EXERCISE FOR CSE202 – WEEK 9

The amortized complexity estimate for a sequence of m union or find operations with rank and path compression can be further improved. The basic observation is that the simple bound $T(m, n, r) \leq nr$ was used in an intermediate step to compute a better bound, that could be used in its place.

Question 1. *Using this idea show that the amortized complexity of that algorithm is actually $O(m \log^* \log^* n)$ array accesses ($m \geq n$). Indicate for which value of n this function $\log^* \log^* n$ becomes larger than 3.*

Solution. The bound we have on $T(F, C)$ is $m + 2n \log^* r$. Using this bound for the high forest gives

$$T(F_+, C_+) \leq m_+ + 2\frac{n}{2^s} \log^* r.$$

Thus, from the inequality at the bottom of slide 18,

$$T(F, C) \leq T(F_-, C_-) + 2m_+ + n + 2\frac{n}{2^s} \log^* r$$

and since $m_+ = m - m_-$,

$$T(F, C) - 2m \leq T(F_-, C_-) - 2m_- + n + 2\frac{n}{2^s} \log^* r.$$

Choosing $s = \lceil \log_2 \log^* r \rceil$, the last summand becomes smaller than $2n$, whence

$$T(F, C) - 2m \leq T(F_-, C_-) - 2m_- + 3n.$$

where now F_- is a forest all whose nodes have rank at most $\log_2 \log^* r$. Iterating this construction on this forest and so on $\log^* \log^* r$ times gives

$$T(F, C) \leq 2m + 3n \log^* \log^* r = O(m \log^* \log^* n).$$

The largest value of k such that $\log^* \log^* k = 3$ satisfies $\log^* k = 16$, which means that k is obtained by iterating 16 times the map $x \mapsto 2^x$ starting from $x = 1$. This is a number that is unimaginably large (and so is its number of digits). This, plus 1, is the value where this function becomes larger than 3. □

Question 2. *Improve this bound further to $O(m \log^{*^3} n)$, where $\log^{*^p}$ denotes the $\log^*$ function iterated p times.*

Solution. The starting point is now

$$T(F_+, C_+) \leq 2m_+ + 3\frac{n}{2^s} \log^* \log^* r,$$

so that the same set of steps leads to

$$T(F, C) \leq 3m + 4n \log^{*^3} r. \qquad \square$$

Question 3. *Improve finally this bound further to $O(m\alpha(n))$, where $\alpha(n)$ is the number of times the $\log^*$ function must be applied before the value becomes at most 1. What is now the smallest value of n where this function becomes larger than 3?*

Solution. By induction, for any integer $k \leq 2$, this reasoning leads to

$$T(F, C) \leq km + (k+1)n \log^{*^k} r.$$

For $k = \alpha(r)$, both terms become $O(km)$. This gives the result.

The largest value of k such that $\alpha(k) = 3$ is the largest k such that $\log^{*^3} k = 1$, i.e., $\log^{*^2} k = 2$, $\log^* k = 4$, $k = 2^{16} = 65536$. So the desired value is 65537, which is actually smaller than before. $\square$