Lecture 16: Midterm recap + Heaps ii

CSE 373 Data Structures and Algorithms

3 Minutes

Practice: Building a minHeap

Construct a Min Binary Heap by inserting the following values in this order:

5, 10, 15, 20, 7, 2

Min Priority Queue ADT

state

Set of comparable values
- Ordered based on "priority"

behavior

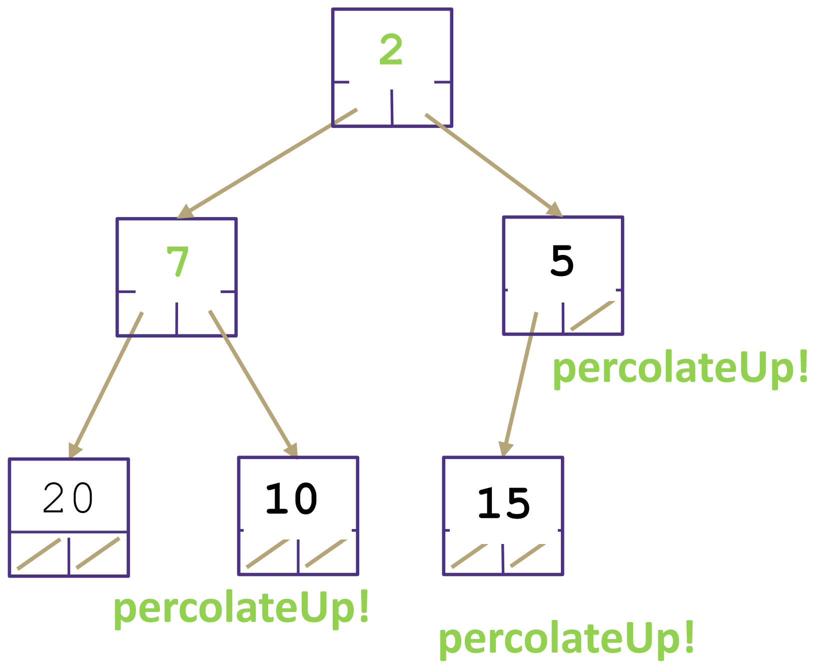
removeMin() – returns the element with the <u>smallest</u> priority, removes it from the collection

peekMin() – find, but do not remove the element with the smallest <u>priority</u>

insert(value) – add a new element to the collection

Min Binary Heap Invariants

1. **Binary Tree** – each node has at most 2 children
2. **Min Heap** – each node's children are larger than itself
3. **Level Complete** - new nodes are added from left to right completely filling each level before creating a new one



Administrivia

HW 4 due Wednesday night

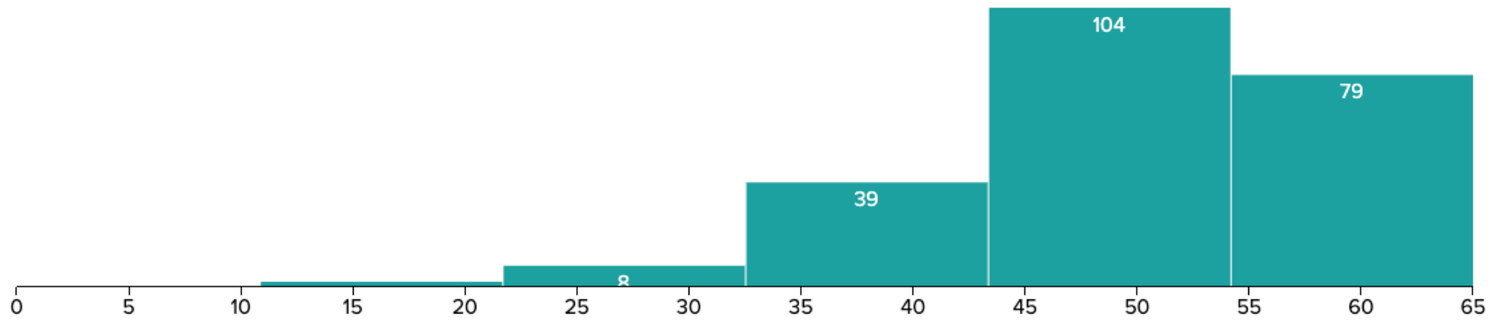
HW 5 out Wednesday (partner project)

