1.K 个一组翻转链表

题目描述

给出一个链表，每 k 个节点一组进行翻转，并返回翻转后的链表。
k 是一个正整数，它的值小于或等于链表的长度。如果节点总数不是 k 的整数倍，那么将最后剩余节点保持原有顺序。
示例 :
给定这个链表：1->2->3->4->5
当 k = 2 时，应当返回: 2->1->4->3->5
当 k = 3 时，应当返回: 3->2->1->4->5
说明 :
你的算法只能使用常数的额外空间。
你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。

思路

其实这个题，是从尾到头开始反转的。
对每一个段，我们先反转其后的段，然后将该段反转后与反转后段的头节点连接。
在这里，其后段返回的头结点作为当前段的 pre 初始值
head 随 currentNode 而走，作为 mid 节点。提前保存 next 以免断链
最后，务必记得将 head 重新置为 next 的前一个节点。因为最后一个 next 指向的必定是下一个段未反转前的开头

WechatIMG56.jpeg

```
import java.util.Scanner;

/**
 * @author wuyafang
 * @Title: LIstReverse
 * @ProjectName ZHAO
 * @Description: TODO
 * @date 18/9/15 下午 3:07
 */
public class ListReverse {
    public class ListNode{
        int val;
        ListNode next;
        public ListNode(int val){
            this.val = val;
        }
    }
    public ListNode createList(String[] strArray){
```

```
        if(strArray.length==0)return null;
        ListNode head = new ListNode(Integer.valueOf(strArray[0]));
        ListNode temp = head;
        for(int i=1;i<strArray.length;i++){
            int tempVal = Integer.valueOf(strArray[i]);
            temp.next = new ListNode(tempVal);
            temp = temp.next;
        }
        temp.next = null;
        return head;
    }

    public ListNode reverseList(ListNode head,int k) {
        ListNode currentNode = head;
        int count =0;
        //遍历 k 个数字，找到下一个反转段开始的节点;如果不足的话，直接返回原 head，说明没有被反转
        while(currentNode!=null&&count!=k){
            count++;
            currentNode = currentNode.next;
        }
        //如果满足 k 个长度了
        if(count==k){
            //先反转后面的链表，返回后面反转完成后链表的头节点
            currentNode = reverseList(currentNode,k);
            while(count>0){
                count--;
                ListNode next = head.next;
                head.next = currentNode;
                currentNode = head;
                head = next;
            }
            //末尾的 head 连接到了下一个段开头(段最后元素提前保存的 next，指向下一个段未反转前的开头)。
            //所以需要恢复到上一个节点，才是该段头节点
            head = currentNode;
        }
        return head;

    }
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String line = in.nextLine();
        String[] strArray = line.split(" ");
```

```
        int k = in.nextInt();
        ListReverse reverse = new ListReverse();
        ListNode head = reverse.createList(strArray);
        ListNode reverseList = reverse.reverseList(head,k);
        while(reverseList!=null&&reverseList.next!=null){
            System.out.print(reverseList.val + " ");
            reverseList=reverseList.next;
        }
        System.out.print(reverseList.val);
    }
}
```

2.求 1！*2！*3！*……*n!末尾有多少连续的 0

http://blog.sina.com.cn/s/blog_147841ead0102vehv.html

题目描述

求 1！*2！*3！*……*n!末尾有多少连续的 0

输入 n，输出末尾 0 的个数

思路：

其实就是转化成求因子中有多少个 5.

1.可以找规律

其实简单分析一下，数字相乘时，末尾每有一个 0 都表示有一个因数 10。而 10 可以转化成 2*5，分解一下阶乘，我们发现 2 的个数远远超过 5，所以 n!末尾有多少 0 的问题转化成求因数中有多少个 5

2.从单个阶乘入手，先求出单个阶乘因数中 5 的个数再相加。

比如 100!中有 24 个 5 做因数，5，10，15，……，100 都有 5，其中 25，50，75，100 中都有 2 个 5，所以一共 24 个 5。即 n 中 5 的个数为 ：

n/5+n/5/5+……+1+0

最终代码如下

```
import java.util.Scanner;

/**
 * @author wuyafang
 * @Title: JieCheng
 * @ProjectName ZHAO
 * @Description: TODO
 * @date 18/9/15 下午 4:04
 */
public class JieCheng {
    public static void main(String[] args) {
```

```
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int count =0;
        int zeronumber = 0;
        for(int i=1;i<=n;i++){
            zeronumber = getFiveNumber(i);
            count+=zeronumber;
        }
        System.out.println(count);
    }
    //100!里面除了5，10，……，100这个20个被5整除的数，还有25，
50，75，100这四个数里还包含4个5。因数里一共24个5.
    public static int getFiveNumber(int n){
        int count=0;
        while(n!=0){
            count += n/5;
            n=n/5;
        }
        return count;
    }
}
```

