[初赛知识点总结]

[NOI 系列比赛 初赛的考纲知识点整理]

目录

系统基础部分

一、计算机的发展历程

1.1 世界第一台电子计算机

1946 年，在**美国**宾夕法尼亚大学诞生了世界上第一台电子计算机 ENIAC（埃尼阿克）。

1.2 关键人物

冯·诺依曼（美籍匈牙利数学家）——现代计算机之父

他提出了**在数字计算机内部的存储器中存放程序**的概念，这是所有现代电子计算机的范式，被称为**“冯· 诺依曼结构”**，按这一结构建造的电脑称为存储程序计算机（通用计算机）。

冯· 诺依曼体系结构的核心内容是：**采用存储程序和程序控制原理。**

图灵（英国数学家）——计算机科学与人工智能之父

图灵在二战爆发后协助军方破解德国著名密码系统，他提出了**图灵机模型（一种抽象模型而非真实机器）**，用机器来模拟人们用手工进行数学运算的过程。他还提出了图灵测试的思想实验，目的在测试机器能否表现出与人等价或无法区分的智能。

1.3 发展定律

摩尔定律（由英特尔创始人之一戈登·摩尔提出）：当价格不变时，集成电路上可容纳的晶体管数目约**每隔 18 个月**便会增加一倍，性能也会增加一倍。

1.4 信息论

信息论是由**克劳德·香农**发展，用来**找出信号处理与通信操作的基本限制**，如数据压缩、可靠的存储和数据传输等。自创立以来，它已拓展应用到许多其他领域，包括统计学、自然语言处理、密码学、神经生物学、进化论和分子编码、生态学的模式选择、热力学、量子计算、模式识别、异常检测和其他形式的数据分析。

1.5 真题练习

例题(2020CSP-S)

1948 年(　) 将热力学中的熵引入信息通信领域，标志着信息论研究的开端。

A. 欧拉(Leonhard Euler)

B. 冯·诺伊曼(John von Neumann)

C. 克劳德·香农(Claude Shannon)

D. 图灵(Alan Turing)

例题(2019CSP-J)

以下哪个奖项是计算机科学领域的最高奖?(　)

A.**图灵奖**

B.鲁班奖

C.诺贝尔奖

D.普利策奖

例题(2018NOIP 提高组)

关于图灵奖的说法中，错误的是（　）

A. 图灵奖是由电气和电子工程师协会（IEEE）设立的。

B. 目前获得该奖项的华人学者只有姚期智教授一人。

C. 其名称取自计算机科学的先驱、英国科学家艾伦·麦席森·图灵。

D. 它是计算机界最负盛名、最崇高的一个奖项，有“计算机界的诺贝尔奖”之称。

例题(2017NOIP 普及组)

