2020-2021届

牛客独家春招实习

备战手册

技术篇

牛客出品

nowcoder.com

目录

一、春招备战概述.....1
二、春招信息渠道.....2
三、技术岗位考前须知.....8
四、春招备战攻略.....10
4.1 简历攻略.....10
4.2 笔试攻略.....16
4.3 面试攻略.....23
五、名企经典笔试题目.....33
5.1 腾讯 2019-2020 校园招聘真题.....33
5.2 字节跳动 2019 校园招聘真题.....39
5.3 百度 2019 校园招聘真题.....46
5.4 美团点评 2019 校园招聘真题.....48
5.5 爱奇艺 2020 校园招聘真题.....52
5.6 京东 2019 校园招聘真题.....58
5.7 网易 2020 校园招聘真题.....65
5.8 网易游戏（互娱）2020 校园招聘研发真题.....74
5.9 拼多多 2019 校园招聘真题.....79
5.10 vivo2020 校园招聘真题.....85
5.11 360 公司校园招聘真题.....91
5.12 深信服 2019 校园招聘真题.....98

一、春招备战概述

1.1 春招介绍

春招就是春季校园招聘，是面向应届生和准应届生进行的，包括了应届生校招和实习生招聘。

有些同学认为，找工作是毕业前的事情，大四再去实习也不迟。这个想法是错误，近年来，互联网企业选人要求越来越严格，应试者的能力水平也在不断提高，绝大部分的同学，在大四求职时，已经有至少一份实习经历。倘若没有实习经历，想在校招中斩获满意的 offer 就难上加难。

所以大三的同学，请一定要抓住 2020 年春招机会，找一份能够给你添彩的实习！

1.2 春招时间

春招开始时间通常是在春节之后。2020 年的春节假期是 1 月 25 日-2.3 日，按照正常情况，今年春招会在 2 月中展开。

但由于今年特殊情况，有部分公司会延迟春招开始时间（部分企业将春招时间延迟到 2 月下旬，而有的企业目前还没有确定时间），而且大部分企业会选择将整个流程转到线上，像往年的线下招聘渠道，例如校园宣讲会，线下双选会都会改为线上进行，整体招聘节奏变快。部分企业甚至会缩减招聘名额。

但由于应届生拿到毕业证的时间不变（集中在 6-7 月），因此留给大家的春招时间同比往年会缩短，算下来大概不过短短三个月。因此建议大家好好利用目前在家的时间，为春招做好充足的准备。

1.3 春招面向对像

春招是面向应届生的校园招聘，以及面向准应届生的实习生招聘

互联网行业面向应届生（即今年 2020 届毕业生）的春招，也为秋招的补招，对应届生来说是正式校园招聘。具体流程：**应聘>>公司发正式 offer>>学生接 offer>>毕业领毕业证>>正式入职、签订劳动合同**。

互联网行业面向准应届生（即今年 2021 届）的春招，则是实习生招聘，分为日常实习和暑期实习。

春招=准应届生实习招聘+应届生校园招聘

1） 春季校园招聘

• 成规模的校园招聘

• 零散的校园招聘补招

2) 春季实习招聘

• 一般以准应届生（大三或者研究生）为主，大厂招聘为多，实习时间一般在暑期（但鉴于今年的特殊情况，暑期可能会缩短，实习期可能会涉及正常授课时间）。

（22 届、23 届的同学如果想要找实习，也可以在春招的时候去试试，如果足够优秀，部分企业也会接纳年级更小的同学进入企业进行实习。）

二、春招信息渠道

2.1 公司官网+公司官方招聘公众号

正常的校园招聘信息获取途径最准确的是公司招聘官网或官微，但是招聘的公司这么多，一个个翻找会浪费很多时间和精力，而且还容易遗漏或者错过简历投递时间。

其次，除大厂的暑期实习生/培训生项目外，其余很多企业的实习生招募是小规模的，比较零散，时间不集中，这类招聘不会发布在官网/官方招聘公众号上。一般由用人部门员工或实习生直接招募，他们会在朋友圈/各大论坛/各大招聘网站发布招聘信息，但如果你没有相应认识的人，可能在你还未知道实习消息之前，实习生招募就已经悄然结束了。**这时候，认准牛客网就可以了。**

2.2 牛客网

牛客网除了拥有丰富的公司真题题库、面经外，还有许多实习招聘信息，涵盖大厂小厂实习生岗位招聘信息，利用好牛客，会为的你春招求职之路提供助力。

2.2.1 春招备战资源类

- **公司真题**

涵盖腾讯、百度、字节跳动等 90+企业历年校招真题汇总，支持在线练习。不需要再去全网搜索甚至付费购买真题资源。

网站地址:**牛客网>题库>公司真题**

https://www.nowcoder.com/contestRoom?from=CZSX2021

移动端地址:**牛客 app>学习>公司套题**，把题目装口袋，随时随地刷题。

- **专项练习**

计算机、通信、产品运营、行测题库分门别类，根据应聘岗位有针对性地刷题，笔试考点逐个击破。

网站地址：**牛客网>题库>专项练习**

https://www.nowcoder.com/intelligentTest?from=CZSX2021

移动端地址：**牛客 app>首页>专项练习**

- **面经汇总**

汇总上百家名企优质的前辈面试经验，通过阅读大量面经，熟悉面试流程，摸透面试套路，有针对性地进行准备。更有分岗位的面试宝典，助大家系统性地复习应聘岗位高频考点。

网站地址：**牛客网>面试>面经汇总**

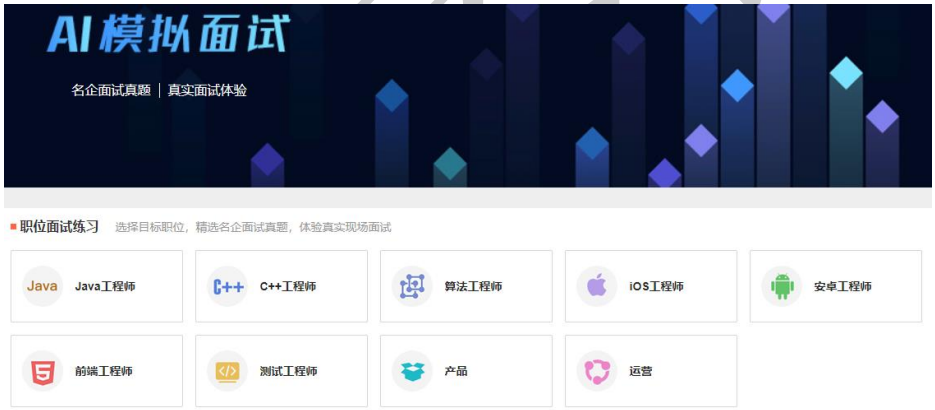
https://www.nowcoder.com/interview/center?from=CZSX2021

- AI 面试

精选百家名企历年面试真题，模拟现场真实面试。目前，越来越多的企业为了提高招聘效率，都会采用线上面试的模式，加上今年的特殊情况，绝大部分的公司都会选择将面试改为线上，因此建议大家在面试前进行 AI 面试练习，提前熟悉真实面试流程，有利于更沉着冷静地面对正式面试，从而大大提高面试成功机率。 j

网站地址：**牛客网>面试>AI 面试**

https://www.nowcoder.com/interview/ai/index?from=CZSX2021



- 专栏+文章

领域大佬分享求职攻略，帮助大家利用碎片化的时间系统化地进行复习，提高复习效率。

目前上线的专栏有：

前阿里产品大佬分享的产品求职之道 30 讲

小米工程师分享的 Java 求职面试题全解析

左程云的程序员代码面试指南

清华博士的 python 金融数据分析与量化投资

百度工程师的 c++开发法求职攻略

0 基础入行-游戏策划校招必备知识

网站地址：**牛客网>学习>专栏+文章**

https://www.nowcoder.com/blog/blogCenter?type=zhuanlan?from=CZSX2021

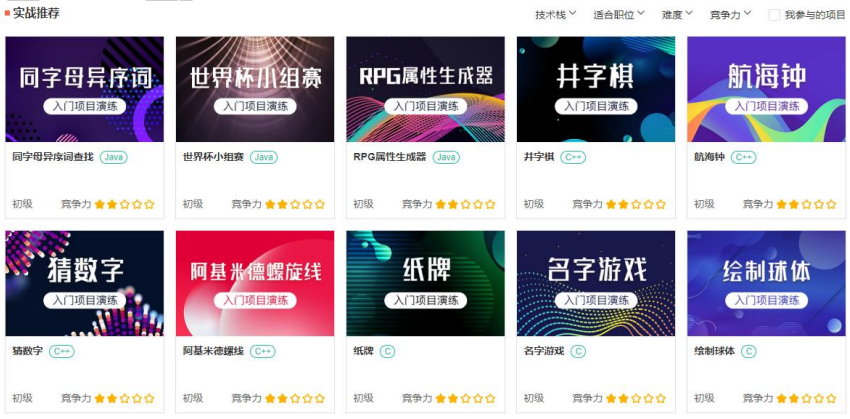
移动端地址：**牛客 app>学习>专栏+文章**

- **项目实战**

涵盖初级到高级，不同难度，不同岗位的项目案例。每一个项目从配置环境到运行实现，一步步详细讲解，帮助大家积累更多的实战经验，提高竞争力，斩获心仪 offer。

网站地址：**牛客网>学习>项目实战**

https://www.nowcoder.com/project/recommend?from=CZSX2021



- **牛客职播**

各家名企校招 HR 作客牛客，为大家揭秘校招，答疑解惑。了解企业招聘计划、招聘要求，做到知己知彼，才能百战百胜。

网站地址：**牛客网>学习>课程>牛客职播**

https://www.nowcoder.com/study/live/120?from=CZSX2021

移动端地址：**牛客 app>学习>视频课程>牛客职播**

- **课程**

名师授课，理论与实践结合，详细讲解校招笔试真题，结合校招的实际案例，剖析面试技巧，带你轻松通关，斩获高薪 offer。

网站地址：**牛客网>学习>课程**

https://www.nowcoder.com/courses?from=CZSX2021

移动端地址：**牛客 app>学习>视频课程>更多好课**

2.2.2 春招招聘资讯类

- **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?from=CZSX2021

移动端地址:**牛客 APP>求职**

- **Offer 神器**

基于牛客网真实 Offer 大数据的预测神器。根据求职意向，帮大家发现未来可能拿到的 Offer，分析手头 Offer 的竞争力，了解校招薪资情况，为大家地 Offer 选择提供参考。

网站地址：**牛客网>发现>offer 神器**

https://www.nowcoder.com/show-offer/index?from=CZSX2021

移动端地址：**牛客 APP>求职>求职工具>offer 神器**

- **讨论区**

数百万已拿到 offer 的老牛友以及 HR 发布各企业各岗位地内推消息，消息来源真实可靠，助力大家快人一步拿到 offer。

网站地址：**牛客网>讨论区>招聘信息**

https://www.nowcoder.com/discuss?type=7&order=0?from=CZSX2021

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**

一站获得所有公司最近招聘动态，不再遗漏任何一家心仪公司的简历投递时间。

网站地址：**牛客网>求职>校招日程**

https://www.nowcoder.com/school/schedule?from=CZSX2021

移动端地址：**牛客 APP>求职>求职工具>校招日程**

小程序地址：扫描右方二维码即可观看

（小程序将于 3 月中旬上线，大家敬请关注）

小程序订阅企业校招进度，第一时间获取最新校招信息。

- **公众号**

招聘日程汇总：每日推送校招/内推资讯以及面经干货。
牛客招聘助手：绑定手机号，可及时查询自己的笔试面试进度和信息。
牛可乐、牛能：已建立春招招备战群，每日分享校招信息、解答疑问。 扫码加牛可乐、牛能好友，回复春招即可进群。

招聘日程汇总

牛客招聘助手

牛能&牛可乐

- **牛客资料大全**

互联网求职简历制作、优质面经、名企校招笔试、面试真题等资料大汇总。
小程序地址：扫描右方二维码即可领取

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

三、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！

3.1 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：**牛客招聘助手>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?from=CZSX2021

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?from=CZSX2021

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

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

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

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

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

第四步：笔试做题流程

1、试卷中会有一种以上个题型，进入考试后**请仔细查看共有几个题型**。

2、可选择任意题型进入做题，**所有题型一旦提交后将无法返回修改**。

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

4、如遇**突发情况**，如断网、电脑死机、断电等，请直接刷新页面，或关闭浏览器后重新通过考试地址进入。题目会自动保存，所以不用担心。

5、考试环境体验：

https://www.nowcoder.com/cts/3942933/summary#?from=CZSX2021

四、春招备战攻略

大二、大三的同学，是否计划好找哪些公司的哪些岗位？是否开始制作简历了？总之，都要抓紧时间准备了，不然等秋招或者毕业了还找不到满意的工作的时候，着急也晚了。

2020 届的同学，秋招是否已有 offer，没有 offer 的同学要分析下原因，是败在了简历这一关，还是笔试这一关，还是面试这一关。冷静分析和总结并对症下药才是最好的解决办法。

4.1 简历攻略

相信大家对简历准备都有一些困惑，不知道该写什么不该写什么，怎么写才能更受企业青睐，下面帮大来家总结一下。

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**。

4.1.1 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。关于简历照片，建议大家放一张干净大方精神的证件照片，不要随便放一张自己的生活照或者过分美颜的照片，正式大方的照片可以增 HR 对你简历的印象。

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，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，学校对找工作肯定有着或多或少的影响。但这也并不是能对你一锤定音的指标，在毕业院校不是特别好的情况下，请一定在简历上充分表现

你的实习经历+项目经历！这一点在接下来介绍实习经历&项目经历的时候，会再给大家详细介绍！

加分项：个人博客 & Github

如果有非常优秀的博客，或者在 Github 有贡献开源项目，可以写到简历上，对自己绝对是一个加分项。如果暂时没有，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。

但如果到到了要投递简历的时候，仍然没有出色的加分项，就不要硬往上写，因为如果你的个人博客&Github 并不出色，HR 会认为你在浪费他的时间，结果可能会适得其反

4.1.2 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，绝对是校招时扣响名企大门的一块最有力的敲门砖。

因为大厂的实习经历代表着你曾被大厂认可过，且具有相应的技能和一定的工作经验，用人单位会很欢迎这样的学生！有实习经历后，要怎么通过简历表现出来也是一门学问，以下有几点大家需要注意：

（1）实习经历不是写得越多越好

把与你要应聘的职位匹配度高且含金量高的实习经历写上去即可，例如在奶茶店打过暑期工的这些经历不需要往上写；又例如你要应聘的是后端工程师，但实习经历写的是某公司运营实习，这样非但没有展现出你的优势，还会让面试官更加慎重地考虑你是否适合后端工程师这个岗位。

（2）无需交代细枝末节的工作任务

之所以实习经历重要，是因为面试官希望通过实习经历，了解你的专业能力、工作能力，以便他们更好地评估你是否适合他们公司。因此在描述实习经时，只需要把你主要负责的、能够体现你专业能力的工作清晰地描述出来即可（负责什么工作？有没有遇到什么困难？通过什么方法解决？有什么成果？）。无需把一些细枝末节的工作内容往上写。

（3）分点描述工作内容，切勿流水账

有的同学在写简历的时候，会有一个毛病，就是容易写成流水账，长篇大论，字数非常多，且不会分段。试想一下，如果你是面试官，看到简历上密密麻麻的字，找不到重点，你还会有心情认真去看这份简历吗？所

