

CS32: Handout for h18

Additional notes on heaps

Here are some notes to help you with the reading assignment in DS 11.1 and 11.2.

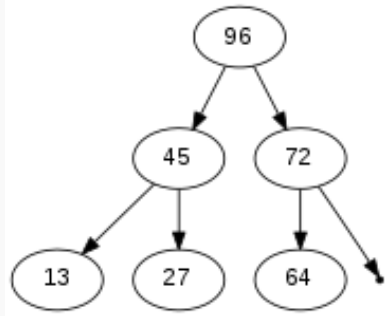
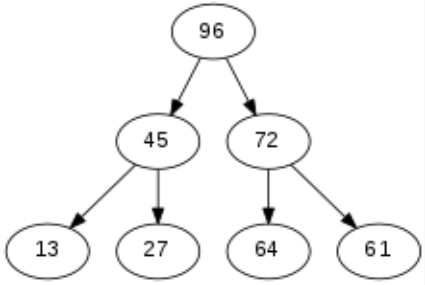
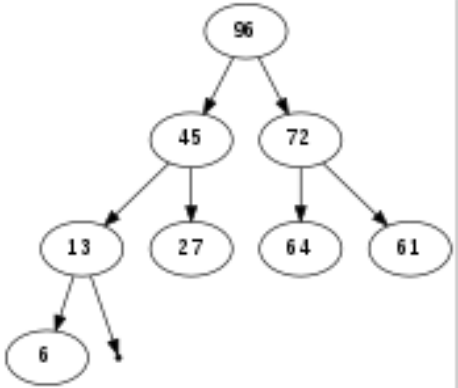
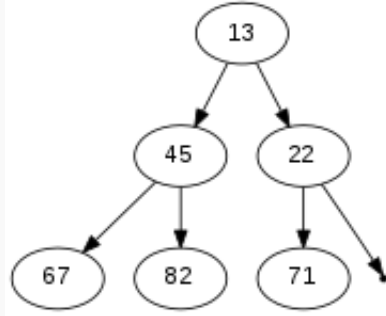
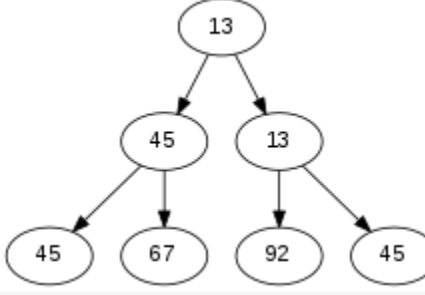
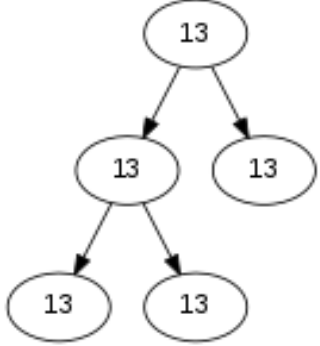
DS introduces a heap as "a binary tree where the entries of the nodes can be compared with the less-than operator of a strict weak ordering [where] in addition, two rules are followed...". The two rules can be paraphrased as:

- (1) **max-heap property**: value in each node is \geq value of children
- (2) **heap shape**: any missing children are in the bottom row at the right.

In the broader sense, this is a definition for a specific kind of heap, known as a "max-heap", because the value at the root is always the largest value (max) in the heap. If we replace rule (1) with a rule that the value in each node must be \leq the value in its children, i.e. the **min-heap property**, then we have a min-heap. It is important to note that the order is between parent and child; **there is no ordering among children**. At any level, the key of the left child may be larger or smaller than the right child, and this **does not matter**—the technical term is that this order is "arbitrary".

A heap is one implementation for a "priority queue", because it forms a queue where the discipline of "who goes first" is set by the priority (i.e. the value of the key. In a priority queue, the value that is "removed" is typically the value at the root (via a delete-max or delete-min operation), followed by internal book-keeping to "fix up the heap" to restore the two properties. This "fixing up" is sometimes called "heapifying".

Note that although the textbook never shows an example of this, the relationship between nodes allows for ties, so a heap that contains all the same number can be a valid heap—although a sort of silly one. The term "degenerate case" is sometimes used to describe situations like this (see: http://en.wikipedia.org/wiki/Degeneracy_%28mathematics%29)

Max Heap with 6 nodes	Max Heap with 7 nodes	Max Heap with 8 nodes
 <pre> graph TD 96((96)) --> 45((45)) 96 --> 72((72)) 45 --> 13((13)) 45 --> 27((27)) 72 --> 64((64)) </pre>	 <pre> graph TD 96((96)) --> 45((45)) 96 --> 72((72)) 45 --> 13((13)) 45 --> 27((27)) 72 --> 64((64)) 72 --> 61((61)) </pre>	 <pre> graph TD 96((96)) --> 45((45)) 96 --> 72((72)) 45 --> 13((13)) 45 --> 27((27)) 13 --> 6((6)) 72 --> 64((64)) 72 --> 61((61)) </pre>
Min Heap with 5 nodes	Min Heap with 7 nodes (note duplicate keys)	Min and Max Heap with 5 nodes (extreme degeneracy)
 <pre> graph TD 13((13)) --> 45((45)) 13 --> 22((22)) 45 --> 67((67)) 45 --> 82((82)) 22 --> 71((71)) </pre>	 <pre> graph TD 13((13)) --> 45((45)) 13 --> 13((13)) 45 --> 45((45)) 45 --> 67((67)) 13 --> 92((92)) 13 --> 45((45)) </pre>	 <pre> graph TD 13((13)) --> 13((13)) 13 --> 13((13)) 13 --> 13((13)) 13 --> 13((13)) </pre>