1-10题

2019年7月7日 星期日　21:04

- 依次递增的二维数组中，判断是否含有给定整数
  - 从数组的左下角当做初始点
  - 比查找数小则往右移，大则往上移
  - 初始i=row-1,j=0;
- 讲字符串的每个空格替换为“%20“
  - 先遍历一遍将空格的个数输出来
  - 再从后向前遍历把要求字符和原字符填进去
- 从尾到头打印链表数值
  - 遍历链表将值存进一个stack中
  - 将stack中的元素全部pop出到array中
- 给出二叉树的前序遍历和后序遍历的数组，重建出二叉树
  - 递归
  - 因为前序和中序里不含有重复的结果，所以遍历中序，找出和前序的第一个根相对应的节点，并以此节点为界，递归左子树和右子树
  - 注意递归时，对前序中序数组的遍历条件（起点和终点）
- 用两个栈实现一个队列
  - 利用两个栈 stack1，stack2
  - push操作就push到stack1就好
  - 然后当pop的时候就把stack1中所有元素折腾到stack2，这样stack2中元素顺序就跟队列一样了
  - 然后pop出来一个stack2的top节点
- 两个队列实现一个栈
  - 压入得时候压入当前的非空队列，若两个都为空则压入1
  - pop的时候将非空队列的所有节点弹出，压入2
  - 若当前节点弹出后1就为空了，则当前节点为最后一个节点，输出且不压入2了
  - 如此每pop一个节点都反复两边折腾
- 将数组的最开始若干元素搬到末尾称为数组的旋转

  输入一个非递减排序的数组的一个旋转，输出旋转数组的最小元素
  - 利用二分查找

若mid大于右	则最小的在右边	left=mid+1
若mid小于右	则最小的在左边	right=mid
若相等	继续缩小范围	right--

- 输出斐波那契数列的第n项
  - 斐波那契数列：第n项等于前面两项的和
  - 就用一个简单的for循环计算出第n项即可
- 青蛙可以一次跳1或2节台阶，求跳上n级台阶的跳法

    - 实际上就是一个斐波那契数列
    - 递归解决
    - 跳上第n级台阶的解法数目等于n-1级数目和n-2级数目之和
- 变态跳台阶：可以一次跳1....n级台阶，问跳到n级台阶的跳法
    - 通过数学归纳得知f(n)=2*f(n-1)
    - 所以利用等比数列第n项公式=2的n-1次幂
    - 也可以递归
- 矩形覆盖：n个2*1的小矩形去覆盖一个2*n的大矩形的分法
    - 第n项等于n-1项和n-2项之和
    - 递归

11-20题

2019年7月8日 星期一　　23:47

- 输入一个整数，输出二进制表示中1的个数，负数用补码表示
  - 补码：符号位不变，其余位取反，然后最后+1
  - 利用一个值为1的flag，每次都和数值进行与操作，对结果不为0的情况计数
  - 每次循环都让flag左移一位
  - 结束条件：flag超int范围（即flag！=0）
  - 输出计数结果
- 求一个double型数的int次方
  - 先处理指数和底数的特殊情况
  - 利用二分快速幂的方法将指数拆分
  - ```
    long long ret = 1;
    long long tmp = x;
    while (y > 0)
    {
            if (y & 1) ret *= tmp;
            y >>= 1;
            tmp *= tmp;
    }
    return ret;
    ```
  - 最后根据指数正负分情况输出
- 调整数组顺序使得奇数都在前偶数都在后，且相对位置不变
  - 双层循环
  - 第一层从头开始遍历数组到尾，i
    - 当当前数i为偶数的时候
      - 第二层循环开始，从i到末尾
        - 如果当前位j为奇数
        - 将i-j之间的所有偶数都向后移一位
        - 并把该奇数放在第i位
        - i++
- 输出链表中的倒数第k个节点
  - 定义两个指针一个指向表头，一个指向第k个节点
  - 然后两个指针同时向后移动，直到后面的那个指针指到了末尾节点
  - 输出前面的那个节点
- 反转链表
  - 借用两个临时节点pPre和pNext
  - ```
    while(pHead!=null){
        pNext=pHead.next;
        pHead.next=pPre;
        pPre=pHead;
        pHead=pNext;
    }
    return pPre;
    ```

