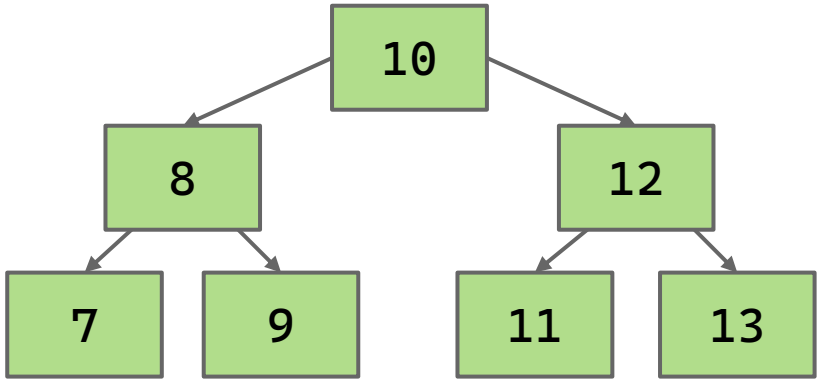
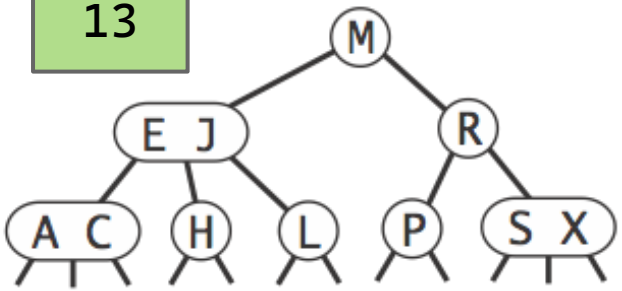
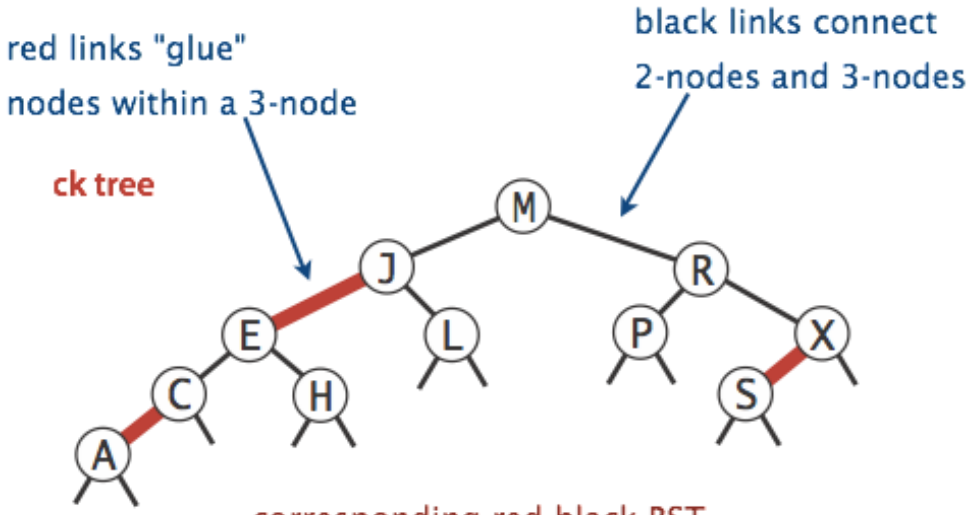


