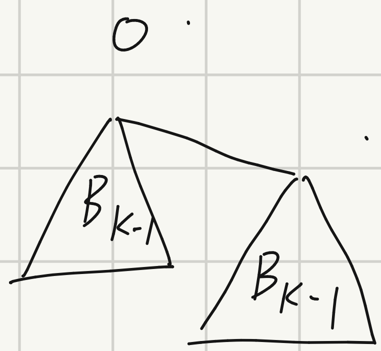
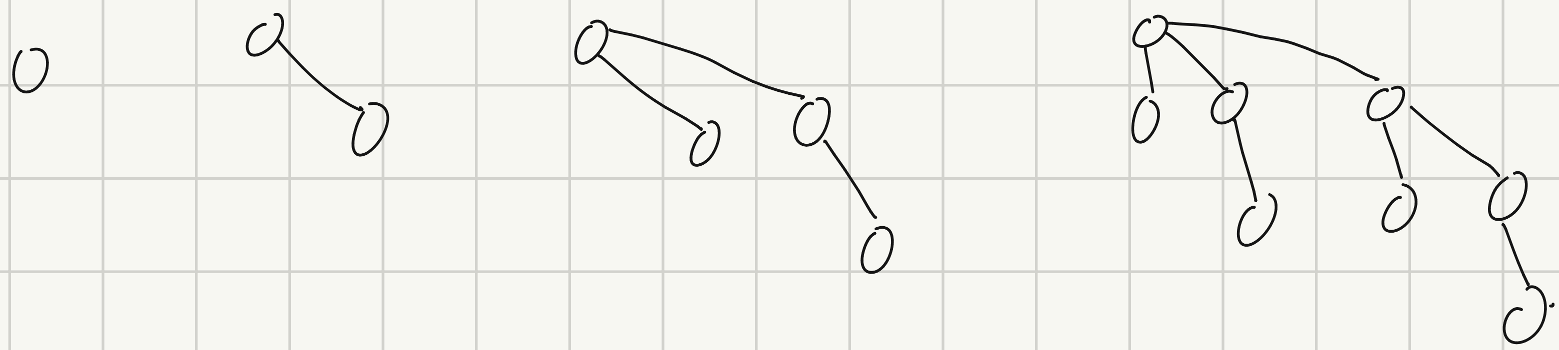
Binomial tree:

$h=0$ $\quad B_0$

$h=K.$ $\quad B_K.$



$B_0 \qquad B_1 \qquad B_2 \qquad B_3$



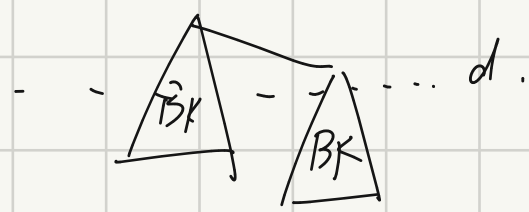
① # nodes in $B_K = 2^K$.

② # nodes at level d of $B_K = \binom{K}{d} = C_K^d = \frac{K!}{(K-d)!\,d!}$

base case $K=0$ true.

