Data Structures

- Given a set of operations, how best to organize data

Priority Queue

Collection of values, totally ordered, distinct

Two operations

find & remove largest value — delete_max()

insert — insert(x)

Linear storage is not efficient

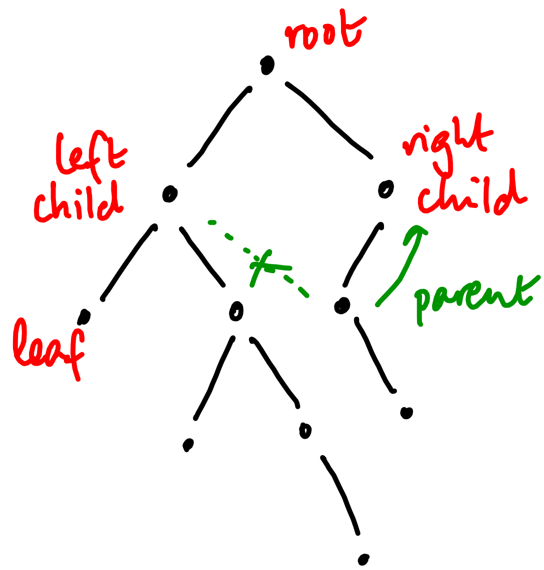
	delete_max	insert
Sorted	$O(1)$	$O(n)$
Unsorted	$O(n)$	$O(1)$

Two dimensional $\sqrt{n} \times \sqrt{n}$ array

Sort each row

Both delete_max(), insert(x) take $O(\sqrt{n})$

Trees (Binary)

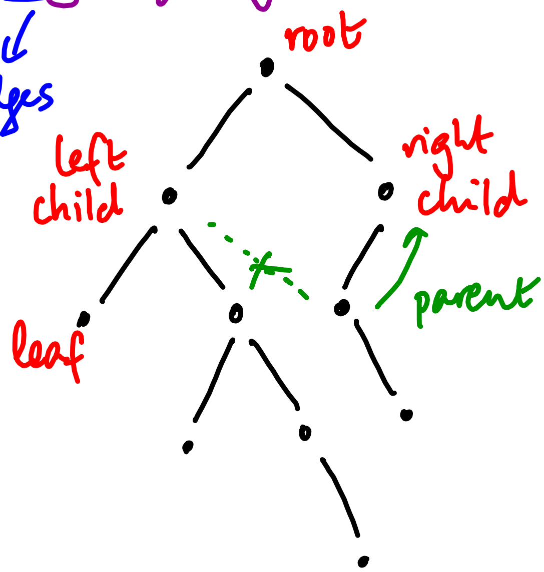


2 parameters

size = number of nodes

height = length of longest path from root to leaf

edges

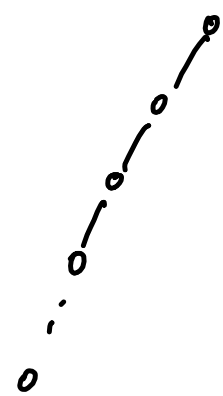


size = 10

height = 4

In general, height < size

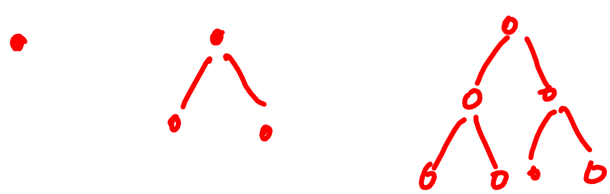
Worst case: height = size − 1



Balanced tree

Perfectly balanced

size(left subtree) = size(right subtree) for all nodes



Complete trees of height 0, 1, 2, . . .

Perfectly balanced tree

$h = 0 \quad size = 1$

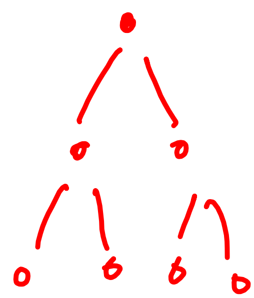
$h = 1 \quad s = 3$

$h = 2 \quad s = 7$

$h = 3 \quad s = 15$

$\vdots$

$h = k \quad s = 2^{k+1} - 1$



Crucial fact

$h = O(\log s)$

$h = O(\log s)$

holds for less restrictive notions of balance

$\forall$ nodes v size(left subtree(v)) = size(right subtree(v))

