**《数据结构》实验报告**

**姓名： 杨孟衡 学号： 8002118240**

**班级： 软件工程1809班 专业： 软件工程**

**报告日期： 2019 年 11 月 15 日**

**实验五 递归**

**一、问题描述与分析**

1. 已知存储结构和基本操作，完成实验1：编写递归算法int max(int a[],int left, int right)，求数组a[left..right]中的最大数；2：编写一个递归算法函数void partion(int a[], int left, int right)中所有的奇数调整到数组的左边，所有偶数调整到数组的右边；3、编程实现二分查找法查找值为key的元素所在的位置；4、基于带头结点的单链表结构，返回表中最大数的节点地址：

1. 编写递归算法int max(int a[],int left, int right)，求数组a[left..right]中的最大数。(lab5\_01.c)
2. 编写一个递归算法函数void partion(int a[], int left, int right)中所有的奇数调整到数组的左边，所有偶数调整到数组的右边。(lab5\_02.c)
3. 请编写递归函数void bubbleSort(int a[], int n),对长度为n的数组采用冒泡法进行升序排序；并编写递归函数int binSearch(int a[], int left, int right,int key)，采用二分查找法在数组a[left..right]中查找值为key的元素所在的位置，若查找失败函数返回-1。(lab5\_03.c)
4. 已知带头结点的单链表结构定义如实验三，假设链表中所有结点值均不相同，请编写一个递归函数linklist max(linklist head),返回表中最大数所在的结点地址，若链表为空，返回NULL。(lab5\_04.c)

2.分析

(1) 递归结束条件为左下标等于右下标，返回数组元素值，若左侧元素大于右侧元素值，则右侧下标减一再递归，若左侧元素小于右侧元素值，则左侧下标加一再递归；

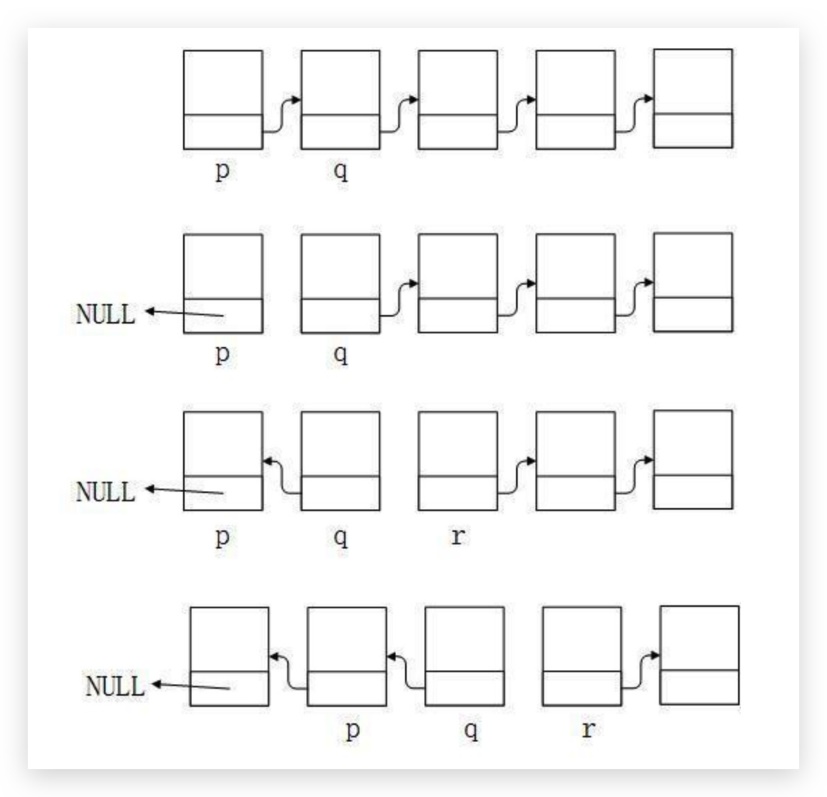
(2) 首先第一步判断是否左侧下标小于右侧下标，设置两个循环直到找到左侧第一个奇数，右侧第一个偶数，接着判断左侧下标是否大于右侧下标，若大于，交换元素值，左右两侧下标值做相应变化，进行下一趟递归；

(3) 若左侧下标大于右侧，返回空值，计算mid值，如果正好中间元素值等于要找的值，返回这个元素值，如果小于中间元素值，mid下标减一在进行递归，大于大于和中间元素值，mid下标加一递归；

(4) 若到达尾结点返回空值，否则若是倒数第三个结点则返回倒数第二个节点，或者常规位置，用p指针记录递归返回的结点，然后返回后续结点值和当前p结点的值比较，返回较大的那个结点；

**二、数据结构与算法设计**

链表图示如下：



**三、算法复杂度分析**

递归，算法复杂度为O(n);

**四、测试计划**

1.编写目的

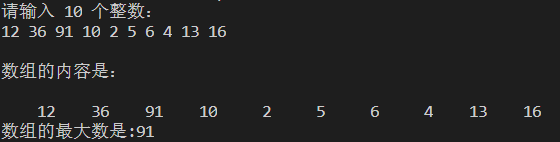
加深对数据结构链表的存储结构，递归程序的理解。

2.开发及运行环境Dev c++

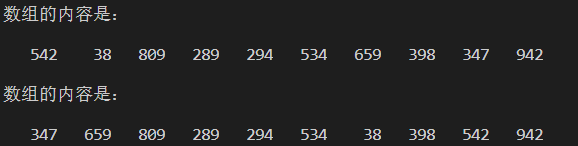
Visual Stdio Code 1.40

3.小项测试截图

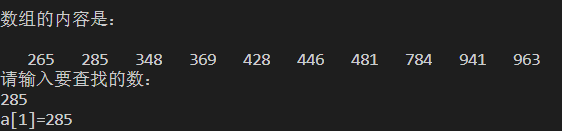
（1）基于递归求数组最大数



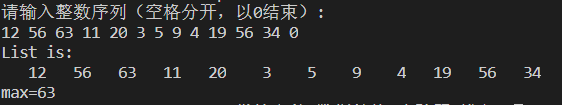
（2）基于递归调整数组



（3）基于递归实现二分查找法



（4）基于带头结点的单链表实现递归



**五、源程序(仅贴算法函数部分)**

**<lab5\_01>:**

 int temp;

    if(left == right)

    {

      return a[left];

    }

    if(a[left] > a[right])

    {

      temp = max(a, left,right - 1);

    }

    else temp = max(a, left + 1, right);

**<lab5\_02>:**

int x;

   if (left<right)

   {

       while (left<right && a[left]%2==1)

            left++;

       while (left<right && a[right]%2==0)

            right--;

       if (left<right)

       {

            x=a[left];

            a[left]=a[right];

            a[right]=x;

            partion(a,left+1,right-1);

       }

   }

**<lab5\_03>:**

int mid;

  if(left > right)

  {

    return NULL;

  }

  mid = (left + right) / 2;

  if(a[mid] == key)

  {

    return mid;

  }

  if(key < a[mid])

  {

    binSearch(a, left, mid - 1, key);

  }

  else binSearch(a, mid + 1, right, key);

**<lab5\_04>:**

linklist maxe(linklist head)

{

    linklist p;

    if (head->next==NULL)

        return NULL;

    else

        if (head->next->next==NULL)

            return head->next;

    else

        {

            p=maxe(head->next);

            return head->next->data>p->data? head->next:p;

        }

}