Recap

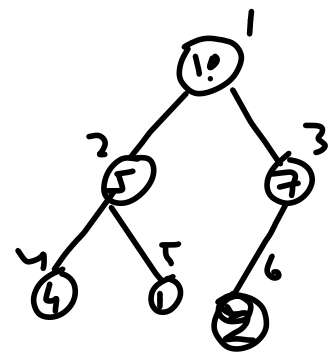
- Array (static)
- Linked-list
- Array (Dynamic)
- Stack
- Queue
- Heap (Max)
- Fibonacci Heap
-

Heap

Max-heap

$\forall x$

$A[parent(x)] \geq A[x]$



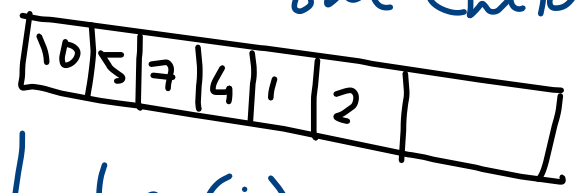
- Find max → O(1)
- Extract max
- Increase key
- Insert

} O(log n)

Priority Queue

Binary tree representation

- parent
- left child
- right child



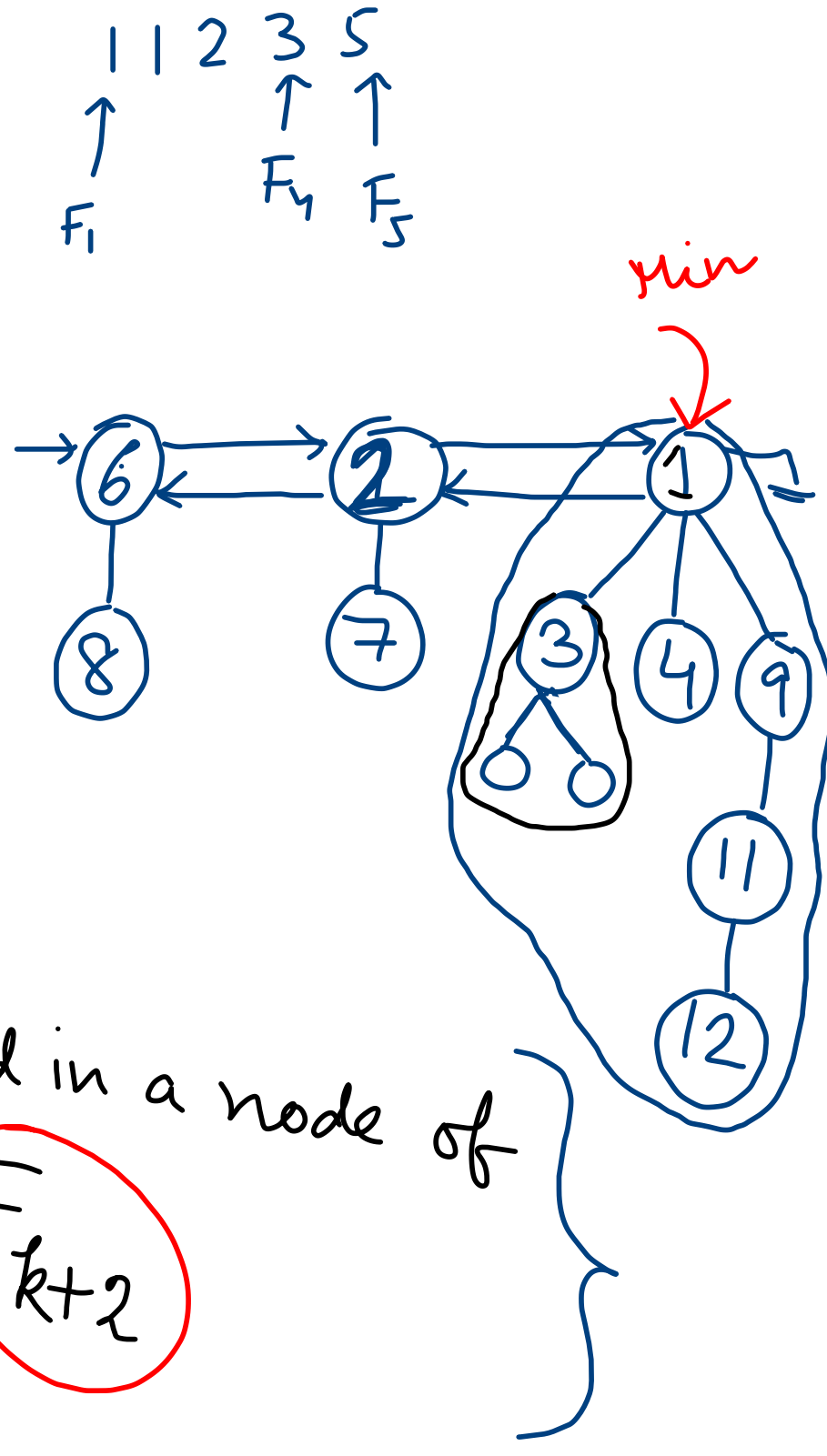
$lc(i) = 2i$

$rc(i) = 2i + 1$

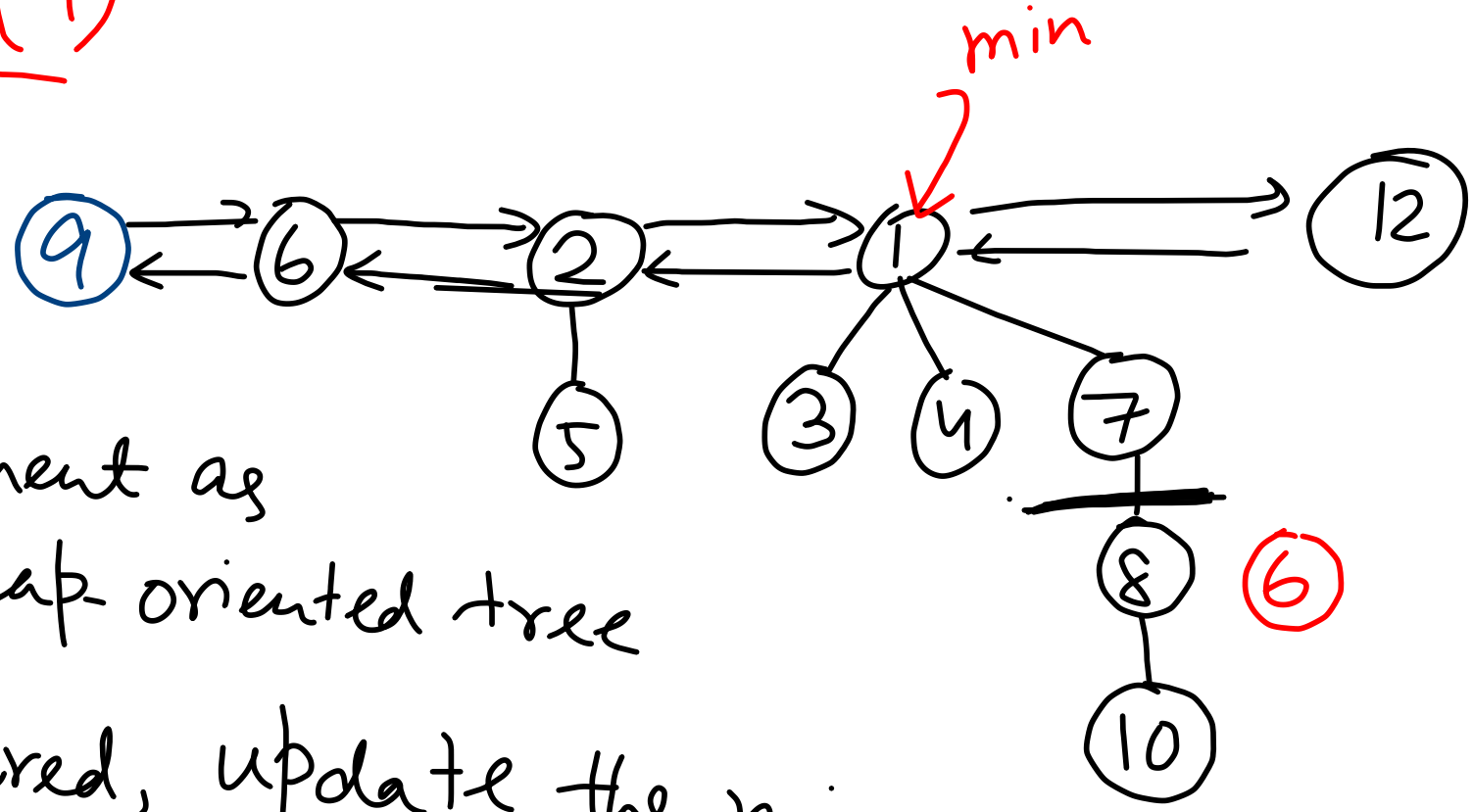
$parent(i) = \lfloor \frac{i}{2} \rfloor$