Instead

$|\text{size(left)} - \text{size(right)}| \leq 1$

height(left) = height(right)

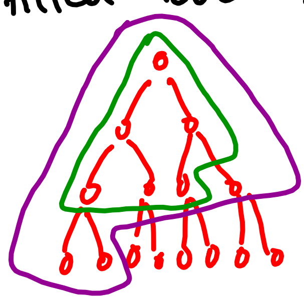
$|\text{height(left)} - \text{height(right)}| \leq 1$

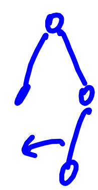
Assume each node contains a value

Heap

Binary tree, filled level by level, left to right

STRUCTURAL CONSTRAINT

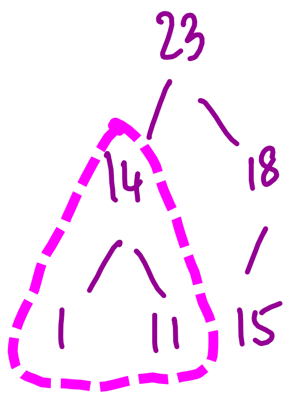




Not a heap

Every node is bigger than both its children

HEAP PROPERTY – Value Constraint



heap property is local to each node & children

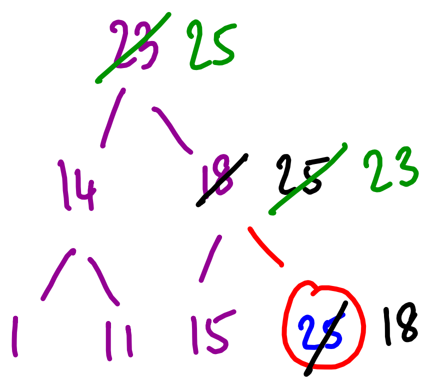
$height(left) - height(right) \leq 1$

In a heap, height is $O(\log size)$

How do we exploit heap for

delete_max

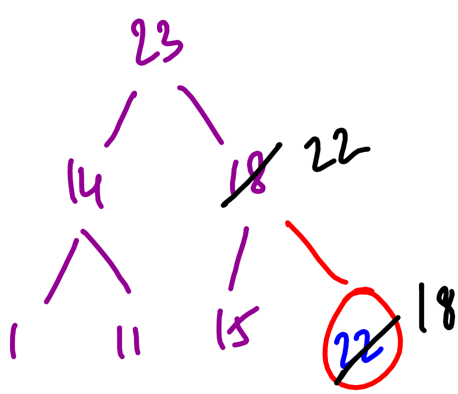
insert



Insert 25

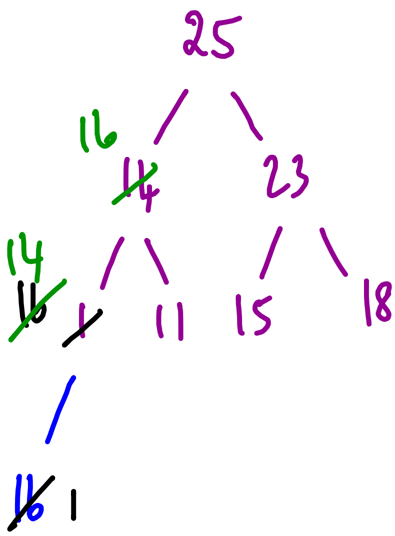
- Add a "position"
- Re-establish heap property

Swap upwards



All modifications occur on path to root

But height is $O(\log \text{size})$

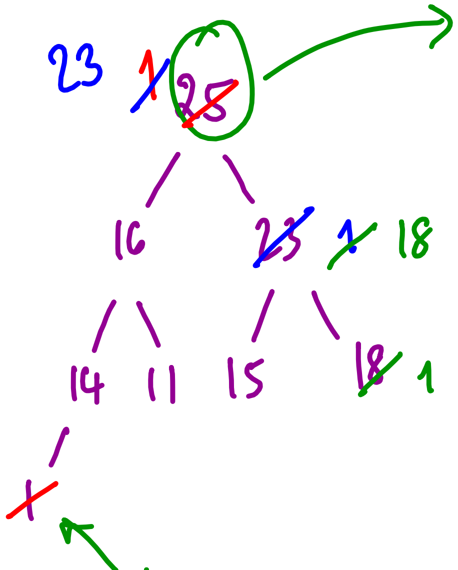


Insert 16

insert(x) takes time

$O(\log n)$

n = no. values = size



return as max

delete_max()

