仅适用2020届准应届生

名企校招实习备战攻略

技术篇

目录

一、前言

1.1 春招介绍

春招，即春季校园招聘，是企业面向应届生和准应届生专门举办的招聘。时间一般在春季（3 月-5 月），包括了应届生补招和实习生招聘。

有些同学的观念是：找工作是毕业前的事情，大四的时候再去实习就来得及。完全错误，尤其是近几年，随着企业要求严格，应试者的能力水平也在不断提高。大家越早的进入实习，越早的储备经验，能力提升才能更快。

所以作为大三学生的你，请一定要抓住 2019 年春招机会，找一份能够给你添彩的实习！

1.1.1 春招时间

3 月-4 月

准确讲，春招开始时间是在春节之后。所谓“金三银四”，3 月份开始就会有很多企业开始春招，4 月左右集中爆发。由于正常的大批量应届生会在 6-7 月毕业，因此春招最多持续到 5 月份，算下来整个春招也就不过短短两个月。

2019 年的春节假期是 2 月 5 日-2 月 11 日，预估今年的**春招会在 2 月底 3 月初展开**，所以准应届生现在就要开始为春招做准备了。

1.1.2 春招面向对象

春招是面向应届生的校园招聘，以及**准应届生的实习生招聘**。

2019 年的春招，即为面向 2019 届应届生的校园招聘（补招），以及面向 2020 届应届生的暑期实习招聘。

（21 届、22 届的同学如果想要找实习，也可以在春招的时候去试试，部分企业也会接受年级更小的学生作为实习生进入企业实习。）

1.1.3 春招信息渠道

• 公司官网+官方招聘公众号

正常的校园招聘信息获取途径最快捷最准确的是公司官网或官微，但是官网那么多，一个个翻找会浪费很多时间和精力。

其次，由于实习生招募，除大厂的暑期实习生/培训生项目外，其余很多企业的实习生招募是非常零散的，不会发布在官网/官方招聘公众号上。一般由用人部门员工或实习生直接招募，他们会在朋友圈/各大论坛/各大招聘网站发布招聘信息，但如果你没有相应认识的人，可能在你还未知道实习消息之前，实习生招募就已经悄然结束了。

这时候，认准牛客网就可以了。

• **牛客网**

实习广场：企业最新招聘信息，由用人部门/企业 HR 直接发布，信息真实可靠，简历处理效率快，岗位齐全，随时可以找到职位发布者直接联系。

地址：牛客网>求职>实习兼职：https://www.nowcoder.com/job/center

讨论区：已拿到 offer 的老牛友以及 HR 发布的内推信息

PC 端地址：牛客网>讨论区>招聘信息 https://www.nowcoder.com/discuss?type=7

小程序地址：扫描右方二维码即可观看

关注牛客论坛，随时查看同行动态、校招/内推信息、offer 比较、技术交流等等。

企业校招日程汇总：各企业校招时间明细

地址：牛客网>公司真题模考>（左侧位置）IT 校招全国统一模拟笔试

公众号：

招聘消息汇总：每日推送校招/内推资讯以及面经干货。

校招小管家：绑定手机号，可及时查询自己的笔试面试进度和信息。

实习助手牛能：已建立全国 13 个城市实习群，每日分享实习信息、解答疑问。扫码加牛可乐好友，回复城市名称即可进群。

招聘消息汇总　　校招小管家　　实习助手@牛能

通过以上方式，大家能更快获取招聘信息，避免浪费大量时间在查找信息上。事半功倍，将时间运用到更重要的事情——准备春招。

1.2 技术岗位考前须知

越来越多的企业在校招笔试中采取线上笔试的方式进行，尤其是技术岗位。部分考生由于线上编程经验缺乏，很容易在考场上出现手足无措的情况，影响做题速度和笔试成绩。

下面将对**如何使用在线考试系统**进行详细说明，以及**在线编程所必须要知道的重点！**

1.2.1 在线考试系统使用说明

下面详细讲述在线笔试的**完整流程以及注意事项**

第一步：投递简历

注意：邮箱和手机号等信息一定检查仔细，因为后续通知全是通过邮件和短信提醒。

第二步：笔试通知邮件和短信

注意：如果收到短信没有收到邮件，可能是你邮箱填错或者邮箱设置了拒收等原因，可以通过关注公众号：**校招小管家 > 绑定收到短信的手机号 > 查询我的笔试。**

第三步：检查考试设备

1、请使用**谷歌 Chrome、火狐浏览器**访问笔试网址。

如遇到页面加载不出来、摄像头不好使等情况，**优先采取措施：换另一个浏览器试一下。**

浏览器下载地址：https://www.nowcoder.com/discuss/3793

2、确保电脑带有摄像头，并确保摄像头能够正常使用。

摄像头检测：https://www.nowcoder.com/cts/3942933/summary#0

（1）**摄像头黑屏、无法拍照等情况：**优先采取措施：换另一个浏览器。其次检查浏览器有没有 adblock adguard 等这种广告屏蔽插件，关闭后重试

（2）**更换为前置摄像头：**请点击地址栏右侧的设置>高级>隐私设置和安全性>内容设置>摄像头，进行调试即可

3、考试前**请关闭其他浏览器窗口，关闭 QQ、微信、Skype 等即时通信软件，关闭屏保，关闭 Outlook 等有弹窗提示消息的软件，**否则会被记录离开网页。

4、确保网络连接畅通，网速应在 100KB/S 以上，**建议使用手机 4G 热点链接网络。**

5、考试时允许使用草稿纸，请**提前准备纸笔**。考试过程中允许上厕所等短暂离开，但请控制离开时间。

第四步：笔试做题流程

1、试卷中会有一种以上个题型，进入考试后**请仔细查看共有几个题型。**

2、可选择任意题型进入做题，**所有题型一旦提交后将无法返回修改。**

3、**可通过试卷页面底部答案卡进行同一题型试题切换**，但一旦进入某一类题型，提交后方可进入下一题型。

4、如**遇突发情况**，如断网、电脑死机、断电等，请直接刷新页面，或**关闭浏览器后重**

新通过考试地址进入。题目会自动保存，所以不用担心。

5、考试环境体验：https://www.nowcoder.com/cts/3942933/summary#

1.2.2 在线编程题重点须知

循环输入输出处理常见问题

1、为什么需要循环输入输出：通常来说 OJ 对于每道题里面有.in 和.out 文件，分别表示测试数据的输入和输出。如果某些编程题的所有数据都只做在一个.in 和一个.out 中，这样就会变成多组测试了，所以需要提交的代码中循环处理。

2、处理方法：其实这个问题可以避免，就是编程题后台每个样例做一组对应的.in 和.out 文件，这样就变成单组测试，代码就不需要循环处理，但是平时练习的题目质量不一，这个问题都会出现。

代码里面循环处理了即使是单组测试也会完全没问题，所以为了偷懒，可以全写成循环处理。

3、还有一个坑：但是这里会发生一个问题(十分常见!!!!)，如果测试数据是多组的，但是恰巧你代码里面需要些标记数组，map，set 等，在循环内一定记得清空，不然可能会产生前面的测试样例影响了后续数据的答案。

对于各种语言的一些基本知识

做编程题强烈建议使用 C/C++，做编程题强烈建议使用 C/C++，做编程题强烈建议使用 C/C++，做编程题强烈建议使用 C/C++

重要的事情比三遍再多说一遍，下面说说具体理由：

1、出题人通常会使用 C/C++编写标程，数据也是由标程制造的，所以使用跟出题人一样的语言会比较稳妥

2、C/C++效率比较高，通常来说一般 OJ 对于一道题目的时限限制会区分 C/C++和其他语言，通常处理方式是假设 C/C++时限是 1s，其他语言就会给 2 倍时限，甚至更多。

3、关于 cin cout 和 scanf printf。做题的时候尽量使用 scanf printf。下面告诉一个小常识，不要惊讶：cin cout 比 scanf printf 慢 20 倍左右!!!!!!!

一旦遇到大数据量，光是读入就有可能跪掉。

你或许可以使用 std::ios::sync_with_stdio(false); 这条语句关掉 scanf 和 cin 的同步，加快效率。但是即使这样 cin 还要慢 5 倍左右，而且一旦使用了这条语句，scanf 和 cin 混用可能就会造成一些奇怪的错误

4、Java 相关：Java 整体效率大概比 C/C++慢 2~3 倍，但是 Java 写编程题也没什么问题，主要就是处理好各种输入输出的情况。

5、python 等等其他语言，做编程题真心不建议使用这些语言，要么效率低下，要么会有些更深的坑。

关于输出格式

格式问题经常令人抓狂，其实主要都有几个常见的坑

1、行末空格：比如我输出需要打印多个数需要使用空格分隔的时候，我们循环使用 printf("%d ",x);这种会很方便，但是这样会导致行末多一个空格，后台系统会严格比对你的输出和.out 文件，这样也会被判错误

2、换行问题，对于每个样例，建议输出完全之后都换行一下。对于一些题目，可能就是不换行就导致了后面输入数据错位，那就肯定不可能过了。

关于时间复杂度分析：

通常来说一般的系统 1s 能跑的算法量级是不足 1e8 的，所以做题的时候评估算法效率很重要，直接判断你的做法能否通过，当然这是以 C/C++为标准的，其他语言自己乘个时间倍数。。

举个例子，比如题目 n = 1e5，那么我就可以很敏感的知道我的算法需要一个 O(n) 或者 O(nlogn)。平方复杂度直接拜拜！

最后关于 "我本地能通过，交上去就是不对"

这个问题很蠢！通不过就是有一些问题。一个是要累积经验，分析到底可能出现的问题在哪里。另外不要使用一些奇怪的函数和行为。之前有见过有人使用了 windows 和 linux 平台那个功能的函数名都不一样的奇葩函数。 如果你使用 C/C++，最好别使用 VS 来写算法 code，这个默认是 MS 的，一般 OJ 上面编译器都不会是这个鬼。

1.2.3 春招备战资源总结

1、简历攻略

相信 2020 届的同学对简历准备都有一些困惑，不知道该写什么不该写什么，下面帮大家总结一下。

完整的简历需包括：**基本信息+实习经历+项目经历+校园经历+掌握技能**。

• 基本信息：

个人信息：姓名+手机号+邮箱地址

该部分需在简历中显著的标识出来，HR 每天要看很多很多简历，需要一眼可以看见你的联系方式。

简历照片的展示上，建议大家放一张干净大方精神的照片，可以增加 HR 对你简历的印象。注意请一定要去认真的拍一张证件照，并且绝对不可以放自拍。

学校学历：你的**毕业院校+你的学历**，如果是本科生，写本科院校，如果是研究生，需写上你的本科毕业院校+研究生毕业院校。

这部分很重要，硬性条件，有些公司部分岗位会对学校学历有额外要求。

相信这时候一部分同学会说："我的毕业院校不够好，怎么办？"说实话，说学校对找工作没有影响，一定在说笑。但这也并不是能对你一锤定音的指标，在毕业院校不是特别好

的情况下，请一定在简历上充分表现你的**实习经历+项目经历**！这一点在接下来介绍实习经历&项目经历的时候，会再给大家详细介绍！

加分项：个人博客。如果是非常优秀的博客一定要写上，对自己绝对是一个加分项。如果没有，现在开始写也来得及，养成一个好习惯，写的有条理一些。但是前提是对其他基本方面没有问题了，可以额外提升一下这一块。尤其是大二大三的同学，这对将来的校招会有很大的帮助。

• 实习经历：

实习经历是简历中的重中之重！如果你有**大厂实习经历**，绝对是秋招时扣响名企大门的一块最有力的敲门砖。

因为大厂的实习经历代表着你曾被大厂认可过，且具有相应的技能和一定的工作经验，用人单位会很欢迎这样的学生！

不过没有实习经历也没有关系，因为咱们毕竟还只是去应聘实习生，况且还有项目经历可以拉你一把！

• 项目经历：

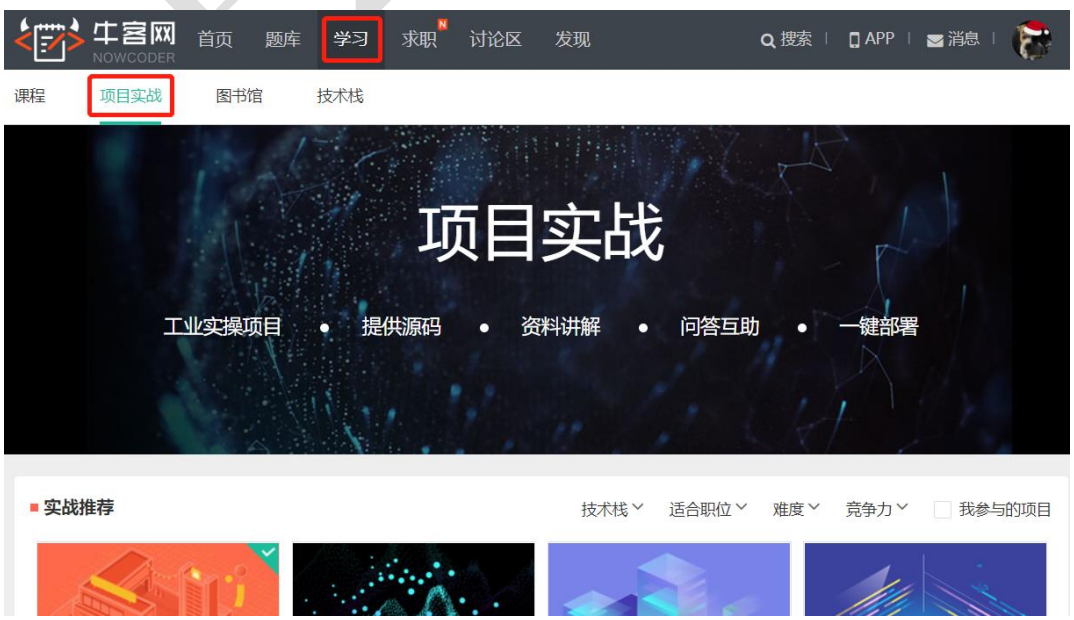
项目经历也是 HR 和面试官会非常看重的一部分！因为项目经历代表着你可能了解的技术栈，好的项目经历能够帮助你渡过简历筛选这一关。