Fibonacci Heap

- Collection of trees satisfying min-heap.
- Roots are connected via doubly linked-list.
- Property: Size of a sub-tree rooted in a node of degree k is at least F_{k+2} (# childs)
- Operation: Lazy manner



Insert(9)



- Add the element as a new heap-oriented tree
- If required, update the min.

O(1)

Amortized Cost

Decrease_Key()

(Case1) heap ordering in-tact: Do nothing
(Case2) Else: Cut the node & make it a new tree

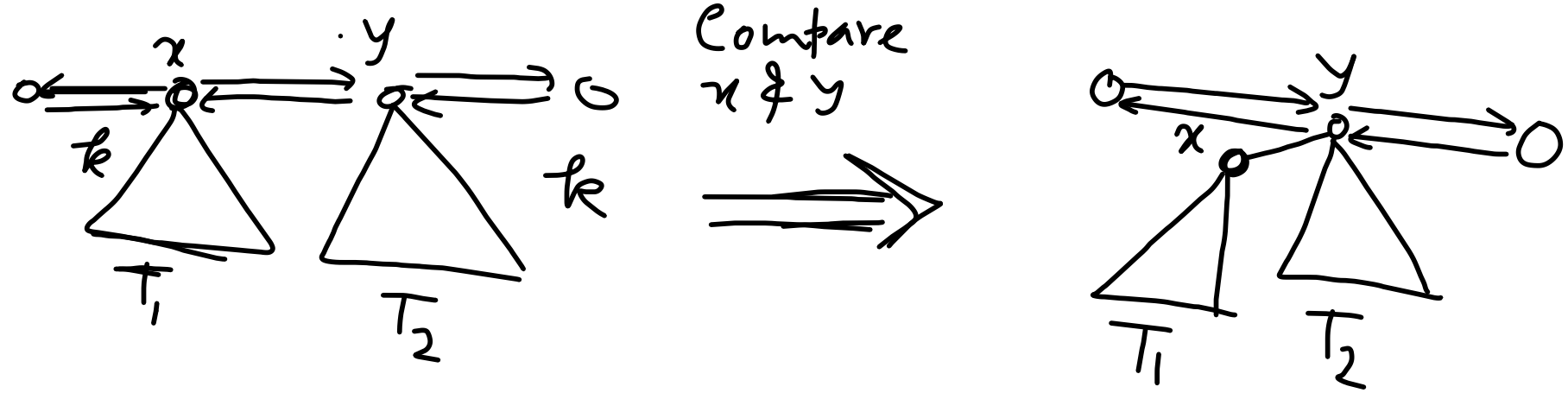
O(1)

