COMP20003 Workshop Week 9

1	Priority Queue
2	Heaps & Binary Heaps
3	Heap Sort
4	Q 8.1
5	Programming 8.1: Natural Merge Sort

Yet Another ADT: Priority Queue

'Can I borrow your baby?...'

What is a PQ?

Is it similar to a queue (LIFO)?

What's priority? highest priority?

Real-life Examples

- a list of to-do works
- a hospital queue where the patient with the most critical situation would be the first in the queue (each patient would be issued a number representing the critical level)

Yet Another ADT: Priority Queue

PQ: queue, where each element is associated with a *priority* (or *weight*), and the elements will be *dequeued* following the order of priority.

'*Can I borrow your baby?...*'

Main operations:

- ***enqueue***: inserts (element, weight) into `PQ (enPQ)`
- ***dequeue***: returns the heaviest element, and removes it from `PQ (dePQ`, or `deleteMax`, or `deleteMin`)
- *peek*: returns the heaviest element
- *changeWeight*: change the weight of a particular element of a queue
- *create*: creates an empty `PQ (makePQ)`
- `isEmptyPQ`

unreasonable, but possible, concrete data structures for PQ

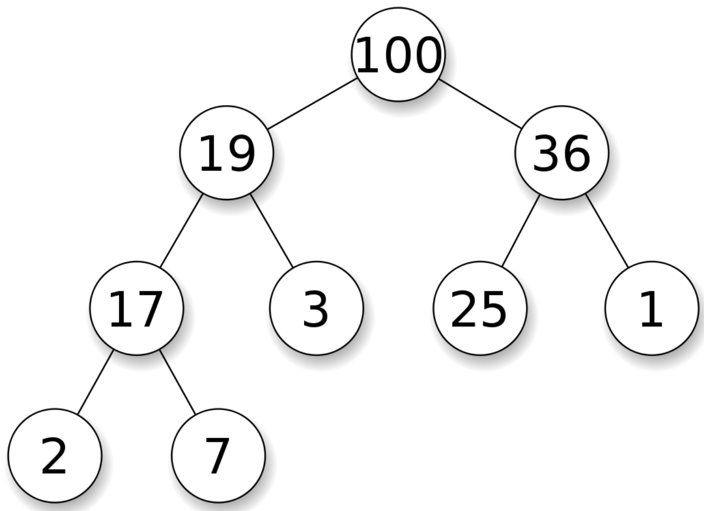
DS	complexity of enPQ	complexity of dePQ	complexity of peek
unsorted arrays or linked list			
sorted arrays or linked lists			

unreasonable, but possible, concrete data structures for PQ

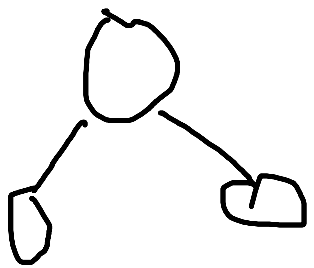
DS	complexity of enPQ	complexity of dePQ	complexity of peek
unsorted arrays or linked list	$\boldsymbol{\theta}(1)$	$\theta(n)$	$\theta(n)$
sorted arrays or linked lists	$O(n)$	$\theta(1)$	$\theta(1)$

Binary Heap = ?

Example of PQ: Binary Max/Min Heap

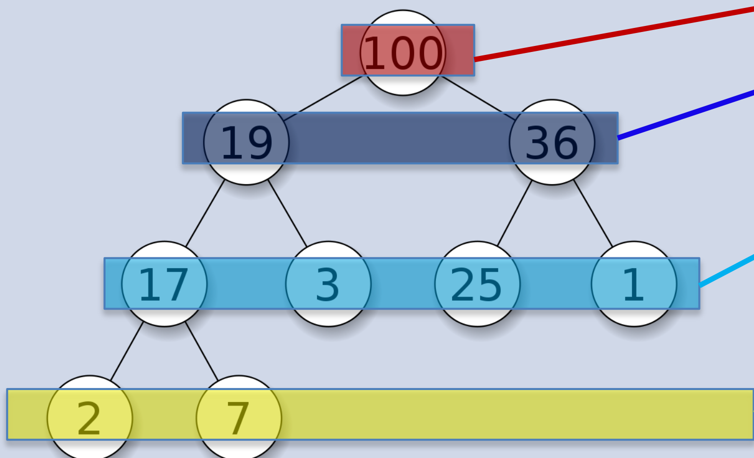
Example	**Conditions**
	1. The tree is complete. 2. *The heap property*: each node has a higher priority (here, is larger) than any of its descendants (or equivalently, just its children).

is binary heap?



Binary Heap is implemented as an array!