测试用例：
输入：11 输出9
输入100，输出1124（100！中有24个5）

3.幼儿园分班问题

题目描述
题意大概是幼儿园要分成两个班，一共有n个小朋友，小朋友提出了m个要求，即不想和谁一个班。输入第一行为n，第二行为m，接下来是m对输入，比如1 4 表示1不想和4同班。问是否可以成功分班。成功输出1，不成功输出0
输入：

```
4
3
1 2
1 3
1 4
```

输出：
1
输入：

4
4
1 2
1 3
1 4
2 3

输出：
0
思路
这个看论坛上的大佬们说 类似于图染色算法。
也即给定 m 种颜色，相邻两点的颜色不同。
图染色问题一般有两种：

- 给定 m 种颜色，问能不能成功染色（分班问题即这样的问题）
- 给定 m 种颜色，问最少用多少颜色

实际上是 dfs，可以递归或者回溯来做。
其实这道题只要找到一种就可以返回了。不过接下来的解法会返回所有的可能。

Snipaste_2018-09-18_18-25-13.jpg

递归
参考：https://blog.csdn.net/qq_28888837/article/details/53284828

```
//这种递归的方式，应该是用了贪心的思想。如果不合适，就放弃这种可能。
反正会把所有可能列举，不对的就及时终止。
    public void sear(int[][] graph,int[] color,int k){
        for(int i=1;i<=m;i++){
            color[k]=i;
            //符合条件的才继续，否则放弃这种选择
            if(isOk(graph,color,k)){
                if(k==graph.length-1){
                    ans++;
                    //找到符合条件的一个组合了
                    ArrayList<Integer> sol = new
ArrayList<Integer>();
                    for(int j=0;j<color.length;j++){
                        sol.add(color[j]);
                    }
                    result.add(sol);
```

```
                }else{
                    //向下递归
                    sear(graph,color,k+1);
                }
            }
        }
    }
```

回溯

```
//dfs+回溯法
    public void sear1(int[][] graph,int[] color){
        int i=0;
        //回溯法的终止条件
        while(i>=0){
            color[i]++;
            while(color[i]<=m){
                if(isOk(graph,color,i)){
                    break;
                }else{
                    color[i]++;
                }
            }
            //已搜索到最后一个节点
            if(color[i]<=m && i==color.length-1){
                ans++;
                //找到符合条件的一个组合了
                ArrayList<Integer> sol = new ArrayList<Integer>();
                for(int j=0;j<color.length;j++){
                    sol.add(color[j]);
                }
                result.add(sol);
            }else if(color[i]<=m && i<color.length-1){//继续向下搜索
                i++;
            }else{//不符合条件，需要回退
（isOk(i)==false||i>color.length-1）
                color[i]=0;
                i--;
            }
        }
    }
```

完整的程序为

```
import java.util.ArrayList;
```

```
import java.util.Scanner;

/**
 * @author wuyafang
 * @Title: SameClass
 * @ProjectName ZHAO
 * @Description: TODO
 * @date 18/9/15 下午 3:39
 */
public class SameClass {
    private static int m;
    private static ArrayList<ArrayList<Integer>> result = new
ArrayList<>();
    private static int ans =0;

    //判断 k 节点是否与其相邻节点颜色相同
    private boolean isOk(int[][] graph,int[] color,int k){
        for(int i=0;i<k;i++){
            if(graph[k][i]==1&&color[i]==color[k]){
                return false;
            }
        }
        return true;

    }
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int requestNumber = in.nextInt();
        int[][] graph  = new int[n][n];//邻接矩阵代表图
        int[] color = new int[n];
        for(int i=0;i<requestNumber;i++){
            int first = in.nextInt();
            int second = in.nextInt();
            graph[first-1][second-1] = 1;
            graph[second-1][first-1]= 1;
        }
        SameClass same = new SameClass();
        //设置染色数。这里班级有两个。染色数为 2
        m=2;
//        same.sear(graph,color,0);
        same.sear1(graph,color);
        if(result.isEmpty()){
            System.out.println("无法满足所有人的请求进行分班");
```

```
            System.out.println(0);
        }else{
            System.out.println("可以分班，一共有"+ans+"种分班方式");
            System.out.println(result.toString());
            System.out.println(1);
        }
    }
}
```