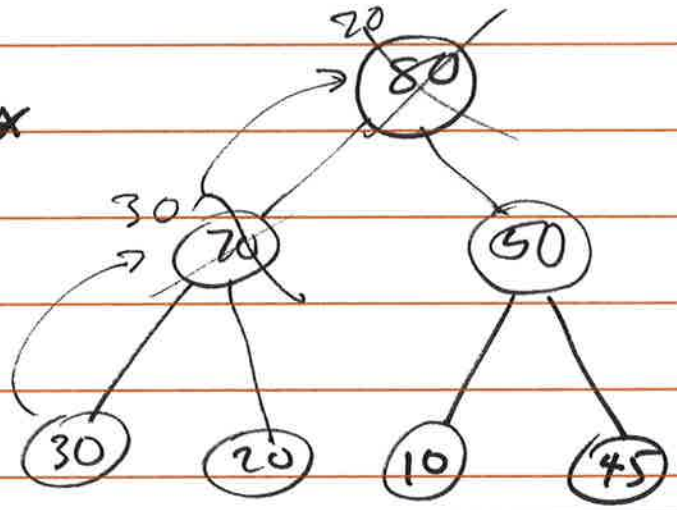
Priority Queues

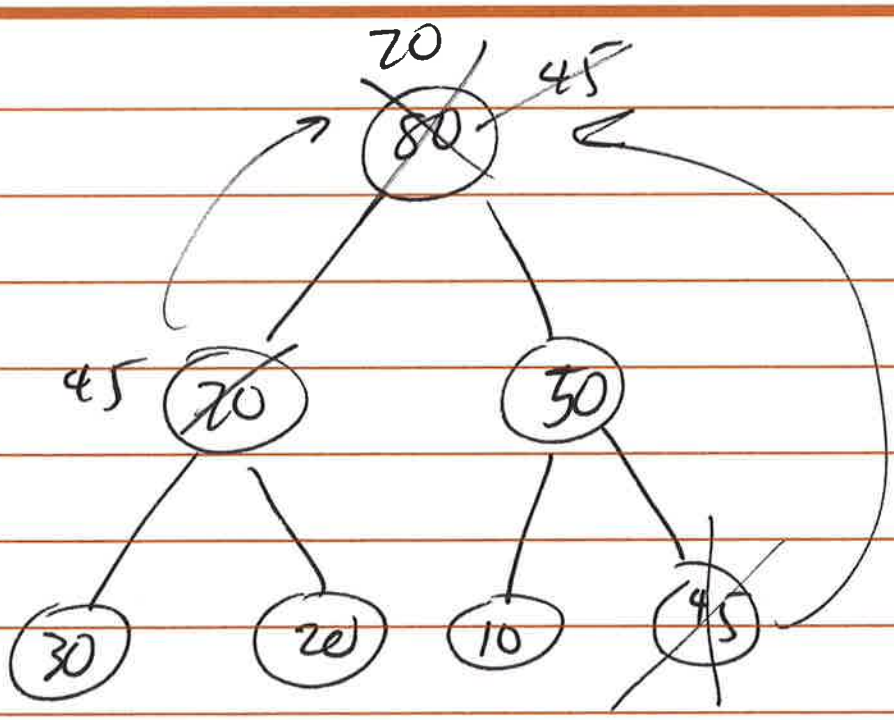
Binary Heap - Part 2

Say we have a Min-heap

To Find the Min element.
this takes O(1).