- 输入两个单调递增的链表，输出两个链表合成之后的新链表，且新链表单调不减
    - 递归
    - 假设pHead1和pHead2各有两个节点，自己用笔模拟一下就能得出递归代码
    - 输出就用PHead1或者2就行
- 判断二叉树B是不是A的子结构
    - 递归
    - 主函数和help用函数都是递归
    - 主函数内
        - 如果当前节点相等，进入help函数
        - 如果结果为false，递归主函数进入左子树，如果依然为false，进入右子树
    - help函数内
        - 如果NULL或者两者不等，返回false
        - 否则return 左子树递归&&右子树递归
- 二叉树的镜像
    - 递归
    - 就是简单的将左孩子和右孩子颠倒
    - 再递归进当前节点的左孩子和右孩子
- 顺时针沿着边儿打印矩阵
    - 定义left，right，top，bottom四个int值
    - while（左<=右，上<=下）时
        - 输出一圈的数（注意最后上=下，左=右的特殊情况）
        - 修改上下左右四个值进入里层的下一圈
    - 输出结果的vector
- 实现一个O(1)的用于栈的输出最小值的min函数
    - 两个栈
    - 1栈用于保存当前2栈内的最小值
    - min函数
        - 如果两个栈top相等
            - 两个栈都pop
        - 否则只pop2

21-30题

2019年7月10日 星期三 21:52

- 输入数组1表示栈的压入顺序，判断输入数组2可不可能为该栈的弹出顺序
  - 声明一个栈s用来模拟，一个int型j用来表示弹出顺序的当前index值
  - 遍历序列1
    - push进辅助栈
    - 若辅助栈的栈顶等于序列2的index项
      - 辅助栈pop
      - index++
  - 如果最后辅助栈为空则true，否则false
- 从上往下，同层从左往右打印二叉树
  - 辅助队列p，push根节点进去
  - while p不为空
    - 将p的头结点加入res
    - 将p头结点的左右儿子push进p
    - 将p的头结点pop
- 判断一个整数数组是不是某二叉搜索树的后序遍历
  - 递归
  - 如果为空，返回false
  - 声明左树数组，右树数组
  - 计算当前右树第一个节点index
    - 若右树有小于根的，直接返回false
  - 将左子树节点push进左树数组
  - 将右子树节点push进右树数组
  - 递归左子树
  - 递归右子树
  - 如果都通过了返回true
- 打印出二叉树中和为某一值得所有路径（从根节点到叶节点才算一个路径）
  - 递归
  - helper函数（根，剩余值，二维数组，tmp数组）
    - 更新剩余值和tmp数组
    - 如果当前已经是根节点了且剩余值为0
      - tmp数组进入res
    - 满足递归进左子树条件就递归左子树
    - 同样就进入右子树
    - tmp里pop掉末尾节点
- 复制一个复杂的链表（每个节点有一个值，一个指向下一个节点的指针，和一个随机指向节点的指针）
  - 设置一个currentNode表示当前节点

    - 先把整个链表的指针复制一遍，新节点就在原节点之后
    - 将新节点的random指针指向对应的新节点
    - 设置一个cloneHead表示当前的新节点的头
    - 删除原来的节点
    - 输出cloneHead
- 输入二叉搜索树，转换成一个排序的双向链表，并且不能添加任何新节点，只能调整指针
    - 全局变量 设一个节点为新的头，一个为current节点
    - 递归
    - 中序遍历
        - 空-》直接return
        - 进左子树
        - current节点为空则将Head和current都设置为当前节点
        - current不为空
            - 将current和当前节点相连（两条）
            - 将current设置为当前节点
        - 进入右子树
- 输入字符串，打印出所有字符所能组合出来的顺序（可能有重复字符，按字典顺序输出）
    - 递归
    - 重载Permutation，参数为给定字符串，结果数组，和当前起始位置
        - for（从当前位置到最后）
            - 第i个和起始位置的调换位置
            - 进入递归，起始位置+1
            - 换回来
        - 如果当前起始位置已经是最后一位
            - 如果res里没有给定str，则将str添加进res
    - 将res，用sort函数排序，输出
- 数组中出现次数超过一半的数字
    - 先sort排好顺序
    - 计算和正中间位置数值相等的数字出现了多少次
    - 如果出现次数大于长度的一半则输出，否则输出0
- 输出最小的K个数
    - 利用priority_queue（可自定义排序规则，默认最小值在最前）
    - 将数组前K个数输入Q，然后如果后面的数比当前队列top小就进入，并pop，否则就忽略
    - 最后将Qpush进res数组
- 输出连续子数组的和的最大值（元素里有负数）
    - 设置两个int，max和res
    - for整个数组