- Partner form due tonight

How to get get a tech job with Kim Nguyen

- Today 4-5pm PAA

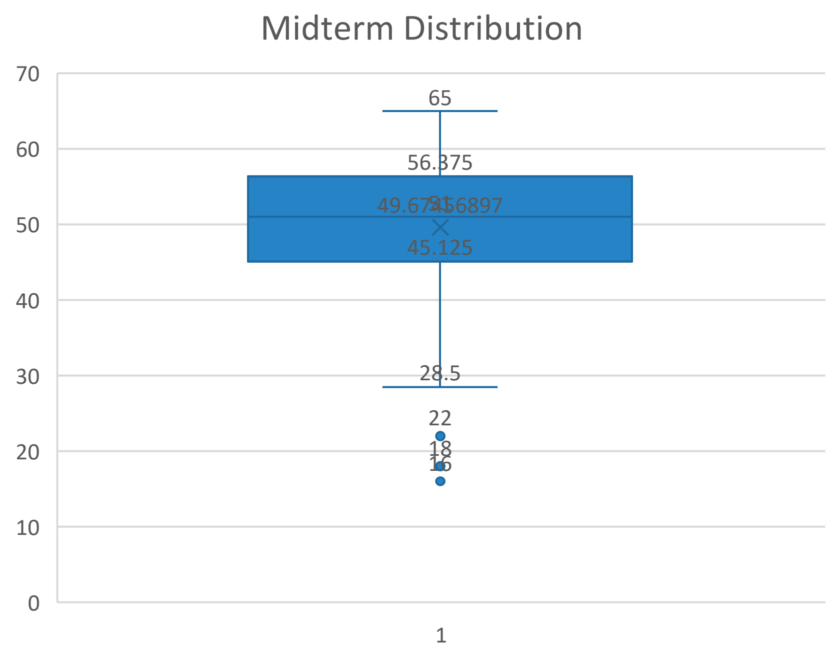
Midterm Grades

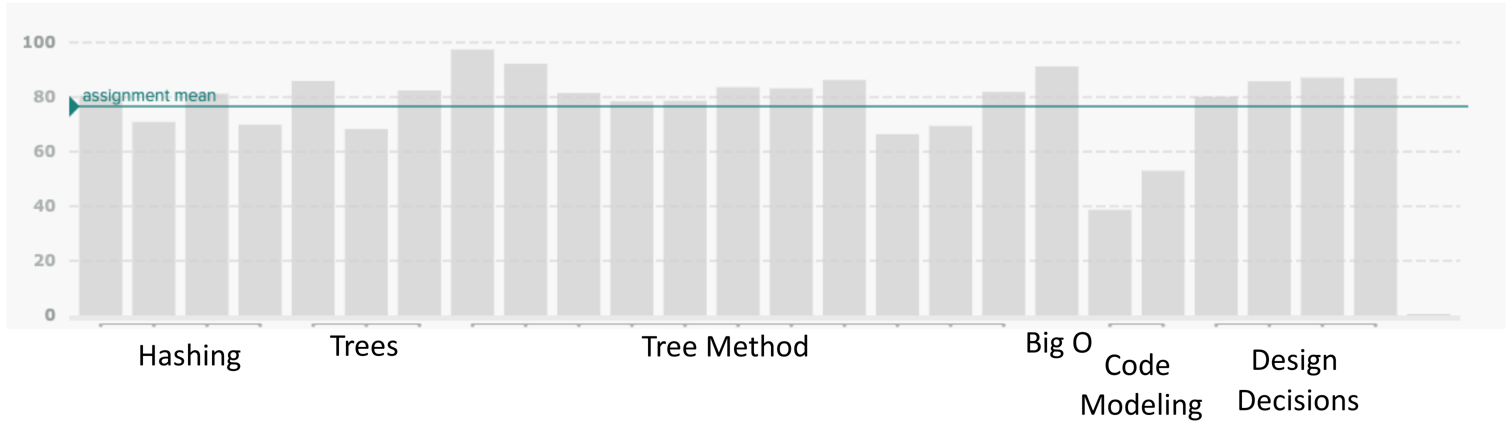


MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
16.0	51.0	65.0	49.67	8.95

Midterm 65.0 points

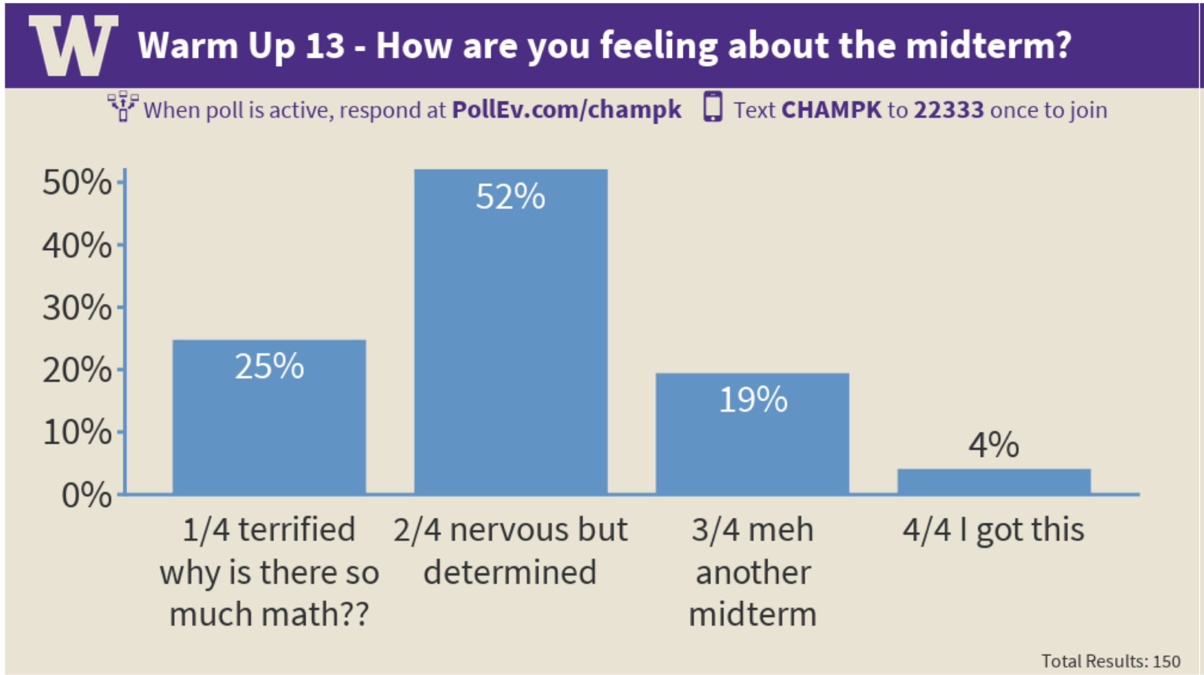
MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
24.62%	78.46%	100.0%	76.42%	13.77%

