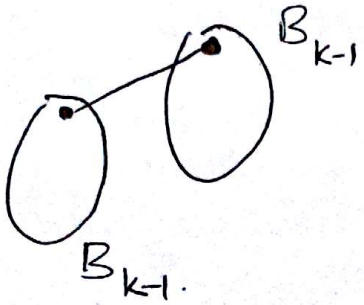
Binomial Heaps

- Collection of heap-ordered trees
- each tree is of special type called "binomial tree".

Def Binomial trees $B_0, B_1, B_2, \ldots$ are defined as

- B_0 is single node
- B_k = 

i.e. B_k is obtained by taking two copies of B_{k-1} and making the root of one a child of the root of the other.

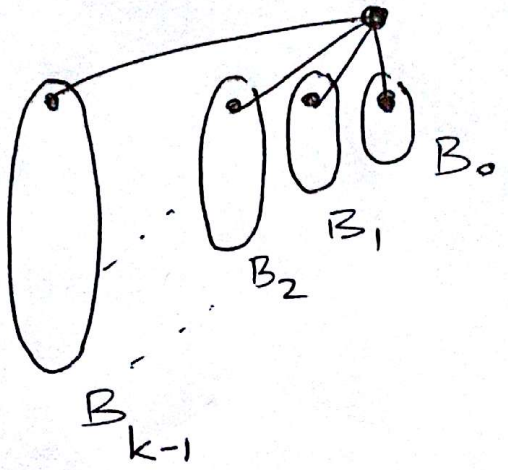
Ex.

B_0 B_1 B_2 B_3 - - -

Note B_k has

- 2^k nodes, height k.
- root has degree k.
-

B_k =



- there are exactly $\binom{k}{i}$ nodes at depth i, $i = 0, 1, 2, \cdots, k$ (hence the name "binomial tree).

Proof. By induction. Exercise.

Def. A binomial heap H is a collection of distinct, heap-ordered trees.

- $H = \{B_{i_1}, B_{i_2}, \cdots, B_{i_p}\}$ $0 \le i_1 < i_2 \cdots < i_p$.
- Each B_{i_j} is heap-ordered.
- roots are maintained in a linked list.

Lemma If a binomial heap H has n nodes then
- # trees in $H \leq \log n + 1$.
- height of all trees $\leq \log n$.

Proof Suppose $H = \{B_{i_1}, \ldots, B_{i_p}\}$ $0 \leq i_1 < i_2 \ldots < i_p$.

Then $i_p \geq p-1$ and $|H| \geq |B_{i_p}| = 2^{i_p} \geq 2^{p-1}$

$\therefore$ $p \leq \log n + 1$,

$i_p \leq \log n$.

Now we show how to implement various operations on binomial heaps.

Find. Minimum
- $O(1)$ time
- Maintain pointer to minimum root.

Union. $O(\log n)$ time.

Ex $H_1 = \{B_0, B_2, B_3, B_8, B_{10}\}$

$H_2 = \{B_0, B_1, B_5, B_8, B_{12}\}$

~~B_0~~ ~~B_1~~ ~~B_2~~ ~~B_3~~ B_4 B_5 ~~B_8~~ B_9 B_{10} B_{12}

~~B_0~~ ~~B_1~~ ~~B_2~~ ~~B_3~~ ~~B_8~~

Thus $H_1 \cup H_2 = \{B_4, B_5, B_9, B_{10}, B_{12}\}$

Running time = # trees in H_1 and H_2

$\leq \log|H_1| + 1 + \log|H_2| + 1$

$\leq O(\log n)$ if $n = |H_1| + |H_2|$.

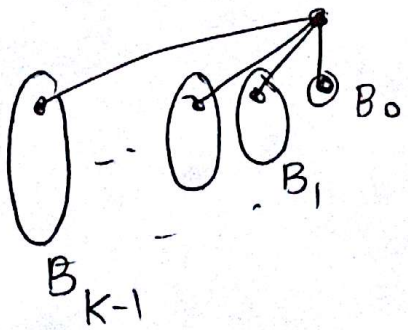
Note. - Merge two trees B_i, B_i into single tree B_{i+1}.