assume B_K. true

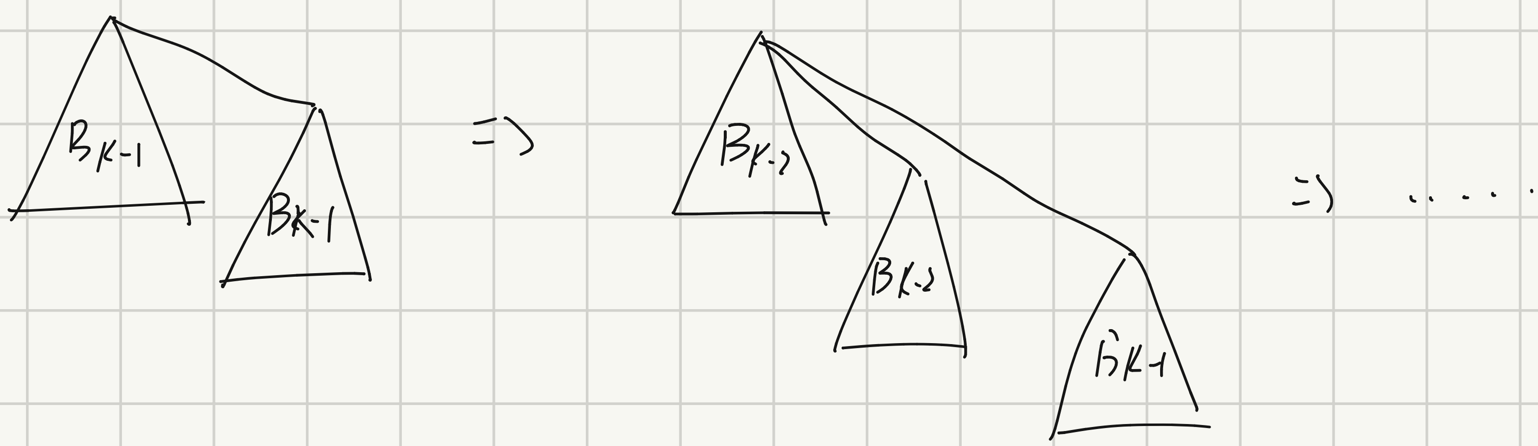
B_{K+1}



$\#\text{nodes} = \binom{K}{d} + \binom{K}{d-1}$

$= \binom{K+1}{d}$

③ the root of B_K has K subtrees: $B_0, B_1, \cdots, B_{K-1}$

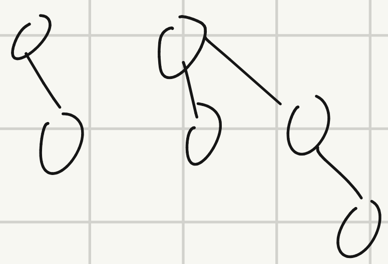


Binomial Heap:

A Binomial heap is a forest of
1. binomial trees of distinct height
2. each of which is in heap order

A Binomial heap with 6 nodes $\quad 6 \xrightarrow{\text{binary}} 110$

$B_2 \; B_1 \; B_0 \times 0.$



trees in a binomial heap with nodes $\leq \log_2^n$

→ their height $\log_2^n$

max # trees: $(1111\cdots11)_2$

$2^0 + 2^1 + \cdots + 2^K = n$

$2^K = n+1$

$K = \log_2^{n+1} \sim \log_2^n$

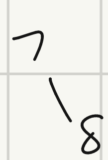
findmin:

$O(\log n)$: compare roots

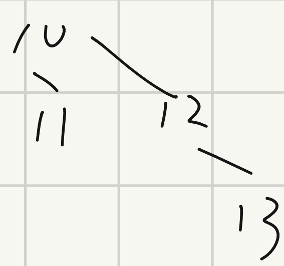
→ $O(1)$ get a pointer points minimum.

Insertion:

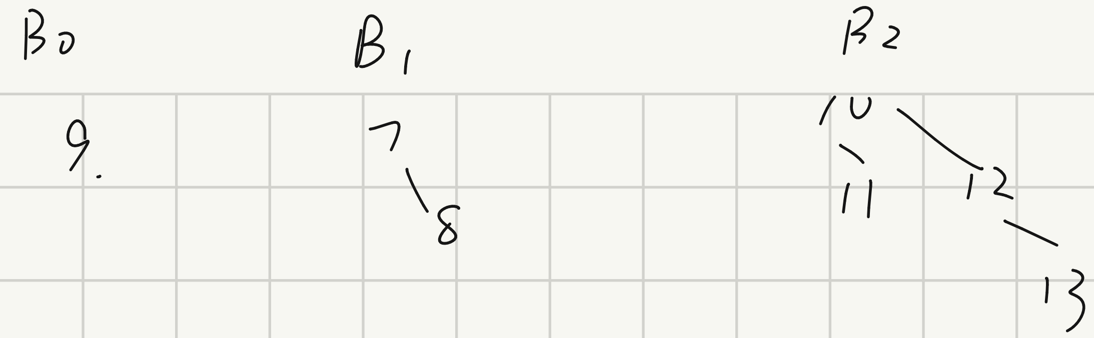
B_1



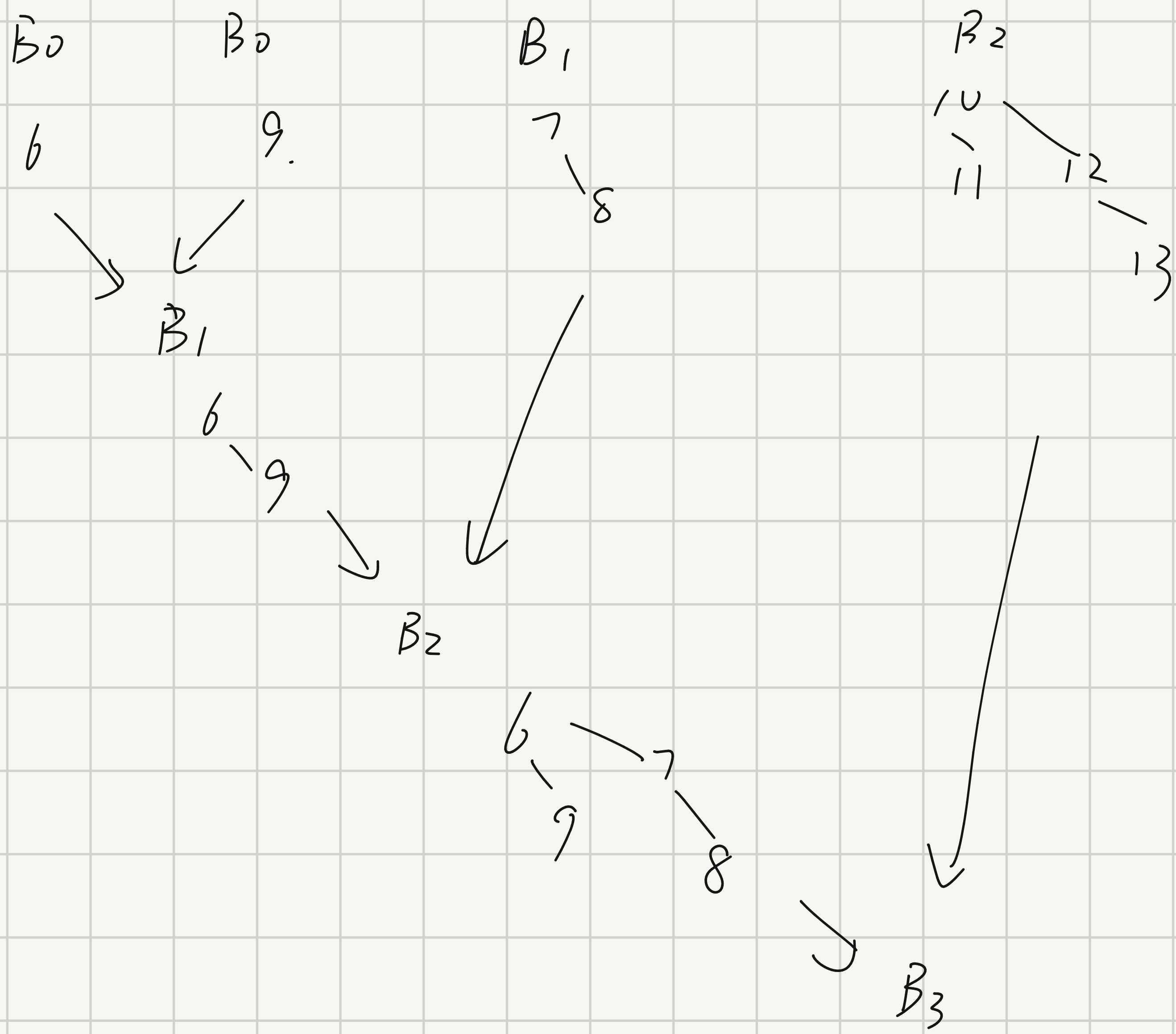
B_2



ins (9)



ins (6).



$110_2 \overset{9}{\Rightarrow} 111_2 \xrightarrow{6} 1000_2$.

insert (H, x)

1. add to H a B_0 with key x
2. $i=0$
3. while H has two trees of height i
4. combine them into B_{i+1}