以，建议大家养成分点描述的好习惯，将自己要表达的内容好好梳理，分 1、2、3 点进行简单明了的描述，提高简历的可读性。

如果暂时还没有实习经历的同学，也不用太过担心，因为咱们毕竟还只是去应聘实习生，况且还有项目经历可以拉你一把！

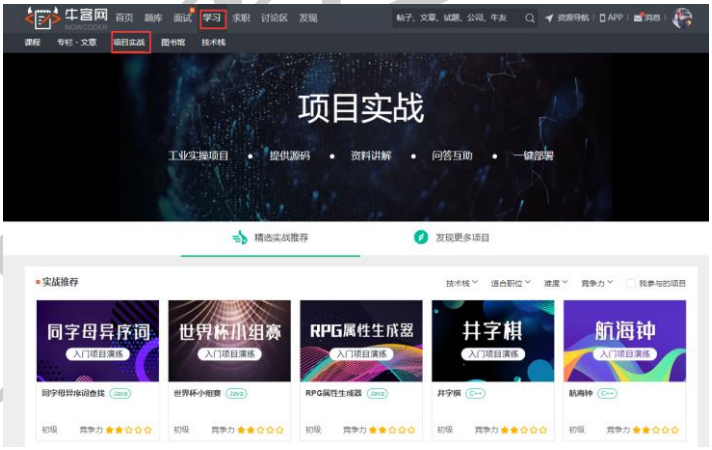
4.1.3 项目经历

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?from=CZSX2021



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?from=CZSX2021

相信很多同学都想知道如何在简历上通过项目经历充分展示自己的能力，那项目经历应该如何写，有什么注意事项？我们总结了以下几点：

（1）项目经历不要只写产品功能，重点要写自己使用的技术

如果使用了算法，就要讲用了 xx 算法后，与其他 xx 算法比较有什么提高，达到了什么目标。

例如：我通过 xx 算法，解决了 xxx 任务中 xxx 问题，xxx 结果数据显示有 xxx 的提高/改进。如果没有用什么牛的算法就总结写通过 xx 组件，xx 开源库实现了 yy 功能，最终熟悉 zz 的 sdk、框架、开源库等。啥都没学会那就写自己写了 xxxxx 行代码，熟悉了设计模式，标准类库之类的。

例子：

有个同学写了下面的项目经历：

2015.03 - 2015.04iunClub 新闻站 PHP 程序员

1.构建服务器端，爬取各大网站的 RSS 数据并存储数据库。

2.构建客户端，解析数据库返回的数据。页面设计由其他成员负责

修改后：

1、通过 Python 的 urllib2 和 HTMLParser 实现爬虫的基本功能，对爬取页面解析后的结果通过敏感哈希算法来去重，最终通过 pymysql 把新闻内容存储到数据库。

2、通过采用多线程，多代理模式手段优化系统提高爬取速度，最终达到单台服务器每天爬取有效的 xxx 个页面。

（2）学会用实际数据体现自己的成果

数据是最能直观体现项目成果的方式，如果一段经历，没有任何数字，要么说明这个工作只是执行或完成了，没有关注结果；要么说明对数字不敏感，缺乏优化和思考的过程。这两者在求职过程中都是硬伤。

举个简单直观的例子

1、我通过 xx 算法，解决了 xxx 任务中 xxx 问题，结果数据显示较大的提高/改进

2、我通过 xx 算法，解决了 xxx 任务中 xxx 问题，xxx 结果数据显示有 xxx 的提高/改进

很显然，第二个描述相比第一个能让别人更直观地了解优化的效果，同时也会增加这段经历的可信度。

看到这里有同学会问，我做项目的时候没有统计过这些数据，那我能不能随便编一组数据？在这里建议大家不要抱着侥幸的心理，数字与数字之间是可以交叉验证的，面试官比你想象的更加聪明细致，要编一组不被

面试官察觉的数据可没那么容易，还是老老实实复盘一下项目经历，把需要展示的数据都计算一遍。

（3）不需要长篇大论地介绍技术原理

有的同学生怕面试官不清楚这个技术、在描述项目经历的时候会花费大量篇幅介绍技术的原理，其实大可不必。且不说面试官非常有可能了解你写的技术，即使不了解，他也不会有兴趣在面试的时候看你写的原理介绍，他不是来上课的，是来招聘人才的。因此面试官关注的重点不是这是一项什么技术，而是你怎么攻克难点实现这个技术，通过这项技术实现了什么。

4.1.4 校园经历

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

可以写一些你的校园经历，尽量写与应聘行业、岗位相关的一些校园经历。比如你要应聘互联网技术类岗位，你可以写在学生会技术部、技术相关社团中担任过什么要职，举办过哪些活动，参加过什么技术相关的比赛，获得什么名次等等。

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。不要好像要写大学 4 年回忆录似的，把自己大大小小的经历都往上堆，尽量挑自己挑大梁的且与应聘行业、岗位沾上边的活动写在简历上。那些与应聘行业、岗位毫无干系的校园经历最好不要写。

另外在这一部分可以列举一些你获得的奖学金。

4.1.5 掌握技能

应聘哪个岗位，一定要具备其相关技能。你可以通过简历告诉面试官你都掌握了哪些相关的技术。 注意“了解”、“基本掌握”、“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。虽说技能掌握这块看似很快就能写完，但是这部分的坑可不少，大家要注意以下几点：

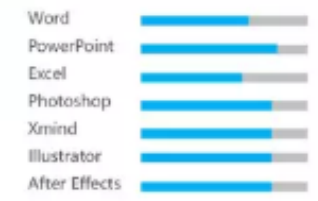
（1）把握技术名词列举的数量

应聘技术岗的同学，在描述掌握技能的时候往往习惯罗列多一些技术名词，想着技术名称越多，越能体现自己的知识储备。其实并非如此，面试官

清楚地知道大学 4 年，或者大学+研究生 7 年，一般的同学所能掌握技术的广度和深度，过多地罗列技术名字，除了有夸大之嫌外，还会让面试官认为，你对技术的认知比较片面，基本处于了解阶段，没有 1-2 个熟练掌握甚至精通的技术。其实技术的学习贵精不贵广，精通 1-2 个技术比了解几十个技术来得更难能可贵。因此当你发现自己在罗列技术名词的时候，不妨给自己掌握的这些技术排个序，优先罗列自己真正从原理到应用都清晰掌握的技术。

（2）描述要清晰明了，不要使用过于花哨、含糊不清的表现方式

有的同学为了追求样式美观，会采用图形的形式去表达自己掌握技能的程度，进度条就是最常见的了。这样一来，样式确实很美观，但是面试官看了之后会不明所以，无法通过进度条进行量化，清晰明了地了解你各项技能的掌握程度，下面例子，我相信大家看了，也无法判断这位同学各项技能的掌握程度。



所以，尽量用文字表达的方式，清楚地描述你掌握技能的程度，不要让面试官有所疑惑。

4.1.6 其他注意事项

（1）简历排版整齐有序、主次突出（该加粗的字体加粗），没有过多细节问题

- 英文部分注意大小写，比如 GitHub，而非 github，O2O 而非 O2O
- 中文始终保持一种字体，英文始终保持一种字体
- 简历模板简单不花哨，慎用图标、进度条等附加元素
- 简历字体不要使用过多的颜色，1-3 种即可
- 简历内容安排合理，描述调理清晰，分点论述，标点符号使用正确
- 谨慎使用了解、熟悉、精通等词汇

（2）去除和目标职位无效的内容

比如个人兴趣爱好、自我评价、年龄、生日、住址等这些无关的信息

（3）尽量压缩简历到一页

如果超过一页，可以根据各个模块的重要性优先级进行删减，各模块重要性优先级如下：技术类实习>完整项目经历>技术比赛获奖>自娱性小工

具项目经历>个人技能和博客>学校内获奖>社团工作>班长支书>非技术实习>自我评价>兴趣爱好。

（4）不断通过修改简历去匹配不同企业的不同岗位

简历的制作过程并不是一蹴而就的，因为每个公司、每个职位的招聘要求都有所不同，每次投递简历前，都要通过修改简历去更好地匹配不同企业的不同职位，这样才能增大简历通过的概率。面试后也要针对简历不足的地方进行修改。

（5）不要造假

简历制作切记：可以美化，但是不可以撒谎！有些同学实习周期 1 个月，洋洋洒洒写了一大段产品功能，让人觉得根本不是你做的项目

（6）使用 pdf 投递简历，不要使用 word

如果对于简历制作还是没有什么头绪的话，推荐观看视频：《技术类应届毕业生简历制作与面试技巧》（点击观看）

4.2 笔试攻略

4.2.1 总攻略

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！**

什么叫有意义的刷题，是指要制定自己的刷题计划（下文推荐刷题专场）：

1、**基础知识巩固：**网络基础+操作系统+Linux+数据库+编程语言基础

基础知识往往是最容易被忽略的，总认为自己都会了就不再去练习。如果基础不牢固，你的笔试很可能就败在这一道简单题上了。

2、**数据结构和算法**

学校的课程中一般都会设置数据结构，但是学校所讲的仅仅是基础中的基础，不足以让你应对严格的笔试。所以千万不可满足于学校所授，自己也要自主去学习新的知识。

- **重要提示**

算法是不管笔试面试中都占重要角色的，如果你的算法优秀，校招就已经成功一半了，笔试面试能过的概率大幅增加。 **算法的学习主要通过：**

看书+刷题+跟大佬学习。至于怎么学好算法，推荐看看算法大佬左程云亲笔写的：程序员该如何学习算法（点击观看原文）

推荐牛客网左程云老师的算法直播课：

- 《牛客算法基础入门班（点击查看详情）》

适合人群：0 基础小白入门同学，例如大一大二，或者大三基础非常薄弱和跨专业求职的同学。

- 《牛客算法基础提升班（点击查看详情）》

适合人群：适合有简单算法入门基础知识，但是基础还不够牢固的同学，帮助大家填补算法入门和求职笔试面试实战之间的空白，知识储备更接近求职考察实战。

当你的算法不再是学校《数据结构》的水平时，你就知道自己多有竞争力了

3、**在线编程练习：**剑指 offer+Leetcode/校招编程真题汇总

把这些题目认认真真刷几遍，这是每位前辈都会说的建议。因为编程题在笔试中所占比重很大，可谓得编程者得天下。

- 《剑指 offer（点击开始刷题）》
- 《Leetcode 经典编程题（点击开始刷题）》
- 校招编程真题汇总
  《2019 校招真题编程汇总（点击开始刷题）》
  《2018 校招真题编程汇总（点击开始刷题）》
  《2017 校招真题编程汇总（点击开始刷题）》
  《2016 校招真题编程汇总（点击开始刷题）》

4、**考前特训：**组合数学+概率统计+智力题

笔试一般会考几道智力题，大家在考前练习练习即可。可以到**牛客网>题库>知识点专项练习>行测** 中进行练习

网址：https://www.nowcoder.com/intelligentTest?from=CZXS2021

5、公司真题练习

真题的价值不用多说，就是完完全全的划重点，大家可以通过企业历年的笔试真题，找准企业重点考察点，有针对性地进行查漏补缺。且历年

也不乏有同学会在笔试的时候发现类似以前校招笔试的题目，所以考前进行真题练习，是很有必要的。大家可以到**牛客>题库>公司真题**进行练习。
网址： https://www.nowcoder.com/contestRoom?from=CZXS2021

6、通过笔试结果，了解自己知识掌握的情况

虽然每次笔试成绩都不会公开，但可以先看看自己是否有笔试通过的公司，所占比例为多少。

• 如果超过了 50%，说明你的笔试成绩还算凑合的，每天留出一些时间巩固、提升即可，学习计划要针对自己的情况去制定。

• 如果低于 50%，说明你的基础还是不够扎实，需要抓紧每天强化基础部分和提升部分，不要以为准备笔试是浪费时间，准备笔试的同时也是在提升自己的能力，面试这关也更好过一些，合理安排好时间和计划才是最重要的。

笔试环节是进入面试环节的门槛，下面将给大家详细列举如何练习。

4.2.2 笔试训练专场

下面将根据不同技术岗位给大家推荐相应的训练专场，请根据自己的情况加入备战计划。

- **考前备战计划**

C++专场（点击进入）

1、C/C++基础知识常考点：
https://www.nowcoder.com/ta/review-c?from=CZXS2021
（点击题目查看答案）

（1）写一个函数返回 1+2+3+…+n 的值（假定结果不会超过长整型变量的范围）

（2）请写一个 C 函数，若处理器是 Big_endian 的，则返回 0；若是 Little_endian 的，则返回 1

（3）说一下 static 关键字的作用

……

Java 专场（点击进入）

1、Java 基础知识常考点：
https://www.nowcoder.com/ta/review-java?from=CZSX2021

（点击题目查看答案）

（1）什么是 Java 虚拟机？为什么 Java 被称作是“平台无关的编程语言”？

（2）JDK 和 JRE 的区别是什么？

（3）“static”关键字是什么意思？Java 中是否可以覆盖(override)一个 private 或者是 static 的方法？

……

前端专场（点击进入）

1、JavaScript 基础知识常考点：

www.nowcoder.com/ta/front-end-interview?from=CZSX2021

（点击题目查看答案）

（1）请你谈谈 Cookie 的弊端

（2）web storage 和 cookie 的区别

（3）CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？CSS3 新增伪类有那些？

2、HTML/CSS 基础知识常考点：

www.nowcoder.com/ta/review-frontend?from=CZSX2021

（点击题目查看答案）

（1）HTML5 的优点与缺点？

（2）Doctype 作用？严格模式与混杂模式如何区分？它们有何意义？

（3）HTML5 有哪些新特性、移除了哪些元素？

……

算法专场（点击进入）

1、算法基础知识常考点：

https://www.nowcoder.com/intelligentTest?from=CZSX2021

2、数据结构和算法

www.nowcoder.com/review?from=CZXS2021

3、算法/机器学习基础知识常考点

https://www.nowcoder.com/ta/review-ml?from=CZSX2021

测试专场（点击进入）

1、软件测试基础知识常考点：

https://www.nowcoder.com/ta/review-test?from=CZSX2021

- 运维专场（点击进入）

- **网络基础常考点：**https://www.nowcoder.com/ta/review-network?from=CZXS2021

（点击题目查看答案）

（1）了解交换机、路由器、网关的概念，并知道各自的用途

（2）TCP 的三次握手过程？为什么会采用三次握手，若采用二次握手可以吗？

（3）DNS 域名系统，简单描述其工作原理。

……

- **操作系统练习专场：**www.nowcoder.com/review?from=CZXS2021

（1）String str= new String(“abc”)，“ abc”在内存中是怎么分配的？

（2）轮询任务调度与抢占式任务调度的区别？

（3）有一个虚拟存储系统，若进程在内存中占 3 页（开始时内存为空），若采 用先进先出（FIFO）页面淘汰算法，当执行如下访问页号序列后 1，2，3，4，5，1，2，5，1，2，3，4，5，会发生多少缺页？

……

- **Linux 专项练习：**www.nowcoder.com/intelligentTest?from=CZXS2021

（点击题目查看答案）

（1）在 linux 下，如何查看物理内存的大小？

（2）在重新启动 Linux 系统的同时把内存中的信息写入硬盘，应使用（）命令实现

（3）不算 main 这个进程自身，到底创建了多少个进程？

```
int main(int argc, char* argv[])
{
    fork();
    fork() && fork() || fork();
    fork();
```

```
}
……
```

