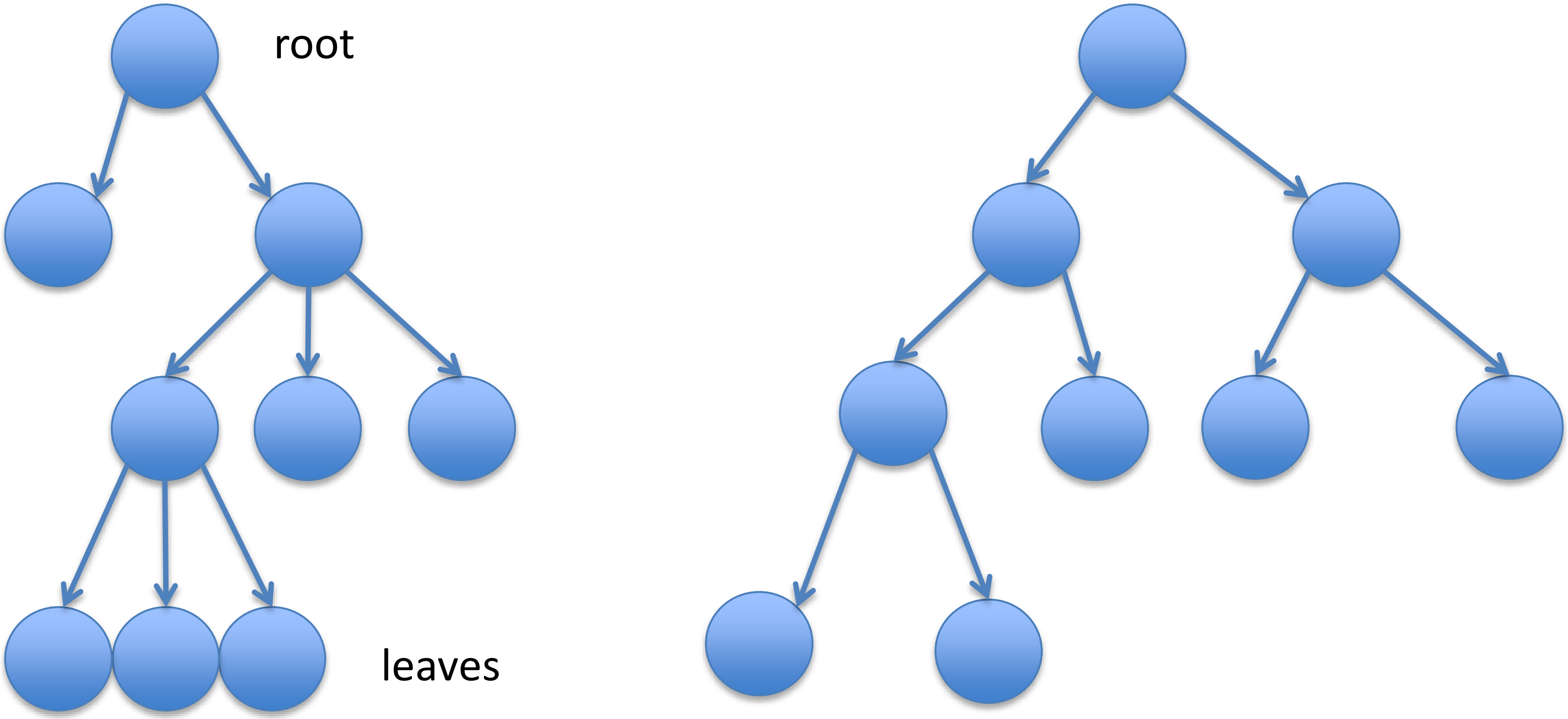
Trees and tree search

- Trees are a useful data structure
    - Convenient for storing data that is hierarchical
    - Stores data in a manner that simplifies searching for information
    - Can capture information
- Search algorithms for trees can be quite efficient
- Particularly useful for making decisions on problems

Tree definition

- A tree consists of one or more nodes
  - A node typically has a value associated with it
- Nodes are connected by branches
- A tree starts with a root node
- Except for leaves, each node has one or more children
  - We refer to a node which has a child as the parent node
- In simple trees, no child has more than one parent, but the generalization (often called a graph) is also very useful

Example trees



Binary trees

- A binary tree is a special version of a tree, where each node has at most two children
- Binary trees are very useful when storing and searching ordered data, or when determining the best decision to make in solving many classes of problems
  - Such trees are often called decision trees

Binary tree class

```
class binaryTree(object):
    def __init__(self, value):
        self.value = value
        self.leftBranch = None
        self.rightBranch = None
        self.parent = None
    def setLeftBranch(self, node):
        self.leftBranch = node
    def setRightBranch(self, node):
        self.rightBranch = node
    def setParent(self, parent):
        self.parent = parent
    # and other methods
```

Binary tree class

```
class binaryTree(object):
    # and other methods
    def getValue(self):
        return self.value
    def getLeftBranch(self):
        return self.leftBranch
    def getRightBranch(self):
        return self.rightBranch
    def getParent(self):
        return self.parent
    def __str__(self):
        return self.value
```

Constructing an example tree

```
n5 = binaryTree(5)
n2 = binaryTree(2)
n1 = binaryTree(1)
n4 = binaryTree(4)
n8 = binaryTree(8)
n6 = binaryTree(6)
n7 = binaryTree(7)
```

```
n5.setLeftBranch(n2)
n2.setParent(n5)
n5.setRightBranch(n8)
n8.setParent(n5)
n2.setLeftBranch(n1)
n1.setParent(n2)
n2.setRightBranch(n4)
n4.setParent(n2)
n8.setLeftBranch(n6)
n6.setParent(n8)
n6.setRightBranch(n7)
n7.setParent(n6)
```

An example tree