Binomial Heaps

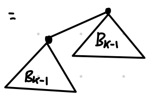
: A way to implement mergeable heaps. Useful in scheduling algorithms and graph algorithms.

Operations:

- Make-Heap()
- Insert(H, x), where x is a node into heap H
- Minimum(H)
- Extract-Min(H)
- Union(H_1, H_2) : merge H_1 and H_2, creating a new heap
- Decrease-key(H, x, k): decrease x.key to k. ($k \le x.key$)

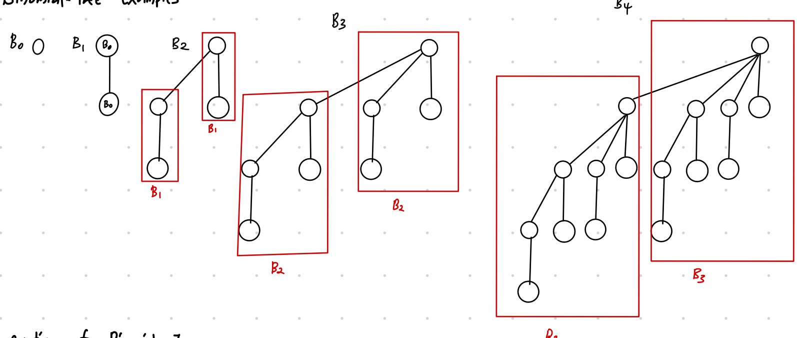
Definitions:

- Binomial Heap: Collection of binomial trees
- Binomial Tree:
  - Definition is inductive
  - Base case: B_0 = single node is a binomial tree
  - Inductive step : B_k =



is a binomial tree

Binomial-Tree Examples



In B_4

Depth	# of nodes
0	1
1	4
2	6
3	4
4	1

Properties of Binomial Trees

- Lemma: For the binomial tree B_k,

① There are 2^k nodes

② Tree height is k

③ $\binom{k}{i}$ nodes at depth i, $i = 0, 1, \dots, k$ [binomial coefficient]

④ Root has degree k, other nodes have smaller degree. i^{th} child of root is a root of subtree B_i, where $i = k-1, k-2, \dots, 0$

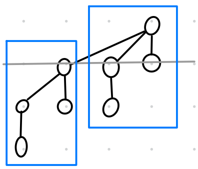
Recap: $\binom{k}{i} = \frac{k!}{i!(k-i)!}$

Proof of ③

: Let $D(k,i)$: # of nodes at depth i in B_k

Want to show $D(k,i) = \binom{k}{i}$

Let's say $k=3$, and $i=1$



depth 1 in B_2
: $D(k-1, i)$

depth 0 at B_2 : $D(k-1, i-1)$

We can see $D(k,i) = D(k-1,i) + D(k-1,i-1)$

$$= \binom{k-1}{i} + \binom{k-1}{i-1} = \frac{(k-1)!}{i!(k-1-i)!} + \frac{(k-1)!}{(i-1)!(k-i)!} = \binom{k}{i}$$

④ Root degree of B_k = 1 + root degree of B_{k-1}

$= 1 + k - 1 = k$.

Now, Binomial Heaps

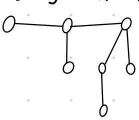
: Set of binomial trees satisfying binomial-heap properties

① Heap ordered: Root of a binomial tree has the smallest key in that tree.

② Set includes at most one binomial tree whose root is a given degree.

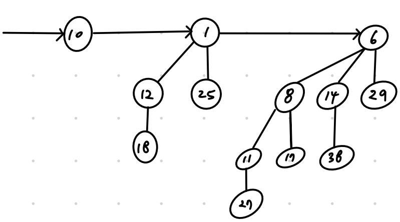
: It implies that binomial heaps with n nodes has at most $\lfloor \log n \rfloor + 1$ binomial trees. It's because each tree has 2^k nodes $= n$.

Let's say $n=8$, then $\lfloor \log 8 \rfloor + 1 = 4$ (Then binomial heap with 8 nodes has at most 4 binomial trees)



Hence, $n = \sum_{i=0}^{\lfloor \lg n \rfloor} b_i 2^i$

Representing Binomial Heaps



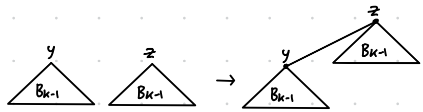
: called the root list

Each node is represented by a structure

parent	
key	
degree	
child	sibling

Linking Two Binomial Trees.

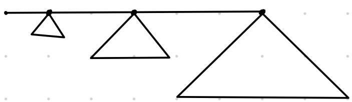
```
Pseudocode
Link(y, z)
  p[y] := z
  Sibling[y] := child[z]
  Child[z] := y
  degree[z] := degree[z]+1
```



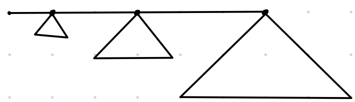
Union binomial heaps

$H_1, H_2 \longrightarrow H_1 \cup H_2$

H_1 =



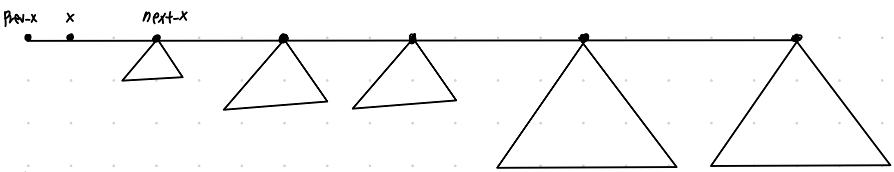
H_2 =



First, simply merge two root lists by root degree

Then, traverses the new root with prev_x, x, next_x pointer

Depending on what x, next-x, and sibling[next-x] point to, Union links trees with the same root degree.