- **数据库专项练习**：www.nowcoder.com/intelligentTest?from=CZXS2021
- **在线编程练习**

（1）剑指 offer：www.nowcoder.com/ta/coding-interviews?from=CZXS2021

（2）Leetcode： www.nowcoder.com/ta/leetcode?from=CZXS2021

- **公司真题练习**：www.nowcoder.com/contestRoom?from=CZXS2021

（1）2019 校招真题编程题汇总：https://www.nowcoder.com/ta/2019test

（2）2018 校招真题编程题汇总:https://www.nowcoder.com/ta/2018test

（3）2017 校招真题编程题汇总:https://www.nowcoder.com/ta/2017test

（4）2016 校招真题编程题汇总:https://www.nowcoder.com/ta/2016test

（5）字节跳动：https://www.nowcoder.com/contestRoom?mutiTagIds=665

（6）百度: https://www.nowcoder.com/contestRoom?mutiTagIds=139

（7）腾讯: https://www.nowcoder.com/contestRoom?mutiTagIds=138

（8）京东: https://www.nowcoder.com/contestRoom?mutiTagIds=151

（9）美团: https://www.nowcoder.com/contestRoom?mutiTagIds=179

（10）拼多多: https://www.nowcoder.com/contestRoom?mutiTagIds=732

（11）阿里巴 https://www.nowcoder.com/contestRoom?mutiTagIds=134

更多公司真题>>>>>>>

4.2.3 笔试必知技巧

- **循环输入输出处理常见问题**

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

2、处理方法：其实这个问题可以避免，就是编程题后台每个样例做一组对应的.in 和.out 文件，这样就变成单组测试，代码就不需要循环处理，但是平时练习的题目质量不一，这个问题都会出现。代码里面循环处理了即使是单组测试也会完全没问题，所以为了偷懒，可以全写成循环处理。

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

4、第一次做在线编程题的同学可能对输入输出不熟悉。只要了解是从控制台输入输出的，就很简单了，大家可以到牛客多练习一下输入输出，熟悉输入输出后，可以加强真题的练习，熟练掌握在线编程要领。
输入输出练习网址：
https://www.nowcoder.com/discuss/216684?from=CZXS2021
在线编程练习题库：
https://www.nowcoder.com/activity/oj?from=CZXS2021

- **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"**

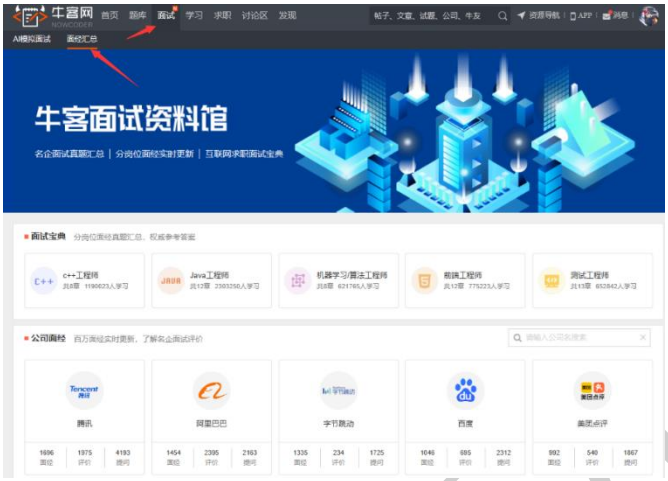
这个问题很蠢！通不过就是有一些问题。一个是要累积经验，分析到底可能出现的问题在哪里。另外不要使用一些奇怪的函数和行为。之前有见过有人使用了 windows 和 linux 平台那个功能的函数名都不一样的奇葩函数。如果你使用 C/C++，最好别使用 VS 来写算法 code，这个默认是 MS 的，一般 OJ 上面编译器都不会是这个鬼。

4.3 面试攻略

4.3.1 总攻略

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。下文会给大家详细列举精选的面经。

大家可以在**牛客网>面试>面经汇总**中有针对性地寻找意向公司的面经，以可以在**讨论区**搜索特定岗位的相关面经。



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

扫一扫，面经装进口袋

看一看，关注行内新态

有的同学会感觉，单纯只是看面经，心理不踏实，因为知识点都很分散，没办法系统地对各知识点、注意事项进行梳理。如果大家也有类似的感受，建议可以通过专栏+面经的交叉学习方式。

牛客专栏是牛客网精心推出的一款面向各个行业和岗位的求职、学习的知识服务产品，每个专栏拥有独立的主题，内容皆由各个主题领域大佬分享，旨在帮助大家通过短、精、全的好内容碎片化时间高效学习。大家可以通过专栏学习系统地梳理知识点，再通过阅读面经的方式，吸取更多的面试经验。

目前已经上线的以下技术专栏，大家可以根据自己的意向岗位进行选择。

1、《C++开发求职攻略》
https://blog.nowcoder.net/zhuanlan/wjvzj1?from=CZXS2021

2、《Java 开发岗高频面试题全解析》
https://blog.nowcoder.net/zhuanlan/YOxvjy?from=CZXS2021

3、《Python 金融数据分析与量化投资》
https://blog.nowcoder.net/zhuanlan/9MJvMQ?from=CZXS2021

4、《校招产品求职之道 30 讲》
https://blog.nowcoder.net/zhuanlan/qmGDjv?from=CZXS2021

5、《程序员代码面试指南》
https://blog.nowcoder.net/zhuanlan/JOeZmp?from=CZXS2021

6、《0 基础入行-游戏策划校招必备知识》
https://blog.nowcoder.net/zhuanlan/LMVLm7?from=CZXS2021

面试考点：基础知识+算法+项目

1、基础知识：操作系统+数据库+数据结构

不论什么岗位，以上三项是必考点。有的同学可能觉得自己笔试都没问题，面试也无需复习，但是面试和笔试是不一样的，你要能够很流畅精确地表达出来。建议看书学习，并配以做题辅助，实践才能出真知。

操作系统+数据结构题库：www.nowcoder.com/review?from=CZXS2021
数据库 SQL 实战：https://www.nowcoder.com/ta/sql?from=CZXS2021

2、算法

面试官最常考的就是让面试者现场手写算法，很多同学会因为紧张而思路卡壳儿。除了心态问题，其他的是真的需要好好写代码才能掌握，不是说看了理解了就真正会的了。算法面试的特点就是没有特点，什么样的题都可能遇到，因为根本没有考纲，面试官就是普通的程序员，他们在工作中或者在网络上遇到什么题不错，就可能考，所以内容真的太多了，而且也无穷尽。这不是一个标准考试，这是能力考试。

推荐大家去看

• 《程序员代码面试指南（点击查看）》，该书作者就是左程云老师（曾就职于亚马逊，担任技术专家，IBM，GrowingIo，百度等，从 2010 年起专注刷题至今，拥有 10 年算法刷题经验）

《程序员代码面试指南》独家在线 OJ：

https://www.nowcoder.com/ta/programmer-code-interview-guide?from=CZXS2021

• 《程序员面试金典》，该书是原谷歌资深面试官的经验之作，层层紧扣程序员面试的每一个环节，全面而详尽地介绍了程序员应当如何应对面试，才能在面试中脱颖而出。此书还有配套练习，建议大家阅读完每一章后都认真地完成课后练习，以此检测自己的学习效果。先进行在线练习的同学，可以到牛客《程序员面试金典》编程题在线练习（点击查看）中进行练习。

为了帮助准备校招的同学更好的学习算法，牛客特意邀请左神分享自己的算法学习经验，从算法的重要性，包括求职和工作两方面、应该如何学习算法，并配合求职中的真实案例进行讲解，给出学习算法的建议和参考。帮助大家更好的规划自己的校招求职之路。 如何学习算法（十年算法刷题大牛分享算法学习经验）

另外左程云老师近期正在直播“技术岗求职面试必考算法课”，这是快速提升自己算法的绝好时机。

大家可根据自身的情况进行选择：

• 《牛客算法基础入门班（点击查看详情）》

适合人群：算法零基础或者基础非常薄弱的同学。

• 《牛客算法基础提升班（点击查看详情）》

适合人群：想提升算法的任何同学，有助于工作中晋升，求高薪 offer 等等

• 《求职算法真题精讲-中级班（点击查看详情）》

适合人群：适合有完整算法知识体系，准备参加校招或找实习的同学

- 《求职算法真题精讲-高级班（点击查看详情）》

适合人群：适合算法基础比较好，准备参加校招或找实习的同学，学完后应对大部分公司的高难度笔试面试题可以做到毫无压力。

3、项目

面试开场，面试官会比较偏向于先从你的项目问起，根据项目不断提出问题，找到一个点，不断深入提问，这就是在考察你对基础知识的掌握是否牢固。所以在备战时，要有一个有分量的项目经验很重要。

如果你现在缺少项目经历或者缺少能拿的出手的项目，建议可直接报名牛客网的项目课。牛客网 CEO 叶神带你做项目，真真正正企业的角度从 0 到 1 搭建项目：

- 《求职入门项目课（点击查看详情）》

适合人群：没有项目经验的同学。只需六周，实习项目经验零突破

- 《高薪求职项目课（点击查看详情）》

适合人群：之前有过项目经验，有任何一门语言的编程基础，最好有 Java 基础的同学。只需 8 周，完成牛客讨论区项目，收获一份高薪求职项目

如果你已经具备了好的项目，那要注意的就是如何讲好项目。

面试官的关注点在你的能力和潜力：

（1）你了解哪些部分？你深入了解哪些部分？你横向了解哪些部分？

（2）你怎么解决问题？你如何举一反三？你怎么优化项目？你如何快速学习？

关于如何应对技术面，推荐观看牛客 CEO 叶神讲解的《技术面之项目面世技巧大揭秘》（点击即可观看）

4、非技术类问题

非技术类问题一般是在 HR 面，切记不要觉得不重要，这恰恰是你的最后一关。 以下列举一些常见问题：

（1）自我介绍

（2）你觉得自己优点有哪些？缺点有哪些？

（3）你并非毕业于名校，为什么不录用 985/211，而录用你？

（4）你对加班的看法？

（5）你对薪资的要求。

（6）在五年的时间内，你的职业规划？

（7）说说你对行业、技术发展趋势的看法。
（8）对工作的期望与目标。
（9）目前有没有男朋友/女朋友
（10）你为什么愿意到我们公司来？
（11）你有什么要问我的么？

推荐一些知乎经典回答和文章：

- HR 面试时的 12 种套路
  https://zhuanlan.zhihu.com/p/21635958
- 面试 70 问经典回答
  https://zhuanlan.zhihu.com/p/48312070
- 面试必知的 23 个回答技巧
  https://zhuanlan.zhihu.com/p/51817315
- 面试阶段如何与 HR 沟通薪酬？
  www.zhihu.com/question/24997992/answer/29731652
- 面试如何回答 HR「你未来五年计划」的问题？
  https://www.zhihu.com/question/49150895/answer/426254885
- 如何回答面试官「为什么不录用 985、211 人才，而录用你」的问题？
  https://www.zhihu.com/question/63825818/answer/239111715
- 面试者如何回应面试官问的「你有哪些要问我的？」？
  https://www.zhihu.com/question/28058827/answer/128278789

4.3.2 面试必备技巧

根据不同的面试类型，给大家总结了相对应的面试攻略 推荐观看牛客 CEO 叶神亲讲：《技术面之项目面试技巧大揭秘（点击可观看）》

1、电话面试：

面试官和学生之前通过电话进行沟通面试

- 考察：专业能力，沟通能力
- 优点：学生看不到面试官，压力会相对小一些
- 缺点：容易信号中断，影响心情，导致面试失败
- 难度：★★★
- 学生惧怕指数：★★

• 面试分析：此种面试适合相对内敛的同学，不太适合有表现力的同学。当然有表现力的同学也不用担心自己的又是展现不出来，你可以通过语言来表现出自己的优势。

• 面试建议：

（1）良好的环境和设备；提前准备一个安静并且信号比较好的环境，保证手机电量充足，身边备好笔纸，记录和验算起来都比较方便。

（2）不要想着作弊；有的同学想着投机取巧，找一个大神在旁边帮助自己，绝对不要这样做，因为一旦被发现后果很严重，而且即使你通过这一轮，后面还有好几轮面试，实力不济还是没法通过。

（3）保持电话通畅；电话一定要调成震动并随时在身边，很多时候错过的电话就是真的错过了。打回去又要输入分机号码，直接 gg 了。

2、视频面试：

面试官和学生之间通过视频的形式进行面试，大多数用的是牛客网的在线视频面试，也有极个别公司采用 qq 或者微信视频的形式。视频面试前，大家可以到**牛客网>面试>AI 模拟面试（点击进行练习）**进行练习，体验真实现场面试，提前熟悉视频面试流程。

• 考察：专业能力，沟通能力，亲和度（性格分析等）

• 优点：不用去现场面试，省去很多时间和经历甚至财力。

• 缺点：看不到面试官会心里没底，面试官看着自己写代码比较紧张（针对技术 同学）

• 难度：★★★

• 学生惧怕指数：★★★★

• 面试分析：此种面试适合性格相对沉稳或者开朗的学生，对有上镜感的同学也是一个先天的优势，当然，不是绝对的优势，但是面试官盯着你能看到你整体解题的思路，学生往往会紧张。

• 面试建议：

（1）良好的心态：找工作是一个持久战，除了一些大牛，多数人可能要面试很多家才能收到一份 offer。所以，千万不要一次被拒就患得患失，沮丧退却，要怀着检验自己、发现问题、持续提高进步的心态去求职。当然，实力是好心态的保证，一定要准备充分了。

（2）提前测试；准备好网络环境和周围环境：面试过程一般是注意力高度集中的，一定要提前检查好网络环境，避免出现网络异常等情况，影响面

试。同时也要注意，面试时准备一个安静的环境，以免受到周遭噪音等的影响。

（3）准备一下算法编程；技术同学在使用面试系统面试的时候，很可能会遇到在线码代码的考验，而且还是在面试官盯着的情况下压力可想而知。所以大家要提前准备一下，能跟大大减缓你在面试时的压力。

3、交叉面试：

往往是一个学生两个面试官轮流面试，根据两个面试官对其的评价来综合评判是否通过面试。需要注意的是，交叉面并不是你通过了一面进入二面的概念。

- 考察：专业能力，沟通能力，亲和度（性格分析等），综合能力
- 优点：较为公平，不会因为一个面试官对你的评价而一棒子打死。
- 缺点：学生被面试的多而疲惫，会导致发挥不好。
- 难度：★★★
- 学生惧怕指数：★★★★
- 面试分析：此种面试适合大多数同学，结果较为公平，不会因为一个面试官的态度影响这个学生的结果，当然，对一些性格稍微偏颇的同学也是有一定程度的优势，毕竟是两个面试官来评判。

- 面试建议：

（1）良好的心态；同“视频面试”