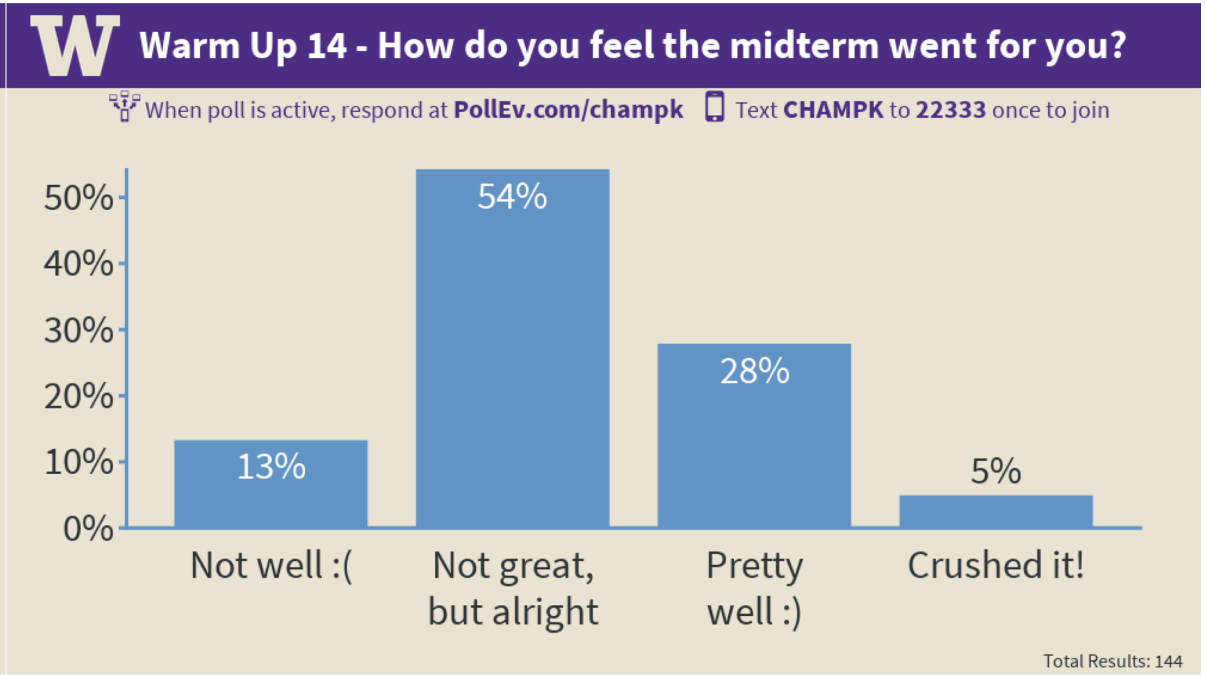


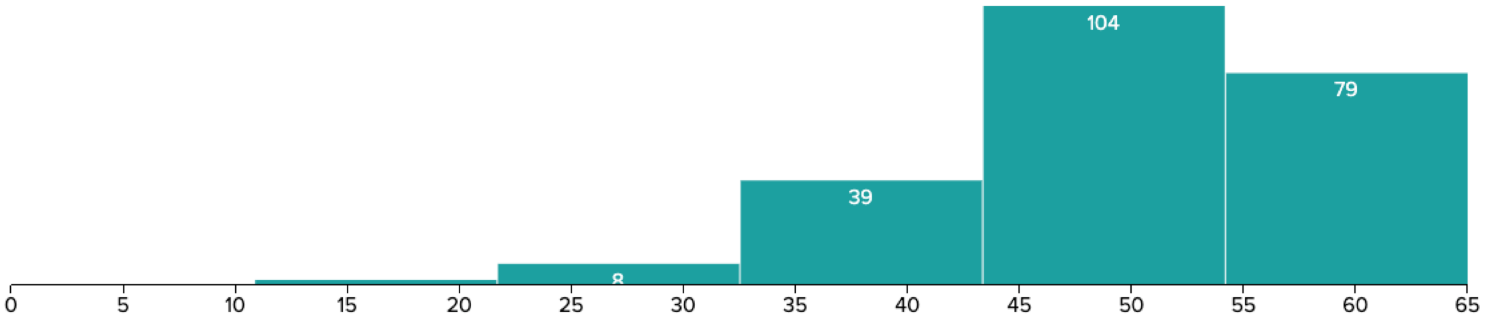


Course Grade Breakdown
Midterm: 20%
Final Exam: 25%
Individual Assignments: 15%
Partner Projects: 40%

