2020

名企笔试面试
求职攻略

技术篇

牛客出品
nowcoder.com

vivo

对话未见的世界

vivo 2020 校园招聘

2020 vivo CAMPUS RECRUITING

HELLO UNKNOWN FUTURE

招聘岗位：

技术研发方向 产品方向 设计方向 市场方向 公共职能方向 制造方向

网申地址：

https://hr.vivo.com/campus

扫码关注vivo最新校招信息

目录

感谢腾讯、字节跳动、华为、百度、网易、京东、vivo、小米、平安科技、度小满、携程、快手、拼多多、顺丰科技、招银网络科技、爱奇艺、搜狐畅游、搜狗、深信服、bilibili、小红书、Shopee、乐信、招行信用卡中心、4399、美团点评、网易互娱等 27 家企业对于本次《2020 名企笔试面试求职攻略》活动的大力支持。

一、前言

1.1 校招介绍

校招，即校园招聘，是企业**面向应届生**专门举办的招聘，一年举办两次，即秋招和春招。秋招，即**秋季校园招聘**，是**面向应届生的最大规模的校园招聘**，一般在每年的秋季举办（8月-10月）；春招，即春季校园招聘，是秋招的补充，规模相较于秋招会小很多，时间一般在春季（3月-5月）。

有些同学的观念是：找工作是毕业前的事情，大四下学期才开始找。结果导致春招的时候才行动，招聘岗位会变少，每个岗位的招聘人数也会很有限。求职要趁早，20届毕业生至少8月份就要开始关注招聘信息，进行网申或者内推了。

1.1.1 秋招时间

7月-8月：内推&校招提前批
9月：正式校招

1.1.2 秋招面向对象

秋招是面向应届生的招聘，今年的秋招是面向2020届毕业生的。
如果是留学生，由于国外的毕业时间跟国内存在差异，一般企业也允许19年毕业的留学生参与20届的校招的。

1.1.3 秋招信息渠道

• 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。

这时候，认准牛客网就可以了。可通过牛客校招指南，一键直达各个公司官网：牛客网>求职>校招日程汇总（**地址**：https://www.nowcoder.com/school/schedule）

• 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。

网站地址：牛客网>求职>实习广场：https://www.nowcoder.com/job/center
移动端地址：牛客APP > 求职

扫一扫，找实习

讨论区：内推消息，由已拿到 offer 的老牛友以及 HR 发布
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>求职> 校招日程：https://www.nowcoder.com/school/schedule

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。
校招助手牛可乐：已建立全国 13 个城市秋招备战群，每日分享校招信息、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


校招助手@牛可乐

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——准备秋招。

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 热点连接网络。**

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一些标记数组，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。如果企业无特殊要求，也可以选择不放照片。

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！

• 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**，绝对是秋招时叩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！

• 项目经历：

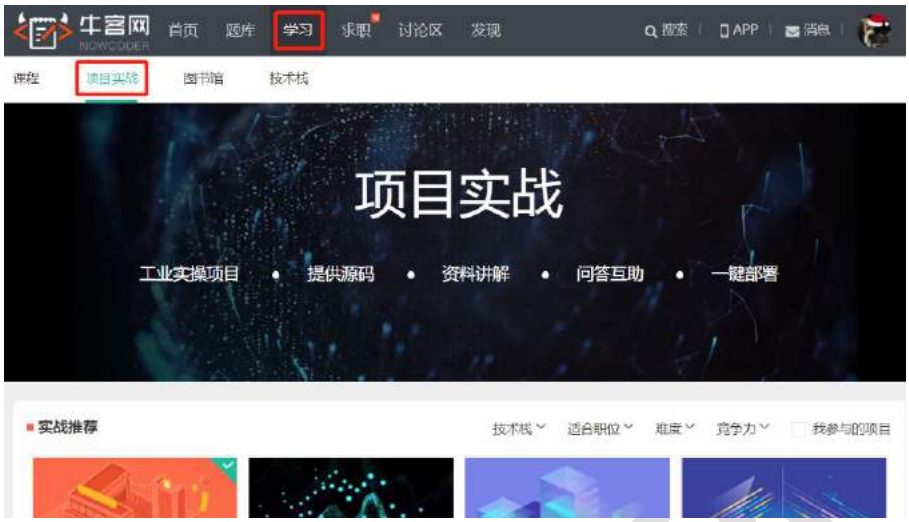
项目经历也是 HR 和面试官会非常看重的一部分！因为项目经历代表着你可能了解的技术栈，好的项目经历能够帮助你渡过简历筛选这一关。

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工作的应届生，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。

加分项：

（1）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。

（2）获奖经历。如果在校期间你参加了一些比赛，并得到了很好的名次，可以将优秀的比赛经历填写上，是一个很好的加分项。书写格式则根据你的具体情况，将最为重要的写在前面，如果是团队类比赛，并且你担任队长或很重要的角色，也可以突出写明。

（3）优质论文。在校期间如发表过论文，可罗列在简历中，是非常好的亮点。

• 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**，直接进行原卷练习。（地址：https://www.nowcoder.com/contestRoom）

• 在线编程题库

地址：牛客网 > 题库 > 在线编程（https://www.nowcoder.com/activity/oj）

《2019 校招真题编程题汇总》

《剑指 offer》

《leetcode 经典编程题》

《前端技能大挑战》

（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 > 题库 > 模拟笔试（https://www.nowcoder.com/mockexam/MockExam）

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

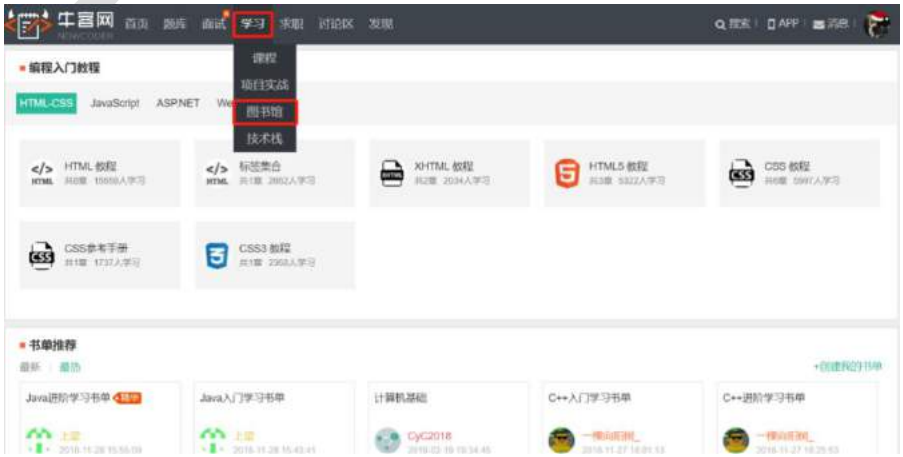
• 企业校招内推机会

（3）多读书，多读学长推荐的书

技术书籍千千万，唯有好书值得看。

牛客图书馆特邀技术大佬为大家推荐书单，帮助后来牛油可以更快更有效率的学习知识。

位置：牛客网>学习>图书馆（地址：https://www.nowcoder.com/library）



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。

• 面试法宝 1：看面经

推荐大家关注“面经大全”和“牛客热帖”小程序，这里面有**上千篇高品质面经汇总**以及众多求职信息分享，帮助你了解心仪公司/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

点一点，百份资料下载

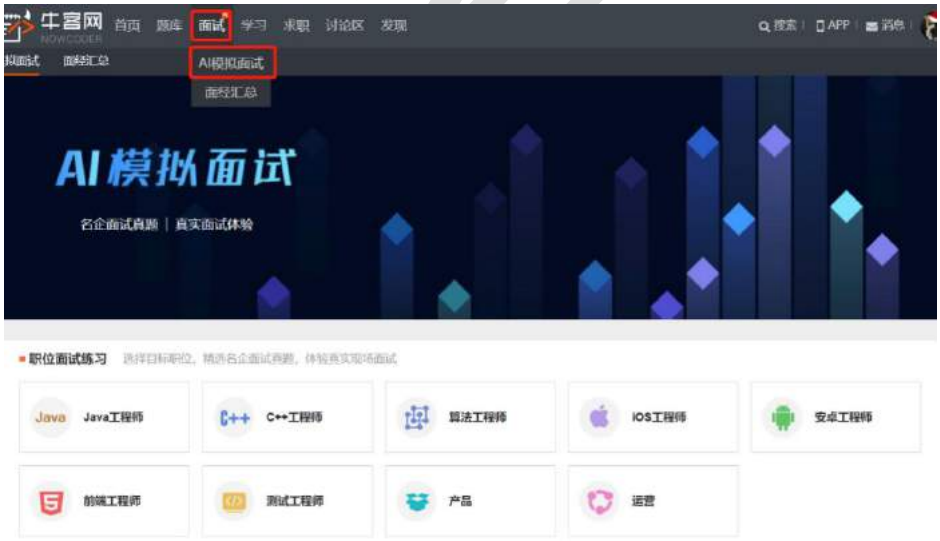
• 面试法宝 2：模拟面试

很多同学**因为缺少面试经验**，在面试中表现得很紧张、胆怯，导致自身实力无从发挥，面试失败。

现在牛客为你推出 **AI 模拟面试系统，利用名企面试真题进行全真模拟面试，全面提升面试能力！**

AI 模拟面试位置：牛客网 > 面试 > AI 模拟面试

链接：https://www.nowcoder.com/interview/ai/index



下面将对各家企业笔试真题进行详细解

二、腾讯 2019 秋招笔试真题

1、小 Q 爬塔

【题目描述】小 Q 正在攀登一座宝塔，这座宝塔很特别，塔总共有 n 层，但是每两层之间的净高却不相同，所以造成了小 Q 爬过每层的时间也不同。如果某一层的高度为 x，那么爬过这一层所需的时间也是 x。

小 Q 还会使用一种魔法,每用一次可以让他向上跳一层或两层,但是每次跳跃后小 Q 都将用完魔法力,必须爬过至少一层才能再次跳跃(你可以认为小 Q 需要跳两次一层才休息，最后也可以跳到塔外即超过塔高，跳是不消耗时间的)。

小 Q 想用最短的时间爬到塔顶,希望你能告诉他最短时间是多少.

输入描述:

第一行一个数 n (n<=10000),表示塔的层数.

接下来的 n 行每行一个数 h(1 <= h <=100),表示从下往上每层的高度.

输出描述:

一个数,表示最短时间

输入样例:

5

3

5

1

8

4

输出样例:

1

【解题思路】

p[i]表示到达第 i 层的最短时间，并且到达第 i 层的方式是爬。

t[i]表示到达第 i 层的最短时间，并且到达第 i 层的方式是跳。

到达第 i 层的方式采用爬还是采用跳。

情况 1.到达第 i 层的方式是爬：

那么到达第 i-1 层的方式可以使爬也可以是跳，从两者中选最小。

p[i]=min{p[i-1],t[i-1]}+a[i]

情况 2.到达第 i 层的方式是跳：

那么可以从第 i-1 层起跳，也可以从第 i-2 层起跳。并且到达第 i-1 层和 i-2 层的方式只能选爬（到第 i 层是跳），所以在两者中选最小。

t[i]=min{p[i-1],p[i-2]}

最后在 p[n]和 t[n]中选最小者做结果

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int p[10005], t[10005];
int main() {
```

```
    int n, m, x;
    cin >> n;
    for(int i = 1; i <= n; i++) {
        cin >> x;
        p[i] = min(p[i - 1], t[i - 1]) + x;
        if(i == 1) continue;
        t[i] = min(p[i - 1], p[i - 2]);
    }
    cout << min(p[n], t[n]) << endl;
    return 0;
}
```

2、妞妞的问题

【题目描述】妞妞公主新得到了一块黑白棋盘。这块棋盘共有 n 行 m 列，任意相邻的两个格子都是不同的颜色（黑或白），坐标为(1,1)的格子是白色的。

这一天牛牛来看妞妞公主时，妞妞公主正望着这块棋盘发呆。牛牛看妞妞公主闷闷不乐的样子，便对妞妞公主说：“只要你告诉我 n 和 m，我能马上算出黑色方块和白色方块的数量。”

“这太简单了。”妞妞公主想了一会儿，“我会在这 n 行 m 列中选择一个左下角坐标为(x0,y0)。右上角坐标为（x1,y1）的矩形，把这个矩形里的共（x1-x0+1）*(y1-y0+1)个方块全部涂白。你还能马上算出黑色方块和白色方块的数量吗？”

“这太简单了。”牛牛自信一笑，“你可以在执行涂白操作后再选一个左下角坐标为（x2,y2），右上角坐标为(x3,y3)的矩形，把这个矩形里的方块全部涂黑。我依然能马上算出黑色方块和白色方块的数量。”

妞妞公主终于惊讶地睁大了眼，于是抛出了她的 T 次提问。

聪明的牛牛当然会做了，但是他想把这个问题交给你，请帮牛牛算出每次提问棋盘的黑白方块数目吧。

输入描述：

第一行一个整数 T，表示妞妞公主一共提问了 T 次。

接下来 3*T 行，

第（1+3*i）行两个整数 n，m。表示第 i 次提问时棋盘的大小；

第（2+3*i）行四个整数 x0，y0，x1，y1。表示第 i 次提问时涂白操作选取的两个坐标。

第（3+3*i）行四个整数 x2，y2，x3，y3。表示第 i 次提问时涂黑操作选取的两个坐标。

1<=T<=10000，1<=x<=n<=1000000000，1<=y<=m<=1000000000，x0<=x1，y0<=y1，x2<=x3，y2<=y3。

输出描述：

共 T 行，每行两个整数分别表示白色方块的数量和黑色方块的数量。

输入样例：

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

3 4
2 1 2 4
1 2 3 3

输出样例：

0 3
3 6
4 8

【解题思路】

格子的个数可以快速维护，然后对应维护出来即可。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
long long x[8], y[8];
int main() {
    int T;
    long long n, m, black, white, a, b, c, d, e;
    scanf("%d", &T);
    while(T--) {
        scanf("%lld%lld", &n, &m);
        black = n * m / 2;
        white = n * m - black;
        for(int i = 0; i <= 3; i++)
            scanf("%lld%lld", &x[i], &y[i]);
        if((x[0] + y[0]) & 1)
            d = ((x[1] - x[0] + 1) * (y[1] - y[0] + 1) + 1) / 2;
        else d = (x[1] - x[0] + 1) * (y[1] - y[0] + 1) / 2;
        white += d;
        black -= d;
        if((x[2] + y[2]) & 1)
            d = (x[3] - x[2] + 1) * (y[3] - y[2] + 1) / 2;
        else d = ((x[3] - x[2] + 1) * (y[3] - y[2] + 1) + 1) / 2;
        white -= d;
        black += d;
        a = max(x[0], x[2]);
        b = max(y[0], y[2]);
        c = min(x[1], x[3]);
        d = min(y[1], y[3]);
        if(c < a || d < b) e = 0LL;
        else{
            if((a + b) & 1)
                e = ((c - a + 1) * (d - b + 1) + 1) / 2;
```

```
            else e = (c - a + 1) * (d - b + 1) / 2;
        }
        white -= e;
        black += e;
        printf("%lld %lld\n", white, black);
    }
    return 0;
}
```

3、小 Q 的最小值序列

【题目描述】小 Q 得到了一个长度为 n 的序列 A，A 中的数各不相同。对于 A 中的每一个数 A_i，求：$min_{(1\le j<i)}|A_i-A_j|$
以及令上式取到最小值的 j（记为 P_i）。若最小值点不唯一，则选择使 A_j 较小的那个。

输入描述:

第一行一个整数 n，第二行 n 个数。

输出描述:

n-1 行，每行 2 个用空格隔开的整数。分别表示当 i 取 2~n 时，对应的 $min_{(1\le j<i)}|A_i-A_j|$和 P_i 的值

输入样例:

```
3
1 5 3
```

输出样例:

```
4 1
2 1
```

【解题思路】

维护一个单调的链表，边读边算边输出，这样每一个数进来时，都能保证链表里的数下标比它小，那么插入后用它两边的数计算答案即可

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
const int inf = 1e9;

struct P {
    int value;
    int pos;
};
P a[N];
int cmp(P x, P y) {
```

```
    return x.value < y.value;
}
int b[N], b2[N], cnt, head, tail, ans1[N], ans2[N];
struct node {
    int value;
    int next, pre;
}e[N];

void init() {
    cnt = 2;
    head = 1, tail = 2;
    e[head].next = tail;
    e[tail].pre = head;
}
void ins(int pos,int x){// 在 pos 位置后面加一个数 X
    e[++cnt].value = x;
    e[cnt].next = e[pos].next;
    e[cnt].pre = pos;
    e[e[pos].next].pre = cnt;
    e[pos].next = cnt;
}
void del(int pos) {
    e[e[pos].next].pre = e[pos].pre;
    e[e[pos].pre].next = e[pos].next;
}
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; ++i) {
        scanf("%d", &a[i].value);
        a[i].pos = i;
    }
    sort(a + 1, a + 1 + n, cmp);
    init();
    for(int i = 1; i <= n; ++i) {
        ins(e[tail].pre, a[i].value);
        b[a[i].pos] = cnt;
        b2[cnt] = a[i].pos;
    }
    for(int i = n; i >= 2; --i) {
        ans1[i] = inf;
        if(e[b[i]].next != tail) {
```

```
                ans1[i] = min(ans1[i], abs(e[b[i]].value - e[e[b[i]].next].value));
                ans2[i] = b2[e[b[i]].next];
            }
            if(e[b[i]].pre != head) {
                if(ans1[i] >= abs(e[b[i]].value - e[e[b[i]].pre].value)) {
                    ans1[i] = abs(e[b[i]].value - e[e[b[i]].pre].value);
                    ans2[i] = b2[e[b[i]].pre];
                }
            }
            del(b[i]);
        }
        for(int i = 2; i <= n; ++i) printf("%d %d\n", ans1[i], ans2[i]);
        return 0;
}
```

三、字节跳动 2019 秋招笔试真题

1、万万没想到之抓捕孔连顺

【题目描述】我叫王大锤，是一名特工。我刚刚接到任务：在字节跳动大街进行埋伏，抓捕恐怖分子孔连顺。和我一起行动的还有另外两名特工，我提议

我们在字节跳动大街的 N 个建筑中选定 3 个埋伏地点。

为了相互照应，我们决定相距最远的两名特工间的距离不超过 D。

我特喵是个天才！ 经过精密的计算，我们从 X 种可行的埋伏方案中选择了一种。这个方案万无一失，颤抖吧，孔连顺！

……

万万没想到，计划还是失败了，孔连顺化妆成小龙女，混在 cosplay 的队伍中逃出了字节跳动大街。只怪他的伪装太成功了，就是杨过本人来了也发现不了的！

请听题：给定 N（可选作为埋伏点的建筑物数）、D（两两埋伏点直接最大的距离）以及可选建筑的坐标，计算在这次行动中，大锤的小队有多少种埋伏选择。

输入描述：

第一行包含空格分隔的两个数字 N 和 D(1 ≤ N ≤ 1000000; 1 ≤ D ≤ 1000000)

第二行包含 N 个建筑物的的位置，每个位置用一个整数（取值区间为[0, 1000000]）表示，从小到大排列（将字节跳动大街看做一条数轴）

输出描述：

一个数字，表示不同埋伏方案的数量。结果可能溢出，请对 99997867 取模

输入样例 1：

4 3

1 2 3 4

输出样例 1：

4

PS：可选方案 (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4)

输入样例 2：

5 19

1 10 20 30 50

输出样例 2：

1

PS: 可选方案 (1, 10, 20)

【解题思路】

可以从左到右遍历所有位置，对于数组中第 i 个建筑，pos[i]是其坐标值，假设第一个特工安插在这里，计算一个最远的安全坐标值 ，X= pos[i] + D，可以通过二分查找右侧最后一个比 X 小的数组下标 j，使得，pos[j] <= X < pos[j + 1]。这样，从 i 到 j 的这段区间，选定 i 点，然后任选两个点，都是合法的组合。计算一个排列组合数 C(j - i, 2)。

【参考代码】

