1. C
2. A
3. 静态查找只对表进行操作，并不会动态添加元素；动态查找过程中能够向查找表中插入元素，或者从查找表中删除某个指定的元素。
4. void justice(Type *arr*[],int *n*)

{

    int left=0,right=*n*-1;

    while(left<right)

    {

        while(*arr*[left]<0&&left<right)

            ++left;

        while(*arr*[right]>0&&left<right)

            --right;

        if(left<right)

        {

           Type x=*arr*[left];

*arr*[left++]=*arr*[right];

*arr*[right++]=x;

        }

    }

}

时间复杂度：O(n)

1. B
2. 4
3. (1) typedef int KeyType;

typedef struct BiTNode

{

KeyType key;

struct BiTNode\* lChild, \* rChild;

}BiTNode, \* BiTree;

//查找

BiTree SearchT(BiTree T, KeyType key, BiTree& p)

{

if (!T)

return NULL;

if (key == T->key)

{

return T;

}

p = T;

if (key < T->key)

return SearchT(T->lChild, key, p);

else

return SearchT(T->rChild, key, p);

}

//插入

bool InsertT(BiTree& T, KeyType key)

{

BiTree p = T;

if (SearchT(T, key, p))

return false;//如果已有该元素，返回false

//否则p指向末端

BiTree s = new BiTNode;

s->key = key;

s->lChild = s->rChild = NULL;

if (!p)//T是空树

{

T = s;

}

else if (key < p->key)

{

p->lChild = s;

}

else

{

p->rChild = s;

}

return true;

}

bool CreateBitree(BiTree &T, KeyType keys[], int n)

{

for(int i = 0;i < n; i++)

{

if(!InsertT(T,keys[i]))

return false;

}

return true;

}

(2)void inOrderRecur(BiTree root)//中序遍历从小到大输出

{

if (root == NULL)

return;

inOrderRecur(root->lChild);

cout << root->key << " ";

inOrderRecur(root->rChild);

}

1. C



1. C
2. A
3. Asl=1/2\*(1+1/(1-a))<=3

a<=0.8

100/L<=0.8

L>=125

L=127