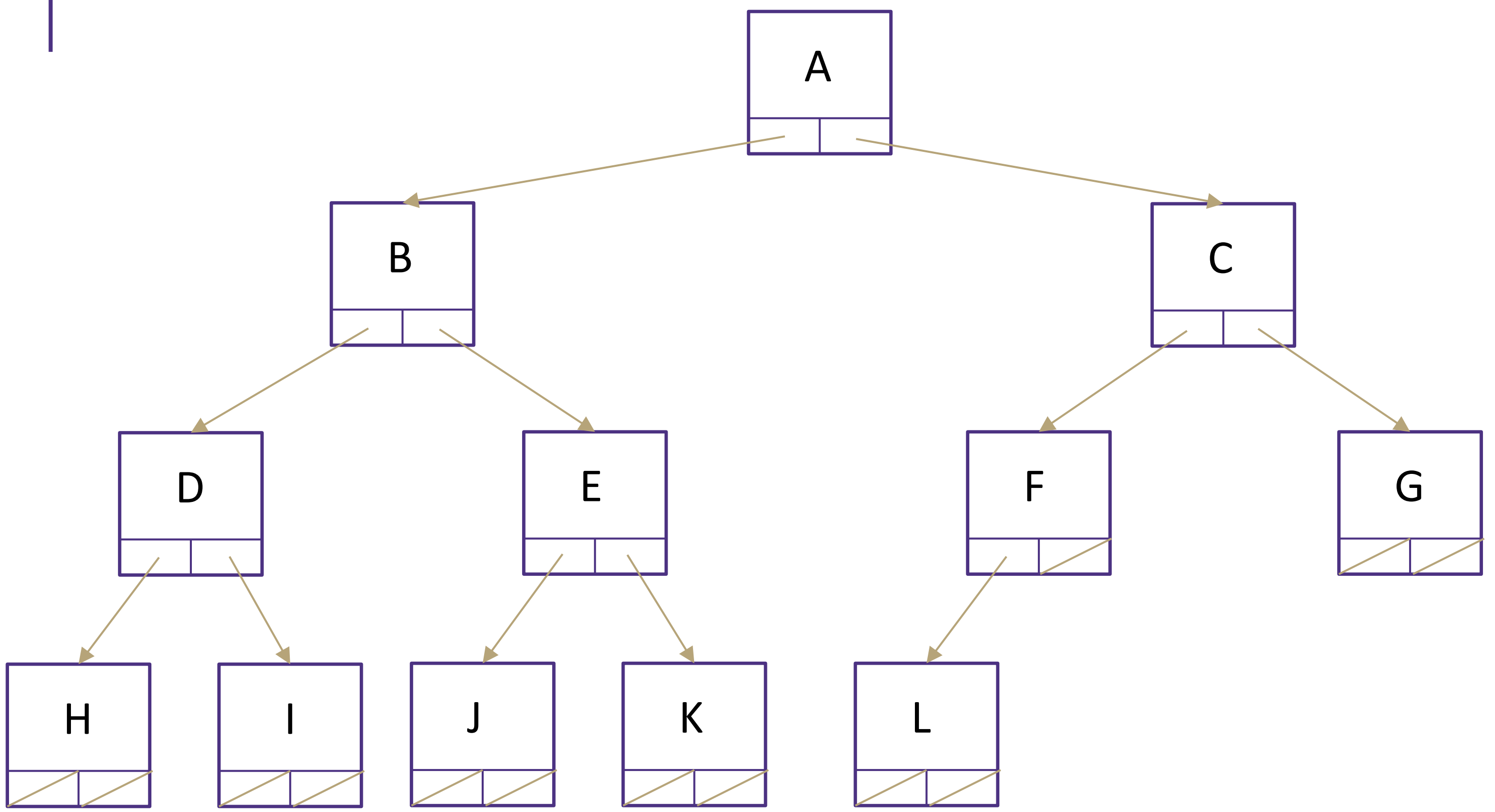
Midterm Performance







Implementing Heaps



Fill array in **level-order** from left to right

0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	E	F	G	H	I	J	K	L		

How do we find the minimum node?

$$peekMin() = arr[0]$$

How do we find the last node?

$$lastNode() = arr[size - 1]$$

How do we find the next open space?

$$openSpace() = arr[size]$$

How do we find a node's left child?

$$leftChild(i) = 2i + 1$$

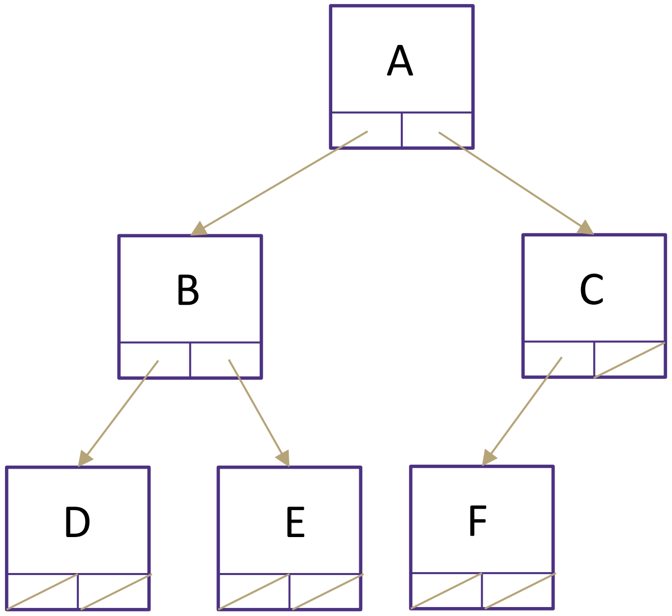
How do we find a node's right child?

$$rightChild(i) = 2i + 2$$

How do we find a node's parent?

$$parent(i) = \frac{(i - 1)}{2}$$

Heap Implementation Runtimes



0	1	2	3	4	5	6	7
A	B	C	D	E	F		

`char peekMin()`

timeToFindMin

Tree $\Theta(1)$

Array $\Theta(1)$

`char removeMin()`

findLastNodeTime + removeRootTime + numSwaps * swapTime

Tree n + 1 + log(n) * 1 $\Theta(n)$

Array 1 + 1 + log(n) * 1 $\Theta(\log(n))$

`void insert(char)`

findNextSpace + addValue + numSwaps * swapTime

Tree n + 1 + log(n) * 1 $\Theta(n)$

Array 1 + 1 + log(n) * 1 $\Theta(\log(n))$

Building a Heap

Insert has a runtime of $\Theta(\log(n))$

If we want to insert a n items…

Building a tree takes $O(n\log(n))$

- Add a node, fix the heap, add a node, fix the heap

Can we do better?

- Add all nodes, fix heap all at once!

Cleaver building a heap – Floyd's Method

Facts of binary trees

- Increasing the height by one level doubles the number of possible nodes
- A complete binary tree has half of its nodes in the leaves
- A new piece of data is much more likely to have to percolate down to the bottom than be the smallest element in heap

1. Dump all the new values into the bottom of the tree
   - Back of the array
2. Traverse the tree from bottom to top
   - Reverse order in the array
3. Percolate Down each level moving towards overall root

see lecture 16 slides for example / animations