Visualisation: as a complete binary tree	Implementation: using arrays



idx	0	1	2	3	4	5	6	7	8	9
val	×	100	19	36	17	3	25	1	2	7

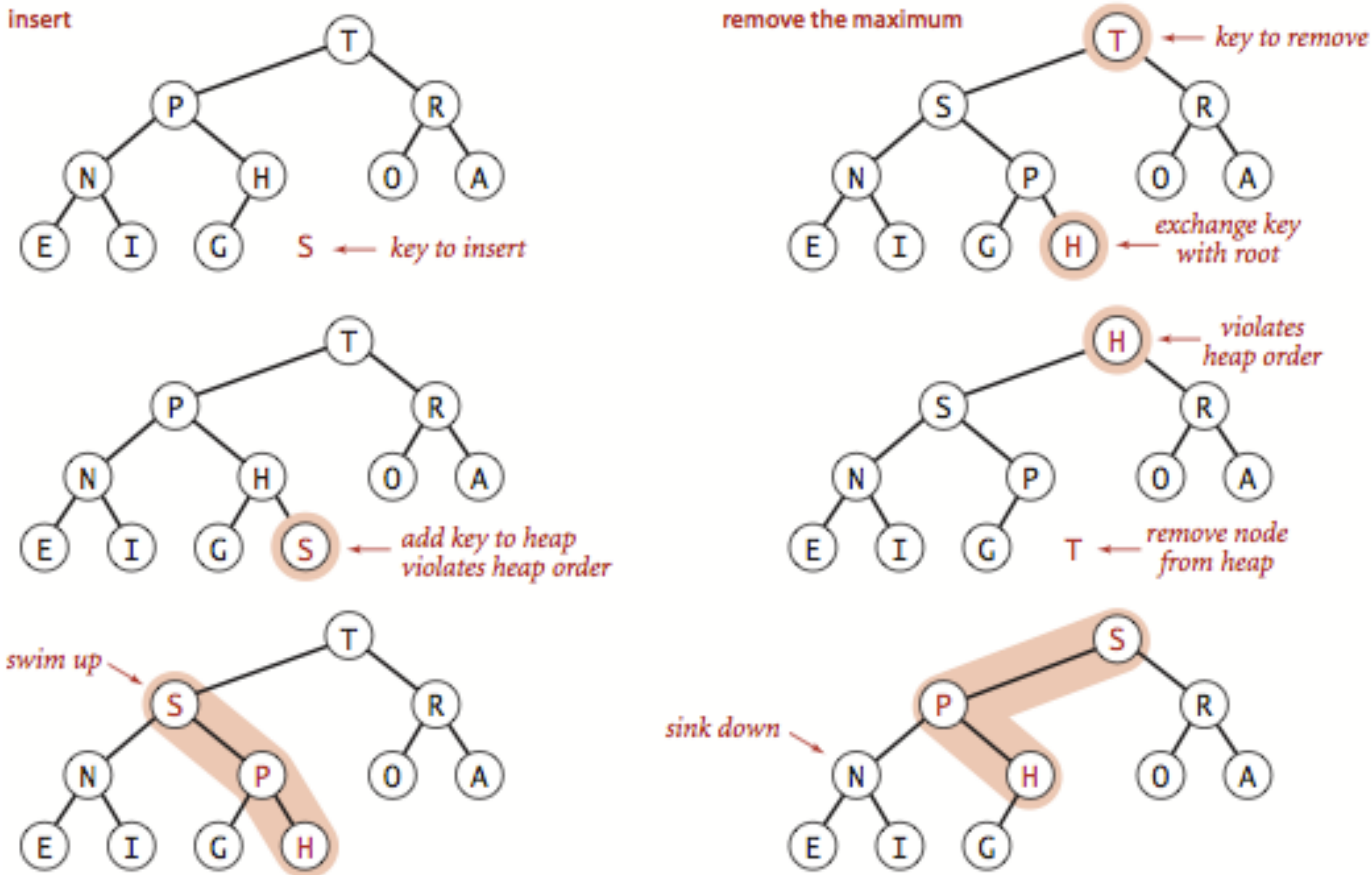
note: `H[1]` is for the root
`H[0]` not used

- the tree is complete
- a non-root node has 1 parent
- a non-leaf node has 1 left child
- a non-leaf node might have 1 right child