- If parent is marked, cut that parent as well.
- Else, mark the parent

Extract_Min

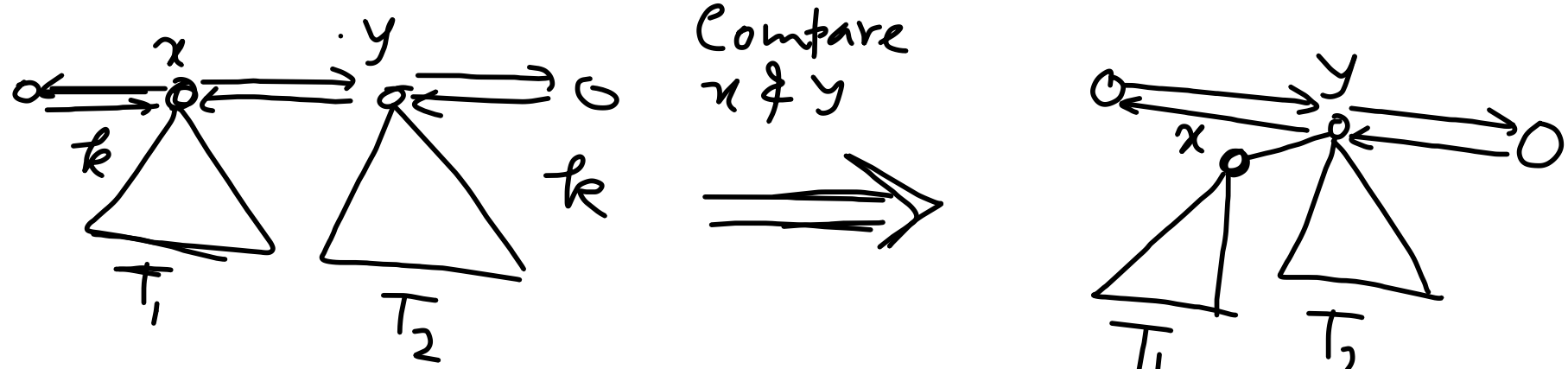
- Remove S^* (S^* pointed by Min)
- Put all children of S^* as roots of new tree in the collection.
- Scan the entire list of roots to find min^n & set Min pointer accordingly.
- Do some cleaning operation

- As long as there are two trees whose roots have degree k, we merge them to obtain one-tree of degree (k+1).



→ $O(\log n)$

- As long as there are two trees whose roots have degree k, we merge them to obtain one-tree of degree $(k+1)$.



Amortized

k-operation $O(nk)$

Cost $\rightarrow O(n)$

$a_1 \rightarrow$ insert $b_1 \rightarrow$ decrease_key

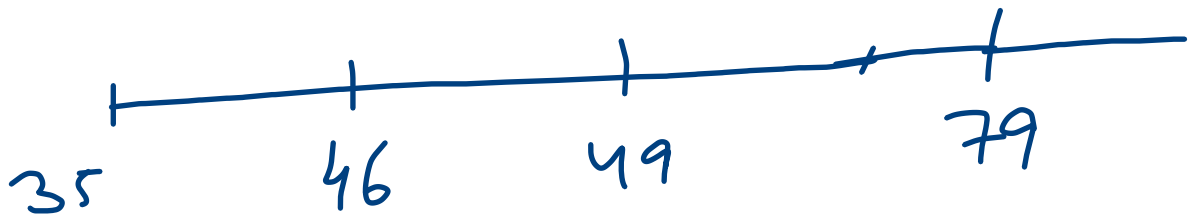
$c_1 \rightarrow$ extract_min

$O(a_1 + b_1 + c_1 \log n)$

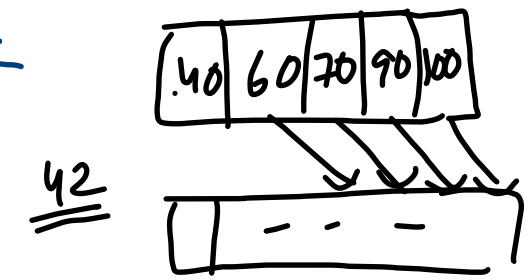
Runway Reservation System

- Airport with single runway
- Reservations for future landing
- Reserve request specifies landing time t.
- Add t to the set R if no other landing is scheduled within k-minutes.

t = 4:00 PM
k = 5 min
$R \rightarrow$ check whether (3:55 - 4:05)



$k = 3$



Req(41) → Yes
Req(43) → No

Conditional Insert

Unsorted Array

Insert → O(1)
Condition → O(n)

Sorted Array

Condition → O(log n)
Insert → O(n)

Unsorted Linked-list

O(n)

Sorted Linked-list

O(n)