且面试时面试官可能会针对你的项目提问，如果你确实认认真真做过项目、熟悉其中的技术难点与技术亮点，你甚至可以引导面试官向你熟悉且擅长的方面提问。

相比之实习经历，项目经历可以说是比较容易在短期内准备的东西了。

如果你还没有做过什么项目并且基础还很薄弱，建议先来**牛客-学习-项目实践，这里有从初级到高级的项目教程，从配置环境到项目实现，一步步详细讲解，并且全部免费。**

项目平台地址：https://www.nowcoder.com/project/recommend



如果你有基础，但是还缺少一个像样的项目，那么建议你来报名一下牛客网中级项目课。牛客网中级项目课是由牛客网 CEO 叶神叶向宇——10 多年一线编程老司机——亲自手把手带你做项目，真正从企业实战角度带你从 0 到 1 搭建项目！

牛客网中级项目课：https://www.nowcoder.com/courses/semester/medium

• 校园经历

如果你作为第一次出门找实习的实习生，没有实习经历，也没有项目经历，可以在这里表现一下自己！

可以写一些你的校园经历，比如你在学生会、社团中担任过什么要职，举办过哪些活动等。主要是体现一下自己的学习能力和合作/组织能力等，注意在书写校园经历的时候，也要体现好这个活动大致是什么内容，你在其中扮演了什么角色。尽量挑自己挑大梁的活动写在简历上。

另外在这一部分可以列举一些你获得的奖学金。

• 掌握技能：

应聘哪个岗位相关技能一定要具备，不然连基本资格都不符合，如何能通过简历呢。

注意“熟练掌握”和“精通”的区别，注意词汇上的应用。防止过于夸大而适得其反，毕竟面试的时候，面试官跟你交流的谈资就是你的简历，简历上写的所有内容，都有可能会被问到。简历制作切记：可以美化，但是不可以撒谎！

2、笔试备考

（1）日常刷题

技术岗位一般都是需要笔试的，而笔试唯一的捷径就是：刷题！刷题！有意义的刷题！

• 企业校招真题

每个公司的笔试都是不同的，侧重知识点都是不一样的，考生可以直接去牛客题库>公司真题模考>职业方向。该题库包含众多企业技术岗位笔试试卷和解析，可直接进行原卷练习。（地址：https://www.nowcoder.com/contestRoom）

• 在线编程题库

《2018 校招真题编程题汇总》

《剑指 offer》

（2）校招全国统一模拟笔试

除了日常刷题，考前进行模拟测试也是很重要的，可以提前熟悉流程和环境，还能定期对自己的水平进行全面的评估，从而查漏补缺，更快的提高能力。

校招全国统一模拟笔试

地址：牛客网>公司真题模考>（左侧位置）IT 校招全国统一模拟笔试

时间：3 月至 7 月，每月一场
收获：
- 求职竞争力报告
- 牛币奖励
- 全真校招笔试流程体验
- 企业校招内推机会

3、面试备考

面试环节是所有招聘环节中至关重要的一项，考察也会更全面更严格。而提高自己面试应对能力的方法就是：看面经。

面经是学长学姐的亲身经历和总结，参考价值之重无需多说。

推荐大家关注“面经大全”和“牛客论坛”小程序，这里面有**上千篇高品质面经汇总**以及众多行内人士信息分享，助你了解心仪公司/岗位的面试真题，面试官套路各个击破。

扫一扫，面经装进口袋


看一看，关注行内新态

感谢腾讯、网易、京东、百度、招商银行信用卡中心、字节跳动、美团点评、小米、搜狐、搜狐畅游、吉比特、拼多多、欢聚时代、美丽联合、七牛云等 15 家企业对于本次《2019 名企校招笔试真题精选》活动的大力支持。

下面将对以上企业笔试真题进行详细解析。

二、腾讯 2018 秋招笔试真题

1、小 Q 的歌单

【题目描述】小 Q 有 X 首长度为 A 的不同的歌和 Y 首长度为 B 的不同的歌，现在小 Q 想用这些歌组成一个总长度正好为 K 的歌单，每首歌最多只能在歌单中出现一次，在不考虑歌单内歌曲的先后顺序的情况下，请问有多少种组成歌单的方法。

输入描述：

每个输入包含一个测试用例。

每个测试用例的第一行包含一个整数，表示歌单的总长度 K(1<=K<=1000)。

接下来的一行包含四个正整数，分别表示歌的第一种长度 A(A<=10)和数量 X(X<=100)以及歌的第二种长度 B(B<=10)和数量 Y(Y<=100)。保证 A 不等于 B。

输出描述：

输出一个整数,表示组成歌单的方法取模。因为答案可能会很大,输出对 1000000007 取模的结果。

输入示例：

5

2 3 3 3

输出示例：

9

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
long long c[105][105];
const int mod = 1000000007;
void init() {
    c[0][0] = 1;
    for(int i = 1; i <= 100; i++) {
        c[i][0] = 1;
        for(int j = 1; j <= 100; j++)
            c[i][j] = (c[i - 1][j - 1] + c[i - 1][j]) % mod;
    }
}
int main() {
    int k, a, b, x, y;
    long long ans = 0;
    init();
    scanf("%d", &k);
    scanf("%d%d%d%d", &a, &x, &b, &y);
    for(int i = 0; i <= x; i++) {
        if(i * a <= k && (k - a * i) % b == 0 && (k - a * i) / b <= y)
```

```
            ans=(ans + (c[x][i] * c[y][(k - a * i) / b]) % mod) % mod;
    }
    printf("%lld\n", ans);
    return 0;
}
```

2、安排机器

【题目描述】小 Q 的公司最近接到 m 个任务，第 i 个任务需要 xi 的时间去完成，难度等级为 yi。
小 Q 拥有 n 台机器，每台机器最长工作时间 zi，机器等级 wi。
对于一个任务，它只能交由一台机器来完成，如果安排给它的机器的最长工作时间小于任务需要的时间，则不能完成，如果完成这个任务将获得 200 * xi + 3 * yi 收益。
对于一台机器，它一天只能完成一个任务，如果它的机器等级小于安排给它的任务难度等级，则不能完成。
小 Q 想在今天尽可能的去完成任务，即完成的任务数量最大。如果有多种安排方案，小 Q 还想找到收益最大的那个方案。小 Q 需要你来帮助他计算一下。

输入描述：

输入包括 N + M + 1 行，
输入的第一行为两个正整数 n 和 m(1 <= n, m <= 100000)，表示机器的数量和任务的数量。
接下来 n 行，每行两个整数 zi 和 wi(0 < zi < 1000, 0 <= wi <= 100)，表示每台机器的最大工作时间和机器等级。
接下来的 m 行，每行两个整数 xi 和 yi(0 < xi < 1000, 0 <= yi<= 100)，表示每个任务需要的完成时间和任务的难度等级。

输出描述：

输出两个整数，分别表示最大能完成的任务数量和获取的收益。

输入示例：

```
1 2
100 3
100 2
100 1
```

输出示例：

```
1 20006
```

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
#define LL long long
const int maxn = 1e5 + 10;
struct node {
    int x, y;
}e[maxn], f[maxn];
```

```
int cnt[105];
int cmp(node a, node b) {
    if(a.x == b.x)return a.y > b.y;
    return a.x > b.x;
}
int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    for(int i = 0; i < n; i++) scanf("%d%d", &e[i].x, &e[i].y);
    for(int i = 0; i < m; i++) scanf("%d%d", &f[i].x, &f[i].y);
    sort(e, e + n, cmp);
    sort(f, f + m, cmp);
    int num = 0;
    LL ans = 0;
    memset(cnt, 0, sizeof(cnt));
    int i, j, k;
    for(i = 0, j = 0; i < m; i++) {
        while(j < n && e[j].x >= f[i].x) {
            cnt[e[j].y]++;
            j++;
        }
        for(k = f[i].y; k <= 100; k++) {
            if(cnt[k]) {
                num++;
                cnt[k]--;
                ans = ans + 200 * f[i].x + 3 * f[i].y;
                break;
            }
        }
    }
    printf("%d %lld\n", num, ans);
    return 0;
}
```

3、贪吃的小 Q

【题目描述】小 Q 的父母要出差 N 天，走之前给小 Q 留下了 M 块巧克力。小 Q 决定每天吃的巧克力数量不少于前一天吃的一半，但是他又不想在父母回来之前的某一天没有巧克力吃，请问他第一天最多能吃多少块巧克力

输入描述：

每个输入包含一个测试用例。

每个测试用例的第一行包含两个正整数，表示父母出差的天数 N(N<=50000)和巧克力的数量 M(N<=M<=100000)。

输出描述：

输出一个数表示小 Q 第一天最多能吃多少块巧克力。

输入示例：

3 7

输出示例：

4

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n, m, temp, now, mid;
    bool ok;
    scanf("%d%d", &n, &m);
    int l = 1, r = m;
    while(l != r) {
        ok = 1;
        mid = (l + r + 1) / 2;
        temp = m;
        now = mid;
        for(int j = 0; j < n; j++) {
            if(temp < now) {
                ok = 0;
                break;
            }
            temp -= now;
            now = (now + 1) / 2;
        }
        if(ok)
            l = mid;
        else
            r = mid - 1;
    }
    printf("%d\n", l);
    return 0;
}
```

4、纸牌游戏

【题目描述】牛牛和羊羊正在玩一个纸牌游戏。这个游戏一共有 n 张纸牌，第 i 张纸牌上写着数字 ai。
牛牛和羊羊轮流抽牌，牛牛先抽，每次抽牌他们可以从纸牌堆中任意选择一张抽出，直到纸牌被抽完。
他们的得分等于他们抽到的纸牌数字总和。
现在假设牛牛和羊羊都采用最优策略，请你计算出游戏结束后牛牛得分减去羊羊得分等于多少。

输入描述:

输入包括两行。
第一行包括一个正整数 n(1 <= n <= 105),表示纸牌的数量。
第二行包括 n 个正整数 ai(1 <= ai <= 109),表示每张纸牌上的数字。

输出描述:

输出一个整数，表示游戏结束后牛牛得分减去羊羊得分等于多少。

输入示例：

3
2 7 4

输出示例：

5

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 10;
int a[maxn];
int n;
int cmp(int x, int y) {
    return x > y;
}
int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    sort(a, a + n, cmp);
    int ans = 0;
    for(int i = 0; i < n; i++) {
        if(i % 2 == 0) {
            ans += a[i];
        } else {
            ans -= a[i];
        }
    }
    cout << ans << endl;
```

```
    return 0;
}
```

三、京东 2018 秋招笔试真题

1、求幂

【题目描述】东东对幂运算很感兴趣,在学习的过程中东东发现了一些有趣的性质: 9^3 = 27^2, 2^10 = 32^2

东东对这个性质充满了好奇,东东现在给出一个整数 n,希望你能帮助他求出满足 a^b = c^d(1 ≤ a,b,c,d ≤ n)的式子有多少个。

例如当 n = 2: 1^1=1^1

1^1=1^2

1^2=1^1

1^2=1^2

2^1=2^1

2^2=2^2

一共有 6 个满足要求的式子

输入描述:

输入包括一个整数 n(1 ≤ n ≤ 10^6)

输出描述:

输出一个整数,表示满足要求的式子个数。因为答案可能很大,输出对 1000000007 求模的结果

输入示例:

2

输出示例:

6

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;
set<int> S;
int n;
int main() {
    cin >> n;
    int res = 1LL * n * (n * 2 - 1) % mod;
    for(int i = 2; i * i <= n; i++) {
        if(S.find(i) != S.end()) continue;
        long long tmp = i;
```

