# 远景智能编程题

1. 找到K个最接近的元素

**题目描述**

给定一个排序好的数组，两个证书k和x，从数组中找到最接近x（两数之差最小）的k个数。返回的结果必须要是按升序排好的。

如果有两个数与x的差值一样，优先选择数值较小的那个数。

**输入描述**

第一行为排序好的数组arr

第二行为查找的个数k

第三行为基准值x

**输出**

排序好的数组

**示例**

输入

1,2,3,4,5

4

3

输出

1,2,3,4

**题目描述**

某风电场每台风机的发电量和距离升压站的距离各不相同，如风机1：发电量30，距离20；风机2：发电量35，距离25；风机3：发电量25，距离18……。要求在输电总距离限定（如小于100）的前提下，选择风机向升压站输电，使得输送的电量最大。

**输入描述**

风机离升压站的距离，如 30 20 35 40

风机发电量，如20 28 25 30

距离升压站距离的限制，如50

**输出**

风机的配置方案与最大风量

**示例**

输入

30 20 35 40

20 28 25 30

50

# 新浪/微博

1. 输入一组版本号字符串数组，找出其中版本号最小的值。版本号的格式为major.mirror.patch.ext，每个点位均为数字，但点分位个数可能不一致。版本好的大小比较按照点分数字从前往后比较，如7.10.2 > 7.3，3.1 > 2.3.4。要求输入String类型的版本号数组，输出其中的最小版本号。

**输入**

[“7.10.2”, “3.10.2”, “3.1”, “7.10.2.2”, “7.2.”]

**输出**

3.1

1. 设计和实现一个LRU（最近最少使用）缓存机制，支持
2. 获取数据get(key)：如果key位于缓存中，获取key对应的值（总是正数），否则返回-1
3. 写入数据put(key,value)：如果key不存在，则写入其数据。当缓存容量达到上限时，应该在写入新数据前删除最近最少使用的数据值，从而为新数据留出空间。要求在O(1)的时间复杂度内完成上述操作。

示例：

2 #设定LRU的缓存容量

put(1,1)

put(2,2)

get(1) #输出1

get(2) #输出2

get(3) #输出-1

put(3,)

1. 有一个包含uid的100GB数据，每个uid为数字，每个uid所占内存不超过10字节，找出100GB数据中，出现次数最多的100个uid，并以列表形式输出。

# 阅文集团

1. 使用2个线程，交替输出 A:1B:1A:2B:2A:3B:3
2. 输入一个数字，输出它的反转。若为负数，则输出它相反数的反转的负数。

如：1234 --> 4321，-1234 --> -4321

1. 负载均衡的作用。常见的负载均衡策略。并用代码实现其一。

# 中兴

LeetCode 179

LeetCode 568

# 百度

1. 输入n, m, k，给定非负整数a, b，求满足(n-a)(m-b) <= k且a+b尽量小，输出这个a+b。
2. 从1号点到n号点，有n个点，m条边，问是否能在两条边以内从1号点走到n号点。3<=n<=2\*105，1<=m<=105。输入n，m，后接m行表示边，每行两个数字表示对应两个点之间存在边。
3. N间机房，1,2,3,…,N，机房之间有M条数据线，每条数据线有两个属性：老化指数a，拥挤指数b。出于经费考虑，关闭部分数据线，但必须保证任意两个机房间能通过数据线直接或间接连通以保证数据传输。定义Ma为未关闭数据线的老化指数最大值，Mb为所有未关闭数据线拥挤指数最大值。求K=Fa\*Ma+Fb\*Mb（Fa, Fb为常数）最小值。输出第一行为N, M，1<=N<=400，1<=M<=50000；第二行为Fa, Fb，Fa>=1，Fb<=109，后接M行，每一行表述数据线连接的两个机房以及老化指数 拥挤指数 u v a b

# 华为

## 9月7号笔试题

1. 给定一个数组array，从数组0号下标开始移动。第一次移动的步长任选，第一步移动到i后，后续移动步长由array[i]决定。若能移动到数组的最后一个位置，则记录下总的移动次数。求任意输入的数组的最小移动次数。

