(1) $$\text{tower}(n) = \begin{cases} 2^{\text{tower}(n-1)} & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases} \quad \text{———} \quad (1)$$

For integers $k \geq 0$ and $j \geq 1$ we define function $A_k(j)$ as;

ACKERMANN'S FUNCTION :

$$A_k(j) = \begin{cases} j+1 & \text{if } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{if } k \geq 1 \end{cases} \quad \text{———} \quad (2)$$

Suppose $A_3(j) > \text{tower}(j)$; then by induction we need to prove $A_3(j+1) > \text{tower}(j+1)$. Consider $j = 1$

$$A_3(j+1) = A_2^{j+2}(j+1) \quad \{\text{frome eq}^n \ (2)\}$$

Considering $F^i(x) = \underbrace{F(F(F(\cdots F(x)))}_{i}$

$$= A_2(A_2^{j+1}(j+1))$$

$$> A_2(A_2^{j+1}(j))$$

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

$$\therefore A_3(j+1) > A_2(A_3(j))$$

$\therefore$ From eq^n 1 and 2

$$A_3(j+1) > A_2(tower(j)) \quad \text{—— } 3$$

Based on ACKERMANN'S FUNCTION ;

$$A_2(j) = 2^{j+1}(j+1) - 1$$

$$A_2(j) < 2^j(j) - 1 \quad \text{for } (j \geq 1) \quad \text{—— } 4$$

Substituting above in eq^n 3 we get ;

$$A_3(j+1) > 2^{tower(j)}$$ { considering $j = tower(j)$ in eq^n 4 }

$\therefore$ From eq^n 1 ;

$$A_3(j+1) > tower(j+1)$$

$\therefore$ ~~when $j=1 \Rightarrow$ $A_3(j) > tower(j)$~~

By above statement we can say that our assumption that $A_3(j) > tower(j)$ is correct. Hence proved.

② (a) $\phi = \sum_{\text{nodes } x} \text{depth}(x)$

ⓘ **Make-set :**

Consider single new node x. So Make set (x) operation will create node x with depth and rank of zero (0)

$\therefore$ Potential Change $\phi_{final} = 0$

As initially their didn't exist potential change of x

$\therefore$ $\phi_{initial} = 0$

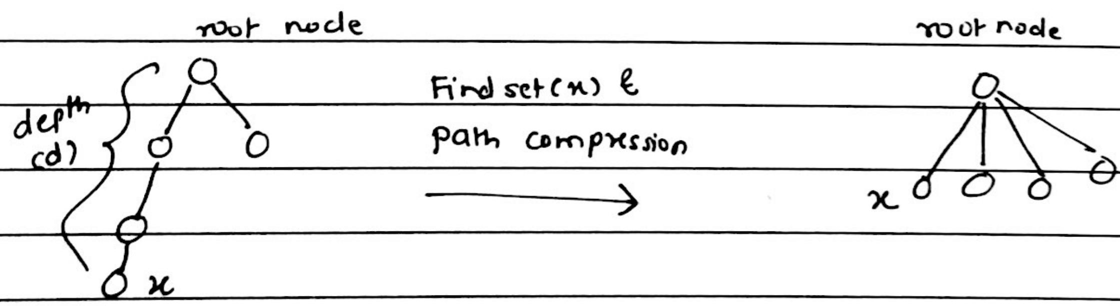
$\therefore$ Amortized Cost $=$ Actual cost $+ \phi_{final} - \phi_{initial}$

$= O(1) + 0$

$\therefore$ Amortized cost $= O(1)$

(ii) **Find - Set :**

Find-set (x) operation will return the root of the set with path compression making ~~all~~ the in path node pointing to root directly



As d = depth of the node

$\therefore$ Actual cost $= O(d)$

The depth of all in path nodes will be decreased and set to 1 as all are directed to root node.