- Make sure that heap order is preserved.

Delete-Min

- Suppose the minimum occurs as root of tree B_k in heap H.

- $B_k =$



- Delete root of B_k. This gives copies of subtrees $B_0, B_1, \cdots, B_{K-1}$.

- form the union $H \setminus B_k \cup \{B_0, B_1, \cdots, B_{K-1}\}$

- $O(\log n)$ time.

Decrease. Key $O(\log n)$ time (= height of a tree) maximum

- Decrease key and move it upward if necessary to preserve heap. order.

Delete (arbitrary node).

- Decrease key to $-\infty$
- Then Delete. Minimum.

Running time = $O(\log n)$.

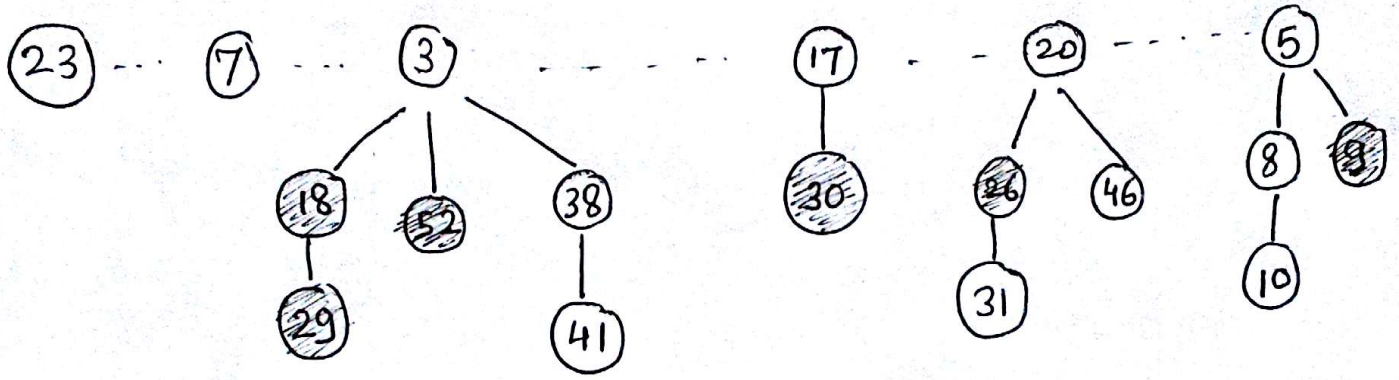
Insert

- Insert as single node tree B_0.
- Take union $H \cup B_0$.
- $O(\log n)$ time.

Fibonacci Heaps

- Fibonacci heap H is a collection of heap-ordered trees.
- Each node is marked or unmarked.

Ex.



- Maximum degree of any node is $D(n)$, $|H| = n$.
- Potential function

$$\phi(H) = t(H) + 2 \cdot m(H)$$
$$= \#\text{trees} + 2 \cdot \#\text{marked nodes}.$$

Note

- Involves cutting nodes from their parents.
- A node is marked iff it has lost exactly one child.
- Difficult part is to show that $D(n) \leq O(\log n)$.

Insert - Insert a single node as a new tree (unmarked).

- ~~Actual cost~~

Amortized cost = Actual cost + $\Delta\phi$

= 1 + 1

= 2.

Find. Minimum

- O(1) time.
- Maintain pointer to minimum root.

Union

- Given two heaps, form a new heap by simply combining the two root lists (i.e. taking union of two collections of trees).
- O(1) time. No change in potential.

Note. - "naive", "lazy" so far.

