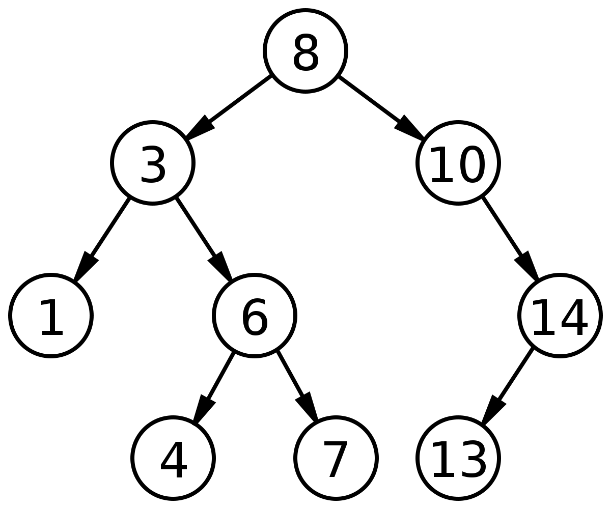
# Binary Search Tree （二叉搜索树）

* How does it look like?



* Top level is known as the Parent Node and each node will have 2 children (a left and a right).
  + The left-child < parent node while the right-child > parent node.
  + Hence the sequence: Left-Child < Parent Node < Right-Child
* A few functions that will be required by A-Level Computing Practical: (refer to pg. 3 for the different cases)
  + Insert
  + Search
  + Delete
  + Update
  + Find Min. and Max.
    - Minimum: Keep searching left branch until you reach the end. (ie. The last left child at the last level)
    - Maximum: Keep searching right branch until you reach the end. (ie. The last right child at the last level)
  + Printing
    - In order (Traversal)
      * In ascending order
      * Left Child → Root → Right Child
    - Post order (Traversal)
      * Left Child → Right Child → Root
    - Pre order (Traversal)
      * Root → Left Child → Right Child
* Normally comes out as Practical Qns 3 or 4.
* Refer to Python IDLE for BST python codes.

# Methods Explanation of the Different Cases (BST)

* Insert (self, Input\_Data)
  + Insert is the most basic operation in BST, after creating your constructor this will normally be your first function to include.
    - 2 main cases:
      1. Input data < Current data (Parent Node)
         * Goes left
         * 2 subcases

When current left child is none 🡺 set left child to Input data

Else, recursively run the input function in the left child

* + - 1. Input data > Current data (Parent Node)
         * Goes right
         * Same as above, change left to right.