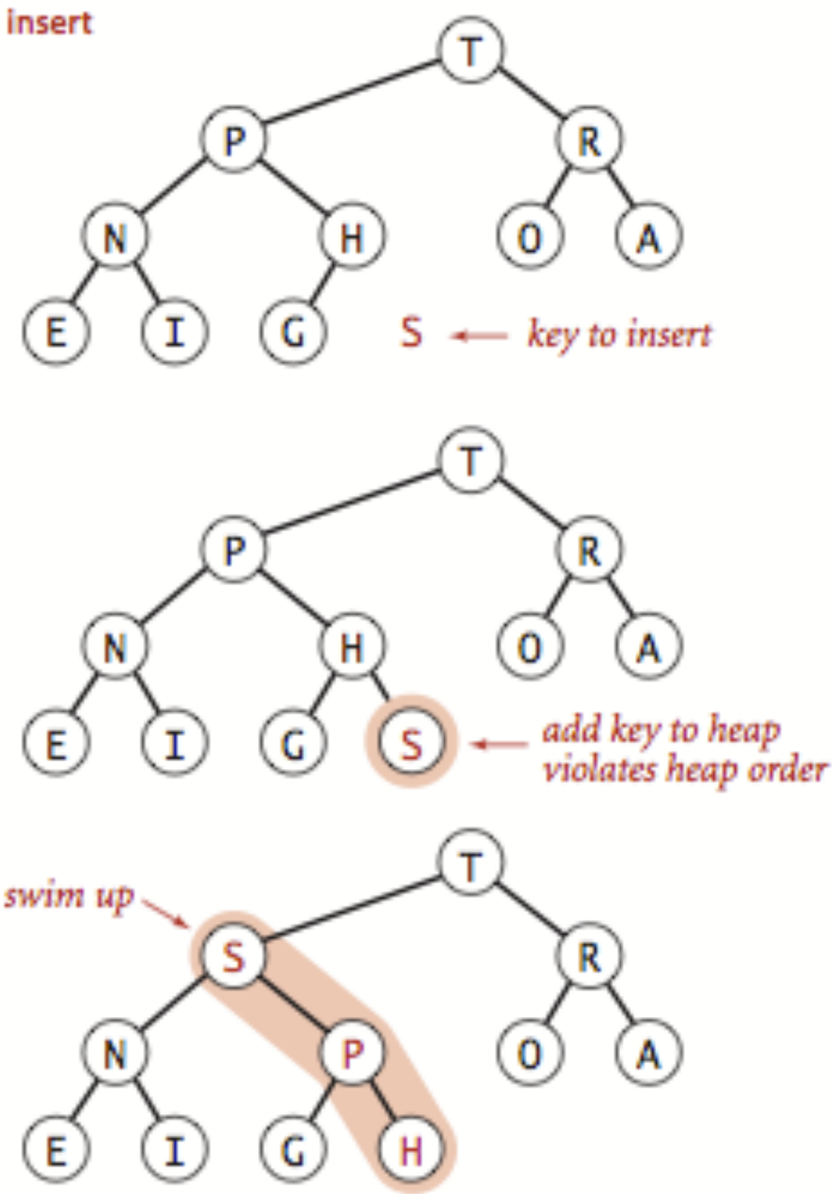
Heap `h` is a pair `{H[], n}`, elements in `H[1..n]`

- there is no "hole" in array `H[1..n]`
- parent of `H[i]` is `H[i/2]` iif `i>1`
- left child of `H[i]` is `H[2*i]` iif `2*i<=n`
- right child of `H[i]` is `H[2*i+1]` iif `2*i+1<=n`

Binary Heaps: Basic Operations (visualisation, overall)



Insert a new elem into a heap. Complexity= ?



H= [T,P,R,N,H,O,A,E,I,C]

H has 10 elements

Insert S

Just added H[11]= S

5=11/2 11

[T,P,R,N,H,O,A,E,I,C,S]

that will likely violate heap order

Need to promote H[11] up using upheap(h,i=11), which repeatedly swap node i with its parent.

2=5/2 5=11/2 11

[T,P,R,N,H,O,A,E,I,C,S]

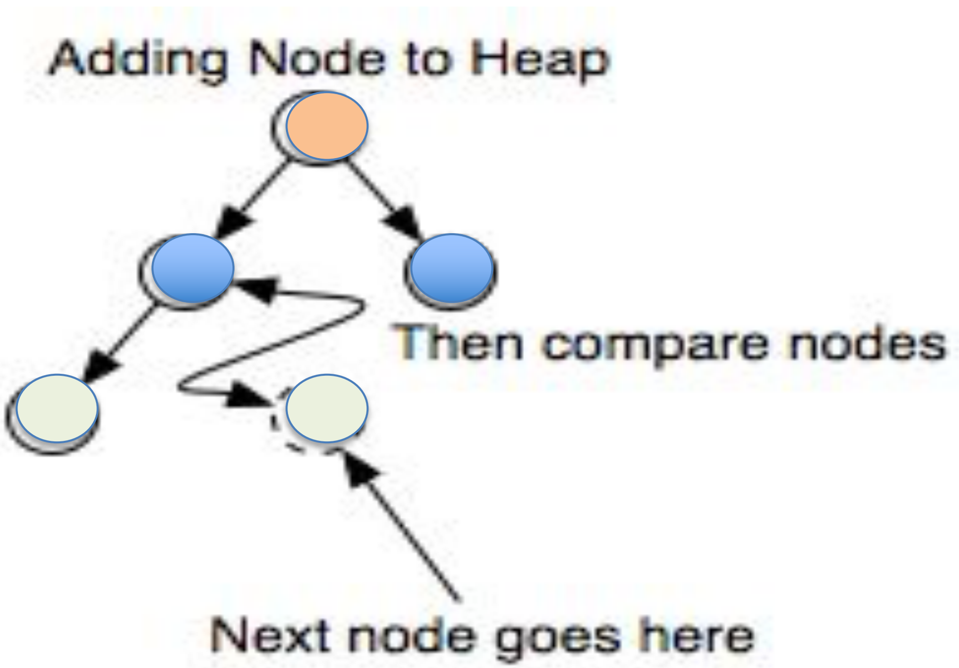
upheap – basic operation for inserting into heaps