计算机应用的最早领域是（）

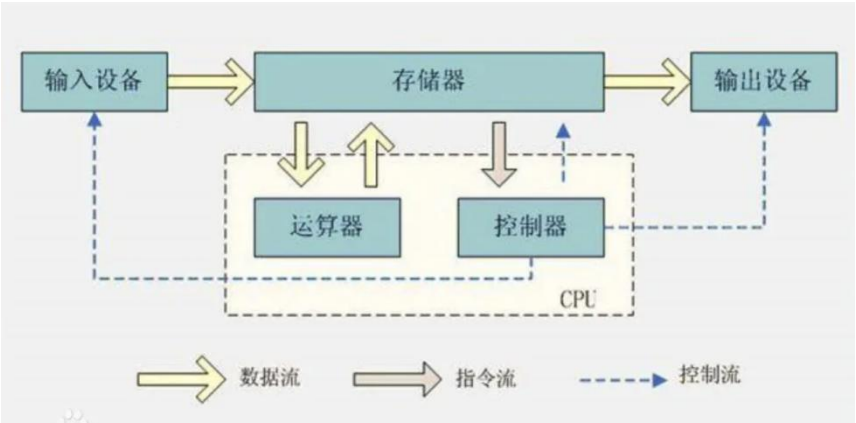
A. 数值计算　B. 人工智能　C. 机器人　D. 过程控制

二、计算机硬件

2.1 硬件体系结构

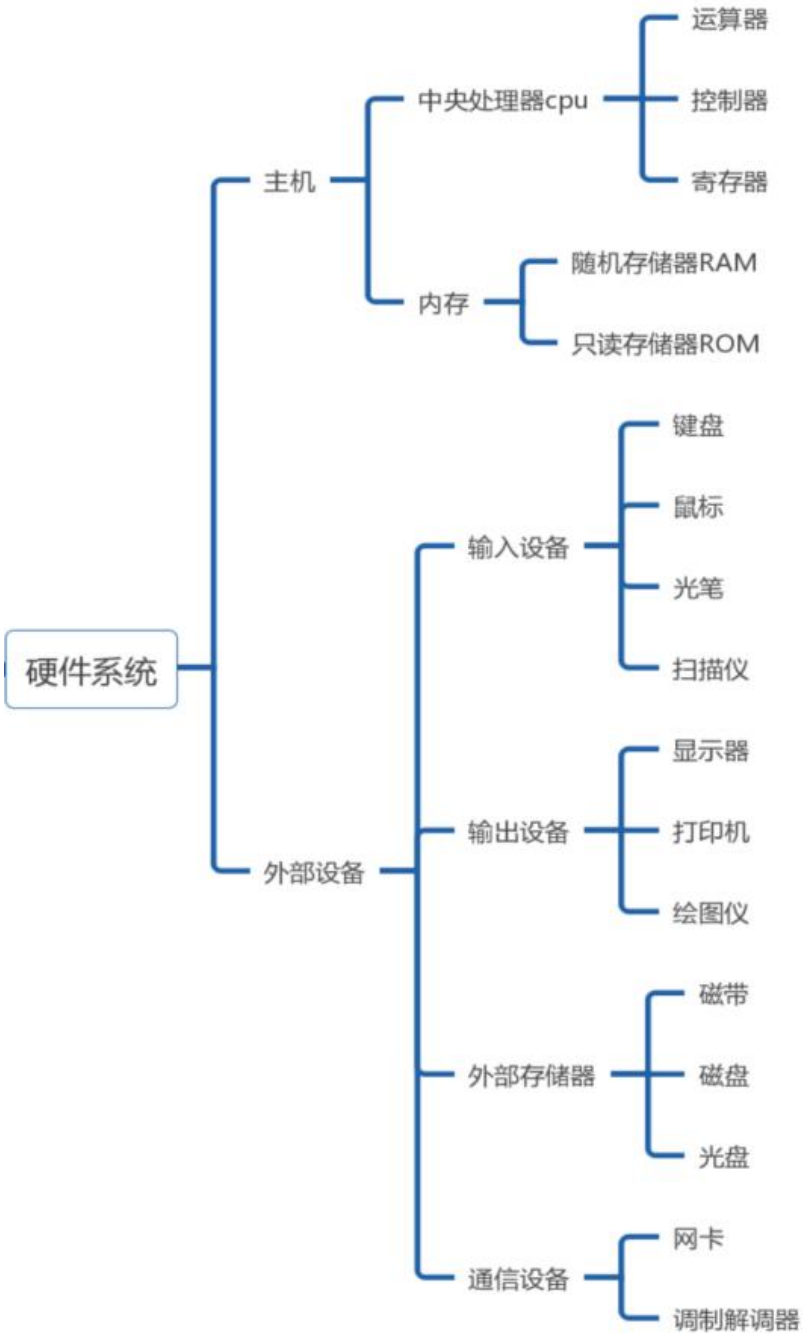
计算机硬件由**运算器、控制器、存储器、输入设备和输出设备**五大部分构成。

他们之间的逻辑关系如下图所示：



2.2 硬件设备

计算机硬件设备主要由**主机与外部设备**两大部分组成，下图所示：



- **CPU(Central Processing Unit)：中央处理器**

  ① 不同位数的处理器对应不同的寻址空间。

  ② 32 位处理器寻址空间为 2^{32} 个地址，相当于 4GB。

③ **每多 1 位，寻址空间变为原来 2 倍。**

④ 64 位处理器理论上的寻址空间为 2^{64} 个地址 ≈ 170 亿 GB。

⑤ 另外,CPU 访问不同存储器速度比较：**寄存器 > Cache（高速缓存）>内存>外存**。

- **BIOS（基本输入输出系统）**，存储在内存 ROM 中。
- **ROM (Read-only memory)：只读存储器**

  只能读取信息，保存厂家写入的系统信息，**断电后所有信息不会丢失**。
- **RAM (Random Access Memory)：随机存储器**

  可读取可写入，存放运行中的程序和数据，**断电后所有信息将全部丢失**。

  每个独立单元都存在一个唯一的地址。
- **输入设备：外界向计算机传送信息的装置**

  例如：键盘，鼠标，话筒（麦克风），扫描仪，数码相机、摄像机，光笔等
- **输出设备:将计算机中的数据信息传送到外部媒介并转化成某种人们所认识的表示形式。**

  例如：显示器，打印机（针式、喷墨、激光），绘图仪，音箱，耳机等
- 输入/输出设备：磁盘驱动器，光刻机，触摸屏等

2.3 真题练习

例题（2020CSP-J）

在内存储器中每个存储单元都被赋予一个唯一的序号，称为(　　)

A.下标　**B. 地址**　C.序号　D.编号

例题(2016NOIP 普及组)

以下是 32 位计算机和 64 位计算机区别的是（）

A. 显示器不同　　B. 硬盘大小不同　**C. 寻址空间不同**　D. 输入法不同

例题(2018NOIP 普及组)

以下哪一种设备属于输出设备：（ ）

A. 扫描仪　B. 键盘　C. 鼠标　**D. 打印机**

例题(2016NOIP 提高组)

某计算机的 CPU 和内存之间的地址总线宽度是 32 位（bit），那么这台计算机最多可以使用（）的内存。

A. 2GB　**B. 4GB**　C. 8GB　D. 16GB

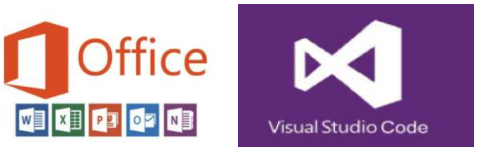
三、计算机软件

3.1 操作系统

- 微软公司 ：Windows xp / vista / 7 / 8 / 10
- IBM 公司：DOS
- 苹果公司：MAC OS
- 开源系统：Linux / Unix
- 少用的罕见系统：Solaris

3.2 常见软件

- **微软公司 ：**
  - office 办公软件系列： Word Excel Powerpoint(PPT)
  - VS 编程软件： Visual Studio



- **Adobe 公司：**
  - Adobe 全家桶：PS、AI、AE、PR 等等

3.3 编程语言

编程语言按照发展的不同阶段可以作出以下分类：

- **机器语言（二进制语言）：** 计算机能够直接识别的语言
- **汇编语言（指令语言）：** 编写源代码后，通过相应的汇编程序将它们转换成可执行的机器代码，通常被应用在底层硬件操作和高要求的程序优化场合
- **高级语言（最接近自然语言）：** 通过编译程序或解释程序翻译成机器语言。

在高级语言中，根据不同的编程特点，可以分为：

- **面向过程语言**（C, Pascal）
- **面向对象语言**（C++，C#, Java,Python 等）

在高级语言中，根据运行前的翻译方式不同，可以分为：

- **先编译后运行的编译性语言**（C,Pascal,C++,C#,Java）
- **边解释翻译边运行的解释性语言**(Python,JavaScript)

3.4 程序的三个阶段

1. 编写：在文本编辑器中完成代码源文件的过程

2. 编译：将代码源文件转换为可执行文件的过程

3. 运行：可执行文件执行的过程

3.5 真题练习

例题(2020CSP-S)

操作系统的功能是(　)

A. 负责外设与主机之间的信息交换

B. 控制和管理计算机系统的各种硬件和软件资源的使用

C. 负责诊断机器的故障

D. 将源程序编译成目标程序

例题(2016NOIP 提高组)

不是微软公司出品的软件是( )

A．Powerpoint　B．Word　C．Excel　**D．Acrobat Reader**

例题(2018NOIP 提高)

下列属于解释执行的程序设计语言是（ ）

A. C　　B. C++　　C. Pascal　　**D. Python**

例题(2017NOIP 普及组)

不属于面向对象程序设计语言的是（ ）

A. C　　B. C++　　C. Java　　D. C#

例题(2020CSP-J)

编译器的主要功能是(　)

A.将源程序翻译成机器指令代码

B.将一种高级语言翻译成另一种高级语言

C.将源程序重新组合

D.将低级语言翻译成高级语言

四、计算机网络

4.1 网络基本概念

互联网（Internet）是指 20 世纪末期兴起的电脑与电脑之间所串连成的巨大网络系統。这些网络以一些标准的网络协议相连。它是由从地方到全球范围內上千万个私人、学术界、企业和政府的网络所构成，通过电子，无线和光纤网络技术等一系列广泛的技术联系在一起。互联网承载着广泛的信息资源和服务，比如超文本文件，还有万维网（WWW）的应用、电子部件、通话，以及文件共享服务等。

4.2 网络分类

4.2.1 地理范围划分

- **广域网（WAN:　Wide Area Network）**
- 城域网（MAN: Metropolitan Area Network）
- **局域网（LAN: Local Area Network）**
- 个人区域网（PAN : Personal Area Network）

4.2.2 传输介质划分

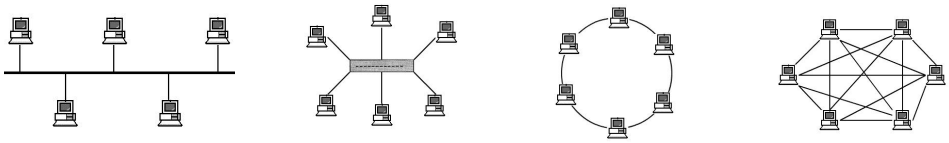
- 双绞线网
- 光纤网
- 无线网 (包含了蓝牙、WIFi、GPRS 等 通信技术)