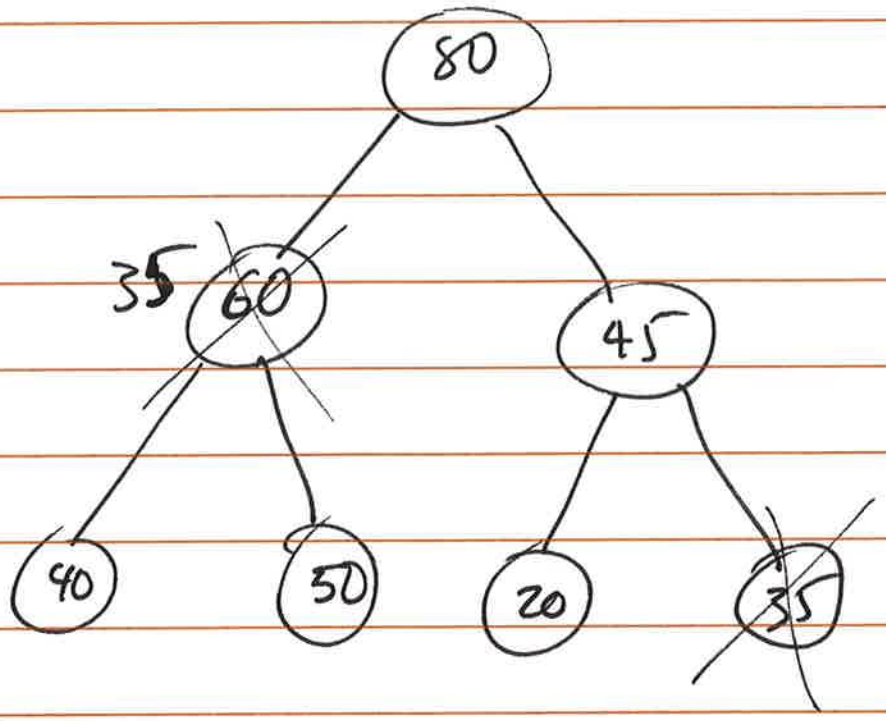
Extract_Max





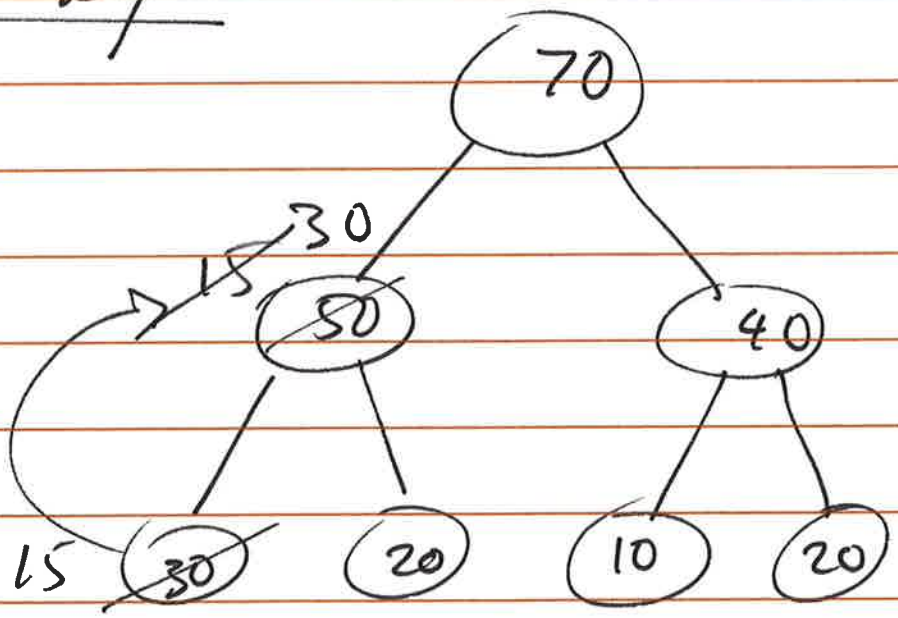
This takes O(log n)