Problem: The last node of (e.g. just inserted into) a heap, and only it, might violate the heap property. Need to repair the heap.

index= 1 2 3 4 5

Operation: `upheap(h,node)`

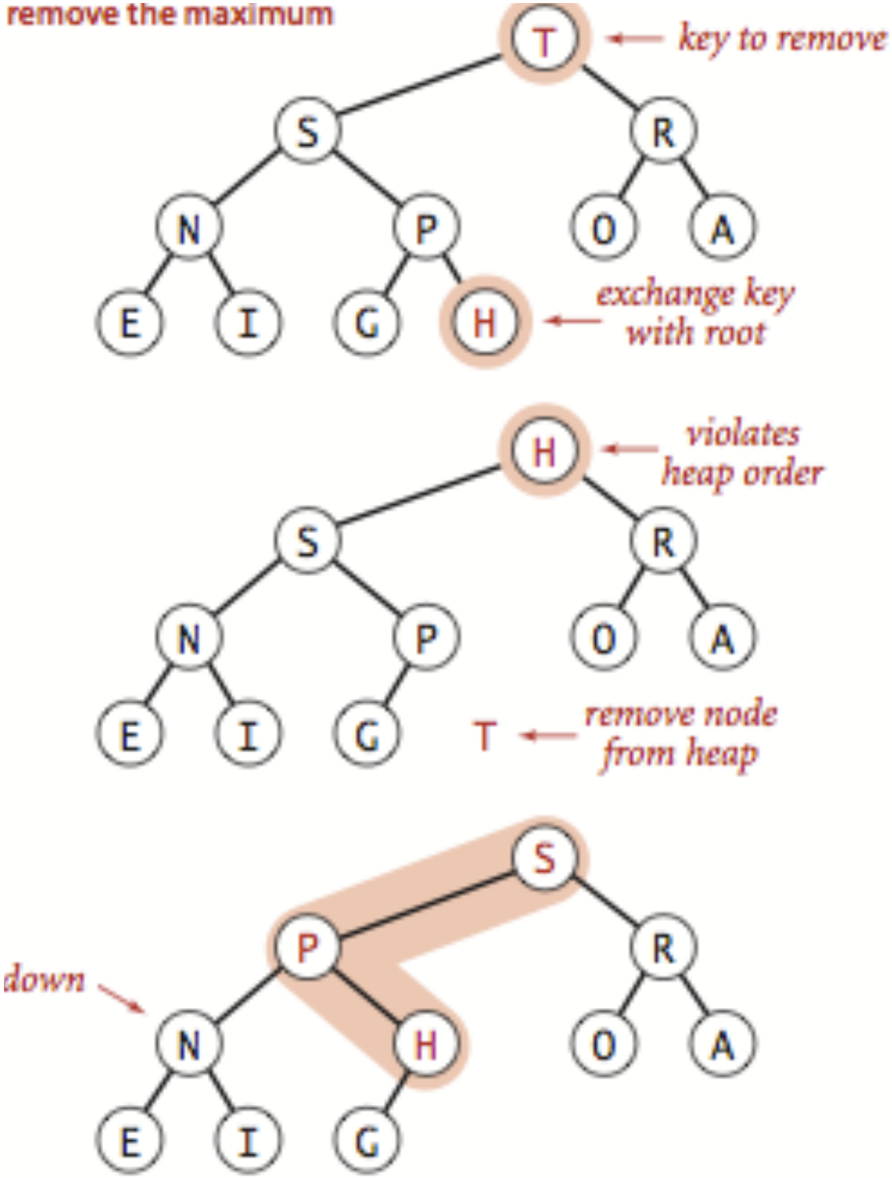
(5/2)/2 5/2 5



loop while(*needed*) {
  swap(node, parent)
}
needed= having_parent
  && parent < node

→ supposing node has index i
while (???) {
  swap( )
  i=
}

deletemax: delete (and returns) the heaviest. Complexity=



H= [T,S,R,N,P,O,A,E,I,G,H]

H has 11 elements

exchange and remove the last -->

[H,S,R,N,P,O,A,E,I,G]

n=11, heap order violated

Need to push H[1] down using downheap(h,i=1), which repeatedly swap node i with its heaviest child.