② $\Delta \phi = -O(n)\, O(d)$ → $O(n)$ elements will have change in depth

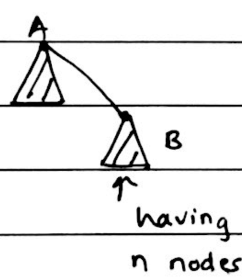
$\therefore$ Amortized Cost $= O(d) - O(n)\, O(d)$

(iii) Union

For union operation we just need to change the pointers. We need to do Link operation. $\therefore$ Union $(x, y) \Rightarrow$ Link (x, y). Considering trees A, B → Link $(A, B) \cong$ Union (A, B)

$\therefore$ Actual cost $= O(1)$ ← just change of pointers

- Consider two trees A and B. Suppose we do Link operation on these trees. Then without the loss of generality, A turns out parent for B if we connect tree B to A.



There will be no change in depth of A. As B is the child of A therefore depth of nodes in B will be increased by 1. Therefore for n nodes it will take $O(n)$

$\therefore$ Amortized cost $=$ Actual cost $+ \Delta \phi$

$= O(1) + O(n)$ ← n nodes in x trees.

$\therefore$ Amortized cost $= O(n)$

(2) (b) $\phi = \sum_{\text{node } x} \text{height}(x)$

(i) Make set

Consider single new node x. So Make set (x) operation will create node x with height and rank of zero (0).

$\therefore \phi_{\text{final}} = 0$

As x didn't exist before

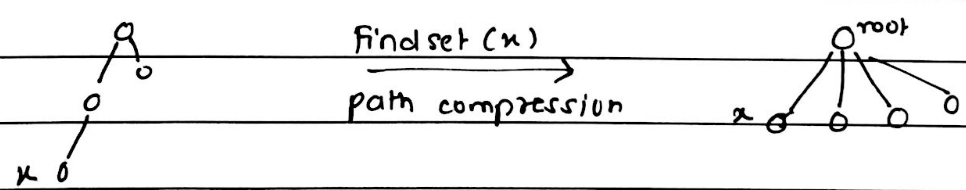
$\therefore \phi_{\text{initial}} = 0$

$\therefore$ Amortized cost = Actual cost + $\Delta \phi$

$\therefore$ Amortized cost = $O(1) + 0 = \underline{\underline{O(1)}}$

(ii) Find-set

Find-set(x) operation will return the root of the set with path compression making the in path node pointing to root directly



h = heigth of the node

$\therefore$ Actual cost = $O(h)$

Findset(x) operation will decrease the height of subtree

② $\therefore \Delta\phi = -O(\log n)$

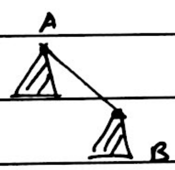
$\therefore$ Amortized cost $= O(h) - O(\log n)$
$= O(\log n)$

(iii) <u>Union</u>

For union operation we just need to change the pointers. We need to do Link operation. $\therefore$ Union (A, B) = Link (A, B)

$\therefore$ Actual cost $= O(1)$ ← just change of pointers

- Consider two trees A and B. Suppose we do Link operation on these trees. Then without the loss of generality, A turnout to be parent of B if we connect tree B to A.



→ considering n nodes with height (h) the potential difference will be $O(h)$ as height of new root will increase

$\therefore$ Amortized cost = Actual cost + $\Delta\phi$

$\therefore$ <u>Amortized cost $= O(1) + O(h)$</u>

(3) Ackermann's Function is defined as $A(i,j)$ for $i, j \geq 1$

$A(1,j) = 2^j$ for $j \geq 1$,

$A(i,1) = A(i-1, 2)$ for $i \geq 2$,

$A(i,j) = A(i-1, A(i, j-1))$ for $i, j \geq 2$

If $m \geq n \rightarrow 1 \leq \alpha(m+n, n) \leq \alpha(m,n)$

$\therefore$ Let $\alpha(m,n) = \min \{ i \geq 1 \mid A(i, \lfloor m/n \rfloor) > \log n \}$

$\alpha \rightarrow$ Functional inverse of Ackermann's function

$m \rightarrow$ Number of find operations

$n \rightarrow$ Number of make-set operations

- Ackermann's function has explosive growth. so it's inverse is added $\rightarrow \alpha$ to slower the growth

If $m \alpha(n,n) \leq n \rightarrow \Omega(n)$ bound

~~If~~

Halving :

- Except the last and next to last node, we make every other node along the find path point to the node two past itself.

