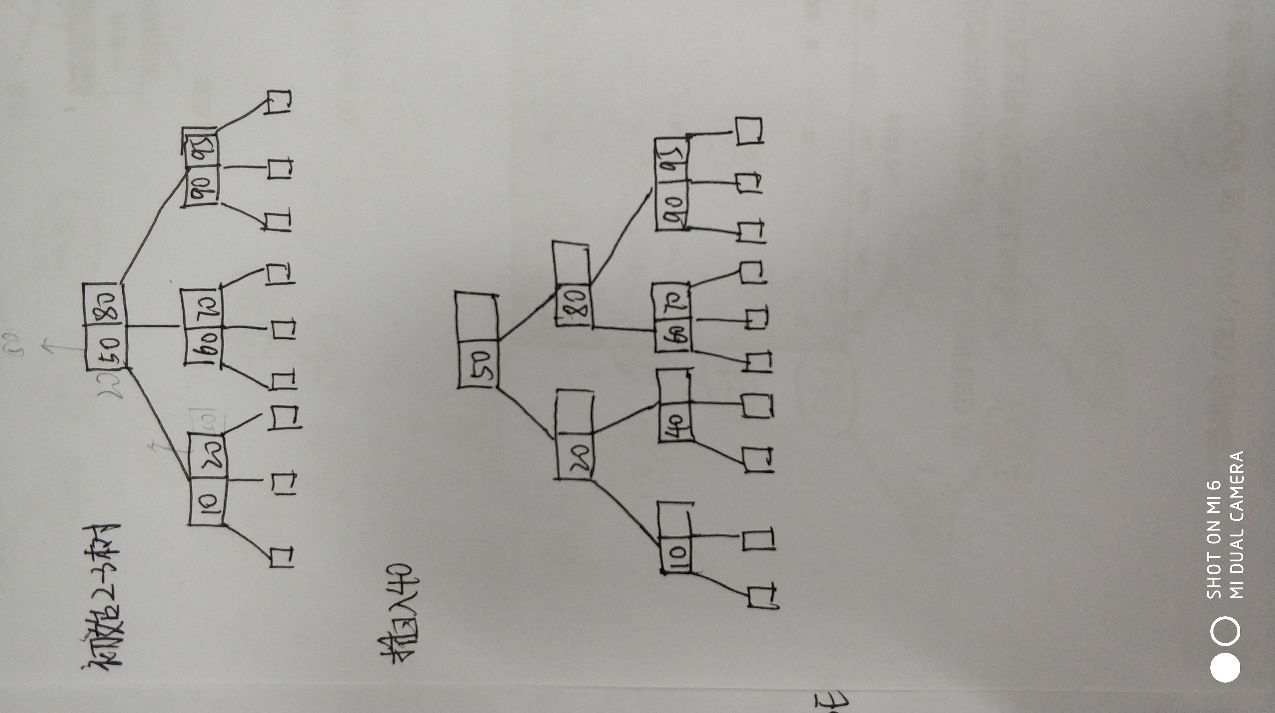
578-5



609-3

假设m路查找树是平衡的B树

时间复杂度O（nlog（n））

void Insert(BTree &q,int i,int x,BTree ap){//关键字x和新结点指针ap分别插到q->key[i]和 //q->ptr[i]

int j,n=q->keynum;

for(j=n;j>=i;j--){

q->key[j+1]=q->key[j];

q->ptr[j+1]=q->ptr[j];

}

q->key[i]=x;q->ptr[i]=ap;

if(ap!=NULL) ap->parent=q;

q->keynum++;

}//时间复杂度O(n)

void InsertBTree(BTree &t,int k,BTree q,int i){

//在B树中q结点的key[i-1]和key[i]之间插入关键字k

//若插入后结点关键字个数等于b树的阶，则沿着双亲指针链进行结点分裂，使得t仍是m阶B树

int x,s,finished = 0,needNewRoot = 0;

BTree ap;

if(NULL==q) newRoot(t,NULL,k,NULL);

else{

x=k;ap=NULL;

while(0==needNewRoot&&0==finished){//循环时间复杂度O(log(n))

Insert(q,i,x,ap);//x和ap分别插到q->key[i]和q->ptr[i]

if(q->keynum<m) finished=1;//插入完成

else{

s = (m+1)/2;split(q,s,ap);x=q->key[s];

if(q->parent!=NULL){

q=q->parent;i=Search(q,x);//在双亲结点中查找x的插入位置

}

else needNewRoot=1;

