{4} \log \frac{n}{4} + c\frac{n}{12} \log \frac{n}{12} + n
\]

\[
= cn \log n - \left( \left( \frac{2}{3} \log 3 + \frac{1}{4} \log 4 + \frac{1}{12} \log 12 \right) c - 1 \right) n
\]

When \( c \geq 1 \), \( \left( \frac{2}{3} \log 3 + \frac{1}{4} \log 4 + \frac{1}{12} \log 12 \right) c - 1 > 0 \), then

\[
T(n) \leq cn \log n
\]

Hence \( T(n) = O(n \log n) \).
3 Intuitively, since \(2 \times \frac{1}{3} + \frac{1}{4} = \frac{11}{12} < 1\), claim that \(T(n) \leq cn - d\), then prove by induction

\[
T(n) \leq 2T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + n
\]

\[
\leq 2 \times \frac{11}{12}cn + n - 2d + \frac{cn}{4} - d + n
\]

\[
= \frac{11}{12}cn + n - 3d
\]

\[
\leq cn - \left(\frac{1}{12}c - 1\right)n - d
\]

\[
\leq cn - d
\]

when \(c \geq 12\). Hence \(T(n) = O(n)\).

4 Assume \(T(1) = 1\), then

\[
T(n) \leq 4T\left(\frac{n}{2}\right) + n^2 \log n
\]

\[
\leq 4(4T\left(\frac{n}{2^2}\right) + \left(\frac{n}{2}\right)^2 \log \frac{n}{2}) + n^2 \log n
\]

\[
= 4^2T\left(\frac{n}{2^2}\right) + 4\left(\frac{n}{2}\right)^2 \log \frac{n}{2} + n^2 \log n
\]

\[
\leq 4^2T\left(\frac{n}{2^2}\right) + n^2 \log n + n^2 \log n
\]

\[
\leq 4^3T\left(\frac{n}{2^3}\right) + n^2 \log n + n^2 \log n + n^2 \log n
\]

\[
\cdots \text{(by substitutions)}
\]

\[
\leq 4^\log nT(1) + \sum_{i=1}^{\log n} n^2 \log n
\]

\[
= O(n^2 \log^2 n)
\]

Remark: The series in the second-to-last line also can be obtained by recursion tree method.