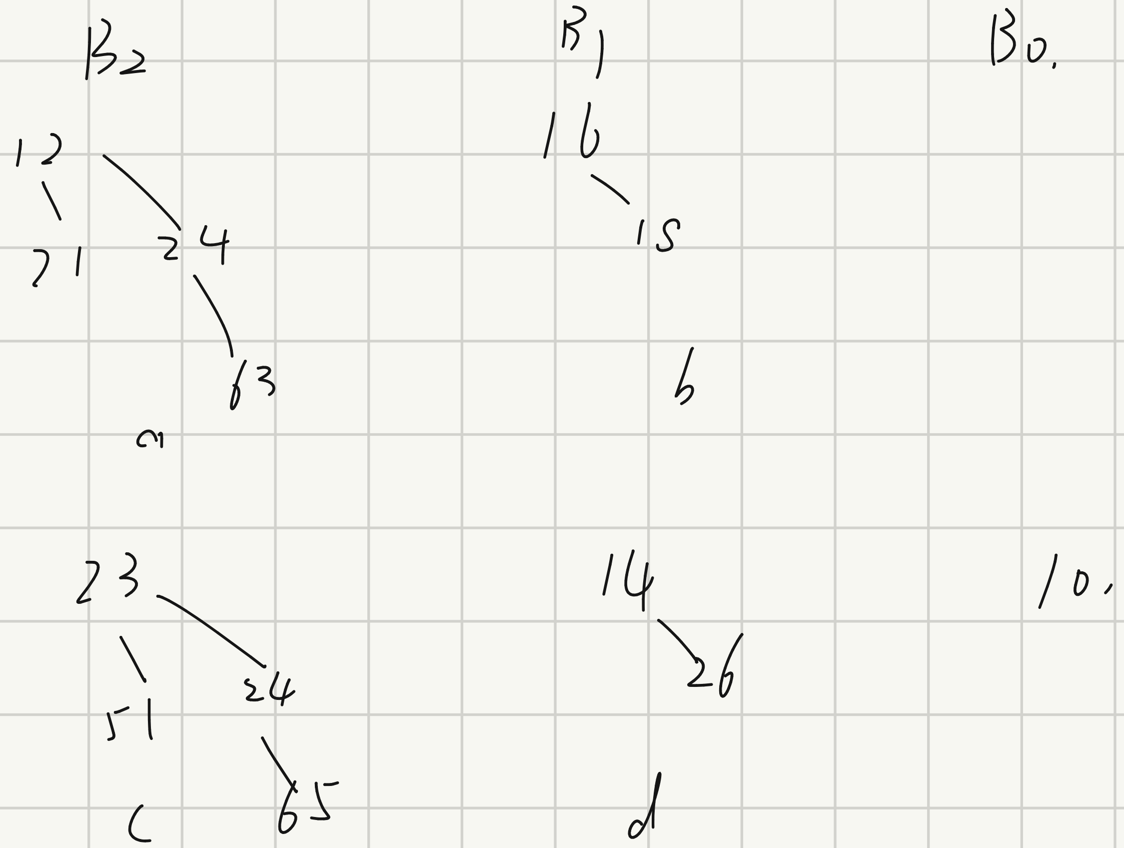
5 $i = i+1$

$O(\#\ \text{combines}) = O(\underbrace{\#\ \text{trees}}_{\text{each combination decrease \# tree}}\ \text{in } H+1) = O(\log_2 n)$

Merge

H1:

H2:



①. combine b. d to e

② combine two of (a. c. e)

110 and 111

$$\begin{array}{r} 110_2 \\ 111_2 \\ \hline 1001_2 \end{array}$$

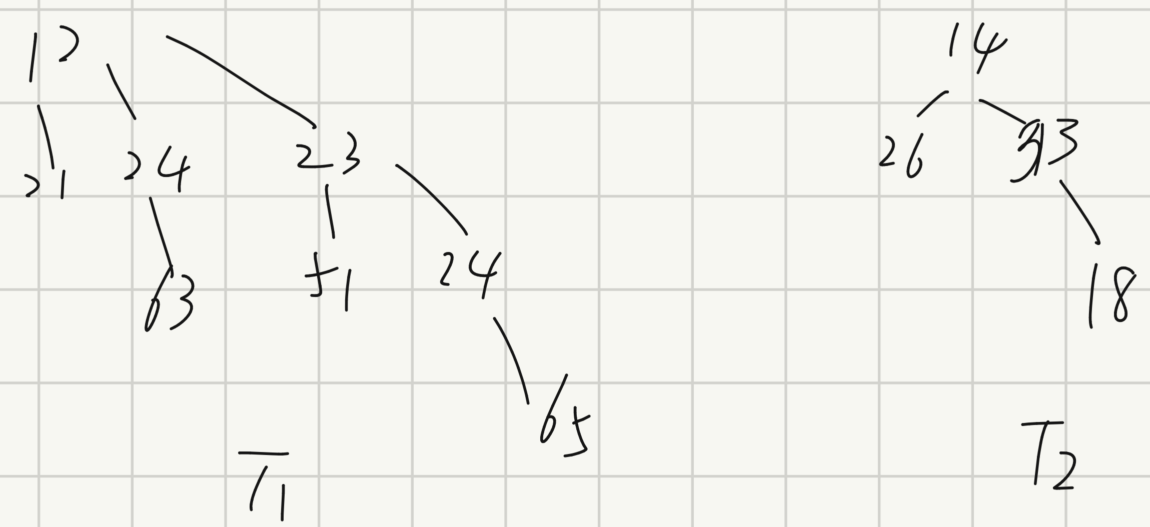
merge (H_1, H_2)