```
MOD = 99997867
class Solution(object):
    def calc(self, N, D, positions):
        count = 0
        for i in range(N):
            # binary search
            X = positions[i] + D
            low = i + 1
            high = N - 1
            j = -1
            while low <= high:
                mid = low + (high - low) // 2
                if positions[mid] > X:
                    high = mid - 1
                else:
                    low = mid + 1
            j = high
            if (j - i - 1) > 0:
                a = (j - i) % MOD
                b = (j - i - 1) % MOD
                count += (a * b) // 2
        return count % MOD


import fileinput
s = Solution()
it = fileinput.input()
rs = next(it).split()
```

```
N = int(rs[0])
D = int(rs[1])
rs = next(it).split()
positions = [int(x) for x in rs]
print(s.calc(N, D, positions))
```

2、毕业旅行

【题目描述】小明目前在做一份毕业旅行的规划。打算从北京出发，分别去若干个城市，然后再回到北京，每个城市之间均乘坐高铁，且每个城市只去一次。由于经费有限，希望能够通过合理的路线安排尽可能的省一些路上的花销。给定一组城市和每对城市之间的火车票的价钱，找到每个城市只访问一次并返回起点的最小车费花销。

输入描述：

城市 n（1<n≤10）

城市间的车票价钱 n 行 n 列的矩阵 m[n][n]

输出描述：

最小车费花销 s

输入样例：

4

0 2 6 5

2 0 4 4

6 4 0 2

5 4 2 0

输出样例：

13

数据范围：1<n≤10

测试数据集：

// 输入

5

0 3 4 2 7

3 0 4 6 3

4 4 0 5 8

2 6 5 0 6

7 3 8 6 0

// 输出

19

// 输入

4

0 30 6 4

30 0 5 10

6 5 0 20

4 10 20 0

// 输出
25
// 输入
4
0 1 2 3
1 0 4 2
2 4 0 6
3 2 6 0
// 输出
11
// 输入
6
0 30 63 52 21 10
30 0 21 25 19 35
63 21 0 17 14 51
52 25 17 0 31 29
21 19 14 31 0 15
10 35 51 29 15 0
// 输出
111
// 输入
7
0 6 5 7 4 3 4
6 0 7 4 3 3 5
5 7 0 4 3 2 8
7 4 4 0 1 3 5
4 3 3 1 0 7 6
3 3 2 3 7 0 5
4 5 8 5 6 5 0
// 输出
22

【解题思路】

贪心算法，动态规划，最小生成树，分治界限

分析：

1.因为最后走的路线为一个环，可以设城市 0 为起点城市。

2.将每个城市看作二进制的一个位(1 代表有，0 代表没有)，则数 k 可以表示一些城市的集合(例如 k=13，二进制表示为 1101，表示城市 0,2,3 的集合)，我们可以求得 k<=2^15-1,令 aim=2^15-1;

3.dp[k][j]表示经过了 k 集合中的所有城市并且以 j 城市为终点的路径的最小值。

则 dp[k][j] = Min{dp[k][j],dp[k-j][i]+dis[i][j] | (0<=i<=n-1 && i 属于集合 k)};(其中 k-j 表示集合 k 中去掉数 j 后的集合(所以 j 应该是集合 k 中的元素))

【参考代码】

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
using namespace std;
const int INF=0x3f3f3f3f;
int dis[12][12],f[1<<12][12],pos[1<<12][12];
//f[st][i]表示处在 i 点，还要访问集合 st 的点各一次后返回 0 点的最短路
int main()
{
    int n;
   // freopen("f.txt","r",stdin);
    while(~scanf("%d",&n)){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                scanf("%d",&dis[i][j]);
                //dis[i][j]=s[i][j];
            }
        }
        int b=1<<(n-1);
        memset(f,-1,sizeof(f));
        for(int i=0;i<n;i++) //边界处理
            f[0][i]=dis[i][0]; //st 是空集，表示都已经访问完了，剩余路程就是从 i 点回到 0 点
        for(int st=1;st<b-1;st++){
            for(int i=1;i<n;i++){
                if(st&(1<<(i-1)))continue; //如果 i 点在集合中没访问，那么 f[st][i]这个状态就
不成立
                int minn=INF;
                for(int j=1;j<n;j++){  //枚举从 i 点要去 j 点
                    if(st&(1<<(j-1))){  //保证 j 点在集合 st 中
                        int tmp=dis[i][j]+f[st^(1<<(j-1))][j]; //把 j 点从 st 集合中去掉
                        if(tmp<minn){  //更新最小值
                            minn=tmp;
                            f[st][i]=tmp;
                            pos[st][i]=j;  //记录路径
                        }
                    }
                }
            }
        }
        int minn=INF;
        for(int k=1;k<n;k++){
```

```
            int tmp=f[(b-1)^(1<<(k-1))][k]+dis[0][k];
            if(tmp<minn)
                minn=tmp,f[b-1][0]=tmp,pos[b-1][0]=k;;
        }
        printf("%d\n",f[b-1][0]);

//        printf("0\n");  //打印路径
//        for(int st=b-1,next=0;st;){
//            next=pos[st][next];
//            printf("%d\n",next);
//            st=st^(1<<(next-1));
//        }
//        printf("0\n");
    }
    return 0;
}
```

3、雀魂启动

【题目描述】小包最近迷上了一款叫做雀魂的麻将游戏，但是这个游戏规则太复杂，小包玩了几个月了还是输多赢少。

于是生气的小包根据游戏简化了一下规则发明了一种新的麻将，只留下一种花色，并且去除了一些特殊和牌方式（例如七对子等），具体的规则如下：

1、总共有 36 张牌，每张牌是 1~9。每个数字 4 张牌。

2、你手里有其中的 14 张牌，如果这 14 张牌满足如下条件，即算作和牌

a. 14 张牌中有 2 张相同数字的牌，称为雀头。

b. 除去上述 2 张牌，剩下 12 张牌可以组成 4 个顺子或刻子。顺子的意思是递增的连续 3 个数字牌（例如 234,567 等），刻子的意思是相同数字的 3 个数字牌（例如 111,777）

例如：

1 1 1 2 2 2 6 6 6 7 7 7 9 9 可以组成 1,2,6,7 的 4 个刻子和 9 的雀头，可以和牌

1 1 1 1 2 2 3 3 5 6 7 7 8 9 用 1 做雀头，组 123,123,567,789 的四个顺子，可以和牌

1 1 1 2 2 2 3 3 3 5 6 7 7 9 无论用 1 2 3 7 哪个做雀头，都无法组成和牌的条件。

现在，小包从 36 张牌中抽取了 13 张牌，他想知道在剩下的 23 张牌中，再取一张牌，取到哪几种数字牌可以和牌。

输入描述:

输入只有一行，包含 13 个数字，用空格分隔，每个数字在 1~9 之间，数据保证同种数字最多出现 4 次。

输出描述:

输出同样是一行，包含 1 个或以上的数字。代表他再取到哪些牌可以和牌。若满足条件的有多种牌，请按从小到大的顺序输出。若没有满足条件的牌，请输出一个数字 0

输入样例 1：

1 1 1 2 2 2 5 5 5 6 6 6 9

输出样例 1：

9

输入样例 2：

1 1 1 1 2 2 3 3 5 6 7 8 9

输出样例 2：

4 7

输入样例 3：

1 1 1 2 2 2 3 3 3 5 7 7 9

输出样例 3：

0

【解题思路】

因为没有例如七对等特殊牌型，所以一张牌所属的结果一定是雀头/刻子/顺子三种之一，所以先将每种牌的数量统计出来，然后枚举给哪一种牌加一张。加上之后使用递归的方法去枚举每种牌是属于一个哪种结果。随后减去这种结果相应的牌的数量后继续递归。当最后发现递归完了 14 张牌，即可认为开始枚举的牌是一个合理的答案。另外要注意，如果开始给的 13 张牌已经有某种牌出现 4 次，就不可以再枚举它了。

【参考代码】

```
a = input().split()
alist = [int(x) for x in a]

#alist = [1, 1, 1, 2, 2, 2, 6, 6, 6, 7, 7, 7, 9]

He_kinds = []

# 判断是否和牌
def isHe(cardlist):
    l = len(cardlist)
    if l == 0:
        return True
    count0 = cardlist.count(cardlist[0])

    # 没出现过雀头，且第一个数字出现的次数 >= 2，且去掉雀头剩下的能组成和牌
    if l % 3 != 0 and count0 >= 2 and isHe(cardlist[2:]) == True:
        return True
    # 如果第一个数字出现次数 >= 3，且去掉这个刻子后剩下的能和牌
    if count0 >= 3 and isHe(cardlist[3:]) == True:
        return True
    # 如果存在顺子，且移除顺子后剩下的能和牌
    if cardlist[0]+1 in cardlist and cardlist[0]+2 in cardlist:
        c_cardlist = cardlist[1:]
        c_cardlist.remove(cardlist[0]+1)
        c_cardlist.remove(cardlist[0]+2)
```

```
        if isHe(c_cardlist) == True:
            return True
    # 以上条件都不满足，则不能和牌
    return False

# 最多有 9 种抽牌方法可以和牌，计算每一种能不能和牌
for i in range(1, 10):
    if isHe(sorted(alist + [i])) == True:       # 如果这种抽牌方式可以和牌
        He_kinds.append(i)                      # 加入和牌类型列表

print(len(He_kinds))            # 输入和牌方式有多少种
#print(He_kinds)
```

四、华为 2019 秋招笔试真题

1、整数反转求和

【题目描述】请您写一个 reverseAdd 函数， 该函数根据输入的两个正整数 a 和 b， 然后分别将它们的数字按照高位在右边的方式反转后求和。

例如，reverseAdd(123, 456) == 321 + 654 == 975

输入描述:

函数原型：int reverseAdd (int a, int b);

输入：

输入的 a,b 参数均为有效取值范围[1, 70000]区间上的正整数。

100 和 200 反转后的值为 1 和 2（前导 0 被忽略）

输出描述:

输出：

通过函数返回值输出结果。

若输入的 a 或 b 参数超出了取值范围（小于 1 或者大于 70000），则应输出-1；

否则应按照要求输出数字反转后的和。

注意：最终交付的函数代码中不要向控制台打印输出任何信息。

输入样例：

123,456

输出样例：

975

【解题思路】

二分答案，然后 check 这个答案的 rank，这一步可以用 two pointers O(n)解决。

复杂度 O(nlogn)

对整数进行拆分之后翻转，然后求和即可。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string m,n;
    while(cin>>m>>n)
    {
        int x=0,y=0,i;
        if(m[0]=='0'&&n[0]=='0')
            break;
        for(i=m.size()-1;i>=0;i--)
            x=x*10+m[i]-'0';
        for(i=n.size()-1;i>=0;i--)
            y=y*10+n[i]-'0';
        cout<<x+y<<endl;
    }
    return 0;
}
```

2、消除重复数字

【题目描述】给定一个正整数，给出消除重复数字以后最大的整数

输入描述:

正整数，注意考虑长整数

输出描述:

消除重复数字以后的最大整数

输入样例：

423234

输出样例：

432

【解题思路】

指针 i，j 从数字字符串的首地址开始遍历，j 作为查重的指针（在 [0,i] 之间的字符子串）

如果出现重复数字，即 str[j] = str[i]，那么根据题目要求，是消除重复数字以后的最大整数，就要做一个比较，判断是移除 i 位置的数字还是 j 位置的数字，如果 str[j+1] > str[j]，应该移除 j 位置的重复数字，否则移除 i 位置的重复数字以保证局部最大整数

【参考代码】

```
#include <iostream>
#include <string>
using name space std;
```

```
int main()
{
    string s;
    while(cin>>s)
    {
        string res;
        res = s[0];
        for (int i = 1; i < s.size(); i++)
        {
            if (res.find(s[i]) == string :: npos)
            {
                res + = s[i];
            }
            else
            {
                int t = res.find(s[i]);
                if ((t+1) < res.size())
                {
                    if (res[t] < res[t+1])
                    {
                        res.erase(t, 1);          // 移走并且调整
                        res + = s[i];
                    }
                }
            }
        }
        cout<< res <<endl;
    }
    return 0;
}
```

3、仿 LISP 运算

【题目描述】LISP 语言唯一的语法就是括号要配对。
形如 (OP P1 P2 ...)，括号内元素由单个空格分割。
其中第一个元素 OP 为操作符，后续元素均为其参数，参数个数取决于操作符类型
注意：参数 P1, P2 也有可能是另外一个嵌套的 (OP P1 P2 ...)
当前 OP 类型为 add / sub / mul / div（全小写），分别代表整数的加减乘除法
其中 add / mul 参数个数 2 或以上，sub / div 参数个数为 2
举例:
- 输入：(mul 3 -7) 输出：-21

- 输入：(add 1 2 3) 输出：6
- 输入：(sub (mul 2 4) (div 9 3)) 输出：5
- 输入：(div 1 0) 输出：error

题目涉及数字均为整数，可能为负；不考虑 32 位溢出翻转

除零错误时，输出 "error"，除法遇除不尽，取整，即 3/2 = 1

输入描述:

合法 C 字符串，字符串长度不超过 512；用例保证了无语法错误.

输出描述:

合法 C 字符串，字符包括'0'-'9'及负号'-'或者"error"

输入样例：

(add 1 2 3)

输出样例：

6

【解题思路】

按照题意模拟语法运算规则。

【参考代码】

```
import java.util.Scanner;
import java.util.Stack;
public class Main {
    //(sub (mul 2 4) (div 9 3))
   public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String line = scanner.nextLine();
        calculatorLISP(line);
        scanner.close();
    }
    private static void calculatorLISP(String str) {
         Stack<Integer> numStack = new Stack<Integer>();
         Stack<String> operStack =new Stack<String>();
         int mark = 0;
         int paramOne = 0;
         int paramTwo = 0;
         for(int i = 0; i < str.length() ; i++) {
             char c = str.charAt(i);
             if(c == '('){
                 operStack.push(str.substring(i+1,i+4));
                 i+=4;
                 mark = i+1;
             }else if(c == ')'){
                 if(mark < i){
                     numStack.push(Integer.valueOf(str.substring(mark,i)));
```

```
                    i++;
                    mark = i+1;
                }
                paramTwo = numStack.pop();
                paramOne = numStack.pop();
                calc(numStack,operStack,paramOne,paramTwo);
            }else{
                if(c == ' '){
                    if(mark < i){
                        numStack.push(Integer.valueOf(str.substring(mark,i)));
                        mark = i + 1;
                    }
                }
            }
    }
     while(!operStack.isEmpty()) {
            paramTwo = numStack.pop();
            paramOne = numStack.pop();
            calc(numStack,operStack,paramOne,paramTwo);
        }
        System.out.println(numStack.pop().toString());
}
    public static void calc(Stack<Integer> numStack,Stack<String> operStack,int paramOne,
int paramTwo){
        switch(operStack.pop()) {
        case "add":
            numStack.push(paramOne + paramTwo);
            break;
        case "sub":
            numStack.push(paramOne - paramTwo);
            break;
        case "mul":
            numStack.push(paramOne * paramTwo);
            break;
        case "div":
            if(paramTwo == 0) {
                System.out.println("error");
            }else{
                numStack.push(paramOne / paramTwo);
                }
            break;
        }
```

```
    }
}
```

四、百度 2019 秋招笔试真题

1、混战世界

【题目描述】战乱年代，整个世界各个军阀的兵团混战，你是 PZ 军团的战略参谋，你手下有 n（保证为 3 的倍数）个士兵，第 i 个士兵的物理攻击数值为 Ai，魔 法攻击数值为 Bi，你需要将这些士兵三等分为三个连，第一个连需要去物理空间参加物理对抗战争，战斗力估值 W1 为士兵的物理攻击数值之 和；第二个连需要去魔法空间参加魔法对抗战争，战斗力估值 W2 为士兵的魔法攻击数值之和；第三个连需要去虚幻空间参加物理魔法兼备的综 合对抗战争，战斗力估值 W3 为所有士兵的物理攻击数值、魔法攻击数值之和除以 2。你希望 W1+W2+W3 最大，这样才最有可能胜利。

输入描述:

第一行一个整数 n，保证为 3 的倍数。（3≤n≤100000） 第二行 n 个整数，第 i 个数表示 Ai。 第三行 n 个整数，第 i 个数表示 Bi。（1≤Ai， Bi≤1000）

输出描述:

一个小数，表示最大数值和，保留两位小数（四舍五入）。

输入样例:

```
6
1 7 3 4 5 9
2 3 9 4 3 3
```

输出样例:

```
35.00
```

【解题思路】

设一个人的物理值为 A，魔法值为 B。

派去一连可得 A 的贡献，二连可得(A+B)/2，三连可得 B。

去一连与去二连相比差了 A-(A+B)/2 = (A-B)/2，同理，去二连比去三连也差了(A-B)/2。

这样，可以根据每个人的 A-B 数值进行排序，较大者去一连，较小者去三连，中间的去二连。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
const int N = 100010;
int a[N], b[N], id[N], n;
inline bool cmp(int x, int y) {
    return a[x] - b[x] > a[y] - b[y];
}
int main() {
    scanf("%d", &n);
```

```
    for (int i = 1; i <= n; ++i)
        scanf("%d", a + i);
    for (int i = 1; i <= n; ++i)
        scanf("%d", b + i);
    for (int i = 1; i <= n; ++i)
        id[i] = i;
    sort(id + 1, id + n + 1, cmp);
    int k = n / 3;
    long long ans = 0;
    for (int i = 1; i <= n; ++i) {
        if (i <= k) ans += a[id[i]] * 2;
        else if (i <= k * 2) ans += a[id[i]] + b[id[i]];
        else ans += b[id[i]] * 2;
    }
    printf("%.2f\n", ans / 2.0);
    return 0;
}
```

2、字符串计数

【题目描述】给定一个仅由小写字母组成且长度不超过 106 的字符串，将首字符移到末尾并记录所得的字符串，不断重复该操作，虽然记录了无限个字符串， 但其中不同字符串的数目却是有限的，那么一共记录了多少个不同的字符串？

输入描述:

输入给定的字符串。

输出描述:

输出记录的不同字符串的数目。

输入样例:

abab

输出样例:

2

【解题思路】

利用 kmp 算法的 next 数组，可以求出字符串的最小循环周期 T，这就是答案。

也可以将字符串在后面复制一次，用字符串 hash 和 std::map 统计。

【参考代码】

```
#include <bits/stdc++.h>
#define N 1000007
using namespace std;
char s[N];
int nxt[N];
```

```
int main() {
    scanf("%s",s);
    int n=strlen(s);
    nxt[0]=-1;
    for(int i=0,j=-1; i<n;)
        if(j==-1 || s[i]==s[j])nxt[++i]=++j;
        else j=nxt[j];
    if(n%(n-nxt[n]))printf("%d",n);
    else printf("%d",n-nxt[n]);
}
```

3、表兄弟

【题目描述】每个人的家族关系可以表示成为一棵树，显而易见，家族关系中不会有环的存在。 假设家族关系树中的每条边的权值都为 1， 我们称 x 是 y 的 k 祖先，当且仅当在家族关系树中 x 是 y 的祖先且 x 到 y 的距离是 k。 我们称 x 和 y 为 k 表兄弟，当且仅当 x 和 y 的 k 祖先为同一人。 现在给出一个家族关系，共有 n 个家族成员，因为历史记录的缺失，所有可能并不知道有些人的父亲是谁(最后的结果可能是 若干个树)，给出 m 个询问，每个询问包含一个 x 和一个 k，请你找出 x 成员的 k 表兄弟的数量。

输入描述:

输入第一行是一个整数 n(1≤n≤105）表示家族关系树中成员数量。 第二行有 n 个数，中间用空格隔开，第 i 个数 fi 表示 i 号成员的父亲为 fi，如果 fi 为 0，表示不知道 i 号成员父亲是谁。 第三行包含一个整数 m（1≤m≤105） ，表示询问的数量。 接下来有 m 行，每行有两个正整数 x，k，中间用空格隔开，表示询问 x 成员的 k 表兄弟有多少个。

输出描述:

对于每一个询问，输出一个整数，表示 x 成员的 k 表兄弟数量，中间用空格隔开。

输入样例:

```
10
0 1 2 0 15 6 3 3 0
1 0
9 1
5 4
2 2
10 1
8 4
7 1
3 2
5 2
4 2
3 1
```

输出样例:

```
1 0 0 0 0 0 1 0 0 0
```

【解题思路】

利用倍增法可以在 O(logn)的时间内找到 u 点的 k 祖先 anc。

问题转化为求 anc 的所有 k 后代。

可以先求出这个森林的 dfs 序，然后将同一深度的点存到一个 std::vector 中。

对于一个祖先 anc，在深度为 dep[u]的 vector 中二分查找 in[anc]和 out[anc]，即可求得所有的 k 后代，减去 u 本身就是 u 的 k 表兄弟数量。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e5+20;
vector <int> v[N],d[N];
int n,m,pa[N],par[N][23],h[N],st[N],fi[N],x1,y2,x,y,p,cnt=0;
void dfs(int s) {
    int y;
    st[s]=++cnt;
    d[h[s]].push_back(st[s]);
    for(int i=0; i<v[s].size(); i++) {
        y=v[s][i];
        h[y]=h[s]+1;
        dfs(y);
    }
    fi[s]=cnt;
}
int main() {
    ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    cin >> n;
    for(int i=1; i<=n; i++) {
        cin >> pa[i];
        if(pa[i]!=0) {
            v[pa[i]].push_back(i);
            par[i][0]=pa[i];
        } else
            par[i][0]=i;
    }
    cnt=0;
    for(int i=1; i<=n; i++) {
        if(pa[i]==0) {
            h[i]=0;
            dfs(i);
        }
    }
```

