Priority queue

Collection of values that supports

Extract (& delete) highest priority element

Add an element with some priority

Modify priority of existing element

Look at a 2D data structure

Clearly balanced binary search trees suffice

Search tree effectively sorts the values

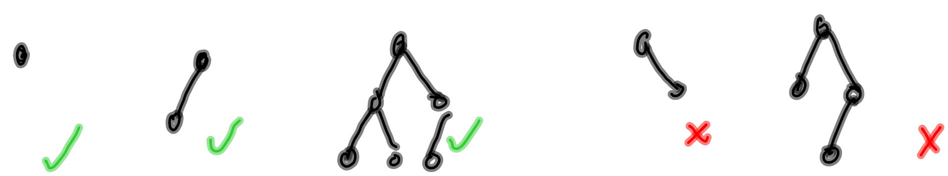
Can we make do with less?

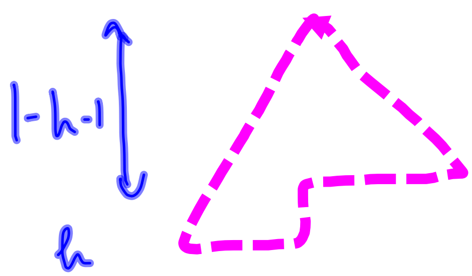
Heap

Binary tree satisfying 2 constraints

1. Structural (shape of tree)

Tree is filled level by level, left to right





Complete tree upto level h-1, leaves at level h from left to right

2. Constraint on values

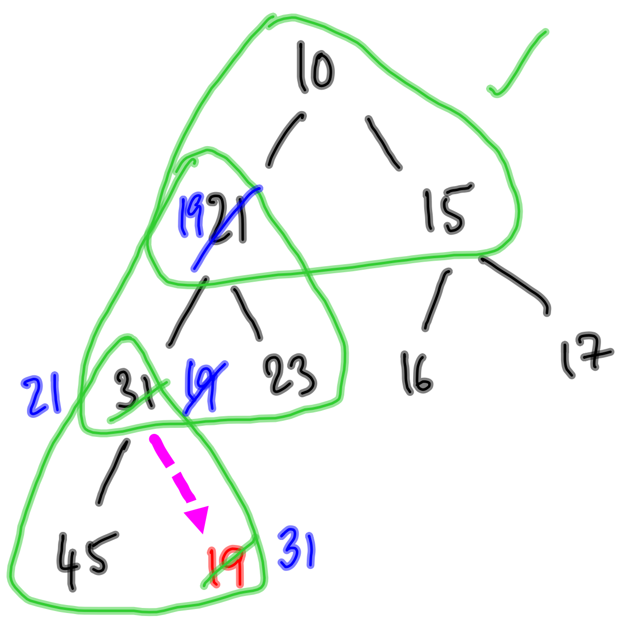
Assume smaller values have higher priority

min heap ‖ Value at v is ≤ (higher priority) the values at its children

Inductively, highest priority value is at root
2nd highest will be a child of root
Rest are "random"

can have duplicated value

Inserting a value into a heap



Heap must grow in this direction

Insert 19

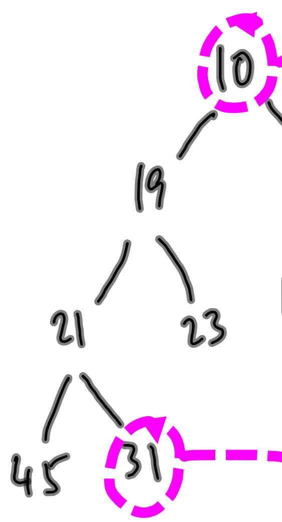
Swap with parent if heap property fails

Repeatedly swap upwards till you can stop

Imagine we inserted 9

Inserting an element walks up a single path
from leaf to root (or less) $O(\log n)$