**思路：**

**1. 从前往后遍历，判断是否能到，将每次成功的移动放在数组中，排序取出第一个元素。**

1. 给定n个不同的数字，从中任取3个数字组合成一个技能，3个数字的顺序不同 视为相同的技能，问输入的n能形成多个种不同的技能。

**思路：实质为求子集问题**

数组的每个元素，可以有两个状态：

1、不在子数组中（用 000 表示）；

2、在子数组中（用 111 表示）。

从 0 到 2 的数组个数次幂（不包括）的整数的二进制表示就能表示所有状态的组合。



作者：liweiwei1419

链接：https://leetcode-cn.com/problems/subsets/solution/hui-su-python-dai-ma-by-liweiwei1419/

来源：力扣（LeetCode）

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

import java.util.ArrayList;

import java.util.List;

public class Solution5 {

public List<List<Integer>> subsets(int[] nums) {

int size = nums.length;

int n = 1 << size;

List<List<Integer>> res = new ArrayList<>();

for (int i = 0; i < n; i++) {

List<Integer> cur = new ArrayList<>();

for (int j = 0; j < size; j++) {

if (((i >> j) & 1) == 1) {

cur.add(nums[j]);

}

}

res.add(cur);

}

return res;

}

public static void main(String[] args) {

int[] nums = {1, 2, 3};

Solution5 solution5 = new Solution5();

List<List<Integer>> subsets = solution5.subsets(nums);

System.out.println(subsets);

}

}

作者：liweiwei1419

链接：https://leetcode-cn.com/problems/subsets/solution/hui-su-python-dai-ma-by-liweiwei1419/