- 如果max大于0（也就是说之前的那些数是有正向的贡献的）
  - 则max就加上当前数
- 如果小于0
  - max重置为当前数
- 如果res小于max
  - res=max

31-40题

2019年7月14日 星期日　　20:44

- 求出给定整数n，从1-n中1出现的次数
  - 通过对每一位1出现次数计算方法的归纳，可以得出
count(i) = (n/(i*10))+	if(k>i*2-1)	else if(k<i)	else
	i	0	k-i+1
      - 其中n为给定整数，i为当前计算的位数，count(i)表示当前位可能出现的1的个数，k表示n对当前位数(i*10)的取余结果
  - 最后用一个for循环讲每一位的结果一次相加，输出
- 讲给定数组的所有数字拼接起来形成一个最小的数
  - 利用sort函数可以自定义规则的用法
  - static bool stringSort（）函数
    - 自定义sort规则函数
    - 判断ab和ba大小的函数，ab小则输出true
  - 将重新排序后的数组数字输入到字符串中，输出字符串
- 输出第n个丑数（硬背吧）
  - 丑数：只包含质因子2，3，5的数
  - ```
    class Solution {
    public://别人的代码就是精简，惭愧啊，继续学习。
        int GetUglyNumber_Solution(int index) {
            if (index < 7)return index;
            vector<int> res(index);
            res[0] = 1;
            int t2 = 0, t3 = 0, t5 = 0, i;
            for (i = 1; i < index; ++i)
            {
                res[i] = min(res[t2] * 2, min(res[t3] * 3, res[t5] * 5));
                if (res[i] == res[t2] * 2)t2++;
                if (res[i] == res[t3] * 3)t3++;
                if (res[i] == res[t5] * 5)t5++;
            }
            return res[index - 1];
        }
    };
    ```
- 找到字符串中第1个只出现一次的字符
  - 利用map容器（可以存值和索引）的特性
  - map<char,int>
  - 将各个字符串出现的次数存入int索引
  - 再根据原来字符串中字符出现顺序遍历，输出第一个
  - 都没有返回-1
- 数组中的逆序对（太复杂了，放弃吧）
- 求两个链表的第一个公共节点
  - 利用两个栈（后进先出的特性），将两个链表全入栈
    ```

        - 然后利用循环，找出从尾开始第一个不相等的节点，输出该节点的前一个节点
            - while(栈不为空且top相等)
- 统计一个数字在排好序的数组中出现的次数
    - 大方向：二分查找分别查找该数字的左index和右index
    - 两个函数分别求左index和右index
        - 二分查找
- 计算树的深度
    - 递归
    - 先设置好nullptr情况
    - 然后用？：判断句
        - 若左递归+1大则返回左递归+1
        - 反之返回右递归+1
- 判断二叉树是否为平衡二叉树
    - 平衡二叉树：左右子树高度差不大于1
    - 递归计算当前子树深度
        - nullptr判断
        - int left， right递归进入左子树和右子树
        - left right差绝对值大于1，res=false
        - 返回左右子树里高度大的那个+1
- 一个整数数组中除了某两个数字以外，其他的数字全部出现了2次，求那两个只出现1次的数字
