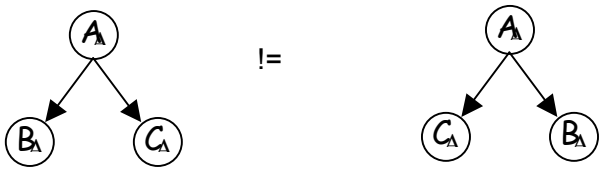
Heap sort

Def. A *binary tree* is a tree for which every node has a maximum of 2 children.

We will be looking at trees for which every node has a sequence of children and not just a set of children. When we speak of a sequence, we mean that the order of the elements is of importance.

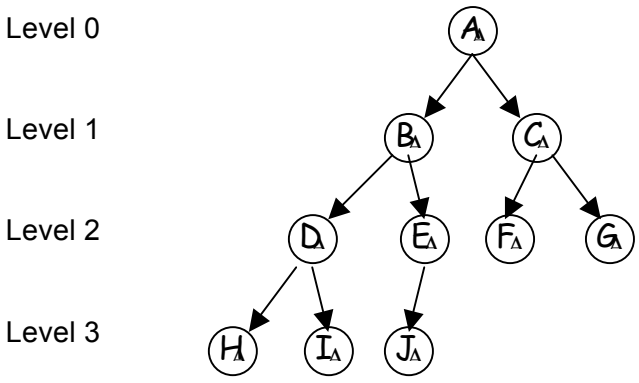


The tree nodes can be divided into levels.

Level 0 only contains the root node of the tree.
Level 1 contains all of the children of the root.
Level 2 contains all of the children of all of the nodes in level 1.
...
Level i contains all of the children of all of the nodes in level i-1.

Def. A binary tree is called *full*, when all levels except for the last are filled in, that is, the level i consists of exactly 2i nodes. The last level may contain less than 2i nodes.

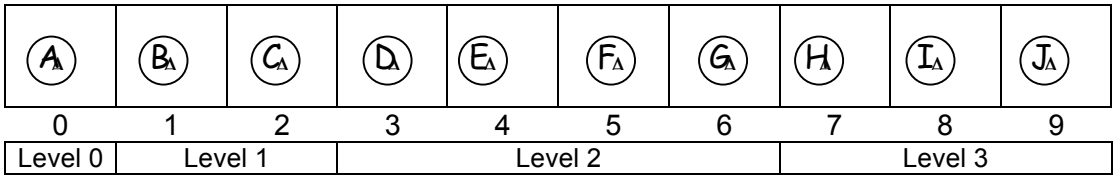
Def. A binary tree is called *complete* when it is full and all of the nodes in the last level are left justified and fully filled in.



If F were to be missing, the tree would not be full. If H were to be missing, the tree would not be complete

---

Trees can be represented in different ways in a computer. We have already seen how a tree can be represented using Object references to the child nodes. These references use some storage space, even if it is not much compared to the space used for the objects. For complete binary trees there is a very elegant representation, that does not use references and which is extremely handy for implementing things like priority queues. We can store a tree with N nodes in an Array F with N elements. We just write the nodes in level order from top to bottom, left to right, in the array.



The array element F[0] contains the root, the components F[1] and F[2] the children of F[0], F[3] and F[4] are the children of F[1] and so on. In general:

1. The children of F[i] are F[2i+1] and F[2i+2].
2. The parent of F[i] is F[(i-1)/2] (/ is integer division that truncates).

Def.: A heap fulfils the *heap condition* when no node F[i] is larger than its parent

How can you take an „almost heap“ that fulfils the heap condition everywhere except at the root, and make it into a heap that fulfils the heap condition? The algorithm `trickle` (sometimes called *seep* or *heapify* or *percolate*) will trickle the wrong root node down through the tree until it is sitting at a proper place.

trickle

1. Make a copy of the root, call it k1
2. 2. Now replace, starting at the root, the node by the largest child, and then this child with its largest child, until a node (called k2) has been copied up, that only has children that are less than or equal to k1.
3. Replace k2s former position with k1.

The heap sort algorithm consists of two major steps:

1. Make a heap out of the array that is to be sorted.
2. Make a sorted array out of the heap.

An algorithm for step 1:
Go through the array A from right to left and make the current element A[i] the root of a “little heap”. Call `trickle(A[i])`. When this has been done for the first element, the array is now a heap

An algorithm for step 2:
2.1 Let i assume all values from N-1 down to 1. For each i:
2.1.1 Swap A[0] with A [i]
2.1.2 Apply `trickle` to the subarray A[0..i-1] .