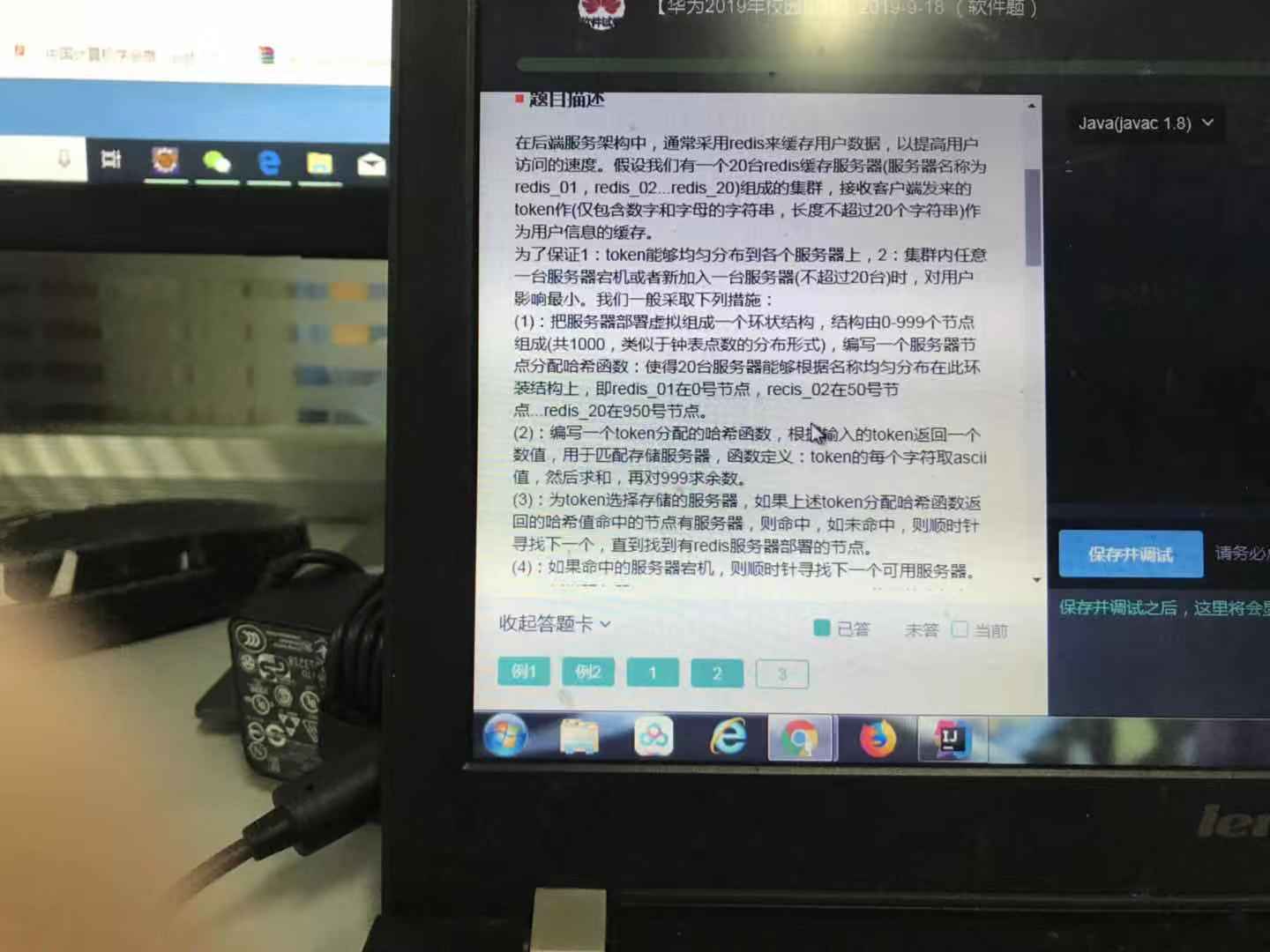
来源：力扣（LeetCode）

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

1. 给定公式形成的字符串，解析字符串，得到公式的值。

## 9月11号笔试题

1. 输入多组数据，判断是否有效
2. 字符串空格转0，大写转小写，倒序输出



## 华为面试算法（亲历）

1. LeetCode316 去除重复字母

给定一个仅包含小写字母的字符串，去除字符串中重复的字母，使得每个字母只出现一次。需保证返回结果的字典序最小（要求不能打乱其他字符的相对位置）。

示例 1:

输入: "bcabc"

输出: "abc"

示例 2:

输入: "cbacdcbc"

输出: "acdb"

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/remove-duplicate-letters

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

1. 打印蛇形矩阵

Description  
蛇形矩阵是由1开始的自然数依次排列成的一个矩阵上三角形.  
Input  
本题有多组数据,每组数据由一个正整数N组成.（N不大于100）  
Output  
对于每一组数据,输出一个N行的蛇形矩阵.两组输出之间不要额外的空行.矩阵三角中同一行的数字用一个空格分开.行尾不要多余的空格.  
Sample Input  
5  
Sample Output  
1 3 6 10 15  
2 5 9 14  
4 8 13  
7 12  
11

## 华为面试算法整理（牛客网）

1. LeetCode575 分糖果

给定一个**偶数**长度的数组，其中不同的数字代表着不同种类的糖果，每一个数字代表一个糖果。你需要把这些糖果**平均**分给一个弟弟和一个妹妹。返回妹妹可以获得的最大糖果的种类数。

示例 1:

输入: candies = [1,1,2,2,3,3]

输出: 3

解析: 一共有三种种类的糖果，每一种都有两个。

最优分配方案：妹妹获得[1,2,3],弟弟也获得[1,2,3]。这样使妹妹获得糖果的种类数最多。

示例 2 :

输入: candies = [1,1,2,3]

输出: 2

解析: 妹妹获得糖果[2,3],弟弟获得糖果[1,1]，妹妹有两种不同的糖果，弟弟只有一种。这样使得妹妹可以获得的糖果种类数最多。

注意:

数组的长度为[2, 10,000]，并且确定为偶数。

数组中数字的大小在范围[-100,000, 100,000]内。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/distribute-candies

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**分析：**

方案1：排序，通过比较排序数组的相邻元素找出唯一的元素，统计不同元素的个数count，返回min(count, n/2)

方案2：利用集合（Set）只包含唯一元素的特性，遍历数组，将元素放入集合Set中。最后统计Set的长度，返回min(set.size(), n/2)。

1. 实现pow(x, n)，求x的n次方

实现 pow(x, n) ，即计算 x 的 n 次幂函数。

示例 1:

输入: 2.00000, 10

输出: 1024.00000

示例 2:

输入: 2.10000, 3

输出: 9.26100

示例 3:

输入: 2.00000, -2

输出: 0.25000

解释: 2-2 = 1/22 = 1/4 = 0.25

说明:

-100.0 < x < 100.0

n 是 32 位有符号整数，其数值范围是 [−231, 231 − 1] 。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/powx-n

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**分析：**

方法一：暴力解法。

直接模拟该过程，将 x 连乘 n 次。若n<0，则x=1/x，n=-n。需关注边界条件，尤其是正整数和负整数的不同范围限制。

复杂度分析

时间复杂度：O(n). 我们需要将 x 连乘 n 次。

空间复杂度：O(1). 我们只需要一个变量来保存最终 x 的连乘结果。

运行结果：214/304个通过测试用例，超出运行时间限制

方法二：直接递归

pow(x, n) = x \* pow(x, n-1)

运行结果：214/304个通过测试用例，栈溢出

时间复杂度：O(n)

空间复杂度：O(n)

方法三：带备忘录的递归，

pow(x, n) = x \* pow(x, n-1)

运行结果：

291/304个通过测试用例，超出内存限制

时间复杂度：O(n)

空间复杂度：O(n)

方法四：快速幂集法（递归）

(xn)2­ = x2\*n。

令 A = xn/2，可以根据 n 的奇偶性来分别讨论xn的值。

如果 n 为偶数，xn=A∗A。

如果 n 为奇数，那么 A∗A=xn-1，xn=A∗A∗x。

该方法可以很方便的使用递归实现。我们称这种方法为 "快速幂"，因为我们只需最多 O(log n)次运算来得到xn。

时间复杂度：O(log n). 每一次我们使用公式 xn = x2\*n ，n 都变为原来的一半。因此我们需要至多 O(logn) 次操作来得到结果。

空间复杂度：O(log n)。 每一次计算，我们需要存储xn/2的结果。 我们需要计算 O(log n)次，所以空间复杂度为O(log n)。

方法五：快速幂算法（循环）

未理解！

1. LeetCode42 接雨水问题

给定 n 个非负整数表示每个宽度为 1 的柱子的高度图，计算按此排列的柱子，下雨之后能接多少雨水。



上面是由数组 [0,1,0,2,1,0,1,3,2,1,2,1] 表示的高度图，在这种情况下，可以接 6 个单位的雨水（蓝色部分表示雨水）。。

示例:

输入: [0,1,0,2,1,0,1,3,2,1,2,1]

输出: 6

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/trapping-rain-water

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**分析：**

方法一：暴力解法，向左、向右遍历找到当前元素左边的最大高度与右边最大高度，取两者的较小值减去当前高度即为当前单位所能存储的水量。每一次的遍历都重复扫描的多余的元素。

时间复杂度O(n^2)

空间复杂度O(1)

方法二：动态规划。利用数组存储每个位置左边最大值、右边最大值，则某个元素左边最大值以及右边最大值只需要考察邻近的元素，避免重复扫描。

时间复杂度O(n)

空间复杂度O(1)

**变体**：求所有凹槽里盛水最多的一个凹槽有多少水

class Solution {

public int trap(int[] height) {

int max = 0;

int len = height.length;

if(len < 3)

return 0;

int[] leftMax = new int[len];

int[] rightMax = new int[len];

leftMax[0] = height[0];

for(int i=1; i<len; i++)

leftMax[i] = Math.max(height[i], leftMax[i-1]);

rightMax[len-1] = height[len-1];

for(int i=len-2; i>=0; i--)

rightMax[i] = Math.max(height[i], rightMax[i+1]);

for(int i=0; i<len; i++){

max = Math.max(max,(Math.min(leftMax[i], rightMax[i]) - height[i]));

}

return max;

