CSE 373: Heaps

*(other operations and variations)*

Chapter 6

---

**Heaps: Quick Recap**

*Heaps:*
- structure is a complete binary tree
- each node must be smaller than its descendants

![Heap Diagram]

- main operations are `insert()` and `deleteMin()`
- heaps have a compact array-based representation
Other Operations: `decreaseKey()`

`decreaseKey()` – lowers a node’s value
while preserving heap ordering

```
H.decreaseKey(1, 6);
```

```
        2
       / \
      4   5
     /   /\
    7   6 10
   / \ / \  
  11 9 12 14 20
```

running time?
**increaseKey()**

*increaseKey() – raises a node’s value*

```c
H.increaseKey(5, 6);
```

**increaseKey() – Continued**

Running time?
**delete()**

**delete()** – removes a node from the heap

H.delete(\ 6);

**delete() – Continued**

running time?
Let’s Write a Heap Routine...

```python
buildHeap()
```

`buildHeap()` – creates a heap from an array

```
[12 5 11 3 10 6 9 4 8 1 7 2]
```

*Straightforward Implementation: `insert()` elements into an empty heap one at a time*

`running time?`
**buildHeap() – Continued**

*Better Implementation:* Treat input array as a heap and “percolate down” first \( n/2 \) values

12 5 11 3 10 6 9 4 8 1 7 2

running time?

**buildHeap() – even more**
**buildHeap() running time**

```
  7
  2 3+4 5 6 7 6 89: ; ; ;=< > ?A@ B @ C DAE F E > F G H I F H G J K E L MAN O P Q G R F S TK
       U V W XZY \[ W 8Z\] V ^ W _ 9
```

**MaxHeaps**

*MaxHeaps*: the dual of the Heaps we’ve defined

- support fast `insert()` and `deleteMax()` ops
- work exactly the same as (Min)Heaps

Why is `deleteMax()` expensive on a normal minheap?

What’s the running time?
\( d \)-Heaps

\( d \)-Heaps: Just like normal heaps but with \( d \) children rather than 2

Intuition: tree will be shallower so ops will be faster
However...
- more comparisons need to be made when percolating down
- finding parent/children may be slower

What about asymptotic running time?

Bottom Line: 4-heaps \textit{may} outperform 2-heaps

Merging Heaps

How to merge heaps effectively?

\textit{Straightforward method:} copy both arrays into a single array and use \texttt{buildHeap()}
running time?

\textit{Advanced methods:}
- pointer-based imbalanced heaps
  - \textit{leftist heaps} – a bit like AVL trees; \( O(\log n) \) merge
  - \textit{skew heaps} – like Splay trees; \( O(\log n) \) amortized ops
  - \textit{binomial queues} – \( O(\log n) \) merge, but \( \sim O(1) \) insert