（2）礼貌+实力；不用想着去取悦某一个面试官，在不了解面试官的情况下过度表现不见得有好的结果，发挥自己的实力，不卑不亢，做自己就好了。

（3）回答一致：两个面试官交叉面试，一定要注意回答的一致性，不要编造一些不存在的东西，因为面试官有很大可能会沟通你的回答，回答不一致就显得很可疑了。

4、现场面试：

是指学生通过去公司现场、临时办公点（如酒店、学校教室）对学生进行面对面的面试考察。

- 考察：专业能力，沟通能力，亲和度（性格分析等），综合能力
- 优点：面试官能够较为全面的考察你的真实水平。

• 缺点：现场人比较多，等待时间往往比较长，且较为不可控。
• 难度：★★★
• 学生惧怕指数：★★★
• 面试分析：此种面试是最为原始的面试形式，所有的面试形式都是根据这个衍生出来的形式，往往是一对一的，一个面试官面对一个学生，对学生进行考察。

• 面试建议：
（1）准备好简历；现场面试往往不可控，有的公司可能会提前为你准备好你的简历，有的公司可能不会为你准备，当然，也可能为你准备了简历但是现场找不到的情况，所以，准备好自己的纸质版简历，最好准备两份到三份，比较稳妥。
（2）准时到达；面试邀请往往会提前发到你的邮箱，手机或者电话通知等等，那么往往就要你提前估算好时间，提前十分钟或者二十分钟到指定地址，以备有特殊临时情况及时处理。
（3）保持一个良好的心态和礼仪；面试有可能当面出结果，也可能后面等结果，如果当场被拒，也要保持一个良好的心态和礼仪，比如进了一个房间，要先敲门，不卑不亢的进行面试，不卑不亢很关键，我们做到礼仪相待，但是也没有必要过于谦卑，因为学生和面试官是同等的，面试就是一个互相选择的过程。

5、群体面试：

是指 n 个面试官对 n 个学生，针对一个主题进行一个无领导小组讨论，针对大家的综合表现来对学生做出判断。

• 考察：沟通能力，领导能力，组织能力，抗压能力等
• 优点：多个面试官在场，从面试官的角度较为公平
• 缺点：往往不可控，跟你一组的同学不一定是什么情况，充满了太多不可控因素，但是反过来也往往也会更能体现出你应对各种人和各种情况的临场反应能力。
• 难度：★★★★
• 学生惧怕指数：★★★★★
• 面试分析：此种面试是大多数同学比较恐惧的一种面试形式，但是其实掌握一些小技巧并且摆正心态，认清自己，这种面试还是挺容易的。

• 面试建议：

（1）提前准备好队员；这一点是在有条件的基础上，比如正好去面试的你们都认识，那么你们可以提前分工，而且认识的人往往心态上比较放松，而且也会敢于去表达自己的观点。

（2）不害怕，不打压；有的同学可能说我根本都不认识谁，也没有办法组队，而且那也是一种非常理想的状态，那么我们在讨论的时候就正常发表自己的观点就好，不要因为某个人比较强势就不敢否定或者因为某个人比较弱势就去压制，正常发言即可。

（3）认清自己的优势和弱势；找准自己的优势和弱势，在交流中发挥出自己的优势，比如自己适合领导组织，那么就申请当领导，但是很多同学会觉得当领导就比较容易通过，但是如果你没有组织好，有时候往往会起反作用，所以认清自己很关键。

6、压力面试：

是指面试官在面试中故意给学生一些压力，看学生的反应和应对情况。

- 考察：抗压能力
- 优点：只要心态把握的好，没有任何难点
- 缺点：学生在面试的过程中都是很脆弱的，往往更容易被击败。
- 难度：★★★
- 学生惧怕指数：★★★★★
- 面试分析：此种面试是大多数同学比较恐惧的一种面试，面对求职期间的种种失败本身就很脆弱，而面对压力，没有点心理承受能力就很容易漏出破绽。

- 面试建议：

（1）摆正心态；压力面可能贯穿在某个面试中，也可能单独是一个面试，不管怎样，这些都是内部人员去定的，所以你根本不知道哪里是压力面，所以无论何时，遇到自己不会的问题，也要沉着面对，哪怕当时面试官表现的如何压抑，你都不要表现出这个问题没有解决仿佛世界就塌下来的样子，或者在群面中所有人 都来评价你的不好，也要不卑不亢，不要落荒而逃。

（2）实事求是；正视面试官提到不足，切勿硬要解释和掩饰，那结果只能是悲剧了，会面临进一步的压力提问。反过来，直面自己的不足，面试官会觉得至少你很真诚，你的不足也会被淡化。承认不足之后，如果可以提

出自己的改善计划，或者很有诚意的请教下面试官，化被动为主动就更好了。

（3）掌握技巧；如果是在 HR 面中遇到了压力面试，可以提前准备一些常用技巧去应对，例如，放缓回答速度，赢取思考时间，避开陷阱，灵活转化问题等等。

7、HR 面试：

HR 面往往是面试的最后一个阶段，HR 会对你的各种情况进行一个综合的深入了解，如家庭情况，婚恋情况，性格等等进行一个综合的了解，因为这些都是涉及到工作的一部分，需要综合考虑你是否适合这个岗位，这个公司。

- 考察：性格，背景
- 优点：学生只需要实事求是，放轻松的面对即可。
- 缺点：每个公司用人要求都不一样，考察的一些点都不太一样，没有办法多去准备，而且性格也不是能马上就改掉的。
- 难度：★★★
- 学生惧怕指数：★★★
- 面试分析：此种面试是很多同学容易忽略的一环，很多公司往往在这个环节刷人还挺多，所以大家容易犯的错误就是不重视。

- 面试建议：

（1）做足重视；太多学生对 hr 面试并不重视，也没有做好准备，导致了最后的落选。

（2）礼仪和诚信；这点其实是在所有面试中都应该有的，不能说给你压力了，你就没有一个好的心态了，甚至还谩骂 HR，诚信就更是上升到做人的一个标准了。

五、名企经典笔试题目

5.1 腾讯 2019-2020 校园招聘真题（点击立刻刷题）

1、逆序对

作为程序员的小 Q，他的数列和其他人的不太一样，他有2^n个数。

老板问了小 Q 一共 m 次，每次给出一个整数$q_i(1 <= i <= m)$，要求小 Q 把这些数每2^{q_i}分为一组，然后把每组进行翻转，小 Q 想知道每次操作后整个序列中的逆序对个数是多少呢？

例如：
对于序列 1 3 4 2，逆序对有(4, 2),(3, 2),总数量为 2。
翻转之后为 2 4 3 1，逆序对有(2, 1),(4, 3), (4, 1), (3, 1),总数量为 4。

输入描述:

第一行一个数$n(0 \le n \le 20)$
第二行2^n个数，表示初始的序列($1 \le$ 初始序列 $\le 10^9$)
第三行一个数$m(1 \le m \le 10^6)$
第四行 m 个数表示$q_i(0 \le q_i \le n)$

输出描述:

m 行每行一个数表示答案。

输入例子 1:

```
2
2 1 4 3
4
1 2 0 2
```

输出例子 1:

```
0
6
6
0
```

例子说明 1:

初始序列 2 1 4 3
$2^{q_1} = 2$ ->
第一次：1 2 3 4 -> 逆序对数为 0
$2^{q_2} = 4$->
第二次：4 3 2 1 -> 逆序对数为 6
$2^{q_3} = 1$->
第三次：4 3 2 1 -> 逆序对数为 6
$2^{q_4} = 4$->
第四次：1 2 3 4 -> 逆序对数为 0

题目详解

/*分治

所有 2^i 上的逆序对，只要记录 2i-1 和另一半 2i-1 之间的逆序对，例如 2341 里面，2^1 上有 1 个，2^2 上一共 3 个，但是有一个在 2^1 中算过了，所以就是 2 个。

同时，要计算 2^i 下所有可能的组合,即 2^{i-1}*2^{i-1},再乘上 2^i 的个数，即 2^{n-i}，得到 2^{n+i-2},在 merge 的过程中同时还要记录等于的情况，因为这些等于的不管是否翻转都不是逆序对，所以对每一个 2^i 还要减去等于的对数。

然后翻转的时候，假如翻转数字的是 x，对于每个 2^i(x >= i),值都改为上面求出的数 - 原值逆序对总数就是从 1-n 求和。*/

```
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int inf = 0x3f3f3f3f;
LL sum[22][2];
int n, a[1 << 21];
void dfs(int l, int r, int deep) {
    if (l >= r)
        return;
    int mid = (l + r) >> 1;
    dfs(l, mid, deep - 1);
    dfs(mid + 1, r, deep - 1);
    for (int i = l; i <= mid; i++) {
        int temp = lower_bound(a + mid + 1, a + r + 1, a[i]) - (a + mid + 1);
        sum[deep][0] += (LL)temp;
        temp = r - mid - (upper_bound(a + mid + 1, a + r + 1, a[i]) - (a + mid + 1));
        sum[deep][1] += (LL)temp;
    }
    sort(a + l, a + r + 1);
}
int main() {
    scanf("%d", &n);
    int y = (1 << n);
```

```
    for (int i = 1; i <= y; i++) scanf("%d", a + i);
    dfs(1, y, n);
    int q, x;
    scanf("%d", &q);
    while (q--) {
        scanf("%d", &x);
        while (x) {
            swap(sum[x][0], sum[x][1]);
            x--;
        }
        LL ans = 0;
        for (int i = 1; i <= n; i++) ans += sum[i][0];
        printf("%lld\n", ans);
    }
    return 0;
}
```

2、逛街

小 Q 在周末的时候和他的小伙伴来到大城市逛街，一条步行街上有很多高楼，共有 n 座高楼排成一行。

小 Q 从第一栋一直走到了最后一栋，小 Q 从来都没有见到这么多的楼，所以他想知道他在每栋楼的位置处能看到多少栋楼呢？（当前面的楼的高度大于等于后面的楼时，后面的楼将被挡住）

输入描述:

输入第一行将包含一个数字 n，代表楼的栋数，接下来的一行将包含 n 个数字 w_i(1<=i<=n)，代表每一栋楼的高度。

1<=n<=100000;

1<=w_i<=100000;

输出描述:

输出一行，包含空格分割的 n 个数字 vi，分别代表小 Q 在第 i 栋楼时能看到的楼的数量。

输入例子 1:

6

5 3 8 3 2 5

输出例子 1:

3 3 5 4 4 4

例子说明 1:

当小 Q 处于位置 3 时，他可以向前看到位置 2,1 处的楼，向后看到位置 4,6 处的楼，加上第 3 栋楼，共可看到 5 栋楼。当小 Q 处于位置 4 时，他可以向前看到位置 3 处的楼，向后看到位置 5,6 处的楼，加上第 4 栋楼，共可看到 4 栋楼。

题目详解

```
/*单调栈
开辟一个数组，保留往右看得到的数量，从右往左遍历，利用单调栈将看得到的数量保留在数组中，再从左往右遍历，获取往左看的计数。
*/

#include <bits/stdc++.h>
using namespace std;
int n, a[100005], ans[100005];
int main() {
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    stack<int> s;
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    memset(ans, 0, sizeof ans);
    for (int i = 1; i <= n; i++) {
        ans[i] += s.size();
        while (!s.empty() && s.top() <= a[i]) s.pop();
        s.push(a[i]);
    }
    while (!s.empty()) s.pop();
    for (int i = n; i >= 1; i--) {
        ans[i] += s.size();
        while (!s.empty() && s.top() <= a[i]) s.pop();
```

```
            s.push(a[i]);
        }
        for (int i = 1; i <= n; i++) cout << ans[i] + 1 << " \n"[i == n];
        return 0;
}
```

3、小 Q 的歌单

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

输入描述:

每个输入包含一个测试用例。

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

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

输出描述:

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

输入例子 1:

5

2 3 3 3

输出例子 1:

9

题目详解

```
/*01 背包问题
问题转化为有 x+y 种物品，其中 x 种的容积为 a，y 种的容积为 b，背包容积为 k，问背包装满一共有多少种解法?
*/
```

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1000000007;
int a, x, b, y, k;
long long dp[1010];
int p[210];
int main() {
    scanf("%d", &k);
    scanf("%d%d%d%d", &a, &x, &b, &y);
    dp[0] = 1;
    for (int i = 1; i <= x; i++) p[i] = a;
    for (int i = x + 1; i <= x + y; i++) p[i] = b;
    for (int i = 1; i <= x + y; i++) {
        for (int j = k; j >= p[i]; j--) dp[j] = (dp[j] % mod + dp[j - p[i]] % mod) % mod;
    }
    printf("%lld\n", dp[k] % mod);
    return 0;
}
```

5.2 字节跳动 2019 校园招聘真题（点击立刻刷题）

1、特征提取-研发

小明是一名算法工程师，同时也是一名铲屎官。某天，他突发奇想，想从猫咪的视频里挖掘一些猫咪的运动信息。为了提取运动信息，他需要从视频的每一帧提取“猫咪特征”。一个猫咪特征是一个两维的 vector<x, y>。如果 x_1=x_2 and y_1=y_2，那么这俩是同一个特征。

因此，如果喵咪特征连续一致，可以认为喵咪在运动。也就是说，如果特征<a, b>在持续帧里出现，那么它将构成特征运动。比如，特征<a, b>在第 2/3/4/7/8 帧出现，那么该特征将形成两个特征运动 2-3-4 和 7-8。现在，给定每一帧的特征，特征的数量可能不一样。小明期望能找到最长的特征运动。

输入描述:

1、第一行包含一个正整数 N，代表测试用例的个数。
2、每个测试用例的第一行包含一个正整数 M，代表视频的帧数。
3、接下来的 M 行，每行代表一帧。其中，第一个数字是该帧的特征个数，接下来的数字是在特征的取值；比如样例输入第三行里，2 代表该帧有两个猫咪特征，<1，1>和<2，2>
所有用例的输入特征总数和<100000
4、N 满足 1≤N≤100000，M 满足 1≤M≤10000，一帧的特征个数满足 ≤10000。
特征取值均为非负整数。

输出描述:

对每一个测试用例，输出特征运动的长度作为一行

输入例子 1:

```
1
8
2 1 1 2 2
2 1 1 1 4
2 1 1 2 2
2 2 2 1 4
0
0
1 1 1
1 1 1
```

输出例子 1:

```
3
```

题目详解

```
/*hash
开辟两个 map，一个用来保存当前帧的特征，一个用来保存上一帧的特征。用 hashmap 比 treemap 更合理，但是 C++ 的 hashmap 不能接受 pair 作为 key 转化成 string 即可
*/

#include<bits/stdc++.h>
```

```
using namespace std;
unordered_map<string, int> last;
unordered_map<string, int> cur;

string make_str(int x, int y) {
    return to_string((long long)x) + to_string((long long)y);
}
int N, M;
int main() {
    scanf("%d", &N);
    while(N--) {
        scanf("%d", &M);
        int res = 1;
        while(M--) {
            int cnt; scanf("%d", &cnt);
            cur.clear();
            while(cnt--) {
                int x, y; scanf("%d%d", &x, &y);
                if(last.count(make_str(x, y))) {
                    cur[make_str(x, y)] += (last[make_str(x, y)] + 1);
                    res = max(res, cur[make_str(x, y)]);
                } else {
                    cur[make_str(x, y)]++;
                }
            }
            last = cur;
        }
        last.clear();
        printf("%d\n", res);
    }
}
```

2、万万没想到之抓捕孔连顺-研发

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

1. 我们在字节跳动大街的 N 个建筑中选定 3 个埋伏地点。
2. 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（相距最远的两名特工间的距离的最大值）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。

注意：

1. 两个特工不能埋伏在同一地点
2. 三个特工是等价的：即同样的位置组合(A,B,C) 只算一种埋伏方法，不能因“特工之间互换位置”而重复使用

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

输入例子 1:

4 3
1 2 3 4

输出例子 1:

4

例子说明 1:

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

输入例子 1:

5 19

1 10 20 30 50

输出例子 1:

1

例子说明 1:

可选方案（1，10，20）

题目详解