```
    for(int i=1; i<=20; i++) {
        for(int j=1; j<=n; j++) {
            par[j][i]=par[par[j][i-1]][i-1];
        }
    }
    cin >> m;
    for(int i=1; i<=m; i++) {
        cin >> x >> p;
        y=x;
        for(int j=20; j>=0; j--)
            if(p & (1<<j))
                y=par[y][j];
        if(h[x]-p<0) {
            cout << 0 << ' ';
            continue;
        }
        x1=st[y];
        y2=fi[y];
        x1=lower_bound(d[h[x]].begin(),d[h[x]].end(),x1)-d[h[x]].begin();
        y2=upper_bound(d[h[x]].begin(),d[h[x]].end(),y2)-d[h[x]].begin();
        x1=y2-x1-1;
        cout << x1 << ' ';
    }
    return 0;
}
```

六、网易 2019 秋招笔试真题

1、翻转翻转

【题目描述】给定一个 N*M 的矩阵，在矩阵中每一块有一张牌，我们假定刚开始的时候所有牌的牌面向上。
现在对于每个块进行如下操作：
> 翻转某个块中的牌，并且与之相邻的其余八张牌也会被翻转。

XXX
XXX
XXX

如上矩阵所示，翻转中间那块时，这九块中的牌都会被翻转一次。
请输出在对矩阵中每一块进行如上操作以后，牌面向下的块的个数。

输入描述:

输入的第一行为测试用例数 t(1 <= t <= 100000)，
接下来 t 行，每行包含两个整数 N,M(1 <= N, M <= 1,000,000,000)

输出描述:

对于每个用例输出包含一行，输出牌面向下的块的个数

输入样例：

5
1 1
1 2
3 1
4 1
2 2

输出样例：

1
0
1
2
0

【解题思路】

先考虑 n = m = 1 的情况，翻转一次，故输出 1

我们令输入的 n <= m

当 n = 1,m > 1 时，最边上两块会被翻转两次，中间的会被翻转三次，故输出（m - 2）

当 2 <= n <= m 时，四个角会被翻转四次，四边上除了角外的部分会被翻转六次，中间剩余的部分会被翻转九次，故输出(n - 2)(m - 2)

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
int main() {
        int t;
        scanf("%d", &t);
        while(t--) {
                long long n, m;
                scanf("%lld%lld", &n, &m);
                if(n > m) swap(n, m);
                if(n == 1 && m == 1){
                        printf("1\n");
                        continue;
                }
                if(n == 1){
                        printf("%lld\n", m - 2);
                        continue;
                }
                printf("%lld\n", (n - 2) * (m - 2));
        }
```

```
        return 0;
}
```

2、社团主席选举

【题目描述】随着又一届学生的毕业，社团主席换届选举即将进行。
一共有 n 个投票者和 m 个候选人，小易知道每一个投票者的投票对象。但是，如果小易给某个投票者一些糖果，那么这个投票者就会改变他的意向，小易让他投给谁，他就会投给谁。
由于小易特别看好这些候选人中的某一个大神，这个人的编号是 1，所以小易希望能尽自己的微薄之力让他当选主席，但是小易的糖果数量有限，所以请你帮他计算，最少需要花多少糖果让 1 号候选人当选。某个候选人可以当选的条件是他获得的票数比其他任何候选者都多。

输入描述:

第一行两个整数 n 和 m，表示投票者的个数和候选人的个数。
接下来 n 行，每一行两个整数 x 和 y，x 表示这个投票者的投票对象，y 表示需要花多少个糖果让这个人改变意向。
满足 1 <= n, m <= 3000，1 <= x <= m，1 <= y <= 109。

输出描述:

一个整数，糖果的最小花费。

输入样例 1：

1 2
1 20

输出样例 1：

0

输入样例 2：

5 5
2 5
3 5
4 5
5 6
5 1

输出样例 2：

6

【解题思路】

暴力枚举这个人需要多少人支持才能当选，每次枚举时，超出这个数量以及和这个数量相等的人一定需要被改变，之后数量不足，再从剩下的人中选择。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 3005;
```

```
vector<int> G[N];
int vis[N];
int main() {
	int n, m;
	scanf("%d%d", &n, &m);
	ll ans = 1e18;
	for (int i = 0;i < n;i++) {
		int p, c;
		scanf("%d%d", &p, &c);
		G[p].push_back(c);
		vis[p]++;
	}
	int mx = 0;
	for (int i = 1;i <= m;i++) {
		if (vis[i] > mx) {
			mx = vis[i];
		}
	}
	int cnt = 0;
	for (int i = 1;i <= m;i++) {
		if (vis[i] == mx) cnt++;
	}
	if (cnt == 1 && vis[1] == mx) return 0*puts("0");
	for (int i = 1; i <= m; i++) sort(G[i].begin(), G[i].end());
	for (int i = n; i >= 1; i--) {
		vector<int> r;
		int num = i - vis[1];
		ll sum = 0;
		for (int j = 1; j <= m; j++) {
			if (vis[j] >= i && j != 1) {
				int tmp = vis[j] - i + 1;
				for (int k = 0; k < min(tmp, (int)G[j].size()); k++) {
					r.push_back(G[j][k]);
				}
			}
		}
		for (int j = 0; j < r.size(); j++) {
			num--;
			sum += r[j];
		}
		if (num <= 0) {
			ans = min(ans, sum);
```

```
                        continue;
                }
                r.clear();
                for (int j = 1; j <= m; j++) {
                        if (vis[j] >= i && j != 1) {
                                int tmp = vis[j] - i + 1;
                                if (tmp >= G[j].size()) continue;
                                for (int k = tmp; k < G[j].size(); k++) {
                                        r.push_back(G[j][k]);
                                }
                        }
                }
                for (int j = 1; j <= m; j++) {
                        if (vis[j] < i && j != 1) {
                                for (int k = 0; k < G[j].size(); k++) {
                                        r.push_back(G[j][k]);
                                }
                        }
                }
                sort(r.begin(), r.end());
                for (int j = 0;j < r.size();j++) {
                        num--;
                        sum += r[j];
                        if (num <= 0) break;
                }
                if (num <= 0) ans = min(ans,sum);
        }
        cout << ans << endl;
        return 0;
}
```

3、香槟塔

【题目描述】节日到啦，牛牛和妞妞邀请了好多客人来家里做客。

他们摆出了一座高高的香槟塔，牛牛负责听妞妞指挥，往香槟塔里倒香槟。

香槟塔有个很优雅的视觉效果就是如果这一层的香槟满了，就会从边缘处往下一层流去。

妞妞会发出两种指令，指令一是往第 x 层塔内倒体积为 v 的香槟，指令二是询问第 k 层塔香槟的体积为多少。

告诉你香槟塔每层香槟塔的初始容量，你能帮牛牛快速回答妞妞的询问吗？

输入描述:

第一行为两个整数 n，m。表示香槟塔的总层数和指令条数。

第二行为 n 个整数 ai，表示每层香槟塔的初始容量。

第三行到第 2+m 行有两种输入，一种输入是“2 x v”表示往第 x 层倒入体积为 v 的香槟；另一种输入是“1 k”表示询问第 k 层当前有多少香槟。

1 <= n, m <= 1000。

1 <= n ,m <= 200000，1 <= ai ,v <= 1000000000。

输出描述:

对于每个询问，输出一个整数，表示第 k 层香槟的容量。

输入样例 1：

```
1  2
8
2  1  9
1  1
```

输出样例 1：

```
8
```

输入样例 2：

```
5  4
1  2  2  10  1
1  3
2  2  5
2  4  3
1  4
```

输出样例 2：

```
0
4
```

【解题思路】

每层塔建立一个 mark 标记指向离他最近的未装满的层，执行 2 操作时对当前层的 mark 进行添加，如果加满移至下一个 mark。使用并查集路径压缩的思想不断把 mark 下移。执行 1 操作时直接输出结果。

【参考代码】

```
#include <bits/stdc++.h>
const int MAXN =   200015;
using namespace std;
int mark[MAXN], a[MAXN], b[MAXN];
int _find(int x) {
        if(x == 0) return x;
        if(a[x] != b[x]) return x;
        return mark[x] = mark[x + 1] = _find(mark[x + 1]);
}
int main() {
        int n, m, x, y, d;
        scanf("%d%d", &n, &m);
        for(int i = 1; i <= n; i++) {
                scanf("%d", &a[i]);
```

```
                b[i] = 0;
                mark[i] = i;
        }
        int op;
        while(m--) {
                scanf("%d", &op);
                if(op == 2) {
                        scanf("%d%d", &x, &y);
                        while(y) {
                                x = _find(x);
                                if(x == 0)break;
                                d = min(a[x] - b[x], y);
                                b[x] += d;
                                y -= d;
                        }
                }
                else {
                        scanf("%d", &x);
                        printf("%d\n", b[x]);
                }
        }
        return 0;
}
```

七、京东 2019 秋招笔试真题

1、完全多部图

【题目描述】给定一张包含 N 个点、M 条边的无向图，每条边连接两个不同的点，且任意两点间最多只有一条边。对于这样的简单无向图，如果能将所有点划分成若干个集合，使得任意两个同一集合内的点之间没有边相连，任意两个不同集合内的点之间有边相连，则称该图为完全多部图。现在你需要判断给定的图是否为完全多部图。

输入描述：

包含多组数据，每组输入格式为：

第一行包含两个整数 N 和 M，1≤N≤1000，0≤M≤N(N-1)/2；

接下来 M 行，每行包含两个整数 X 和 Y，表示第 X 个点和第 Y 个点之间有一条边，1≤X，Y≤N。

输出描述：

每组输出占一行，如果给定的图为完全多部图，输出 Yes，否则输出 No。

输入样例：

5 7

1 3

1 5
2 3
2 5
3 4
4 5
3 5

输出样例：

Yes

【解题思路】

根据给的边，可以判断哪些点应该属于一个集合。如果存在一个点同时属于多个集合，则不是完全多部图。

复杂度 O(n^2)

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000 + 5;
int n, m;
int a[N][N];
int belong[N], cnt;
int main() {
    scanf("%d%d", &n, &m);
    while(m--) {
        int u, v;
        scanf("%d%d", &u, &v);
        a[u][v] = a[v][u] = 1;
    }
    for(int u = 1; u <= n; ++u) {
        if(belong[u]) continue;
        belong[u] = ++cnt;
        for(int v = u + 1; v <= n; ++v) {
            if(!a[u][v]) {
                if(belong[v] != 0) {
                    puts("No");
                    return 0;
                }
                belong[v] = cnt;
            }
        }
    }
    for(int u = 1; u <= n; ++u) {
        for(int v = u + 1; v <= n; ++v) {
```

```
            if(belong[u] == belong[v] && a[u][v] ||
               belong[u] != belong[v] && !a[u][v]) {
                puts("No");
                return 0;
            }
        }
    }
    puts("Yes");
    return 0;
}
```

2、对比

【题目描述】现有 n 个物品，每个物品有三个参数，定义 i 物品不合格品的依据是:若存在物品 j,且，则称 i 物品为不合格品。

现给出 n 个物品的 a,b,c 参数，请你求出不合格品的数量。

输入描述：

第一行包含一个整数 n(1<=n<=500000),表示物品的数量。接下来有 n 行，每行有三个整数，表示第 i 个物品的三个参数

输出描述：

输出包含一个整数，表示不合格品的数量。

输入样例：

3

1 4 2

4 3 2

2 5 3

输出样例：

1

样例解释：

物品 1 的 a,b,c 均小于物品 3 的 a,b,c,因此 1 为不合格品。

【解题思路】

三维偏序问题。

先对第一维排序，然后降序处理。对于第二三维，可以用一个平衡树（可以使用 std::multiset）维护一个外围包络，这样可以快速判断一个点的右上方是否有点（使用 lower_bound），有的话，即为不合格产品。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 500000 + 5;
int n;
```

```
struct node {
    int a, b, c;
    void read() {
        scanf("%d%d%d", &a, &b, &c);
    }
    bool operator < (const node &rhs) const {
        return a < rhs.a;
    }
}a[N];
int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; ++i) {
        a[i].read();
    }
    sort(a, a+n);
    queue<pair<int,int> > Q;
    int last = -1, ans = 0;
    multiset<pair<int,int> > s;
    for(int i = n - 1; i >= 0; --i) {
        if(a[i].a != last) {
            while(Q.size()) {
                auto u = Q.front();
                Q.pop();
                auto it = s.lower_bound(u);
                if(it == s.end() || it->second < u.second) {
                    it = s.insert(u);
                    while(it != s.begin()) {
                        auto tmp = it;
                        --tmp;
                        if(tmp->second <= u.second) {
                            s.erase(tmp);
                        }else {
                            break;
                        }
                    }
                }
            }
            last = a[i].a;
        }
        pair<int,int> p(a[i].b, a[i].c);
        auto it = s.lower_bound(p);
        if(it == s.end() || it->second <= p.second) {
```

```
            Q.push(p);
        }else {
            ++ans;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

3、相似字符串

【题目描述】对于仅由小写字母组成的字符串 A 和 B，如果存在一个小写字母 a 到 z 的排列，使得将 A 中所有字母 a 替换为排列的第一个字母，所有字母 b 替换为排列的第二个字母……所有字母 z 替换为排列的最后一个字母之后，A 和 B 完全相同，那么称字符串 A 和 B 相似，如 abcc 和 xyaa。现在给定仅由小写字母组成且长度不超过 105 的字符串 S 和 T，求 S 中有多少子串与 T 相似？

输入描述：

第一行和第二行分别输入字符串 S 和 T。

输出描述：

输出字符串 S 中与 T 相似的子串数目。

输入样例：

ababcb

xyx

输出样例：

3

样例解释：

S 中与 T 相似的子串分别是 aba、bab、bcb，总共 3 个。

【解题思路】

统计 T 的每个字符的数量，存为长 26 的 vector，排序后，作为模式串。

使用滑动窗口统计 S 中长度为|T|的子串中每种字符的数量，拷贝一份并排序后，与模式串相同即算匹配成功。

复杂度 O(|S|*26*log26)

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100000 + 5;
int n, m;
char s[N], t[N];
int main() {
    scanf("%s%s", s, t);
    n = strlen(s);
```

```
    m = strlen(t);
    if(n < m) {
        puts("0");
        return 0;
    }
    vector<int> cnt(26), ptn(26);
    for(int i = 0; i < m; ++i) {
        ++cnt[s[i]-'a'];
        ++ptn[t[i]-'a'];
    }
    sort(ptn.begin(), ptn.end());
    int ans = 0;
    for(int i = m; i < n; ++i) {
        vector<int> tmp = cnt;
        sort(tmp.begin(), tmp.end());
        if(tmp == ptn) ++ans;
        ++cnt[s[i]-'a'];
        --cnt[s[i-m]-'a'];
    }
    vector<int> tmp = cnt;
    sort(tmp.begin(), tmp.end());
    if(tmp == ptn) ++ans;
    printf("%d\n", ans);
    return 0;
}
```

八、vivo 2019 秋招笔试真题

1、字符串排序

【题目描述】请对一组字符串进行排序，字符串由大小写字母和数字组成，需要满足以下比较规则

1、长度不同时，长度较短在排前面

2、长度相同时，按照字典顺序排列(AaBb - Zz，0-9 顺序)，即大写字母在小写字母前，数字排在字母后。

要求时间复杂度为 O(nlogn)。

比如：

abc Abc 123 1 1bc CBD abcd a

排序后结果为：

a 1 Abc abc CBD 1bc 123 abcd

【解题思路】

按照题目要求实现 cmp 函数，调用 sort 函数。

【参考代码】

```
public class Main {
    public static void main(String[] args) {
        String[] input = {"abc", "Abc", "123", "1", "1bc", "CBD", "abcd", "a" };
        sort(input);
        for (String s : input) {
            System.out.println(s);
        }
    }

    private static void sort(String[] input) {
        quickSort(input, 0, input.length - 1);
    }

    private static void quickSort(String[] data, int left, int right) {
        if (left > right) {
            return;
        }
        int i, j;
        String t, temp;
        temp = data[left];
        i = left;
        j = right;
        while (i != j) {
            while (cmp(data[j], temp) >= 0 && i < j) {
                j--;
            }
            while (cmp(data[i], temp) <= 0 && i < j) {
                i++;
            }
            if (i < j) {
                t = data[i];
                data[i] = data[j];
                data[j] = t;
            }
        }
        data[left] = data[i];
        data[i] = temp;
        quickSort(data, left, i - 1);
        quickSort(data, i + 1, right);
    }
```

```
    private static int cmp(String left, String right) {
        if (left.length() != right.length()) {
            return left.length() - right.length();
        }
        int len = left.length();
        for (int i = 0; i < len; i++) {
            char l = left.charAt(i);
            char r = right.charAt(i);
            if (l == r) {
                continue;
            }
            int lcv = getCharValue(l);
            int rcv = getCharValue(r);
            if (lcv != rcv) {
                return lcv - rcv;
            }
        }
        return 0;
    }

    private static int getCharValue(char c) {
        if (c >= 'a') {
            return (c - 'a') * 2 + 1;
        } else if (c >= 'A') {
            return (c - 'A') * 2;
        }
        return c - '0' + 52;
    }
}
```

2、链表拆分

【题目描述】设 C={a1,b1,a2,b2,...,an,bn}为线性表，采用带头结点的 hc 单链表存放，设计一个算法，将其拆分为两个线性表，使得奇数位保持正序，偶数位转化为逆序。即：

A={a1,a2,...,an}，B={bn,...,b2,b1}

【解题思路】

奇数使用尾插法，偶数使用头插法。

【参考代码】

```
int main()
```

```
{
 struct Data *ha,*ta,*hb;
 ha = (struct Data *)malloc(sizeof(struct Data));
 ta = (struct Data *)malloc(sizeof(struct Data));
 hb = (struct Data *)malloc(sizeof(struct Data));
 ha->next = hc->next;
 ta->next = NULL;
 hb->next = NULL;

 struct Data *p,*q;
 int i = 0;
 p = hc->next;
 while(p!=NULL)
 {
  q = p->next;
  if(i % 2 == 0)
  { //尾插法生成链表 A
   p->next = ta->next;
   ta->next = p;
   ta = p;
  }
  else
  { //头插法生成转置链表 B
   p->next = hb->next;
   hb->next = p;
  }
  i++;
  p = q;
 }
 hc->next = NULL;
}
```

3、最长对称子串

【题目描述】定义前后两端完全一致的字符串为对称字符串，如"abba"、"caddac"，编写程序，输出字符串"abcdefiiaaovivoovivcaideumncca"的最长对称子字符串。

（注："aba"这种不算为对称字符串）

【解题思路】

暴力枚举中心。

或者使用 Manacher 算法。

【参考代码】

```
public class StringManager {
    public static String getMaxMirrorString(String str) {
  String maxStr = "";
  int maxStep = 0;
  // 顺序遍历字符串，以 i 为中心点，查找对称字串；
        for(int i = 0; i < str.length(); i++) {
   for(int step = 0; i - step >= 0 && i + step + 1 < str.length(); step++) {
    if(str.charAt(i-step) != str.charAt(i+step+1))
    break;
            }
   if(step > maxStep) {
    maxStep = step;
    maxStr = str.substring(i-step+1, i+step);
   }
        }
        return maxStr;
    }

    public static void main(String[] args){
        System.out.println(getMaxMirrorString("abcdefiiaaovivoovivcaideumncca"));
    }
}
```

九、小米 2019 秋招笔试真题

1、厨艺大赛奖金

【题目描述】小米食堂每年都会举办一次厨艺大赛，假设参赛的厨师一共有 n 位（n < 1000），比赛结束后没有公布评分，但是站在领奖台上的一排厨师中每位厨师都能看到与自己相邻的厨师（左或者右）里评分比自己低（看不到比自己分数高的人的分数）的评分。比赛结束之后要发奖金，以 1K 为单位，每位厨师至少会发 1K 的奖金，另外，如果一个厨师发现自己的奖金没有高于比自己评分低的厨师的奖金，就会不满意，作为比赛组织方，小米食堂至少需要发放多少奖金才能让所有厨师满意。

输入描述:

每组数据为 n+1 个正整数单空格分割，其中第一个数为参赛厨师的人数，后面 n 个数为每位厨师的得分（0-100）

输出描述:

输出至少需要多少 K 的奖金

输入样例:

10 60 76 66 76 85 55 61 71 84 62

输出样例:

20

【参考代码】

```
import java.util.Scanner;

public class Main {

  /*请完成下面这个函数，实现题目要求的功能
  当然，你也可以不按照下面这个模板来作答，完全按照自己的想法来 ^-^
  ******************************开始写代码******************************/
  static int calMinBonus(int[] score) {
    int[] bonus = new int[score.length];
    for (int i = 0; i < bonus.length; i++) {
      bonus[i] = 1;
    }
    for (int i = 0; i < score.length - 1; i++) {
      if (score[i + 1] > score[i]) {
        bonus[i + 1] = bonus[i] + 1;
      }
    }
    for (int i = score.length - 1; i > 1; i--) {
      if (score[i - 1] > score[i] && bonus[i - 1] <= bonus[i]) {
        bonus[i - 1] = bonus[i] + 1;
      }
    }
    int sum = 0;
    for (int ele : bonus) {
      sum += ele;
    }
    return sum;