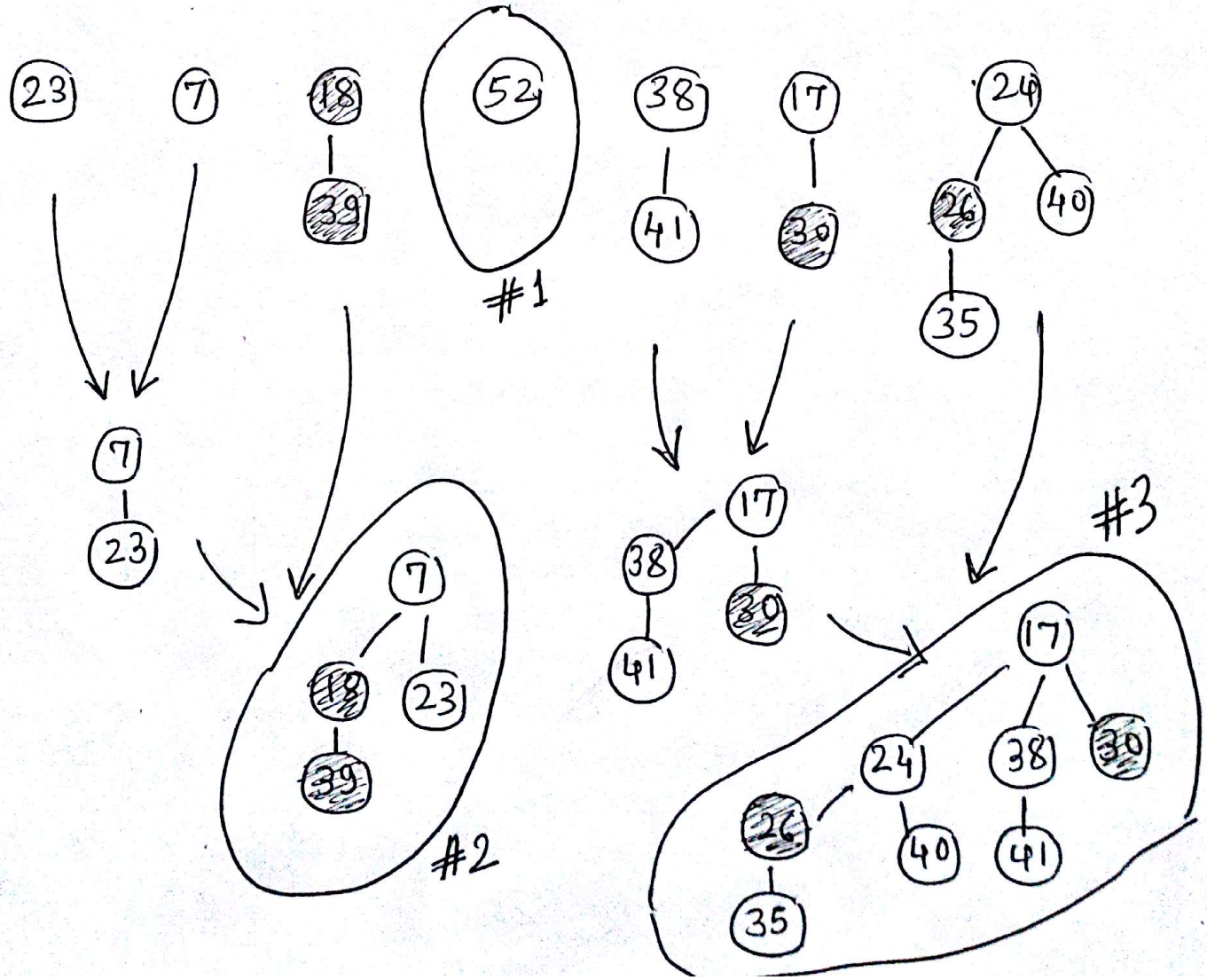
- "post-poning" work
- real work done next.

Delete-Min

- Consider the tree whose root is minimum.
- Remove that root and add its subtrees to the collection.
- Consolidate (let "rank" of a tree be the degree of its root).