4.2.3 数据传输速率划分

- 宽带网
- 窄带网

4.2.4 拓扑结构划分

- 总线型网络
- 星型网络
- 环形网络
- 网状网络

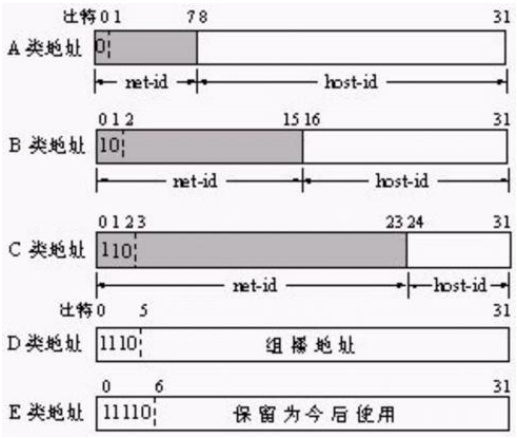


4.3 IP 地址

- IP 地址的是 **32 位二进制数，分成 4 组，每组 8 位二进制数**，每组之间用圆点隔开，通常用十进制来表示，**范围是：0.0.0.0～255.255.255.255**
- 如果把整个 Internet 网作为一个单一的网络，IP 地址就是给每个连在 Internet 网的主机分配一个全世界范围内唯一的标示符，Internet 管理委员会定义了 A、B、C、D、E 五类地址。
- IP 地址中，是由**网络编号**和**主机编号**组成，即：**IP 地址 = 网络号+主机号，**如下图。
  - **A 类地址网络号**占 8 位．开头为 0，因此第一段范围是 00000001~01111111

（1~127）其中 127 为保留地址

- **B 类地址网络号**占 16 位.开头为 10，因此第一段范围是 10000000~10111111（128~191）
- **C 类地址网络号**占 24 位.开头为 110,因此第一段范围是 11000000~110111111（192~223）

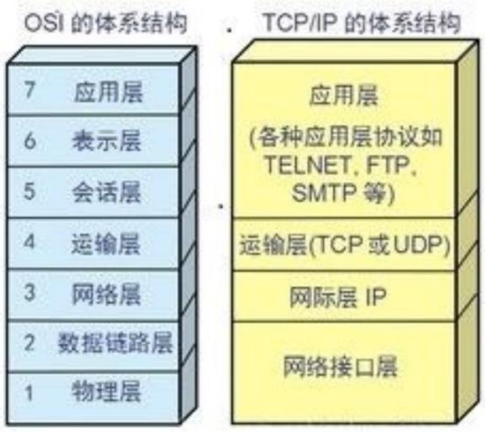


- 普通家庭用户的宽带上网是动态 IP，而是连网时由 ISP(网略服务提供商)随机给的。
- 每个网络上的计算机的 IP 不会相同。

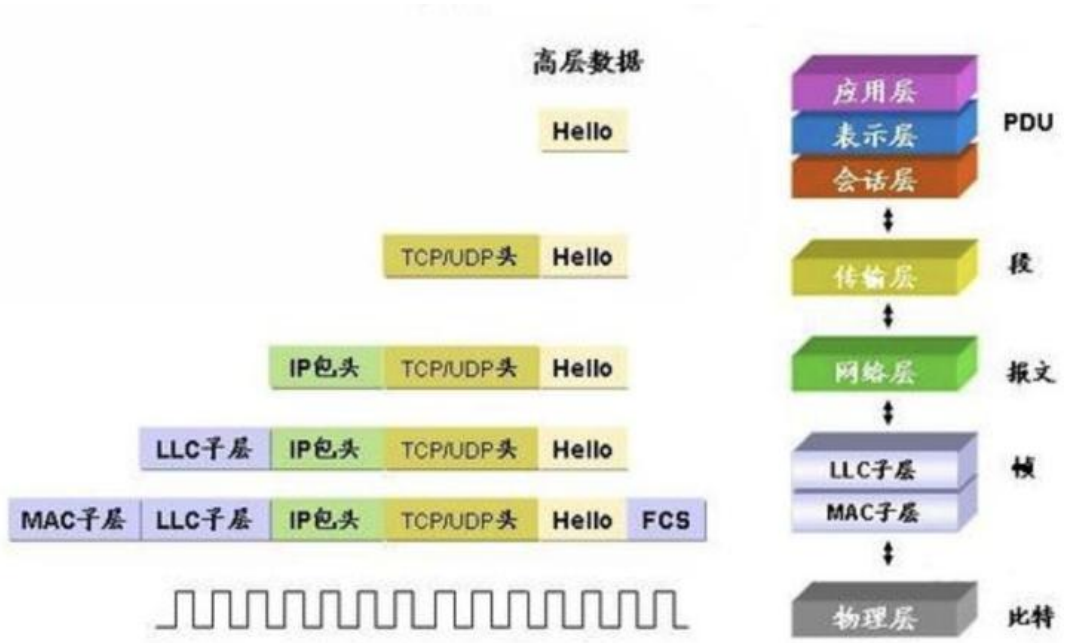
4.4 网络协议

4.4.1 协议分层

- **OSI 体系**模型（**不实用**）与 **TCP/IP 体系**模型（**更普遍**）如下图：

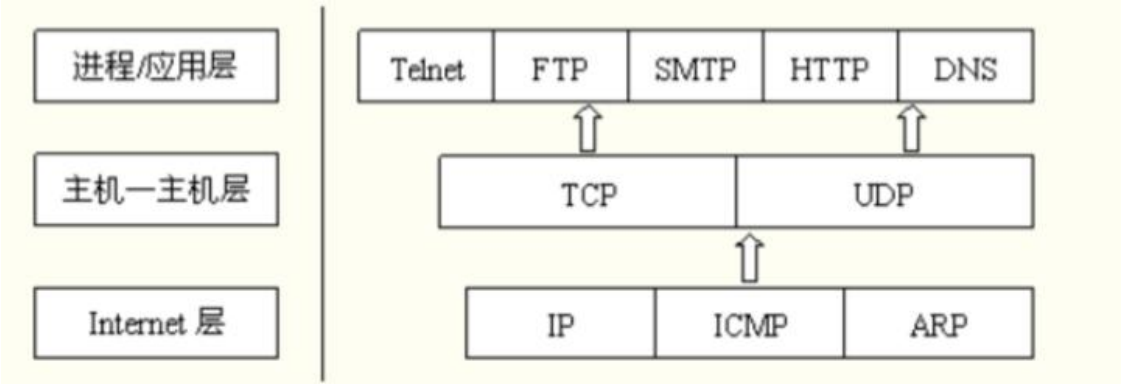


- 网络传输时每一层对数据的处理过程，见下图：



4.4.2 应用层传输协议

网络中常见的协议与层次关系，如下图：



- **IP（Internet Protocol）:**Internet 协议，负责 TCP/IP 主机间提供数据报服务，进行数据封装并产生协议头，TCP 与 UDP 协议的基础。
- **ICMP（Internet Control Message Protocol）**：Internet 控制报文协议。ICMP 协议其实是 IP 协议的的附属协议，在 ICMP 包中携带了控制信息和故障恢复信息。
- **ARP（Address Resolution Protocol）**：地址解析协议。