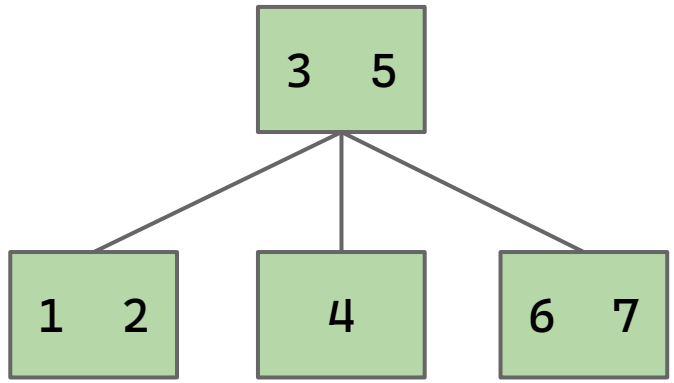


2-3 tree



corresponding red-black BST

BSTs, B-trees, AVL trees, Red-black trees



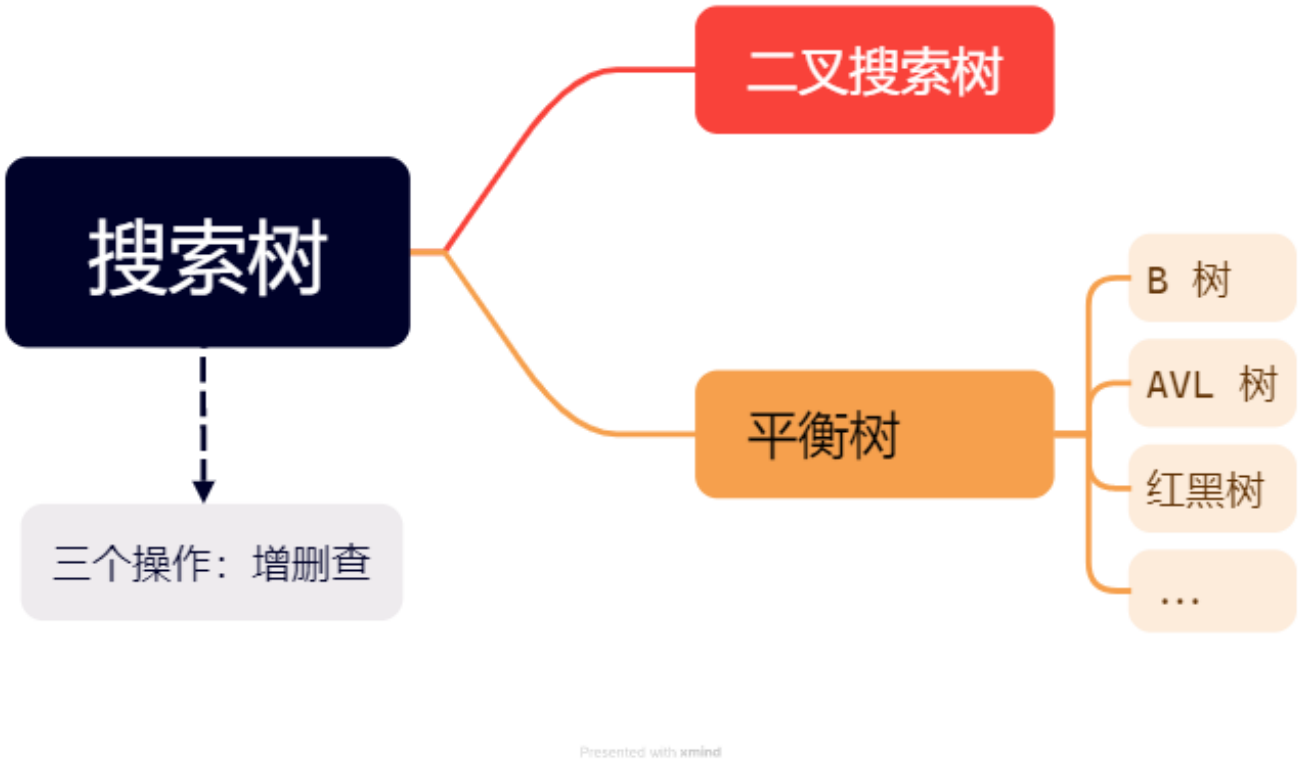
主讲人：七海Nana7mi

课程大纲：CS61B

课程说明：

- 课程内容基于 UC-Berkeley 的课程 CS61B-sp18 与 CS61B-fa23。可以理解为课程的汉化视频。
- 课程使用的编程语言为 Java。
- AI 语音模型来源 BiliBili 用户 Xz乔希。
- 七海也在学习中，有错误敬请指出！