Delete



This takes $O(\lg n)$

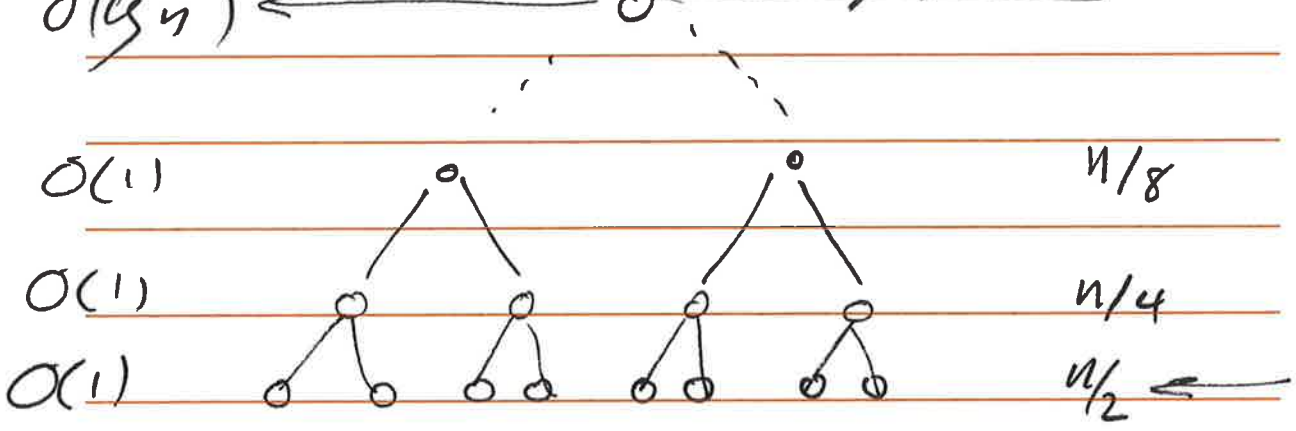
Decrease-key



takes $O(\lg n)$

Construction of the binary heap:

by inserting one element at a time we get $O(n \lg n)$



$n/4 * 1$

$n/8 * 2$

$n/16 * 3$

$\vdots$

$1 * \lg n$

$$T = n/4 * 1 + n/8 * 2 + n/16 * 3 + n/32 * 4$$

$$T/2 = n/8 * 1 + n/16 * 2 + n/32 * 3$$

$$T - T/2 = n/4 * 1 + n/8 * 1 + n/16 * 1 - \cdots$$

$T/2 = n/2 \Rightarrow \boxed{T = n}$

Complexity of construction is $O(n)$

Merge two binary heaps of size n.

Can be done in $O(n)$ using the construction op.

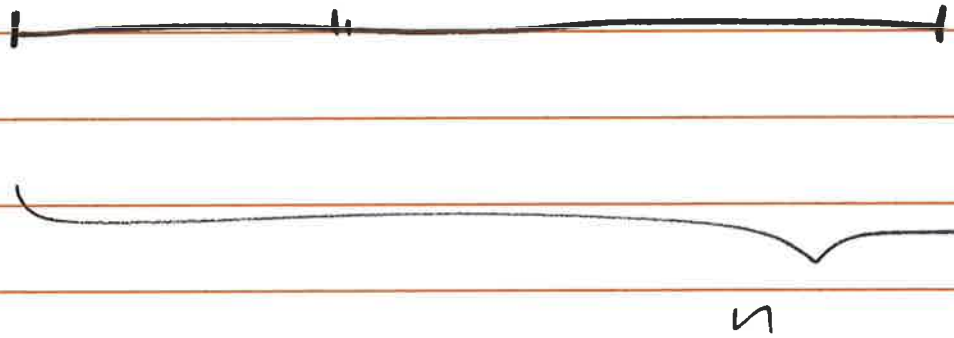
Input: An unsorted array of length n

output: top k values in the array ($k < n$)

Constraint: you cannot use any additional memory!