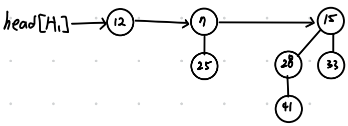


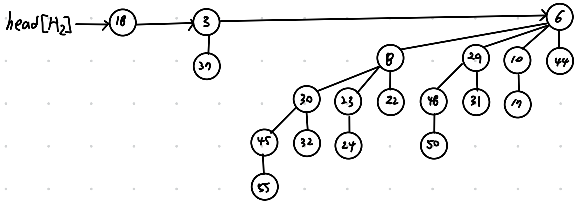
Pseudo code :

```
Union(H1, H2)
    H := new heap;
    head[H] := merge(H1, H2);          /* simple merge of root lists */
    if head[H] = NIL then return H fi;
    prev-x := NIL;
    x := head[H];
    next-x := sibling[x];
    while next-x ≠ NIL do
        if (degree[x] ≠ degree[next-x]) or
          (sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x]) then
Cases 1,2 {   prev-x := x;
              x := next-x
        else
            if key[x] ≤ key[next-x] then
Case 3 {        sibling[x] := sibling[next-x];
                Link(next-x, x)
            else
Case 4 {        if prev-x = NIL then head[H] := next-x else sibling[prev-x] := next-x fi
                Link(x, next-x);
                x := next-x
            fi
        fi;
        next-x := sibling[x]
    od;
    return H
```

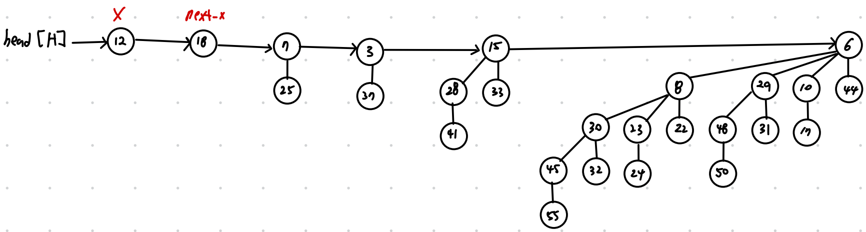
: $O(\lg n)$

Union Example

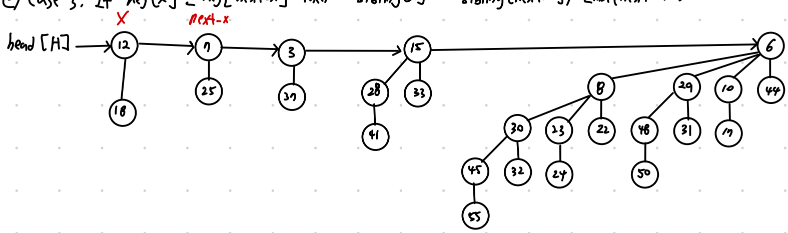




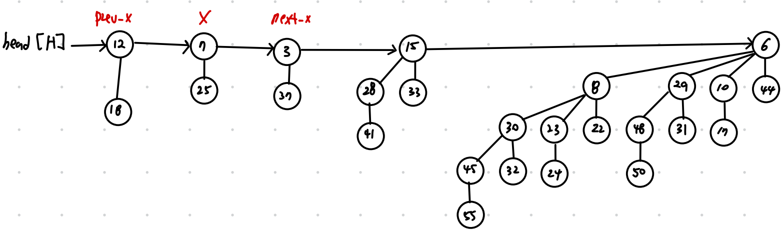
① Merge



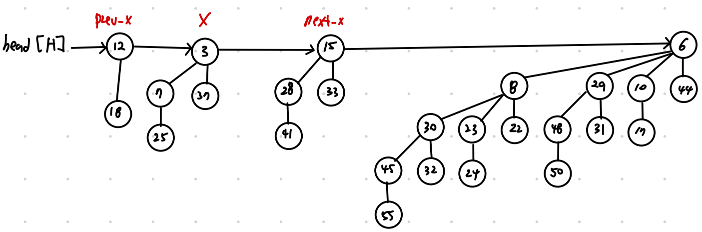
② Case 3: If key[x] ≤ key[next-x] then sibling[x] := sibling[next-x]; Link(next-x, x)



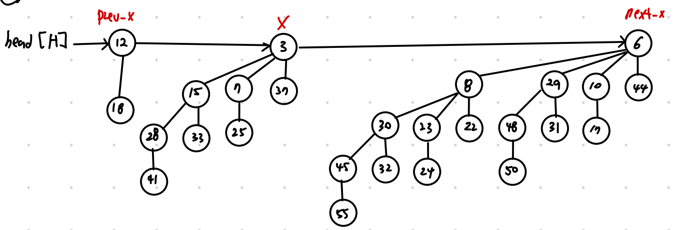
③ Case 2: sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x] then prev-x := x, x := next-x



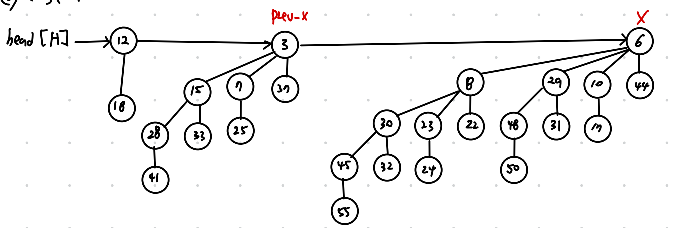
④ Case 4



⑤ Case 3 : If key[x] ≤ key[next-x] then sibling[x] := sibling[next-x]; Link(next-x, x)



⑥ Case 1 : degree[x] != degree[next-x]



Terminate : next-x = NIL