章节目录



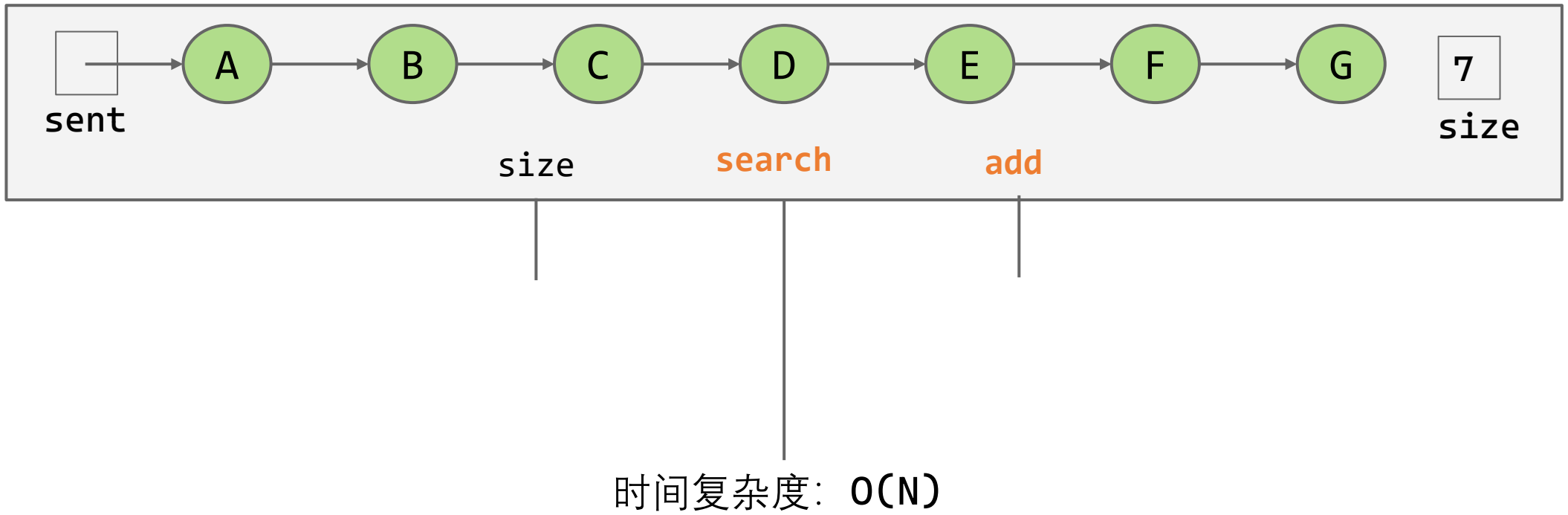
二叉搜索树：导入

Lecture 1

二叉搜索树
- 导入
- 定义
- `contains()`
- `insert()`
- `delete()`

二叉搜索树的应用

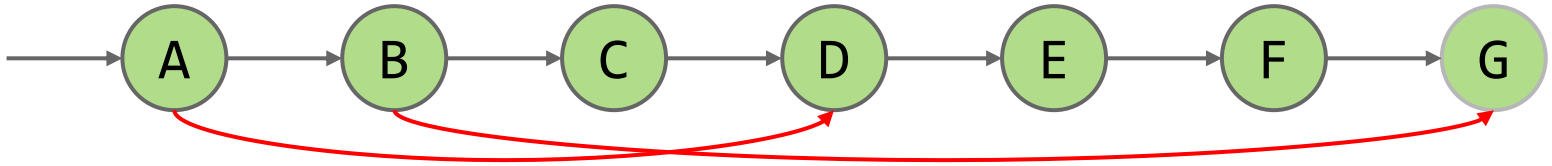
For the *order linked list* implementation below, an operation of search can take worst case linear time, Θ(N).



- How to do?

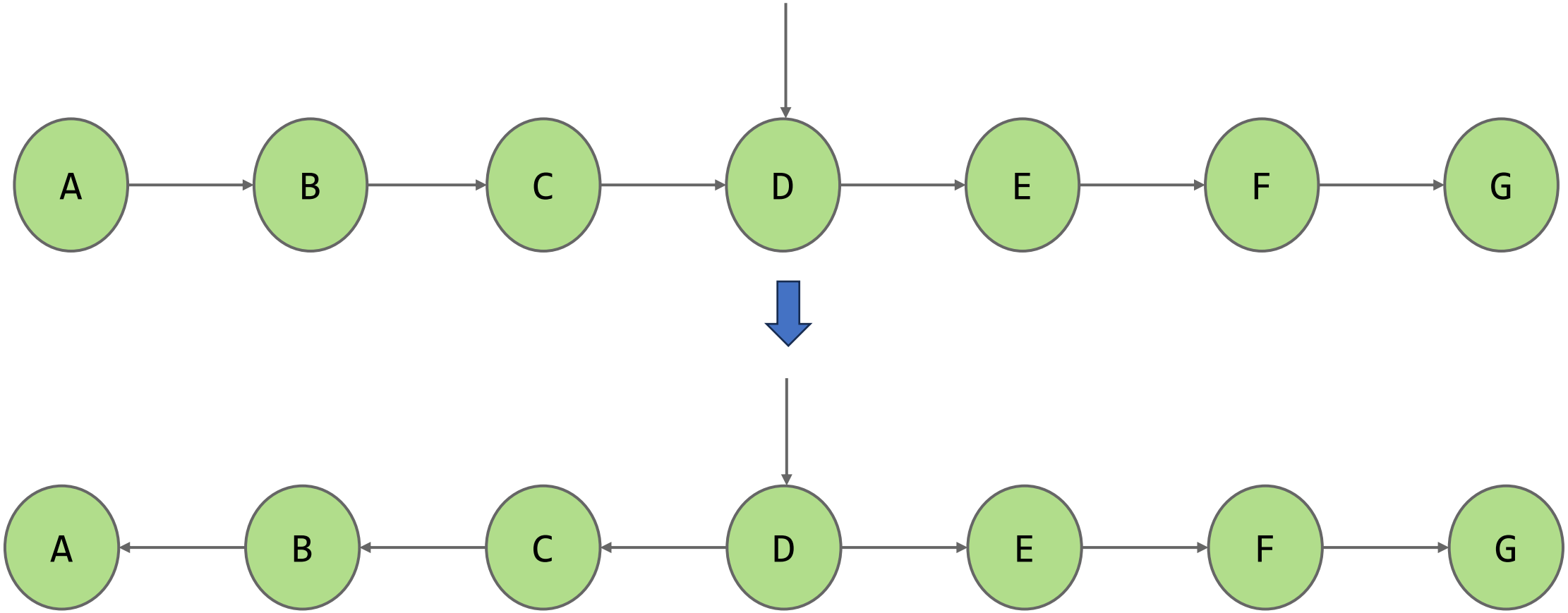
Fundamental Problem:
Slow search, even though it’s in order.