}

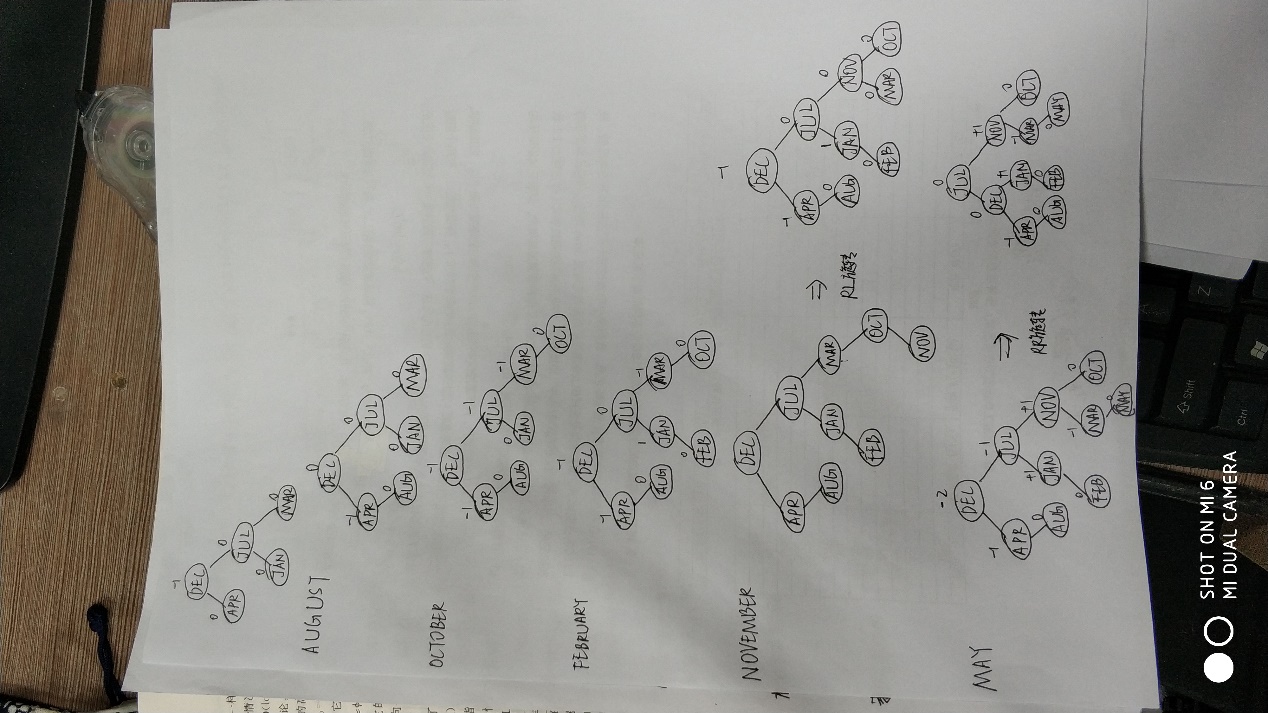
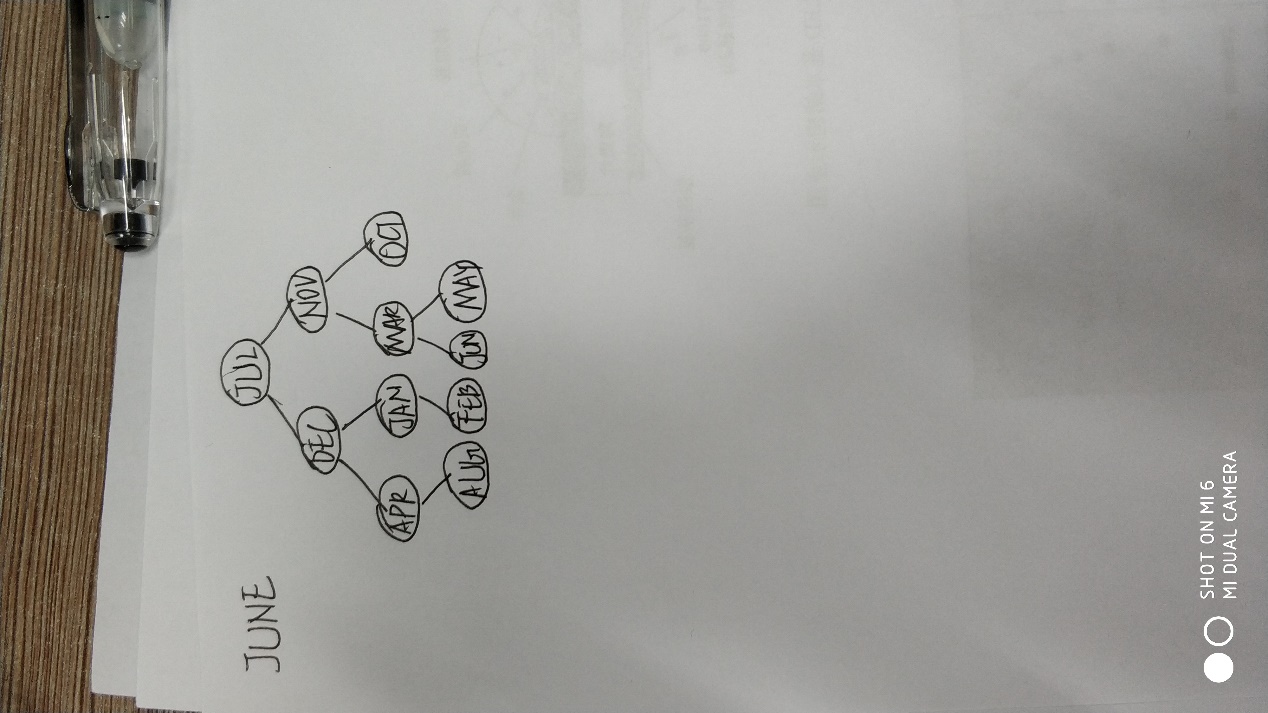
}

if(1==needNewRoot)//t是空树或者根结点已经分裂成为q和ap结点

newRoot(t,q,x,ap);

}

}

623-2

