ASSIGNMENT: HW7

NAME: PRASANTH BHAGAVATULA

CWID: A20355611

1. ~~Let~~ ~~Consider from empty structure;~~

1. 'We know by definition of Union operation, ~~th~~ as long ~~as~~ 'x' does not have any ~~root~~ parent 'rank(x)' will increase for all the union operations. Once, 'x' gets ~~at~~ a parent, the rank(x) will ~~might~~ not change. But rank(x.p), where x.p is the parent, will be monotonically increasing till it gets a parent. Hence, by the definition of level(x), the value might remain same or increase by 1 at most for every union operation performed on x.p and x.p still does not have a parent.

Similarly, the same relationship ~~logic~~ applies to x.p and its parent (x.p).p. Note that, once 'x.p' gets a parent the rank(parent(x)) will not change and so, the level(x) is not going to change. But rank(x.p.p) will increase causing level(x.p) increase. .

$\therefore$ level(x.p) $\geq$ level(x)

ASSIGNMENT: HW7

NAME: PRASANTH BHAGAVATULA

CWID: A20355611

② Given tower(n) function ican be re-written as

$$\text{tower}(j) = \underbrace{2^{2^{2^{\cdot^{\cdot^{2}}}}}}_{j} \equiv 2\uparrow\uparrow j,$$ if it is represented in the Knuth- Up-arrow notation. If we consider the value of $A_3(1)$

$$A_3(1) = A_2^{(1+1)}(1) = A_2^{2}(1) = A_2(A_2(1))$$

We know that $A_2(j) =$ ~~$2j+1$~~ $2^{(j+1)} \times (j+1) - 1$

~~$\Rightarrow A_2(1) = 2\times(1)+1 = 3$ and $A_2(3) = 2\times 7\ 7$~~

$\therefore A_2(1) = 2^{(1+1)} \times (1+1) - 1 = 2^2 \times 2 - 1 = 7$

and $A_2(7) = 2^{(7+1)} \times (7+1) - 1 = 2^8 \times 8 - 1 = 2^{11} - 1 = 2047$

$\therefore A_3(1) = A_2(A_2(1)) = A_2(7) = 2047$

and $\text{tower}(1) = 2^{\text{tower}(n-1)} = 2^{\text{tower}(0)} = 2^1 = 2$ as going by the definition

$\therefore A_3(1) > \text{tower}(1)$

Consider $A_3(j)$

$\Rightarrow A_3(j) = A_2^{j+1}(j)$

$\Rightarrow A_3(j) = A_2^{j}(A_2(j))$

$\Rightarrow A_3(j) = A_2^{j}(2^{(j+1)} * (j+1) - 1)$

$\Rightarrow A_3(j) = A_2^{j}((2 \times 2^{j} \times (j+1)) - 1)$ ——— ①

$$\Rightarrow A_3(j) = A_2^{j-1}\left(\left(2^{(2\times 2^j\times j+1)} \times (2\times 2^j\times j+1)\right) - 1\right)$$

$$\Rightarrow A_3(j) = A_2^{j-1}\left(\left(2^2 \times 2^{2^j\times(j+1)} \times (2\times 2^j\times(j+1))\right) - 1\right) \qquad \text{——} \; \textcircled{2}$$

If we observe, the first factor in the ~~obo~~ equation ① and ② when the ~~power~~ recursive operation number of 'A₂' is 'j', we have a '2' already and when the recursive operation of 'A₂' is 'j-1' we have '2^2' and similarly when the recursive operation number become 'j-2 and we will have '2^{2^2}' and so at the end we will have

$$\underbrace{2^{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}}_{j+1} \equiv 2\uparrow\uparrow(j+1)$$

and tower(j) ≡ $2\uparrow\uparrow(j)$, so clearly

$$A_3(j) > \text{tower}(j)$$

ASSIGNMENT: HW7

NAME: PRASANTH BHAGAVATULA

CWID: A20355611

(3) In oder to maintain the mark-bit and counters, the MAKESETT operation simply create a single-ton set with 'mark-bit' ~~setting~~ set as 'not-a-ghost' and ~~coun~~ number of elements in tree - counter to '1' and number of ghost elements in tree counter to '0'. This takes constant time to do and so, the ~~cost~~ amortized cost of MAKESET operation will remain as O(1).

Similarly, in the UNION operation we just need to sum-up the counters of both the roots and ~~re~~ ~~ke~~ store those value in the new root. ~~For~~ For example, if ~~a~~ we are joining two sets x and ~~y~~ and y becomes the parent then

new number of ~~ne~~ elements in 'y' = old number of elements in 'y' + 'number of elements in 'x''

new number of ghost element in 'y' = old numbers of ghost elements in 'y' + number of ghost element in 'x'

Doing simple addition takes constant time and so, the amotized cost is still $O(\alpha(n))$

The FIND operation does not have any affect on the mark-bits (or) counter, as, the counter values of the root one is the only ones that are considered and since the root does not change, there is no impact and so, the amortized cost is still $O(\alpha(n))$

The DELETE operation involves two major -steps. First step to find the root and ~~maintain~~ the second ~~set~~ ~~to~~ step is to re-build incase the number of ghost elements increases the threshold limit of $\lfloor |S|/2 \rfloor$. ~~Finding an element~~ Finding the root is done using the FIND operation. We will consider the second step:- where in we need to build the ~~A~~ dis-joint set forest by removing the marked elements.

The actual cost involves choosing one non-marked element as root ~~and~~ and making all other non-marked elements ~~a~~ as its children.. So, this involves going through all elements $O(S)$. Consider, the potential of S after re-buiding. We know for sure, that ~~if~~ the rank of parent is greater than the rank of children. If we make the rank of parent in the new structure as '1', the we make the rank of all of its children as '0', which makes the potential after as $\alpha(n)$

~~let the potential before e be some constan~~

~~So~~, ~~E~~ So, the amortized cost of the DELETE operation is the sum of amortized cost of FIND operation and amortized cost of Re-building set S. ~~on~~

∴ $O(s) - (s - (\alpha(n) + 2))$ is the amortized cost of FIND

and $O(s) + \alpha(n) - \phi_q(s)$ is the amortized cost of Rebuiding

Ignoring the decrease of $\phi_q(s)$ in the re-buiding amortized cout, we have the amortized cost of DELETE q

$$O(s) - s + \alpha(n) + 2 + O(s) + \alpha(n)$$

$$= 2\,O(s) - s + 2(\alpha(n) + 1)$$

$$\equiv O(\alpha(n) + 1)$$

Similar, to the FIND operation, we can scale up the units of potential to dominate the constant hidden in $O(s)$.

∴ For a sequence of m operations of MAKESET, UNION, FIND, DELETE which includes n ~~m~~ MAKESET operations the amortized cost is $\underline{O(m + m\alpha(n))}$