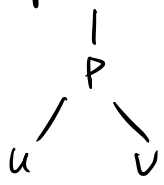
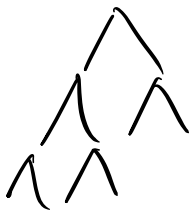
binary heap

1. heap property



$p<u$ & $p<v$

2. ~~per~~ almost perfect binary tree

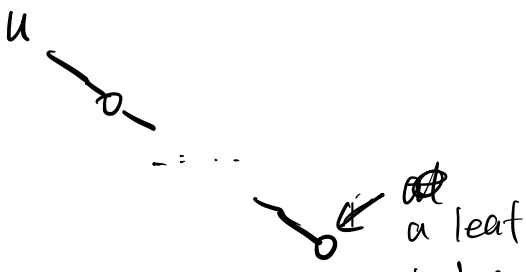


Merge: rebuilt $O(n)$

leftist heap

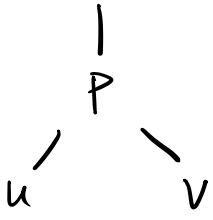
1. heap property
2. binary ~~tree~~ with leftist property (leftist tree)

for any node u, the shortest descending path from u to any node with ~~most~~ at most one child



(NULL path length)

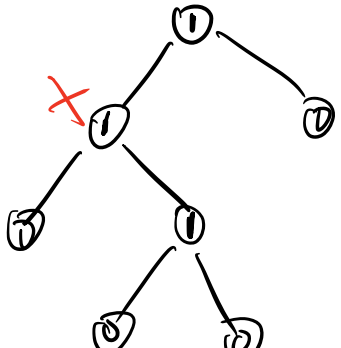
npl(u) = # edges on this path

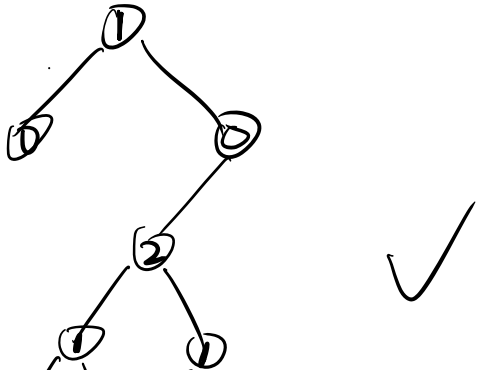


$npl(u.left) \geq npl(u.right)$ for any u.

$npl(NULL) = -1$

$npl(p) = \min(npl(u), npl(v)) + 1$





Lemma

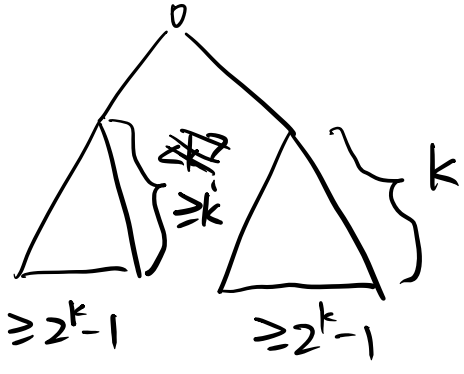
there are at most $\log_2(n+1)$ nodes the right path of a leftist tree with n nodes