- 我们可以任意的增加不同元素之间的连接线，来缩短路程



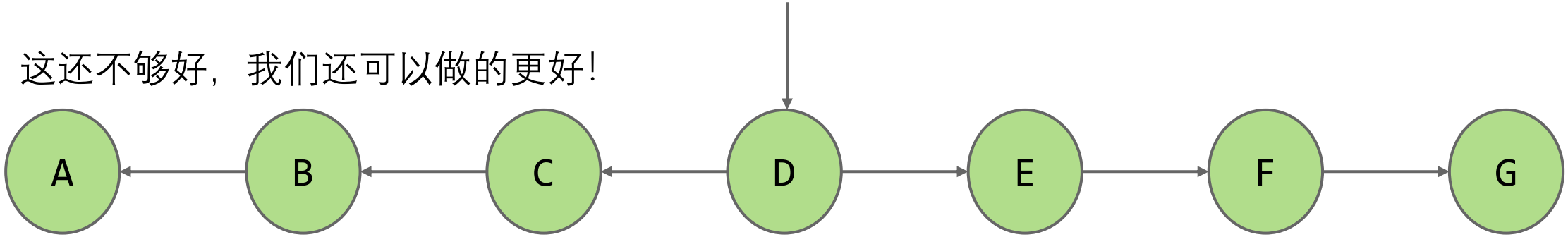
这种方法可以改进为一中数据结构也被叫做跳跃列表，我们对他的讲解讲止步于此，感兴趣的脆鲨可以自行搜索。

- 利用有序性，我们可以把入口指针指向中间的元素



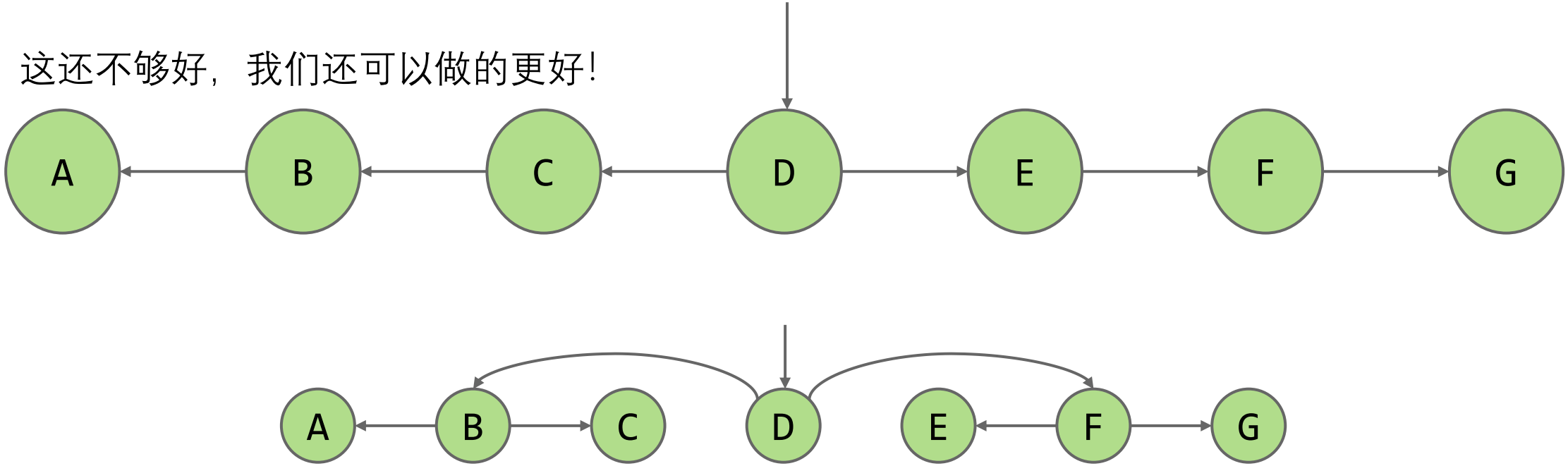
- 利用有序性，我们可以把入口指针指向中间的元素

这还不够好，我们还可以做的更好!



