CS 2110: Priority Queues and Heaps

Prelim 2 scores released soon
median: 75

Priority Queue: extracts highest-priority element (minimum)

- shortest path (+A*)
- simulation (next event)
- compression (Huffman coding)
- sorting

element ↓ priority ↙

```
interface PriorityQueue<E, P> {
    /** Add element to queue. */   // aka insert, add.
    void push(E element);
    /** Effect: remove highest-priority element */
    E pop();   // aka extractMin, remove, ...
    /** Effect: increase priority of e.
        Requires: e is in queue. */
    void increasePriority(E elem, P priority);   ← Java PriorityQueue lacks this.
}
```

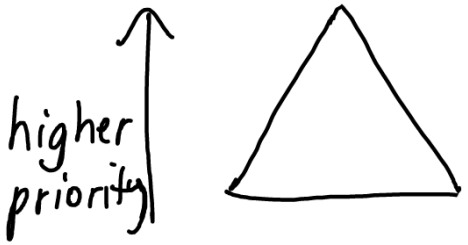
Implementation: Binary Heap

good asymptotic performance & simple good constants

"binary heap" ≠ "memory heap"

Order

Heap: tree satisfying order invariant



For all $x \neq$ root,

$x.priority \geq x.parent.priority$

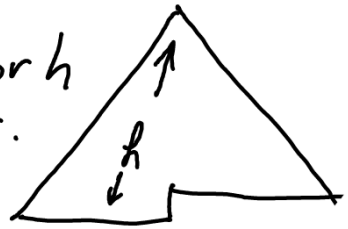
higher priority

Binary heap: binary tree also satisfying Shape invariant.

- All leaves at depth h or $h-1$
- If a leaf is at depth h:
  - all leaves to right are at depth $h-1$ or h
  - all possible depth-h leaves to left exist.

$\Rightarrow h = O(\lg n)$

Shape



Binary heaps?

Heap: tree satisfying order invariant

Order

For all $n \neq$ root,

$n.\text{priority} \geq n.\text{parent.priority}$

higher priority

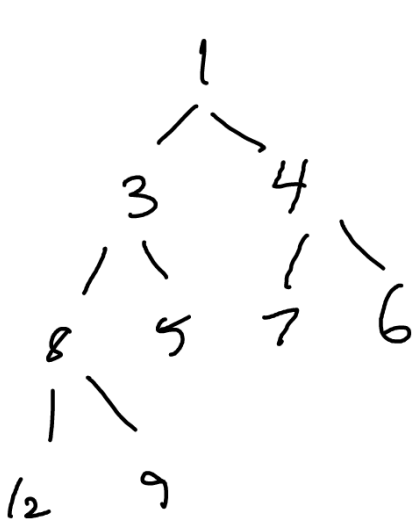
higher priority

Binary heap: binary tree also satisfying Shape invariant.

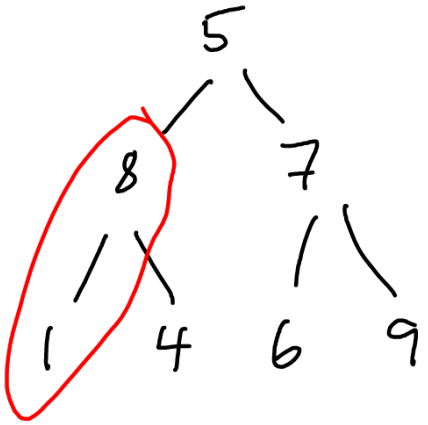
- All leaves at depth h or $h-1$
- All depth-h leaves are to the left of the $h-1$ leaves.