- **TCP（Transmission Control Protocol）：** 传输控制协议，面向连接，可靠传输。
- **UDP（User Datagram Protocol）：** 用户数据报协议，面向无连接，不可靠传输。
- **elnet (Telnet) ：** 远程登录协议，用于实现远程登录功能。
- **FTP (File Transfer Protocol) ：** 文件传输协议，用于实现交互式文件传输功能。
- **SMTP (Simple Mail Transfer Protocol) ：** 简单邮件传送协议，用于实现电子邮箱传送功能（另外邮件相关协议还有 IMAP – Internet Mail Access Protocol，交互式邮件存取协议，以及 POP3 – Post Office Protocol 3，邮局协议第三版本）。
- **HTTP (HyperText Transfer Protocol) ：** 超文本传输协议，用于实现 WWW 服务。
- **DNS (Domain Name System) ：** 域名系统，用于实现网络设备域名到 IP 地址映射的网络服务。

4.5 网络域名

4.5.1 国际域名

- com：Commercial organizations,商业组织,公司
- net：网络服务商
- top：顶级、高端、适用于任何商业 公司 个人
- org：Other organizations,非盈利组织
- gov：Governmental entities,政府部门
- edu：Educational institutions,教研机构
- int：International organizations,国际组织

4.5.2 中国域名

- cn: 中国国家顶级域名
- com.cn 中国公司和商业组织域名
- net.cn 中国网络服务机构域名
- gov.cn 中国政府机构域名
- org.cn 中国非盈利组织域名

4.6 真题练习

例题(2018NOIP 普及组)

广域网的英文缩写是（ ）

A. LAN **B．WAN** C．MAN D．LNA

例题(2016NOIP 提高组)

以下不属于无线通信技术的有（）

A. 蓝牙 B.WiFi C.GPRS **D.以太网**

例题(2017NOIP 普及组)

下列协议中与电子邮件无关的是（）

A. POP3 B. SMTP **C. WTO** D. IMAP

例题（2019CSP-J）

中国的国家顶级域名是( )

A. .cn B. .ch C. .chn D. .china

五、计算机相关知识

5.1 文件类型

- 文档类型：.txt .doc .ppt …
- 图片格式：.jpeg .jpg .png .gif …
- 音频: .mp3 .wma …
- 视频: .mp4 .wmv .avi .mpeg …

5.2 科学计数法

5.2.1 大写 E

✧ 2E+03 ----> 2×10^3

✧ 2.178E+05 ----> 2.178×10^5

5.2.2 小写 e

✧ 1e3 ----> 1×10^3

✧ 6e5+7 ----> $6 \times 10^5 + 7$

5.3 存储空间

5.3.1 存储空间单位

计算机存储数据的**最小单位**是：**bit（位/比特，只能表示位二进制数 0 或 1）**

计算机存储数据的**基本单位**是：**Byte（字节）**

转换关系如下：

- **1Byte = 8bit**
- 1KB = 1024 Byte
- 1MB = 1024KB
- 1GB = 1024MB
- 1TB = 1024GB

5.3.2 网络传输速率单位

✧ **bps** 即 bits per second，每秒传输的 bit 位数/比特数

- 注意这里的小写 b 的单位是 位 并不是 字节。

✧ **Bps** 即 Byte per second，每秒传输的字节

- 注意这里的大写 B 的单位是字节。

✧ 由于 1Byte = 8bit 转换关系如下：

- 1Kbps = 1024bps = 128 Bps
- 1Mbps = 1024Kbps = 128 KBps
- 1Gbps = 1024Mbps = 128 MBps

问题 1：200M 带宽的网络为什么速度还这么慢？

解答：通信公司提供的 200M 带宽的单位是 Mbps(注意单位中的 b 是小写的) 因此 200Mbps 等于 25MBps(25MB/s) 。因此理论的速度上限是 25MB 每秒。 况且考虑到计算机性能、网络设备质量、资源使用情况、网络高峰期、网站服务能力、线路衰耗，信号衰减等多因素的影响，实际速度因此更慢。

问题 2：5G 技术到底有多快？

解答: 5G 技术的峰值理论传输速度可达 20Gbps（ 注意单位中 b 是小写）， 因此 20Gbps = 2.5GBps（2.5GB/s ）。当然实际速度会受到各种因素的影响，但相比 4G 技术要快 10 倍以上。

5.3.3 文件空间大小计算

- **图片空间大小（单位 bit）：**
  - **= 长（length）x 宽(width) x 像素位数（bits）**
  - 其中**真彩图片的像素位数是 32 位**。
  - 举例：分辨率为 500 x 300 的 16 位图片所占空间?

    500*300*16 = 2.4e6 bit = 3e5 Byte ≈ 300Kb ≈ 0.3 Mb
- **音频空间大小（单位 bit）：**
  - **=时间(time)x 采样频率 (frequency) x采样位数(bits)x声道数(Track number)**
  - 举例：一段 2 分钟 的两声道录音，1.5KHZ 录制频率,采样位数 24 位所占空间?

    2*60*1500*24*2=8.64e6 bit=1.08e6 Byte≈1080 Kb ≈ 1Mb
- **字体空间大小（单位 bit）：**
  - **一个字体大小（单位 bit） = 点阵长 x 点阵宽**
  - 举例：存储 100 个点阵 16*16 的字模需要多少空间?

    16*16*100 = 2.56e4 bit = 3.2e3 Byte ≈ 3.2 Kb

5.4 ASCII 码

ASCII (American Standard Code for Information Interchange)美国信息交换标准代码

- 是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言。

- 它是最通用的信息交换标准。
- **从 0~127 一共可以代表 128 个字符。**
- 其中 0～31 及 127(共 33 个) 是控制字符或通信专用字符（为不可显示字符）
- 另外 32～126（ 共 95 个) 是可显示字符
    - ✧ 32 是空格 ' '
    - ✧ 48～57 为'0'到'9'十个阿拉伯数字
    - ✧ 65～90 为 26 个大写英文字母 'A'~'Z'
    - ✧ 97～122 号为 26 个小写英文字母
    - ✧ 'a'~'z'其余为一些标点符号、运算符号等。

5.5 真题练习

例题(2019CSP-J)

32 位整型变量占用(　)个字节。

A.32　　B.128　　**C.4**　　D.8

例题(2017NOIP 普及组)

计算机存储数据的基本单位是（ ）

A Bit　　**B. Byte**　　C. GB　D. KB

例题(2020CSP-J)

现有一张分辨率为 2048x1024 像素的 32 位真彩色图像。请问要存储这张图像，需要多大的存储空间?( )

A. 4MB　　**B. 8MB**　　C. 32MB　　D. 16MB

例题(2020CSP-S)

现有一段 8 分钟的视频文件，它的播放速度是每秒 24 帧图像，每帧图像是 一幅分辨率为 2048x1024 像素的 32 位真彩色图像。请问要存储这段原始无 压 缩视频，需要多大的存储空间?(　)