Repeatedly, if there are two trees of rank r then combine them into single tree of rank $r+1$ (ensure heap order).

Ex.



Note. There are three trees, of rank 0, 2, 3 after consolidation.

Amortized cost = Actual cost + $\Delta\phi$

$= D(n) + k + (-k)$

- $D(n)$: # trees added
- k: # mergings
- $(-k)$: decrease in # trees

$= D(n)$.

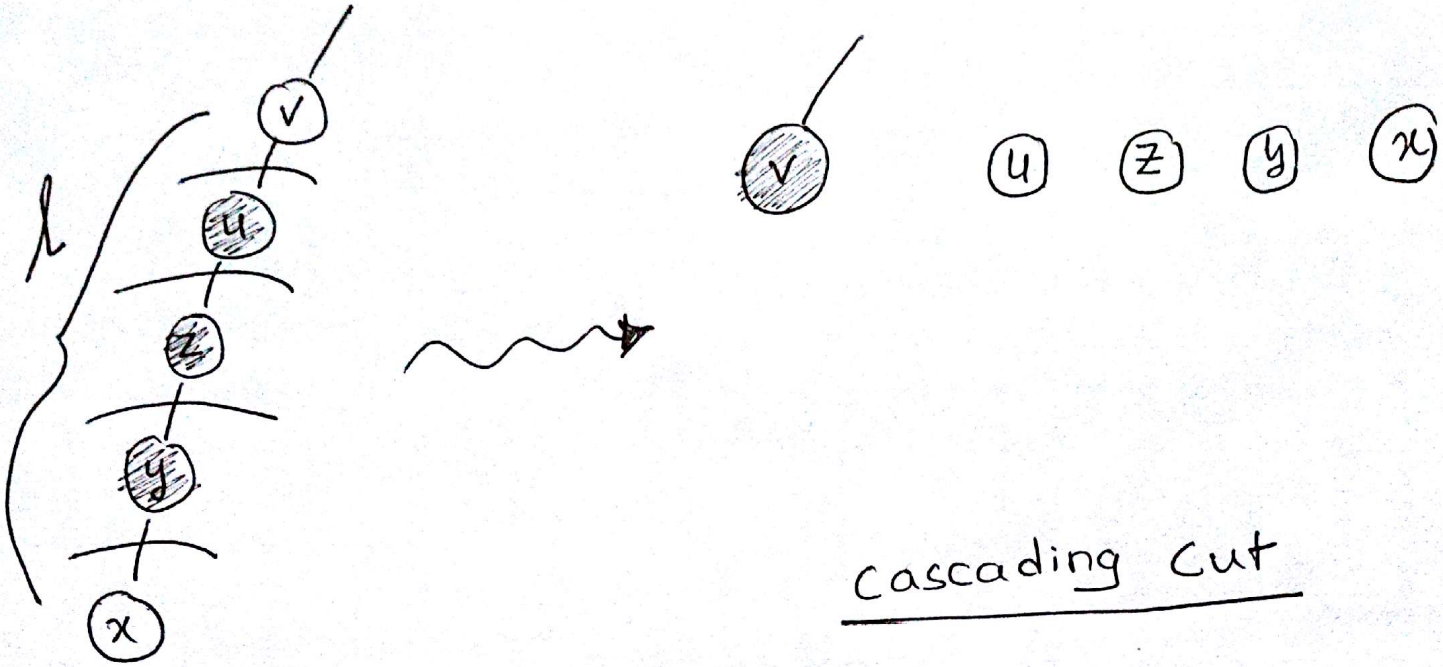
Note: Work involved in merging trees is "paid for" by decrease in # trees.

Decrease Key on node x.

- Decrease Key on x. If heap order is preserved, done.

- Otherwise
  - Cut x from its parent y, unmark x, add x as single-node tree in the heap.
  - If y is marked, cut y from its parent, unmark y, add y as single node tree.

