Homework Assignment-7 Dhayalini Nagaraj
A20359686

1. Prove that $A_3(j) > tower(j)$, where

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

Ackermann's function

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

$$\alpha(n) = \min\{k \mid A_k(1) \geq n\}$$

Let us take j as 1

(ie) $j = 1$

Then $A_3(j) > tower(j)$ — ①

If $A_3(j) > tower(j)$ when $j \geq 1$, then

$$A_3(j+1) = A_2^{j+2}(j+1) \quad \left(\text{ie } A_2^{(j+1)+1}(j+1)\right)$$

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

$$= A_2(A_3(j))$$

$$> A_2(tower(j)) \quad \text{— (from ①)}$$

$$> 2^{tower(j)}$$

$$= tower(j+1)$$

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

(ie) $A_3(j) > tower(j)$

Hence Proved.

Scanned by CamScanner

2a) Given Potential function

$$\Phi = \Sigma_{node\ x}\ depth(x)$$

Make-set(x) - Create a singleton set containing x

Change is only due to depth. It creates single node with rank (0).

The potential change in this case is negligible.

Amortised cost = Actual cost + change in potential

$$= O(1) + \sum_{i=0}^{n} (depth(x_{i+1}) - depth(x_i))$$

$$= O(1) + (depth\ x_1 - depth\ x_0) + (depth\ x_2 - depth\ x_1) + \dots (depth\ x_n - depth\ x_{n-1})$$

$$= O(1) + 0 \quad \text{(change is negligible)}$$

$$= O(1)$$

Find-Set(x) - Return an identifier of the set containing x.

Potential function general $\Phi_i(x) = \begin{cases} \alpha(n) \cdot rank(x) & \text{if } rank(x)=0 \\ (\alpha(n) - level(x)) \cdot rank(x) - iter(x) & \text{if } rank(x) \geq 1 \end{cases}$

Let the find-set path be considered as m

If it is not the root node the x's potential will decrease by atleast 1.

$\therefore$ the change in potential would be $m-1$

Amortized cost = Actual cost + change in potential

$$= O(m) - O(m-1) = O(1)$$

<u>Union</u>: Combine two sets into a new set, destroying the other two sets.

Let two sets be A and B.

If A and B are root nodes then if rank(A) $\geq$ rank(B). Make B a child of A

If rank(A) = rank(B), increase rank(A) by one.

When B becomes child of A, $n/2$ nodes can be created in that case.

Change in potential is then $O(n)$

Actual cost is $O(1)$ it takes constant time.

Amortized cost = Actual cost + Change in potential

$$= O(1) + O(n)$$

$$= O(n)$$

2b) $\Phi = \sum_{node_x} height(x)$

<u>Make-Set(x)</u> - Create a singleton set containing x.

Change is only due to height which will not affect much in make-set(x). The node x is kept as root and set is created.

$\therefore$ Actual cost is $O(1)$

Change in potential is $= (height_{x_1} - height_{x_0}) + (height\, x_2 - height\, x_1) + \cdots (height\, x_n - height_{n-1})$

Amortized cost = Actual cost + Change in potential

= O(1)

(The change in potential is due to height by new node and that is considered 0)

Find-set(x)

In this, ~~any~~ all the nodes are affected by the ~~find path set~~ find-set operation except the last node.

If x is not the root node, the potential will decrease by 1.

Size of find set be m.

Change in potential be m-1

∴ Amortized cost = Actual cost + change in potential

~~= O(m) + (m-1)~~

= O(1)

Union:

The height of the nodes ~~does~~ in the union will not change except the height of the root node by 1.

∴ change in potential = O(1)

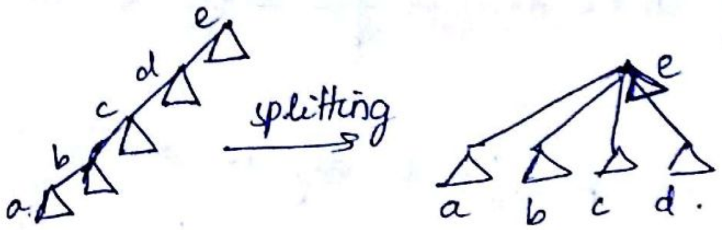
Actual cost = O(1)

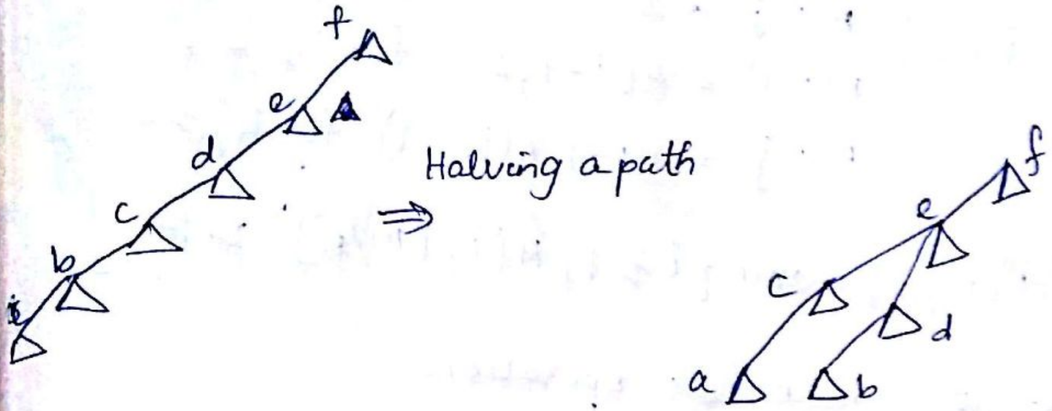
∴ Amortized cost = Actual cost + change in potential

= O(1) + O(1)

= O(1)

3. Do the amortized analysis of Union Find Data structure with rank and path-halving path compression in which the grand parent of each node is made its parent during a find operation.





Halving is ~~a~~ not a form of compaction, with a minor changes, it gives a tight bound for halving. The initiation operation makeset(e), find(e), union(e,f). The algorithm performs an arbitary sequence of steps to access find(e) One more restriction is imposed on the algorithm called separability. Any correct seperable algorithm must perform ~~canonical node of the root~~ atleast one pointer construction step per link

$\Omega(n)$ lower bound on the no. of steps needed for seperable algorithm.

For $m = \Omega(n)$

then $\Omega(m\,\alpha(m,n))$ lower bount, α functional inverse of Ackermann's function is as follows.

For $i, j \geq 1$, $A(i,j)$ is defined as

$$A(i,j) = \begin{cases} A(1,j) = 2^j & \text{for } j \geq 1, \\ A(i,1) = A(i-1,2) & \text{for } i \geq 2, \\ A(i,j) = A(i-1, A(i,j-1)) & \text{for } i,j \geq 2 \end{cases}$$

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

n- no. of make set operations

m - no. of find operations.

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

Runtime $O(n + m\alpha(m+n,n))$ time.

Reference:

citeseerx.ist.psu.edu

worst cost Analysis of set Union Algorithm - Robert E. Tarjan

https://massivedatasets.files.wordpress.com