A.30G　　**B. 90G**　　C. 150G　　D. 450G

六、NOI 通识

6.1 知识点

- 官网网址：**http://www.noi.cn/**
- Noi 全称：**全国青少年信息学奥林匹克竞赛**
- 全国青少年信息学奥林匹克活动的主办方是：**中国计算机学会**
- CCF NOIP 复赛全国统一评测时使用的系统软件：　**NOI Linux**
- 中国计算机学会于 **1984 年**创办全国青少年计算机程序设计竞赛
- 从 **2020 年**开始，除 NOIP 以外的 NOI 系列其他赛事（包括冬令营、CTSC、APIO、NOI）将不再支持 Pascal 语言和 C 语言
- 从 **2022 年**开始，NOIP 竞赛也将不再支持 Pascal 语言。即从 NOIP2022 开始，NOI 系列的所有赛事将全部取消 Pascal 语言。
- 在无新增程序设计语言的情况下，NOI 系列赛事自 NOIP2022 开始将仅支持 C++语言。

6.2 真题练习

例题(2018NOIP普及组)

中国计算机学会于（　）年创办全国青少年计算机程序设计竞赛。

A.1983　**B. 1984** C.1985 D.1986

例题(2017NOIP普及组)

NOI 的中文意思是（ ）

A. 中国信息学联赛　**B. 全国青少年信息学奥林匹克竞赛**

C. 中国青少年信息学奥林匹克竞赛　D. 中国计算机协会

例题(2017NOIP普及组)

从（ ）年开始，NOIP 竞赛将不再支持 Pascal 语言。

A 2020　B. 2021 **C. 2022** D. 2023

例题(2016NOIP提高组)

参加 NOI 比赛，不能带入考场的有( )

A.钢笔　　B.适量的衣服　　**C.U盘**　　D.铅笔

数学专题

一、排列组合问题及计算方式

1.集合

由一个或多个确定的元素所构成的整体

1.1 集合的三个性质

确定性：一个元素 a 是否属于一个集合 A，是确定的，不存在既属于，又不属于的关系。

互异性：同一个集合内,任何两个元素均是不同的。

无序性：同一个集合内，元素是无序的，即｛a,b}与{b,a}是同一个集合。

1.2 一些概念

- **空集：**没有任何元素的集合
- **子集：**若集合 A 中的所有元素都包含在集合 B 中，则定义集合 A 是集合 B 的子集。
  - 符号为 A ⊆ B。
  - 例如{1,2} 是{1,2,3} 的子集，但{1,4} 就不是{1,2,3} 的子集。
  - 依照定义，任一个集合也是本身的子集，不考虑本身的子集称为真子集。
  - 一个集合中包含 n 个元素，那么它的所有子集（包括空子集）一共有 2^n 个
- **并集：**指两个集合所有包含的部分，符号∪。
  - 如集合{1,2,3}和集合{2,3,4}的交集为集合{1,2,3,4}。（可以理解为∨）
- **交集：** 指两个集合间相同的部分，符号∩。

- 如集合{1,2,3}和集合{2,3,4}的交集为集合{2,3}。（可以理解为∧）

- **补集：**对于一个集合 A 和全集 B 来说，A 的补集就是在全集 B 中所有不属于 A 的元素，符号 ~ 。
  - 如对于 A{1,2,3,4,5}来讲，B{1,2,3}的补集为{4,5}。
- **差集：**对于两个集合来说的，假设两个集合 A 和 B，A - B 就是属于 A 但是不属于 B 的元素的集合，符号 - 。
  - 若令 A∩B=C，那么 A-B=A-C，也可以认为是 C 在全集 A 中的补集。
  - 对于 A{1,2,3,4,5}来讲，A 对 B{1,2,3,4}的差集为{5}

1.3 基本原理

- **加法原理：**如果完成事件 A 有 n 种方法，完成事件 B 有 m 种方法，那么完成两者之一有 n+m 种方法。
- **乘法原理：**如果完成事件 A 有 n 种方法，完成事件 B 有 m 种方法，那么先完成 A 再完成 B 有 n*m 种方法。
- **容斥原理：**为了使重叠部分不被重复计算，人们研究出的一种新的计数方法
  - 容斥原理的**基本思想**是：
    - 先不考虑重叠的情况，把包含于某内容中的所有对象的数目先计算出来
    - 然后再把计数时重复计算的数目排斥出去，使得计算的结果既无遗漏又无重复
  - **特点：**把多加的减掉，把多减的加上。
  - 对有限集合 s，用|s|表示 s 的元素个数：
    - 设 A，B 为有限集合，则有|A∪B| = |A|+|B| -|A∩B|
    - 设 A，B，C 为有限集合，则有

$$|A \cup B \cup C| = |A|+|B|+|C|-|A \cap B|-|B \cap C|-|A \cap C|+|A \cap B \cap C|$$

1.4 往届真题

[CSP-J2019-T13]

一些数字可以颠倒过来看，例如 0、1、8 颠倒过来还是本身，6 颠倒过来是 9, 9 颠倒过来看还是 6,其他数字颠倒过来都不构成数字。

类似的，一些多位数也可以颠倒过来看，比如 106 颠倒过来是 901。假设某个城市的车牌只由 5 位数字组成，每一位都可以取 0 到 9。

请问这个城市最多有多少个车牌倒过来恰好还是原来的车牌？（）

A. 60　B. 125　**C. 75**　D. 100

[CSP-J2019-T12]

一副纸牌除掉大小王有 52 张牌，四种花色，每种花色 13 张。假设从这 52 张牌中随机抽取 13 张纸牌，则至少（ ）张牌的花色一致。

A. 4　B. 2　C. 3　D. 5

2.排列组合

2.1 排列

- **概念：**从 n 个不同元素中，任取 m 个元素，按照一定的顺序排成一列。
- **特点：**考虑顺序
- **公式：** $P_n^m = A_n^m = \frac{n!}{(n-m)!}$
- 全排列问题：
- n 个不同元素排列成一排，排列方法有：$P_n^n = A_n^n = n!$

2.2 组合和组合数

2.2.1 组合

- **概念：**从 n 个不同元素中，任取 m 个元素并成一组
- **特点：**不考虑顺序

2.2.2 组合数

- **概念：**从 n 个不同元素中取出 m 个元素的所有组合的个数
- **公式：** $C_n^m = C_n^{n-m} = \frac{n!}{m! * (n-m)!}$

2.3 往届真题

[CSP-J2020-T10]

5 个小朋友并排站成一列，其中有两个小朋友是双胞胎，如果要求这两个双胞胎必须相邻，则有（）种不同排列方法?

