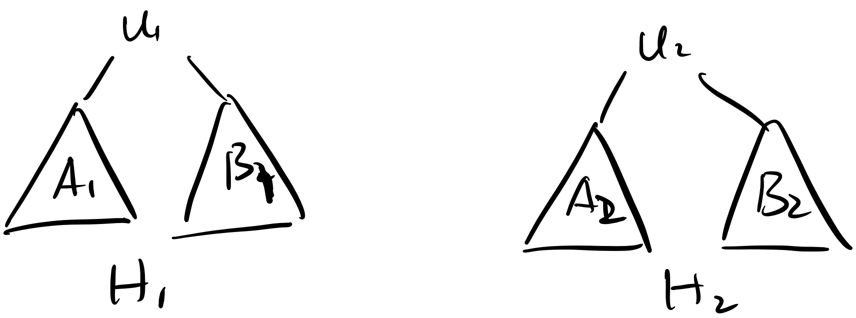
self-adjusting version of leftist heap

Merge (H_1, H_2)

1. Define A_1, u_1, B_1 and A_2, u_2, B_2 as follow.



2. If $u_1 == NULL$

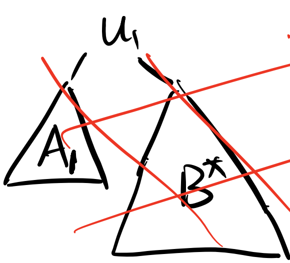
   return H_2

3. If $u_2 == NULL$

   return H_1

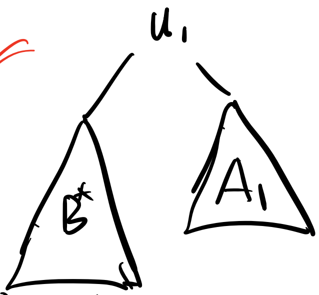
swapchildren for all nodes on right path of H_1 and H_2

4. If $u_1.key < u_2.key$
5. $B^* = merge(B_1, H_2)$
6. $H =$



~~if $npl(B^*.root) \le npl(A_1.root)$~~

~~or~~



~~otherwise~~.