[H,**S**,R,N,P,O,A,E,I,G] cs= 2,3

[S,H,R,N,**P**,O,A,E,I,G] cs= 4,5

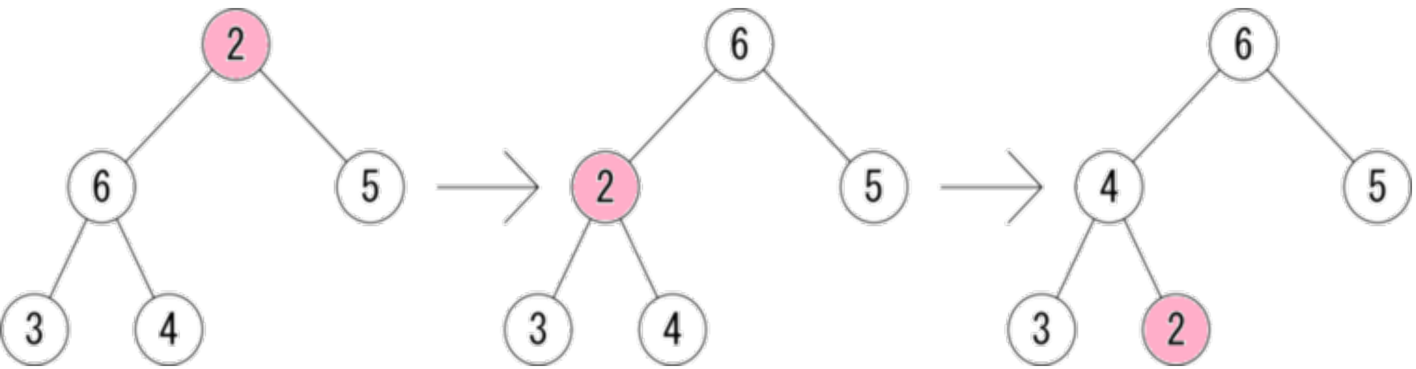
[S,P,R,N,H,O,A,E,I,**G**] cs= 10

[S,P,R,N,H,O,A,E,I,G] <no swap>

downheap – basic operation deletion from heap

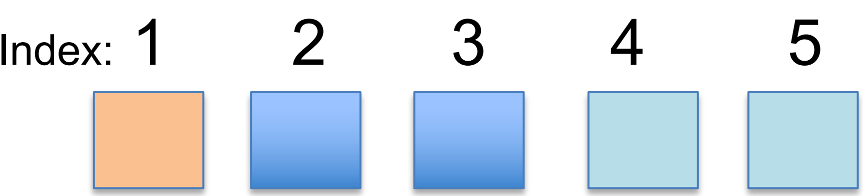
Problem after replacing `H[1]`: The root (and only the root) might violate the heap property. Need to repair the heap.

Operation: `downheap(h,node)`



maxchild(i)= ?

```
loop while (needed) {
    swap (node, maxchild)
}
```



needed = having_child
&& node < maxchild

Notes& Complexity

Notes: `upheap` and `downheap` can be performed for any node of the heap.
Example: changing the priority of a node in heap.

Heapsort=

How?

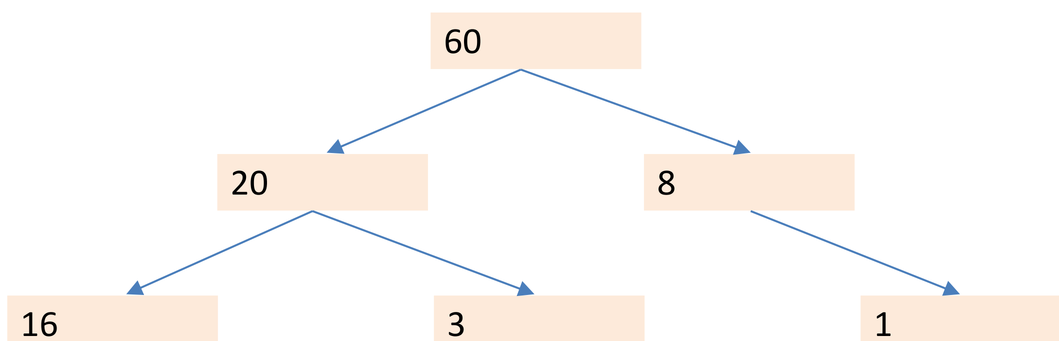
Heapsort example (Grady's slides)

Sort the keys: 20,3,60, 8,1,16

First make them into heap.