A. 48　B. 36　C. 24　D. 72

[CSP-J2020-T14]

10 个三好学生名额分配到 7 个班级，每个班级至少有一个名额，一共有（）种不同的分配方案。

A. 84　B. 72　C. 56　D. 504

[CSP-J2020-T15]

有五副不同颜色的手套（共 10 只手套，每副手套左右手各 1 只），一次性从中取 6 只手套，请问恰好能配成两副手套的不同取法有（ ）种。

A. 120　B. 180　C. 150　D. 30

[CSP-J2019-T7]

把 8 个同样的球放在 5 个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的分法？（ ）

提示：如果 8 个球都放在一个袋子里，无论是哪个袋子，都只算同一种分法。

A. 22　B. 24　**C. 18**　D. 20

二、时间天数计算问题

2.1 相关重要内容

- **闰年：366 天**
- **平年：365 天**
- **1、3、5、7、8、10、12 月：31 天**
- **2 月**
  - **闰年：29 天**
  - **平年：28 天**
- **4、6、9、11 月：30 天**

❖ 周几推算问题：

① 计算出两个日期间的总天数

② **规律：每七天一重复，**

③ 所以**总天数 % 7** 得到

2.2 往届真题

[NOIP-普及 2017-T8]

2017 年 10 月 1 日是星期日，1999 年 10 月 1 日是（）。

A. 星期三 B. 星期日 **C. 星期五** D. 星期二

三、最大公因数（最大公约数 GCD）问题

3.1 理论知识部分

- 概念：两个或多个数字共有因数中最大的一个。
- 计算最大公因数的方法（设有 a，b 两个数字）：
  - 暴力枚举法——时间长
    - 从 min(a,b)开始，倒着找到 1 为止，第一个即使 a 的因数，又是 b 的因数，就是最大公因数
  - 辗转相除法(欧几里得算法)——速度快
    - 想求 a 个 b 的最大公因数，可以换为求 b 和 r 的最大公因数
    - gcd(a,b) = gcd(b, r);
    - r 表示 a % b 的结果
    - **当 b 为 0 时，a 为最大公因数**
- **最小公倍数：a * b / gcd(a, b)**
- **最大公因数延伸的问题”**
  - **互质：**最大公因数为 1
  - **分式化为最简：**分子分母同除最大公因数
  - **计算与数字 k 互质个数：**
    ① 先求出 k 的质因数
    ② 计算 1~k 之间，k 的每个质因数的倍数个数

③ 互质个数 = k - 所有质因数的倍数个数

3.2 往届真题

[CSP-J2019-T10]

319 和 377 的最大公约数是（ ）。

A. 27 B. 33 **C. 29** D. 31

[NOIP-普及 2018-T13]

10000 以内，与 10000 互质的正整数有（ ）个。

A. 2000 **B. 4000** C. 6000 D. 8000

四、质数问题

4.1 理论知识部分

- **定义：**因数只有 1 和本身的数字
- 性质特征：
  - 0 和 1 既不是质数也不是合数
  - 100 以内的质数有 25 个
  - 1000 以内的质数有 168 个
  - 10000 以内的质数有 1229 个
- 判断一个数字是否为质数的算法时间复杂度：
  - 时间 O(n)：枚举 2~n-1 之间是否存在因数
  - 时间 O(n/2)： 枚举 2~n/2 之间是否存在因数
  - 时间 O(sqrt(n))： 枚举 2~sqrt(n)之间是否存在因数

- 区间内判断质数：
  - 埃氏筛法 O(nloglogn)：质数的倍数一定为合数
  - 欧拉筛法 O(n)：每个合数只能被最小的质数筛除

4.2 往届真题

[CSP-J2019-T9]

100 以内最大的素数是（ ）。

A. 89 **B. 97** C. 91 D. 93

五、进制转换问题

5.1 进制

- **常见进制有：**
  - 十进制(简写表示 D)：逢十进一，0，1，2，3，4，5，6，7，8，9
  - 二进制(简写表示 B)：逢二进一，0，1
  - 八进制(简写表示 O)：逢八进一，0，1，2，3，4，5，6，7
  - 十六进制(简写表示 H)：逢十六进一，0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F
- **其它进制：**
  - X 进制：允许选用的基本数字符号的个数为 X 个，分别为 0～X-1，每一个数位满 X 就向高位进一，即“逢 X 进一”。

5.2 进制转换

- **任意进制转 10 进制：**
  - **整数部分：**从个位到高位，每位依次乘进制的次方(进制数^0,进制数^1,进制数^2.....)，并求和。例如：
    - 将二进制数转换成十进制数：101 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 = 5
    - 将八进制数转换成十进制数：102 = 2 * 8^0 + 0 * 8^1 + 1 * 8^2 = 66
    - 将十六进制数转换成十进制数：102 = 2*16^0 + 0 * 16^1 + 1 * 16^2 = 258
  - **小数部分：**从小数点开始，每位依次乘进制的次方(进制数^-1,进制数^-2,进制数^-3.....)，并求和。例如：
    - 八进制转十进制 1234.56：
      - 整数部分 = 4 * 8 ^ 0 + 3 * 8 ^ 1 + 2 * 8 ^ 2 + 1 * 8 ^ 3 = 668
      - 小数部分 = 5 * 8 ^ (- 1) + 6 * 8 ^ (- 2) = 0.71875
      - 结果 = 668.71875
    - 二进制转十进制 10101.11：
      - 整数部分 = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4
      - 小数部分 = 1 * 2^-1 + 1 * 2^-2
      - 结果 = 21.75
- **十进制转 X 进制（利用短除法）：**
  - **整数部分：**除以 X 取余，逆序输出
  - **小数部分：**乘以 X 取整，顺序输出
- **几种特殊的进制转换**
  - **八进制转二进制：**1 位八进制数转 3 位二进制数

- **二进制转八进制：** 3 位二进制数转 1 位八进制数
- **十六进制转二进制：** 2 位 16 进制数转 4 位二进制数
- **二进制转十六进制：** 4 位二进制数转 1 位 16 进制数

5.3 位运算

- **二进制运算**
  - **&** 按位与：两边为 1 才为 1
  - **|** 按位或：一边为 1 就为 1
  - **~** 取反操作：颠倒结果
  - **^** 异或：相同为 0，不同为 1
  - **<<** 左移：二进制左移一位，相当于乘 2
  - **<<** 右移：二进制右移一位，相当于除 2
- **逻辑运算：**
  - 真 true，假 false
  - 非： ！ not ¬
    - ！真 --> 假
    - ！假 --> 真
  - 与： && and ∧
    - 真 && 真 --> 真
    - 真 && 假 --> 假
    - 假 && 真 --> 假
    - 假 && 假 --> 假

- 或： || or ∨
  - 真 || 真 --> 真
  - 真 || 假 --> 真
  - 假 || 真 --> 真
  - 假 || 假 --> 假

- **运算符优先级：括号 > 非 > 与 > 或**

5.4 负数的二进制

- 在二进制码中，采用**最高位是符号位**的方法来区分正负数
  - 正数的符号位为 0
  - 负数的符号位为 1。
  - 剩下的就是这个数的绝对值部分。
- 负数以**补码**形式进行存储：通过将负数转为二进制原码，再求其原码的反码，最后求得的补码即负数的二进制表示结果（后面有具体原码、反码、补码转换规则）。
  - 比如整数-1。
    - -1 的原码：10000000 00000000 00000000 00000001
    - 反码： 11111111 11111111 11111111 11111110
    - 补码： 11111111 11111111 11111111 11111111
    - 即-1 在计算机里用二进制表示结果。
- **原码、反码、补码：**
  - 计算机中的**计算都以补码**进行计算，**显示以原码进行显示**
  - **正数：**

