Heap Sort

CLRS Reading: Sections 6.1 - 6.4, pages 147 - 162

Sort of Sorted

908

703

897

275

503

512

061

170

087

154
Definitions

heap
binary tree (complete or almost complete) that obeys a heap condition

binary tree
tree in which each node can have up to two children

perfect/completed binary tree
binary tree in which all the leaves have the same depth (distance from the top / length of a path from the root to the node)

height
distance from the bottom / length of the longest path from the bottom of the tree to the node

almost complete binary tree
binary tree missing the right portion of the last row

heap condition/property
property of the binary tree that must hold for it to be a certain type of heap

max-heap condition
for all nodes i other than the root, $A[\text{Parent}(i)] \geq A[i]$
### Heaps

The array implementation of heaps make it particularly easy for parents to find children and vice versa.

### Families Reunited

- The array implementation of heaps make it particularly easy for parents to find children and vice versa.
Maintaining the Heap Property

node out of position

Max-Heapify

```
Max-Heapify(A, i)
  l = Left(i)
  r = Right(i)
    largest = l
  else largest = i
    largest = r
  if largest != i
    exchange A[i] with A[largest]
  Max-Heapify(A, largest)
```
**Build-Max-Heap**

Build-Max-Heap(A)
A.heap-size = A.length
for i = \(\lceil A.length/2 \rceil \) downto 1
  Max-Heapify(A, i)

---

**The Cost of Building a Heap**

*Lemma.* An \(n\)-element heap has height \(\lceil \lg n \rceil\) and has at most \(\lceil n/2^{h+1} \rceil\) nodes of height \(h\).