}

}

1. 链表相关的代码
2. 爬楼梯
3. Dikstra算法
4. 找出一个字符串中最长的回文子串
5. 一个由数字组成的字符串分割成左右两半L、R，左边满足对L%a==0，右边满足R%b==0,找出最短的L，提示以0开头的都是非法的；
6. 给出一个数组，找出数组中的满足和为100的两个数的所有组合，提示如{98，2，98}结果为2，{98，2，2，98}结果为4
7. N个人找出法官是谁
8. 链表反转
9. 剑指offer剪绳子
10. 快速排序
11. 连续元素和等于目标值的最大长度
12. 字符串最长回文子串
13. 24点游戏
14. m和n升水凑k升水的最少步骤
15. 2n+1个数字，每个数字都出现2次，仅有一个元素出现一次，找出出现一次的那个数字。
16. 最长回文序列
17. 栈实现队列
18. 三数之和改编版
19. 链表相加
20. 给定一个数字，输出1-n的全排列
21. 手写算法，给定一个数，小数无限循环，写出这个小数的分数形式
22. 栈实现括号匹配
23. 最大子集合
24. 链表的反转
25. 四则运算的字符串求结果
26. 单词反转
27. 链表找中间结点
28. 股票买入卖出的最大值
29. 堆排序
30. 求两个有序数组的交集

# 招银网络科技

1. 和为S的连续正数序列

【题目描述】小明很喜欢数学,有一天他在做数学作业时,要求计算出9~16的和,他马上就写出了正确答案是100。但是他并不满足于此,他在想究竟有多少种连续的正数序列的和为100(至少包括两个数)。没多久,他就得到另一组连续正数和为100的序列:18,19,20,21,22。现在把问题交给你,你能不能也很快的找出所有和为S的连续正数序列? Good Luck!

【输出描述】输出所有和为S的连续正数序列。序列内按照从小至大的顺序，序列间按照开始数字从小到大的顺序。

【解题思路1】暴力解法

//1. 双层循环，找出所有和为S的连续正数序列。

【源码1】

import java.util.ArrayList;

public class Solution {

public ArrayList<ArrayList<Integer> > FindContinuousSequence(int sum) {

ArrayList<ArrayList<Integer>> arrs = new ArrayList<ArrayList<Integer>>();

for(int i=1; i<sum; i++){

int s = 0;

ArrayList<Integer> arr = new ArrayList<Integer>();

for(int j=i; j<sum; j++){

if(s < sum){

s += j;

arr.add(j);

if(s == sum){

arrs.add(arr);

break;

}

}else{

break;

}

}

}

return arrs;

}

}

【解题思路2】夹逼方案的变体

//1. 延续题目“和为S的两个数字”的思想。设置一大一小两个指针。

//2. 初始状态下，small指向1，big指向2。如果从small到big的和大于S，则从序列中去掉较小的值，也就是small向后移动一个位置。若从small到big的序列和小于S，则big向后移动一个位置，以便序列包含更多的数字。

//3. 因为题目中要求最少是两个数字，所以small最大为（s+1）/2。

【源码2】

import java.util.ArrayList;

public class Solution {

public ArrayList<ArrayList<Integer>> FindContinuousSequence(int sum) {

ArrayList<ArrayList<Integer>> lists=new ArrayList<ArrayList<Integer>>();

if(sum<=1){return lists;}

int small=1;

int big=2;

while(small!=(1+sum)/2){ //当small==(1+sum)/2的时候停止

int curSum=sumOfList(small,big);

if(curSum==sum){

ArrayList<Integer> list=new ArrayList<Integer>();

for(int i=small;i<=big;i++){

list.add(i);

}

lists.add(list);

small++;big++;

}else if(curSum<sum){

big++;

}else{

small++;

}

}

return lists;

}

public int sumOfList(int head,int leap){ //计算当前序列的和

int sum=head;

for(int i=head+1;i<=leap;i++){

sum+=i;

}

return sum;

}

}

a. 有个小技巧，每次循环过程中都要求当前连续序列的和，但考虑到每次操作之后的序列和操作之前的序列大部分数字都是一样的，只是增加一位或者减少一位数字，所以我们可以在前一个序列的和基础上求操作之后的序列的和。这样可以减少不必要的运算。