Function Findnode (x)

Define y (local variable)

Assign value of x to y $\{ y = \cancel{x} \}$

do

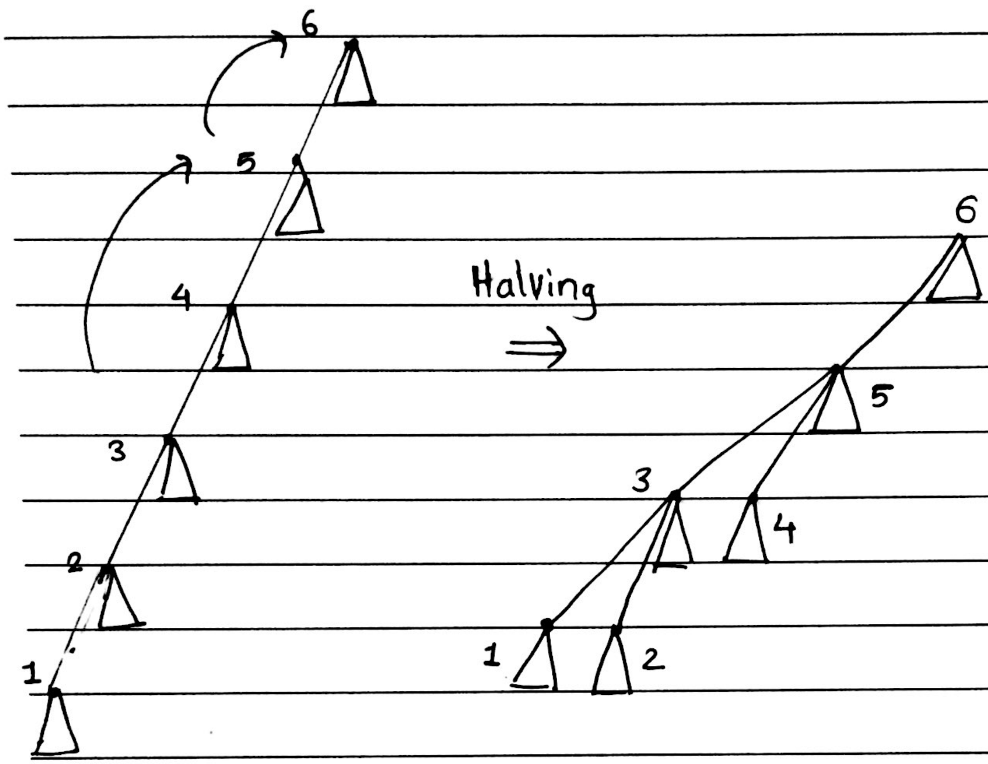
$p(p(y)) \neq p(y)$ then ← check if node ~~does~~n't have grandparent

$y = p(y) = p(p(y))$;

return $p(y)$

End Findnode ;

③ Example :



For cases :

① $m \geq n$ → Running time will be $\Theta(m \log_{(1+m/n)} n)$

② $m < n$ → $O(n \log m)$ upper bound

$\Omega(n + m \log_n)$ lower bound

③ A sequence of make-set, link, find operations with rank and path-halving path compression runs in $O(n + m\,\alpha(m+n, n))$ time

References :-

Tarjan, Robert E. and Jan Van Leeuwen Research Paper on "Worst case analysis of set union algorithms". Journal of the ACM (JACM)