一、为最近对问题的一维版本设计一个直接基于分治技术的算法,并确定它的时间复杂度。假设输入的点是以升序保存在数组A中。（最近点对问题定义：已知上m个点的集合，找出对接近的一对点。）

1、基本思想

进行递归分治运算，分为2个数字或者1个数字基本单位，如果是2个数字，就运算出两点距离，然后和最小值比较，最后算出全局最小

1. 实验过程

在分的过程中，分为（l，m）和（m+1，r），还得计算（m，m+1），最后算出全局最小。

1. 实验结果

时间复杂度o（n），空间复杂度o（1）

IMG_256

代码：

#include<stdio.h>

#include<math.h>

struct node

{

int x;

int y;

};

struct node devide(int num[],int l,int r)

{

if(l==r){

struct node max = {-99999,999999};

return max;

}else if(r-l==1){

struct node min = {num[l],num[r]};

return min;

}else{

int m = (l+r)/2;

struct node leftNode = devide(num,l,m);

struct node rightNode = devide(num,m+1,r);

struct node mNode = devide(num,m,m+1);

struct node minNode = leftNode.y-leftNode.x < rightNode.y-rightNode.x ? leftNode:rightNode;

minNode = minNode.y -minNode.x < mNode.y-mNode.x ? minNode:mNode;

return minNode;

}

}

int main(){

int num[7] = {1,4,5,7,8,13,16};

struct node node1 = devide(num,0,6);

printf("%d %d\n",node1.x,node1.y);

}

二、设计一个分治算法来计算二叉树的层数.(空树返回0,单顶点树返回1),并确定它的时间复杂度.

1、基本思想

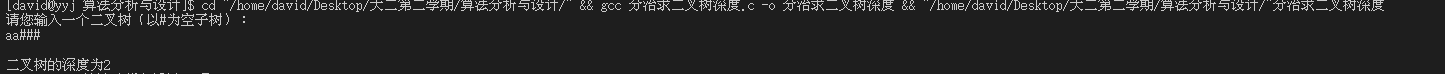
普通的递归遍历，看看这个结节的高度是左子树的最大值和右子树的最大值。

2、实验过程

写出递归求深度，创建二叉树测试，测试成功

1. 实验结果

时间复杂度为0（n）



代码：

#include <stdio.h>

#include<malloc.h>

#include <malloc.h>

typedef struct BiTNode

{

char data; /\*结点的数据域\*/

struct BiTNode \*lchild , \*rchild; /\*指向左孩子和右孩子\*/

} BiTNode , \*BiTree;

/\*创建一棵二叉树\*/

void CreatBiTree(BiTree \*T)

{

char c;

scanf("%c",&c);

if(c == '#')

\*T = NULL;

else

{

\*T = (BiTNode \* )malloc(sizeof(BiTNode)); /\*创建根结点\*/

(\*T)->data = c; /\*向根结点中输入数据\*/

CreatBiTree(&((\*T)->lchild)); /\*递归地创建左子树\*/

CreatBiTree(&((\*T)->rchild)); /\*递归地创建右子树\*/

}

}

//计算二叉树的深度

int getBitreeDepth(BiTree T)

{

int leftHeight, rightHeight, maxHeight;//左子树，右子树，最大深度

if (T != NULL) //如果为空树

{

leftHeight = getBitreeDepth(T->lchild);//左子树深度

rightHeight = getBitreeDepth(T->rchild);//右子树深度

maxHeight = leftHeight>rightHeight?leftHeight:rightHeight;//最大深度

return maxHeight+1;//二叉树深度=最大深度+1

}

else

{

return 0;

}

}

void main()

{

BiTree T = NULL; /\*最开始T指向空\*/

printf("请您输入一个二叉树（以#为空子树）：\n");

CreatBiTree(&T); /\*创建二叉树\*/

printf("\n二叉树的深度为%d\n",getBitreeDepth(T));

}