Find an alg. that runs in time $O(n \log k)$

Min heap of size k



- Construct a Min heap of size k inside the array — $O(k)$
- go thru the rest of $n-k$ elements in the array. If you find an el. that is bigger than the root element in the ~~element~~ heap then replace the root el w/ this new el..

$$(n-k)\lg k$$

$$\text{overall cost} = O(k) + O((n-k)\lg k)$$

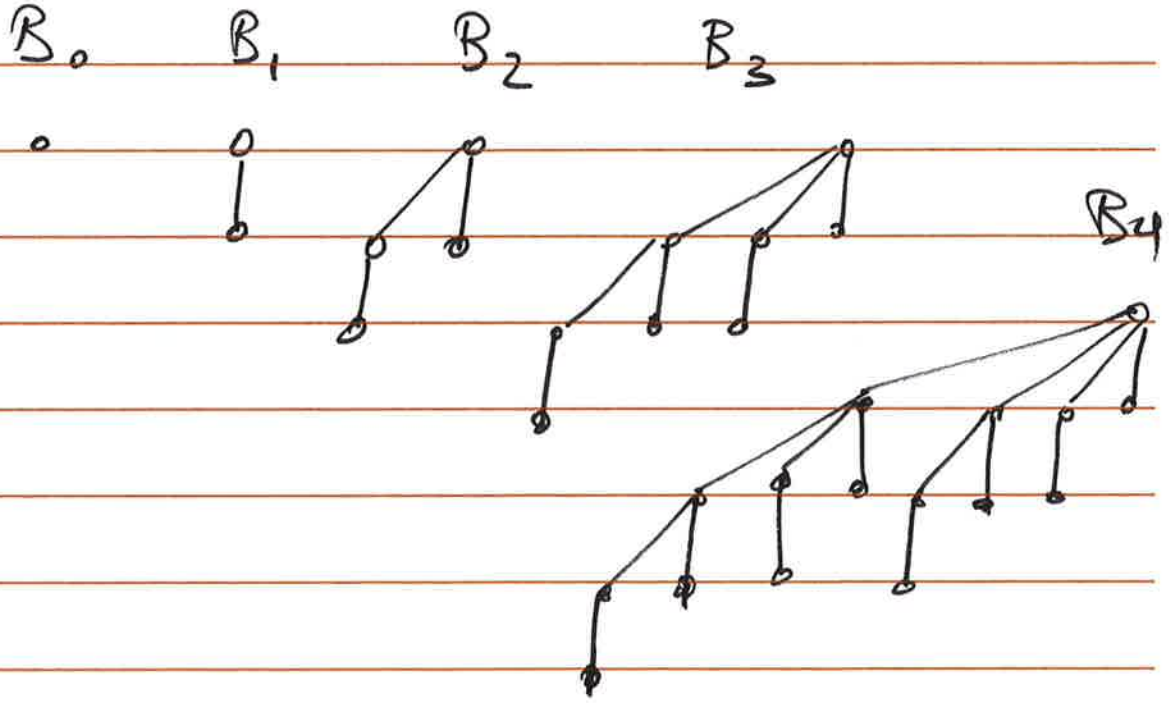
$$= O\{(n-k)\lg k)$$

$$= O(n \lg k)$$

Background material

Def. A binomial tree B_k is an ordered tree defined recursively

