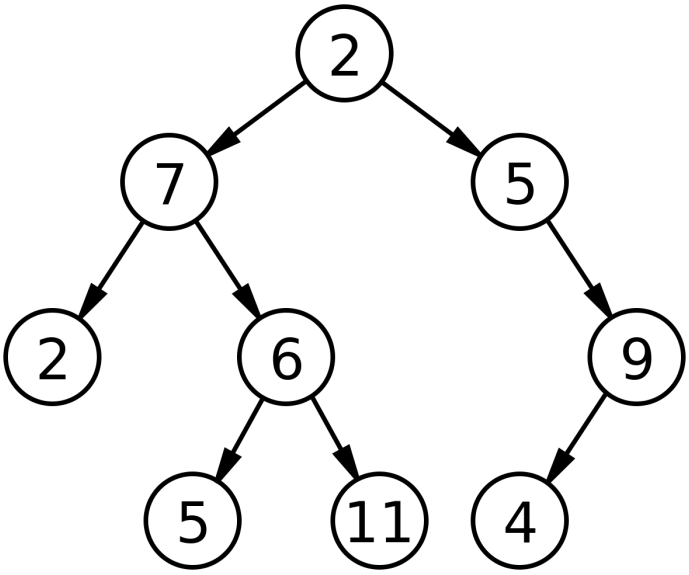
Heap sort

What is a binary tree?




```
A = [2, 7, 5, 2, 6, 9, -, -, 5, 11, 4]
```

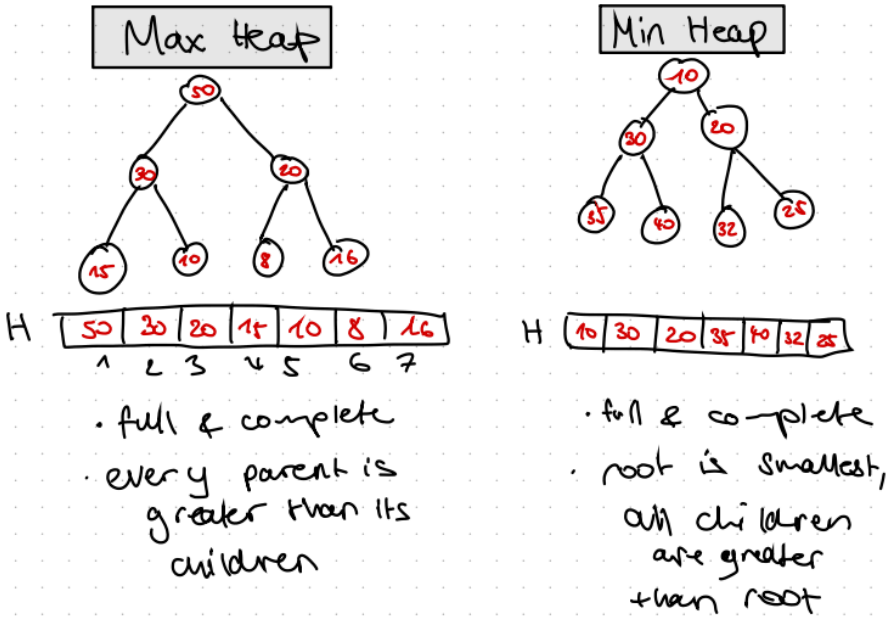
```
node at index i
left child at index 2*i
right child at index (2*i+1)
parent at floor(i/2)
```

We start at the root of the tree (2), and then display the left (7) and right (5) child of the root. Each node is not allowed to have more than one parent. The only node, which has no parent, is the root. A leaf is an element (here 2, 5, 11, 4) which has no children.

The depth of a binary tree is the level where an element sits. The root is level 0. The height of the binary tree is determined what the longest path of a node to a leaf is. The root is level 1.

How do you build a heap?

Heaps can be found in 2 variations, max heaps and min heaps.



Complete binary trees

A complete binary tree is a tree where every node has 2 child elements. The formula for each level is 2^k - this means a binary tree with 3 levels has $2^3 = 8$ elements.

Nearly complete binary trees

A nearly complete binary tree has all elements need upto the $k-1$ level. The last level must have it's children as far left as possible.

Heapify

Heapify is the process used to "clean up" an unsorted binary tree. By default, we sort it to a Max Heap.

```
void Heapify(int A[], int i, int s){
    int m = i;
    int l = Left(i);
    int r = Right(i);

    // Checks if left exists and if left elements is larger than parent
    if (l < s && A[l] > A[m]){
        m = l;
    }
```

```
    // Checks if right exists and if right elements is larger than parent
    if (r < s && A[r] > A[m]){
        m = r;
    }


    if (i != m){
        exchange(A[i], A[m]);
        Heapify(A, m, s);
    }
}
```

Quick sort

Lomuto partitioning

Hoare partitioning