$\Rightarrow h = O(\lg n)$

Shape

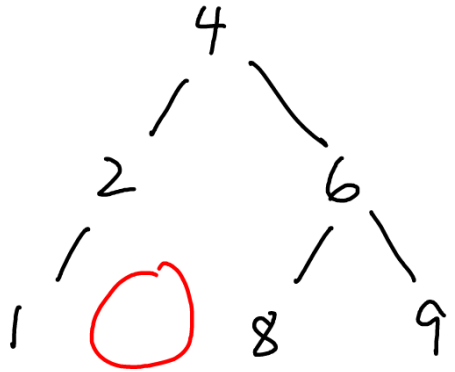


A



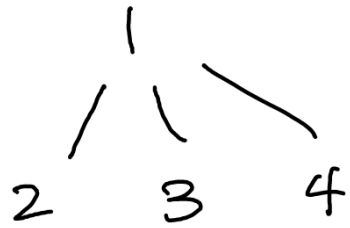
Order

B



Shape

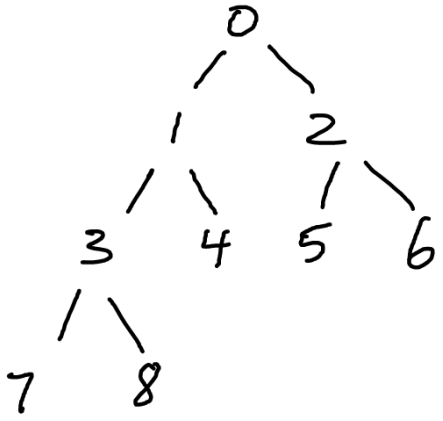
C



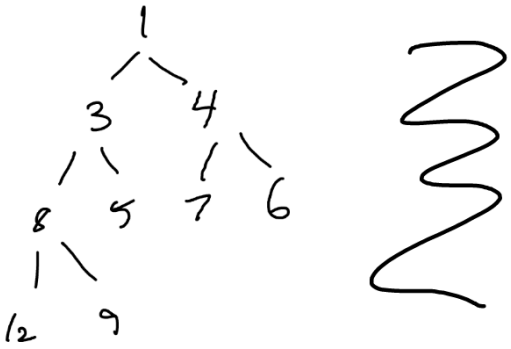
D

- not binary

Trick: store in resizable array. (no pointers)



array indices



0	1	2	3	4	5	6	7	8
1	3	4	8	5	7	6	12	9

Shape: no gaps.

Tree w/o pointers.

Children of node at index i?

left: $2i+1$

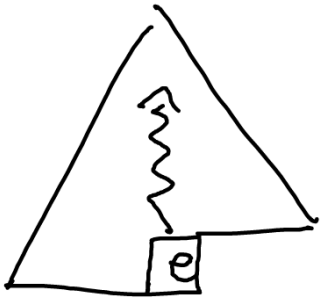
right: $2i+2$

Parent of node at index i?

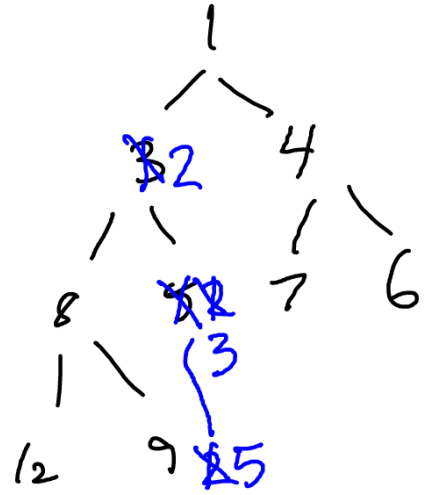
$\left\lfloor \frac{i-1}{2} \right\rfloor = (i-1) >> 1$

push(e)

- Add at end of array to preserve Shape.
- "Bubble up" to restore Order.
  - swap with parent until parent has higher priority



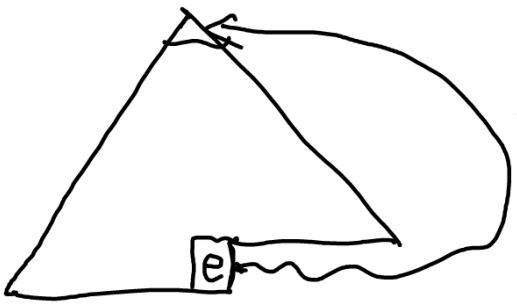
push(2)



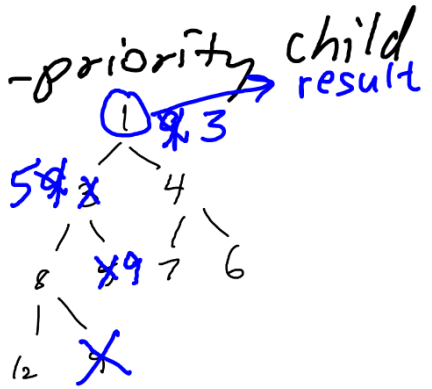
Time: $O(h) = O(\lg n)$

pop(e) / ExtractMin(e)