```
        int cnt = 0;
        while(tmp <= n) {
            S.insert(tmp);
            tmp = tmp * i;
            cnt++;
        }
        for(int i = 1; i <= cnt; i++) {
            for(int j = i + 1; j <= cnt; j++) {
                res = (res + n / (j / __gcd(i, j) ) * 2LL ) % mod;
            }
        }
    }
    cout << res << endl;
}
```

2、神奇数

【题目描述】东东在一本古籍上看到有一种神奇数,如果能够将一个数的数字分成两组,其中一组数字的和等于另一组数字的和,我们就将这个数称为神奇数。例如 242 就是一个神奇数,我们能够将这个数的数字分成两组,分别是{2,2}以及{4},而且这两组数的和都是 4.东东现在需要统计给定区间中有多少个神奇数,即给定区间[l, r],统计这个区间中有多少个神奇数,请你来帮助他。

输入描述:

输入包括一行,一行中两个整数 l 和 r(1 ≤ l, r ≤ 10^9, 0 ≤ r - l ≤ 10^6),以空格分割

输出描述:

输出一个整数,即区间内的神奇数个数

输入示例:

1 50

输出示例:

4

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
bool check(int n) {
    char s[11];
    int cur = 0, t = 0;
    while(n > 0) {
        s[cur] = n % 10;
        t += s[cur++];
        n /= 10;
```

```
    }
    if(t % 2) return false;
    t /= 2;
    bool ok[42] = {0};
    ok[s[0]] = true;
    for(int i = 1; i < cur; i++) {
        int v = s[i];
        for(int j = 41; j >= 0; j--) {
            if(ok[j] && j + v <= 41) {
                ok[j + v] = true;
            }
        }
        if(ok[t]) {
            return true;
        }
    }
    return false;
}
int l, r;
int main() {
    int res = 0;
    cin >> l >> r;
    for(int i = l; i <= r; i++) {
        if(check(i)) res++;
    }
    cout << res << endl;
    return 0;
}
```

3、整除

【题目描述】牛牛对整除非常感兴趣。牛牛的老师给他布置了一道题:牛牛的老师给出一个 n,然后牛牛需要回答出能被 1 到 n 之间(包括 1 和 n)所有整数整除的最小的数。牛牛犯了难,希望你能编程帮他解决这个问题。

输入描述:

输入包括一个整数 n(1 ≤ n ≤ 100000)

输出描述:

输出一个整数，即满足要求的最小整数。答案可能很大,请输出这个整数对于 987654321 取模的结果

输入示例:

3

输出示例:

6

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100000 + 5;
int tmp[maxn];
int n;
int main() {
    cin >> n;
    for(int i = 1; i <= n; i++) {
        int k = i;
        for(int j = 2; j * j <= n; j++) {
            int s = 0;
            while(k % j == 0) {
                s++;
                k /= j;
            }
            tmp[j] = max(tmp[j], s);
        }
        if(k > 1) tmp[k] = max(tmp[k], 1);
    }
    long long res = 1;
    for(int i = 1; i <= 100000; i++) {
        for(int j = 0; j < tmp[i]; j++) {
            res = res * i % 987654321;
        }
    }
    cout << res << endl;
    return 0;
}
```

四、百度 2018 秋招笔试真题

1、字符串匹配

【题目描述】牛牛有两个字符串 A 和 B,其中 A 串是一个 01 串,B 串中除了可能有 0 和 1,还可能有'?',B 中的'?'可以确定为 0 或者 1。 寻找一个字符串 T 是否在字符串 S 中出现的过程,称为字符串匹配。牛牛现在

考虑所有可能的字符串 B,有多少种可以在字符串 A 中完成匹配。

例如:A = "00010001", B = "??"

字符串 B 可能的字符串是"00","01","10","11",只有"11"没有出现在字符串 A 中,所以输出 3

输入描述:

输入包括两行,第一行一个字符串 A,字符串 A 长度 length(1 ≤ length ≤ 50),A 中每个字符都是'0'或者'1'。

第二行一个字符串 B,字符串 B 长度 length(1 ≤ length ≤ 50),B 中的字符包括'0','1'和'?'。

输出描述:

输出一个整数,表示能完成匹配的字符串种数。

输入示例:

00010001

??

输出示例:

3

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
string A, B;
set<string> s;
int main() {
    cin >> A >> B;
    int ans = 0;
    for(int i = 0; i < (int)A.size(); i++) {
        int j = i + B.size() - 1;
        if(j >= A.size()) continue;
        string cur = A.substr(i, j - i + 1);
        if(s.count(cur)) continue;
        s.insert(cur);
        bool flag = true;
        for(int k = 0; k < B.size(); k++) {
            if(cur[k] != B[k]) {
                if(B[k] == '?') continue;
                else {
                    flag = false;
                    break;
                }
            }
        }
        if(flag) ans++;
    }
```

```
    cout << ans << endl;
    return 0;
}
```

2、完成括号匹配

【题目描述】合法的括号匹配序列被定义为:

1. 空串""是合法的括号序列
2. 如果"X"和"Y"是合法的序列,那么"XY"也是一个合法的括号序列
3. 如果"X"是一个合法的序列,那么"[X]"也是一个合法的括号序列
4. 每个合法的括号序列都可以由上面的规则生成

例如"", "[]", "[][]", "[[]]", "[[[[]]]]"都是合法的。

牛牛现在给出一个括号序列 s,牛牛允许你执行的操作是:在 s 的开始和结尾处添加一定数量的左括号('[')或者右括号(']')使其变为一个合法的括号匹配序列。牛牛希望你能求出添加最少的括号之后的合法的括号匹配序列是什么。

输入描述:

输入包括一个字符串 s,s 的长度 length(1 ≤ length ≤ 50),s 中只包含'['和']'。

输出描述:

输出一个字符串,表示括号完全匹配的序列。

输入示例:

```
][
```

输出示例:

```
[][]
```

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
string s;
int main() {
    cin >> s;
    string res = s;
    string append;
    int cnt = 0;
    for(int i = 0; i < s.size(); i++) {
        if(s[i] == '[') cnt++;
        else cnt--;
        if(cnt < 0) {
            cnt++;
            append += "[";
        }
```

```
    }
    cout << append + s + string(cnt, ']') << endl;
    return 0;
}
```

3、字符覆盖

【题目描述】小度有一个小写字母组成的字符串 s. 字符串 s 已经被写在墙上了.
小度还有很多卡片,每个卡片上有一个小写字母,组成一个字符串 t。小度可以选择字符串 t 中任意一个字符,然后覆盖在字符串 s 的一个字符之上。小度想知道在选取一些卡片覆盖 s 的一些字符之后,可以得到的字典序最大的字符串是什么。

输入描述:

输入包括两行,第一行一个字符串 s,字符串 s 长度 length(1 ≤ length ≤ 50),s 中每个字符都是小写字母
第二行一个字符串 t,字符串 t 长度 length(1 ≤ length ≤ 50),t 中每个字符都是小写字母

输出描述:

输出一个字符串,即可以得到的字典序最大字符串

输入示例:

fedcba
ee

输出示例:

feeeba

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
string s, t;
int cmp(int a, int b) {
    return a > b;
}
int main() {
    cin >> s >> t;
    sort(t.begin(), t.end(), cmp);
    int pos = 0;
    for(int i = 0; i < s.size(); i++) {
        if(s[i] < t[pos]) {
            s[i] = t[pos];
            pos++;
        }
    }
```

```
    cout << s << endl;
    return 0;
}
```

4、最大子序列

【题目描述】对于字符串 x 和 y， 如果擦除 x 中的某些字母(有可能全擦掉或者都不擦)能够得到 y,我们就称 y 是 x 的子序列。例如."ncd"是"nowcoder"的子序列,而"xt"不是。

现在对于给定的一个字符串 s,请计算出字典序最大的 s 的子序列。

输入描述:

输入包括一行,一个字符串 s,字符串 s 长度 length(1 ≤ length ≤ 50).

s 中每个字符都是小写字母

输出描述:

输出一个字符串,即字典序最大的 s 的子序列。

输入示例:

test

输出示例:

tt

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
string s;
int main() {
    cin >> s;
    ostringstream ss;
    while(!s.empty()) {
        string::iterator it = max_element(s.begin(), s.end());
        ss << *it;
        s.erase(s.begin(), it + 1);
    }
    cout << ss.str() << endl;
    return 0;
}
```

五、网易 2018 秋招笔试真题

1、字符串碎片

【题目描述】一个由小写字母组成的字符串可以看成一些同一字母的最大碎片组成的。例如,"aaabbaaac"

是由下面碎片组成的:'aaa','bb','c'。牛牛现在给定一个字符串,请你帮助计算这个字符串的所有碎片的平均长度是多少。

输入描述:

输入包括一个字符串 s,字符串 s 的长度 length(1 ≤ length ≤ 50),s 只含小写字母('a'-'z')

输出描述:

输出一个整数,表示所有碎片的平均长度,四舍五入保留两位小数。

如样例所示: s = "aaabbaaac"

所有碎片的平均长度 = (3 + 2 + 3 + 1) / 4 = 2.25

输入例子 1:

aaabbaaac

输出例子 1:

2.25

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
string s;
int main() {
    cin >> s;
    char c = s[0];
    double n = 1, d;
    for(int i = 1; i < s.size(); i++) {
        if(c != s[i]) {
            c = s[i];
            n++;
        }
    }
    d = (double)s.size() / n;
    printf("%.2lf\n", d);
    return 0;
}
```

2、游历魔法王国

【题目描述】魔法王国一共有 n 个城市,编号为 0~n-1 号,n 个城市之间的道路连接起来恰好构成一棵树。小易现在在 0 号城市,每次行动小易会从当前所在的城市走到与其相邻的一个城市,小易最多能行动 L 次。如果小易到达过某个城市就视为小易游历过这个城市了,小易现在要制定好的旅游计划使他能游历最多的城市,请你帮他计算一下他最多能游历过多少个城市(注意 0 号城市已经游历了,游历过的城市不重复计算)。

输入描述:

输入包括两行,第一行包括两个正整数 n(2 ≤ n ≤ 50)和 L(1 ≤ L ≤ 100),表示城市个数和小易能行动

的次数。

第二行包括 n-1 个整数 parent[i](0 ≤ parent[i] ≤ i)，对于每个合法的 i(0 ≤ i ≤ n - 2),在(i+1)号城市和 parent[i]间有一条道路连接。

输出描述:

输出一个整数,表示小易最多能游历的城市数量。

输入例子 1:

5 2

0 1 2 3

输出例子 1:

3

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 50 + 5;
int n, L;
int parent[maxn];
int dp[200];
int main() {
    scanf("%d%d", &n, &L);
    for(int i = 0; i < n - 1; i++) scanf("%d", &parent[i]);
    int mx = 0;
    for(int i = 0; i < n - 1; i++) {
        dp[i + 1] = dp[parent[i]] + 1;
        mx = max(mx, dp[i + 1]);
    }
    int d = min(L, mx);
    cout << min((n), 1 + d + (L - d) / 2);
    return 0;
}
```

3、重排数列

【题目描述】小易有一个长度为 N 的正整数数列 A = {A[1], A[2], A[3]..., A[N]}。

牛博士给小易出了一个难题:

对数列 A 进行重新排列,使数列 A 满足所有的 A[i] * A[i + 1](1 ≤ i ≤ N - 1)都是 4 的倍数。

小易现在需要判断一个数列是否可以重排之后满足牛博士的要求。

输入描述:

输入的第一行为数列的个数 t(1 ≤ t ≤ 10),

接下来每两行描述一个数列 A,第一行为数列长度 n(1 ≤ n ≤ 10^5)

第二行为 n 个正整数 A[i](1 ≤ A[i] ≤ 10^9)

输出描述:

对于每个数列输出一行表示是否可以满足牛博士要求,如果可以输出 Yes,否则输出 No。

输入例子 1:

2
3
1 10 100
4
1 2 3 4

输出例子 1:

Yes
No

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
int n;
int main() {
    int t;
    scanf("%d", &t);
    while(t--) {
        scanf("%d", &n);
        int cnt4 = 0;
        int cnt2 = 0;
        int cnt1 = 0;
        for(int i = 0; i < n; i++) {
            int x;
            scanf("%d", &x);
            if(x % 4 == 0) cnt4++;
            else if(x % 2 == 0) cnt2++;
            else cnt1++;
        }
        if(cnt2 == 0) {
            if(cnt4 >= cnt1 - 1)
                printf("Yes\n");
            else
                printf("No\n");
        } else {
            if(cnt4 >= cnt1)
                printf("Yes\n");
            else
```

```
                printf("No\n");
            }
        }
        return 0;
}
```

4、射击游戏

【题目描述】小易正在玩一款新出的射击游戏,这个射击游戏在一个二维平面进行,小易在坐标原点(0,0),平面上有 n 只怪物,每个怪物有所在的坐标(x[i], y[i])。小易进行一次射击会把 x 轴和 y 轴上(包含坐标原点)的怪物一次性消灭。

小易是这个游戏的 VIP 玩家,他拥有两项特权操作:

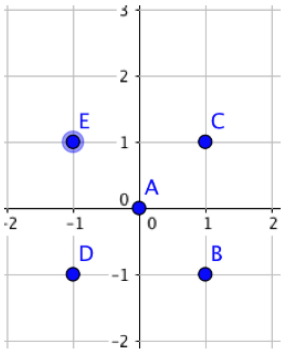
1、让平面内的所有怪物同时向任意同一方向移动任意同一距离

2、让平面内的所有怪物同时对于小易(0,0)旋转任意同一角度

小易要进行一次射击。小易在进行射击前,可以使用这两项特权操作任意次。

小易想知道在他射击的时候最多可以同时消灭多少只怪物,请你帮帮小易。

如样例所示:



所有点对于坐标原点(0,0)顺时针或者逆时针旋转 45°,可以让所有点都在坐标轴上,所以 5 个怪物都可以消灭。

输入描述:

输入包括三行。

第一行中有一个正整数 n(1 ≤ n ≤ 50),表示平面内的怪物数量。

第二行包括 n 个整数 x[i](-1,000,000 ≤ x[i] ≤ 1,000,000),表示每只怪物所在坐标的横坐标,以空格分割。

第二行包括 n 个整数 y[i](-1,000,000 ≤ y[i] ≤ 1,000,000),表示每只怪物所在坐标的纵坐标,以空格分割。

输出描述:

输出一个整数表示小易最多能消灭多少只怪物。

输入例子 1:

5

0 -1 1 1 -1

0 -1 -1 1 1

输出例子 1:

5

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 50 + 5;
int x[maxn], y[maxn];
int n;
int solve() {
    if(n <= 2) return n;
    int res = 1;
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < n; j++) {
            if(i != j) {
                int dx1 = x[j] - x[i];
                int dy1 = y[j] - y[i];
                for(int k = 0; k < n; k++) {
                    int cnt = 0;
                    if(i != k && j != k) {
                        for(int r = 0; r < n; r++) {
                            int dx2 = x[r] - x[i];
                            int dy2 = y[r] - y[i];
                            if(dy1 * dx2 == dy2 * dx1) {
                                cnt++;
                            } else {
                                dx2 = x[r] - x[k];
                                dy2 = y[r] - y[k];
                                if(dy1 * dy2 == -dx2 * dx1) {
                                    cnt++;
                                }
                            }
                        }
                    }
                    res = max(res, cnt);
                }
            }
        }
    }
    return res;
```

```
}
int main() {
    cin >> n;
    for(int i = 0; i < n; i++) cin >> x[i];
    for(int i = 0; i < n; i++) cin >> y[i];
    cout << solve() << endl;
}
```

六、招商银行信用卡中心 2018 秋招笔试真题

1、字符串是否由子串拼接

【题目描述】给出一个非空的字符串，判断这个字符串是否是由它的一个子串进行多次首尾拼接构成的。例如，"abcabcabc"满足条件，因为它是由"abc"首尾拼接而成的，而"abcab"则不满足条件。

输入描述:

非空字符串

输出描述:

如果字符串满足上述条件，则输出最长的满足条件的的子串；如果不满足条件，则输出 false。

输入示例:

abcabc

输出示例:

Abc

【答案及解析】

```
import java.util.*;
public class Main {
    public Scanner cin = new Scanner(System.in);
    Main() {
        while (cin.hasNext()) {
            String str = cin.next();
            int ans = str.length() - 1;
            for (int i = 0; ans > 0; ans--) {
                for (i = 0; str.length() % ans == 0 && i < str.length()
                        && str.charAt(i) == str.charAt(i % ans); i++) {
                }
                if (i == str.length())
                    break;
            }
            System.out.println(ans != 0 ? str.substring(0, ans) : "false");
        }
```

```
    }
    public static void main(String[] args) {
        new Main();
    }
}
```

2、寻找合法字符串

【题目描述】给出一个正整数 n，请给出所有的包含 n 个'('和 n 个')'的字符串，使得'('和')'可以完全匹配。

例如：

'(())()'，'()()()' 都是合法的；

'())()('是不合法的。

请按照__字典序__给出所有合法的字符串。

输入描述：

输入为 1 个正整数

输出描述：

输出为所有合法的字符串，用英文逗号隔开

输入示例：

2

输出示例：

(()),()()

【答案及解析】

```
class Solution {
public:
    vector<string> generateParenthesis(int n) {
        this->n = n;
        vector<string> ret;
        dfs(0, 0, 0, "", ret);
        return ret;
    }
private:
    int n;
    void dfs(int x, int l, int r, string str, vector<string> &ret)
    {
        if (x == 2 * n) {
            ret.push_back(str);
            return ;
        }
```

```
        if (l < n)
            dfs(x + 1, l + 1, r, str + "(", ret);
        if (r < n && l > r)
            dfs(x + 1, l, r + 1, str + ")", ret);
        }
};
```

3、序列找数

【题目描述】从非负整数序列 0，1，2，...，n 中给出包含其中 n 个数的子序列，请找出未出现在该子序列中的那个数。

输入描述:

输入为 n+1 个非负整数，用空格分开。

其中：首个数字为非负整数序列的最大值 n，后面 n 个数字为子序列中包含的数字。

输出描述:

输出为 1 个数字，即未出现在子序列中的那个数。

输入示例;

3 3 0 1

输出示例;

2

【答案及解析】

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        if (n >= 0){
            StringBuilder b = new StringBuilder();
            for (int i = 0; i < n; i++){
                b.append(sc.nextInt());
            }
            findNoNumber(b.toString());
        }
    }
    /**
     * 找到未出现在该子序列中的数
     * @param s
     */
```

```
    private static void findNoNumber(String s) {
        if (s == null || s.length() == 0)
            return;
        for (int i = 0; i <= s.length(); i++){
            if (!s.contains(" "+i+" ")){
                System.out.println(i);
                return;
            }
        }
    }
}
```

七、字节跳动 2018 秋招笔试真题

1、拼硬币

【题目描述】现有 n1+n2 种面值的硬币，其中前 n1 种为普通币，可以取任意枚，后 n2 种为纪念币，每种最多只能取一枚，每种硬币有一个面值，问能用多少种方法拼出 m 的面值？

输入描述:

第一行三个整数 n1, n2, m，分别表示普通币种类数，纪念币种类数和目标面值

第二行 n1 个整数，第 i 种普通币的面值 a[i]。保证 a[i]为严格升序。

第三行 n2 个整数，第 i 种纪念币的面值 b[i]。保证 b[i]为严格升序。

对于 30%的测试，保证 1<=n1+n2<=10，1<=m<=100，1<=a[i]<=100 1<=b[i]<=100

对于 100%的测试，保证 1<=n1+n2<=100, 1<=m<=100000, 1<=a[i]<=100000 1<=b[i]<=100000

输出描述:

输出一行，包含一个数字 x，表示方法总数对 1000000007（1e9+7）取模的结果。

注意：不要忘记取模！

备注:

两个方法，它们任意一种或以上的硬币数量不同，则认为是两种拼法。

输入示例:

3 1 5

1 2 3

1

输出示例:

9

说明:

(x)代表面值为 x 的普通币，[x]代表面值为 x 的纪念币，样例所有方法数如下：

(1) (1) (1) (1) (1)

(1) (1) (1) (2)

(1)(1)(3)

(1)(2)(2)

(2)(3)

(1)(1)(1)(1)[1]

(1)(1)[1](2)

(1)[1](3)

[1](2)(2)

【答案及解析】

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 110, M = 100010, MOD = 1000000007;
int n1, n2, m;
int a[N], b[N], f[M];
//典型的背包问题，普通币是完全背包问题，纪念币是 0/1 背包问题。
//两种背包问题做法类似，都是先枚举物品，再枚举容量，不同点在于完全背包问题要从小到大枚举容量，0/1 背包问题要从大到小枚举容量。
int main()
{
cin >> n1 >> n2 >> m;
for (int i = 0; i < n1; i++) cin >> a[i];
for (int i = 0; i < n2; i++) cin >> b[i];
f[0] = 1;
for (int i = 0; i < n1; i++)
for (int j = a[i]; j <= m; j++)
{
       f[j] += f[j - a[i]];
       if (f[j] >= MOD) f[j] -= MOD;
    }
for (int i = 0; i < n2; i++)
for (int j = m; j >= b[i]; j--)
{
       f[j] += f[j - b[i]];
       if (f[j] >= MOD) f[j] -= MOD;
}
cout << f[m] << endl;
return 0;
```

```
}
```

2、矩形游戏

【题目描述】小 a 在玩一个很简单的游戏，游戏的内容是控制一个小人在一块矩形的空地内走，一旦小人走出矩阵范围，游戏就失败。游戏机有上，下，左，右四个按键，每按一下小人就向相应的方向走一步。这个游戏过于简单，小 a 说：“这种游戏我闭着眼睛玩都输不了”。于是他便闭上眼睛,进行一连串的操作。但若他中途输了的话就会停止。

那么问题来了:给定小 a 的操作，进行 Q 次询问，你能算出每次询问小人能走多少步吗？

输入描述:

第一行为长度 L 的字符串 S，每个字符依次代表小 a 的一次操作。'u'代表向上，'d'代表向下，'l'代表向左，'r'代表向右。字符串 S 不会包含其他字符。

第二行是整数 Q，代表 Q 次询问

接下来 Q 行，每行有四个整数，N，M，X，Y，保证 1<=X<=N，1<=Y<=M，矩阵大小为 N*M，小人初始位置为(X，Y)。

对于 30%的测试，0<X<=1000，0<Y<=1000，0<L<=1000，0<Q<=1000

对于 100%的测试，0<X<=100000(1e5)，0<Y<=100000(1e5)，0<L<=100000(1e5)，0<Q<=30000(3e4)

输出描述:

每次询问要求你打印一个整数 s（单独一行），代表小人所走的步数。

备注:

小人踏出矩阵外的那一步也要算入结果哦

输入示例:

uuurrdddddl

3

5 6 3 3

5 6 4 2

6 6 4 2

输出示例:

3

10

11

说明:

上下左右关系以下图为例，一个 N=3，M=2 的矩阵是这样的：

(1,1)(1,2)

(2,1)(2,2)

(3,1)(3,2)

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
```

```
const int N = 100010;
string ops;
int n, m, x, y;
int l[N], r[N], u[N], d[N];
int ln[N], rn[N], un[N], dn[N];
bool check(int mid)
{
return l[mid] <= y - 1 && r[mid] <= m - y
    && d[mid] <= n - x && u[mid] <= x - 1;
}
void init(int a[], int b[])
{
for (int i = 1, j = 1; i <= ops.size(); i++)
{
    while (j <= ops.size() && a[j] < i) j++;
    b[i] = j;
}
}
int main()
{
cin >> ops;
for (int i = 0; i < ops.size(); i++)
{
    char op = ops[i];
    if (op == 'u') x--;
    else if (op == 'd') x++;
    else if (op == 'l') y--;
    else y++;
    l[i + 1] = max(l[i], -y);
    r[i + 1] = max(r[i], y);
    u[i + 1] = max(u[i], -x);
    d[i + 1] = max(d[i], x);
}
init(l, ln), init(r, rn), init(u, un), init(d, dn);
int Q;
cin >> Q;
while (Q--)
{
    cin >> n >> m >> x >> y;
    int t = ops.size();
```

```
        int sl = y > t ? t : ln[y];
        int sr = m - y + 1 > t ?  t : rn[m - y + 1];
        int su = x > t ? t : un[x];
        int sd = n - x + 1 > t ? t : dn[n - x + 1];
        cout << min(sl, min(sr, min(su, sd))) << endl;
    }
    return 0;
}
```

3、电容充电

【题目描述】有一台用电容组成的计算器，其中每个电容组件都有一个最大容量值（正整数）。
对于单个电容，有如下操作指令：
指令 1：放电操作 - 把该电容当前电量值清零
指令 2：充电操作 - 把该电容当前电量补充到其最大容量值
对于两个电容 A 和 B，有如下操作指令：
指令 3：转移操作 - 从 A 中尽可能多的将电量转移到 B，转移不会有电量损失，如果能够充满 B 的最大容量，那剩余的电量仍然会留在 A 中
现在已知有两个电容，其最大容量分别为 a 和 b，其初始状态都是电量值为 0，希望通过一系列的操作可以使其中某个电容（无所谓哪一个）中的电量值等于 c（c 也是正整数），这一系列操作所用的最少指令条数记为 M，如果无论如何操作，都不可能完成，则定义此时 M=0。
显然对于每一组确定的 a、b、c，一定会有一个 M 与其对应。

输入描述:

每组测试样本的输入格式为：
第一行是一个正整数 N
从第二行开始，每行都是 3 个正整数依次组成一组 a、b、c，一共有 N 组

输出描述:

输出为 N 行，每行打印每一组的对应的 M

备注:

数据范围：
N：0<N<100
a、b、c：
0<a、b、c<10^5（50%）
0<a、b、c<10^7（30%）
0<a、b、c<10^9（20%）

输入示例:

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

输出示例:

4

0

说明:

对于（3，4，2）,最少只需要 4 个指令即可完成：

（设最大容量为 3 的是 A 号电容，另一个是 B 号电容）

充电 A 转移 A->B 充电 A 转移 A->B

此时 A 中当前电量为 2，操作完成，所以输出 4。

对于（2，3，4），显然不可能完成，输出 0。

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
LL gcd(LL a, LL b, LL &x, LL &y)
{
if (b == 0)
{
    x = 1, y = 0;
    return a;
}
LL q = gcd(b, a % b, y, x);
y -= a / b * x;
return q;
}
int main()
{
int T;
cin >> T;
while (T--)
{
    LL a, b, c, x, y;
    cin >> a >> b >> c;
    int d = gcd(a, b, x, y);
    if (c > a && c > b || c % d)
    {
      cout << 0 << endl;
      continue;
    }
    if (c == a || c == b)
    {
      cout << 1 << endl;
```

```
        continue;
    }
    if (y > 0) swap(x, y), swap(a, b);
    LL a2 = a / d, b2 = b / d;
    x *= c / d, y *= c / d;
    LL k = x / b2;
    x -= k * b2, y += k * a2;
    LL res;
    if (c > a) res = 2 * (x - y);
    else res = 2 * (x - y - 1);
    x -= b2, y += a2;
    if (c > b) res = min(res, 2 * (y - x));
    else res = min(res, 2 * (y - x - 1));
    cout << res << endl;
}
return 0;
}
```

4、程序注释统计

【题目描述】C 语言有两种注释，单行注释 // 和多行注释 /* */。请编写程序，统计注释数量。注意引号中的不要统计，以及引号中可能出现的转义字符影响。

输入描述：

C 语言源程序

输出描述：

两种注释加起来的总数量

备注：

数据范围：

对于 50%的数据，0 < 输入行数 < 20

对于 100%的数据，0 < 输入行数 < 1000

输入示例：

```
// line comment
// line comment2
int main() {
  return 0;
}
```

输出示例：

2

说明：

输入有两个单行注释

输入示例 2:

```
// line comment //
/* block comment */ /* block comment 2 */
int main() {
  char[] s = "/* string */";
  return 0;
}
```

输出示例 2:

3

说明:

输入有一个单行注释和两个多行注释，共 3 条注释；注意引号中的不是注释

【答案及解析】

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <vector>
#include <string>
using namespace std;
int main()
{
cout << '\q' << endl;
int res = 0;
char c;
string code;
while ((c = getchar()), c != -1) code += c;
int c1 = 0, c2 = 0, qs = 0;
for (int i = 0; i < code.size(); i++)
{
    if (code[i] == '\n')
{
      c1 = 0;
    }
    else if (c1)
    {
    }
    else if (c2)
    {
       if (i + 1 < code.size() && code[i] == '*'
         && code[i + 1] == '/')
```

```
        {
          c2 = 0;
          i++;
        }
      }
      else if (qs)
      {
        if (code[i] == '"')
        {
          int t = 0;
          for (int j = i - 1; j >= 0; j--)
            if (code[j] == '\\')
            {
              t++;
        }
            else
            {
              break;
            }
          if (t % 2 == 0) qs = 0;
      }
}
else if (code[i] == '"')
{
       qs = 1;
}
else if (i + 1 < code.size() && code[i] == '/'
       && code[i + 1] == '/')
{
      c1 = 1;
      res++;
      i++;
}
else if (i + 1 < code.size() && code[i] == '/'
       && code[i + 1] == '*')
      {
        c2 = 1;
        res++;
        i++;
}
```

```
    else if (code[i] == '\'')
    {
        if (code[i + 1] == '\\') i += 3;
        else i += 2;
    }
}
cout << res << endl;
return 0;
}
```

八、美团点评 2018 秋招笔试真题

1、病毒传播

【题目描述】给出一个图 G(V,E)，图上有 n 个点，m 条边，所有的边都是无向边。
最开始，也就是第 0 天的时候，这 n 个点中有一个点 v 感染了病毒，之后的每一天，凡是感染病毒的点都会向它的邻居点传播病毒。经过了 t 天之后，得到了感染病毒的点集 S。要求找出第 0 天感染病毒的点 v。如果 v 有很多不同的答案，把它们都找出来。

输入描述：

第一行两个数 n，m，接下来有 m 行，每行两个数 u，v，表示点 u，v 之间有一条无向边。接下来一行两个数 k，t，其中 k 表示集合 S 的大小。最后一行 k 个数，集合 S 中的元素。输入的图可能有自环和重边，输入保证 S 中的数互不相同。(1≤n≤〖10〗^3,0≤m≤〖10〗^3,1≤t≤〖10〗^9,1≤u,v,k≤n,S 中所有元素在区间[1,n])

输出描述：

输出一行，如果不存在这样的 v，输出-1。
否则输出所有可能的 v，按照从小到大的顺序输出，数字之间用空格隔开，不要在行末输出多余的空格。

输入样例：

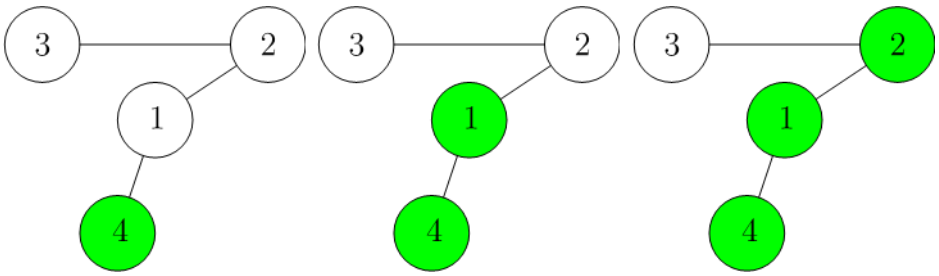
```
4 3
3 2
1 2
1 4
3 2
4 2 1
```

输出样例：

```
4
```

说明：

第 0 天，第 1 天，第 2 天感染病毒的点如图。



【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
bool ans[1005];//感染了病毒的点标记为 1 没感染为 0
int temp[1005];//bfs 过程中每个点最早被传染的时间 为 0 说明没被传染
vector <int> vec[1005];//用邻接表表示的边集
queue <int> q;//bfs 中用到的队列
int n,t;
bool ok(int x)//用 bfs 跑出以 x 为起点 传播 t 天的结果比较和实际结果是否相同
{
    for(int i=1;i<=n;i++)
        temp[i]=0;
    while(!q.empty())
        q.pop();
    //以上是预处理
    temp[x]=1;
    q.push(x);
    int now;
    while(!q.empty())
    {
        now=q.front();
        q.pop();
        if(temp[now]>t)
            break;
        for(int i=0;i<vec[now].size();i++)
        {
            if(temp[vec[now][i]]==0)
            {
                temp[vec[now][i]]=temp[now]+1;
                q.push(vec[now][i]);
            }
        }
    }
```

```
    //以上是 bfs 可以保证复杂度 o(n)
    for(int i=1;i<=n;i++)
    {
        if(ans[i]==0 && temp[i]!=0)
            return 0;
        if(ans[i]!=0 && temp[i]==0)
            return 0;
    }
    //以上是验证是否符合给出的点集
    return 1;
}
int main()
{
    int m,u,v,num=0;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
        scanf("%d%d",&u,&v);
        vec[u].push_back(v);
        vec[v].push_back(u);
    }
    //以上是建图
    int s,k;
    scanf("%d%d",&k,&t);
    for(int i=1;i<=k;i++)
    {
        scanf("%d",&s);
        ans[s]=1;
    }
    //标记被感染的点
    for(int i=1;i<=n;i++)
    {
        if(ok(i))//输出符合结果的点
        {
            if(num>0)
                printf(" ");//防止行末空格
            num++;
            printf("%d",i);
        }
    }
```

```
    if(num==0)//无解
        printf("-1");
    printf("\n");
    return 0;
}
```

2、公交车

【题目描述】一座城市有 n 个公交站点，站点从 1 到 n 编号，和 m 班公交车，公交车从 1 到 m 编号，乘坐每班公交车只需花费 1 元钱。第 i 班公交车一共经过 t_i 个站点，分别为站点 a_(i,1),a_(i,2),…,a_(i,t_i)，小明可以乘坐第 i 班公交车从这 t_i 个站点中的任意一个到达任意另一个站点。如一班公交车经过站点 1,2,3，那么小明花费 1 元钱就可以从 1 到 2，从 1 到 3，从 2 到 1，从 2 到 3，从 3 到 1，从 3 到 2。

小明想从 1 号站点到 n 号站点，问他最少花费多少钱。

输入描述:

第一行两个数 n,m。(2≤n≤100000,1≤m≤100000)

接下来 m 行，依次描述每班公交车经过的站点，第 i 行开头一个数 t_i，表示第 i 班公交经过的站点数，接下来 t_i 个数，依次表示这 t_i 个站点。(〖2≤t〗_i≤n,∑_(i=1)^m▒〖t_i≤100000〗)

输出描述:

输出一个数，从 1 号站点到 n 号站点的最小代价，如果不能达到，输出-1。

输入示例:

5 3

3 1 2 3

3 3 4 2

3 3 5 4

输出示例:

2

样例解释:

先坐第 1 班公交从 1 号站点到 3 号站点，再做第 3 班公交从 3 号站点到 5 号站点，一共花费 2 元。

【答案及解析】

```
#include<iostream>
#include<cstdio>
#include<vector>
#include<queue>
using namespace std;
int dis[200005];
vector <int> vec[200005];
queue <int> q;
int main()
```

```
{
    int n,m,t,a;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
        scanf("%d",&t);
        for(int j=1;j<=t;j++)
        {
            scanf("%d",&a);
            vec[a].push_back(i+100000);
            vec[i+100000].push_back(a);
        }
    }
    //以上是建图 同一辆公交车通过的点建一条边到这辆公交车的抽象点，再从抽象点建一条边到这些点
    dis[1]=1;
    q.push(1);
    int now;
    while(!q.empty())
    {
        now=q.front();
        q.pop();
        for(int i=0;i<vec[now].size();i++)
        {
            if(dis[vec[now][i]]==0)
            {
                dis[vec[now][i]]=dis[now]+1;
                q.push(vec[now][i]);
            }
        }
    }
    //以上是 bfs 求最短路
    if(dis[n]==0)
        printf("-1\n");
    else
        printf("%d\n",dis[n]/2);//因为中间设立了抽象点，起点距离又设为了 1，所以每个点的距离都是实际距离的两倍加一
    return 0;
}
```

3、重要节点

【题目描述】给出一张有向图 G(V,E)，所有的边都是有向边。对于图上的一个点 v，从 v 出发可以到达的点的集合记为 S_v，特别地，v∈S_v。再定义一个点的集合 T_v：从 T_v 中的任意一个点出发，可以达到点 v，特别地，v∈T_v，。简而言之，S_v 是 v 能到的点的集合，而 T_v 是能到 v 的点的集合。

如果对于一个点 v，如果 T_v 中的点数严格大于 S_v 中的点数，那么 v 就是一个重要节点。输入一张图，输出图中重要节点的个数。

输入描述：

第一行输入两个数 n,m(1≤n,m≤1000)，分别表示点数和边数。

接下来 m 行，每行两个数 u，v。表示一条从 u 到 v 的有向边，输入中可能存在重边和自环。(1≤u,v≤n)

输出描述：

输出一个数，重要节点的个数。

输入示例：

4 3

2 1

3 1

1 4

输出示例：

2

样例解释：

重要节点是 1、4。

【答案及解析】

```
#include<iostream>
#include<vector>
#include<queue>
#include<cstdio>
using namespace std;
bool ok[1005][1005];//ok[i][j]为 1 表示 i 可以到达 j 为 0 表示 i 不能到达 j
int numin[1005],numout[1005];//numin[i]储存能到达点 i 的点的数量 numout[i]储存点 i 能到达的点的数量
vector <int> vec[1005];//以邻接表的形式存边
queue <int> q;//bfs 过程中用到的队列
int main()
{
    //freopen("10.in","r",stdin);
    //freopen("10.out","w",stdout);
    int n,m,u,v;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++)
    {
```

```
        scanf("%d%d",&u,&v);
        vec[u].push_back(v);
    }
    int now;
    for(int i=1;i<=n;i++)
    {
        q.push(i);
        ok[i][i]=1;
        while(!q.empty())
        {
            now=q.front();
            q.pop();
            for(int j=0;j<vec[now].size();j++)
            {
                if(ok[i][vec[now][j]]==0)
                {
                    ok[i][vec[now][j]]=1;
                    q.push(vec[now][j]);
                }
            }
        }
    }
    //枚举起点进行 bfs 找出这个点所能达到的所有点 在 ok 数组中进行标记
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
        {
            if(ok[i][j]==1)
            {
                numin[j]++;
                numout[i]++;
            }
        }
    //根据 ok 数组记录计算每个点能达到的点数 和能达到每个点的点数
    int ans=0;
    for(int i=1;i<=n;i++)
    {
        if(numin[i]>numout[i])
            ans++;
    }
    //计算重要节点的数量
```

