


Lecture 18:
Heaps & Heapsort

+ Today 

- Reading
 - JS 13.4-13.6 (Heaps and HeapSort)
- Objectives
 - Midterm review
 - Assignment review
 - Heap operations
 - Heapsort

+ Midterm Exam

- Monday March 10th
- 50 minute, in-class exam
- Covers everything through Splay trees
- Studying essentials
 - The weekly assignment!
 - Study guide and sample exams
 - Form study groups
- No Quiz on Friday

+ Weekly Programming Assignment

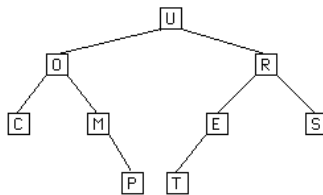
- Last minute change!
- Data structures digest
 - Help you in your studying
 - A list of all you've learned!
- Format the digest in a way that is helpful for you to understand

+ Recap: Priority Queues

- Priority Queue allows access to only the smallest (largest) element
- Implementation based on trees
- Contrasting priority queues
 - Unlike stacks and queues, order in does not determine order out
 - Unlike lists, cannot control where element is stored
 - Cannot traverse a priority queue

+ Array Representation

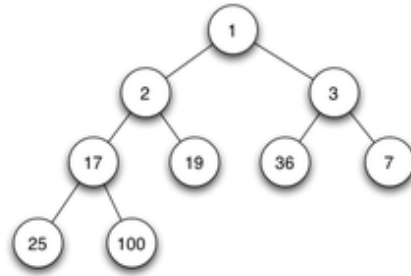
- Let `data` be an array
 - `data[0]` contains the root
 - the left child of `data[i]` is in `data[2*i+1]`
 - the right child of `data[i]` is in `data[2*i+2]`



indices: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
 data: U O R C M E S - - - P T - - -

+ Heaps

- A heap is a complete binary tree whose root contains the minimum value and whose subtrees are, themselves, heaps
- A heap is a complete binary tree whose values are in ascending order on every path from the root to the leaf



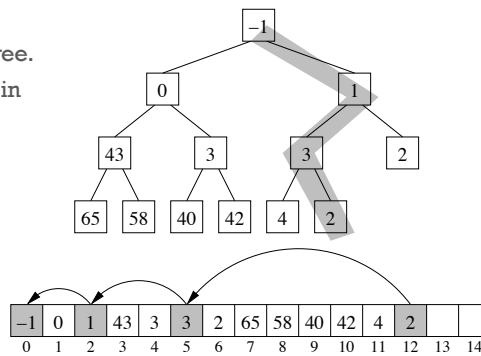
from the course
webpage!

+ Priority Queues Interface

```
public interface PriorityQueue<E extends Comparable<E>> {  
  
    // returns the minimum value  
    public E getFirst();  
  
    // removes and returns the minimum value  
    public E remove();  
  
    // adds a value to the priority queue  
    public E add();  
    ...  
}
```

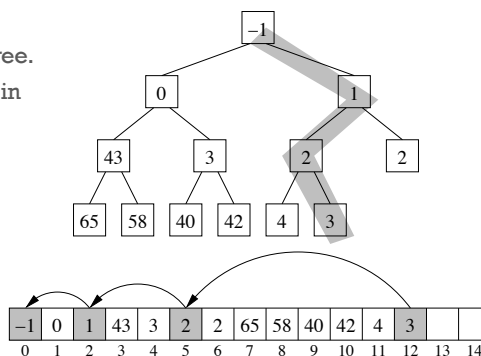
+ Adding to a Heap

- Pre-condition
 - Heap is a complete binary tree.
 - Values along every path are in ascending order
- Adding
 - Add value to end of data []
 - Percolate upward
- Complexity?



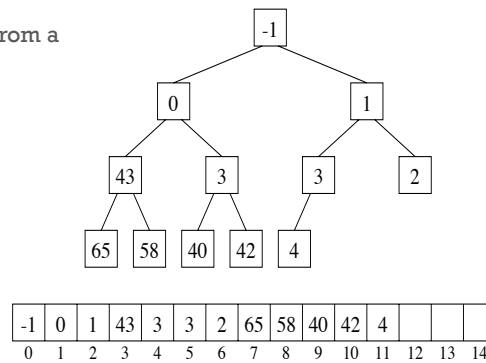
+ Adding to a Heap

- Pre-condition
 - Heap is a complete binary tree.
 - Values along every path are in ascending order
- Adding
 - Add value to end of data []
 - Percolate upward
- Complexity?



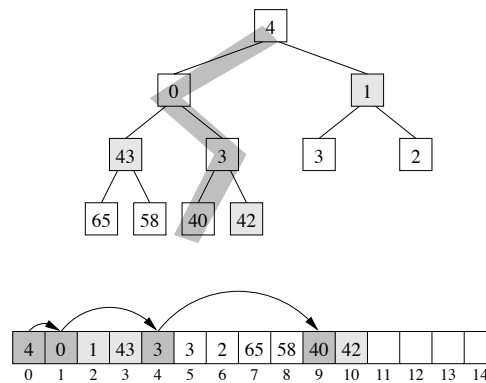
+ Removing from a Heap

- Removing
 - How would we remove from a heap?
- Complexity?



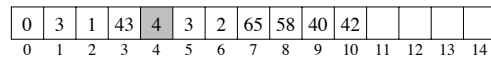
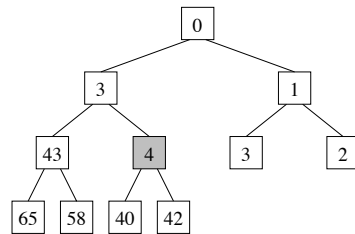
+ Removing from a Heap

- Removing
 - Remove root
 - Move last leaf to root
 - Push down
- Complexity?



+ Removing from a Heap

- Removing
 - Remove root
 - Move last leaf to root
 - Push down
- Complexity?



+ HeapSort

- Suggests a new sorting algorithm!
- Add N integers to a heap
- Remove N integers from the heap
- What is the complexity of HeapSort?
- On what type of data would HeapSort perform well?

+ Comparison of Sorting Algorithms

- Quicksort
 - fastest on average $O(n \log n)$ but worst case is $O(n^2)$
 - takes $O(\log n)$ extra space
- Heapsort
 - $O(n \log n)$ in average and worst case
 - No extra space
 - Bit slower on average than Quicksort & Mergesort.
- Mergesort:
 - $O(n \log n)$ in average and worst case
 - $O(n)$ extra space.
 - Performs well on external files where not all fit in memory.