# nodes on right path	# nodes in leftist tree
r	$\geq 2^r - 1 \Rightarrow n \geq 2^r - 1$, $r \leq \log_2(n+1)$

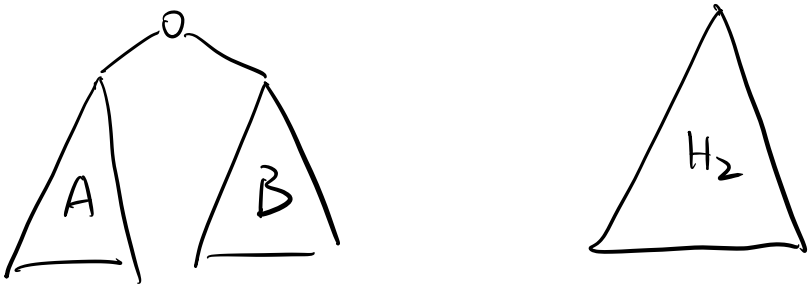
base $r=1$ $\quad \geq 2^1 - 1 = 1$

inductive hypothese $r=k$ $\quad \geq 2^k - 1$

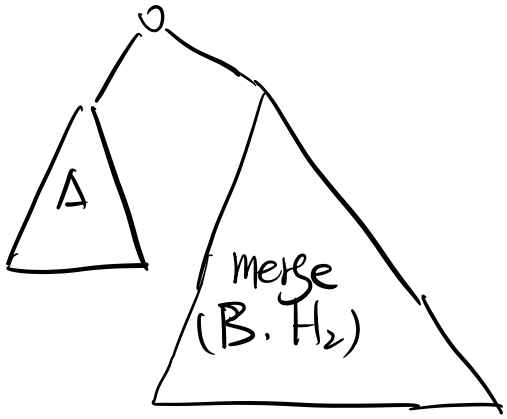
$r = k+1$ $\quad \geq 2^{k+1} - 1$



Merge



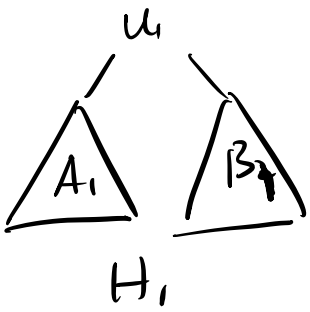
Assume $H_1.root \leq H_2.root$

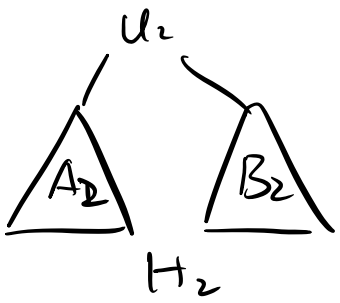


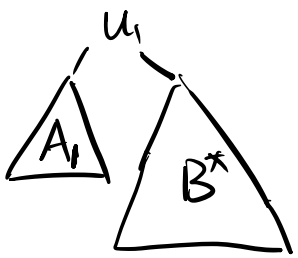
$\Rightarrow$ swap children if neccessary

Merge (H_1, H_2)

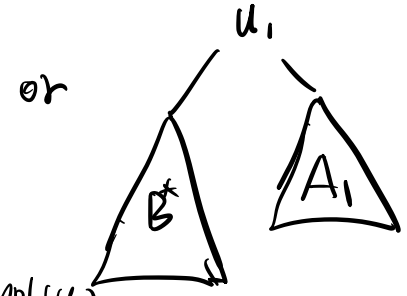
1. Define A_1, u_1, B_1 and A_2, u_2, B_2 as follow.





2. If $u_1 == NULL$
   return H_2
3. If $u_2 == NULL$
   return H_1
4. If $u_1.key < u_2.key$
5. $B^* = merge(B_1, H_2)$
6. $H =$ 

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



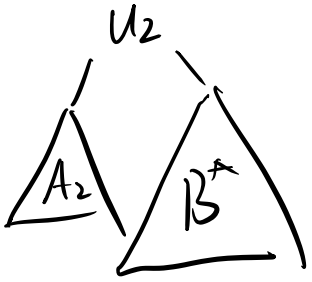
or otherwise.

update npl(u_1)
return H

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

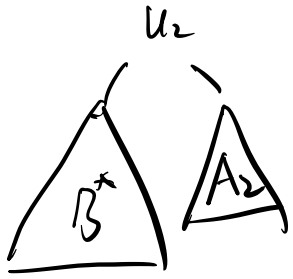
$B^* = Merge(H_1, B_2)$

$H =$



If - - -

or



Otherwise.

update npl(u_2)
return H

nodes on right paths of H_i

levels of recursion $\leq r_1 + r_2$

time of each level $O(1)$

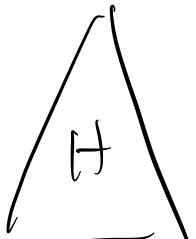
$n_1 + n_2$

merge $O(r_1 + r_2) = O(\lg n_1 + \lg n_2) = O(\lg n)$

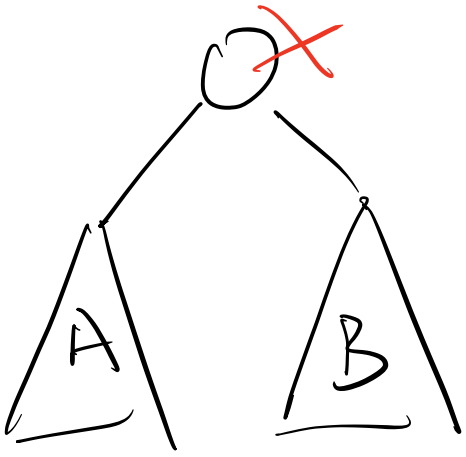
Insertion

merge $O(\lg n)$

(x)



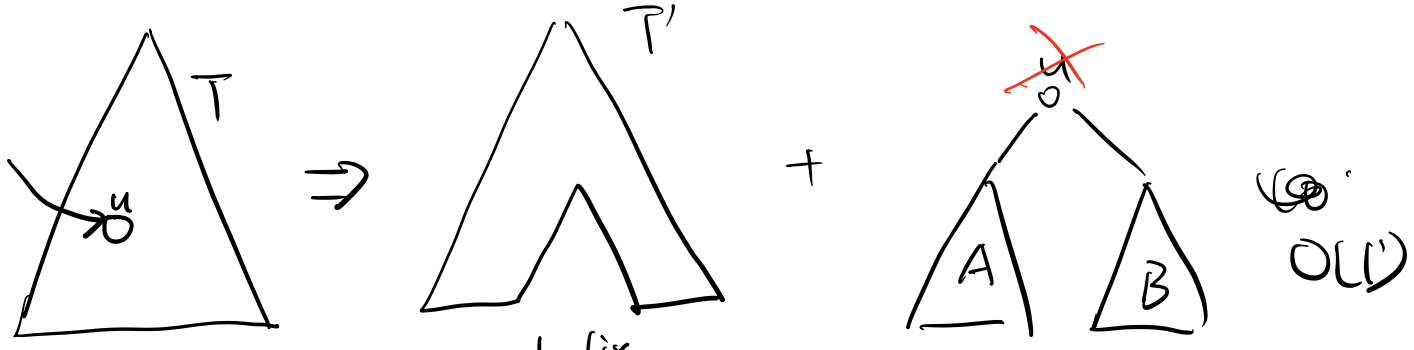
Deletemin



Merge (A, B)