Heap in memory:

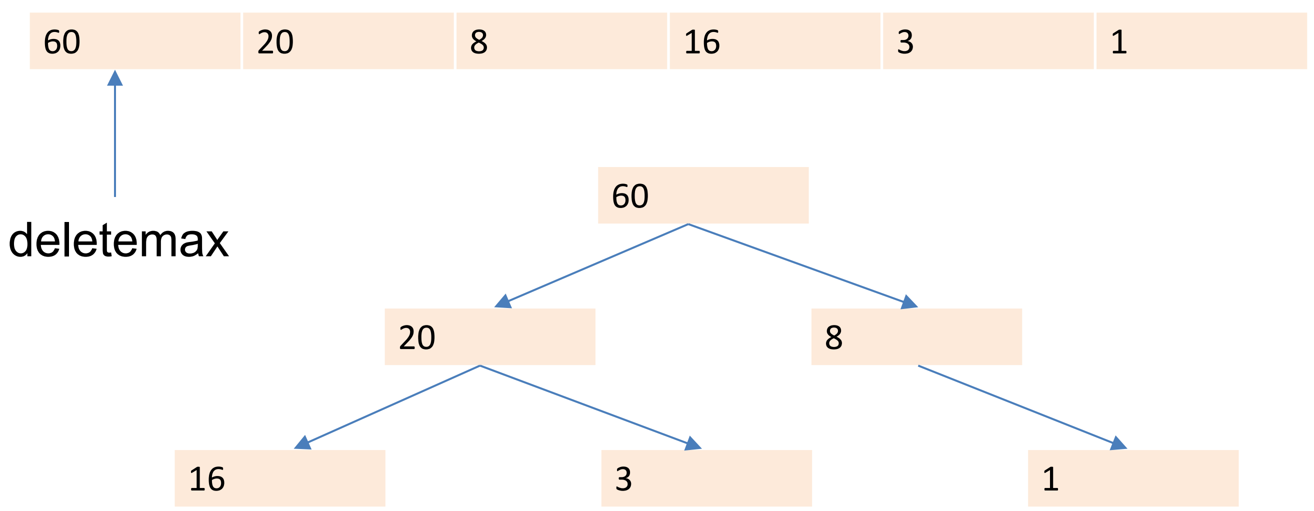
60	20	8	16	3	1



Heapsort

Then, repeatedly deletemax and place it at the end

Heap in memory:

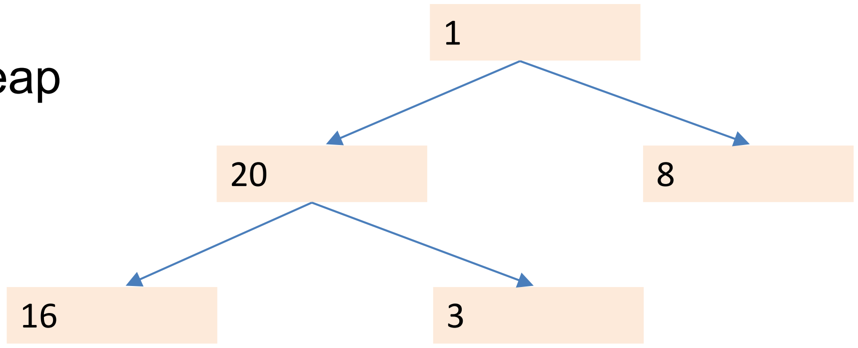


Heapsort

Heap in memory:

1	20	8	16	3	60

downheap

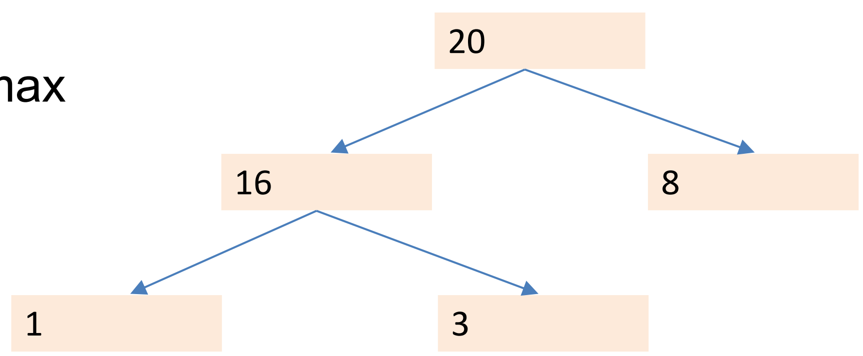


Heapsort

Heap in memory:

20	16	8	1	3	60

deletemax

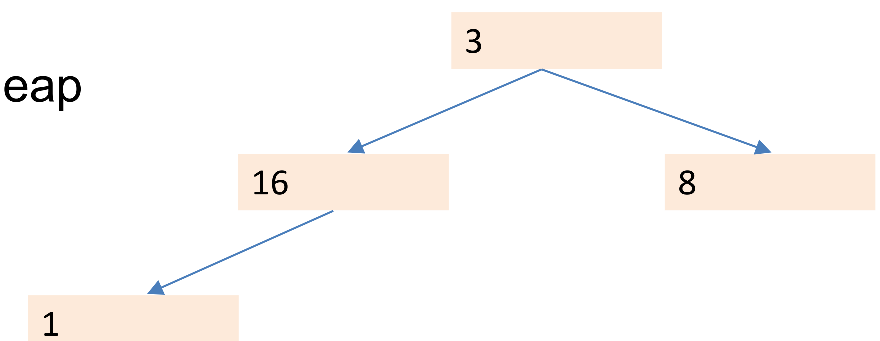


Heapsort

Heap in memory:

3	16	8	1	20	60

downheap

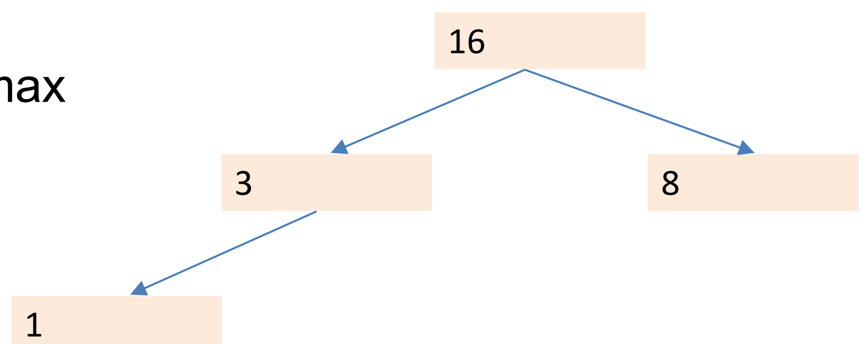


Heapsort

Heap in memory:

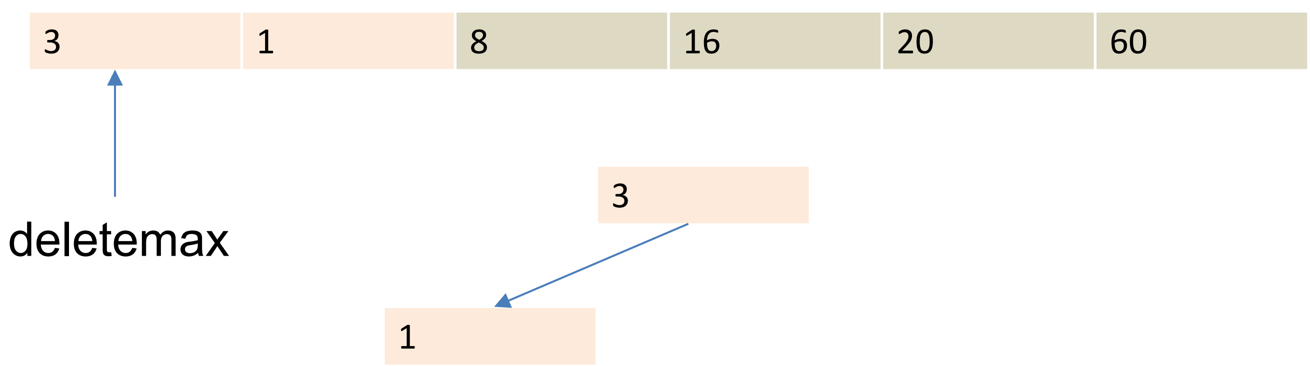
16	3	8	1	20	60

deletemax



Heapsort

Heap in memory:



Heapsort

done when heap just has 1 element

Heap in memory:

1	3	8	16	20	60

1

HeapSort summary

- Construct a max heap of n elements.
- Swap root (max) with the (current) last element.
- Remove last element from further consideration, *i.e.* decrease size of heap by 1.
- Fix heap using....
  ... `downheap(for node i=1)`
- Repeat until finished (ie heap just has 1 element).
- Complexity= ?

How to efficiently build a heap with n elements?

- Solution 1: insert each element into the (initially empty) heap, and do `upheap` after each insertion.