  }

  /******************************结束写代码******************************/


  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int res;

    int _score_size = 0;
    String[] ps = in.nextLine().split(" ");
    _score_size = Integer.parseInt(ps[0]);
```

```
    int[] _score = new int[_score_size];
    int _score_item;
    for (int _score_i = 0; _score_i < _score_size; _score_i++) {
      _score_item = Integer.parseInt(ps[_score_i + 1]);
      _score[_score_i] = _score_item;
    }

    res = calMinBonus(_score);
    System.out.println(String.valueOf(res));

  }
}
```

2、计算原子的个数

【题目描述】给出一个字符串格式的化学分子式，计算原子的个数

每个化学元素都是由一个大写字母，或者一个大写字母后跟着若干个小写字母组成，例如 H 是一个化学元素，Mg 也是一个化学元素。

每个分子式中，原子的个数写在元素后面，如果原子个数是 1，那么原子个数省略。例如 H2O 和 H2O2 都是有效的分子式，但 H1O2 不是有效分子式。

每个分子式中包含若干括号，为简单起见，分子式中只有小括号。

每次输入一个分子式，对每个给定的分子式，求出每个原子的个数，按照原子字母表的顺序排列，并输出。

输入描述:

输入分子式为

H2O

Mg(OH)2

K4(ON(SO3)2)2

输出描述:

分别输出

H2O

H2MgO2

K4N2O14S4

输入样例:

K4(ON(SO3)2)2

输出样例:

K4N2O14S4

【参考代码】

```
#include <iostream>
#include <string>
#include <map>
#include <stack>
```

```
using namespace std;

string countOfAtoms(string formula)
{
 string res = "";
 stack<map<string, int>> st;
 map<string, int> cur;
 int n = formula.size(), pos = 0;

 while (pos < n) {
  if (formula[pos] == '(')
  {
   ++pos;
   st.push(move(cur));
  }
  else if (formula[pos] == ')')
  {
   int i = ++pos;
   while (pos < n && isdigit(formula[pos]))
    ++pos;
   int multiple = stoi(formula.substr(i, pos - i));
   for (auto a : cur)
    st.top()[a.first] += a.second * multiple;
   cur = move(st.top());
   st.pop();
  }
  else
  {
   int i = pos++;
   while (pos < n && islower(formula[pos]))
    ++pos;
   string elem = formula.substr(i, pos - i);
   i = pos;
   while (pos < n && isdigit(formula[pos]))
    ++pos;
   string cnt = formula.substr(i, pos - i);
   cur[elem] += cnt.empty() ? 1 : stoi(cnt);
  }
 }

 for (auto a : cur)
```

```
    res += a.first + (a.second == 1 ? "" : to_string(a.second));

  return res;
}


int main(void)
{
  string formula, result;
  cin >> formula;

  result = countOfAtoms(formula);
  cout << result << endl;

  return 0;
}
```

十、平安科技 2019 秋招笔试真题

1、鸡蛋掉落

【题目描述】你有 K 个鸡蛋，并可以使用一栋从 1 到 N 共有 N 层楼的建筑。每个蛋的功能都是一样的，如果一个蛋碎了，你就不能再把它掉下去。你知道存在楼层 F ，满足 0 <= F <= N 任何从高于 F 的楼层落下的鸡蛋都会碎，从 F 楼层或比它低的楼层落下的鸡蛋都不会破。每次移动，你可以取一个鸡蛋（如果你有完整的鸡蛋）并把它从任一楼层 X 扔下（满足 1 <= X <= N）。你的目标是确切地知道 F 的值是多少。无论 F 的初始值如何，你确定 F 的值的最小移动次数是多少？

输入描述：

输入两个数，（第一个数为 K，第二个数为 N）

输出描述：

返回最小移动次数

输入样例：

1,2

输出样例：

2

【解题思路】

动态规划

dp[k][n]表示 k 个鸡蛋 n 层楼至少要试几次。

枚举丢的楼层 i，则 dp[k][n] = min(dp[k][n-i], dp[k-1][i-1]) （对应没碎与碎）

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100 + 5;
const int inf = 0x3f3f3f3f;
int n, k;
int dp[N][N];
int dfs(int k, int n) {
    if(k == 1) {
        return n;
    }
    if(n <= 1) {
        return 0;
    }
    int &ans = dp[k][n];
    if(~ans) return ans;
    ans = inf;
    for(int i = 1; i <= n; ++i) {
        int tmp = min(dfs(k, n-i), dfs(k-1, i-1)) + 1;
        ans = min(ans, tmp);
    }
    return ans;
}

int main() {
    scanf("%d%d", &k, &n);
    memset(dp, -1, sizeof(dp));
    printf("%d\n", dfs(k, n));
    return 0;
}
```

2、水壶问题

【题目描述】有两个容量分别为 x 升和 y 升的水壶以及无限多的水。请判断能否通过使用这两个水壶，从而可以得到恰好 z 升的水？如果可以，最后请用以上水壶中的一或两个来盛放取得的 z 升水。

你允许进行以下三种操作：

1. 装满任意一个水壶
2. 清空任意一个水壶
3. 从一个水壶向另外一个水壶倒水，直到装满或者倒空

输入描述：

三个数

输出描述：

布尔格式 True/False

输入样例：

3,4,5

输出样例：

True

【解题思路】

dfs，枚举所有的操作。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100 + 5;
const int inf = 0x3f3f3f3f;
int x, y, z;
int vis[N][N];

int dfs(int a, int b) {
    if(a == z || b == z) {
        return 1;
    }
    vis[a][b] = 1;
    int ans = 0;
    if(!vis[x][b]) {
        ans |= dfs(x, b);
    }
    if(!vis[a][y]) {
        ans |= dfs(a, y);
    }
    if(!vis[x][0]) {
        ans |= dfs(x, 0);
    }
    if(!vis[0][y]) {
        ans |= dfs(0, y);
    }
    int pour = min(b, x - a);
    int na = a + pour;
    int nb = b - pour;
    if(!vis[na][nb]) {
        ans |= dfs(na, nb);
    }
    pour = min(a, y - b);
    na = a - pour;
    nb = b + pour;
```

```
    if(!vis[na][nb]) {
        ans |= dfs(na, nb);
    }
    return ans;
}

int main() {
    scanf("%d%d%d", &z, &x, &y);
    puts(dfs(0, 0) ? "True" : "False");
    return 0;
}
```

3、查找元素

【题目描述】给定一个包含 n 个整数的数组 nums，判断 nums 中是否存在三个元素 a，b，c ，使得 a + b + c = 0 ？找出所有满足条件且不重复的三元组。

输入描述：

一个数组

输出描述：

返回多个满足条件的元素组合

输入样例：

[-1,0,1,2,-1,-4]

输出样例：

-1 -1 2

-1 0 1

【解题思路】

枚举 a 和 b，在排序后的数组里二分查找 c=-a-b。注意在找 c 的时候不能与 a 和 b 重复。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1000 + 5;
const int inf = 0x3f3f3f3f;
int n;
int a[N];
set<pair<int,int> > vis;
multiset<int> s;
int main() {
    scanf("%d", &n);
    for(int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
```

```
        s.insert(a[i]);
    }
    sort(a+1, a+n+1);
    for(int i = 1; i <= n; ++i) {
        s.erase(s.find(a[i]));
        for(int j = i + 1; j <= n; ++j) {
            s.erase(s.find(a[j]));
            if(s.count(-a[i]-a[j])) {
                int b[3] = {a[i], a[j], -a[i]-a[j]};
                sort(b, b+3);
                auto p = make_pair(b[0], b[1]);
                if(!vis.count(p)) {
                    printf("%d %d %d\n", a[i], a[j], -a[i]-a[j]);
                    vis.insert(p);
                }
            }
            s.insert(a[j]);
        }
        s.insert(a[i]);
    }
    return 0;
}
```

十一、度小满 2019 秋招笔试真题

1、火车站台

【题目描述】在 Z 省，有若干个个城市坐落在一条直线上，从左到右依次标号 1,2,3,…，其中每个城市有一个火车站点，现今已经开放了 n 条火车路线，第 i 条火车路线是从第 Yi 个城市到第 Xi 个城市，因为 Z 省地势奇特，标号小的城市地势较低，所以火车是从高往低开的，再通过神秘力量传送回高地，即 Yi 一定大于 Xi，它在沿途的所有城市都会停靠（显然不包括起点 Yi，但是包括终点 Xi），火车停靠就需要火车站台来运营维护。火车站台随着运营线路的数量不同，其损耗速度、维护成本也各异，现在我们想知道所有站台中，所运营的线路数最大是多少。

输入描述：

第一行一个数 n。（1≤n≤100000）

接下来 n 行每行两个数 Xi 和 Yi，分别代表第 i 条火车路线的终点和起点。（1≤Xi<Yi≤1e5）

输出描述：

共一行，一个非负整数，表示最大运营路线数。

输入样例：

4

4 7

2 6
2 5
1 2

输出样例：

3

【参考代码】

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

int n, cnt[1000010];

int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        int x, y;
        scanf("%d%d", &x, &y);
        cnt[x]++; cnt[y]--;
    }
    int ans = 0, now = 0;
    for (int i = 1; i <= 1000000; ++i) {
        now += cnt[i];
        ans = max(ans, now);
    }
    printf("%d\n", ans);
    return 0;
}
```

2、商品交易

【题目描述】珐达采下个月要去鸥洲各国考察一趟，采购流通神秘石并从中搞点油水。
珐达采会按顺序依次经过序号分别为 1，2，3，…，
n 的鸥洲国家，在第 i 个国家神秘石的流通价格为 Ai 鸥。因为行程紧张，在每个国家的停留时间有限，所以他只能花费 Ai 鸥买入一块神秘石，或者卖出一块手中的神秘石获得 Ai 鸥，或者什么都不做，而且因为神秘石的保存需要极其先进的高级材料容器，其材料稀有且制作困难，珐达采只有一份容器，故无论何时珐达采手里
最多只能拥有一块神秘石。
珐达采想知道最终能从中获利最大多少鸥。因为交易需要手续费，所以珐达采还想知道在获利最大收益的同时，最少需要交易多少次。因为珐达采是大财阀，所以你可以认为他一开始金钱无限。

输入描述：

第一行一个数 n。（1≤n≤100000）
第二行 n 个数，第 i 个数表示 Ai。（1≤Ai≤1e9）

输出描述：

共一行，两个数，分别代表最大收益和对应的最少交易次数。

输入样例：

5

9 7 10 1 5

输出样例：

7 4

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int N=100010;
LL f[N];
int g[N];
int n;
int main() {
    scanf("%d", &n);
    LL p = -(1LL << 60), tran = 0, x = 0, ans = 0, anst = 0;
    for (int i = 1; i <= n; ++i) {
        scanf("%lld", &x);
        if (p + x > 0)
            f[i] = p + x, g[i] = tran + 2;
        else f[i] = g[i] = 0;
        LL ptmp = ans - x, ttmp = anst;
        if (ptmp > p || (ptmp == p && ttmp < tran)) {
            p = ptmp;
            tran = ttmp;
        }
        if (f[i] > ans || (f[i] == ans && g[i] < anst)) {
            ans = f[i];
            anst = g[i];
        }
    }
    cout << ans << ' ' << anst << endl;
    return 0;
}
```

十二、携程 2019 秋招笔试真题

1、提取优惠券码

【题目描述】携程客户下单的时候系统会发放一个优惠券码，用户在前端界面看到的展现形式如下：“1Ct3r4ip_ti4C6k9Etˆ”，包含数字、字母、和 [\]ˆ_’ 6 个特殊字符。
为了安全性，系统在使用的时候做解密操作，删除里面的数字部分，只验字母 A~z 以及[\]ˆ_’ 6 个特殊字符部分才是有效的，并且在解密后不会重新排序字母的顺序，按照之前的对应顺序排列。
如上所示，如果一个客户拿到的优惠券码是这个：1Ct3r4ip_ti4C6k9Etˆ ， 请用 java 语言提取正确的优惠券码。

输入描述:

1Ct3r4ip_ti4C6k9Etˆ

输出描述:

Ctrip_tiCkEtˆ

输入样例：

1Ct3r4ip_ti4C6k9Etˆ

输出样例：

Ctrip_tiCkEtˆ

【解题思路】

按照题意提取对应有用的信息字符即可。

【参考代码】

```
public class Main {
    public static void main(String[] args) {

        String strings = "1Ct3r4ip_ti4C6k9Etˆ";

        char[] chars = strings.toCharArray();
        for (int i = 0; i < chars.length; i++) {
            if (chars[i] >= 48 && chars[i] <= 57) {
                continue;
            } else {
                System.out.print(chars[i]);
            }

        }
    }
}
```

2、1-10 排序算法

【题目描述】请使用 random() 函数，生成 10 个随机数，并对它进行降序排序 ，要求有效率。

注意：不允许使用代码库提供的 sort() 之类现成的排序函数。

输出描述:

1
2
2
4
5
5
6
6
8
10

备注:

因为是 random 产生，所以只要排序对就行

输入样例：

1

输出样例：

1

说明：

因为是 random 产生，所以只要排序对就行

【解题思路】

得到随机数据之后，实现一个排序算法(如快速排序)即可。

【参考代码】

```
import java.util.Arrays;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n=in.nextInt();
        int[] nums = new int[n];
        for (int i = 0; i < n; i++) {
            int num = (int) (Math.random() * (n + 1))+1;
            nums[i] = num;
        }
        quickSort(nums, 0, nums.length-1);
        for(int i=nums.length-1;i>=0;i--) {
            System.out.println(i+1);
        }

    }
    private static void quickSort(int[] a, int low, int high) {
```

```
        if (low >= high) {
            return;
        }
        int i = low;
        int j = high;
        int key = a[low];
        while (i < j) {
            while (i < j && a[j] > key) {
                j--;
            }
            while (i < j && a[i] <= key) {
                i++;
            }
            if (i < j) {
                int p = a[i];
                a[i] = a[j];
                a[j] = p;
            }
        }
        int p = a[i];
        a[i] = a[low];
        a[low] = p;
        quickSort(a, low, i - 1);
        quickSort(a, i + 1, high);
    }
}
```

十三、快手 2019 秋招笔试真题

1、魔法深渊

【题目描述】前几个月放映的头号玩家简直火得不能再火了，作为一个探索终极 AI 的研究人员，月神自然去看了此神剧。

由于太过兴奋，晚上月神做了一个奇怪的梦，月神梦见自己掉入了一个被施放了魔法的深渊，月神想要爬上此深渊。

已知深渊有 N 层台阶构成(1 <= N <= 1000)，并且每次月神仅可往上爬 2 的整数次幂个台阶(1、2、4、...)，请你编程告诉月神，月神有多少种方法爬出深渊

输入描述:

输入共有 M 行，(1<=M<=1000)

第一行输入一个数 M 表示有多少组测试数据，

接着有 M 行，每一行都输入一个 N 表示深渊的台阶数

输出描述:

输出可能的爬出深渊的方式

备注:

为了防止溢出，可将输出对 10^9 + 3 取模

输入样例：

4
1
2
3
4

输出样例：

1
2
3
6

【解题思路】

第 6 个台阶可以从 2,4,5 一次性到达，把 dp[2],dp[3],dp[4],dp[5]求和即可

第 1000 个台阶可以从 488(1000-512),744（1000-256）...999 一次性到达，把 dp[488]+...+dp[999]求和即可

【参考代码】

```
#include <iostream>
#include <vector>
using namespace std;

typedef unsigned long long lld;

const lld MOD = 1000000003;

vector<lld> P2;

lld Ans[1001];

void CalcValue();
int main() {
    CalcValue();
    int iCase;
    cin >> iCase;
    int n;
    while (iCase--) {
        cin >> n;
        cout << Ans[n] << endl;
    }
```

```
}

void CalcValue() {
    for (int i = 0; i < 10; ++i) {
        P2.push_back(1 << i);
    }

    Ans[0] = 1;
    for (int i = 1; i < 1001; ++i) {
        Ans[i] = 0;
        for (int j = 0; j < P2.size(); ++j) {
            if (P2[j] > i) {
                break;
            }
            Ans[i] += Ans[i - P2[j]];
            Ans[i] %= MOD;
        }
    }
}
```

2、latex 爱好者

【题目描述】latex 自然是广大研究人员最喜欢使用的科研论文排版工具之一。

月神想在 iPhone 上查阅写好的 paper，但是无赖 iPhone 上没有月神喜欢使用的阅读软件，于是月神也希望像 tex 老爷爷 Donald Knuth 那样自己动手 do it yourself 一个。

在 DIY 这个阅读软件的过程中，月神碰到一个问题，已知 iPhone 屏幕的高为 H，宽为 W，若字体大小为 S(假设为方形)，则一行可放 W / S(取整数部分）个文字，一屏最多可放 H / S （取整数部分）行文字。

已知一篇 paper 有 N 个段落，每个段落的文字数目由 a1, a2, a3,...., an 表示，月神希望排版的页数不多于 P 页（一屏显示一页），那么月神最多可使用多大的字体呢？

1 <= W, H, ai <= 1000

1 <= P <= 1000000

输入描述:

每个测试用例的输入包含两行。

第一行输入 N,P,H,W

第二行输入 N 个数 a1,a2,a3,...,an 表示每个段落的文字个数。

输出描述:

对于每个测试用例，输出最大允许的字符大小 S

备注:

以上所有输入、输出均为整数。

且所有测试用例均保证有解。

两个段落之前不空行，并且段落顶格写。

输入样例：

1 10 4 3
10
2 10 4 3
10 10

输出样例：

3
2

【解题思路】

二分答案，然后进行可行性判断。

【参考代码】

```
#include <iostream>
#include <algorithm>
using namespace std;

int N, P, W, H;
int Paragraph[10000];

bool HandleCase();

int main() {


    while (HandleCase()) {
        /**empty*/
    }
}


bool OK(int S);

bool HandleCase() {
    if (!(cin >> N >> P >> W >> H)) {
        return false;
    }

    for (int i = 0; i < N; ++i) {
        cin >> Paragraph[i];
    }
    int l = 1, r = min(W, H), m;
    while (l < r) {
        m = (l + r) >> 1;
```

```
            if (OK(m)) {
                l = m + 1;
            }
            else {
                r = m - 1;
            }
        }

        while (!OK(l)) --l;

        cout << l << endl;

        return true;
}

bool OK(int S) {
        int iEachLine = W / S;
        int iLine = H / S;
        int TotalLine = 0;
        if (iEachLine <= 0 || iLine <= 0)
            return false;
        for (int i = 0; i < N; ++i) {
            if (iEachLine == 1) {
                TotalLine += Paragraph[i];
            }
            else {
                TotalLine += (Paragraph[i] + iEachLine - 1) / iEachLine;
            }
        }
        return (TotalLine + iLine - 1) / iLine <= P;
}
```

3、回文字符串

【题目描述】最大回文子串是被研究得比较多的一个经典问题。最近月神想到了一个变种，对于一个字符串，如果不要求子串连续，那么一个字符串的最大回文子串的最大长度是多少呢。

输入描述:

每个测试用例输入一行字符串（由数字 0-9，字母 a-z、A-Z 构成），字条串长度大于 0 且不大于 1000.

