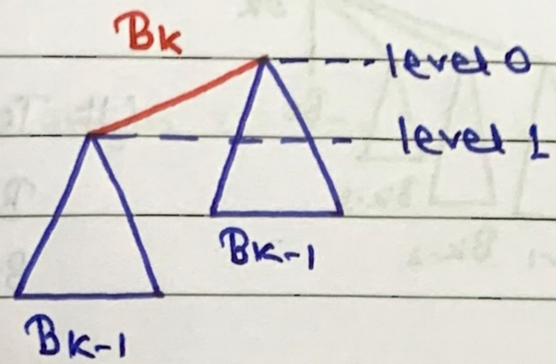
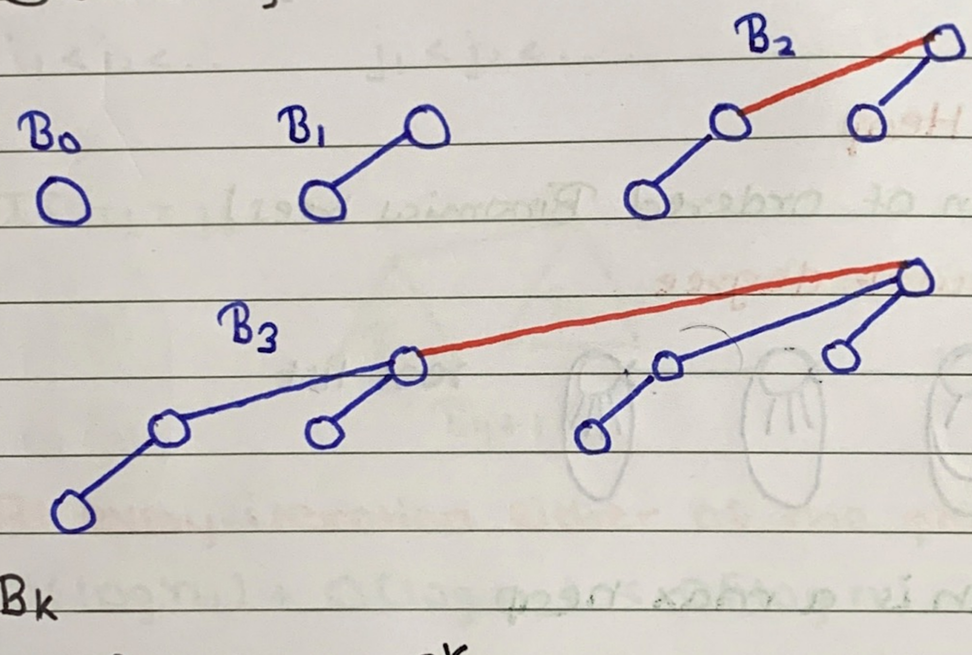
★ Binomial Heap

(unordered set of binomial trees)

1. max Heap
2. get max
3. meld
4. insert
5. extract max
6. increase key
7. delete key
8. build key





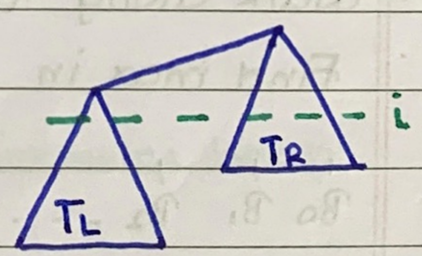
→ B_k

no. of nodes $n = 2^k$

Height $= k$

$\therefore h = O(\log n)$

Nodes at depth i : $\binom{k}{i}$

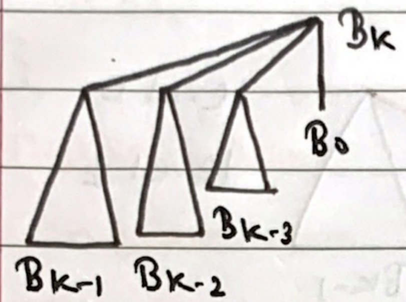


#(depth i) = #(depth $i-1$ in T_L) + #(depth i in T_R)

$\therefore \binom{k-1}{i-1} + \binom{k-1}{i} = \binom{k}{i}$ (# Pascal)

Deg (root) = k (# ${}^{k}C_1 = k$ or use induction)