before	after
heavy	light
light	heavy
	light

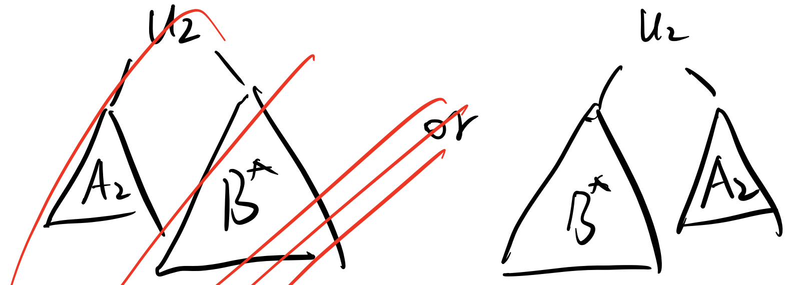
update $npl(u_1)$

return H

7. If $u_1.key > u_2.key$

   $B^* = Merge(H_1, B_2)$

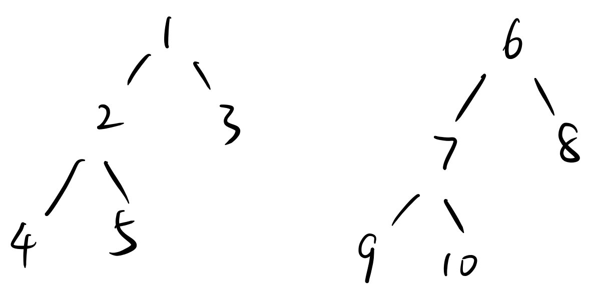
$H =$



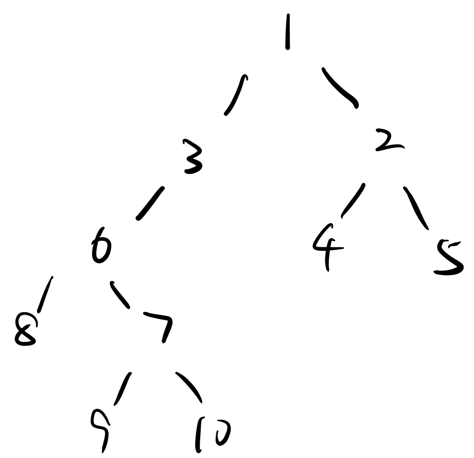
~~If - - -~~ or ~~otherwise.~~

update npl(u_2)
return H

\# nodes on right paths of H_i



H_1 H_2



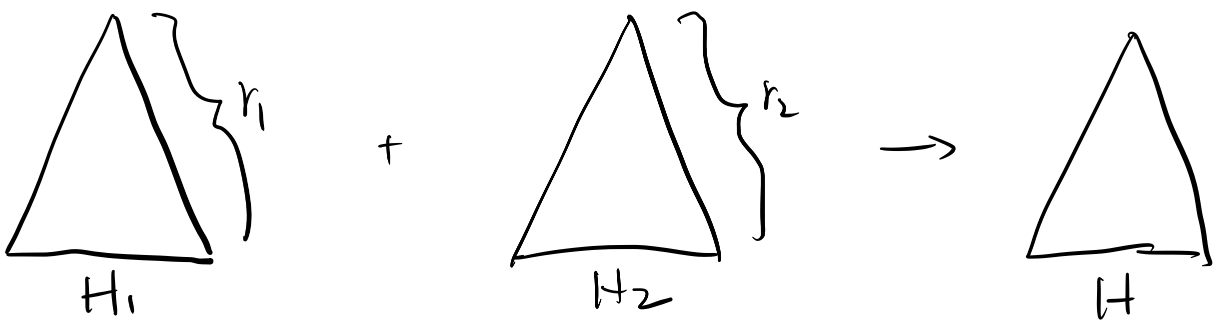
Ins & deletemin via ~~meg~~ merge.
delete & decreasekey not supported ?

Potential function
Given a node $u \in H$,

v is heavy if $size(T_{u.right}) \geq size(T_{u.left})$
light otherwise

$\Phi(H) = \#$ heavy nodes in H.

Merge(H_1, H_2)



actual cost $= O(r_1 + r_2) = O(l_1 + h_1 + l_2 + h_2)$

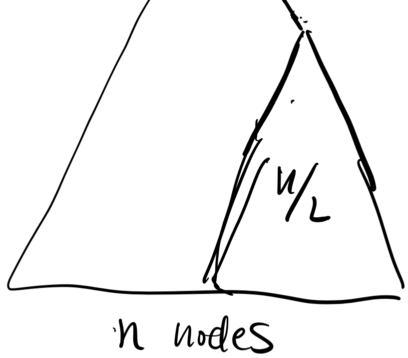
$\Phi(H_1) + \Phi(H_2) = h_1 + h_2 + h$

$\Phi(H) \leq h + l_1 + l_2$

$\Delta\Phi = l_1 + l_2 - h_1 - h_2$

amortized cost $= O(l_1 + h_1 + l_2 + h_2) + O(l_1 + l_2 - h_1 - h_2)$
$= O(l_1 + l_2) = O(\lg n)$

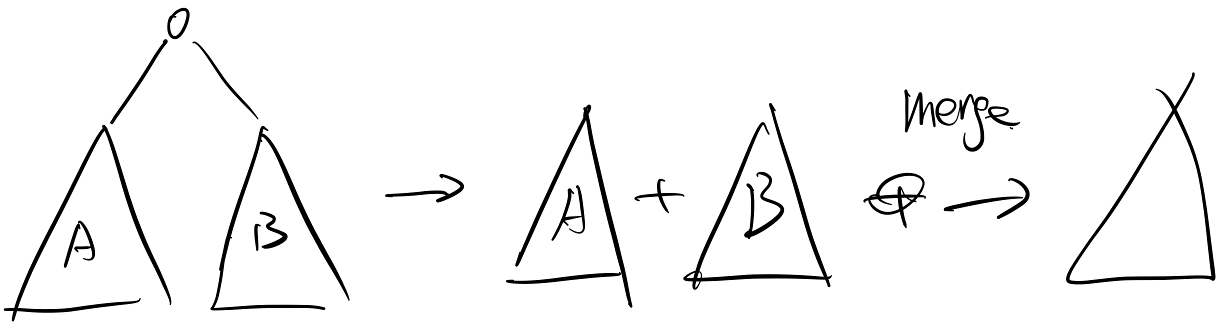
claim $l_1 \leq \lg n_1 \quad l_2 \leq \lg n_2$



$\lg n$

Insertion $O(\lg n)$

Deletemin $O(\lg n)$



actual cost	$O(1)$	$O(l_1+l_2+h_1+h_2)$
$\Delta\Phi$	≤ 0	$O(l_1+l_2-h_1-h_2)$