1. for $i=0$ to $\max\{\#\text{trees in } H_1, \#\text{ trees in } H_2\}$
2. if there are more than one trees of height i
3. combine them into a B_{i+1}

$O(\#\text{trees in } H_1 + \text{" " in } H_2) = O(\lg n_1 + \lg n_2) = O(\lg n)$

Deletemin.

H:



① get T_1 and delete 12 and get a new H' with B_0 B_1 B_2

② Merge H' with T_2.

deletemin (H)

1. find the B_K that contains the min $O(1)$
2. $H = H - B_K$ $O(1)$
3. $H' = B_K$ - root $O(K) = O(\lg n)$ (delete K pointers)
4. Merge (H, H')

decreaseKey (x, t)

$O(k) = O(\log n)$ Percolate height times.

Deletion.

decreasemin $(X, -\infty)$ + deletemin $O(\log n)$

make heap. $(k_1, \cdots k_n)$

1. $H(\phi)$
2. for $i=1$ to n. $O(n\log n)$
3. insert (H, k_i) $\uparrow$ not tight

starting with empty heap n consecutive insertions take $O(n)$ time in total.

Proof 1.

$$O(\#\text{ combines}) = O(\underset{\substack{\downarrow \\ B_0, B_1}}{\frac{n}{2}}, \underset{B_1, B_2}{\frac{n}{4}}, \cdots)$$

$$= O(n)$$

$\Phi(H) = \#$ trees in H

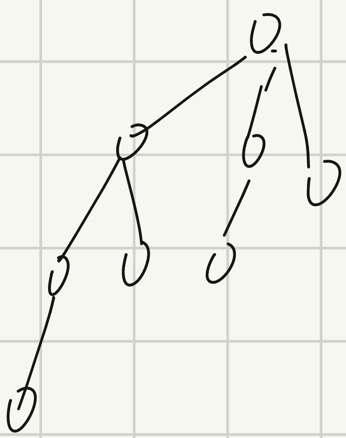
$\Phi(\text{empty}) = 0$ $\Phi(H) \geq 0$

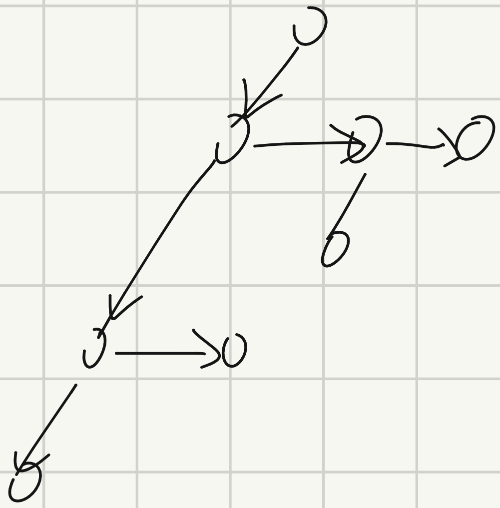
actual cost $= 1 + \#$ combines

$\Delta\Phi = 1 - \#$ combines

amortized cost = 2.

B_3





left child → next sibling

Inverted File Index

相交： $O(n_1+n_2) + O(n_1+n_3) + \cdots + O(n_1+n_k)$

$= O(kn_1 + \sum_{i=1}^{k} n_k)$

$= O(\sum_{i=1}^{k} n_k)$ ⇒ 线性了.

从短的排到.