Tutorial 2, Sep 22, 2023

Heapify

- The MAX-HEAPIFY operation restores heap property for a single element by swapping it with the greater of its two children if needed, and then recursing on that child
  - The precondition is that the left and right subtrees are valid heaps
- Define a procedure BUILD-HEAP, which takes any arbitrary array, and then turns it into a valid max-heap by calling MAX-HEAPIFY on every element, in reverse, starting from halfway through the array
  - The second half of the array is ignored because those nodes will all be leaves, so they are already valid max-heaps
  - Note we have to go in reverse to satisfy the precondition for MAX-HEAPIFY
- We know BUILD-HEAP is at least $\Omega(n)$; if we give it an array that's already a valid heap, it will still traverse it
- The upper bound is clearly at most $O(n \log n)$ but can we do better?
  - Calling MAX-HEAPIFY on a node on level d takes $O(h-d)$
  - In the worst scenario, every single node calls MAX-HEAPIFY and recurses down every single level
    * The runtime would be $\displaystyle\sum_{d=0}^{h-1} 2^d c(h-d)$, since there are 2^d nodes in level d and each performs $h-d$ operations
    * Let $i = h-d$ so we have $\displaystyle\sum_{i=1}^{h} 2^{h-i} i = c2^h \sum_{i=1}^{h} \frac{i}{2^i}$
      - Recall $\displaystyle\sum_{i=1}^{\infty} x^i = \frac{1}{1-x} \implies \sum_{i=1}^{\infty} i x^{i-1} = \frac{1}{(1-x)^2} \implies \sum_{i=1}^{\infty} i x^i = \frac{x}{(1-x)^2}$
      - Therefore $\displaystyle\sum_{i=1}^{h} \frac{i}{2^i} < \sum_{i=1}^{\infty} \frac{i}{2^i} = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = 2$
      - So our overall bound is $2c2^h = 2c2^{\log n} = 2cn = O(n)$