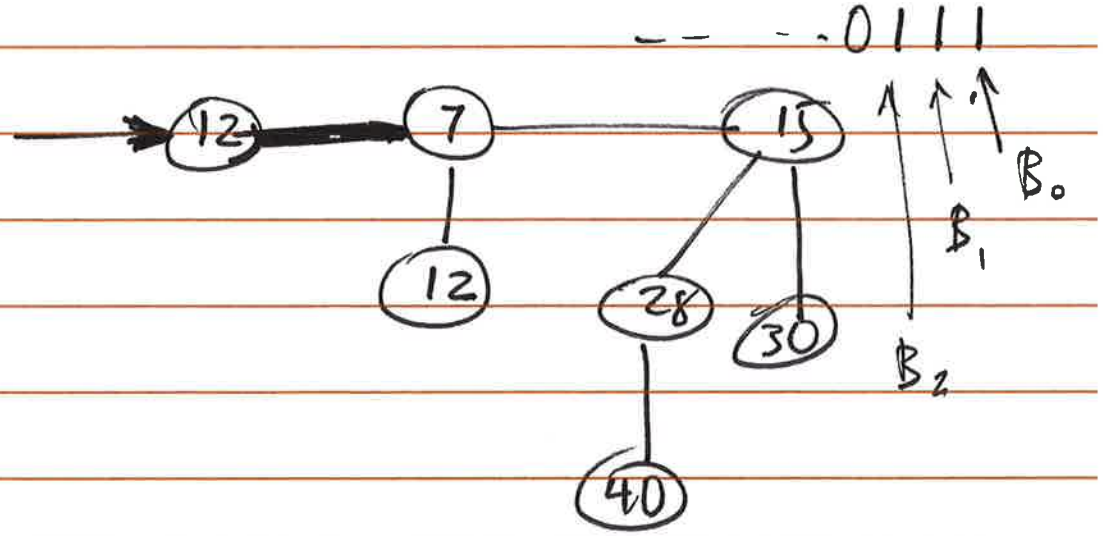
- Binomial tree B_0 consists of one node
- Binomial tree B_k consists of 2 binomial trees of B_{k-1} that are linked together such that root of one is the leftmost child of the root of the other.



Def. A binomial Heap H is a set of binomial trees that satisfies the following properties:

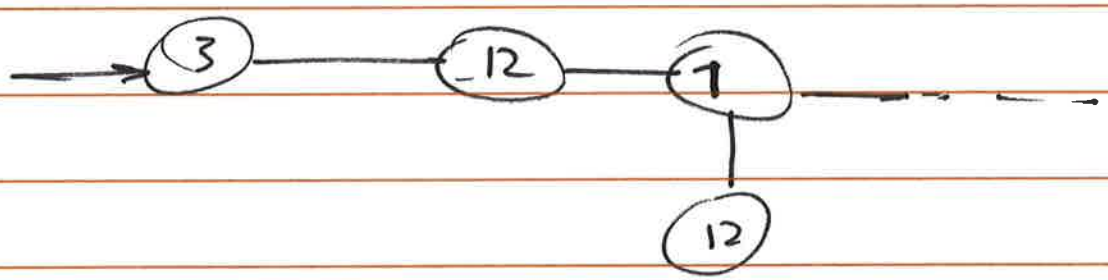
1- Each binomial trees in H obeys the Min-heap property

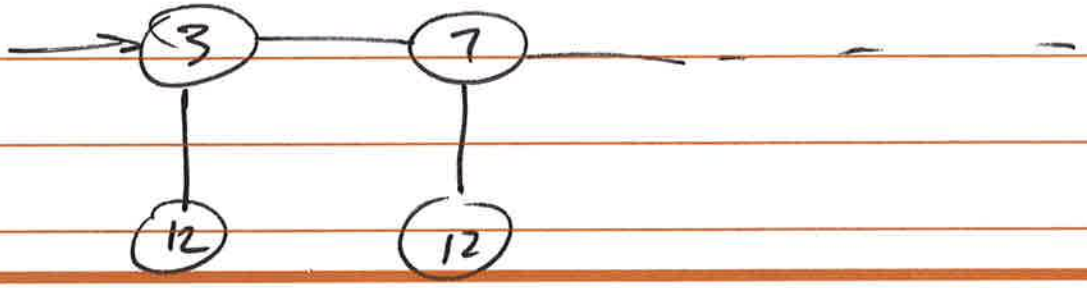
2- For any non-negative integer k, there is at most one binomial tree in H whose root has degree k.