Cascading cut - Repeat this upwards from y to the root.



cascading cut

↳ { - If y was unmarked, mark it and stop the cascading cut.

Note: The mark/unmark scheme retains the property that "a node is marked iff it has lost exactly one child.".

Let l = length of the "cascade". ($l=3$ above)

Amortized cost = Actual cost + $\Delta\Phi$

$= l + (l+1 + (-2(l-1)))$

↑ # trees increased

↑ # marked nodes decreased

$= 3$!!!

Delete (any node)

Amortized cost.

- Decrease Key to $-\infty$. — 3
- Delete Minimum. — $D(n)$

$O(D(n))$.

Bounding the maximum degree $D(n)$:

Lemma Let x be a node in a Fibonacci heap with degree k.

Let $y_1, y_2, \ldots, y_k$ be its children in the order in which they got attached to x.

Then $\text{degree}(y_i) \geq i-2$ for $2 \leq i \leq k$.

Proof when y_i got attached to x, degree of x was $\geq i-1$ and moreover degrees of x and y_i were equal

($\because$ the "attachment" happens only while merging trees and two trees are merged only if they have same rank).

Since then y_i may have lost one child (it couldn't have lost two or more children, since otherwise it would have been cut off from x).

$\therefore \deg(y_i) \geq i-2.$ ▨

Def size(x) = Number of nodes in subtree rooted at x.

Lemma For any node x of degree k,

$$\text{size}(x) \geq F_{k+2}$$ ($(k+2)^{th}$ Fibonacci Number)

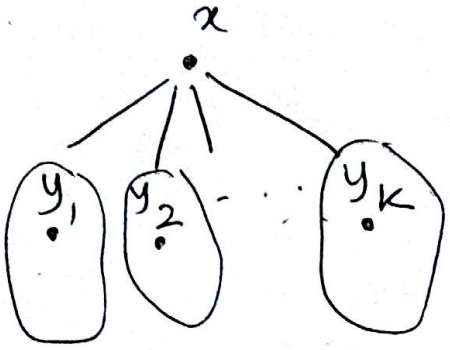
Proof By induction.

F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, ...

$F_0 = 0, \quad F_1 = 1, \quad F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$.

The claim is correct for $k=0$, since, size(x) ≥ 1 and $F_2 = 1$.

For $k \geq 1$



$$\text{size}(x) = 1 + \text{size}(y_1) + \sum_{i=2}^{k} \text{size}(y_i)$$

$$\geq 1 + \underbrace{1}_{=} + \sum_{i=2}^{k} F_i$$

$$= 1 + (F_0 + F_1) + \sum_{i=2}^{k} F_i$$

$$= 1 + \sum_{i=0}^{k} F_i$$

$$= F_{k+2}.$$

$\because$ degree of y_i is $\geq i-2$, inductive hypothesis

Prove by induction. exercise.

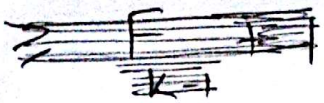
Lemma $\forall k \geq 0, \quad F_{k+2} \geq \phi^k$ where

$$\phi = \frac{1+\sqrt{5}}{2} \approx 1.6.$$

Proof Claim correct for $k = 0, 1$.

Inductively,

$$F_{K+2} = F_{K+1} + F_K$$



$$\geq \phi^{k-1} + \phi^{K-2} \quad \text{by inductive hyp.}$$

$$= \phi^{K-2}(\phi + 1)$$

$$= \phi^{K-2} \cdot \phi^2 \qquad \because \phi = \frac{1+\sqrt{5}}{2},$$

$$= \phi^K. \qquad 1 + \phi = \phi^2.$$

This shows that in a ~~bin~~ Fibonacci heap, with n nodes, if maximum rank of a tree is $D(n)$, then that tree alone contains $\geq (1.6)^{D(n)-2}$ nodes. Hence

$$\boxed{D(n) \leq O(\log n)}$$