(Min & Max) BINARY HEAPS

Cristafalk
4/6/17
A2

low priority ← → high priority

- **priority queue** - acts like a queue (FIFO, dequeue by rem item from front) HOWEVER, the logical order of items in queue is determined by their PRIORITY (high priority = front, low priority = back)
  - impl as lists: insert = O(1), sort O(n log n)
  - impl as b. heap: enq & deq = O(log n)
  - ▷ so enqueuing an item may move it to front

- **Binary heap** - Diagram - looks like tree
  Implementation - single list

To maintain the heap property b/w new item & parent, you must swap it down/up (percolate)

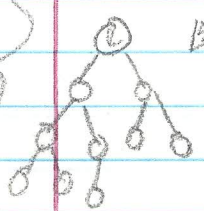
2 variations
- ▷ **min heap** - smallest key is always @ front
- ▷ **max heap** - largest key val @ front

BASIC HEAP OPERATIONS

- BinaryHeap() - creates new, empty bin heap
- insert(k) - adds new item k to heap
- findMin() - returns item w/ min key value
- delMin() - " " and deletes min value
- isEmpty() - True = heap empty
- size() - returns # of items in heap
- buildHeap(list) - builds new heap from list of keys

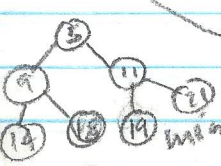
The Structure Property - balanced tree / logarithmic nature of binary tree to make heap

about trees, but applies to heaps



▷ complete binary tree - tree w/ each level having all of its nodes (except bottom level)

(# left nodes = # right nodes)



min heap

[0, 5, 9, 11, 14, 18, 19, 21]

The HEAP ORDER PROPERTY - method to store items in a heap relies on this

"In a heap, for every node x w/ parent p, the key in p is smaller than or equal to the key in x"

CONSTRUCTOR

```
class BinHeap:
    def __init__(self):
        self.heapList = [0]
        self.currentSize = 0
```