- 正数的原码 = 反码 = 补码
- 正数的原码 = 符号位为 0 + 数字绝对值对应的二进制数

- **负数：**
  - 负数的原码 = 符号位为 1 + 数字绝对值对应的二进制数
  - 负数的反码 = 符号位不变，其余各个位取反
  - 负数的补码 = 负数的反码 + 1

5.5 往届真题

[CSP-J2020-T9]

二进制数 1011 转换成十进制数是（ ）。

A. 11 B. 10 C. 13 D. 12

[NOIP-普及 2017-T15]

十进制小数 13.375 对应的二进制数是（ ）。

A. 1101.011 B. 1011.011 C. 1101.101 D. 1010.01

[CSP-J2019-T2]

二进制数 11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑与运算的结果是（ ）。

A. 01 0010 1000 1011

B. 01 0010 1001 0011

C. 01 0010 1000 0001

D. 01 0010 1000 0011

[NOIP-普及 2018-T2]

下列四个不同进制的数中，与其它三项数值上不相等的是（ ）

A. $(269)_{16}$ B. $(617)_{10}$ C. $(1151)_{8}$ **D. $(1001101011)_{2}$**

[NOIP-普及 2018-T14]

为了统计一个非负整数的二进制形式中 1 的个数，代码如下：

```
int CountBit (int x) {
    int ret = 0;
    while (x) {
        ret ++;
        _________;
    }
    return ret;
}
```

则空格内要填入的语句是（ ）。

A. x >>= 1

B. x &= x - 1

C. x |= x >> 1

D. x <<= 1

[NOIP-普及 2017-T1]

在 8 位二进制补码中，10101011 表示的数是十进制下的（ ）。

A. 43 **B. -85** C. -43 D. -84

六、概率问题

6.1 理论知识部分

- **概率，是反映随机事件出现的可能性大小。**
- **事件：在依次实验中出现的实验结果。**
  - 必然事件：在一定条件下，必然发生的事件
  - 不可能事件：在一定条件下，必然不发生的事件
  - 确定事件：必然事件和不可能事件统称确定事件。
  - 随机事件：在一定条件下，可能发生也可能不发生的事件。
- **公式：P(A) = (A 事件包含的基本事件的个数) / （基本事件的总数）**
  - 必然事件概率为 1
  - 不可能事件概率为 0，

- 事件 B 包含事件 A：事件 A 发生，事件 B 一定发生，记作？？？
- 事件 A 与事件 B 相等：若，？，则事件 A 与 B 相等，记作 A=B

- **并(和)事件：**某事件发生，当且仅当事件 A 发生或 B 发生，记作 A∪B(或 A+B)
- **并(积)事件：**某事件发生，当且仅当事件 A 发生且 B 发生，记作 A∩B(或 AB)
- **互斥事件：**若 A∩B 为不可能事件（A∩B = ），则互斥
- **对立事件：**若 A∩B 为不可能事件，若 A∪B 为必然事件，则对立

6.2 往届真题

[NOIP-普及 2017-T19]

一家四口人，至少两个人生日属于同一月份的概率是（ ）（假定每个人生日属于每个月份的概率相同且不同人之间相互独立）。

A. 1/12　B. 1/144　**C. 41/96**　D. 3/4

七、偏奥数的数学逻辑问题

7.1 斐波那契数列

又称兔子数列，指的是：0、1、1、2、3、5、8、13、21、34.....

定义：F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2) ,(n>=2)

7.2 水仙花数

是自幂数的一种，指的是三位数的每个位上数字的 3 次幂之和等于它本身，

如 153 = 1^3 + 5^3 + 3^3

7.3 等差数列

等差数列求和公式：$S_n = na_1 + \frac{n(n-1)}{2}d$

公式描述：公式中首项为 a1，末项为 an，项数为 n，公差为 d，前 n 项和为 Sn。

7.4 等比数列

等比数列求和公式：　$S_n = na_1 \ (q = 1)$

$$S_n = a_1 * \frac{1-q^n}{1-q} = \frac{a_1 - a_n * q}{1-q} \quad (q \neq 1)$$

公式描述：公式中 a1 为首项，an 为数列第 n 项，q 为等比数列公比，Sn 为前 n 项和。

7.5 指数运算

$$a^m * a^n = a^{m+n}$$

$$(a^m)^n = a^{m*n}$$

$$(a * b)^n = a^n * b^n$$

$$a^m \div a^n = a^{m-n}$$

运算规则：

- 同底数幂相乘，底数不变，指数相加；
- 同底数幂相除，底数不变，指数相减；
- 幂的乘方，底数不变，指数相乘；
- 同指数幂相乘，指数不变，底数相乘；
- 同指数幂相除，指数不变，底数相除。

7.6 往届真题

[NOIP-普及 2018-T16]

甲乙丙丁四人在考虑周末要不要外出郊游。

已知①如果周末下雨，并且乙不去，则甲一定不去；②如果乙去，则丁一定去；③如果丙去，

则丁一定不去；④如果丁不去，而且甲不去，则丙一定不去。

如果周末丙去了，则甲_A_，乙_B_，丁_B_，周末_B_。

1.A. 去了 B. 没去

2.A. 去了 B. 没去

3.A. 下雨 B. 没下雨

4.A. 下雨 B. 没下雨

[NOIP-普及 2018-T17]

从 1 到 2018 这 2018 个数中，共有_**544**_个包含数字 8 的数。

C++编程语言部分

一、栈

1.1 概念部分

栈的特点：**后进先出**

栈顶指针	栈 zhan[]
top	E
	D
	C
	B
	A

想象往一个有底杯子里放乒乓球(球径稍小于杯径),很明显先放进去的球后出来;

初赛此考点一般考察出栈顺序：**模拟**即可.

- **进出栈都发生在栈顶的位置**
- **进栈：**

  栈顶指针向上移动一格：top ++;

  对应数组元素赋值入栈：zhan[top] = data;

- **出栈：**

  对应数组元素赋值给临时变量：tmp = zhan[top];

栈顶指针向下移动一格：top --;

- **假如从下标为 1 开始存储数据**
  - 栈空的判断：top == 0
  - 栈满的判断：top == n

1.2 重难点汇总

一般的考察点：

- 进出栈的合法顺序（看清楚入队顺序 FEDCBA,ABCDEF）
- 四个选项加一个入队的顺序，不合法。
- 栈所需要的大小问题
- 一个入队的顺序加上进出栈指令：
  - ABCDEFG
  - 进栈，进栈，出栈，出栈，进栈，进栈，出栈，进栈。
- 最后全部执行完以后：
  - 此时的栈顶元素
  - 此时栈里面还剩多少个元素
  - 整个过程所需要栈的最小容量。

例题分析：

一个栈的入栈序列为 ABCDE，则不可能的出栈序列为?（不定项选择题）

A： ECDBA

B： DCEAB

C： DECBA

D： ABCDE

E： EDCBA

此类题做题技巧 1：抓住第一个出栈的元素(可以快速排除不行的选项)：

- A 选项:第一个出栈的是 E,则 ABCD 只能按 DCBA 出 故 A 选项不行;
- B 选项:第一个出栈的是 D,则 ABC 只能按 CBA 出,E 可以插入到任意位置,故 B 选项不行;

此类题做题技巧 2：原序(ABCDE)和倒序(EDCBA)是绝对可以的,故 DE 绝对正确.