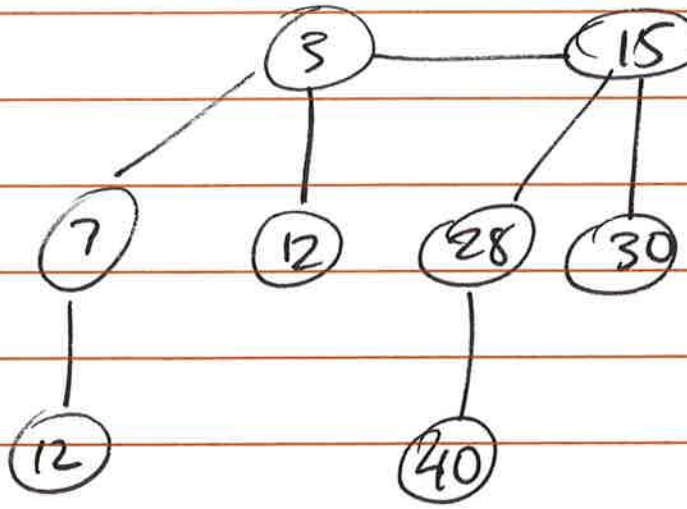


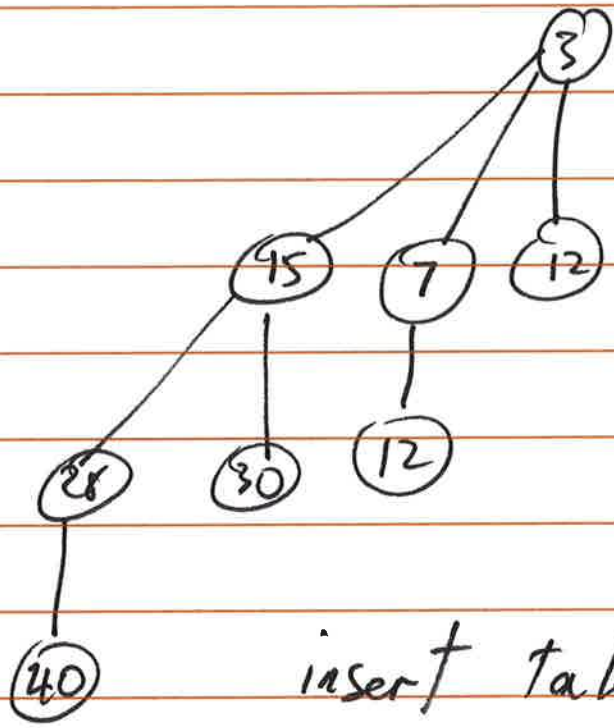
Find Min el. takes $O(\lg n)$

insert









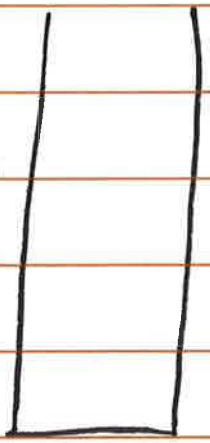
insert takes $O(\lg n)$

All operations except for construction take $O(\lg n)$ time in a binomial heap.

for $i = 1$ to n
    push or pop or multi-pop
end for

$O(n)$

$O(n)$ # $O(n)$



Stack multi-pop

$O(n^2)$?

Amortized cost of multi-pop =

$$\frac{O(n)}{n} = O(1)$$

Fibonacci heaps are loosely based on binomial heaps.

FH is a collection of min-heap trees similar to Binomial heaps

However, trees in FHs are not constrainted to be binomial trees.

Also, unlike binomial heaps, trees in FHs are not ordered.

insert in a FH takes O(1)

Find el. w/ Min key takes O(1)

decrease-key O(1)

merge O(1)

	Binary Heap Worst case	Binomial Heap Worst Case	Fib. Heap
Find_Min	$O(1)$	$O(\lg n)$	$O(1)$
insert	$O(\lg n)$		$O(1)$
extract_Min	$O(\lg n)$		$O(\lg n)$
delete	$O(\lg n)$		$O(\lg n)$
decrease_key	$O(\lg n)$		$O(1)$
merge	$O(n)$		$O(1)$
Construct	$O(n)$	$O(n)$	$O(n)$

Amortized cost