- where is max value?

  at root

- remove value at root, move last value to root

- reestablish heap property

  exchange with max of children

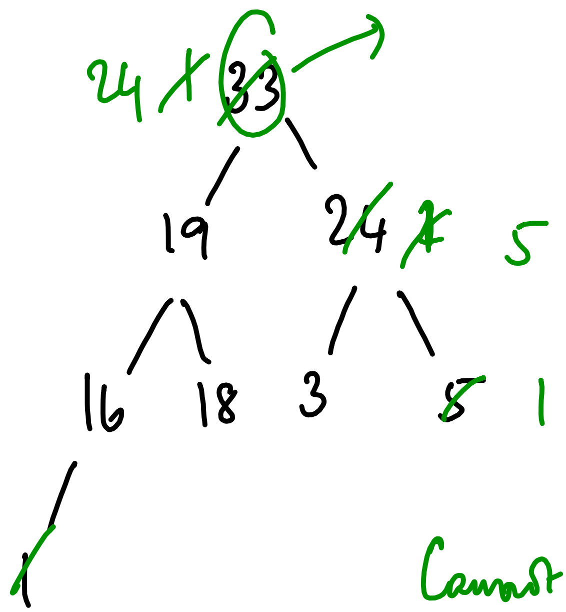
this position must go

move "homeless" value to empty slot

delete_max

- Walk down one path from root
- At most height = $O(\log n)$

	delete_max()	insert(x)
Sorted list	$O(1)$	$O(n)$
Unsorted list	$O(n)$	$O(1)$
$\sqrt{n} \times \sqrt{n}$ array	$O(\sqrt{n})$	$O(\sqrt{n})$
Heap	$O(\log n)$	$O(\log n)$



Cannot directly argue in terms of global order of values

Dual problem

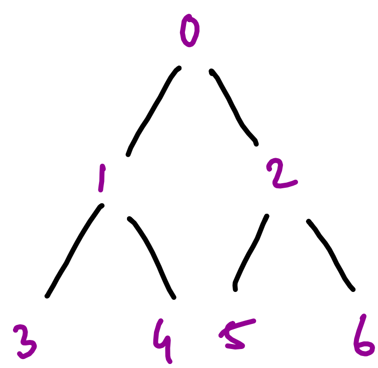
delete_min()

insert(x)

Min heap property : node is smaller than both its children

Our earlier heaps were max heaps

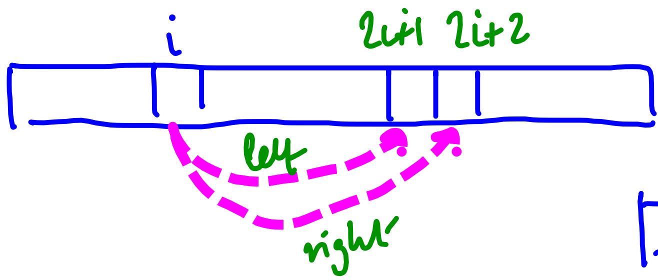
How to actually code this?



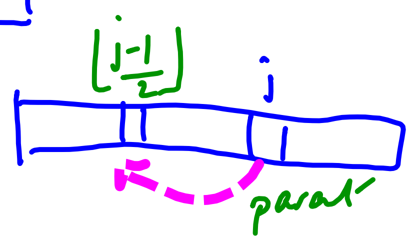
$\text{left\_child}(i) = 2i+1$

$\text{right\_child}(i) = 2i+2$

Store a heap in a list



$\text{parent}(i) = \left\lfloor \frac{i-1}{2} \right\rfloor$



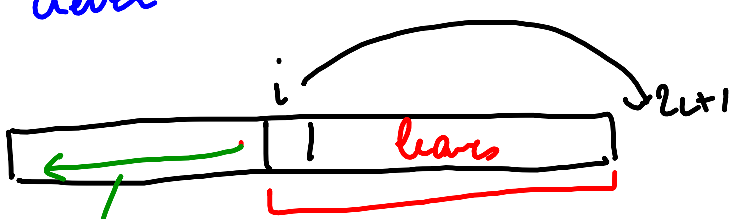
Can sort using a heap

Delete max n times

Building a heap from a list

- Empty heap, insert n values $n \log n$
- More clever

$O(n)$



fix heap property wrt children

heap property wrt children automatic