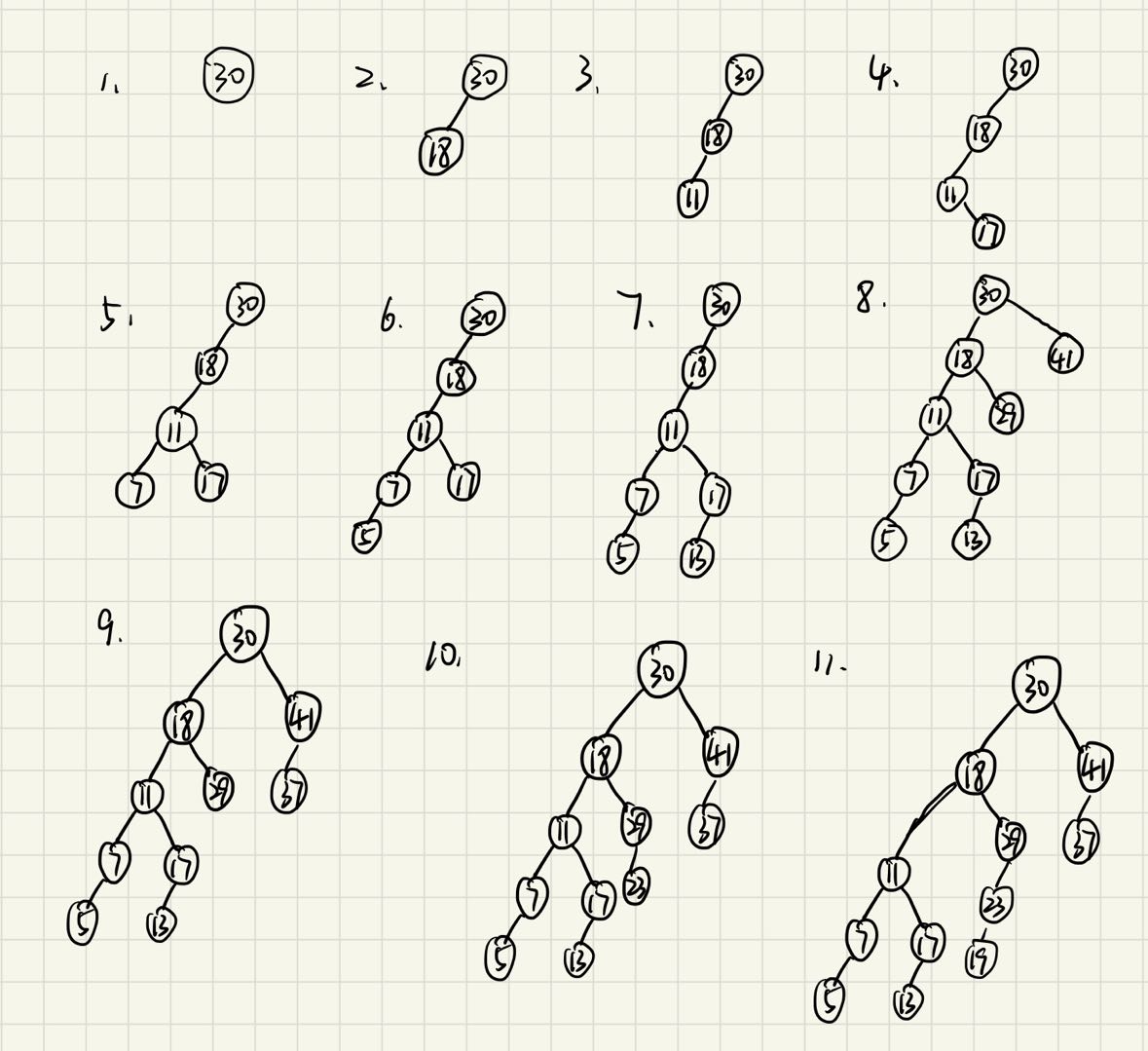
**第八章 查询**

1. 有一个有序表为{1,3,9,12,32,41,45,62,75,77,82,95,100}，当二分查找值为82的结点时，（ 4 ）次比较后查找成功。
2. 设哈希表长m=11，哈希函数H（key）=key%11。表中已有4个结点：addr（15）=4， addr（38）=5，addr（61）=6，addr（84）=7，其余地址为空，如果线性探测再散列处理冲突，关键字为49的结点地址是 8 。
3. 从空的二叉排序树开始依次插入30，18,11,17,7,5,13,41,29,37,23,19。画出该二叉排序树。



1. 读程序，写出程序的功能

int func3 (int r[ ], int n, int k)

{ int i,low=0, high=n-1, mid, find=0;

while (low<=high &&find==0)

{ mid=(low+high)/2;

if (k<r[mid]) high=mid-1;

else if (k>r[mid]) low=mid+1;

else ｛i＝mid;

find=1;｝

}

if (!find) return -1;

return i;

}

**二分查找**

5、读程序填空

下面的函数在给定的查找树（二叉排序树）中插入一个新结点。在插入之前，首先对查找树进行查找，如树中不包含插入的新结点，则进行插入，否则不插入。

typedef struct tnode {

elemtype data;

struct tnode \*lchild.\*rchlid;

} tnode, \*bitreptr ;

bitreptr insert(bitreptr root , char ch)

{ bitreptr p, q, r;

p=null ; q=root;

while (q && q→data!=ch)

{p=q;

if (ch<q→data) (A) ;

else (B) ;

}

if (!q) {q = (bitreptr)malloc(sizeof(tnode));

q→data=ch; q→lchild=q→rchild=null;

if (!p) (C) ;

else if (ch<p→data) (D) ;

else (E) ;

}

return root;

}

A q = p->lchild；

B q = p->rchild

C root = q

D p->lchild = q

E p->rchild = q

6.写出创建二叉排序树的算法

typedef int KeyType;

typedef char ElemType;

typedef struct **BSTNode**

{   KeyType key;

    ElemType data;

    struct **BSTNode** \*lchild,\*rchild;

}**BSTNode**,\***BSTree**;

void **CreatBST**(**BSTree** &T,**KeyType** a[],int n)

{

    T = NULL;

    for(int i = 0;i < n;i++){

**InsertBST**(T,a[i]);

    }

}