输出描述:

输出该字符串的最长回文子串的长度。（不要求输出最长回文串，并且子串不要求连续）

备注:

因为不要求子串连续，所以字符串 abc 的子串有 a、b、c、ab、ac、bc、abc7 个

输入样例：

adbca

输出样例：

3

说明：

因为在本题中，不要求回文子串连续，故最长回文子串为 aba(或 ada、aca)

【解题思路】

直接 dp 或者使用马拉车算法计算即可。

【参考代码】

```
#include <string>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <cstdlib>
#include <math.h>
#include <algorithm>
using namespace std;

bool HandleCase();
int dp[1001][1001];

int n;

int main() {
    while (HandleCase()) {
        /**empty here*/
    }
}

bool HandleCase() {
    string str;
    if (!(cin >> str)) {
        return false;
    }

    string rstr = str;
    std::reverse(rstr.begin(), rstr.end());

    n = str.length();
    memset(dp, 0, sizeof(dp));
```

```
    int r = 1;
    dp[0][0] = (str[0] == rstr[0]);
    for (int i = 1; i < n; ++i) {
        dp[i][0] = (str[i] == rstr[0]) || dp[i - 1][0];
        dp[0][i] = (str[0] == rstr[i]) || dp[0][i-1];
    }

    for (int i = 1; i < n; ++i) {
        for (int j = 1; j < n; ++j) {
            dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
            if (str[i] == rstr[j]) {
                dp[i][j] = max(dp[i - 1][j - 1] + 1, dp[i][j]);
            }
            else {
                dp[i][j] = max(dp[i][j], dp[i - 1][j - 1]);
            }

            r = max(r, dp[i][j]);
        }

    }

    cout << r << endl;


    return true;
}
```

十四、拼多多 2019 秋招笔试真题

1、回合制游戏

【题目描述】你在玩一个回合制角色扮演的游戏。现在你在准备一个策略，以便在最短的回合内击败敌方角色。在战斗开始时，敌人拥有 HP 格血量。当血量小于等于 0 时，敌人死去。一个缺乏经验的玩家可能简单地尝试每个回合都攻击。但是你知道辅助技能的重要性。

在你的每个回合开始时你可以选择以下两个动作之一：聚力或者攻击。

聚力会提高你下个回合攻击的伤害。

攻击会对敌人造成一定量的伤害。如果你上个回合使用了聚力，那这次攻击会对敌人造成 buffedAttack 点伤害。否则，会造成 normalAttack 点伤害。

给出血量 HP 和不同攻击的伤害，buffedAttack 和 normalAttack，返回你能杀死敌人的最小回合数。

输入描述:

第一行是一个数字 HP
第二行是一个数字 normalAttack
第三行是一个数字 buffedAttack
1 <= HP,buffedAttack,normalAttack <= 10^9

输出描述:

输出一个数字表示最小回合数

输入样例：

13
3
5

输出样例：

5

【解题思路】

Dfs 暴力搜索枚举所有可能的回合数，求出最小的回合数即可。

【参考代码】

```
#include <string>
#include <iostream>
#include <algorithm>
#include <cstring>
#include <vector>
#include <map>
#include <sstream>
#include <set>
#include <stack>
#include <cmath>
#include <iomanip>

using namespace std;

int dfs(long HP,long nA,long bA,long cur){
    int times1=0;
    int times2=0 ;
    if (cur < HP){
            times1 = 1 + dfs(HP,nA,bA,cur+nA);//
            times2 = 2 + dfs(HP,nA,bA,cur+bA);
        return (times1<times2?times1:times2);
    }
    else
        return 0;
}
int main(){
```

```
    long HP,nA,bA;
    while(cin>>HP>>nA>>bA){
        int times ;
        bool flag = false;
        cout<<dfs(HP,nA,bA,0)<<endl;
    }
    return 0;
}
```

2、种树

【题目描述】小多想在美化一下自己的庄园。他的庄园毗邻一条小河，他希望在河边种一排树，共 M 棵。小多采购了 N 个品种的树，每个品种的数量是 Ai（树的总数量恰好为 M）。但是他希望任意两棵相邻的树不是同一品种的。小多请你帮忙设计一种满足要求的种树方案。

输入描述：

第一行包含一个正整数 N，表示树的品种数量。

第二行包含 N 个正整数，第 i（1 <= i <= N）个数表示第 i 个品种的树的数量。

数据范围：

1 <= N <= 1000

1 <= M <= 2000

输出描述：

输出一行，包含 M 个正整数，分别表示第 i 棵树的品种编号（品种编号从 1 到 N）。若存在多种可行方案，则输出字典序最小的方案。若不存在满足条件的方案，则输出"-"。

输入样例：

3

4 2 1

输出样例：

1 2 1 2 1 3 1

【解题思路】

贪心，从前往后找当前第一种数量最多的数作为当前种的树。

【参考代码】

```
#include <algorithm>
#include <cassert>
#include <cstdio>
#include <vector>
using namespace std;

const int N = 1000;
const int M = 2000;
```

```
int main() {
  int n, sum = 0, ma = 0;
  scanf("%d", &n);
  assert(n >= 1 && n <= N);
  vector<int> a(n);
  for (auto& x : a) {
    scanf("%d", &x);
    assert(x >= 1 && x <= M);
    sum += x;
    ma = max(ma, x);
  }
  assert(sum <= M);

  if (ma - (sum - ma) > 1) {
    puts("-");
    return 0;
  }

  for (int prev = -1, k = 0; sum-- > 0; a[k]--, prev = k) {
    int ma = *max_element(a.begin(), a.end());
    for (k = 0; k < n; ++k) {
      if (k == prev || a[k] == 0) continue;
      if (a[k] == ma || ma - (sum - ma) <= 1) break;
    }
    printf("%d%c", k + 1, sum == 0 ? '\n' : ' ');
  }
}
```

3、选靓号

【题目描述】A 国的手机号码由且仅由 N 位十进制数字(0-9)组成。一个手机号码中有至少 K 位数字相同则被定义为靓号。A 国的手机号可以有前导零，比如 000123456 是一个合法的手机号。

小多想花钱将自己的手机号码修改为一个靓号。修改号码中的一个数字需要花费的金额为新数字与旧数字之间的差值。比如将 1 修改为 6 或 6 修改为 1 都需要花 5 块钱。

给出小多现在的手机号码，问将其修改成一个靓号，最少需要多少钱？

输入描述:

第一行包含 2 个整数 N、K，分别表示手机号码数字个数以及靓号至少有 K 个数字相同。

第二行包含 N 个字符，每个字符都是一个数字('0'-'9')，数字之间没有任何其他空白符。表示小多的手机号码。

数据范围:

2 <= K <= N <= 10000

输出描述:

第一行包含一个整数，表示修改成一个靓号，最少需要的金额。

第二行包含 N 个数字字符，表示最少花费修改的新手机号。若有多个靓号花费都最少，则输出字典序最小的靓号。

输入样例：

6 5

787585

输出样例：

4

777577

说明：

花费为 4 的方案有两种：777577 与 777775，前者字典序更小。

【解题思路】

枚举可以改为的靓号和花费，使用优先队列(堆)维护。

【参考代码】

```
#include <cmath>
#include <cstdio>
#include <queue>
#include <string>
using namespace std;

struct foo {
  int a, b, c;
  foo(int a, int b, int c) : a(a), b(b), c(c) {}
  bool operator<(const foo& rhs) const {
    if (a < rhs.a) return true;
    if (a > rhs.a) return false;
    if (b < rhs.b) return true;
    if (b > rhs.b) return false;
    if (b == 1)
      return c > rhs.c;
    else
      return c < rhs.c;
  }
};

int main() {
  int n, m, ans = 0x7f7f7f7f;
  char tmp[10005];
  string s, s1, as;
  scanf("%d%d%s", &n, &m, tmp);
  s = tmp;
```

```
  for (int i = 0; i < 10; ++i) {
    priority_queue<foo> q;
    s1 = s;
    for (int j = 0; j < n; ++j) {
      int a = abs(s[j] - '0' - i);
      int b = (s[j] - '0' < i);
      int c = j;
      q.push(foo(a, b, c));
      if (q.size() > m) q.pop();
    }
    int sum = 0;
    for (; !q.empty(); q.pop()) {
      sum += q.top().a;
      s1[q.top().c] = '0' + i;
    }
    if (sum < ans || sum == ans && s1 < as) {
      ans = sum;
      as = s1;
    }
  }
  printf("%d\n%s\n", ans, as.c_str());
}
```

十五、顺丰科技 2019 秋招笔试真题

1、单词数组左右对齐

【题目描述】给定一个单词数组和一个长度 maxWidth，重新排版单词，使其成为每行恰好有 maxWidth 个字符，且左右两端对齐的文本。

你应该使用“贪心算法”来放置给定的单词；也就是说，尽可能多地往每行中放置单词。必要时可用空格’’填充，使得每行恰好有 maxWidth 个字符。

要求尽可能均匀分配单词间的空格数量。如果某一行单词间的空格不能均匀分配，则左侧放置的空格数要多于右侧的空格数。

文本的最后一行应为左对齐，且单词之间不插入额外的空格。

说明:

单词是指由非空格字符组成的字符序列。每个单词的长度大于 0，小于等于 maxWidth。输入单词数组 words 至少包含一个单词。

输入样例:

words = ["What","must","be","acknowledgment","shall","be"]

maxWidth = 16

输出样例:

'What must be'
'acknowledgment '
'shall be '

解释:

注意最后一行的格式应为 "shall be " 而不是 "shall be",
因为最后一行应为左对齐，而不是左右两端对齐。
第二行同样为左对齐，这是因为这行只包含一个单词。
难度：4 级
时间限制：C/C++语言 1000MS；其他语言 3000MS
内存限制：C/C++语言 65536KB；其他语言 589824KB

输入描述：

一个单词数组和一个长度 maxWidth

输出描述：

重新排版单词，使其成为每行恰好有 maxWidth 个字符，且左右两端对齐的文本

输入样例：

Today,I,consider,myself,the,luckiest,man,on,the,face,of,the,earth.
16

输出样例：

'Today I consider'
'myself the'
'luckiest man on'
'the face of the'
'earth. '

提示：

```
public class Main {

    public static void main(String[] args) {
        String str1 = "What,must,be,acknowledgment,shall,be";
        String str2 = "16";

         List result = fullJustify(str1.split(","), Integer.valueOf(str2));
         for(String str : result){
              System.out.println("'"+str+"'");
         }
    }

    /**
     * TODO 算法实现
     * @param words 单词数组
     * @param maxWidth 每行长度
     * @return 结果集合
     */
```

```
    public static List fullJustify(String[] words, int maxWidth) {

    }
}
```

【解题思路】

按题意模拟。

先解析出所有词，贪心的往每行中放尽量多的词，如果放不下，稍加计算即可得出单词间的空格数。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
char s[N];
int w;

int main() {
    scanf("%s%d", s, &w);
    int n = strlen(s);
    s[n++] = ',';
    vector<string> words;
    string word = "";
    for(int i = 0; i < n; ++i) {
        if(s[i] == ',') {
            words.push_back(word);
            word = "";
        }else {
            word += s[i];
        }
    }
    vector<string> line;
    int width = -1;
    int space = w;
    for(string word : words) {
        if(width + word.size() + 1 > w) {
            int m = line.size();
            if(m == 1) {
                line[0] += ' ';
                printf("\'%-*s\'\n", w, line[0].c_str());
            }else {
                int gap = space / (m - 1);
                int extra = space % (m - 1);
                putchar('\'');
```

```
                for(int i = 0; i < m; ++i) {
                    printf("%s", line[i].c_str());
                    if(i != m - 1) {
                        int sp = i < extra ? gap + 1 : gap;
                        printf("%s", string(sp, ' ').c_str());
                    }else {
                        printf("\'\n");
                    }
                }
            }
            width = -1;
            space = w;
            line.clear();
        }
        if(width + word.size() + 1 <= w) {
            line.push_back(word);
            width += word.size() + 1;
            space -= word.size();
        }
    }
    int m = line.size();
    string str = "";
    for(int i = 0; i < m; ++i) {
        str += line[i];
        str += ' ';
    }
    if(str.size() > w) {
        str.resize(w);
    }
    printf("\'%-*s\'\n", w, str.c_str());
    return 0;
}
```

2、快递方案

【题目描述】快递小哥有一批紧急快递需要派送，需要你采用最佳方案来派送，派送总的承载量为 M 单位重量，有 N 件快递。

输入描述：

输入 M 和 N 行开头，其中有 N 行数据，每行包括 J[i]和 F[i]数据。J[i]表示运送快递利润，F[i]表示快递重量（KG），现在希望你给出一个利润最高的快递运送方案（设定快递可以不取整件），输入以-1 结束，要求不能使用 C、Java 语言集合类工具，比如排序等。

输出描述：

输出对应最佳快递方案的最高利润。

输入样例：

6 3
8 2
5 3
6 7
-1 -1

输出样例：

13.857

【解题思路】

由于可以不取整件，可以按利润率（单位重量的利润）从高到低的贪心选取。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;

struct node {
    int j, f;
    void read() {
        scanf("%d%d", &j, &f);
    }
    bool operator < (const node &rhs) const {
        return j * rhs.f > rhs.j * f;
    }
}a[N];
int n, m;

int main() {
    while(scanf("%d%d", &m, &n) == 2 && !(n == -1 && m == -1)) {
        for(int i = 0; i < n; ++i) {
            a[i].read();
        }
        sort(a, a+n);
        double ans = 0;
        for(int i = 0; i < n; ++i) {
            if(a[i].f <= m) {
                m -= a[i].f;
                ans += a[i].j;
            }else {
                ans += 1.0* m * a[i].j / a[i].f;
                break;
```

```
            }
        }
        printf("%.3f\n", ans);
    }
    return 0;
}
```

3、行程安排

【题目描述】我是一个大帅哥，因此有很多粉丝想和我合影，想请我吃饭，也有很多签售会演唱会等着我，总之我很忙。可是，我的秘书非常不靠谱，他总是把一些日程安排在重复的时间上，比如我今天的日程是：早上 8:00-10:00 粉丝见面会。早上 9:00-9:30 粉丝早餐会。下午 1:30-5:00 午睡(是的，这很重要)。晚上 8:00-9:30 婚礼表演嘉宾 所以，由于粉丝见面会更重要，我不得不取消粉丝早餐会了，因为他们在同一时间进行。那么问题来了，现在我需要一套算法，当我输入一天的行程，我需要这个算法告诉我，今天至少要取消多少个行程才能让每个日程之间时间不重叠。skrskr~

输入：

原始输入为时间点数目（行程数*2）以及各个行程的开始结束时间点。

需要先转化为一个二元组的 list，如

list[(8.0, 10.0), (8.0, 10.0), (8.0, 10.0), (8.0, 10.0), (12.0, 14.5)]

其中二元组内第一第二个元素分别为事项的开始和结束时间，以 float 显示，如早上 9：00-10：00 表示为(9.0,10.0)，下午 1：30-下午 5:00 表示为(13.5,17.0)。

输出：

需要取消多少个行程，以 int 显示。

输入样例：

```
10
8.0
10.0
8.0
10.0
8.0
10.0
8.0
10.0
12.0
14.5
```

输出样例：

```
3
```

【解题思路】

这等价于最多能选多少个互不重叠的区间，是经典的问题。可以使用贪心策略，即：在不与已选区间重叠的前提下，优先选择右端点最小的区间。

所以按区间右端点排序后，依次 check，不重叠就选择，并更新已选区间的最优端，方便判断重叠。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
struct node {
    int l, r;
    void read() {
        static double _l, _r;
        scanf("%lf%lf", &_l, &_r);
        l = _l * 100 + 0.5;
        r = _r * 100 + 0.5;
    }
    bool operator < (const node &rhs) const {
        return r < rhs.r;
    }
}a[N];

int main() {
    int n;
    scanf("%d", &n);
    n /= 2;
    for(int i = 0; i < n; ++i) {
        a[i].read();
    }
    sort(a, a+n);
    int right = -1, cnt = 0;
    for(int i = 0; i < n; ++i) {
        if(a[i].l >= right) {
            right = a[i].r;
        }else {
            ++cnt;
        }
    }
    printf("%d\n", cnt);
    return 0;
}
```

十六、招银网络科技 2019 秋招笔试真题

1、微机主机间编码转换

【题目描述】在银行的系统中存在微机和主机的交互,两者编码格式不同，譬如微机往主机传输的时候需要对字符串进行如下转码”abc 中国 ad 美” 转为”abc<>中国<>ad<>美<>” (其中每个<>表示一个字节)再进行传输。现在对于某个字段，主机只接收字节长度为 len 的字符串，现在需要将微机的字符串 str 在传输给主机前作转码截取处理。
请根据要求写出函数或方法实现。

【解题思路】
解析连续的汉字和字母。注意汉字的编码特点。

【参考代码】

```
#include <iostream>
#include <vector>
#include <time.h>
#include <sstream>
using namespace std;

string substring(string s, int length) {
    if(s.empty()){
        return "";
    }
    char *bytes;
    bytes = s.c_str();

    int byteArr=s.length();
    int n = 0; // 表示当前的字节数
    int i = 0; // 要截取的字节数，从第 3 个字节开始
    bool isFirst=false;
    for (; i < byteArr && n < length; i++){
        // 奇数位置，如 3、5、7 等，为 UCS2 编码中两个字节的第二个字节
        if (i % 2 != 0){
            n++; // 在 UCS2 第 1 个字节时 n 加 1
        }
        else{
            // 当 UCS2 编码的第 1 个字节不等于 0 时，该 UCS2 字符为汉字，一个汉字算两个字节
            if (bytes[i] != 0){
                n++;
                if(!isFirst){
                    n+=2;//如果是第一个汉字 长度加 2
                }
```

```
                isFirst=true;
            }else{
                isFirst=false;
            }
        }

    }
    //unicode 是双字节 对于 i 为奇数 全角 则减 1 半角 加 1
    if (i % 2 != 0){
        if(bytes[i-1]!=0){
            i -= 1;
        }else{
            i+=1;
        }
    }
    return string(bytes, 0, i);
}
```

2、网络 url 遍历

【题目描述】一个门户网站，包含各种超链接，每个链接属于某种类别，比如运动、新闻、购物、美妆等类别。要求实现：把整个网站包含的所有 url 分类，如某一行数据：运动：url1，url2….。请编程实现分类方法。

（1）Java：给定方法 String getType(String url)可以返回 url 类型，List<String>getUrl(String url)返回该 url 页面下的其他 url。

（2）C++:给定方法 string getType(string url)可以返回 url 类型，vector<string>getUrl(string url)返回该 url 页面下的其他 url。

问题：请完成完成编码，实现网站 URL 遍历。

【解题思路】

根据一个 URL 可以得到更多的 URL 列表，而 URL 列表有可能包含已经处理过的 URL，也即 URL 的遍历不是单纯的树遍历，而是图的遍历。

【参考代码】

```
#include <map>
#include <set>
#include <string>
#include <vector>
#include <iostream>
using namespace std;

class UrlClassifer {
public:
```

```
    static set<string> classifiedUrls;
    static map<string, set<string>> result;
    staticvoid classifyUrl(string url);
    static vector<string> getUrl(string url);
    static string getType(string url);
};

set<string> UrlClassifer::classifiedUrls;
map<string, set<string>> UrlClassifer::result;

void UrlClassifer::classifyUrl(string url) {
    // 非空判断
    if (url == "") {
        return;
    }
    // 加入分类结果
    string type = getType(url);
    map<string, set<string>>::iterator it = result.find(type);
    if (it != result.end()) {
        it->second.insert(url);
    } else {
        set<string> set;
        set.insert(url);
        result.insert(make_pair(type, set));
    }
    // 加入遍历 set
    classifiedUrls.insert(url);
    // 获取下一个 URL 列表
    vector<string> nextUrls = getUrl(url);
    // 判断 URL 是否为空:出口 1
    if (nextUrls.empty()) {
        return;
    }

    for (vector<string>::iterator vit = nextUrls.begin(); vit != nextUrls.end(); vit++) {
        set<string>::iterator sit = classifiedUrls.find(*(vit));
        // 是否已经遍历过 URL:出口 2
        if (sit != classifiedUrls.end()) {
            continue;
        }
        // 递归遍历分类 URL
        classifyUrl(*(vit));
```

```
    }
}
vector<string> UrlClassifer::getUrl(string url) {}
string UrlClassifer::getType(string url) {}
```

十七、爱奇艺 2019 秋招笔试真题

1、局长的食物

【题目描述】局长有 N 种食物，每种食物有 Ai 份。 每天局长会吃一份食物，或者买一份食物，这样过了 M 天 现在局长想知道 M 天后第 p 种食物的份数排名（从大到小，相同算并列，例如 3 3 2，则排名为 1 1 3） N,M,P<=100,Ai<=1000

输入描述：

第一行 N M P

第二行 N 个数 Ai

接下来 M 行，每行 A i 或者 B i 分别表示买一份食物 i，吃一份食物 i

输出描述：

一个答案

输入样例：

3 4 2 5 3 1 B 1 A 2 A 2 A 3

输出样例：

1

【解题思路】

按题意模拟 M 天的情况，求得比 P 种食物多的食物种类，即可求得 P 的排名。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100 + 5;

int n, m, p;
int a[N];

int main() {
    scanf("%d%d%d", &n, &m, &p);
    for(int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
    }
    while(m--) {
        char o[5];
        int x;
```

```
        scanf("%s%d", o, &x);
        if(o[0] == 'A') {
            ++a[x];
        }else {
            --a[x];
        }
    }
    int rk = 1;
    for(int i = 1; i <= n; ++i) {
        if(a[i] > a[p]) {
            ++rk;
        }
    }
    printf("%d\n", rk);
    return 0;
}
```

2、清雨的自助餐

【题目描述】清雨又在吃自助餐了。
排在清雨面前的有 N 种食物，排成一排，清雨可以选择其中的若干种食物，但是不能连续选择相邻的食物，请问他有多少种方法呢？
1 <= n <= 90