Delete minimum value



Min value is at root

Move the homeless value
to the root

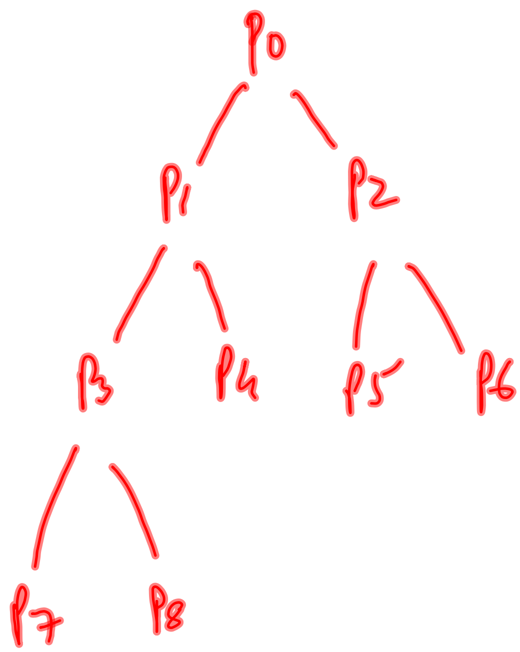
Demolish
this node

Swap 31 with its smallest child

Repeat till we can stop

Again, we walk down a single path, so $O(\log n)$

A heap can be stored as a list/array



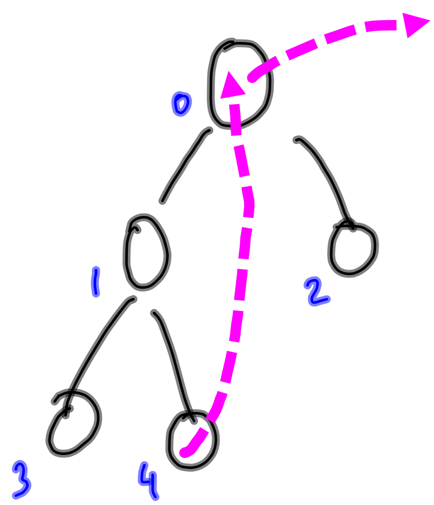
$[p_0, p_1, p_2, \ldots, p_8]$

Children of p_i

are p_{2i+1}, p_{2i+2}

Parent of p_i

is $p_{\lfloor (i-1)/2 \rfloor}$



next = h[0]

h[0] = h[4]

last position

Building a heap from a given list of values

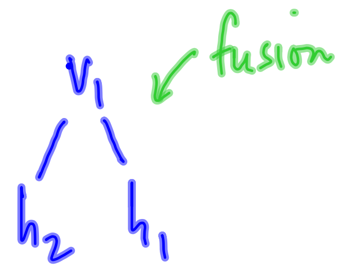
h[i] is a leaf iff 2i+1 > len(h)