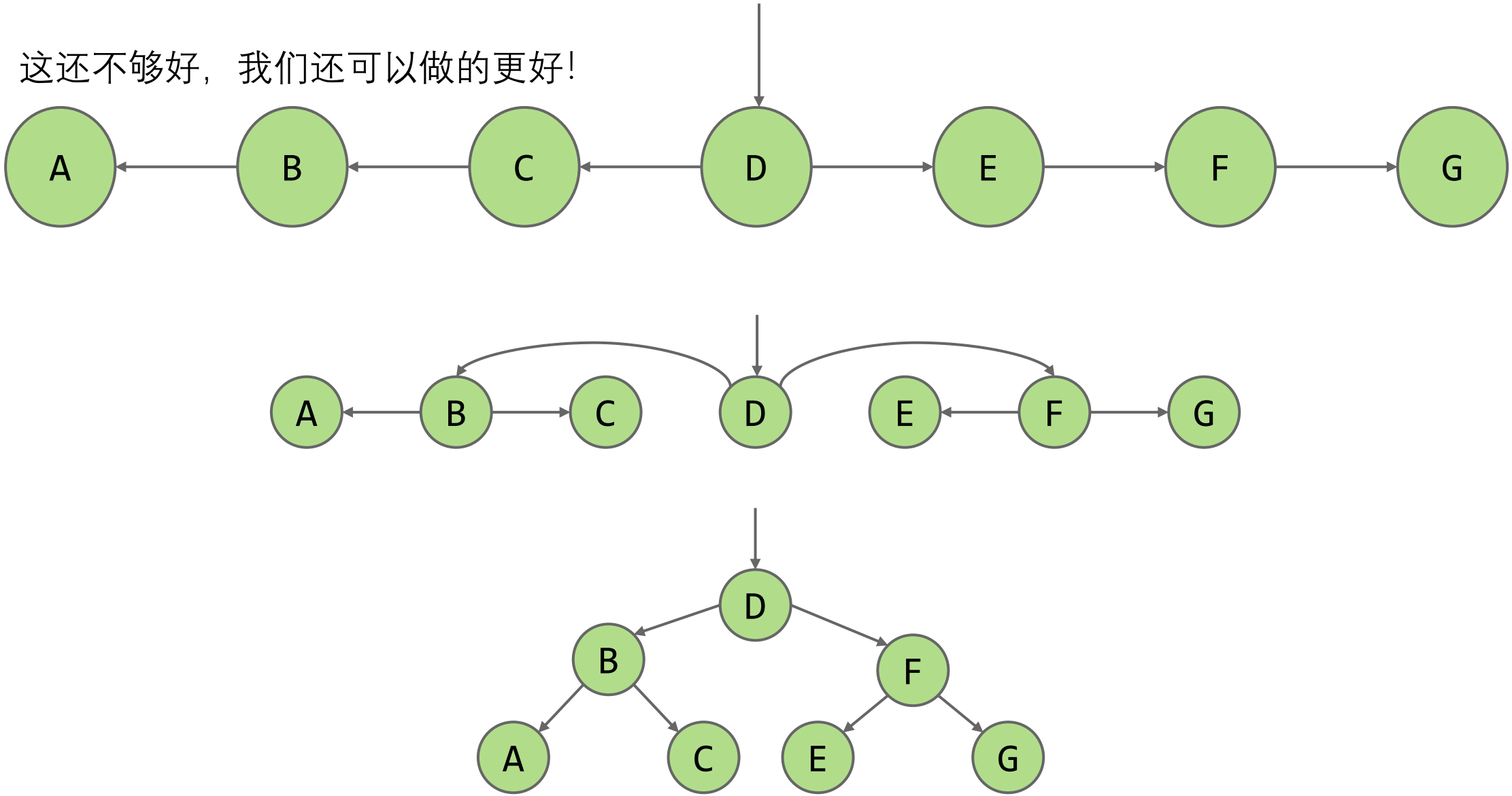
- 利用有序性，我们可以把入口指针指向中间的元素

这还不够好，我们还可以做的更好!



