对于大量的数据输入，链表的线性访问时间很长，可以考虑使用树来进行访问，其操作的平均运行时间为



如上图所示，结点A是根，结点F有一个父亲A，并有儿子K，L，M。

没有儿子的结点称为叶结点，B，C，H，I，P，Q，K，L和M都是叶结点。具有相同父亲结点的结点称为兄弟。

从结点到的路径定义为结点，，的一个序列，使得对于，结点是的父亲。路径的长为路径上的边的条数，即。一颗树，从根到每个结点只存在一条路径，两个不同的父结点不会有相同的子结点。

如果存在从到的一条路径，则为的祖先，为的后裔。如果，则为的真祖先，而为的真后裔。

任意结点，的深度为从根到的唯一路径的长度。根的深度为0.的高度为是从到一片树叶的最长路径的长度。所有树叶的高度均为0，因其没有子结点。一颗树的高为其根的高度。上图所示的树的高度为3，E的深度为1，高度为2. F的深度为1，高度也为1.

二叉树：每个结点不能有多于两个儿子的树

二叉查找树：对于结点，它的左子树的所有项的值小于中项的值，而它的右子树所有项的值大于中的项的值。

二叉查找树的平均深度为

例：程序binary\_tree\_test1

二叉树的遍历：

前序遍历：根->左子树->右子树

中序遍历：左子树->根->右子树

后序遍历：左子树->右子树->根

参考<http://jingyan.baidu.com/article/a24b33cd6c33a319fe002b3f.html>



前序遍历：A->B->C->D->E->F-G

说明：A为根结点，B是A的左结点，C是B的左结点，D是B的右结点，E是A的右结点，F是E的右结点（E相当于F的根结点），G是F的左结点（F相当于G的根结点）。

中序遍历：C->B->D->A->E->G->F

说明：C是B的左结点，B相当于C,D的根结点，D是B的右结点，B是A的左结点，E是A的右结点，G是F的左结点（G在F之前），F是E的右结点。

后序遍历：C->D->B->G->F->E->A

说明：C是B的左结点，D是B的右结点，B是A的左结点，由于A是整颗树根结点，所有先遍历其右子树。G是F的左结点，F是E的右结点，E是A的右结点。

二叉树遍历时，当遇到某一结点时，将其视作根结点，然后根据遍历规则确定顺序。例如，在上述后序遍历时，遍历根结点A的右子树，遇到结点E，E有右结点F，这样遇到F时，F又有左结点G，根据后序遍历的规则：左子树->右子树->根，所以A的右子树后序遍历为：G->F->E

参考：<http://zhidao.baidu.com/link?url=k6waq15OH_MOR5IMyE1ymwM0B4ZehCixyklJZIg14_FxtPw8Xu9of1NPLVdVR3vCopkj9IPa0ramTv7N_O8Oaa>

参考：程序binary\_search\_tree\_test1

参考：程序binary\_tree\_test1