  Complexity: O(n logn)

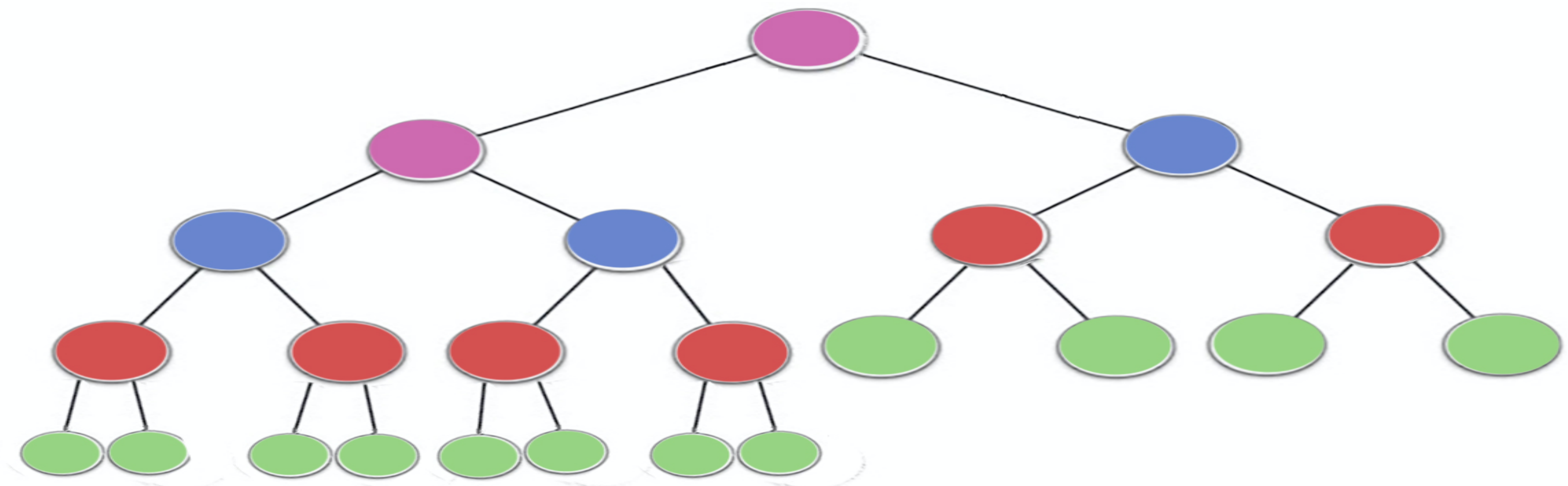
How to efficiently build a heap with n elements? heapify

Solution 2: populate the heap array with n elements in the input order, then turn the array to a heap (ie make it to satisfy the heap condition). Algorithm:

```
for (i=n/2; i>0; i--) {
// for i from last parent to fisrt parent
    downheap(h, i);
}
```

= **Θ(n)** (see lectures and/or ask Google for a proof)

The operation is aka. **Heapify**/Fixheap/Makeheap/ Bottom-Up Heap Construction



Heap & Heap Sort: Complexity

Heap operations:

- `upheap`:
- `downheap`:
- insert 1 element:
- `deleteMax`:
- heapify:
- `heapsort`:

Q 8.1

Construct a max binary heap from the following keys:

8 7 16 10 5 13 5 11 15 12 1 17

a) Construct a max binary heap using the up-heap, inserting one number at a time.

b) Now construct a max binary heap from the same keys, using downheap (ie convert the original array into a heap).

What is the complexity of each method? Did the time it took you to do the exercise on paper correlate (roughly) with the theoretical complexity?

a) by inserting one-by-one

Construct a max binary heap from the following keys:

8 7 16 10 17

b) by heapify

Construct a max binary heap from the following keys:

8 7 16 10 17

MST Questions

If you have any questions from the MST you'd like to go over, let us know

Demonstration – Adaptive Merge Sort (Grady's slides)

Bottom-up merge sort improvement

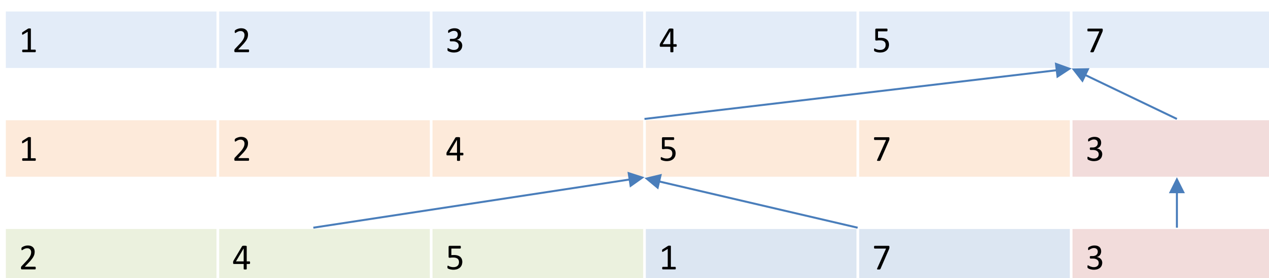
- Monotonic increasing runs already sorted
- Insert monotonic runs into queue instead of singletons

2	4	5	1	7	3

Demonstration – Adaptive Merge Sort

Bottom-up merge sort improvement

- Best Case: $\Theta(n)$
- Worst Case: $\Theta(n \log n)$



Peer Programming Exercises

Work in breakout rooms through writing adaptive merge sort program