```
    printf("%d\n",ans);
    return 0;
}
```

4、硬币兑换

【题目描述】A 国一共发行了几种不同面值的硬币，分别是面值 1 元，2 元，5 元，10 元，20 元，50 元，100 元。假设每种面值的硬币数量是无限，现在你想用这些硬币凑出总面值为 n 的硬币，同时你想让选出的硬币中，不同的面值种类尽可能多；在面值种类尽可能多的情况下，你想让选择的硬币总数目尽可能多，请问应该怎么选择硬币？

输入描述：

第一行包含一个数字n，表示要凑出的面值。$1 \leqslant n \leqslant 10^9$

输出描述：

输出两个整数，分别表示最多能有多少种类型的硬币以及在类型最多的情况下最多能用上多少枚硬币。

输入样例 1：

3

输出样例 1：

2 2

输入样例 2：

10

输出样例 2：

3 5

样例解释：在样例 2 中，最优的选择方法是 3 枚面值为 1 的，1 枚面值为 2 的，1 枚面值为 5 的。

【答案及解析】

```
#include<iostream>
#include<cstdio>
using namespace std;
int num[10]={0,1,2,5,10,20,50,100};
int sum[10];
int main()
{
    //freopen("10.in","r",stdin);
    //freopen("10.out","w",stdout);
    for(int i=1;i<=7;i++)
        sum[i]=sum[i-1]+num[i];
    int n;
    scanf("%d",&n);
    for(int i=7;i>=1;i--)
    {
```

```
        if(n>=sum[i])
        {
            printf("%d %d\n",i,n-sum[i]+i);
            break;
        }
    }
    //策略为从小到大，每种面值的硬币用一个，剩下的全部用面值为 1 的硬币填充
    return 0;
}
```

九、拼多多 2018 秋招笔试真题

1、小熊吃糖

【题目描述】有 n 只小熊，他们有着各不相同的战斗力。每次他们吃糖时，会按照战斗力来排，战斗力高的小熊拥有优先选择权。前面的小熊吃饱了，后面的小熊才能吃。每只小熊有一个饥饿值，每次进食的时候，小熊们会选择最大的能填饱自己当前饥饿值的那颗糖来吃，可能吃完没饱会重复上述过程，但不会选择吃撑。 现在给出 n 只小熊的战斗力和饥饿值，并且给出 m 颗糖能填饱的饥饿值。 求所有小熊进食完之后，每只小熊剩余的饥饿值。

输入描述:

第一行两个正整数 n 和 m，分别表示小熊数量和糖的数量。（n <= 10, m <= 100） 第二行 m 个正整数，每个表示着颗糖能填充的饥饿值。 接下来的 n 行，每行 2 个正整数，分别代表每只小熊的战斗力和当前饥饿值。 题目中所有输入的数值小于等于 100。

输出描述:

输出 n 行，每行一个整数，代表每只小熊剩余的饥饿值。

输入示例:

2 5
5 6 10 20 30
4 34
3 35

输出示例:

4
0

说明:

第一只小熊吃了第 5 颗糖
第二只小熊吃了第 4 颗糖
第二只小熊吃了第 3 颗糖
第二只小熊吃了第 1 颗糖

【答案及解析】

```
#include <cstdio>
#include <algorithm>
using namespace std;
//按照规则模拟即可。需要用到排序算法：按战斗力对小熊们排序；按能填充的饥饿值对糖排序。唯一需要注意的是最终答案需要按照小熊输入的顺序进行输出，在排序之后要记录一下这个信息。
struct bear {
    int fight;
    int hunger;
    int id;
    friend bool operator < (bear a, bear b) {
        return a.fight > b.fight;
    }
};
int a[100];
bear b[10];
bool v[100];
int ans[10];
int n, m;
int main() {
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; ++i) {
        scanf("%d", a + i);
    }
    for (int i = 0; i < n; ++i) {
        scanf("%d %d", &b[i].fight, &b[i].hunger);
        b[i].id = i;
    }
    sort(b, b + n);
    for (int i = 0; i < n; ++i) {
        int index = -1;
        for (int j = 0; j < m; ++j) {
            if (!v[j] && b[i].hunger >= a[j] &&  (index == -1 || a[j] > a[index])) {
                index = j;
            }
        }
        if (index != -1) {
            b[i].hunger -= a[index];
            v[index] = true;
            --i;
        } else {
```

```
            ans[b[i].id] = b[i].hunger;
        }
    }
    for (int i = 0; i < n; ++i) {
        printf("%d\n", ans[i]);
    }
    return 0;
}
```

2、数三角形

【题目描述】

给出平面上的 n 个点，现在需要你求出，在这 n 个点里选 3 个点能构成一个三角形的方案有几种。

输入描述:

第一行包含一个正整数 n，表示平面上有 n 个点（n <= 100）第 2 行到第 n + 1 行，每行有两个整数，表示这个点的 x 坐标和 y 坐标。(所有坐标的绝对值小于等于 100，且保证所有坐标不同）

输出描述:

输出一个数，表示能构成三角形的方案数。

输入:

4
0 0
0 1
1 0
1 1

输出:

4

说明:

4 个点中任意选择 3 个都能构成三角形

【答案及解析】

```
#include <cstdio>
using namespace std;
struct Point {
    int x;
    int y;
};
//算三角形的方案数有一些精巧的做法，但在这里不做考察。注意到这题的输入数据范围很小，只需要O(n3)暴力枚举即可。
Point a[100];
int n;
```

```
int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) {
        scanf("%d %d", &a[i].x, &a[i].y);
    }
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            for (int k = j + 1; k < n; ++k) {
                if ((a[j].x - a[i].x) * (a[k].y - a[i].y) - (a[k].x - a[i].x) * (a[j].y -
a[i].y) != 0) {
                    ++ans;
                }
            }
        }
    }
    printf("%d\n", ans);
}
```

3、列表补全

【题目描述】在商城的某个位置有一个商品列表，该列表是由 L1、L2 两个子列表拼接而成。当用户浏览并翻页时，需要从列表 L1、L2 中获取商品进行展示。展示规则如下：

1. 用户可以进行多次翻页，用 offset 表示用户在之前页面已经浏览的商品数量，比如 offset 为 4，表示用户已经看了 4 个商品
2. n 表示当前页面需要展示的商品数量
3. 展示商品时首先使用列表 L1，如果列表 L1 长度不够，再从列表 L2 中选取商品
4. 从列表 L2 中补全商品时，也可能存在数量不足的情况

请根据上述规则，计算列表 L1 和 L2 中哪些商品在当前页面被展示了

输入描述：

每个测试输入包含 1 个测试用例，包含四个整数，分别表示偏移量 offset、元素数量 n，列表 L1 的长度 l1，列表 L2 的长度 l2。

输出描述：

在一行内输出四个整数分别表示 L1 和 L2 的区间 start1，end1，start2，end2，每个数字之间有一个空格。注意，区间段使用半开半闭区间表示，即包含起点，不包含终点。如果某个列表的区间为空，使用[0, 0)表示，如果某个列表被跳过，使用[len, len)表示，len 表示列表的长度。

输入示例：

2 4 4 4

1 2 4 4

4 1 3 3

输出示例:

2 4 0 2

1 3 0 0

3 3 1 2

【答案及解析】

```
import java.io.IOException;
import java.util.Scanner;

public class Answer {

    public static String slice(int offset, int count, int a, int b) {

        int startA = Math.min(a, Math.max(0, offset));
        int endA = Math.max(0, Math.min(a, offset + count));
        int startB = Math.min(b, Math.max(0, offset - a));
        int endB = Math.max(0, Math.min(b, offset + count - a));
        return startA + " " + endA + " " + startB + " " + endB;
    }

    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        while (input.hasNext()) {
            System.out.print(slice(input.nextInt(), input.nextInt(), input.nextInt(),
input.nextInt()));
        }
    }
}
```

4、最大乘积

【题目描述】

给定一个无序数组，包含正数、负数和 0，要求从中找出 3 个数的乘积，使得乘积最大，要求时间复杂度：O(n)，空间复杂度：O(1)

输入描述:

第一行是数组大小 n，第二行是无序整数数组 A[n]。n >=3

输出描述:

满足条件的最大乘积

输入示例:

4

3 4 1 2

输出示例:

24

【答案及解析】

考虑 3 种情况：

1、全是正数

2、全是负数

3、有正有负

前两种情况都只需要取得最大的 3 个数相乘即可。

第三种情况最好的情况肯定是两个最小的负数相乘得到一个正数，然后跟一个最大的正数相乘，这样得到的数和三个最大数相乘的结果中较大的那个肯定是最大的数。

问题转化成求出数组中最大的三个数：max1, max2, max3, 两个最小的数 min1,min2，然后比较 max1 * max2 * max3 和 max1 * min1 * min2 的大小。

```
#include<stdio.h>
int maxProduct(int a[],int n)
{
      int max1,max2,max3,min1,min2, product;
      max1=a[0] > a[1] ? a[0]:a[1];
      max2=a[0] > a[1] ? a[1]:a[0];
      if(a[2]>max1){
            max3=max2;
            max2=max1;
            max1=a[2];
      }
      else if(a[2]>max2){
            max3=max2;
            max2=a[2];
      }
     else {
            max3=a[2];
      }
      min1=max3,min2=max2;
      for(int i=3;i<n;i++){
            if(a[i]<min1){
                  min2=min1;
                  min1=a[i];
            }
            else if(a[i]<min2)
```

```
                min2=a[i];
            if(a[i]>max1){
                max3=max2;
                max2=max1;
                max1=a[i];
            }
            else if(a[i]>max2){
                max3=max2;
                max2=a[i];
            }
            else if(a[i]>max3){
                max3=a[i];
            }
        }
        product = max1*max2*max3 > max1*min1*min2 ? max1*max2*max3 : max1*min1*min2;
        return product;
}
int main()
{
        int i,n;
        int a[1000];
        scanf("%d",&n);
        for(i=0;i<n;i++) {
            scanf("%d",&a[i]);
        }
        printf("%d\n",maxProduct(a, n));
        return 0;
}
```

十、吉比特 2018 秋招笔试真题

1、二进制个位不同个数

【题目描述】输入两个整数，求两个整数二进制格式有多少个位不同。

输入示例：

22 33

输出示例：

5

解题思路：

两个数进行异或，将异或的结果与其减一进行与操作，直至为零，就是二进制不同位数的数量。

【答案及解析】

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();
        int m = scanner.nextInt();
        int num = n ^ m;
        int count = 0;
        while (num != 0)
        {
            count++;
            num = (num - 1) & num;
        }
        System.out.println(count);
    }
}
```

2、走格子游戏

【题目描述】G 社正在开发一个新的战棋类游戏，在这个游戏中，角色只能向 2 个方向移动：右、下。移动需要消耗行动力，游戏地图上划分 M * N 个格子，当角色移动到某个格子上时，行动力就会加上格子上的值 K（-100~100），当行动力小于等于 0 时游戏失败，请问要从地图左上角移动到地图右下角至少需要多少起始行动力，注意（玩家初始化到起始的左上角格子时也需要消耗行动力）。

输入说明：第一行输入格子行列数（格式为：M N），第 2~M+1 行每行输入 N 个数，作为格子值 K，中间以空格分割；0 < M, N < 1000，-100 < K < 100

输入示例：

2 3

-2 -3 3

-5 -10 1

输出示例：

6

解题思路：

动态规划问题，状态 s(x,y)为在位置为(x,y)时到达目标点所需的最小行动力，任何情况下，当前行动力不可小于 1，分析可得 状态转移方程为：

```
if x == col - 1 && y == row - 1
```

```
    s(x,y) = max(1 - grid[y][x], 1)
if x == col - 1
    s(x,y) = max(s(x, y + 1) - grid[y][x], 1)
if y == row - 1
    s(x,y) = max(s(x + 1, y) - grid[y][x], 1)
else
    s(x,y) = max(max(s(x, y + 1) - grid[y][x],
                     s(x + 1, y) - grid[y][x]),
                 1)
```

【答案及解析】

```
class Solution
{
    int row;
    int col;
    int* dp;
public:
    int calculateMinimumHP(vector<vector<int>>& grid)
    {
        row = (int)grid.size();
        col = (int)grid[0].size();
        dp = new int[row * col];
        for (size_t i = 0; i < row * col; ++i)
            dp[i] = INT_MAX;
        return foo(grid, 0, 0);
    }
    int foo(vector<vector<int>>& grid, int x, int y)
    {
        if (dp[y * col + x] != INT_MAX)
            return dp[y * col + x];
        auto val = grid[y][x];
        int min_hp;
        if (x == col - 1 && y == row - 1)
            min_hp = max(1 - val, 1);
        else if (x == col - 1)
            min_hp = max(foo(grid, x, y + 1) - val, 1);
        else if (y == row - 1)
            min_hp = max(foo(grid, x + 1, y) - val, 1);
        else
        {
            auto hp1 = max(foo(grid, x, y + 1) - val, 1);
```

```
            auto hp2 = max(foo(grid, x + 1, y) - val, 1);
            min_hp = min(hp1, hp2);
        }
        dp[y * col + x] = min_hp;
        return min_hp;
    }
};
void main()
{
    int M, N;
    scanf("%d %d", &M, &N);
    vector<vector<int>> grids;
    for (int i = 0; i < M; i++)
    {
        vector<int> arr;
        for (int j = 0; j < N; j++)
        {
            int K;
            scanf("%d", &K);
            arr.push_back(K);
        }
        grids.push_back(arr);
    }
    Solution sol;
    printf("%d\n", sol.calculateMinimumHP(grids));
    return;
}
```