【解题思路3】数学公式法

//1. 从start到end的连续正数序列的和记为sum。

//2. sum = (start+end)\*(end-start+1)/2

//3. 遍历一次数组，令start为当前遍历值，解上述方程，求出end的值，跟最大值比较，若在给定的数组之内，则为一组解。

链接：https://www.nowcoder.com/questionTerminal/c451a3fd84b64cb19485dad758a55ebe

来源：牛客网

//我竟然解了二元一次方程，我是菜鸟我怕谁。。

//(start+end)\*(end-start+1)/2=sum;

//根据start解出end。。。。。

public ArrayList<ArrayList<Integer> > FindContinuousSequence(int sum) {

         ArrayList<ArrayList<Integer>> result=new ArrayList<ArrayList<Integer>>();

        if(sum<3) return result;

        for(int i=1;i<=sum/2;i++){

            int value=1+4\*i\*i-4\*i+8\*sum;

            int valueSqrt=(int)Math.sqrt(value);

            if(value>=25&&valueSqrt\*valueSqrt==value){

                ArrayList<Integer> path=new ArrayList<Integer>();

                for(int j=i;j<=(valueSqrt-1)>>1;j++)

                    path.add(j);

                result.add(path);

            }

        }

        return result;

}

1. 中缀表达式转后缀表达式

# 字节跳动

1. 逆转字符串
2. 手写线程安全的生产者和消费者
3. 求数组中第K大的值
4. 求树的路径和
5. 写一个归并排序
6. 手写红黑树
7. 求正整数的平方根
8. 链表分段反转
9. 非递归中序遍历二叉树
10. 数组中某个数字出现一次，其他出现两次，找出出现一次的数字
11. 判断树B是否为树A的子树
12. 给出N个32位无符号数，有重复，找出重复率最高的前K条
13. 给出几千万条数据，存在外存里，如何找到最大的前100条（分块搜索+多路归并）
14. 二维数组，从左到右升序，从上到下降序，查找某个元素是否存在
15. 快速排序
16. 找一个完全二叉树的最后一个节点，时间复杂度要求logN
17. 反转链表
18. 找出前K大的数（建堆）
19. 查找二叉树的最大深度
20. 二叉树遍历
21. 查找数组中出现次数大于数组长度一半的元素
22. 两个链表有交叉，找出交叉点
23. 输入一个集合，输出它的子集
24. 爬楼梯，1楼到2楼有20个阶梯，每次走1步或2步，有多少种走法。
25. 找出字符串不重复的最长子串
26. 求包含n个数字的数组中最大的m个数
27. n\*n的矩阵，找一条最长的路径，可上下左右移动。
28. 非递归先序遍历二叉树，层序遍历二叉树实现求最大值。
29. 一次买卖，计算最大收益
30. 生成全排列
31. 最小编辑距离
32. 检查输入字符串是否为ip地址
33. 2n个人围成一圈，两两握手，没有交叉一共有多少种。
34. 二叉树的广度优先遍历（针对第三个爬虫）
35. 简述迪杰斯特拉算法（针对第一个项目）
36. 栈：

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

push(x) -- Push element x onto stack.

pop() -- Removes the element on top of the stack.

top() -- Get the top element.