→ max deg is for root in B_k



(# Telescope

$B_k = B_{k-1} + B_{k-1}$

$B_{k-1} = B_{k-2} + B_{k-2}$

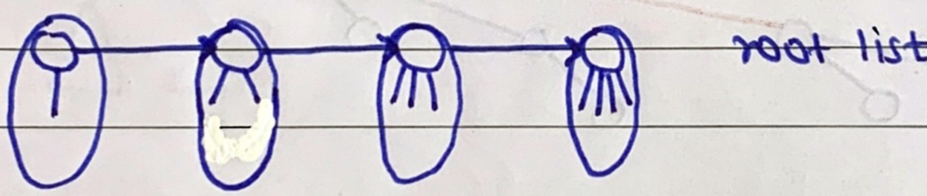
⋮

$B_1 = B_0 + B_0$)

★ Binomial Heap

(Collection of ordered Binomial trees)

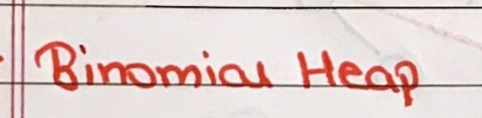
→ Ordered wrt degree



★ each is a max heap

→ walk along the root list

Find max in O(root list . si) (Naive)



If B_i present ith bit set to 1 else 0 (b_i = 0 or 1)

$\therefore n = \sum_{i=0}^{\lfloor \log n \rfloor} b_i 2^i$

classmate

Date
Page

★ $\lfloor \log n \rfloor + 1$ bits required to store n keys

maxdeg $\lfloor \log n \rfloor$ # $B_{\lfloor \log n \rfloor}$ exists

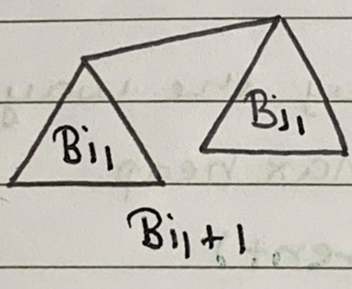
∴ Length of root list $\lfloor \log n \rfloor + 1 = O(\log n)$

★ Getmax : $O(\log n)$ (# length of root list)

★ Meld : $O(\log n)$

n_1	n_2
$B_{i_1}\ B_{i_2} \ldots$	$B_{j_1}\ B_{j_2}\ B_{j_3} \ldots$
$i_1 < i_2 < \ldots$	$j_1 < j_2 < \ldots$

If $i_1 = j_1$



→ Consolidation

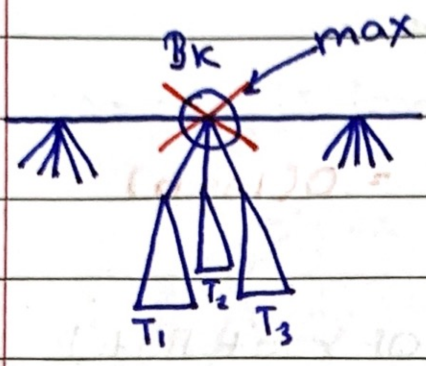
At every iteration either of the pointer moves by 1

$O(\log n_1) + O(\log n_2) \leq O(\log n)$ (# AM - GM)

★ Insert $O(\log n)$

assume other node to be a binomial heap

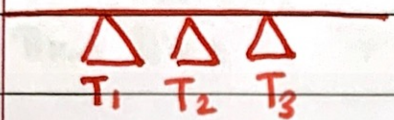
★ Extract max:



B_K : Deg : K

$O(K) \le O(\log n)$

max Deg of B_K is $O(\log n)$



new root list

★ Hence we have 2 binomial heaps meld in $O(\log n)$

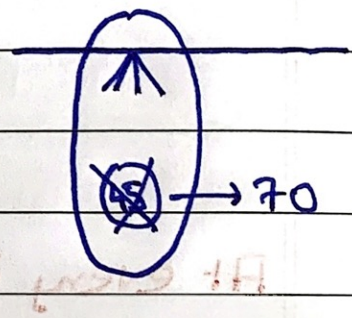
∴ Extract max : $O(\log n)$

★ Increase key:

Just increase the key the way we increase key in Max heap

(# compare with parent)

$O(h) \le O(\log n)$



★ Delete key

Increase key value to ∞

then extract max : $O(\log n)$