输入描述：

一个整数 n

输出描述：

一个数的答案

输入样例：

3

输出样例：

5

【解题思路】

动态规划。
设 dp[i]表示 1~i 的食物有多少种选法。考虑第 i 种食物，若选，则 i-1 不能选，所以方案为 dp[i-2]；若不选，则方案数为 dp[i-1]。故转移方程 dp[i] = dp[i-1] + dp[i-2]。
容易求得初始条件为 dp[1] = 2, dp[2] = 3。
90 项用 long long 刚好存得下。

【参考代码】

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

```
typedef long long LL;
const int N = 90 + 5;

int n;
LL a[N];

int main() {
    scanf("%d", &n);
    a[1] = 2;
    a[2] = 3;
    for(int i = 3; i <= n; ++i) {
        a[i] = a[i-1] + a[i-2];
    }
    printf("%lld\n", a[n]);
    return 0;
}
```

3、库特君的面条

【题目描述】库特君在吃面条！

他将面条放在了数轴上，每根面条对应数轴上了两个点 a 和 b(a<b)，他想知道在任意两根面条不重叠(端点可以重叠）的情况下最多能选出多少根面条。

1 <= n <= 100

-999 <= a < b <= 999

输入描述：

第一行一个整数 N

接下来，N 行每行 N 个整数 a 和 b

输出描述：

一个数的答案

输入样例：

3 6 3 1 3 2 5

输出样例：

2

【解题思路】

贪心。

经典的问题，贪心策略：在不与已选区间重叠的前提下，优先选择右端点最小的区间。

所以按区间右端点排序后，依次 check，不重叠就选择，并更新已选区间的最优端，方便判断重叠。

【参考代码】

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

```
const int N = 100 + 5;
const int inf = 0x3f3f3f3f;

struct node {
    int l, r;
    void read() {
        scanf("%d%d", &l, &r);
        if(l > r) swap(l, r);
    }
    bool operator < (const node &rhs) const {
        return r < rhs.r;
    }
}a[N];

int main() {
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; ++i) {
        a[i].read();
    }
    sort(a, a+n);
    int right = -1000, cnt = 0;
    for(int i = 0; i < n; ++i) {
        if(a[i].l >= right) {
            ++cnt;
            right = a[i].r;
        }
    }
    printf("%d\n", cnt);
    return 0;
}
```

十八、搜狐畅游 2019 秋招笔试真题

1、两种进制的转换

【题目描述】小张同学新发明了一个小机器，但是这个机器只能识别四进制的数字，你能否对于给定的一个十进制数（这个数取值范围在 0 到 9999），将它的四进制的表示出来，然后能让这个机器识别出来呢？

输入描述:

每个数据输入一个数

输出描述:

每组数据输出其对应的四进制表示

输入样例 1：

6

输出样例 1：

12

输入样例 2：

17

输出样例 2：

101

【解题思路】

直接把数转换为四进制即可。

【参考代码】

```
ans = ""
x = int(input())
num = x
while x:
    n =x%4
    ans +=str(n)
    x//=4
if num%4 == 0:
    ans += str(0)
ans = ans[::-1]
print(ans)
```

2、找到最近的 NPC

【题目描述】在 2D 游戏的一张地图中随机分布着 n 个 NPC，玩家君莫笑进入地图时随机出生在了一个坐标（x,y）。请找到距离玩家最近的 NPC。假设地图大小为 128*128，NPC 和玩家均不能出现在地图外面。

输入描述：

参数一：整形，玩家出生坐标 x

参数二：整形，玩家出生坐标 y

参数三：整形，NPC 数量 n

参数四：NPC 二维坐标数组的一维表示，使用字符串形式传入，注意逗号前后不要加空格，比如地图中有两个 NPC，坐标分别是(32,33)和(25,25)，则此处传入 32,33,25,25

输出描述：

查询到的 NPC 坐标，注意坐标值前后有圆括号

备注：

NPC 数量不超过 1000 个

输入样例：

32,48,3,33,40,40,50,32,45

输出样例：

(32,45)

【解题思路】

暴力枚举所有的 NPC 位置，然后维护出最小的距离即可。

【参考代码】

```
#include <iostream>
#include <cstdio>
#include <string.h>
#include <memory>

using namespace std;

int main()
{
    int x, y, iNPCNum = 0;
    char iNPCPos[4000];
    int iDistance = 128 * 128;
    int iFindX = 0, iFindY = 0;
    char cTemp;
    cin >> x >> cTemp >> y >> cTemp >> iNPCNum >> cTemp >> iNPCPos;
    int iTemp = 0;
    char iTempPos[5];
    int iX = 0, iY = 0;
    bool bIsX = true;
    for (int i = 0; ; ++i)
    {
        if (iNPCPos[i] == ',' || iNPCPos[i] == '\0')
        {
            memcpy(iTempPos, iNPCPos + iTemp, i - iTemp);
            iTempPos[i - iTemp] = '\0';
            iTemp = i + 1;
            if (bIsX)
            {
                iX = atoi(iTempPos);
                bIsX = false;
            }
            else
            {
                iY = atoi(iTempPos);
                if (iX != 0)
                {
                    if ((iX - x) * (iX - x) + (iY - y) * (iY - y) < iDistance)
```

```
                    {
                        iFindX = iX;
                        iFindY = iY;
                        iDistance = (iX - x) * (iX - x) + (iY - y) * (iY - y);
                    }
                }
                bIsX = true;
            }
        }
        if (iNPCPos[i] == '\0')
            break;
    }
    cout << "(" << iFindX << "," << iFindY << ")" << endl;
    return 0;
}
```

3、学数学

【题目描述】数学老师正在教授小畅和小游两人素数的概念。为了帮助巩固两人的知识，老师说出一个数，要求小游和小畅合作，每人说出一个素数，使得两人说出的素数的和刚好等于老师说出的数。请编写程序计算两人说出的素数对的个数。如，老师说 10，小畅和小游可以说出两对素数，分别为（5,5）和（3,7）（不考虑顺序）。

输入描述:

输入包括一个整数 n,(3 ≤ n < 1000)

输出描述:

输出符合条件的素数对的个数

输入样例：

10

输出样例：

2

【解题思路】

筛出范围的素数，然后枚举统计对数即可。

【参考代码】

```
import java.util.HashSet;
import java.util.Collections;
import java.util.Arrays;
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
```

```
        int n = in.nextInt();
        int res = 0;
        HashSet<Integer> set = new HashSet<Integer>();
        set.add(2);
        boolean[] prime = new boolean[n + 1];
        Arrays.fill(prime, true);
        isPrime(2, prime);
        for(int i = 3; i <= n; i++){
            if(prime[i]){
                isPrime(i, prime);
                set.add(i);
                if(set.contains(n - i))
                    res++;
            }
        }
       System.out.println(res);
    }
    private static void isPrime(int i, boolean[] prime){
        int len = prime.length;
        for(int j = 2; i * j < len; j++){
            prime[i * j] = false;
        }
        return;
    }
}
```

十九、搜狗 2019 秋招笔试真题

1、糖果

【题目描述】小明和小红都很喜欢吃糖果，今天他们一起玩一个糖果游戏。他们将装有不同数量的透明糖果盒子摆放成一个环，现在两人依次选择糖果盒子，小明先拿，且小明第一次挑选可以选择环中任意一个糖果盒子，将环分割成一列有首尾的序列，之后两人每次选择时只能从剩余的糖果盒子序列的行首或者行尾挑选，直到两人将糖果盒子全部拿完，最后糖果多的为胜者。假设两人都希望自己是最后的赢家，且糖果的总数是奇数，现给定一个糖果的环，用一个数组表示环中的各糖果盒子中糖果的数量，数组首尾相连成环，元素个数为 N。请输出胜利者比失败者至少多拿多少糖果。

输入描述:

第一行：N

第 2 至 N+1 行：每行一个数，代表糖果数

输出描述:

一个数，请输出胜利者比失败者多拿多少糖果

输入样例:

4
1
55
41
2

输出样例:

54
小明先选择 55，此时环从 55 处断开，变为序列[41, 2, 1]
小红选择行首的 41，此时剩余的序列为[2, 1]
小明选择行首的 2，此时剩余的序列为[1]
小红选择剩余的 1。
此时小明胜利，比小红多 15 个糖果

【解题思路】

带博弈的动态规划。
设 dp[i][j]为只考虑 i~j 的糖果序列，先手能比后手多的糖果数，则转移方程为
dp[i][j] = max(a[i] - dp[i+1][j], a[j] - dp[i][j-1])
为环的话，只需要枚举第一个取的糖果，即可转为序列上的问题。

【参考代码】

```
#include <iostream>
#include <vector>
using namespace std;

int predictWinner(vector<int> candies) {
    int length = candies.size();
    if (length == 0) return false;
    vector<vector<int> > dp(length, vector<int>(length, 0));
    for (int i = 0; i < length; i++) {
        dp[i][i] = candies[i];
    }
    for (int d = 1; d < length; d++) {
        for (int i = 0; i < length; i++) {
            dp[i][(i + d) % length] = max(candies[i] - dp[(i + 1) % length][(i + d) %
length], candies[(i + d) % length] - dp[i][(i + d - 1) % length]);
        }
    }
    int result = INT_MIN;
    for (int i = 0; i < length; i++) {
        result = max(result, dp[i][(i + length - 1) % length]);
    }
    return result;
```

```
}

int main() {
    int length = 0;
    int candies[10000];
    cin >> length;
    for (int i = 0; i < length; i++) {
        cin >> candies[i];
    }

    vector<int> v(candies, candies+length);
    cout << predictWinner(v) << endl;
}
```

2、死锁

【题目描述】判断一个消息队列是否可能死锁。
消息队列的缓冲区长度为 L 单位，读操作为每次从缓冲区读取 R 单位，写操作为每次写入缓冲区 W 单位。消息队列会持续进行读写操作。具体为写操作会在缓冲区还剩余大于等于 W 单位空间时保持进行，当缓冲区内空间小于 W 时，写操作停止，等待读操作进行；类似的，读操作会在缓冲区可读内容大于等于 R 时保持进行，当可读内容小于 R 时，读操作停止，等待写操作进行。读写都是原子操作。
若读写操作均无法进行，定义此时状态为死锁。
给定 L,R,W，问消息队列是否可能进入死锁状态，若能，输出 YES，否则输出 NO

输入描述:

第 1 行输入 N(N<=10)表示数据组数。
从第 2 行到第 N+1 行，每行三个整数 L,R,W。

输出描述:

输出 N 行，每行输出'YES'或'NO'

备注:

50% L,R,W<=10^5
50% L,R,W<=10^18

输入样例:

```
2
5  2  3
5  3  4
```

输出样例:

```
NO
YES
```

【解题思路】

死锁产生的条件为：
缓冲区内空间小于 W，即 L - x < W => X > L - W

可读内容小于 R，即 X < R
这等价于解不定方程 L - W < Rx - Wy < R
事实上，Rx - Wy 的所有取值为 gcd(R, W)的倍数，所以只需要判断(L-W, R)内是否存在 gcd(R, W)的倍数即可。

【参考代码】

```
#include<cstdio>

using namespace std;

#define ll long long
ll gcd(ll a,ll b){ return b==0?a:gcd(b,a%b);}

int main()
{
    int N;
    scanf("%d",&N);
    while(N--)
    {
        ll L,R,W,g;
        scanf("%lld%lld%lld",&L,&R,&W);
        g = W<R?gcd(R,W):gcd(W,R);
        if(W+R-g>L) puts("YES"); else puts("NO");
    }
}
```

3、古巴比伦迷宫

【题目描述】一群探险家被困古巴比伦迷宫 ，经过努力，探险家破译出了密码。原来通关需要先打开前方的 M 个机关的任意一个或多个为打开状态，然后关闭所有机关才能安全通过。探险家还发现了一种带有凸起的圆盘，圆盘可以用来同时反转若干个机关的状态（打开状态反转为关闭状态，关闭状态反转为打开状态，这若干个机关必须同时反转）。为了速记，探险家们用十六进制数字表示一个圆盘。如圆盘能同时控制第 3、4、5 个开关，圆盘就记为 1C（11100）。每次操作一个圆盘，比特位为 1 的位置的机关同时被反转；而且每个圆盘使用一次后就会碎掉。目前探险家收集到了 N 个圆盘，问现在探险家可以顺利走出迷宫么？
若探险家手里有 0、1、2、3、5 五个圆盘，M=7，由于存在 1 xor 2 xor 3 = 0， 因此结果是 yes；
若探险家手里有 0、1、2、2、5 五个圆盘，M=7，由于存在 2 xor 2 = 0， 因此结果是 yes；
若探险家手里有 0、1、3、5、9 五个圆盘，M=7，由于不存组合使得 xor = 0， 因此结果是 no。

输入描述:

包含多组数据，第一行是数据组数。
每组数据中首行为 M N，M 为机关个数，M <= 360(大部分情况下小于 40)，N 为圆盘个数，
N<=50000；其余为 N 行圆盘的 16 进制 ID，可以保证每行输入小于 2^M

输出描述:

每组一行输出，满足条件输出 yes，反之输出 no

输入样例:

2
2 3
0
2
2
5 2
4
1C

输出样例:

yes
no

【解题思路】

题目抽象:

题目可以简化为在 N 个最大值为 2^M - 1 数中，能否找到一组数，使得这组数的异或值为 0。如果按照每个数的 1~M 个 bit 位计算，这个问题可以转化为 MN 的线性方程。

以 M = 2 、N = 5、【0、1、2、2、3】为例，，由于 0 不能控制开关，首先需要把 0 去掉，此时数组为 1、2、2、3 四个数，我们假设 X1、X2、X3、X4 代表是否使用对应的数字，第一、二个 bit 为 0 的函数组是：

(X1 + X4) mod 2 = 0
(X2 + X3 + X4) mod 2 = 0

此方程组是线性齐次方程组，由于未知数个数大于等式个数，因此必然存在非 0 解。

线性方程可以转换为矩阵计算，上述方程组可以对应为：

```
1 0 0 1                    0
      xor [X1 X2 X3 X4] =
0 1 1 1                    0
```

样例算法:

交换矩阵的第 i、j 行，在调用前判断了 i != j

inline void swap(char MN[BIT][BIT], int i, int j)

按照线性方程求解的过程实现消元法求解方程

void reduce(char MN[BIT][BIT], int M, int N)

函数过程：

1、过滤输入的 0
2、将输入的 16 进制转换为 2 进制写入数组
3、当 N ==0 或 N > M 时，直接返回结果
4、N <= M 时，消元求解

【参考代码】

```
#include <stdio.h>
#include <string.h>
#include <stdint.h>
```

```
#define BIT 360

inline void swap(char MN[BIT][BIT], int i, int j)
{
    char tmp[BIT];
    memcpy(tmp, MN[i], BIT);
    memcpy(MN[i], MN[j], BIT);
    memcpy(MN[j], tmp, BIT);
}
void reduce(char MN[BIT][BIT], int M, int N)
{
    for(int i = 0; i < N; i++)
    {
        // 找到第 i 列为 1 的行，放到第 i 行
        for(int j = i; j < M; j++)
        {
            if(MN[j][i] == 1)
            {
                if(j != i)
                {
                    swap(MN, j, i);
                }
                break;
            }
        }
        // 第 i+1 到 M 行，第 i 列消 0
        for(int j = i + 1; j < M; j++)
        {
            if(MN[j][i] == 1)
            {
                for(int m = i; m < N; m++)
                {
                    MN[j][m] ^= MN[i][m];
                }
            }
        }
    }
    // 最后都为 0 才可以
    for(int i = N - 1; i < M; i++)
    {
        if(MN[i][N-1] != 0)
        {
```

```
            printf("no\n");
            return;
        }
    }
    printf("yes\n");
}
int main()
{
    int n;
    char line[256];
    char MN[BIT][BIT];
    scanf("%d", &n);
    for(int t1 = 0; t1 < n; t1++)
    {
        int M, N = 0, LN;
        scanf("%d %d", &M, &LN);
        memset(MN, 0, sizeof(MN));
        for(int t2 = 0; t2 < LN; t2++)
        {
            int idx = 0;
            scanf("%s", line);
            if(*line == '0')
            {
                continue;
            }
            else
            {
                if (N < M)
                {
                    int len = strlen(line);
                    while(len > 0)
                    {
                        len--;
                        uint8_t w;
                        if(line[len] >= '0' && line[len] <= '9')
                        {
                            w = line[len] - '0';
                        }
                        else
                        {
                            w = line[len] + 10 - 'A';
                        }
```

```
                        for(int x = 0; x < 4; x++)
                        {
                            MN[idx++][N] = w % 2;
                            w = w / 2;
                        }
                    }
                }
                N++;
            }
        }
        if(N == 0)
        {
            printf("no\n");
        }
        else if(N > M)
        {
            printf("yes\n");
        }
        else
        {
            // 消元法解线性方程
            reduce(MN, M, N);
        }
    }
    return 0;
}
```

二十、深信服 2019 秋招笔试真题

1、Cuboid

【题目描述】一个长方体，长宽高分别为 x,y,z，都为自然数。

现在要把若干个相同的立方体摆成高为 N 的一根柱形体。

每层摆 1 个，如果两种摆法的高度是一样的，则认为这两种摆法等价，所以每层只有三种摆法。

求一共有多少种摆法。

输入描述:

第一行为一个数字 N，N>=1 且 N<=100

第二行为长方体的长宽高,x、y、z 都为无符号整数，按升序排列。

输出描述:

摆法总数，已知该总数会小于 10000000

【解题思路】

动态规划。

dp[i] = dp[i-x] + dp[i-y] + dp[i-z]

【参考代码】

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef unsigned int U32;

unsigned long long cnt[1000];

unsigned long long calc(U32 n, U32 x, U32 y, U32 z)
{
    U32 i;
    cnt[0] = 1;
    for (i = 0; i <= n; ++i) {
        if (i >= x && cnt[i - x])
            cnt[i] += cnt[i - x];

        if (i >= y && cnt[i - y])
            cnt[i] += cnt[i - y];

        if (i >= z && cnt[i - z])
            cnt[i] += cnt[i - z];
    }
    return cnt[n];
}

int main()
{
    U32 n;
    U32 x, y, z;

    scanf("%u\n", &n);
    scanf("%u %u %u", &x, &y, &z);

    printf("%llu\n", calc(n, x, y, z));
    return 0;
}
```

2、IPrange

【题目描述】一个数字段由首尾两个数字标识，表示一个自然数集合，
比如数字段[beg, end)表示从 beg 到 end 之间的所有自然数，
包含 beg，但不包含 end。
有若干个数字段，这些数字段之间可能有重叠，
怎么把这些数字段合并去重，用最少个数的数字段来表示。
合并前后，整个集合包含的数字不发生变化。

输入描述:

第一行为数字 N，表示接下来有 N 个数字段(N<=100000)
第二行开始一共有 N 行，每行两个数字，分别表示一个数字段的 beg 和 end
(beg 和 end 为无符号 32 位整数)

输出描述:

合并去重后形成的数字段集合，按升序排列。

【解题思路】

按区间左端点排序后，保留当前区间的左端点，不停合并的同时维护最右端点，当下一个区间的左端点不与之前的最右端点重合时，之前的那些区间就合并完成了。

【参考代码】

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

struct range {
    unsigned int beg;
    unsigned int end;
};

int compare(struct range *lhs, struct range *rhs)
{
    if (lhs->beg < rhs->beg)
        return -1;
    else if (lhs->beg > rhs->beg)
        return 1;
    if (lhs->end < rhs->end)
        return -1;
    else if (lhs->end > rhs->end)
        return 1;
    return 0;
}

void sort(struct range *sets, size_t cnt)
```

```
{
    size_t i, j;
    for (i = 0; i < cnt; ++i) {
        for (j = 0; j < cnt - i - 1; ++j) {
            int ret = compare(&sets[j], &sets[j + 1]);
            if (ret > 0) {
                struct range tmp;
                tmp = sets[j];
                sets[j] = sets[j + 1];
                sets[j + 1] = tmp;
            }
        }
    }
}
#define MAX(lhs, rhs) ((lhs) > (rhs) ? (lhs) : (rhs))

int merge(struct range *sets, size_t cnt)
{
    size_t i;
    size_t rp = 0;
    size_t wp = 0;

    qsort(sets, cnt, sizeof(*sets), (int (*)(const void *, const void*))compare);
//    sort(sets, cnt);
//    for (i = 0; i < cnt; ++i) {
//        printf("%d %d\n", sets[i].beg, sets[i].end);
//    }
//    printf("\n");

    for (i = 1; i < cnt; ++i) {
        if (sets[i].beg <= sets[wp].end) {
            sets[wp].end = MAX(sets[wp].end, sets[i].end);
        } else {
            ++wp;
            sets[wp] = sets[i];
        }
    }
    return wp + 1;
}

int main()
{
```

```
    struct range sets[100000];

    int n;
    int i;

    scanf("%d\n", &n);
    for (i = 0; i < n; ++i) {
        scanf("%d %d\n", &sets[i].beg, &sets[i].end);
    }
    n = merge(sets, n);

    for (i = 0; i < n; ++i) {
        printf("%d %d\n", sets[i].beg, sets[i].end);
    }
    return 0;
}
```

3、Match

【题目描述】从字符串 string 开始完整匹配子串 sub，返回匹配到的字符个数。
sub 中如果出现'?'表示可以匹配一到三个除'\0'以外的任意字符。
如果 sub 还有找不到匹配的字符，则说明不能完整匹配。
如果能完整匹配，返回匹配到的字符个数，如果有多种匹配方式，返回匹配字符数最少的那个，如果不能完整匹配，返回-1

输入描述:

第一行输入字符串 string，长度小于 10000，第二行输入子串 sub，长度小于 100.

