Lecture 16: Midterm recap + Heaps ii
**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 smallest priority, removes it from the collection
- `peekMin()` – find, but do not remove the element with the smallest priority
- `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

---

![Min Binary Heap Diagram]

- `percolateUp!`
- `percolateUp!`
- `percolateUp!`
Administrivia

HW 4 due Wednesday night

HW 5 out Wednesday (partner project)
- Partner form due tonight

How to get a tech job with Kim Nguyen
- Today 4-5pm PAA
Midterm Grades

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

Midterm Distribution

MINIMUM 16.0
MEDIAN 51.0
MAXIMUM 65.0
MEAN 49.67
STD DEV 8.95

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

Hashing Trees Tree Method Big O Code Modeling Design Decisions
Midterm Performance

Warm Up 13 - How are you feeling about the midterm?

- 1/4 terrified: 25%
- 2/4 nervous but determined: 52%
- 3/4 meh another midterm: 19%
- 4/4 I got this: 4%

Total Results: 150

Warm Up 14 - How do you feel the midterm went for you?

- Not well :(: 13%
- Not great, but alright: 54%
- Pretty well :): 28%
- Crushed it!: 5%

Total Results: 144
Implementing Heaps

How do we find the minimum node?
\( \text{peekMin}() = \text{arr}[0] \)

How do we find the last node?
\( \text{lastNode}() = \text{arr}[\text{size} - 1] \)

How do we find the next open space?
\( \text{openSpace}() = \text{arr}[\text{size}] \)

How do we find a node’s left child?
\( \text{leftChild}(i) = 2i + 1 \)

How do we find a node’s right child?
\( \text{rightChild}(i) = 2i + 2 \)

How do we find a node’s parent?
\( \text{parent}(i) = \frac{(i - 1)}{2} \)

Fill array in level-order from left to right

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
<td>J</td>
<td>K</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CSE 373 19 SP - KASEY CHAMPION
Heap Implementation Runtimes

char peekMin()
	timeToFindMin

Tree \( \Theta(1) \)
Array \( \Theta(1) \)

cchar removeMin()
	findLastNodeTime + removeRootTime + numSwaps * swapTime

Tree \( n + 1 + \log(n) * 1 \quad \Theta(n) \)
Array \( 1 + 1 + \log(n) * 1 \quad \Theta(\log(n)) \)

void insert(char)
	findNextSpace + addValue + numSwaps * swapTime

Tree \( n + 1 + \log(n) * 1 \quad \Theta(n) \)
Array \( 1 + 1 + \log(n) * 1 \quad \Theta(\log(n)) \)
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!
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