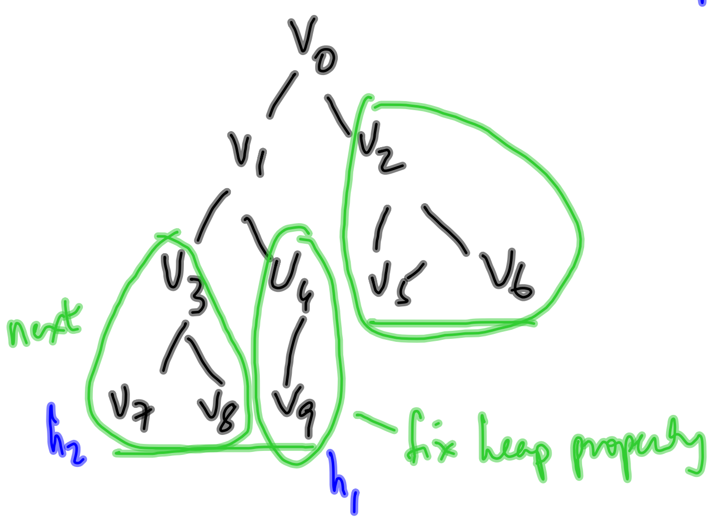
Heapify

Every leaf on its own is a heap

Work bottom up from leaves

Merge





may need to push v_1 further down into h_1 or h_2

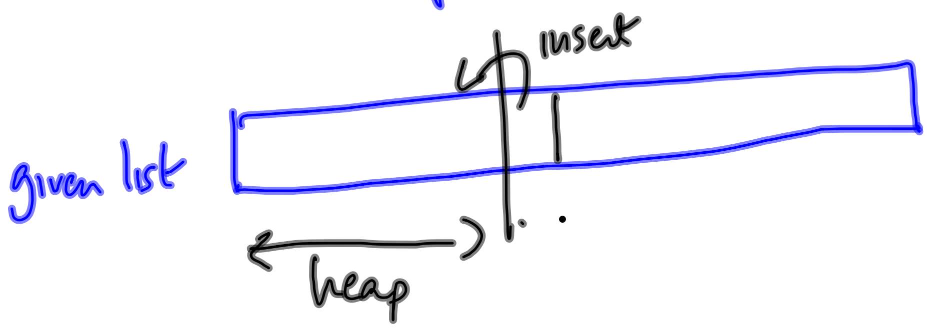
Conservatively $O(n \log n)$

Assume we have an empty heap

Insert each element in the given list into heap

$O(n \log n)$ conservatively

Don't need a separate space for heap



Option 1 is actually $O(n)$ if we analyze more carefully

Leave it for the accountants!

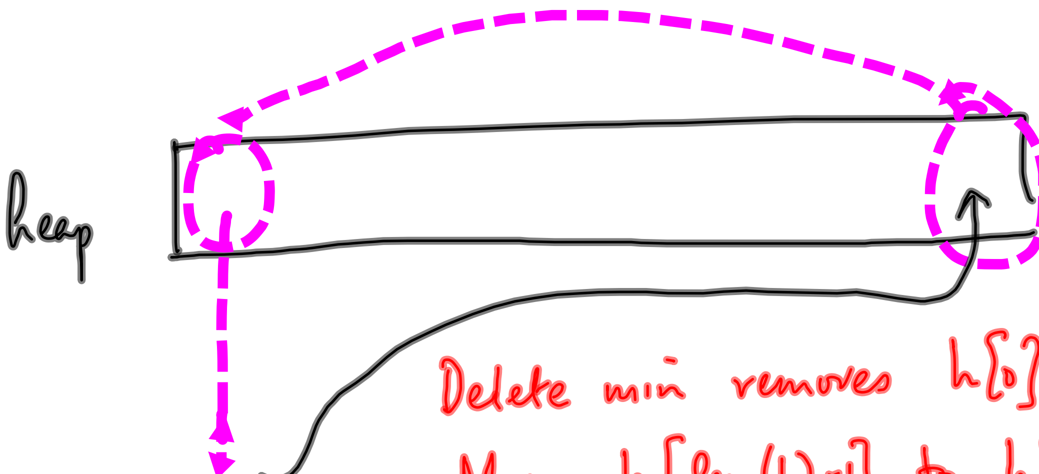
Given a list

Build a heap

Repeatedly delete min till heap is empty

$O(n \log n)$ sort! Heap Sort

Heap sort can be done in place



Delete min removes h[0]

Moves h[len(h)-1] to h[0]

Move h[0] to h[len(h)-1]

After n deletemin ops, heap is sorted in descending order

Modifying priorities in a heap

Modify v

Increase v?

Propagate changes down (like fusion)

$O(\log n)$

Decrease v?

Propagate up (like insert) $O(\log n)$

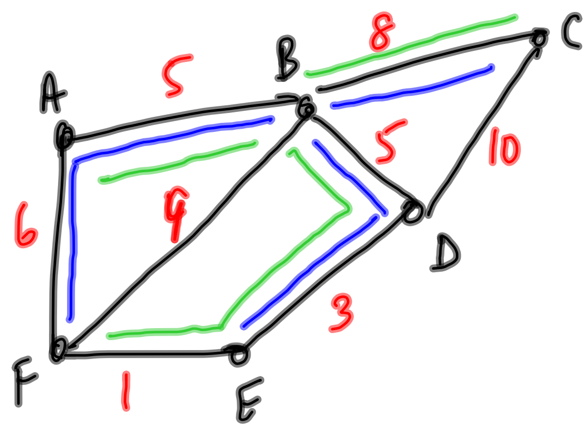
But where is v in the heap?

Need to maintain & update pointers to & from heap to objects on heap

Dijkstra's algorithm is greedy

At each stage we make a locally optimal choice that is never reviewed later

Given a weighted undirected graph identify a tree that connects all vertices with minimum weight



Blue tree has weight

6+5+8+5+3 = 27

Green tree

5+5 +3+1 +8 = 22

Find any minimum cost spanning tree (MCST)