    - 预备知识
        - 相同的数字异或结果为0
    - 思路
        - 将整个数组依次异或，得到一个结果（要求的那两个数的异或）
        - 找到这个结果2进制中第1个为1的位数，记为index
        - 将这个数组分为第index位为1的和不为1的
        - 分别求着两个分组的异或结果，即为所求
    - 代码
        - 2个私有函数
            - 一个用来求第一个为1的位数
            - 一个用来当做主函数的循环条件，求当前数第index位是不是得1
        - 先设置特殊条件，长度为2，则。。。
        - 求整个数组异或结果
        - 求出index（利用&1和>>）
        - for整个数组
            - 如果第index位得1
                - num1异或当前数
            - 不得1
                - num2异或当前数

- 最后Num1，num2即为所求

41-50题

2019年7月16日 星期二 20:00

- 计算所有的和为S的正整数序列
  - 利用一个begin，一个end，初始化为1和2
  - cur不断更新，表示当前begin到end的和
  - 这部分背下来把
    - while如果和大于cur，则从cur中删除begin，之后begin++
    - 如果相等，调用私有函数将结果push进res
    - 否则end++，cur+=end
    - 整个大while条件为begin小于(sum+1)/2且end小于sum
- 输入一个递增排序的数组和一个数字s，在数组中找出两个数使得其和等于s（若有多组输出乘积最小的）
  - 设置i=0，j=size-1
  - while（i<j）
    - 如果和等于sum
      - push进res
      - break
    - 和大于sum
      - j--
    - 和小于sum
      - i++
- 给定一个字符串，输出将其循环左移n位之后的字符串（如abcxzydef左移3位变成xzydefabc）
  - 思路
    - 题意等价于
      - 先倒序0到n-1位
      - 再倒序n到length-1位
      - 最后整个倒序
  - 写一个私有函数用于倒序第几位到第几位
- 输入字符串student a am i输出i am a student
  - string res和tmp初始化""
  - for从头到尾
    - 如果当前位空格
      - res=空格+tmp+res
      - 清空tmp
    - 如果当前位不为空格
      - tmp+=当前位
  - 最后tmp可能还有剩余没加进去的，如果有，再res=tmp+res
- 判断输入的扑克牌是不是顺子（0可以视为任何数）

    - 长度为0返回false
    - sort整个数组
    - 先统计0的个数jokersum
    - 从非0到尾若有相邻两位相等的，返回false
    - max=最后1位-第一个非0位
        - 若max>4则false（总结规律出来的，不好想，最好背下来）
    - 执行到最后就返回true
- 一共有n个小朋友围坐成一圈，指定一个数m，从编号为0的小朋友开始报数，每次喊到第m-1个小朋友就让他出圈，并接着从0开始报数报到m-1，一直到最后剩下一个小朋友，求他的编号
    - 用一个队列模拟这个环
    - while 队列size不为1
        - 对于每次从0数到m-1都把第一个元素pop出去再加到末尾
        - pop一个元素（那个第m-1个）
    - 最后剩的那一个元素即为所求
- 不用任何循环语句和乘除法来计算1+2+3+...+n
    - 利用&&运算符来代替for
        - 当&&前语句为假时不计算
    - &&内嵌套递归
- 不用加减乘除做加法
    - 有点不好想，背吧
    - 利用二进制的异或和与操作