输出描述:

从 string 开头位置完整匹配 sub，匹配到的字符个数。

```
string:abcdefg
sub:a?C
result:3
string:aabcddefg
sub:a?C
result:4
string:aabcddefg
sub:b?E
result:-1
string:aabcddefg
sub:a?D
result:5
```

【解题思路】

动态规划。
dp[i][j]表示主串 s(1~i)是否能匹配模式串 t(1~j)，1 表示能，0 表示不能。
若 t[j] = ‘?’，则 dp[i][j] = dp[i-1][j-1] | dp[i-2][j-1] | dp[i-3][j-1]
若 t[j] != ‘?’，则 dp[i][j] = dp[i-1][j-1] & (s[i] == t[j])
初始条件为 dp[0][0] = 1
最后，满足 dp[i][|t|] = 1 的最小 i 即为答案。若不存在这样的 i，则无法匹配。

【参考代码】
（暴力）

```
#include <stdio.h>
#include <string.h>
#include <malloc.h>

//返回匹配字符个数
static int match(const char *str, const char *sub)
{
    int cnt = 0;
    while (*sub && *sub == *str) {
        ++str;
        ++sub;
        ++cnt;
    }
    if (*sub == '?') {
        int i, ret;
        for (i = 1; i <= 3; ++i) {
            ret = match(str + i, sub + 1);
            if (ret >= 0)
                return cnt + ret + i;
        }
        return -1;
    } else if (*sub == '\0') {
        return cnt;
    } else {
        return -1;
    }
}

char string[12800];
char sub[128];

int main()
{
```

```
    int ret = 0;

    scanf("%12800s\n", string);
    scanf("%128s\n", sub);

    ret = match(string, sub);

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

二十一、bilibili 2019 秋招笔试真题

1、山寨金闪闪

【题目描述】金闪闪死后，红 A 拿到了王之财宝，里面有 n 个武器，长度各不相同。红 A 发现，拿其中三件武器首尾相接，组成一个三角形，进行召唤仪式， 就可以召唤出一个山寨金闪闪。（例如，三件武器长度为 10、15、20，可以召唤成功。若长度为 10、11、30，首尾相接无法组成三角形，召 唤失败。）红 A 于是开了一个金闪闪专卖店。他把王之财宝排成一排，每个客人会随机抽取到一个区间[l,r],客人可以选取区间里的三件武器进行 召唤（客人都很聪慧，如果能找出来合适的武器，一定不会放过）。召唤结束后，客人要把武器原样放回去。m 个客人光顾以后，红 A 害怕过多 的金闪闪愉悦太多男人，于是找到了你，希望你帮他统计出有多少山寨金闪闪被召唤出来。

输入描述:

第一行武器数量:n <= 1*10^7

第二行空格分隔的 n 个 int，表示每件武器的长度。

第三行顾客数量：m <= 1*10^6

后面 m 行，每行两个 int l，r，表示每个客人被分配到的区间。（l<r）

输出描述:

山寨金闪闪数量

输入样例:

```
5
1 10 100 95 101
4
1 3
2 4
2 5
3 5
```

输出样例:

```
3
```

【解题思路】

一个序列中任意三个数不能组成三角形，等价于序列排序后相邻的 3 个数不能组成三角形。考虑构造一个最长的不能组成三角形的序列，容易发现是 1 2 3 5 8...斐波拉契数列，在 10^6 范围内最长只有 30。所以，对于一个询问，如果 r-l+1 > 30，则肯定能构成三角形，否则，排序后依次考查相邻的三个数即可。

复杂度 O(m*30log30)

【参考代码】

```
#include <iostream>
#include <algorithm>
using namespace std;
int list[11000000], arr[50];
int main() {
    int m, n, result, l, r;
    int a1, a2, a3, flag;
    cin >> n;
    for(int i = 1; i <= n; ++i) {
        cin >> list[i];
    }
    cin >> m;
    result = 0;
    for (int i = 0; i < m; ++i) {
        cin >> l >> r;
        int count = r - l;
        if (count > 60) {
            ++result;
        } else {
            for(int j = l; j <= r; ++j) {
                arr[j - l] = list[j];
            }
            sort(arr, arr + count + 1);
            flag = 0;
            for (int j = 0; j < count - 1; j ++) {
                if (arr[j] + arr[j + 1] > arr[j + 2]) {
                    flag = 1;
                    break;
                }
            }
            if (flag == 1) ++result;
        }
    }
    cout << result << endl;
    return 0;
```

```
}
```

2、红茶

【题目描述】高贵的蕾米莉亚大小姐每天需要饮用定量 B 型血的红茶以保持威严，并且要分两杯在不同时段饮用。 女仆长十六夜咲夜每天可以制作很多杯不同剂量 B 型血的红茶供蕾米莉亚大小姐饮用。 某日，你和天才妖精琪露诺偷偷潜入红魔馆被咲夜抓住，要求在今日份的红茶中挑出所有满足大小姐要求的茶杯，否则……

输入描述:

每个样例有三行输入，第一行输入表示茶杯个数，第二行输入表示每份茶杯里的 B 型血剂量，第三行表示大小姐今天的定量

输出描述:

对每一个样例，输出所有可能的搭配方案，如果有多种方案，请按每个方案的第一杯 B 型血剂量的大小升序排列。 如果无法找到任何一种满足大小姐的方案，输出一个字符⑨并换行。

输入样例:

```
7
2 4 6 1 3 5 7
7
```

输出样例:

```
1 6
2 5
3 4
```

【解题思路】

先确定一个 a[i]，问题变为查找是否存在 S-a[i]，这可以将数组排序后二分查找。

复杂度 O(nlogn)

【参考代码】

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define NOT_FOUND -2
#define MAXN 0xffff
int a[MAXN];
int check[MAXN];
int cmp(const void* a, const void* b) {
    return *(int*)a - *(int*)b;
}
int binary_search(int a[], int left, int right, int key) {
    int mid;
    while(left <= right) {
```

```
        mid = (right + left) / 2;
        if(a[mid] == key) return mid;
        else if(a[mid] > key)
            right = mid - 1;
        else if(a[mid] < key)
            left = mid + 1;
    }
    return NOT_FOUND;
}
int main(void) {
    int N,x;
    int i;
    int key,pos;
    int no_way_count = 0;
    scanf("%d", &N);
    for(i=0; i<N; i++) {
        scanf("%d", &a[i]);
    }
    scanf("%d", &x);
    qsort(a, N, sizeof(a[0]), cmp);
    memset(check, 0, sizeof(check));
    for(i=0; i<N; i++) {
        if(1 == check[i]) continue;
        key = x - a[i];
        pos = binary_search(a, 0, N-1, key);
        if(NOT_FOUND != pos && i != pos) {
            printf("%d %d\n", a[i], key);
            check[pos] = 1;
        } else {
            no_way_count++;
        }
    }
    if ( no_way_count == N ) {
        printf("⑨\n");
    }
    return 0;
}
```

3、简单表达式计算

【题目描述】给定一个合法的表达式字符串，其中只包含非负整数、加法、减法以及乘法符号（不会有括号），例如 7+3*4*5+2+4-3-1，请写程序计算该表 达式的结果并输出

输入描述:

输入有多行，每行是一个表达式，输入以 END 作为结束；

输出描述:

每行表达式的计算结果；

输入样例:

7+3*4*5+2+4-3-1

2-3*1

END

输出样例:

69

-1

【解题思路】

（python 的 eval 秒

用栈将中缀表达式转为后缀表达式再求值。

【参考代码】

```
#include<stack>
#include<iostream>
using namespace std;
bool isOp(char c) {
    return c=='+'||c=='-'||c=='*';
}
int evaluate(const string& expr) {
    if (expr.empty()) {
        return 0;
    }
    stack<int> operand;
    operand.push(0);
    stack<char> ops;
    ops.push('+');
    int preSum=0;
    for(size_t i=0; i<expr.size(); i++) {
        if ( expr[i]==' ')
            continue;
        else if ( isOp(expr[i]) )
            ops.push(expr[i]);
        else if (isdigit(expr[i] )) {
            preSum=preSum*10+expr[i]-'0';
            if ( i==expr.size()-1||!isdigit(expr[i+1])) {
                operand.push(preSum);
                preSum=0;
                if ( ops.top()=='*' ) {
```

```
                    int a =operand.top();
                    operand.pop();
                    int b =operand.top();
                    operand.pop();
                    ops.pop();
                    operand.push(a*b);
                } else if ( ops.top()=='-' ) {
                    int a=operand.top();
                    operand.pop();
                    ops.top()='+';
                    operand.push(-a);
                }
            }
        } else {
            return -1;
        }
    }
    while(!ops.empty()) {
        int a =operand.top();
        operand.pop();
        int b =operand.top();
        operand.pop();
        int type= ops.top();
        ops.pop();
        int t=type=='+'?a+b:b-a;
        operand.push(t);
    }
    return operand.top();
}
int main() {
    char buf[1024];
    while (scanf("%s", buf)) {
        if (std::string(buf) == "END") {
            break;
        }
        printf("%d\n", evaluate(buf));
    }
}
```

二十二、小红书 2019 秋招笔试真题

1、每 K 个一组反转链表

【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。

说明：

1. 你需要自行定义链表结构，将输入的数据保存到你的链表中；
2. 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；
3. 你的算法只能使用常数的额外空间。

输入描述：

第一行输入是链表的值

第二行输入是 K 的值，K 是大于或等于 1 的整数

输入形式为：

1 2 3 4 5

2

输出描述：

当 k = 2 时，应当输出：

2 1 4 3 5

当 k = 3 时，应当输出：

3 2 1 4 5

当 k=6 时，应当输出：

1 2 3 4 5

输入样例：

1 2 3 4 5

2

输出样例：

2 1 4 3 5

【解题思路】

实现链表反转操作之后，然后根据题设完成。

【参考代码】

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        int k = Integer.valueOf(sc.nextLine());
        String[] strings = str.split(" ");
        Node head = new Node(-1);
        Node tail = head;
```

```
        for(String val : strings){
            Node node = new Node(Integer.valueOf(val));
            tail.next = node;
            tail = tail.next;
        }
        head = head.next;

        Node newHead = reverseKGroup(head,k);
        String res = "";
        while(null != newHead){
            res = res + newHead.val + " ";
            newHead=newHead.next;
        }
        System.out.print(res.substring(0,res.length()-1));

    }

    public static Node reverseKGroup(Node head,int k){
        if(null == head || 0 == k){
            return null;
        }
        Node prevStart = head;
        Node prevEnd = head;
        int count = k -1;
        while(count-- !=0){
            Node next = prevEnd.next;
            if(null == next){
                break;
            }
            prevEnd = next;
        }
        if(count != -1){
            return  head;
        }
        Node onePastEnd = prevEnd.next;
        prevEnd.next = null;

        Node rest = reverseKGroup(onePastEnd, k);
        Node reversed = reverse(prevStart);
        reversed.next = null;

        prevStart.next = rest;
```

```
        return  prevEnd;
    }

    public static Node reverse(Node head){
        if(null == head){
            return  head;
        }
        if(null == head.next){
            head.next = head;
            return  head;
        }
        Node lastOfRest =  reverse(head.next);
        Node newHead = lastOfRest.next;
        lastOfRest.next = head;
        head.next = newHead;
        return  head;
    }
}

class Node{
    int val;
    Node next;

    Node(int val){
        this.val = val;
        next = null;
    }
}
```

2、求表达式 f(n)结果末尾 0 的个数

【题目描述】输入一个自然数 n，求表达式 f(n) = 1!2!3!.....n! 的结果末尾有几个连续的 0?

输入描述:

自然数 n

输出描述:

f(n)末尾连续的 0 的个数

输入样例：

11

输出样例：

9

【解题思路】

本质是统计 5 的个数。

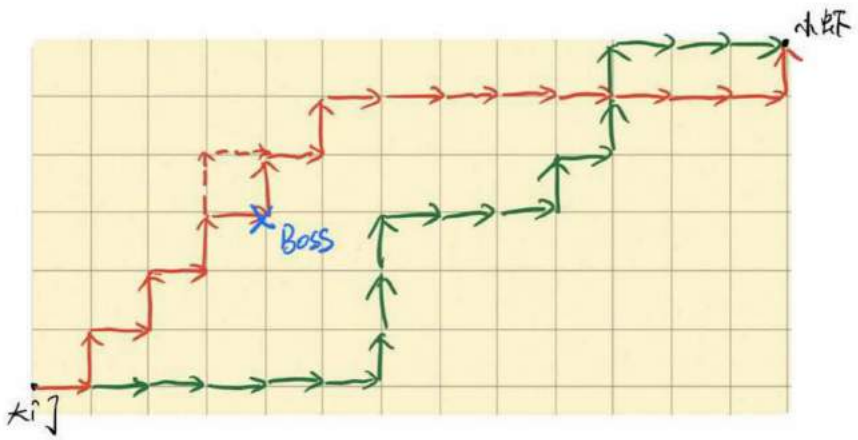
【参考代码】

```
import os
def zero_cnt(n):
    if n < 0:
        return 0;
    cnt = 0;
    while n:
        cnt += n/5;
        n = n/5;
    return cnt;

if __name__ == "__main__":
    tmp_str = raw_input("");
    cnt = 0
    n = int(tmp_str);
    for i in range(1,n+1):
        cnt += zero_cnt(i);
    print cnt;
```

二十三、Shopee 2019 秋招笔试真题

1、Shopee 的办公室

【题目描述】Shopee 的办公室非常大，小虾同学的位置坐落在右上角，而大门却在左下角，可以把所有位置抽象为一个网格（门口的坐标为 0，0），小虾同学很聪明，每次只向上，或者向右走，因为这样最容易接近目的地，但是小虾同学不想让自己的 boss 们看到自己经常在他们面前出没，或者迟到被发现。他决定研究一下如果他不通过 boss 们的位置，他可以有多少种走法？



输入第一行表示数据样例数 T，表示接下来会有 T 个相同格式的样例输入

每个样例输入如下

第一行 x,y,n （0<x<=30， 0<y<=30， 0<=n<= 20） 表示 x,y 小虾的座位坐标,n 表示 boss 的数量（ n <= 20）

接下来有 n 行，表示 boss 们的坐标

x1， y1

x2， y2

……

xn， yn

对于每个样例需要输出小虾的走法

【解题思路】

动态规划，dp[i][j] = dp[i-1][j] + dp[i][j-1]。

如果某个地方有 boss，则 dp[i][j] = 0。

【参考代码】

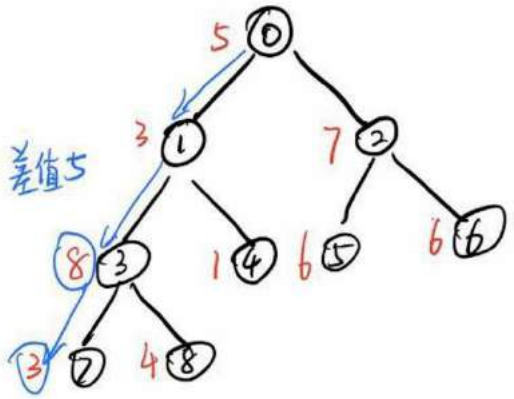
```
#include<cstdio>
#include<algorithm>
#include<iostream>
#include<cstring>
#include<string>
#include<cmath>
#include<vector>
#include<queue>
using namespace std;
long long dp[63][63];
struct point{
	int x, y;
}boss[12];

int init(){
	memset(dp,0,sizeof(dp));
	dp[0][0] = 1;
	for(int x=1; x < 63; x++ ){
		dp[x][0] = 1;
		for(int y=1; y<=x; y++){
			dp[x][y] = dp[x-1][y-1] + dp[x-1][y];
		}
	}
	return 0;
}

bool cmp(point a, point b){
	return a.x < b.x?true: a.x == b.x?a.y <= b.y:false;
```

```
}

long long count(int x, int y, int n){
	long long ans = 0;
	for(int i=1; i<(1<<n); i++){
		long long tmp = 1;
		bool flag = false;
		int bit = 0, posy = 0, posx = 0;
		for(int j=0; j<n; j++){
			if(i&(1<<j)){
				bit++;
				if(boss[j].y < posy){
					flag = true;
					break;
				}
				tmp *= dp[boss[j].x-posx+boss[j].y-posy][boss[j].x-posx];
				posx = boss[j].x;
				posy = boss[j].y;
			}
		}
		if(flag){
			continue;
		}
		tmp *= dp[x-posx+y-posy][x-posx];
		if(!(bit&1)){
			tmp *= -1;
		}
		ans += tmp;
	}
	return ans;
}

int main()
{
	init();
	int T,x,y,n;
	scanf("%d",&T);
	for(int t=1;t<=T;t++)
	{
		scanf("%d %d %d",&x,&y,&n);
		for(int i=0; i<n; i++){
			scanf("%d %d",&boss[i].x,&boss[i].y);
```

```
            }
            sort(boss, boss+n, cmp);
            printf("%lld\n", dp[x+y][x]-count(x, y, n));
        }
        return 0;
}
```

2、Shopee 的 Orange Day

【题目描述】Shopee 每个月都有属于大家的节日，每到这个时候，大家都会穿着橙色的 T 恤，吃着水果蛋糕，做着游戏。瞧，今天又是 Orange Day 了，吃货小虾同学早早的来到现场，看看有没有什么吃的，刚刚走进去就发现我们的前台 mm 愁眉苦脸的看着挂满礼物的发财树，细看发现，发财树的树枝因为承重不均，东倒西歪。是时候展现真正的技术啦，小虾同学，马上走上去，让前台 mm 把挂礼物的方案拿出来，前台 mm 拿出一大叠方案，小虾同学顿时傻眼了，但是他马上想到了一个方法，写一个程序判断每种方案的承重误差，把不合格的都干掉就行了。经过分析，把发财树抽象为一颗树。只要发财树的从顶部到各个叶子的各个路径上面的最大最小重量的差值在方案的误差范围内，就可以正常承重。



输入第一行表示数据样例数 T，表示接下来会有 T 个相同格式的样例输入
每个样例输入如下
输入的第一行输入 n,（2 ≤ n ≤ 100000，节点的编号为 0 到 n-1,0 为根节点）组成，
第二行有 n 个数，表示每个节点挂的礼物的重量 w（1<= w<= 1000）
下面是 n-1 行，每行有两个整数，第一个数表示父节点的编号，第二个数表示子节点的编号
输出最大差值。

【解题思路】

路径数量为叶子节点数量。
对每个叶子节点，可以算出从根到该点路径上的最大值和最小值。
可以通过动态规划减少计算量。

【参考代码】

```
#include<cstdio>
#include<algorithm>
```

```
#include<iostream>
#include<cstring>
#include<string>
#include<cmath>
#include<vector>
#include<queue>
using namespace std;
int fa[100005], weight[100005], dp[100005][2];
bool leaf[100005];
int dp_dfs(int x){
	if(dp[x][0]==0){
		dp_dfs(fa[x]);
		dp[x][0] = max(weight[x], dp[fa[x]][0]);
		dp[x][1] = min(weight[x], dp[fa[x]][1]);
	}
	return 0;
}
int main()
{
	int T,n,x,y;
	scanf("%d",&T);
	for(int t=1;t<=T;t++)
	{
		scanf("%d",&n);
		memset(fa,0,sizeof(fa));
		memset(dp,0,sizeof(dp));
		for(int i=0; i<n; i++){
			scanf("%d",&weight[i]);
			leaf[i] = true;
		}
		for(int i=0; i<n-1; i++){
			scanf("%d %d",&x,&y);
			fa[y] = x;
			leaf[x] = false;
		}
		dp[0][0] = dp[0][1] = weight[0];
		int max_diff = 0;
		for(int i=0; i<n; i++){
			if(leaf[i]){
				dp_dfs(i);
				max_diff = max(max_diff, dp[i][0]-dp[i][1]);
			}
```

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

3、Shopee 的零食柜

【题目描述】Shopee 的零食柜有着各式各样的零食，但是因为贪吃，小虾同学体重日益增加，终于被人叫为小胖了，他终于下定决心减肥了。他决定每天晚上去操场跑两圈，但是跑步太累人了，他想转移注意力，忘记痛苦，正在听着音乐的他突然有个想法，他想跟着音乐的节奏来跑步，音乐有 7 种音符，对应的是 1 到 7，那么他对应的步长就可以是 1-7 分米，这样的话他就可以转移注意力了，但是他想保持自己跑步的速度，在规定时间 m 分钟跑完。为了避免被累死，他需要规划他每分钟需要跑过的音符，这些音符的步长总和要尽量小。下面是小虾同学听的歌曲的音符，以及规定的时间，你能告诉他每分钟他应该跑多少步长？
输入第一行表示数据样例数 T，表示接下来会有 T 个相同格式的样例输入
每个样例输入如下
输入的第一行输入 n（1 ≤ n ≤ 1000000，表示音符数），m（1<=m< 1000000, m <= n）组成，
第二行有 n 个数，表示每个音符（1<= f <= 7）

【解题思路】

二分答案，然后贪心的 check。
Check 的时候，只要和不超过该分钟的步长，就加入；否则加入下一分钟。最后判断所需分钟数是否超过 m。

【参考代码】