Searching

Linear: O(n)
Binary: O(log n)

$T(n) = T(n/2) + 1$

$n = 2^k$

$T(2^k) = T(2^{k-1}) + 1$ - -

$\Rightarrow O(k)$

Binary Search Tree

Pointer

node x

↳ key(x)
↳ parent(x)
↳ left(x)
↳ right(x)

$\forall x$, if y is in the left sub-tree of x then

$$key(y) \leq key(x)$$

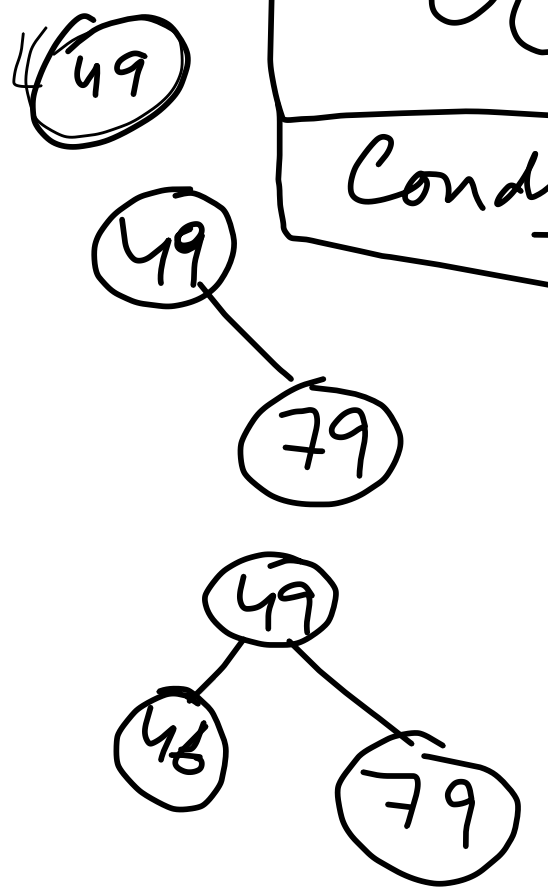
if y in right subtree of x
$key(y) \geq key(x)$

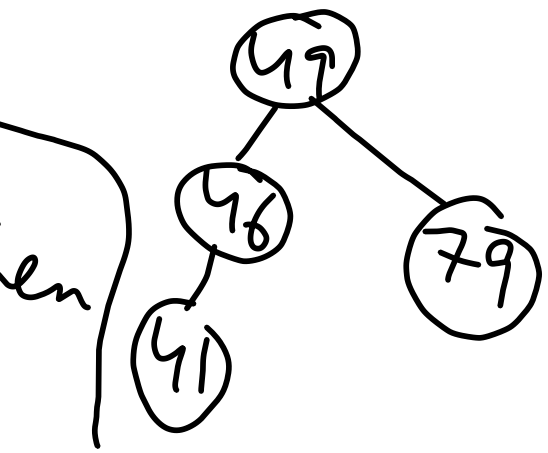
Insert (49):

Insert (79):

Insert (46):

Insert (41):





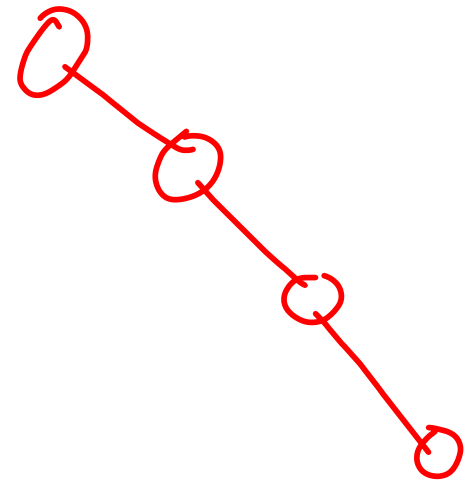
Height → h

$O(h)$

Conditional Insert

$\lceil h \rceil$

Problem



Tree is un-balanced

↳ Operation $h = n$

↳ $O(n)$

↓

Balanced Binary Search Tree

↳ AVL tree

↳ Red-Black tree