getMin() -- Retrieve the minimum element in the stack

1. 把汉字表示的数字转换成阿拉伯数字
2. 进程间通信、中断机制
3. 对一个奇数位递增，偶数位递减的单链表实现整体的递增
4. 用数组存储二叉树。

# 网易互娱

1. 实现LRU
2. 实现区间合并
3. 实现并查集（Union-Find）以及并查集的改进
4. 合并列表中的字符串并输出

# 国金证券

1. 求两个有序数组的交集
2. 求两个数字之间的所有素数
3. 股票利润最大
4. 滑动窗口最大值

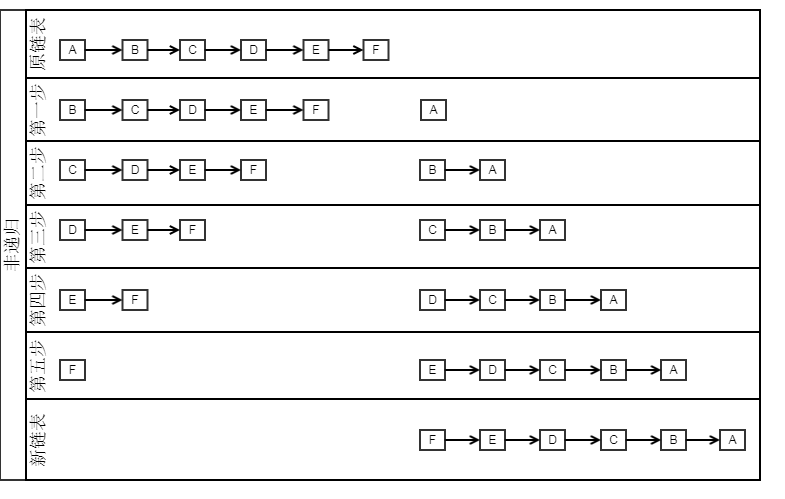
# 德拓信息

1. 统计字符串中各个字符出现的次数

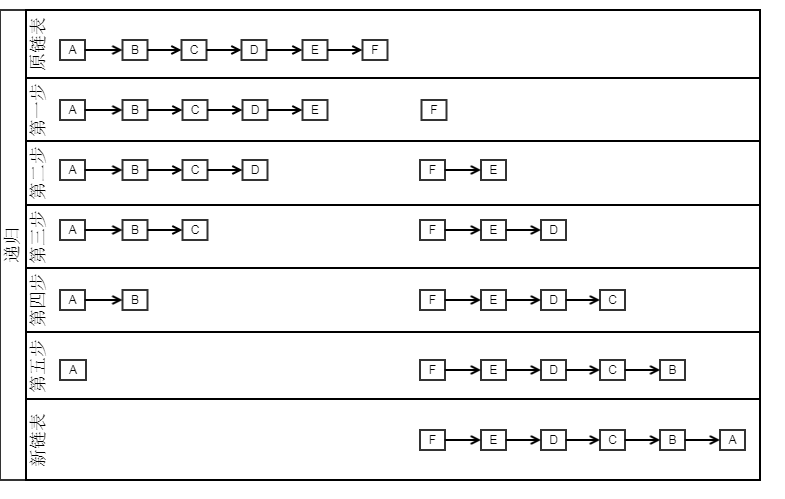
HashMap。

1. 单链表翻转

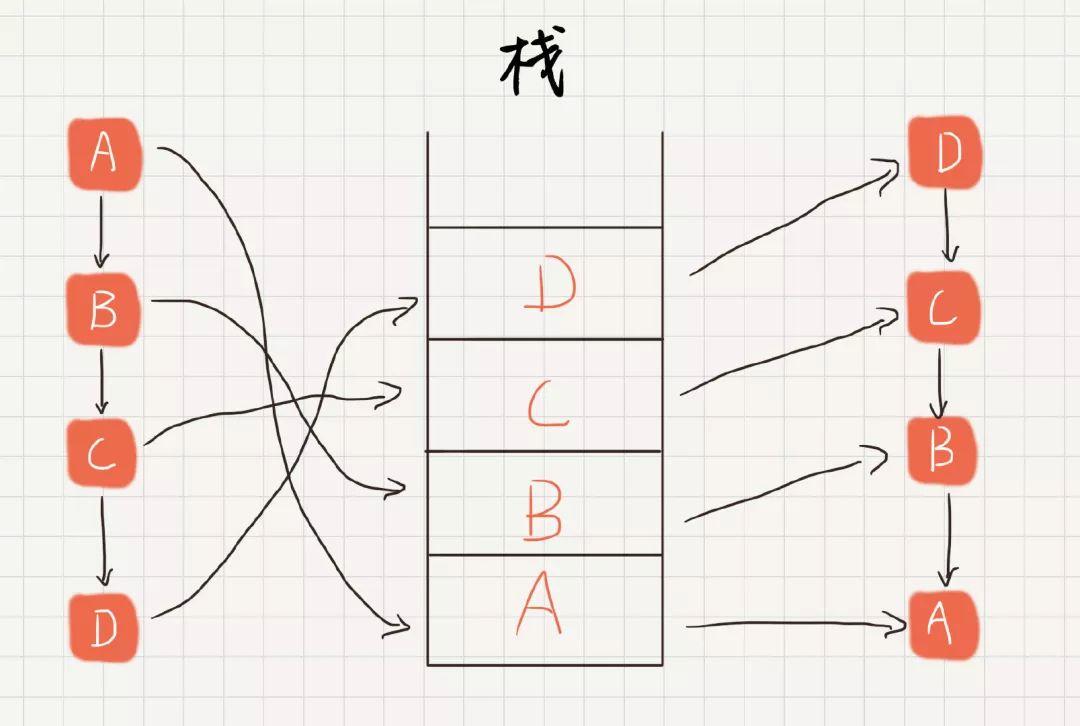
方法一：遍历法（链表遍历的过程中将指针顺序置换）。p指向前一个结点，q指向当前结点，temp临时保存q的下一个结点。时间复杂度O(N)，空间复杂度O(1)