```
首先看十进制是如何做的： 5+7=12，三步走
第一步：相加各位的值，不算进位，得到2。
第二步：计算进位值，得到10. 如果这一步的进位值为0，那么第一步得到的值就是最终结果。

第三步：重复上述两步，只是相加的值变成上述两步的得到的结果2和10，得到12。

同样我们可以用三步走的方式计算二进制值相加： 5-101，7-111 第一步：相加各位的值，不算进位，得到010，二进制每位相加就相当于各位做异或操作，101^111。

第二步：计算进位值，得到1010，相当于各位做与操作得到101，再向左移一位得到1010，(101&111)<<1。

第三步重复上述两步， 各位相加 010^1010=1000，进位值为100=(010&1010)<<1。
      继续重复上述两步：1000^100 = 1100，进位值为0，跳出循环，1100为最终结果。
```

- 把字符串格式的数字转换为整数，数值为0或者字符串不合法时返回0
    - 特殊情况判断
    - 结果res用long long格式的
    - 先判断字符串第一位的正负
    - 然后若有正负从第二位开始，逐位
        - 判断是不是>'0' <'9'
        - 该位得数字计算方法为str[i]-'0'
    - 最后输出时再乘以正负

- 求数组中重复的数字（第一次出现的）
  - temp数组用来存放该数字出现的次数（index为原数组的值）
  - for
    - 如果temp对应位值=0
      - ++
    - 如果不为0
      - 重复的数就是它
      - 返回true
  - 返回false

51-60题

2019年7月17日 星期三 20:24

- 给定一个数组A，构建一个等长的数组B，使得b[i]=A[0]*A[1]*...*A[I-1]*A[I+1]*...*A[N-1]（即a0一直乘到an-1但刨了ai）不许使用除法
  - 长度判断
  - 先从头到尾将b的值设为从第0项乘到第i-1项（即前半部分）
  - 再从尾到头将b的值乘等，设置为第i+1项到n-1项
- 字符串中'.'表示任意一个字符，'*'表示前面一个字符可以出现任意次（包括0次），判断给定2个字符串匹不匹配
  - 递归
    - *str即表示当前字符
    - ‘\0’表示空字符
    - *(str+1)表示下一位字符
  - 特殊情况
    - 当前两个都为空
      - true
    - 第一个不空，第二个空
      - false
  - 如果下一个字符为*
    - 如果当前字符匹配或者（第一个不为空，第二个为.）
      - 递归进下一个字符
    - 否则false
  - 下一个字符为*
    - 如果当前字符匹配或者（第一个不为空，第二个为.）
      - 可能有2种情况
        - ab,a*ab
          - 即第二个字符串递归到+2的位
        - aaba,aab*a
          - 即第一个字符串递归到+1位
            - 变成a,b*a，进入到当前不匹配的循环中
        - 但是2种情况可以用一个return里或的形式解决
    - 如果当前字符不匹配
      - 递归进第二个字符串+2
- 判断一个字符串所表示的字符是否合法（关键在于搞清楚什么状况不合法）
  - 用三个bool分别表示有没有正负，有没有小数点，有没有E
  - for
    - 如果是e或者E
      - 如果已经有E了
        - false

                - 如果在最后一位
                    - false
                - 将hasE设为true
            - 如果+或者-
                - 如果已经有正负了且前一位不是e或E
                    - false
                - 如果没有正负，且不是第一位，且前一位不是e或者E
                    - false
                - 正负为true
            - 如果是小数点
                - 如果有E或者已经有小数点
                    - false
                - 小数点true
            - 如果字符串值不在0-9的区间内
                - false
    - true
- 查找字符串中第一个出现一次的字符
    - 利用hashtable
        - 用字符作为索引值计算出现次数
        - 红黑树底层
- 找出链表中环的入口节点，没有环则输出null
    - 节点数小于等于3
        - null
    - 设置一个快节点
        - 起始在next-next
        - 速度为一次向后走2
    - 慢节点
        - 起始在next
        - 速度一次走1
    - 找出2个节点相等的时候
        - 找不到则返回null
    - 经过计算若此时让快指针回到开头并让两个指针速度一样继续走，会在环开始出相遇，输出此时快慢任何一个指针即可
- 删除排序列表中相同的节点（1233445输出为125）
    - 用一个pre节点和last节点
    - 新建一个节点以防止原头节点丢失
    - pre初始化为新头，last为新头next
    - while（last不为null）
        - 如果下一个节点不为空且当前last和last下一个相等
            - 找到最后一个相等节点
                - 把pre的next指向最后一个相等节点的next

                - Last next
            - 否则pre，last都指向下一个
    - 输出新头的下一个