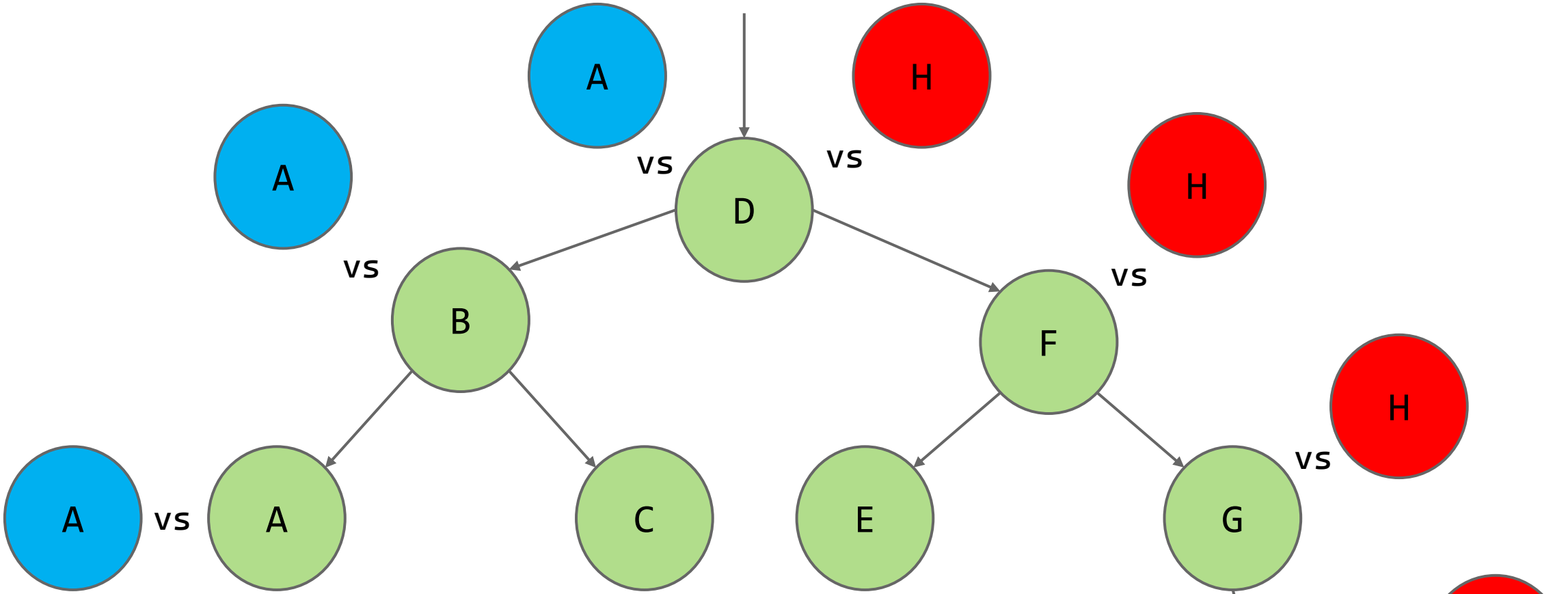
- 利用有序性，我们可以把入口指针指向中间的元素

这还不够好，我们还可以做的更好!



contains('A')?

contains('H')?



return true;

return false;

二叉搜索树：定义

Lecture 1

二叉搜索树
- 导入
- 定义
- `contains()`
- `insert()`
- `delete()`

二叉搜索树的应用

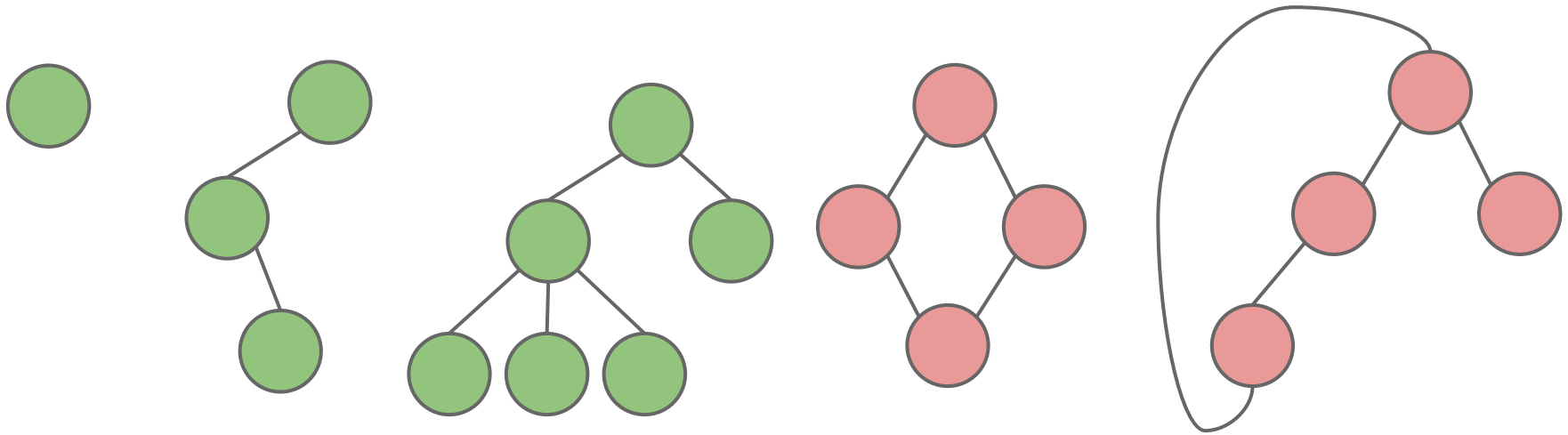
树:

A tree consists of:
- A set of nodes. -> 有限个结点的集合
- A set of edges that connect those nodes. -> 有有限个边相互连接
  - Constraint: There is exactly one path between any two nodes.

树:

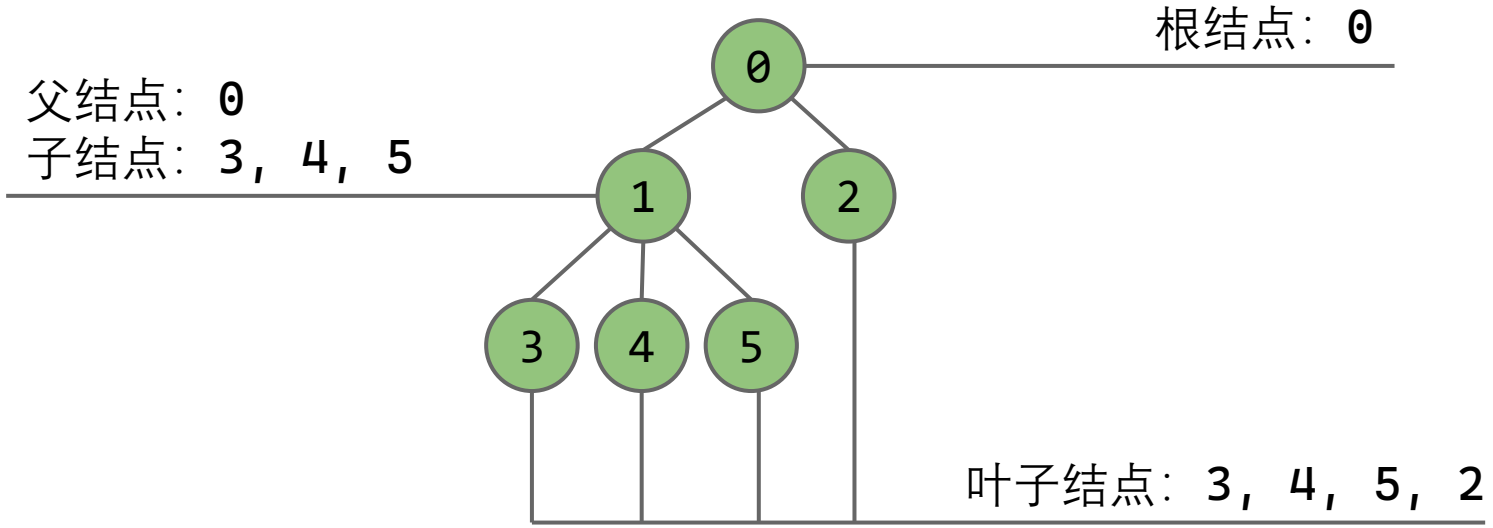
A tree consists of:

- A set of nodes. -> 有限个结点的集合
- A set of edges that connect those nodes. -> 有有限个边相互连接
  - Constraint: There is exactly one path between any two nodes.



树 & 二叉树

- Every node N except the root has exactly one parent. -> 没有父结点的结点就是根
- the root is usually depicted at the top of the tree. -> 根结点常常表示在最上方
- A node with no child is called a leaf. -> 没有子结点的结点称为叶结点



树 & 二叉树

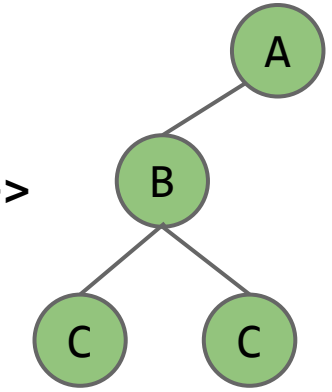
- Every node N except the root has exactly one parent. -> 没有父结点的结点就是根
- the root is usually depicted at the top of the tree. -> 根结点常常表示在最上方
- A node with no child is called a leaf. -> 没有子结点的结点称为叶结点