```
/*滑动窗口
设置两个指针，先移动右边，当不满足得时候再移动左边，每次需要计数
*/

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

long long C(long long n) { return (n - 1) * n / 2; }
int main() {
    long long n, d, count = 0;
    cin >> n >> d;
    vector<long long> v(n);
    for (int i = 0, j = 0; i < n; i++) {
        cin >> v[i];
        while (i >= 2 && (v[i] - v[j]) > d) {
            j++;
        }
        count += C(i - j);
    }
    cout << count % 99997867;
    return 0;
}
```

3、毕业旅行问题-研发

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:

城市个数 n（1<n≤20，包括北京）

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

输入例子 1:

4

0 2 6 5

2 0 4 4

6 4 0 2

5 4 2 0

输出例子 1:

13

例子说明 1:

共 4 个城市，城市 1 和城市 1 的车费为 0，城市 1 和城市 2 之间的车费为 2，城市 1 和城市 3 之间的车费为 6，城市 1 和城市 4 之间的车费为 5，依次类推。假设任意两个城市之间均有单程票可购买，且票价在 1000 元以内，无需考虑极端情况。

题目详解

```
/*状压 DP
dp[i][j]中的 i 是一个二进制形式的数，表示经过城市的集合，如 0111 表示经过了城市 0,1,2。dp[i][j]表示经过了 i 中的城市，并且以 j 结尾的路径长度
*/

#include <bits/stdc++.h>
```

```
using namespace std;

int getAns(vector<vector<int>> &nums){
      const int MAX = 0x0fffffff;
      int n = nums.size();
     int stateNum = 1 << n;
     vector<vector<int> > dp(stateNum,vector<int>(n,MAX));
      dp[1][0] = 0; //从城市 0 出发，所以经过城市 0，以城市 0 结尾的路径为 0
      //从城市 0 出发，更新和其他城市的距离
      for(int i=1;i<stateNum;i++){
            for(int j=0;j<n;j++){
                  if(dp[i][j] != MAX){ //如果已经访问过
                        for(int k=0;k<n;k++){
                              if( (i & (1 << k) ) == 0){
                                    //没有访问过 k，且从这里到 k 的距离小于原来的距离，
则更新
                                    dp[i | (1 << k)][k] = min(dp[i | (1 << k)][k],dp[i][j] +
nums[j][k]);
                              }
                        }
                  }
            }
      }
      int res = MAX;
      for(int i=1;i<n;i++){
            res = min(res,dp[stateNum-1][i] + nums[i][0]);
      }
      return res;
}
int main(int argc, char const *argv[])
{
    int n;
      while(cin>>n){
            vector<vector<int>> edges(n,vector<int>(n,0));
```

```
            int x;
            for(int i=0; i<n; i++){
                for(int j=0; j<n; j++){
                    cin>>edges[i][j];
                }
            }
            cout<<getAns(edges)<<endl;
        }
        return 0;
}
```

5.3 百度 2019 校园招聘真题（点击立刻刷题）

1、探险安排-Web 前端工程师

小明要为 n 个人计划一次火星的探险，其中一个重要的任务是为每个参与者安排食物。仓库里面有 m 个能用一天的食物包裹，每个食物包裹有不同的类型 a_i。

每个人每天必须用且只能用一个食物包裹。由于某些原因，在整个过程中，每个人只能用同一种类型的食物包裹，但是不同的人用的食物包裹可以不一样。

给出人数以及食物包裹的情况，请你求出这趟探险最多可以持续多少天。

输入描述:

第一行两个整数，n 和 m，表示人数和食物包裹的个数。

第二行 m 个整数，表示每个食物包裹的类型。

满足 $1<=n<=100,1<=m<=100,1<=a_i<=100$。

输出描述:

一个整数，表示最多持续的天数；如果一天都无法持续，输出 0。

输入例子 1:

```
4 10
1 5 2 1 1 1 2 5 7 2
```

输出例子 1:

2

题目详解

```
/*
暴力枚举可存活的天数，依次判断食物的份数，若满足就更新最大持续天数。
*/

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

int food[101] = { 0 };
int main() {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < m; ++i) {
        int t;
        cin >> t;
        food[t]++;
    }
    for (int k = m / n; k > 0; --k) {
        int cnt = 0;
        for (int i = 1; i <= 100; ++i) cnt += food[i] / k;
        if (cnt >= n) {
            cout << k << endl;
            return 0;
        }
    }
    cout << 0 << endl;
    return 0;
}
```

5.4 美团点评 2019 校园招聘真题（点击立刻刷题）

1、考试策略

小明同学在参加一场考试，考试时间 2 个小时。试卷上一共有 n 道题目，小明要在规定时间内，完成一定数量的题目。考试中不限制试题作答顺序，对于 i 第道题目，小明有三种不同的策略可以选择:

(1)直接跳过这道题目，不花费时间，本题得 0 分。

(2)只做一部分题目，花费 pi 分钟的时间，本题可以得到 ai 分。

(3)做完整个题目，花费 qi 分钟的时间，本题可以得到 bi 分。

小明想知道，他最多能得到多少分。

输入描述:

第一行输入一个 n 数表示题目的数量。

接下来 n 行，每行四个数 p_i，a_i，q_i，b_i。(1≤n≤100，1≤p_i≤q_i≤120，0≤a_i≤b_i≤1000)。

输出描述:

输出一个数，小明的最高得分。

输入例子 1:

```
4
20 20 100 60
50 30 80 55
100 60 110 88
5 3 10 6
```

输出例子 1:

```
94
```

题目详解

```
/*
01 背包问题
dp[x]表示容量为 x 的背包最多能装多少
*/

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

```
const int maxn = 100 + 5;
int n, p[maxn], a[maxn], q[maxn], b[maxn], dp[125];

int main() {
    scanf("%d", &n);
    for(int i = 0; i < n; i++)
        scanf("%d%d%d%d", &p[i], &a[i], &q[i], &b[i]);
    for(int i = 0; i < n; i++)
        for(int j = 120; j >= 0; j--) {
            if(p[i] <= j)   //考虑这一题全做完
                dp[j] = max(dp[j], dp[j-p[i]] + a[i]);
            if(q[i] <= j)    //考虑这一题只做一半
                dp[j] = max(dp[j], dp[j-q[i]] + b[i]);
        }
    cout<<dp[120]<<endl;
}
```

2、外卖满减

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。

输入例子 1:

5 20

18 19 17 6 7

输出例子 1:

23

题目详解

```
/*
01 背包问题
*/

#include<bits/stdc++.h>
using namespace std;
int main() {
    int n,X;
    cin >> n >> X;
    vector<int> dp(X+1,10001);//dp[i]是消费达到 i 元所需的最低消费
    int price;
    for(int i =0;i<n;i++){
        cin >> price;
        for(int j = X;j>=0;j--){
            if(j>price){
                dp[j]=min(dp[j],dp[j-price]+price);
            }
            else{
                dp[j]=min(dp[j],price);
            }
        }
    }

    cout << dp[X]<< endl;
    return 0;
}
```

3、路由器

一条直线上等距离放置了 n 台路由器。路由器自左向右从 1 到 n 编号。第 i 台路由器到第 j 台路由器的距离为|i-j|。每台路由器都有自己的信号强度，第 i 台路由器的信号强度为 ai。所有与第 i 台路由器距离不超过 ai 的路由器可以收到第 i 台路由器的信号（注意，每台路由器都能收到

自己的信号）。问一共有多少台路由器可以收到至少 k 台不同路由器的信号。

输入描述:

输入第一行两个数 n ， k（1≤n ， k≤10^5）

第二行 n 个数，a1 ， a2 ， a3…… ， an（0≤ai≤10^9）

输出描述:

输出一个数，一共有多少台路由器可以收到至少 k 台不同路由器的信号。

输入例子 1:

4 4

3 3 3 3

输出例子 1:

4

题目详解

```
/*
差分数组
*/

#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int maxn = 2e5 + 100;
const int mod = 1e9 + 7;
ll n, k;
ll a[maxn];
ll b[maxn];
ll ans;
int main() {
    ios::sync_with_stdio(0);
    cin.tie();
    cout.tie();
    cin >> n >> k;
    for (int i = 1; i <= n; i++) cin >> a[i];
```

```
    for (int i = 1; i <= n; i++) {
        ll l = max(1LL, i - a[i]);
        ll r = min(n, i + a[i]) + 1;
        b[l]++;
        b[r]--;
    }
    for (int i = 1; i <= n; i++) b[i] = b[i - 1] + b[i];
    for (int i = 1; i <= n; i++)
        if (b[i] >= k)
            ans++;
    cout << ans << endl;
    return 0;
}
```

5.5 爱奇艺 2020 校园招聘真题（点击立刻刷题）

1、切割块

有一个 x*y*z 的立方体，要在这个立方体上砍 k 刀，每一刀可以看作是用一个平行于立方体某一面的平面切割立方体，且必须在坐标为整数的位置切割，如在 x=0.5 处用平面切割是非法的。
问在切割 k 刀之后，最多可以把立方体切割成多少块。

输入描述:

输入仅包含一行，一行包含 4 个正整数 x,y,z,k 分别表示 x*y*z 的立方体和切割 k 刀。（1<=x,y,z<=10^6,0<=k<=10^9）

输出描述:

输出仅包含一个正整数，即至多切割成多少块。

输入例子 1:

2 2 2 3

输出例子 1:

8

题目详解

```
/*
```

```
三个方向尽量平均分掉刀数，这样切的块数最多
三个方向依次增加刀数，最多为边长-1 刀
当无法再切或刀数用尽时，结束
三面切的刀数，各自加一，然后乘积，就是结果
*/

#include <bits/stdc++.h>
using namespace std;
int main() {
    int a[5], k;
    long long int maxd, maxk, m = 104909296875;
    scanf("%d%d%d%d", &a[1], &a[2], &a[3], &k);
    //刀数
    maxd = (long long int)(a[1] - 1) + (a[2] - 1) + (a[3] - 1);
    //快数
    maxk = (long long int)a[1] * a[2] * a[3];
    // k 超过刀数
    if (k >= maxd) {
        printf("%lld\n", maxk);
        return 0;
    } else
        while (maxd != k) {
            //找最长的那条边
            sort(a + 1, a + 4);
            //最长边减 1
            a[3]--;
            //新的刀数和快数
            maxd = (long long int)(a[1] - 1) + (a[2] - 1) + (a[3] - 1);
            maxk = (long long int)a[1] * a[2] * a[3];
        }
    printf("%lld", maxk);
    return 0;
}
```

2、排列计数

给定一个大小为 N-1 且只包含 0 和 1 的序列 A_1 到 A_{N-1}，如果一个 1 到 N 的排列 P_1 到 P_N 满足对于 $1 \leqslant i < N$，当 $A_i=0$ 时 $P_i<P_{i+1}$，当 $A_i=1$ 时 $P_i>P_{i+1}$，则称该排列符合要求，那么有多少个符合要求的排列？

输入描述:

第一行包含一个整数 N，$1<N \leqslant 1000$。

第二行包含 N-1 个空格隔开的整数 A_1 到 A_{N-1}，$0 \leqslant A_i \leqslant 1$。

输出描述:

输出符合要求的排列个数对 109+7 取模后的结果。

输入例子 1:

4

1 1 0

输出例子 1:

3

例子说明 1:

符合要求的排列为{3 2 1 4}、{4 2 1 3}和{4 3 1 2}。

题目详解

```
/*
动态规划
dp(i, j)表示确定了排列中到 P[i] 为止的前 i+1 个元素，并且 P[i]和未选择元素的相对大小为 j 的方案数（即未选择的元素中，有 j 个元素比 P[i]小）。
在状态转移时，dp(i, j) 会从 dp(i-1, k) 转移而来，其中 k 代表了 P[i-1] 的相对大小。
如果 S[i-1]为 D，那么 k 不比 j 小；如果 S[i-1]为 I，那么 k 必须比 j 小。
*/

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

const int mod = 1e9 + 7;
int n;
int S[1005];
int numPermsDISequence() {
```

```
    vector<vector<int> > dp(n + 1, vector<int>(n + 1, 0));
    for (int i = 0; i < n + 1; i++) dp[0][i] = 1;
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j <= i; j++) {
            if (S[i - 1] == 1) {
                for (int k = j; k < i; k++) {
                    dp[i][j] += dp[i - 1][k];
                    dp[i][j] %= mod;
                }
            } else {
                for (int k = 0; k < j; k++) {
                    dp[i][j] += dp[i - 1][k];
                    dp[i][j] %= mod;
                }
            }
        }
    }
    int ans = 0;
    for (int i = 0; i <= n; i++) {
        ans += dp[n][i];
        ans %= mod;
    }
    return ans;
}
int main() {
    scanf("%d", &n);
    n--;
    for (int i = 0; i < n; i++) scanf("%d", &S[i]);
    printf("%d\n", numPermsDISequence());
    return 0;
}
```

3、或和异或

首先给出一个长度为 2^n 的序列 $A=\{a_1,a_2,\ldots,a_{2^n-1},a_{2^n}\}$，我们定义一个值 v，这个值是由如下计算方法得到的，首先将 A 序列的第 2*i-1 位和第 2*i 位(i=1,2,.., 2^{n-2})进行 OR 操作得到新数列 A'，然后再对 A'序列操作，将 A'序列的第 2*i 位和第 2*i-1 位(i=1,2,...,2^{n-2})进行 XOR 操作得到 A''，对 A''按照第一次操作方式进行 OR 操作，因为序列长度为 2^n,所以显然进行 n 次操作之后就只剩下一个数字了此时这个数字就是 v。
例如 A={1，2，3，4}，第一次操作之后为{1|2=3，3|4=7}，第二次操作后，{3^7=4},所以 v=4。

但是显然事情并没有那么简单，给出 A 序列后，还有 m 个操作，每个操作表示为"a b"，表示将 A[a]改为 b，之后再对 A 序列求 v 值。

输入描述:

输入第一行包含两个正整数 n，m，分别表示 A 序列的长度为 2^n,操作数量为 m。（1<=n<=17,1<=m<=10^5）

输入第二行包含 n 个正整数，中间用空格隔开，表示 A 序列。（0<=ai<=2^30）

接下来有 m 行，每行包含两个正整数 a，b，表示一次操作，即把 A[a]变成 b。(1<=a<=2^n，0<=b<=2^30)

输出描述:

输出包含 m 行，第 i 行表示进行了第 i 次操作之后，A 序列的 v 值。

输入例子 1:

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

输出例子 1:

1
2
7
7

题目详解

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

int find_v(int *s, int n)   // n 为整个数组的长度
{
    int i;
    int j;
    int *s1 = s;
    int len;
    int x = n;   // x 中保存数组中的元素个数，每执行一次操作就会减小一半
    for (i = 1; i <= n; i++) {
        len = n / (pow(2, i));   //执行完操作后的长度
        int *p = new int[len];
        if (i % 2 == 1) {
            for (j = 0; j < x; j += 2) {
                int k = j / 2;
                p[k] = (s1[j]) | (s1[j + 1]);
            }
        } else if (i % 2 == 0) {
            for (j = 0; j < x; j += 2) {
                int k = j / 2;
                p[k] = (s1[j]) ^ (s1[j + 1]);
            }
        }
        if (len == 1)
            return p[0];
        else {
            s1 = p;
            x = x / 2;
        }
    }
}
int main() {
    int n, m;
```