方法二：递归实现。时间复杂度O(N)，空间复杂度O(N)



方法三：栈实现。时间复杂度O(N)，空间复杂度O(N)



1. 精确计算1000的阶乘

**分析：利用BigInteger类。**

1. 最大子序和（LeetCode53）

给定一个整数数组 nums ，找到一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

示例:

输入: [-2,1,-3,4,-1,2,1,-5,4],

输出: 6

解释: 连续子数组 [4,-1,2,1] 的和最大，为 6。

进阶:

如果你已经实现复杂度为 O(n) 的解法，尝试使用更为精妙的分治法求解。

**分析：动态规划**

# 平安科技

1. 根据输入的子网掩码，判断输入的两个ipv4地址是否属于同一个子网。

**分析：子网掩码是用来判断任意两个ipv4地址是否属于同一个子网的根据。ipv4地址与子网掩码相“与”后可获取ip地址的网络号。**

**子网掩码与ipv4地址的结构相同，是32位二进制数，分为按“.”分割的四个8位二进制数。其中，网络号部分全为“1”，主机号部分全为“0”。若两个ipv4地址分别与子网掩码相“与”后的获取的网络号相同，则说明他们同属一个子网。**

**示例：**

IP地址 192.168.0.1

子网掩码 255.255.255.0

转化为二进制进行运算：

IP地址 11010000.10101000.00000000.00000001

子网掩码　 11111111.11111111.11111111.00000000

AND运算

11000000.10101000.00000000.00000000

转化为十进制后为：

192.168.0.0

IP地址 192.168.0.254

子网掩码 255.255.255.0

转化为二进制进行运算：

IP地址 11010000.10101000.00000000.11111110

子网掩码 11111111.11111111.11111111.00000000

AND运算

11000000.10101000.00000000.00000000

转化为十进制后为：

192.168.0.0

对两台计算机IP地址与子网掩码的AND运算后，运算一致，均为192.168.0.0。所以这二台计算机可视为是同一子网络。

子网掩码的给出主要有两种方法：

1. 给出形如ipv4地址形式的子网掩码，如192.0.0.0（等价于11000000.00000000.00000000.00000000）
2. 给出子网掩码的位数（即从最高位开始1的数目），如2（等价于 11000000.00000000.00000000.00000000）.
3. 求给定字符串A和B 的最长公共子串，并输出。若A和B存在多个最长公共子串，则按照字母序排序后输出。

**分析：使用动态规划进行求解。**

**将字符串划分为子串，以字符串的每个字符作为格子的坐标轴。**

**格子中的数值表示以字符A[i]结尾的A的子串A’ 和 以B[j]结尾的B的子串B’ 的 最大公共长度。若字符A[i] == B[j]，则dp[i][j] = dp[i-1][j-1] + 1。**

**A和B最长公共子串的长度为格子中的最大值，即max(dp[i][j]) （0 <= i <= lenA, 0 <= i <= lenB）。**

**可将 最长公共子串存入 String数组，之后直接调用Arrays.sort(String[])即可自动获取按照字典序排序后的String 数组。**