- Result = root value (remove)
- Bubble down root
- Swap with higher-priority child [if necessary]



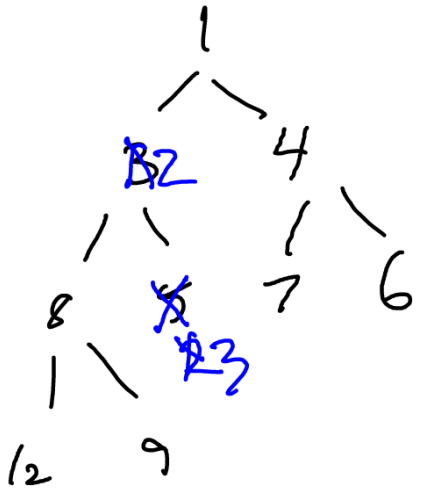
fix Shape



Time: $O(h) = O(\lg n)$

Increase Priority (e, p)

- Set new priority
- Bubble up to restore Order
- Time: $O(h) = O(\lg n)$



How to find element in the array?

Need location.

⇒ store index in element

or in hash table (element → index)

⇒ bubbling changes location

Heapsort

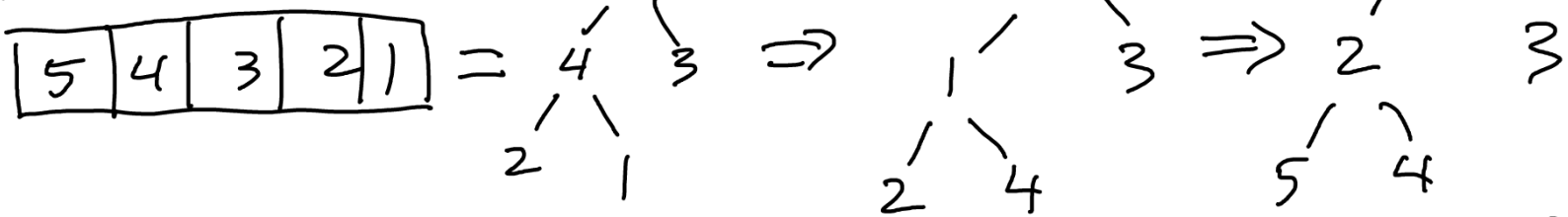
To sort an array:
1. heapify
2. pop() until empty
   — pops elements in priority order.

— $O(n \lg n)$

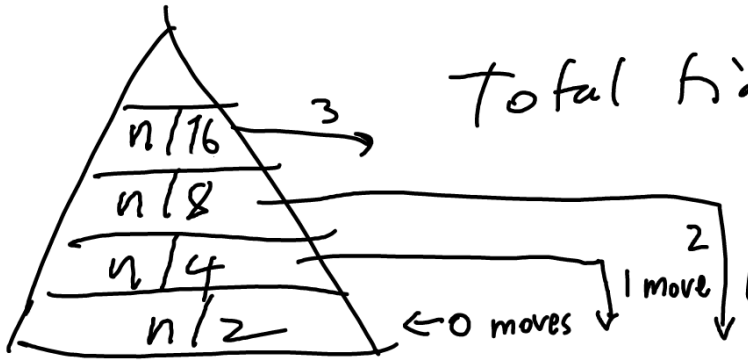
Heapify: $O(n)$

```
for (i = ⌊n/2⌋ - 1; i >= 0; i--) {
    bubbleDown(i);
}
```

$\lfloor \frac{n}{2} \rfloor \ldots n-1$ are leaves



loop invariant: subtrees of nodes $i+1, \ldots, n-1$ are heaps
subtree of i is a heap exc. for i
bubble down ⇒ fix subtree at i



Total time: $\frac{n}{4} \cdot 1 + \frac{n}{8} + 2 + \frac{n}{16} \cdot 3 + \dots$

$t = O(n)$