```
        cin >> n;
        cin >> m;
        int a[m];
        int b[m];
        int num = pow(2, n);
        int s[num];
        int i = 0;
        int j = 0;
        while (i < num) {
                cin >> s[i];
                i++;
        }

        while (j < m) {
                cin >> a[j];
                cin >> b[j];
                j++;
        }

        for (j = 0; j < m; j++) {
                s[a[j] - 1] = b[j];
                int result = find_v(s, num);
                cout << result << endl;
        }
}
```

5.6 京东 2019 校园招聘真题（点击立刻刷题）

1、还原-算法

有一个含有 n 个数字的序列，每个数字的大小是不超过 200 的正整数，同时这个序列满足以下条件：

1. a_1<=a_2
2. a_n<=a_(n-1)（此时 n>2）

3. a_i<=max(a_{i-1},a_{i+1})

但是很不幸的是，在序列保存的过程中，有些数字丢失了，请你根据上述条件，计算可能有多少种不同的序列可以满足以上条件。

输入描述:

1、输入第一行是一个 n，表示这个序列的长度。(3<=n<=10^4)

2、输入第二行有 n 个非负整数，中间用空格隔开，如果数字为 0，说明这个数字丢失了，其他数字则都在 1-200 之间。

输出描述:

输出仅包含一个整数，即方案数对 998244353 取模的结果。

输入例子 1:

3

2 0 1

输出例子 1:

1

题目详解

```
#include <bits/stdc++.h>
using namespace std;
const int maxN = 1e4 + 7;
const int mod = 998244353;

int n, a[maxN];
// dp[i][j][0/1/2] 代表以 a[i]为结尾，a[i] = j 时的序列种数。
// 第三维代表 a[i-1]和 a[i]的大小关系(>, ==, <)。
long long dp[maxN][207][3];
long long preSum1[207], preSum2[207];

int main(){
    cin >> n;
    for(int i = 1; i <= n; i++) cin >> a[i];
    a[0] = a[n + 1] = 1;// a[-1] = 1 // 一共 3 个哨兵
    // 预处理
    dp[0][1][1] = 1;
```

```
    for(int i = 1; i <= 200; i++) preSum1[i] = preSum2[i] = 1;
    for(int i = 1; i <= n + 1; i++) {
        int s = 1, t = 200;
        if(a[i]) s = t = a[i];
        for(int j = s; j <= t; j++) {
            dp[i][j][0] = (preSum2[200] - preSum2[j]) % mod;
            dp[i][j][1] = (dp[i - 1][j][0] + dp[i - 1][j][1] + dp[i - 1][j][2]) % mod;
            dp[i][j][2] = preSum1[j - 1] % mod;
        }
        // 更新前缀和
        for(int j = 1; j <= 200; j++) {
            preSum1[j] = preSum1[j - 1] + dp[i][j][0] + dp[i][j][1] + dp[i][j][2];
            preSum2[j] = preSum2[j - 1] + dp[i][j][0] + dp[i][j][1];
        }

    }
    cout << (dp[n + 1][1][0]+dp[n + 1][1][1]) % mod << endl;
    return 0;
}
```

2、双色塔-测试开发

现在有红，绿两种颜色的石头，现在我们需要用这两种石头搭建一个塔，塔需要满足如下三个条件：

1．第 1 层应该包含 1 块石头，第 2 层应该包含两块，第 i 层需要包含 i 块石头。

2．同一层的石头应该是同一个颜色（红或绿）。

3．塔的层数尽可能多。问在满足上面三个条件的前提下，有多少种不同的建造塔的方案，当塔中任意一个对应位置的石头颜色不同，我们就认为这两个方案不相同。石头可以不用完。

输入描述:

输入仅包含两个正整数，分别表示红和绿砖块的数量 a，b（0<=a,b<=2*10^5,a+b>=1）。

输出描述:

输出和仅包含一个正整数，表示不同的方案数对 1000000007 取模的结果。

输入例子 1:

4 6

输出例子 1:

2

题目详解

```
#include <bits/stdc++.h>
using namespace std;
const int MOD = 1000000007;
const int N = 200007;
//滚动数组
//dp[level&1][j]表示前 level 层放 j 个石头
//（个数较少那个颜色的石头，假定为绿色）
//这样在两种颜色石头不均匀时
//可以优化复杂度
//dp 数组全部更新的最坏复杂度为
//O(level*min(a,b))
//(level 可计算出最大约为 400sqrt(5))
//所以差不多是 O(10^8)的复杂度
int dp[2][N + 5];

void solve(int a, int b) {
    if(a==0||b==0) {
        cout<<1<<endl;
        return;
    }
    memset(dp, 0, sizeof(dp));
    int level = sqrt(2 * (a + b));
    if (a > b) swap(a, b);
    dp[1][0] = dp[1][1] = 1;
    int sum = 1, lower, upper;
    int cur, last;
```

```
    for (int i = 2; i <= level; i++) {
        sum += i;
        //前 i 层最多要放的绿石个数（最多 sum 个，但不能超过绿石总个数 a）
        int tmp_upper = min(sum, a);
        //前 i 层最少要放的绿石个数
        //最坏情况，前 i 层将 b 个红石放完(sum-b>0)，那么前 i 层至少放 sum-b
个绿石
        //若 sum-b<0，说明前 i 层最少可以不放红石，即 dp[i][j]中 j 从 0 开始更新
        //综上，j 的更新范围下界为 max(sum - b, 0)
        int tmp_lower = max(sum - b, 0);
        //下界大于上界，说明红石放完，绿石总数量也不够放下一层了，那么就是
最高层了
        //停止更新
        if (tmp_lower > tmp_upper) break;
        upper = tmp_upper;
        lower = tmp_lower;
        cur = i & 1;
        last = !cur;
        //dp[i-1][j]表示第 i 层放红石
        // dp[i-1][j-i]表示第 i 层放绿石（那么前 i 层至少 i 个绿石（即 j>=i））
        //转移方程：dp[i][j] = dp[i-1][j] + dp[i-1][j-i](j>=i)
        //前 i 层的绿石少于 j，说明第 i 层只可能放了红石
        //dp[i][j] = dp[i-1][j](j<i)
        for (int j = lower; j < i; j++) dp[cur][j] = dp[last][j];
        for (int j = i; j <= upper; j++) {
            dp[cur][j] = (dp[last][j] + dp[last][j - i]) % MOD;
        }
    }
    int ans = 0;
    for (int j = lower; j <= upper; j++) {
        ans = (ans + dp[cur][j]) % MOD;
    }
    cout << ans << endl;
}
```

```
int main() {
    int a, b;
    while (cin >> a >> b) {
        solve(a, b);
    }
    return 0;
}
```

3、有序图-Andrioid 开发

现在给出一张含有 n 个点的有向无环图，我们称这张图是“有序图”当且仅当这个图满足以下条件：

1. 存在一个 1-n 数字的全排列 p，并令 i 号结点的权值为 p[i]。
2. 如果图中存在 u 号结点到 v 号结点的一条边，则 u 号结点的权值要小于 v 号结点的权值。显然可能有多个序列满足条件，请你找出字典序最小的全排列 p，使得这个图成为有序图。

输入描述:

第一行包含两个正整数 n，m,分别表示图上结点是数量和有向边的数量。（1≤n,m≤100000） 接下来 m 行每行有两个正整数 u，v，表示存在一条从 u 结点到 v 结点的有向边。

输出描述:

输出一个字典序最小的，1-n 的全排列，使得这张图是有序图，元素中间使用空格隔开。

输入例子 1:

3 3
1 2
1 3
3 2

输出例子 1:

1 3 2

题目详解

```
/*拓扑排序*/
```

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

const int M = 100003;
vector<int> E[M];
int indegree[M];

int main() {
    int n, m, u, v;
    cin >> n >> m;
    int r[n + 1];
    for (int i = 0; i < m; i++) {
        cin >> u >> v;
        E[v].push_back(u);
        indegree[u]++;
    }
    priority_queue<int> q;
    for (int i = 1; i <= n; i++)
        if (!indegree[i])
            q.push(i);
    int k = n;
    while (!q.empty()) {
        int p = q.top();
        q.pop();
        r[p] = k--;
        for (int i = 0; i < E[p].size(); i++) {
            v = E[p][i];
            indegree[v]--;
            if (indegree[v] == 0)
                q.push(v);
        }
    }
    for (int i = 1; i <= n; i++) {
        if (i == n)
```

```
                cout << r[i] << endl;
            else
                cout << r[i] << " ";
        }
        return 0;
}
```

5.7 网易 2020 校园招聘真题（点击立刻刷题）

1、小易的英语软件-算法

小易是班级的英语课代表，他开发了一款软件开处理他的工作。小易的软件有一个神奇的功能，能够通过一个百分数来反应你的成绩在班上的位置。“成绩超过班级 ...% 的同学”。

设这个百分数为 p，考了 s 分，则可以通过以下式子计算得出 p：

p = （分数不超过 s 的人数-1）÷班级总人数×100%

突然一天的英语考试之后，软件突然罢工了，这可忙坏了小易。成绩输入这些对于字写得又快又好的小易当然没有问题，但是计算这些百分数……这庞大的数据量吓坏了他。

于是他来找到你，希望他编一个程序模拟这个软件：给出班级人数 n，以及每个人的成绩，请求出某几位同学的百分数。

输入描述:

第一行一个整数 n，表示班级人数。

第二行共 n 个自然数，第 i 个数表示第 i 位同学的成绩 a_i。

第三行一个整数 q，表示询问的次数。

接下来 q 行，每行一个数 x，表示询问 x 位同学的百分数。

$1 <= n, q <= 10000, 0 <= a_i <= 150$

输出描述:

输出应有 q 行，每行一个百分数，对应每一次的询问。

为了方便，不需要输出百分号，只需要输出百分号前的数字即可。四舍五入保留六位小数即可。

输入例子 1:

3

100 98 87

3
1
2
3

输出例子 1:

66.666667
33.333333
0.000000

题目详解

```
/*
这题最关键的一点在于，怎么求出分数不超过 s 的人数。
有多种不同的做法。这里介绍一种比较好理解的：桶排+前缀和。
分数只有 151 种，考虑对于每一种分数开一个桶 buc_i，表示分数等于 i 的人数，
输入的时候就可以搞定。
现在分数不超过 s 的人数，就是求：buc_0 + buc_1 + ... + buc_s
这个用前缀和可以快速求出。
最后要求输出百分数，这个在纸上推一下就好了。
*/

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

const int MAXN = 10000;

int n, m;
int a[MAXN+1];
int buc[151], pre[151];

int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &a[i]);
        ++buc[ a[i] ];
    }
```

```
    pre[0] = buc[0];
    for (int i = 1; i <= 150; ++i)
        pre[i] = pre[i - 1] + buc[i];
    scanf("%d", &m);
    for (int i = 1; i <= m; ++i) {
        int x;
        scanf("%d", &x);
        double ans = 100.0 * (pre[ a[x] ] - 1) / n;
        printf("%.6lf\n", ans);
    }

    return 0;
}
```

2、跳柱子-算法

小易有 n 根柱子，第 i 根柱子的高度为h_i。一开始小易站在第一根柱子上。小易能从第 i 根柱子跳到第 j 根柱子，当且仅当$h_j \le h_i$且$1 \le j-i \le k$。其中 k 为指定的一个数字。

另外小易拥有一次释放超能力的机会。这个超能力能让小易从柱子跳到任意满足 i 的柱子$1 \le j-i \le k$而无视柱子 j 高度的限制。

现在小易想知道，小易是否能到达第根柱子。

输入描述:

第一行数据组数 T

对于每组数据，第一行数字 n, k，接下来一行 n 个数字表示h_i.

输出描述:

对于每组数据，输出 YES 或 NO

输入例子 1:

1

5 3

6 2 4 3 8

输出例子 1:

YES

输入例子 2:

1

5 2

1 8 2 3 4

输出例子 2:

NO

题目详解

```
/*
动态规划
*/

#include <bits/stdc++.h>

#define MAX_N 1000 + 100
int T;
int N, k;
int H[MAX_N];
int dp[MAX_N][3];
int main() {
	scanf("%d", &T);
	while (T--) {
		memset(dp, 0, sizeof(dp));
		dp[0][1] = 1;
		dp[0][0] = 1;
		scanf("%d%d", &N, &k);
		for (int i = 0; i < N; i++) {
			scanf("%d", &H[i]);
		}

		for (int i = 1; i < N; i++) {
			for (int j = 1; j <= k; j++) {
				if (i - j >= 0 && H[i] <= H[i - j]) {
					dp[i][0] |= dp[i - j][0];
					dp[i][1] |= dp[i - j][1];
```

```
                    }

                    dp[i][1] |= dp[i - j][0];
                }
            }
            if (dp[N - 1][0] || dp[N - 1][1])
                printf("YES\n");
            else
                printf("NO\n");
        }
        return 0;
}
```

3、序列维护-数据分析

小易在维护数据的时候遇到一个需求，具体来说小易有一系列数据，这些数据了构成一个长度为 n 的数字序列，接下来小易会在这个序列上进行 q 次操作。

每次操作有一个查询的数字 x，小易需要将序列数据中所有大于等于 x 的数字都减一，并输出在本次操作中有多少个数字被减一了。

小易犯了难，希望你能帮帮他。

输入描述:

第一行 n, q，表示数字个数和操作个数。

接下来一行 n 个数表示初始的数字。

接下来 q 行，每行一个数，表示指定的数字 x。

$1 \le n, q \le 200000$, $1 \le a_i, x \le n$

输出描述:

对于每个询问，输出一个数字表示答案

输入例子 1:

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

输出例子 1:

1

2

4

输入例子 2:

3 2

1 2 3

3

3

输出例子 2:

1

0

题目详解

```
/*
注意到如果 a_i <= a_j，那么不管怎么操作都有 a_i <= a_j。
于是把数字从小到大排序后，每次询问就二分出对应的位置，并把后面的数字全部减一。具体实现可以用树状数组或者更高级的数据结构。
*/

#include <bits/stdc++.h>
using namespace std;
int c[200005];
int n, q;
inline int lowbit(int x) { return x & (-x); }
void add(int x) {
    while (x <= n) {
        c[x]++;
        x += lowbit(x);
    }
}
int query(int x) {
    int ans = 0;
    while (x) {
```

```
            ans += c[x];
            x -= lowbit(x);
        }
        return ans;
}
int a[200005];
int main() {
        scanf("%d%d", &n, &q);
        for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
        sort(a + 1, a + n + 1);
        while (q--) {
            int x;
            scanf("%d", &x);
            if (x > a[n] - query(n)) {
                puts("0");
                continue;
            }
            int l = 1, r = n;
            while (l < r) {
                int mid = (l + r >> 1);
                if (a[mid] - query(mid) >= x)
                    r = mid;
                else
                    l = mid + 1;
            }
            printf("%d\n", n - l + 1);
            add(l);
        }
        return 0;
}
```