- 给定二叉树中的一个节点，返回中序遍历中的下一个节点（节点同时包含指向父节点的指针）
    - 空返回空
    - 若当前有右节点，找到右子树中最下边的左节点，输出
    - 若当前为叶节点
        - 设root为当前节点的父节点
        - 若父节点的左儿子为当前节点（即当前节点位于左子树，下一个输出就该是父节点）
            - 输出root
        - 再向上寻找父节点
        - 循环结束条件为当前节点已经是跟节点
    - 若都没找到则返回null
- 判断一个二叉树是不是对称二叉树
    - 递归
    - 若跟节点null
        - true
    - 新建一个函数参数为左右节点
    - 若左为空
        - 若右也为空返回true
        - 否则false
    - 若右为空
        - False
    - 若左右数值不等
        - False
    - 递归 左的左，右的右 且 左的右，右的左
- 之字形打印二叉树（第一行左到右，第二行右到左）
    - 用2个栈来辅助，一个int layer来记录当前层数
    - 大循环条件（s1不为空或s2不为空）
        - 奇数层
            - 将s1中的节点一个一个执行以下操作
                - Tmp=s1top
                - 将tmp的值push进临时数组
                - 若tmp有左
                    - 左进s2
                - 若有右
                    - 右进s2
            - layer更新，临时数组进入res
        - 偶数层跟奇数层操作一样

- 从上到下打印二叉树，每层都从左到右
    - 辅助队列
    - 特殊情况
    - 根入队列
    - 大循环当队列不为空
        - 跟上一题一样处理
        - 只不过循环条件变为length--（执行length次）

60-66题

2019年7月18日 木曜日　　14:46

- 实现两个函数，分别用来序列化和反序列化二叉树（就是把二叉树转化为字符串并转化回去，用什么序遍历不重要，反正都要转化回去）
  - 序列化
    - 就是单纯的递归深度查找
    - 若当前节点为空，push进一个不可能出现的数字（例如0x23333），来表示叶节点
  - 反序列化
    - 也是递归
    - 如果当前为0x23333，跳到下一位
      - ++p
      - 返回null
    - Res=当前
    - ++p
    - 递归进左
    - 递归进右
    - 返回res
  - 注意主函数中的参数格式
    - 反序列化子函数格式为int*&，所以str要转换为int*
- 给定一颗二叉搜索树，找出其中第k小的节点
  - 中序遍历递归
  - 用一个全局index表示现在找到第多少个了
  - 新建一个node来表示tmp
    - Tmp=left，k
  - 如果index==k
    - 返回proot
  - 注意，递归到左右节点的代码下面都要加上一句
    - 如果node不为空
      - 返回node
    - 因为如果node不为空，就说明已经在左/右子树找到了目标值，所以直接返回node
- 获取一个数据流的中位数，奇数个则输出中间值，偶数个输出中间俩的平均值
  - 用两个优先队列
    - priority_queue<int,vector<int>,less<int>> p;表示大顶堆
    - greater的表示小顶堆
    - 大顶堆的数都小于等于小顶堆的数
    - 且大顶堆永远比小顶堆长1，或者等长
  - 如果大顶堆长度比小顶堆多2

        - 挪一个到小顶堆
    - 如果大顶堆比小顶堆短1
        - 挪一个到大顶堆
    - 最后输出
        - 两个的平均值或大顶堆的top
- 找出所有滑动窗口里数值的最大值（题目解释不清楚，看例子吧）
    - 用一个双向队列deque来保存当前窗口元素
    - 同时保证front元素是当前窗口最大值的下标
    - 思路能搞懂，但是代码比较难想
    - 结合思路把代码背下来
- 判断矩阵中是否存在一条包括某字符串所有字符的路径
    - 通过递归来实现回溯法
    - 用一个相同大小的int数组来存flag，表示每一位是否可行
    - 选取任意一个格子作为起点（双for）
        - 如果helper函数返回true
            - 返回true
    - 否则返回false
    - helper函数
        - 参数（矩阵及其行列，当前ij，字符串，当前扫描到字符串第几位k，flagint数组）
        - 特殊情况或者i，j越界，或者当前字符不匹配或者当前flag位已经为1
            - 返回false
        - 如果已经扫描到字符串最后一位
            - True
        - flag当前位设为1
        - 递归进上下左右格子（或链接）
            - 返回true
        - 都没通过flag当前位重制为0
        - 返回false
- 从0，0开始移动，每次上下左右中移动一格
    - 和上一题思路差不多
    - 只不过判断语句变成一个sum函数
        - 输入i，计算i各位相加的和
    - 主函数
        - 定义flag数组
        - 进入helper
    - helper函数
        - 特殊情况及越界判断或者当前地址（用sum函数）已经不合法，或者当前已经走过了
            - False
        - 否则返回上下左右（或者连接）再+1（表示走得通）