3、直线上的点

【题目描述】给定 N 个三维坐标点(包含整形 x,y,z)，找到位于同一条直线上点的最大个数。

输入说明：第一行输入坐标点的个数 N，第 2~N+1 行输入 N 个点（格式为 x y z），0 < N < 2000; -10000 < x,y,z < 10000

输入示例：

4

0 0 0

1 1 1

-1 -1 -1

0 1 0

输出示例：

3

解题思路：

两个点决定一条直线，所以两层遍历，可以遍历所有直线，并在遍历的过程中，把直线缓存起来，下次再遇到的相同直线时候就只是把这条直线对应的存在的点的个数累加。这样我们就可以算出经过某个点的某条直线上最多的点的个数。要注意的是第二层遍历的时候，可以只遍历从 i + 1 ~ size - 1 的直线，因为与之前的点能够组成的直线已经处理过，不需要再处理（a,b 决定的直线和 b,a 决定的直线是一样的）

【答案及解析】

```
#include <stdio.h>
#include <time.h>
#include <map>
#include <unordered_map>
#include <vector>
#include <algorithm>
using namespace std;
struct Point3D
{
    int x, y, z;
    Point3D(int x, int y, int z)
        : x(x), y(y), z(z)
    {}
    bool operator == (const Point3D& pt) const
    {
        return x == pt.x && y == pt.y && z == pt.z;
    }
};
struct Line3D
{
    // dx / dz
    float kx;
    // dy / dz
    float ky;
    // dz == 0 ?
    bool bz;
    bool operator==(const Line3D& l) const
    {
        return kx == l.kx && ky == l.ky && bz == l.bz;
    }
    bool operator < (const Line3D& l) const
    {
        return kx < l.kx ||
            kx == l.kx && ky < l.ky;
```

```
    }
};
namespace std
{
    template<>
    struct hash<Line3D>
    {
        size_t operator() (const Line3D& line) const
        {
            size_t hash1 = 11111111;
            size_t hash2 = 4444;
            size_t hash3 = 89898989;
            return (size_t)(line.kx * hash1) + (size_t)(line.ky * hash2) + hash3 *
(int)line.bz;
        }
    };
}
int maxPoints(vector<Point3D> points)
{
    int maxNum = 0;
    for (size_t i = 0; i < points.size(); ++i)
    {
        int sameNum = 0;
        int ptMaxCount = 0;
        unordered_map<Line3D, int> lineMap;
        for (size_t j = i + 1; j < points.size(); ++j)
        {
            auto& p1 = points[i];
            auto& p2 = points[j];
            Line3D line;
            if (p1 == p2)
            {
                sameNum++;
                continue;
            }
            int dz = p2.z - p1.z;
            if (dz == 0)
            {
                line.bz = true;
                line.kx = line.ky = 0;
```

```
            }
            else
            {
                line.bz = false;
                line.kx = (float)(p2.x - p1.x) / dz;
                line.ky = (float)(p2.y - p1.y) / dz;
            }
            int count;
            if (lineMap.find(line) == lineMap.end())
                count = lineMap[line] = 1;
            else
                count = ++lineMap[line];
            ptMaxCount = max(ptMaxCount, count);
        }
        maxNum = max(ptMaxCount + sameNum + 1, maxNum);
    }
    return maxNum;
}
```

十一、搜狐 2018 秋招笔试真题

1、包裹运输

【题目描述】工厂生产的产品包装在相同高度 h，尺寸为 1 * 1,2 * 2，3 * 3，4 * 4，5 * 5，6 * 6 的方形包装中。 这些产品始终以与产品高度相同的尺寸为 6 * 6 的包裹交付给客户。因为邮费很贵，所以工厂要想方设法的减小每个订单运送时的包裹数量。他们很需要有一个好的程序帮他们解决这个问题从而节省费用。现在这个程序由你来设计。

输入描述:

输入文件包括几行，每一行代表一个订单。每个订单里的一行包括六个整数，中间用空格隔开，分别为 1*1 至 6*6 这六种产品的数量。输入文件将以 6 个 0 组成的一行结尾。

输出描述:

除了输入的最后一行 6 个 0 以外，输入文件里每一行对应着输出文件的一行，每一行输出一个整数代表对应的订单所需的最小包裹数。

输入示例:

0 0 4 0 0 1

7 5 1 0 0 0

0 0 0 0 0 0

输出示例:

2

1

【答案及解析】

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Main main = new Main();
        int[] nums = new int[6];
        List<Integer> list = new ArrayList<Integer>();
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            for (int i = 0; i < 6; i++) {
                nums[i] = sc.nextInt();
            }
            if (main.isEnd(nums))
                break;
            list.add(main.solve(nums));
        }//while
        for (int i = 0; i < list.size(); i++) {
            System.out.println(list.get(i));
        }
    }//main
    private boolean isEnd(int[] nums) {
        for (int i = 0; i < 6; i++) {
            if (nums[i] != 0) {
                return false;
            }
        }
        return true;
    }//isEnd
    private int solve(int nums[]) {
        int count = 0;
        int spareSum55 = 0;//装完 5*5 的产品后剩余的空间总和
        int spareSum44 = 0;//装完 4*4 的产品后剩余的空间总和
        int spareSum33 = 0;
        int spareSum22 = 0;
        int need22 = 0;//开始装 2*2 的产品时需要的空间大小
        int need11 = 0;
        for (int i = 3; i <= 5; i++) {
            if (nums[i] != 0)
```

```
                count += nums[i];
            }
            spareSum55 = nums[4] * 11;
            spareSum44 = nums[3] * 20;
            //开始装 3*3
            if (nums[2] % 4 == 0) {
                count += nums[2] / 4;
                spareSum33 = 0;
            } else {
                count += nums[2] / 4 + 1;
                spareSum33 = 36 - (nums[2] % 4) * 9;
            }
            //开始装 2*2,不能往 5*5 的剩余空间中装
            if (spareSum44 + (spareSum33 / 4) * 4 >= nums[1] * 4) {
                spareSum22 = spareSum44 + (spareSum33 / 4) * 4 - nums[1] * 4;
            } else {
                need22 = nums[1] * 4 - spareSum44 + (spareSum33 / 4) * 4;
                if (need22 % 36 == 0) {
                    count += need22 / 36;
                    spareSum22 = 0;
                } else {
                    count += need22 / 36 + 1;
                    spareSum22 = 36 - (nums[1] % 9) * 4;
                }
            }
            //开始装 1*1
            if (count * 36 - nums[1] * 4 - nums[2] * 9 - nums[3] * 16 - nums[4] * 25 - nums[5] *
36 > nums[0]) {
                return count;
            } else {
                need11 = nums[0] - (count * 36 - nums[1] * 4 - nums[2] * 9 - nums[3] * 16 -
nums[4] * 11 - nums[5] * 36);
                if (need11 % 36 == 0) {
                    count += need11 / 36;
                } else {
                    count += need11 / 36 + 1;
                }
            }
            return count;
        }
```

2、Unix 路径简化

【题目描述】简化 Unix 风格的路径，需要考虑的包括 "/../", "//", "/./" 等情况

输入描述:

Unix 风格的路径

输出描述:

简化后的 Unix 风格路径

输入示例:

/a/./b/../../c/

输出示例:

/c

【答案及解析】

```
#include <iostream>
#include <string>
#include <stack>
using namespace std;
int main(){
    string str;
    cin >> str;
    int len = str.length();
    stack<string> stringStack;
    int i = 0;
    string tmp;
    while (i < len)
    {
        while (i < len && str[i] == '/')
            i++;
        tmp.clear();
        while (i < len && str[i] != '/')
        {
            tmp += str[i];
            i++;
        }
        if (tmp == "..")
        {
            if (!stringStack.empty())
                stringStack.pop();
        }
        else if (tmp == ".")
```

```
            continue;
        else if (!tmp.empty())
        {
            stringStack.push(tmp);
        }
    }
    if (stringStack.empty()){
        cout << "/";
        return 0;
    }
    string result = "";
    while (!stringStack.empty())
    {
        result = "/" + stringStack.top() + result;
        stringStack.pop();
    }
    cout << result;
    return 0;
}
```

3、回文数组

【题目描述】对于一个给定的正整数组成的数组 a[]，如果将 a 倒序后数字的排列与 a 完全相同，我们称这个数组为“回文”的。

例如，[1, 2, 3, 2, 1] 的倒序是他自己，所以是一个回文的数组；而 [1, 2, 3, 1, 2] 的倒序是 [2, 1, 3, 2, 1]，所以不是一个回文的数组。

对于任意一个正整数数组，如果我们向其中某些特定的位置插入一些正整数，那么我们总是能构造出一个回文的数组。

输入一个正整数组成的数组，要求你插入一些数字，使其变为回文的数组，且数组中所有数字的和尽可能小。输出这个插入后数组中元素的和。

例如，对于数组 [1, 2, 3, 1, 2] 我们可以插入两个 1 将其变为回文的数组 [1, 2, 1, 3, 1, 2, 1]，这种变换方式数组的总和最小，为 11 ，所以输出为 11 。

输入描述:

输入数据由两行组成： 第一行包含一个正整数 L ，表示数组 a 的长度。 第二行包含 L 个正整数，表示数组 a 。 对于 40% 的数据： 1 < L <= 100 达成条件时需要插入的数字数量不多于 2 个。 对于 100% 的数据： 1 < L <= 1,000 0 < a[i] <= 1,000,000 达成条件时需要插入的数字数量没有限制。

输出描述:

输出一个整数，表示通过插入若干个正整数使数组 a 回文后，数组 a 的数字和的最小值。

输入示例

8

51 23 52 97 97 76 23 51

输出示例

598

【答案及解析】

```
import java.util.Scanner;
public class PalindromeArray {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		while (sc.hasNext()) {
			int len = sc.nextInt();
			int[] nums = new int[len];
			for (int i = 0; i < len; i++) {
				nums[i] = sc.nextInt();
			}
			Integer[][] dp = new Integer[len][len];
			int res = palindromeArrayHelper(nums, dp, 0, len - 1);
			System.out.println(res);
		}
		sc.close();
	}
	public static int palindromeArrayHelper(int[] nums, Integer[][] dp, int i, int j) {
		if (i > j) {
			return 0;
		}
		if (i == j) {
			return nums[i];
		}
		if (dp[i][j] != null) {
			return dp[i][j];
		}
		if (nums[i] == nums[j]) {
			dp[i][j] = 2 * nums[i] + palindromeArrayHelper(nums, dp, i + 1, j -
1);
		} else {
			dp[i][j] = Math.min(2 * nums[i] + palindromeArrayHelper(nums, dp, i
+ 1, j),
					2 * nums[j] + palindromeArrayHelper(nums, dp, i, j -
1));
		}
		return dp[i][j];
```

```
        }
    }
```

十二、搜狐畅游 2018 秋招笔试真题

1、第二大的数

【题目描述】输入 n 个整数，查找数组中第二大的数

输入描述:

第一行 n 表示 n 个数，第二行 n 个空格隔开的数

输出描述:

输出第二大的数

输入例子 1:

5

1 2 3 4 5

输出例子 1:

4

【答案及解析】

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int max = Integer.MIN_VALUE;
        int sec = Integer.MIN_VALUE;
        while(sc.hasNext()){
            int temp = sc.nextInt();
            if(temp >= max){
                sec = max;
                max = temp;
            }
            else if(temp >= sec)
                sec = temp;
        }
        System.out.print(sec);
    }
}
```

2、数据分页说明

【题目描述】数据分页,对于指定的页数和每页的元素个数，返回该页应该显示的数据。

输入描述:

第一行输入数据个数，第二行全部数据，第三行输入页数，第四行输入每页最大数据个数

输出描述:

输出该页应该显示的数据，超出范围请输出'超过分页范围'

输入例子 1:

6

1 2 3 4 5 6

1

2

输出例子 1:

3

4

【答案及解析】

```
//将数组存入内存中，内存分页大小固定，让输出给定页号的内容。
#include <iostream>
using namespace std;
int main()
{
        int n;
        while (cin >> n) {
                int *arr = new int[n];
                for (int i = 0; i < n; ++i)
                        cin >> arr[i];
                int page, num;
                cin >> page >> num;
                if (page > n / num - 1) {
                        cout << "超过分页范围" << endl;
                        continue;
                }
                for (int i = page * num; i < page * num + num; ++i)
                        cout << arr[i] << endl;
                delete[]arr;
        }
        return 0;
}
```

十三、小米 2018 秋招笔试真题

1、24 点

【题目描述】有 n 个 1~23 的整数，写一个算法，求出有多少个相互不同的子集合的和为 24 点。

输入描述:

输入数据包含一组，每组占一行，包括 n 个整数 （1<=整数 <=23）

输出描述:

对于每个测试实例，要求输出能组成 24 点的所有子集合的数量（子集合相互不同）。如果不存在，则输出 0。每个测试实例的输出占一行。

输入样例:

1 2 22 23

输出样例:

2

【答案及解析】

```
#include <bits/stdc++.h>
using namespace std;
int a[30], n;
set<set<int> > S;
void dfs(int sum, int pos, set<int> s) {
    if(pos == n + 1) return;
    if(sum == 24) {
        S.insert(s);
        return;
    }
    dfs(sum, pos + 1, s);
    s.insert(a[pos]);
    dfs(sum + a[pos], pos + 1, s);
}
int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    set<int> s;
    dfs(0, 0, s);
    printf("%d\n", (int)S.size());
    return 0;
}
```

2、完美矩形

【题目描述】给定 n 个轴对齐的矩形其中 n>0，判断他们组合在一起能否覆盖一个完美的矩形区域（无重叠，无空隙）

每个矩形使用左下和右上的点表示。例如，一个矩形的定义为 [1,1,2,2]，(左下坐标点 (1, 1) 和右上坐标点 (2, 2)的一个单元的正方形)。

输入描述:

输入包含一组数据，有 n 行，每行代表一个矩形 （左下坐标点和右上坐标点），数字用空格隔开。

输出描述:

对于每个测试实例，输出能否组合覆盖一个矩形（true/false） 。

输入样例:

1 1 3 3

3 1 4 2

3 2 4 4

1 3 2 4

2 3 3 4

输出样例:

true

【答案及解析】

本题有一定难度，需要遍历每个矩形，遍历过程中与前面矩形是否有重叠，并调整最小公共父矩形的数值，最后判断所有矩形面积是否与父矩形一样。

```
 #include <bits/stdc++.h>
using namespace std;
class Solution {
    public:
        bool isRectangleCover(vector<vector<int>>& rectangles) {
            unordered_set<string> st;
            int min_x = INT_MAX, min_y = INT_MAX, max_x = INT_MIN, max_y = INT_MIN, area =
0;
            for (auto rect : rectangles) {
                min_x = min(min_x, rect[0]);
                min_y = min(min_y, rect[1]);
                max_x = max(max_x, rect[2]);
                max_y = max(max_y, rect[3]);
                area += (rect[2] - rect[0]) * (rect[3] - rect[1]);
                string s1 = to_string(rect[0]) + "_" + to_string(rect[1]); // bottom-left
                string s2 = to_string(rect[0]) + "_" + to_string(rect[3]); // top-left
                string s3 = to_string(rect[2]) + "_" + to_string(rect[3]); // top-right
                string s4 = to_string(rect[2]) + "_" + to_string(rect[1]); // bottom-
right
```

```
                if (st.count(s1)) st.erase(s1);
                else st.insert(s1);
                if (st.count(s2)) st.erase(s2);
                else st.insert(s2);
                if (st.count(s3)) st.erase(s3);
                else st.insert(s3);
                if (st.count(s4)) st.erase(s4);
                else st.insert(s4);
            }
            string t1 = to_string(min_x) + "_" + to_string(min_y);
            string t2 = to_string(min_x) + "_" + to_string(max_y);
            string t3 = to_string(max_x) + "_" + to_string(max_y);
            string t4 = to_string(max_x) + "_" + to_string(min_y);
            if (!st.count(t1) || !st.count(t2) || !st.count(t3) || !st.count(t4) ||
st.size() != 4) return false;
            return area == (max_x - min_x) * (max_y - min_y);
        }
    };
int main() {
    int n;
    cin >> n;
    vector<vector<int> > rect;
    for(int i = 0; i < n; i++) {
        vector<int> v;
        for(int j = 0; j < 4; j++) {
            int x;
            cin >> x;
            v.push_back(x);
        }
        rect.push_back(v);
    }
    Solution s;
    if(s.isRectangleCover(rect)) printf("true\n");
    else printf("false\n");
}
```

十四、七牛云 2018 秋招笔试真题

1. 单向链表

【题目描述】输入一个单向链表，输出该链表中倒数第 k 个节点，链表的倒数第 1 个节点为链表的尾指针。

比如链表为：1->2->3->4->5->6->7->8

则其倒数第 3 个节点为：6

链表节点定义为：

```
typedef struct node_ {
  int key;
  struct node_ *next;
} Node;
```

函数定义：

```
Node *the_reciprocal_kth_node(Node *head, int k)
{
// 请补全该函数
}
```

【答案及解析】

通过两个指针来解决问题，但要注意代码中的各种边界条件检查。

```
Node *the_reciprocal_kth_node(Node *head, int k)
{
  // check k value
  if (k < 1)
     return NULL;
  Node *slow, *fast;
  slow = fast = head;
  int i = k;
  for (; i > 0 && fast != NULL; i--) {
     fast = fast->next;
  }
  // if k value bigger than the Link's length
  if(i > 0)
     return NULL;
  while (fast != NULL) {
      slow = slow->next;
      fast = fast->next;
  }
  return slow;
}
```

2、Young Tableau

【题目描述】Young Tableau 是满足如下定义的二维数表：

1）所有行均为左对齐，所有行的最左端在同一列，且每一行均连续无空位；

2）从上到下每一行的列数非严格单调递减；

3）每一行中的数从左到右严格单调递增；

4）每一列中的数从上到下严格单调递增。

请编写一个程序，判断一个数表是否是 Young Tableau。数表的行数和列数至多为 10，所有数均为正整数。

输入样例：

1 2 3

4 5

6

是 Young Tableau，而

1 3 2

4 5

6

和

1 4 5

2 3

6

都不是。

输入格式：

每组测试数据包含多个测试点，输入的第一行表示测试点的数量（至多 5 组）。

每个测试点是一个二维数表，输入中的每一行表示数表的一行，不同的数之间用一个空格隔开。 测试点之间用一个空行隔开。 例如

2

1 3 4 5 6

2 7 10

8 9

2 5 8 9

7 11 10

输出格式：

针对每一个测试点输出一行，内容为 true/false（全部小写）表示是/不是 Young Tableau。例如

true

false

【答案及解析】

```
bool isYoungTableau(int a[10][10]) {
```

```
        if (a == NULL) {
                return false;
        }
        unsigned int prev_length = UINT_MAX;
        for (int i = 0; i < 10; i++) {
            int row_length = 0;
            // the line to annotate end of table
            if (a[i][0] == 0) {
                break;
            }
            for(int j = 0; j < 10; j++) {
                // the end of row
                if (a[i][j] == 0) {
                    row_length = j;
                    break;
                } else {
                // must be greater than the number on the left
                if (j > 0 && a[i][j] <= a[i][j-1]) {
                    return false;
                }
                // must be greater than the number above
                if (i > 0 && a[i][j] <= a[i-1][j]) {
                    return false;
                }
            }
        }
        // row length decrease
        if (row_length > prev_length) {
            return false;
        }
        prev_length = row_length;
    }
    return true;
}
```

3、字符串处理

【题目描述】给定一个只包含大写英文字母的字符串 s，按照以下规则消除：

1)如果 s 包含长度为 2 的由相同字母组成的子串，那么这些子串会被消除，余下的子串拼成新的字符串。

例如"ABCCBCCCAA"中"CC","CC"和"AA"会被同时消除，余下"AB", "C"和"B"拼成新的字符串"ABBC"。

2)重复上述操作，直到新的字符串不包含相邻的相同字符为止。

例如”ABCCBCCCAA”经过一轮消除得到"ABBC"，再经过一轮消除得到"AC"

输入描述：

第一行输入一个正整数 T(1 <= T <= 50) ， 表示有 T 组测试数据.

对于每组测试数据输入只有一行， 由大写字母组成的字符串 s，长度不超过 100.

输出描述：

对于每组测试数据， 若最后可以把整个字符串全部消除， 就输出 Yes， 否则输出 No.

输入样例：

2

ABCCBA

ABCCCCCBBBBB

输出样例：

Yes

No

【答案及解析】

```
#include <cstdio>
#include <cstring>
char ch[110];
char s[110];
int main(){
  int i, j, t;
  scanf("%d", &t);
  while(t--){
      scanf("%s", ch);
      int f = 1;
      while(f){
          f = 0;
          j = 0;
          int len = strlen(ch);
          for(i = 0; i < len; i++){
              if(ch[i] != ch[i + 1])
              s[j++] = ch[i];
              else if(i != len - 1)
                  f = 1, i++;
          }
          for(i = 0; i < j; i++)
          ch[i] = s[i];
          ch[i] = '\0';
      }
      if(!strlen(ch))
```

```
        printf("Yes\n");
        else
        printf("No\n");
    }
    return 0;
}
```

十五、美丽联合 2018 秋招笔试真题

1、派分糖果

【题目描述】有 N 个孩子站成一排，每个孩子有一个分值。给这些孩子派发糖果，需要满足如下需求：

1、每个孩子至少分到一个糖果

2、分值更高的孩子比他相邻位的孩子获得更多的糖果

求至少需要分发多少糖果？

输入描述:

0,1,0

输出描述:

4

输入示例:

5,4,1,1

输出示例:

7

【答案及解析】

```
public int candy(int[] ratings) {
    int pre = 1, countDown = 0, total = 1;
    for (int i = 1; i < ratings.length; i++) {
        if (ratings[i] >= ratings[i - 1]) {
            if (countDown > 0) {
                total += countDown * (countDown + 1) / 2;
                if (countDown >= pre) {
                    total += countDown - pre + 1;
                }
                pre = 1;
                countDown = 0;
            }
            pre = ratings[i] == ratings[i - 1] ? 1 : pre + 1;
            total += pre;
        }
```

```
        else {
            countDown++;
        }
    }
    if (countDown > 0) {
        total += countDown * (countDown + 1) / 2;
        if (countDown >= pre) {
            total += countDown - pre + 1;
        }
    }
    return total;
}
```

2、字符串的排列

【题目描述】输入一个字符串,按字典序打印出该字符串中字符的所有排列。例如输入字符串 abc,则打印出由字符 a,b,c 所能排列出来的所有字符串 abc,acb,bac,bca,cab 和 cba。

输入描述:

输入一个字符串,长度不超过 9(可能有字符重复),字符只包括大小写字母,例如 ac

输出描述:

[ac, ca]

输入示例:

acc

输出示例:

[acc, cac, cca]

【答案及解析】

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
using namespace std;
class Solution {
public:
  vector<string> Permutation(string str) {
    vector<string> a;
    if(str.empty())
      return a;
    Permutation(a,str,0);
```

```
    sort(a.begin(),a.end());//按照字典序输出
    return a;
   }
   void Permutation(vector<string> &array, string str, int begin)//遍历第 begin 位的所有可能
性
   {
    //一次遍历的结束条件
    if(begin == str.size()-1)
    {
      array.push_back(str);
    }
    for(int i=begin;i<str.size();i++)
    {
      if(i!=begin && str[i] == str[begin])
      {
        continue;//有与 begin 位重复的字符串不进行交换，跳过
      }
      swap(str[i],str[begin]);
      //当 i==begin 时，也要遍历其后面的所有字符
      //当 i!=begin 时，先交换，使第 begin 位取到不同的可能字符，再遍历后面的字符
      Permutation(array,str,begin+1);
      swap(str[i],str[begin]);//为了防止重复的情况，还需要将 begin 处的元素重新换回来
    }
   }
};
int main()
{
  string a = "abc";
  Solution s;
  vector<string> b;
  b = s.Permutation(a);
  for(int i=0;i<b.size();i++)
  {
    cout<<b[i]<<endl;
  }
  return 0;
}
```

十六、欢聚时代 2018 秋招笔试真题

1、计算重复字符串长度

【题目描述】请从字符串中找出至少重复一次的子字符串的最大长度

输入描述:

字符串，长度不超过 1000

输出描述:

重复子串的长度，不存在输出 0

输入示例:

ababcdabcefsgg

输出示例:

3

说明:

abc 为重复的最大子串，长度为 3

【答案及解析】

```
import java.util.*;
public class Main {
    private static int statLen(String X, int k, int j) {
        int cur_len = 0;
        while(k < X.length() && j < X.length() && X.charAt(k) == X.charAt(j)){
            k++;
            j++;
            cur_len++;
        }
        return cur_len;
    }
    // O(n^3)
    public static int naiveLRS(String x){
        int maxlen = 0;
        int length = x.length();
        for(int i = 0;i < length;i++){
            int len = 0;
            int k = i;//第一个游标 k
            //第二个游标 j
            for(int j = i + 1;j < length;j++){
                len = statLen(x, k, j);
                if(maxlen < len){
                    maxlen = len;
                }
```

```
            }
        }
        return maxlen;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String X = sc.nextLine();
        System.out.println(naiveLRS(X));
    }
}
```

牛客网
NOWCODER

校招冲刺集训营

算法通关

招牌题归纳总结，助力斩获30w年薪

BAT都认可的讲师教学

左程云老师直播授课，十年算法刷题经验，学员遍布百度、阿里、腾讯、今日头条、美团等一线互联网名企

最新校招真题精讲

19年秋招最新考题+历年经典题，
完全解读校招套路

金牌助教一对一答疑

ACM金牌教练，丰富的算法和数据结构教学经验，一对一作业辅导

专属优惠券：nianxin30w
(立减200元，数量有限，先到先得)

报名咨询：1440073724
(qq，加时请备注来意哦)

扫码报名