4、放置货物-深度学习系统研发

小易有一个体积巨大的货物，具体来说，是个在二维平面上占地 c*d 的货物。

小易有一个 n*m 的广场，想把货物放在这个广场上。不幸的是，广场上已经有了一些障碍物，障碍物所在的格子不能放置你的货物。小易现在想知道能否成功地放置货物。

输入描述:

第一行数字 t，表示有 t 组数据。

对于每一组数据，第一行三个数字 n,m,k，表示广场的大小和障碍物的个数。接下来 k 行，每行两个数 x,y，表示一个障碍物的坐标。

接下来一行两个数 c,d，表示货物的大小。

$1 \le n,m \le 1000$, $1 \le c \le n, 1 \le d \le m, 0 \le k \le n \times m$

输出描述:

对于每组数据，输出"YES"或者"NO"表示货物是否可以被放置。

输入例子 1:

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

输出例子 1:

```
YES
NO
```

题目详解

```
/*
枚举货物放置的左上角，剩下的只需要检查某个矩形里面有没有障碍物。 把障碍物看成 1，空格看成 0，维护二维前缀和，就可以快速查询矩形和。如果矩形和是 0 则可以放置。
*/

#include<bits/stdc++.h>
using namespace std;
int n , m , k;
```

```
int a[1005][1005];
int c , d;
void solve() {
    scanf("%d%d%d",&n,&m,&k);memset(a , 0 , sizeof(a)) ;
    for(int i = 1;i <= k;i++) {
        int x , y;scanf("%d%d",&x,&y);
        a[x][y] = 1;
    }
    scanf("%d%d",&c,&d);
    for(int i = 1;i <= n;i++) {
        for(int j = 1;j <= m;j++) {
            a[i][j] += a[i-1][j] + a[i][j-1] - a[i-1][j-1] ;
        }
    }
    for(int i = 1;i <= n - c + 1 ; i ++) {
        for(int j = 1;j <= m - d + 1 ; j++) {
            int s = a[i+c-1][j+d-1] - a[i-1][j+d-1] - a[i+c-1][j-1] + a[i-1][j-1] ;
            if(!s) {
                puts("YES") ;return ;
            }
        }
    }
    puts("NO") ; return ;
}
int main() {
    int t ;scanf("%d",&t) ;
    while(t--) solve();
}
```

5.8 网易游戏（互娱）2020 校园招聘研发真题（点击立刻刷题）

1、水平线-研发

伞屋国是一个以太阳能为主要发电手段的国家，因此他们国家中有着非常多的太阳能基站，链接着的基站会组合成一个发电集群。但是不幸的是伞屋国不时会遭遇滔天的洪水，当洪水淹没基站时，基站只能停止发电，同时被迫断开与相邻基站的链接。你作为伞屋国的洪水观察员，有着这样的任务：在洪水到来时，计算出发电集群被洪水淹没后被拆分成了多少个集群。

由于远古的宇宙战争的原因，伞屋文明是一个二维世界里的文明，所以你可以这样理解发电基站的位置与他们的链接关系：给你一个一维数组 a，长度为 n，表示了 n 个基站的位置高度信息。数组的第 i 个元素 a[i]表示第 i 个基站的海拔高度是 a[i],而下标相邻的基站才相邻并且建立链接，即 x 号基站与 x-1 号基站、x+1 号基站相邻。特别的，1 号基站仅与 2 号相邻，而 n 号基站仅与 n-1 号基站相邻。当一场海拔高度为 y 的洪水到来时，海拔高度小于等于 y 的基站都会被认为需要停止发电，同时断开与相邻基站的链接。

输入描述:

1、每个输入数据包含一个测试点。
2、第一行为一个正整数 n，表示发电基站的个数 (0<n<= 200000)
3、接下来一行有 n 个空格隔开的数字，表示 n 个基站的海拔高度，第 i 个数字 a[i]即为第 i 个基站的海拔高度，对于任意的 i(1<=i<=n),有(0 <= a[i]<2^31-1)
4、接下来一行有一个正整数 q(0<q<= 200000)，表示接下来有 q 场洪水
5、接下来一行有 q 个整数，第 j 个整数 y[j]表示第 j 场洪水的海拔为 y[j],对于任意的 j(1<=j<=n),有(-2^31<y[j]<2^31-1)

输出描述:

输出 q 行，每行一个整数，第 j 行的整数 ans 表示在第 j 场洪水中，发电基站会被分割成 ans 个集群。标准答案保证最后一个整数后也有换行。

输入例子 1:

```
10
6 12 20 14 15 15 7 19 18 13
6
```

15 23 19 1 17 24

输出例子 1:

2

0

1

1

2

0

题目详解

```
/*
如果洪水是逐渐上涨，那么每个发电机只会被淹没掉一次，沉下去的发电机不可能再浮起来。则我们对洪水高度和发电机都排序，每次不同的洪水高度，看成是洪水上涨又新淹没了一堆发电机，将其记录在答案数组中，最后一并输出即可。离线算法的典型运用。
注意边界条件。我们可以假设 0 号和 n+1 号发电机一直在水下即可。
*/

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

struct nums {
	int idx;
	int h;

	bool operator<(nums other) { return h < other.h; }
};
const int maxn = 200006;
nums a[maxn], b[maxn];
int ans[maxn];
int sunk[maxn];

int n, m;

int main() {
```

```
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        a[i].idx = i;
        scanf("%d", &a[i].h);
    }
    sort(a + 1, a + n + 1);
    scanf("%d", &m);
    for (int i = 1; i <= m; i++) {
        b[i].idx = i;
        scanf("%d", &b[i].h);
    }
    sort(b + 1, b + m + 1);
    int last = 1, ret = 1;
    sunk[0] = sunk[n + 1] = 1;
    for (int i = 1; i <= m; i++) {
        while (last <= n && a[last].h <= b[i].h) {
            int idx = a[last].idx;
            sunk[idx] = 1;
            int adj = sunk[idx - 1] + sunk[idx + 1];
            if (adj == 0)
                ret++;
            else if (adj == 2)
                ret--;
            last++;
        }
        ans[b[i].idx] = ret;
    }
    for (int i = 1; i <= m; i++) {
        printf("%d\n", ans[i]);
    }
    return 0;
}
```

2、游泳池-研发

小明作为一个游泳池管理员，以玩弄给水管和排水管为乐，也因此产生了很多数学题考验小朋友。

现在小明想把这个行动升级，考验一下程序员，做了一个自动装置来控制给水管和排水管。在开始时，给水管和排水管都是打开状态的，并且游泳池里没有水。在自动装置的作用下，每经过 t1 分钟，给水管的状态都会改变，即从打开状态变为关闭状态或从关闭状态变为打开状态，而同时每经过 t2 分钟，排水管的状态也会改变。当给水管打开时，给水管每分钟会向游泳池里注入 m1 升水；当排水管打开时，排水管每分钟会把游泳池里水排走 m2 升；当给水管和排水管同时打开时，游泳池的水量变化为每分钟(m1-m2)升。当然泳池的水量不能变为负数，同时泳池也有个最大容量 m，水量不能超过 m 升。那么经过 t 分钟后，游泳池里有多少升水？

输入描述:

1、输入第一行为一个正整数 T，表示有 T 组数据。

2、每组数据的为一行包含六个整数，分别表示 m，t，m1，t1，m2，t2。

3、数据范围：

对于所有数据，满足 1<=T<=10，1<=m<=100000，1<=t<=86400，

1<=m1,m2<=100，1<=t1,t2<=10。

输出描述:

对于每一个数据，输出一行，包括一个整数，为在 t 分钟后游泳池中的水量。

输入例子 1:

```
5
10 2 1 5 2 5
10 2 10 5 2 5
10 2 3 5 2 5
100 100 3 4 4 3
10000 1000 10 5 5 3
```

输出例子 1:

```
0
10
2
```

3

2495

题目详解

```
/*
模拟
*/

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

int T;
int m, t, m1, t1, m2, t2;
int p1, p2;

int main(){
	scanf("%d", &T);
	while(T--){
		scanf("%d%d%d%d%d%d",&m, &t, &m1, &t1, &m2, &t2);
		p1 = p2 = 0;
		int ans = 0;
		for(int i = 0; i < t; i++){
			if(i % t1 == 0)p1 = 1 - p1;
			if(i % t2 == 0)p2 = 1 - p2;
			ans = ans + p1 * m1 - p2 * m2;
			if(ans < 0)ans = 0;
			if(ans > m)ans = m;
		}
		printf("%d\n",ans);
	}
}
```

5.9 拼多多 2019 校园招聘真题（点击立刻刷题）

1、选靓号

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。

输入例子 1:

6 5

787585

输出例子 1:

4

777577

例子说明 1:

花费为 4 的方案有两种：777577 与 777775，前者字典序更小。

题目详解

```
/*
堆
枚举可以改为的靓号和花费，使用优先队列(堆)维护。
*/
```

```
#include <bits/stdc++.h>
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());
}
```

2、种树

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"-"。

输入例子 1:

3

4 2 1

输出例子 1:

1 2 1 2 1 3 1

题目详解

```
/*
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。
*/

#include <bits/stdc++.h>
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、回合制游戏

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

输入例子 1:

13

3

5

输出例子 1:

5

题目详解

```
/*
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。
*/

#include <bits/stdc++.h>

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;
}
```

5.10 vivo2020 校园招聘真题（点击立刻刷题）

1、运矿石

小 v 最近在玩一款挖矿的游戏，该游戏介绍如下：
1、每次可以挖到多个矿石，每个矿石的重量都不一样，挖矿结束后需要通过一款平衡矿车运送下山；
2、平衡矿车有左右 2 个车厢，中间只有 1 个车轮沿着导轨滑到山下，且矿车只有在 2 个车厢重量完全相等且矿石数量相差不超过 1 个的情况下才能成功运送矿石，否则在转弯时可能出现侧翻。

假设小 v 挖到了 n（n<100）个矿石，每个矿石重量不超过 100，为了确保一次性将 n 个矿石都运送出去，一旦矿车的车厢重量不一样就需要购买配重砝码。请问小 v 每次最少需要购买多少重量的砝码呢？（假设车厢足够放下这些矿石和砝码，砝码重量任选）

输入描述:
输入 n 个正整数表示每个矿石的重量

输出描述:
输出一个正整数表示最少需要购买的砝码重量

输入例子 1:
3 7 4 11 8 10

输出例子 1:
1

例子说明 1:
小 v 可以将重量 3,7 和 11 的矿石放到左车厢，重量 4，8 和 10 放到右车厢，然后购买重量为 1 的砝码放到左车厢

题目详解

```
/*
有限制的背包问题
*/

int solution(int n, int weight[]) {
    int s = 0;
    for (int i = 0; i < n; ++i) {
```

```
            s += weight[i];
        }
        int S = s >> 1;
        int N = (n + 1) >> 1;
        vector<vector<int> > dp(S + 1, vector<int>(N + 1, -INF));
        for (int i = 0; i <= S; ++i) dp[i][0] = 0;
        for (int i = 0; i < n; ++i) {
            auto dp1 = dp;
            for (int j = weight[i]; j <= S; ++j) {
                for (int k = 1; k <= N; ++k) {
                    dp1[j][k] = max(dp1[j][k], dp[j - weight[i]][k - 1] + weight[i]);
                }
            }
            swap(dp1, dp);
        }
        if (n & 1) {
            return s - 2 * max(dp[S][N], dp[S][N - 1]);
        }
        return s - 2 * dp[S][N];
}
```

2、报数

今年 7 月份 vivo 迎来了新入职的大学生，现在需要为每个新同事分配一个工号。人力资源部同事小 v 设计了一个方法为每个人进行排序并分配最终的工号，具体规则是： 将 N（N<10000）个人排成一排，从第 1 个人开始报数；如果报数是 M 的倍数就出列，报到队尾后则回到队头继续报，直到所有人都出列；

最后按照出列顺序为每个人依次分配工号。请你使用自己擅长的编程语言帮助小 v 实现此方法。

输入描述：

输入 2 个正整数，空格分隔，第一个代表人数 N，第二个代表 M：

输出描述：

输出一个 int 数组，每个数据表示原来在队列中的位置用空格隔开，表示出列顺序：

输入例子 1:

6 3

输出例子 1:

3 6 4 2 5 1

例子说明 1:

6 个人排成一排，原始位置编号即为 1-6。最终输出 3 6 4 2 5 1 表示的是原来编号为 3 的第一个出列，编号为 1 的最后一个出列。

题目详解

```
# 约瑟夫环
x = input()
N = int(x.split()[0])
M = int(x.split()[1])
p = [i + 1 for i in range(N)]
count = 1
index = 0
while len(p) > 0:
    if index >= len(p):
        index = index % len(p)
    if count % M == 0:
        print(p[index], end=" ")
        p[index] = 0
        p.remove(p[index])
        count += 1
        continue
    count += 1
    index += 1
```

3、服务部署

小 v 是公司的运维工程师，现有一个有关应用程序部署的任务如下：

1、一台服务器的磁盘空间、内存是固定的，现在有 N 个应用程序要部署；
2、每个应用程序所需要的磁盘、内存不同，每个应用程序允许访问的用户数也不同，且同一个应用程序不能在一台服务器上部署多个。

对于一台服务器而言，如何组合部署应用程序能够使得单台服务器允许访问的用户数最多？

输入描述:

1、输入包括三个参数，空格分隔，分别表示服务器的磁盘大小、内存大小，以及应用程序列表；
2、其中第三个参数即应用程序列表，表述方式为：多个应用程序信息之间用 '#' 分隔，每个应用程序的信息包括 ',' 分隔的部署所需磁盘空间、内存、允许访问的用户量三个数字；比如 50,20,2000 表示部署该应用程序需要 50G 磁盘空间，20G 内存，允许访问的用户数是 2000

输出描述:

单台服务器能承载的最大用户数

输入例子 1:

15 10 5,1,1000#2,3,3000#5,2,15000#10,4,16000

输出例子 1:

31000

例子说明 1:

组合部署服务 5,2,15000、10,4,16000 ，可以让单台服务器承载最大用户数 31000

题目详解

```
/*
背包问题，递归求解
*/

#include <bits/stdc++.h>
using namespace std;
int getCountOfApp(const char* input) {
    if (NULL == input) {
        return 0;
    }
```

```
    int cnt = 0;
    for (int i = 0; input[i] != 0; ++i) {
        if (input[i] == ',') {
            ++cnt;
        }
    }
    return cnt / 2;
}

// id start from 0
int getPositionOfApp(const char* input, const int id) {
    int cntOfComma = id * 2 + 1;
    int i = 0;
    for (; input[i] != 0 && cntOfComma != 0; ++i) {
        if (input[i] == ',') {
            --cntOfComma;
        }
    }
    while (input[--i] != ' ' && input[i] != '#')
        ;
    return i + 1;
}

#define APP_MAX 1000
#define DP_MAX 2048
int disks[APP_MAX];
int mems[APP_MAX];
int users[APP_MAX];
int dp[DP_MAX * DP_MAX];

int solve_res(int disks[], int mems[], int users[], int N, int DISK, int MEM, int index) {
    if (index < 0 || DISK <= 0 || MEM <= 0) {
        return 0;
    }
```