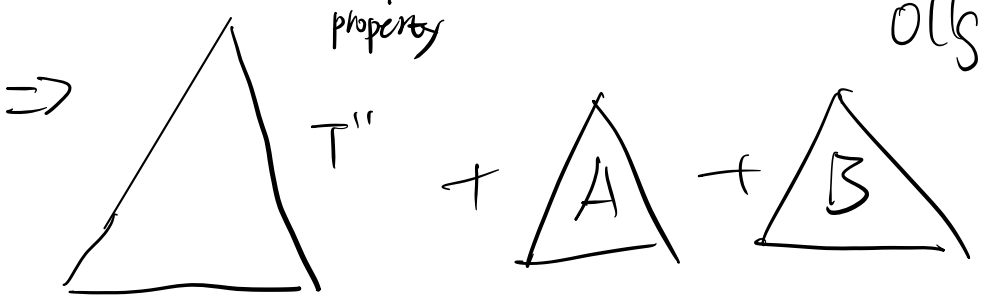
$O(\lg n)$

Delete $O(\lg n)$



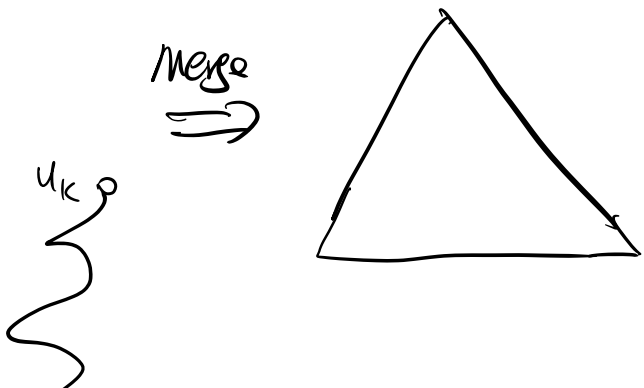
↓ fix the leftist property

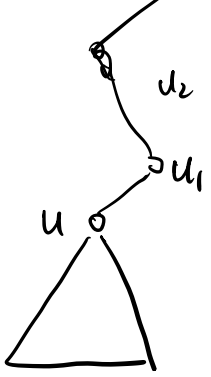
$O(\lg n)$



$O(\lg n)$

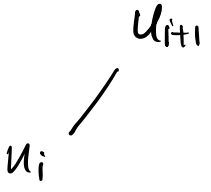
Merge ⇒





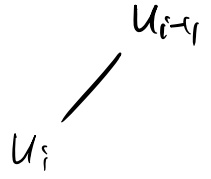
$npl'(u_1)=0 \Rightarrow u_1 = \dots$

$npl'(u_i) \rightarrow$ update $npl(u_{i+1})$



$npl'(u_i) \geq npl(u_{i+1}.right)$

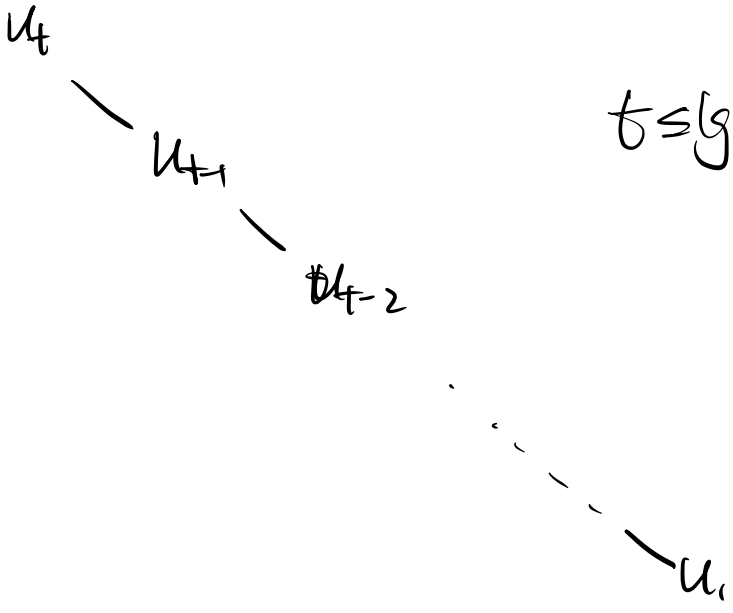
done



$npl'(u_i) < npl(u_{i+1}.right)$

swap

update $npl(u_{i+1})$



$t \leq \lg n$

Decreasekey $O(\lg n)$