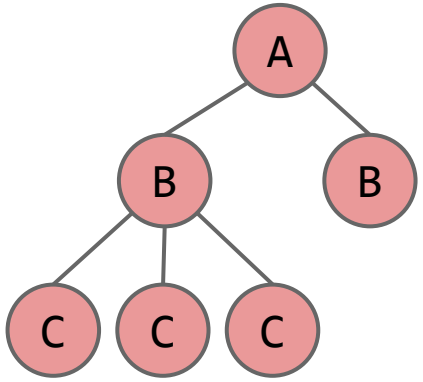
In a binary tree, every node has either 0, 1, or 2 children (subtrees).

Simple Java Code:

```
class Tree {
  int key;
  Tree leftPtr;
  Tree rightPtr;
}
```

Binary! ->





<- Not binary!

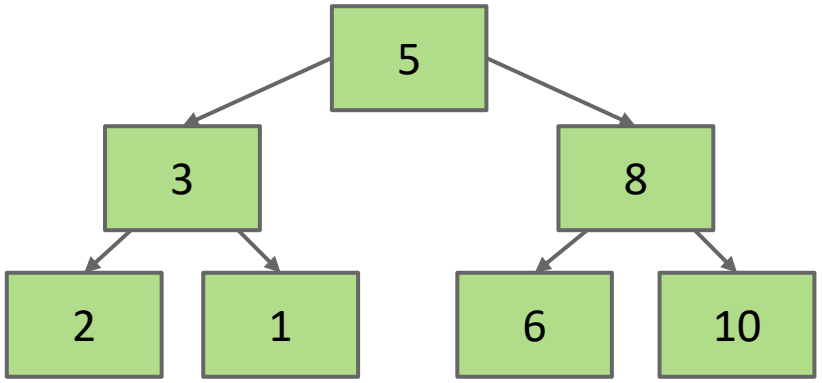
二叉搜索树

二叉搜索树就是在二叉树的定义基础上再增加一些条件限制：

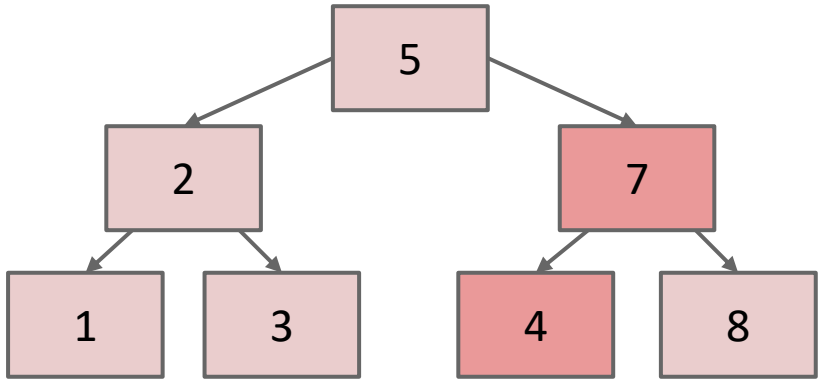
For every node X in the tree: -> 对于树中所有结点：

- Every key in the left subtree is less than X's key. -> 一个结点左子树的所有结点的值小于这个结点的值

- Every key in the right subtree is greater than X's key. -> 一个结点右子树的所有结点的值大于这个结点的值



Binary Search Tree



Binary Tree,
but not a Binary Search Tree

二叉搜索树

二叉搜索树就是在二叉树的定义基础上再增加一些条件限制：

For every node X in the tree: -> 对于树中所有结点：

- Every key in the left subtree is less than X's key. -> 一个结点左子树的所有结点的值小于这个结点的值

- Every key in the right subtree is greater than X's key. -> 一个结点右子树的所有结点的值大于这个结点的值

推论：二叉树不能有相同的值，结点的值之间没有相等的情况。

二叉搜索树: contains()

Lecture 1

二叉搜索树
- 导入
- 定义
- contains()
- insert()
- delete()

二叉搜索树的应用

二叉搜索树: insert()

Lecture 1

二叉搜索树
- 导入
- 定义
- contains()
- insert()
- delete()

二叉搜索树的应用

二叉搜索树: delete()

Lecture 1

二叉搜索树
- 导入
- 定义
- contains()
- insert()
- delete()

二叉搜索树的应用

二叉搜索树：应用

Lecture 1

二叉搜索树
- 导入
- 定义
- contains()
- insert()
- delete()

二叉搜索树的应用