```
    int res = solve_res(disks, mems, users, N, DISK, MEM, index - 1);
    if (disks[index] <= DISK && mems[index] <= MEM) {
        res = res > (users[index] +
                    solve_res(disks, mems, users, N, DISK - disks[index], MEM -
mems[index], index - 1))
                ? res
                : (users[index] +
                    solve_res(disks, mems, users, N, DISK - disks[index], MEM -
mems[index], index - 1));
    }
    return res;
}

int solution(const char* input) {
    const int countOfApp = getCountOfApp(input);
    if (0 == countOfApp) {
        return 0;
    }
    int res = 0;

    int disk = 0;
    int mem = 0;

    sscanf(input, "%d %d", &disk, &mem);

    for (int i = 0; i < countOfApp; ++i) {
        const int pos = getPositionOfApp(input, i);
        sscanf(input + pos, "%d,%d,%d", &disks[i], &mems[i], &users[i]);
    }
    res = solve_res(disks, mems, users, countOfApp, disk, mem, countOfApp - 1);
    // TODO Write your code here!

    return res;
}
```

```
int main(int argc, char* args[]) {
    char input[10000];
    cin.getline(input, 10000);
    cout << solution(input) << endl;
}
```

5.11 360 公司校园招聘真题（点击立刻刷题）

1、圈地运动

圈地运动，就是用很多木棍摆在地上组成一个面积大于 0 的多边形～

小明喜欢圈地运动，于是他需要去小红店里面买一些木棍，期望圈出一块地来。小红想挑战一下小明，所以给小明设置了一些障碍。障碍分别是：

1. 如果小明要买第 i 块木棍的话，他就必须把前 i-1 块木棍都买下来。
2. 买了的木棍都必须用在圈地运动中。

那么请问小明最少买多少根木棍，才能使得木棍围成的图形是个面积大于 0 多边形呢？

输入描述:

第一行一个数 n，表示木棍个数。

第二行 n 个数，第 i 个数表示第 i 个木棍的长度 ai

1<=n<=10000

1<=ai<=10000

输出描述:

输出一个数，表示最少需要的木棍个数，如果无解输出-1

输入例子 1:

3

6 8 10

输出例子 1:

3

例子说明 1:

用三根 6，8，10 的木棍可以组成一个直角三角形的图形。

题目详解

```
/*
思路是 n-1 边的和大于最长边，就能组成封闭多边形;
*/

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

int main() {
    int NumOfStick = 0;   //木棍个数
    cin >> NumOfStick;
    int *a = new int[NumOfStick]();
    for (int i = 0; i < NumOfStick; i++) {
        cin >> a[i];
    }
    int Max_line = a[0] > a[1] ? a[0] : a[1];
    int Sum = a[0] + a[1];
    int flag = 0;
    for (int d = 2; d <= NumOfStick; d++) {
        if (a[d] > Max_line)   //更新最大的边
        {
            Max_line = a[d];
        }
        Sum += a[d];   //前 d 根边的所有和
        if (Sum - Max_line > Max_line) {
            cout << d + 1 << endl;
            flag = 1;
            return 0;
        }
    }
    if (flag == 0) {
        cout << -1 << endl;
    }
    delete[] a;
```

```
    return 0;
}
```

2、看花

小明有一个花园，花园里面一共有 m 朵花，对于每一朵花，都是不一样的，小明用 1～m 中的一个整数表示每一朵花。

他很喜欢去看这些花，有一天他看了 n 次，并将 n 次他看花的种类是什么按照时间顺序记录下来。 记录用 a[i]表示，表示第 i 次他看了 a[i]这朵花。

小红很好奇，她有 Q 个问题,问[l,r]的时间内，小明一共看了多少朵不同的花儿，小明因为在忙着欣赏他的花儿，所以想请你帮他回答这些问题。

输入描述:

1、输入两个数 n,m;(1<=n<=2000,1<=m<=100);分别表示 n 次看花，m 表示一共有 m 朵花儿。

2、接下来输入 n 个数 a[1]~a[n]，a[i]表示第 i 次，小明看的花的种类;

3、输入一个数 Q(1<=Q<=1000000);表示小红的问题数量。

4、输入 Q 行 每行两个数 l,r(1<=l<=r<=n);表示小红想知道在第 l 次到第 r 次，小明一共看了多少不同的花儿。

输出描述:

一共 Q 行

每一行输出一个数 表示小明在[l,r]的时间内看了多少种花。

输入例子 1:

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

输出例子 1:

```
3
2
3
```

题目详解

```
/*
先对询问进行离线。 然后按照右端点从小到大排序。
树状数组统计答案，对于第一次出现直接加 1
第二次出现，先减去上一次的影响，在加 1
然后枚举右端点，有询问的话，拿出左端点放到树状数组进行查询
整体复杂度 O(n*logn)
*/

#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 5;
int a[N];
struct Bit {
    int c[N];
    int n;
    void init() { memset(c, 0, sizeof(c)); }
    int lowbits(int x) { return x & -x;}
    void add(int i, int v) {
        for(; i <= n; i += lowbits(i)) c[i] += v;
    }
    int get(int i) {
        int ans = 0;
        if(i <= 0) return 0;
        for(; i > 0; i -= lowbits(i)) ans += c[i];
        return ans;
    }
}B;
struct node {
    int l, r, id;
    bool friend operator < (node a, node b) {
        return a.r < b.r;
    }
}b[N];
```

```
int id[N], ans[N];
int main()
{
	int n, m, k;
	scanf("%d %d", &n, &k);
	B.init();
	B.n = n;
	memset(id, -1, sizeof(id));
	for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
	scanf("%d", &m);
	for(int i = 1; i <= m; i++) {
		int l, r;
		scanf("%d %d", &l, &r);
		b[i] = {l, r, i};
	}
	sort(b + 1, b + m + 1);
	int j = 1, sum = 0;
	for(int i = 1; i <= n; i++) {
		if(id[a[i]] != -1) B.add(id[a[i]], -1), sum--;
		id[a[i]] = i; sum++;
		B.add(i, 1);
		while(j <= m && b[j].r == i) {
			int tmp = sum - B.get(b[j].l-1);
			ans[b[j].id] = tmp;
			j++;
		}
	}
	for(int i = 1; i <= m; i++) {
		printf("%d\n", ans[i]);
	}
	return 0;
}
```

3、魔法排列

众所周知，集合{1 2 3 … N}有N!种不同的排列，假设第 i 个排列为P_i且$P_{i,j}$是该排列的第 j 个数。将 N 个点放置在 x 轴上，第 i 个点的坐标为x_i且所有点的坐标两两不同。对于每个排列（以P_i为例），可以将其视为对上述 N 个点的一种遍历顺序，即从第$P_{i,1}$个点出发，沿直线距离到达第$P_{i,2}$个点，再沿直线距离到达第$P_{i,3}$个点，以此类推，最后到达第$P_{i,N}$个点，将该路线的总长度定义为$L(P_i)$，那么所有N!种路线的总长度之和是多少，即$L(P_1)+L(P_2)+L(P_3)+...+L(P_{N!})$的结果是多少？

输入描述:

第一行包含一个整数 N，$1 \leqslant N \leqslant 10^5$。

第二行包含 N 个空格隔开的整数x_1到x_N，$0 \leqslant x_1 < x_2 < x_3 < ... < x_N \leqslant 10^9$。

输出描述:

输出$L(P_1)+L(P_2)+L(P_3)+...+L(P_{N!})$对$10^9+7$取模后的结果。

输入例子 1:

3

0 1 3

输出例子 1:

24

例子说明 1:

P_1={1 2 3}，P_2={1 3 2}，P_3={2 1 3}，P_4={2 3 1}，P_5={3 1 2}，P_6={3 2 1}；

$L(P_1)=3$，$L(P_2)=5$，$L(P_3)=4$，$L(P_4)=5$，$L(P_5)=4$，$L(P_6)=3$。

题目详解

```
import java.util.Scanner;


/**
 * 思路: 这道题的主要思路是排列组合
 *
 *       例如有 4 个数[1, 2, 3, 4], 那么所有的排列组合为:
 *       1 2 3 4 --- 1 2 4 3 --- 1 3 2 4 --- 1 3 4 2 --- 1 4 2 3 --- 1 4 3 2
 *       2 1 3 4 --- 2 1 4 3 --- 2 3 1 4 --- 2 3 4 1 --- 2 4 1 3 --- 2 4 3 1
 *       3 1 2 4 --- 3 1 4 2 --- 3 2 1 4 --- 3 2 4 1 --- 3 4 1 2 --- 3 4 2 1
```

```
 *        4 1 2 3 --- 4 1 3 2 --- 4 2 1 3 --- 4 2 3 1 --- 4 3 1 2 --- 4 3 2 1
 *
 *        可以发现，任意连个数相邻的情况都为 6，例如有 6 个 1 2
 *        这个 6 是怎么来的呢？可以把 1 2 捆绑在一起，那么就变成了如何将 1 2 3 4 放在三个坑中
 *        得到 A(3, 3) = 6 种
 *
 *        其他的方式也一样，另外 1 2 和 2 1 是两种不同的组合，在计算时只需要乘以 2 即可
 */
public class Test {

    public static long mod = 1000000007;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
        int[] x = new int[N];
        for (int i = 0; i < N; i++) {
            x[i] = scanner.nextInt();
        }
        int n = factorial(x.length - 1);

        /**
         * 时间复杂度 O(N)
         *
         * 思想:
         *        例如 1 2 3 4 要计算 len(1, 4) + len(2, 4) + len(3, 4)
         *        = (4 - 1) + (4 - 2) + (4 - 3)
         *        = 4 * 3 - (1 + 2 + 3)
         */
        long sum = 0, tem = 0, a;
        for (int i = 1; i < x.length; i++) {
            tem = (tem + x[i - 1]) % mod;
```

```
                // 这里不能直接写成 a = x[i] * i;
                // 因为 a 的类型为 long，而 x[i] 和 i 都是 int 类型
                // 两个 int 类型的数相乘就有导致溢出
                a = x[i] * (long)i;
                sum = (sum + a - tem) % mod;
            }
            sum = (sum << 1) % mod;
            sum = (sum * n) % mod;
            System.out.println(sum);
        }

        // 计算 N!
        public static int factorial (int n) {
            long result = 1;
            for (int i = 1; i <= n; i++) {
                /**
                 * 这里可以这样计算的原因是
                 *          (a * b) % c = ((a % c) * (b % c)) % c
                 */
                result = (result * i) % mod;
            }
            return (int) result;
        }
}
```

5.12 深信服 2019 校园招聘真题（点击立刻刷题）

1、长方体的摆放

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长方体摆成高为 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，表示要摆放的高度

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

输入例子 1:

10 5

6 7

输出例子 1:

1

题目详解

```
/*
设 dp[i][j]为第 i 层，高度为 j 的方案数，那么第 i+1 层的高度为 j+x j+y j+z 的方案数都等于第 i 层的方案数，所以可以得出递推式为:
dp[i+1][j+x]+=dp[i][j]
dp[i+1][j+y]+=dp[i][j]
dp[i+1][j+z]+=dp[i][j]
*/

import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int x=sc.nextInt();
         int y=sc.nextInt();
         int z=sc.nextInt();
        int[][] dp=new int[200][400];
        dp[1][x]=1;
        dp[1][y]=1;
        dp[1][z]=1;
        for(int i=2;i<200;i++){
            for(int j=0;j<300;j++){
```

```
                    dp[i][j+x]+=dp[i-1][j];
                    dp[i][j+y]+=dp[i-1][j];
                    dp[i][j+z]+=dp[i-1][j];
                }
            }
            int sum=0;
            for(int i=0;i<200;i++){
                sum+=dp[i][n];
            }
            System.out.println(sum);
            sc.close();
        }
}
```

2、IP 段合并

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)
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。

输入例子 1:

4
3 8
3 7
4 6
7 9

输出例子 1:

3 9

题目详解

```
/*
先排序，再合并
*/

#include <bits/stdc++.h>
using namespace std;
bool cmp(pair<int, int> p1, pair<int, int>p2) {
    if (p1.first != p2.first)
        return p1.first < p2.first;
    else
        return p1.second <= p2.second;
}
int main()
{
    int N;
    cin >> N;
    vector<pair<int, int>> data;
    vector<pair<int, int>> res;
    while (N--) {
        int tmp1, tmp2;
        cin >> tmp1;
        cin >> tmp2;
        data.push_back(make_pair(tmp1, tmp2));
    }
    sort(data.begin(), data.end(), cmp);

    res.push_back(data[0]);
    for (int i = 1; i < data.size(); i++) {
        pair<int,int> tmp=res.back();
        if (tmp.second < data[i].first) {
            res.push_back(data[i]);
```

```
            }
            else{
                res.pop_back();
                res.push_back(make_pair(tmp.first, max(data[i].second, tmp.second)));
            }
        }
        for (auto c : res) {
            cout << c.first << " " << c.second<<endl;
        }
}
```

3、查找重复序列

已知某序列 S=<e1, e2, …, en>，序列中的元素类型为整数（en <= 2^10），序列的长度为可变长度。

现在有若干序列 S1，S2, …, Sn，现在要求设计一种算法，找出这些重复的序列。输出重复序列的序号，如果有多组重复，需全部输出。

所有序列中的数字个数加起来，小于 1000000，序列个数小于 10000 个。

例如现有 3 个序列

S1=<65, 43, 177, 655>

S2=<1, 2, 3, 4, 5, 6, 7>

S3=<65, 43, 177, 655, 3>

这时序列无重复。又如

S1=<65, 43, 177, 655, 3>

S2=<1, 2, 3, 4, 5, 6, 7>

S3=<65, 43, 177, 655, 3>

这时序列有重复。

输入描述:

第一行为一个正整数 N，N>=1 且 N<10000

接下来为 2*N 数据，每两行表示一个序列，序列的第一行为序列长度 L，第二行为序列的数字，一共 L 个

输出描述:

重复序列的序号，每一行 X 个数字，表示一组相同的序列，这一组相同序列共有 X 个，输出这 X 个序列的序号

输入例子 1:

```
11
10
794 472 991 500 615 872 518 827 673 203
1
427
7
367 718 202 187 683 321 831
10
1023 78 310 816 158 500 518 705 553 470
8
205 190 306 492 166 49 791 961
6
665 211 1009 614 15 683
2
195 946
3
678 198 495
8
205
190 306 492 166 49 791 961
5
83 74 1023 453 692
2 176 157
```

输出例子 1:

```
4 8
```

题目详解

```
/*
hash 比较
*/

#include <bits/stdc++.h>
using namespace std;
const int N = 10010;
map<LL,int> mp;
vector<int> v[N];
```

```
int main()
{
	int n;
	scanf("%d", &n);
	for(int i = 0; i < n; i++) {
		scanf("%d", &m);
		LL tmp = 0;
		for(int j = 0; j < m; j++)    tmp = tmp*2333+readl()+131;
		auto it = mp.find(tmp);
		if(it==mp.end()) mp[tmp] = i;
		else v[it->second].pb(i);
	}
	bool f = false;
	for(int i = 0; i < n; i++) {
		if(v[i].size()>0){
			f = true;
			printf("%d ",i);
			for(auto e:v[i]) printf("%d ",e);
			printf("\n");
		}
	}
	if(!f) puts("no");
	return 0;
}
```

2021届实习资源群

已更新500+实习岗位 快人一步拿offer

无需转发分享

扫码回复【学校&岗位】进群

回复【领资料】更能免费领取
重量级“名企求职攻略”
电子版一份
限量300本，先到先得