```
#include<cstdio>
#include<algorithm>
#include<iostream>
#include<cstring>
#include<string>
#include<cmath>
#include<vector>
#include<queue>
using namespace std;
int a[1000005], n, m;
bool check(int l){
        int cnt=0, tmp=0;
        for(int i=0; i<n; i++){
                if(tmp+a[i]<= l){
                        tmp += a[i];
                }else{
                        tmp = a[i];
```

```
                        cnt++;
                }

        }
        return m >= cnt+1;
}

int find(){
        int r, l, mid, ans;
        r = n*7;
        l = n;
        ans = r;
        while(l<=r){
                mid = (l+r)>>1;
                if(check(mid)){
                        ans = min(mid, ans);
                        r = mid-1;
                }else{
                        l = mid+1;
                }
        }
        return ans;
}

int main()
{
        int T;
        scanf("%d",&T);
        for(int t=1;t<=T;t++)
        {
                scanf("%d %d",&n, &m);
                for(int i=0; i<n; i++){
                        scanf("%d",&a[i]);
                }
                printf("%d\n", find());
        }
        return 0;
}
```

二十四、乐信 2019 秋招笔试真题

1、3 数和

【题目描述】给定一个包括 n 个整数的数组 nums 和 一个目标值 target。找出 nums 中的三个整数，使得它们的和与 target 最接近。返回这三个数的和。假定每组输入只存在唯一答案

输入描述:

nums = [-1，2，1，-4] target = 1

输出描述:

与 target 最接近的三个数的和为 2. (-1 + 2 + 1 = 2)

输入样例：

[-1, 2, 1, -4]

1

输出样例：

2

【解题思路】

预处理所有可能的两数之和，枚举两数之和，二分查找与 target 和两数之和的差最接近的数。

【参考代码】

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;

/**
 * Created by billylin on 2018/9/14.
 */
public class Main {



    public static int threeSumClosest(int[] nums, int target)
    {
        // sort the nums array
        Arrays.sort(nums);
        int closestSum = 0;
        int diff = Integer.MAX_VALUE;


        // iterate nums array, no need for the last two numbers because we need at least
three numbers
        for(int i=0; i<nums.length-2; i++)
```

```
        {
            int left = i + 1;
            int right = nums.length - 1;

            // use two pointers to iterate rest array
            while(left < right)
            {
                int temp_sum = nums[i] + nums[left] + nums[right];
                int temp_diff = Math.abs(temp_sum - target);
                // if find a new closer sum, then update sum and diff
                if(temp_diff < diff)
                {
                    closestSum = temp_sum;
                    diff = temp_diff;
                }

                if(temp_sum < target) // meaning need larger sum
                    left++;
                else if(temp_sum > target) // meaning need smaller sum
                    right--;
                else // meaning temp_sum == target, this is the closestSum
                    return temp_sum;
            }
        }

        return closestSum;
    }


    public static int[] stringToIntegerArray(String input) {
        input = input.trim();
        input = input.substring(1, input.length() - 1);
        if (input.length() == 0) {
            return new int[0];
        }

        String[] parts = input.split(",");
        int[] output = new int[parts.length];
        for(int index = 0; index < parts.length; index++) {
            String part = parts[index].trim();
            output[index] = Integer.parseInt(part);
        }
```

```
        return output;
    }

    public static void main(String[] args) throws IOException {
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        String line;
        while ((line = in.readLine()) != null) {
            int[] nums = stringToIntegerArray(line);
            line = in.readLine();
            int target = Integer.parseInt(line);

            int ret = threeSumClosest(nums, target);

            String out = String.valueOf(ret);

            System.out.print(out);
        }
    }
}
```

2、字符串距离

【题目描述】给定一个字符串 str，和一个字母 ch，请实现相应的代码求出一个数组，使数组中每个数字表示该位置与字母 ch 之间的最短距离。

比如 str=”lexinfintech” ch=”i”

则输出为：[3,2,1,0,1,1,0,1,2,3,4,5]

输入描述:

第一行为字符串

第二行为字母

输出描述:

一个数字数组

备注:

假定所有输入的字符 ch 都在字符串 str 中，且 str 中的所有字母为小写，str 长度不超过 10000

输入样例：

lexinfintech

i

输出样例：

[3,2,1,0,1,1,0,1,2,3,4,5]

【参考代码】

```
import java.util.Scanner;
public class Main {
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        char ch = sc.nextLine().charAt(0);
        int[] len = new int[str.length()];
        for(int i = 0; i < str.length(); i++) {
            char ltemp = str.charAt(i);
            char rtemp = str.charAt(i);
            int left = i, right = i;
            while(ltemp != ch && rtemp != ch) {
                if(left > 0)
                    left--;
                if(right < str.length()-1)
                    right++;
                if(str.charAt(left) == ch) {
                    len[i] = Math.abs(left - i);
                    break;
                }
                if(str.charAt(right) == ch){
                    len[i] = Math.abs(right - i);
                    break;
                }
                ltemp = str.charAt(left);
                rtemp = str.charAt(right);
            }
        }
        System.out.print("[");
        for(int i = 0; i < len.length-1; i++)
            System.out.print(len[i]+",");
        System.out.println(len[len.length-1]+"]");
    }
}
```

二十五、招行信用卡中心 2019 秋招笔试真题

1、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

输入样例:

2

输出样例:

(()),()()

【解题思路】

卡特兰数。

【答案及解析】

```
public static void test3(){
        Scanner scanner=new Scanner(System.in);
        while (scanner.hasNext()){
            int n=scanner.nextInt();
            test33(n,n,"");
            for (int i=strList.size()-1;i>=0;i--){
                System.out.print(i>0?strList.get(i)+",":strList.get(i)+"\n");
            }
            strList.clear();
        }
}
public static void test33(int l,int r,String s){
        if (l==0&&r==0){
            strList.add(s);
            return;
        }
        if (l<r){
            test33(l,r-1,s+")");
        }
        if (l>0){
            test33(l-1,r,s+"(");
        }
}
```

2、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。

输入样例：

abcabc

输出样例：

abc

【解题思路】

枚举所有可能的子串，然后看能否拼接完成。

【参考代码】

```
public static void test11(String str){
            int n = str.length();
            int m;
            int flag=0;
            for(int i=n/2; i>=1; i--){     //i 为原字符串的可重复子串的子串的长度；对每种可能长度的子串进行遍历，i 为子串长度。
                if( n%i == 0 ){         //n%i == 0：长度为 i 的子串可以切分 str，否则，str 的长度不能整除 i，表示 str 就不可能由若干子串重复组成
                    m = n/i;            //m: str 中长度为 i 的子串的个数
                    for(int j=0; j <i; j++){      //比较 m 个长度为 i 的子串是否相等，j 遍历子串的每个字符
                        flag=0;                 //标记所有子串所有元素是否相等
                        for(int k=1;k<m;k++) {          //k 遍历所有子串
                            if(str.charAt(j) != str.charAt(j+k*i)){        //比较子串的第 j 个位置的字符是否相等
                                flag = 1;                                   //如果不等，则退出比较，当前子串长度 i 的划分不满足要求，结束，更新 i，进行下一次的搜索
                                break;
                            }
                        }
                        if(flag == 1) {
                            break;
                        }
                    }
                    if(flag == 0){
                        for(int j=0; j <i; j++){
                            System.out.print(str.charAt(j));
                        }
                        System.out.println();//true
                        return;
```

```
                    }
                }
            }
            System.out.println(false);//如果子串长度 i 的所有划分都不满足要求，则返回 flase，
表示该字符串不能由某个子串的若干倍组成

    }
}
```

3、小招喵跑步

【题目描述】小招喵喜欢在数轴上跑来跑去，假设它现在站在点 n 处，它只会 3 种走法，分别是：
1.数轴上向前走一步，即 n=n+1
2.数轴上向后走一步,即 n=n-1
3.数轴上使劲跳跃到当前点的两倍，即 n=2*n
现在小招喵在原点，即 n=0，它想去点 x 处，快帮小招喵算算最快的走法需要多少步？

输入描述:

小招喵想去的位置 x

输出描述:

小招喵最少需要的步数

输入样例：

3

输出样例：

3

【解题思路】

在点 n 处三种方式，根据 n 的奇偶性来进行讨论
dp[i]：
1.当 i 是偶数，dp[i/2]+1
2.当 i 是奇数,dp[(i+1)/2]+2,dp[i-1]+1
3.每个位置先进行初始化，然后就是上面两种情况下多取一下 Min

【答案及解析】

```
#include <iostream>
#include <vector>
using namespace std;
class Solution{
public:
        int solve(int n){
              if(n<=3)
                    return n;
              vector<int> dp(n+1,0);
```

```
            //先进行初始化
            dp[1]=1,dp[2]=2,dp[3]=3;
            for(int i=4;i<=n;++i){  //1，2,3 不用进行讨论，无论如何就那么几步
                  if(i%2==0)
                        dp[i]=min(dp[i-1]+1,dp[i/2]+1);
                  else
                        dp[i]=min(dp[(i+1)/2]+2,dp[i-1]+1);
            }
            return dp[n];
      }
};
int main(){
      int n;  //想去的位置
      cin>>n;
      if(n>=0 && n<3){
            cout<<n;
            return 0;
      }
      Solution sol;
      if(n<0)
            cout<<sol.solve(-n);
      else
            cout<<sol.solve(n);
      return 0;
}
```

二十六、4399 2019 秋招笔试真题

1、游戏开发

【题目描述】古典问题：有一对兔子，从出生后第 3 个月起每个月都生一对兔子，小兔子长到第三个月后每个月又生一对兔子，假如兔子都不死，问每个月的兔子总数为多少？

【参考代码】

参考答案为 java 代码，具体编码视情况而定

```
/**
 * 兔子问题
 * 斐波那契数列求值
 * 程序分析：  兔子的规律为数列 1,1,2,3,5,8,13,21....
 */
public class rabbit {
    public static final int MONTH = 15;
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        long f1 = 1L, f2 = 1L;
        long f;
        for(int i=3;i<MONTH;i++){
            f=f1+f2;
            f1=f2;
            f2=f;
            System.out.println("第"+i+"个月的兔子对数："+f2);
        }
    }
    //递归方法实现
    public static int fib(int month){
        if(month == 1 || month == 2){
            return 1;
        }else{
            return fib(month-1)+fib(month-2);
        }
    }
}
```

2、Web 开发

【题目描述】判断一个数是否为完美数。完美数的定义：它所有的真因子（即除了自身以外的约数）的和，恰好等于它本身。要求思路准确，性能更优得分更高。

【参考代码】

法 1：

```
int is_perfect(int n)
{
    int i,sum = 0;
    for(i=1; i<=n/2; i++) //此处判断遍历的数
    {
        if(n%i == 0)
        {
            sum=sum+i;
        }
    }
    if(sum==n)
    {
        return 1;
    }
```

```
    else
    {
        return 0;
    }
}
法 2：
int is_perfect(int n)
{
    int i,sum = 1;
    if (n == 1)
    {
        return 0;
    }
    for(i=2; i * i < n; i++)
    {
        if(n%i == 0)
        {
            sum = sum + i + (n/i);
        }
    }
    if(i * i == n)
    {
        sum = sum + i;
    }
    if(sum==n)
    {
        return n;
    }
    else
    {
        return 0;
    }
}
```

二十七、美团点评 2019 秋招笔试真题

1、图的遍历

【题目描述】给定一张包含 N 个点、N-1 条边的无向连通图，节点从 1 到 N 编号，每条边的长度均为 1。假设你从 1 号节点出发并打算遍历所有节点，那么总路程至少是多少？

输入描述：

第一行包含一个整数 N，1≤N≤10^5。

接下来 N-1 行，每行包含两个整数 X 和 Y，表示 X 号节点和 Y 号节点之间有一条边，1≤X，Y≤N。

输出描述:

输出总路程的最小值。

输入样例:

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

输出样例:

```
4
```

【解题思路】

这其实是一棵以 1 为根的树。

如果要回到起点，则每条边走了恰好 2 次。

不回到起点，则可以选择一条停在最深的叶子节点处。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 100000 + 5;
int n;
vector<int> G[N];
int dep[N];

void dfs(int u, int pre) {
    for(int v : G[u]) {
        if(v != pre) {
            dep[v] = dep[u] + 1;
            dfs(v, u);
        }
    }
}

int main() {
    scanf("%d", &n);
    for(int i = 1; i < n; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        G[u].push_back(v);
        G[v].push_back(u);
    }
    dep[1] = 0;
```

```
    dfs(1, 0);
    int ans = (n - 1)*2 - *max_element(dep+1, dep+n+1);
    printf("%d\n", ans);
    return 0;
}
```

2、最长全 1 串

【题目描述】给你一个 01 字符串，定义答案=该串中最长的连续 1 的长度，现在你有至多 K 次机会，每次机会可以将串中的某个 0 改成 1，现在问最大的可能答案

输入描述:

输入第一行两个整数 N,K，表示字符串长度和机会次数

第二行输入 N 个整数，表示该字符串的元素

(1 <= N <= 300000 , 0 <= K <= N)

输出描述:

输出一行表示答案

输入样例:

10 2

1 0 0 1 0 1 0 1 0 1

输出样例:

5

【解题思路】

two pointers

维护 0 的个数不超过 K 个的区间。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 300000 + 5;
int n, k;
int a[N];

int main() {
    scanf("%d%d", &n, &k);
    for(int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
    }
    int r = 0, cnt = 0;
    int ans = 0;
    for(int l = 1; l <= n; ++l) {
        while(r + 1 <= n && cnt + (a[r+1] == 0) <= k) {
```

```
            ++r;
            cnt += (a[r] == 0);
        }
        ans = max(ans, r-l+1);
        cnt -= (a[l] == 0);
    }
    printf("%d\n", ans);
    return 0;
}
```

3、外卖满减

【题目描述】你打开了美了么外卖，选择了一家店，你手里有一张满 X 元减 10 元的券，店里总共有 n 种菜，第 i 种菜一份需要 A_i 元，因为你不想吃太多份同一种菜，所以每种菜你最多只能点一份，现在问你最少需要选择多少元的商品才能使用这张券。

输入描述:

第一行两个正整数 n 和 X，分别表示菜品数量和券的最低使用价格。（1≤n≤100，1≤X≤10000） 接下来一行 n 个整数，第 i 个整数表示第 i 种菜品的价格。（1≤A_i≤100）

输出描述:

一个数，表示最少需要选择多少元的菜才能使用这张满 X 元减 10 元的券，保证有解。

输入样例:

```
5 20
18 19 17 6 7
```

输出样例:

```
23
```

【解题思路】

01 背包

找大于等于 X 的最小可达状态。

【参考代码】

```
#include <bits/stdc++.h>
using namespace std;
const int N = 10100 + 5;
int n, X, C;
int dp[N];

int main() {
    scanf("%d%d", &n, &X);
    C = X + 100;
    dp[0] = 1;
    for(int i = 0; i < n; ++i) {
```

```
        int a;
        scanf("%d", &a);
        for(int j = C; j >= a; --j) {
            dp[j] |= dp[j-a];
        }
    }
    for(int i = X; i <= C; ++i) {
        if(dp[i] == 1) {
            printf("%d\n", i);
            break;
        }
    }
    return 0;
}
```

二十八、网易互娱 2019 秋招笔试真题

1、会话列表

【题目描述】小云正在参与开发一个即时聊天工具，他负责其中的会话列表部分。
会话列表为显示为一个从上到下的多行控件，其中每一行表示一个会话，每一个会话都可以以一个唯一正整数 id 表示。
当用户在一个会话中发送或接收信息时，如果该会话已经在会话列表中，则会从原来的位置移到列表的最上方；如果没有在会话列表中，则在会话列表最上方插入该会话。
小云在现在要做的工作是测试，他会先把会话列表清空等待接收信息。当接收完大量来自不同会话的信息后，就输出当前的会话列表，以检查其中是否有 bug。

输入描述:

输入的第一行为一个正整数 T（T<=10），表示测试数据组数。
接下来有 T 组数据。每组数据的第一行为一个正整数 N（1<=N<=200），表示接收到信息的次数。第二行为 N 个正整数，按时间从先到后的顺序表示接收到信息的会话 id。会话 id 不大于 1000000000。

输出描述:

对于每一组数据，输出一行，按会话列表从上到下的顺序，输出会话 id，相邻的会话 id 以一个空格分隔。

输入样例：

```
3
5
1 2 3 4 5
6
1 100 1000 1000 100 1
7
1 6 3 3 1 8 1
```

输出样例:

5 4 3 2 1

1 100 1000

1 8 3 6

【参考代码】

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cassert>

using namespace std;

const int MAXN = 200 + 10;

int g[MAXN];

int main()
{
    int T, n;
    scanf("%d", &T);
    assert(T <= 10);
    while (T--)
    {
        scanf("%d", &n);
        assert(n >= 1 && n <= 200);

        memset(g, 0, sizeof(g));

        for (int i = 0; i < n; ++i)
        {
            scanf("%d", &g[i]);
            assert(g[i] > 0 && g[i] <= 1000000000);
        }

        bool first = true;

        for (int i = n - 1; i >= 0; --i)
        {
            bool ok = true;
            for (int j = i + 1; j < n; ++j)
            {
                if (g[j] == g[i])
                {
```

```
                    ok = false;
                    break;
                }
            }

            if (ok)
            {
                if (first) printf("%d", g[i]);
                else printf(" %d", g[i]);

                first = false;
            }
        }
        printf("\n");
    }
    return 0;
}
```

2、时钟

【题目描述】小 W 有一个电子时钟用于显示时间，显示的格式为 HH:MM:SS，HH，MM，SS 分别表示时，分，秒。其中时的范围为[‘00’，‘01’…‘23’]，分的范围为[‘00’，‘01’…‘59’]，秒的范围为[‘00’，‘01’…‘59’]。

但是有一天小 W 发现钟表似乎坏了，显示了一个不可能存在的时间“98:23:00”，小 W 希望改变最少的数字，使得电子时钟显示的时间为一个真实存在的时间，譬如“98:23:00”通过修改第一个’9’为’1’，即可成为一个真实存在的时间“18:23:00”。修改的方法可能有很多，小 W 想知道，在满足改变最少的数字的前提下，符合条件的字典序最小的时间是多少。其中字典序比较为用“HHMMSS”的 6 位字符串进行比较。

输入描述：

每个输入数据包含多个测试点。每个测试点后有一个空行。 第一行为测试点的个数 T(T<=100)。 每个测试点包含 1 行，为一个字符串”HH:MM:SS”，表示钟表显示的时间。

输出描述：

对于每个测试点，输出一行。如果钟表显示的时间为真实存在的时间，则不做改动输出该时间，否则输出一个新的”HH:MM:SS”，表示修改最少的数字情况下，字典序最小的真实存在的时间。

输入样例：

2

19:90:23
23:59:59

输出样例：

19:00:23
23:59:59

【参考代码】

```
#include <iostream>
#include <cstdio>

int main()
{
    int T;
    scanf("%d", &T);
    while (T--)
    {
        int hh, mm, ss;
        scanf("%d:%d:%d", &hh, &mm, &ss);
        if (hh > 23) hh = hh % 10;
        if (mm > 59) mm = mm % 10;
        if (ss > 59) ss = ss % 10;
        printf("%02d:%02d:%02d\n", hh, mm, ss);
    }
    return 0;
}
```

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

据说跟着学习的都轻松去了BAT，TMD…

大佬手把手教学

前亚马逊高级算法专家直播授课
已帮助5000+学员成功入职百度、腾讯等互联网名企

招牌题完全解读

全新校招面试考题完全解读，讲解解题思路并提供最优解代码

金牌助教坐阵答疑

ACM金牌助教一对一答疑解惑，扫除问题，校招冲刺快速提升

专属优惠券：nianxin30W

立减235元，报名即免费赠送左程云
《程序员代码面试指南》纸质第二版

报名咨询：niukewang985

（微信，加时请备注来意）

扫码报名