其余选项,**模拟**即可.

1.3 关于栈的简单代码操作

数组模拟:

```
#include <bits/stdc++.h>
using namespace std;
int st[10],top;//栈的声明(注意不要跟系统的 stack 重名)   栈顶下标
int main(){
    st[top++]=123;//入栈                    乒乓球放入杯子
    cout<<st[top-1]<<endl;//引用栈顶元素      杯子最上面的乒乓球
    top--;//弹出栈顶元素                    最上面的乒乓球取出来
    cout<<top<<endl;//栈大小                  杯子里乒乓球的数量
```

```
    return 0;
}
```

stl:

```
#include <bits/stdc++.h>
using namespace std;
stack<int>st;
int main(){
    st.push(123);//入栈                    乒乓球放入杯子
    cout<<st.top()<<endl;//引用栈顶元素    杯子最上面的乒乓球
    st.pop();//弹出栈顶元素                最上面的乒乓球取出来
    cout<<st.size()<<endl;//栈大小         杯子里乒乓球的数量
    cout<<st.empty()<<endl;//栈是否为空     1 为空  0 为非空
    return 0;
}
```

二、队列

2.1 基本队列概念

队列的特点:**先进先出**

想象往一个无底杯子里放乒乓球(球径稍小于杯径),很明显先放进去的球先出来(杯子一端只进另一端只出);

- 队头（队头指针 head）出队，
- 队尾（队尾指针 tail）入队（先进先出，后进后出）

- 入队，尾指针增加 1；
- 出队，头指针增加 1
- 队空的判断：head == tail

2.2 循环队列

入队：队尾指针 tail 发生改变

tail = (tail + 1) % Maxsize

出队：队首指针 head 发生改变

head = (head + 1) % Maxsize

队空条件：head == tail

队满条件：head == (tail + 1) % Maxsize

循环队列的元素个数计算：（tail － head + Maxsize）% Maxsize

2.3 关于队列的简单代码操作

数组模拟

```
#include <bits/stdc++.h>
using namespace std;
int q[100],h,t;//队列的声明 队首 队尾
int main(){
    q[t++]=123;//入队                    乒乓球从上面放入杯子
    cout<<q[h]<<endl;//引用队首元素        杯子最下面的乒乓球
    h++;//弹出队首元素                    最下面的乒乓球取出来
```

```
    cout<<t-h<<endl;//队列大小                    杯子里乒乓球的数量
    cout<<(t==h)<<endl;//队列是否为空             1 为空  0 为非空
    return 0;
}
```

stl:

```
#include <bits/stdc++.h>
using namespace std;
queue<int>q;
int main(){
    q.push(123);//入队                            乒乓球从上面放入杯子
    cout<<q.front()<<endl;//引用队首元素  杯子最下面的乒乓球
    q.pop();//弹出队首元素                        最下面的乒乓球取出来
    cout<<q.size()<<endl;//队列大小               杯子里乒乓球的数量
    cout<<q.empty()<<endl;//队列是否为空      1 为空  0 为非空
    return 0;
}
```

三、链表

3.1 概念部分

链表（Linked list）是一种常见的基础数据结构，是一种**线性表**，但是并不会按线性的顺序存储数据，而是**在每一个节点里存到下一个节点的指针**(Pointer)。

链表的特点:**不能随机访问**

3.2 单向链表

每个节点只知道两个信息:

① 我是谁;

② 我的下一个在哪;

结构体存节点:

```
struct Node {
    type data;    当前节点的数据
    struct Node *next;    下个节点的位置(下标)
} link[N];
```

link[].data	小王	小张	小红	小明	小李
link[].next	5	0	4	3	2

head=4 表示表头的下标

链表问题分析一定要自己手动画出链式结构，分析代码的合理性

保证每一个节点:

- 除了头节点以外都是其他节点的 next
- 除了尾指针以外都指向其他节点

- 所有数据在一条路线上

单向链表的遍历:

```
for (int i = head; i ; i = link[i].next) {
    cout << link[i].data;
}
```

单向链表的查找:

```
for (int i = head; link[i] != "小王"; i = link[i].next);
```

单向链表的删除:

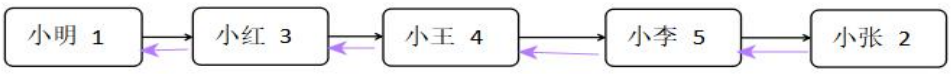
我想把 x 后面的踢了;

```
t=link[x].next;
link[x].next=link[link[x].next].next;
link[t].next=0;
```

单向链表的插入:

有一个 y 想插在 x 的后面

```
link[y].next=link[x].next;
link[x].next=y;
```



3.3 双向链表:

每个节点只知道 3 个信息:

① 我是谁;

② 我的上一个在哪;

③ 我的下一个在哪;

struct node { //**结构体存节点**

data; 当前节点的数据

pre; 上个节点的位置(下标)

next; 下个节点的位置(下标)

}link[N];

head=1 表示表头的位置(下标)

link[].data	小明	小张	小红	小王	小李
link[].next	3	0	4	5	2
link[].pre	0	5	1	3	4

双向链表的遍历:

```
for(int i=head;i;i=link[i].next){
    cout<<link[i].data;//
}
```

双向链表的查找:

```
for(i=head;link[i]!="小王";i=link[i].next);
```

双向链表的删除:

```
我想把 x 踢了
if(x==head) head = link[x].next;
link[link[x].pre].next=link[x].next;
link[link[x].next].pre=link[x].pre;
```

link[x].next=link[x].pre=0;

双向链表的左插: y 想插在 x 左边

if(head==x) head=y;

link[y].next=x;

link[y].pre=link[x].pre

link[link[x].pre].next=y;

link[x].pre=y;

双向链表的右插: y 想插在 x 右边

link[y].pre=x;

link[y].next=link[x].next;

link[link[x].next].pre=y;

link[x].next=y;

四、时间复杂度

无脑总结:**一个程序里面执行最多的那句话的执行次数.**

常见:

- 常数阶:

  for(int i=1;i<=5;i++)....

- 线性:

  for(int i=1;i<=n;i++)....

- 幂次:

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

```
for(int i=1;i<=n;i++){
    ...
}
}
```

- 指数:

```
f(n){
    f(n-1)+f(n-2);
}
```

- 平方根:

```
for(int i=1;i*i<=n;i++)....
```

- 对数:

```
for(int i=1;i<=n;i*=2)....
```

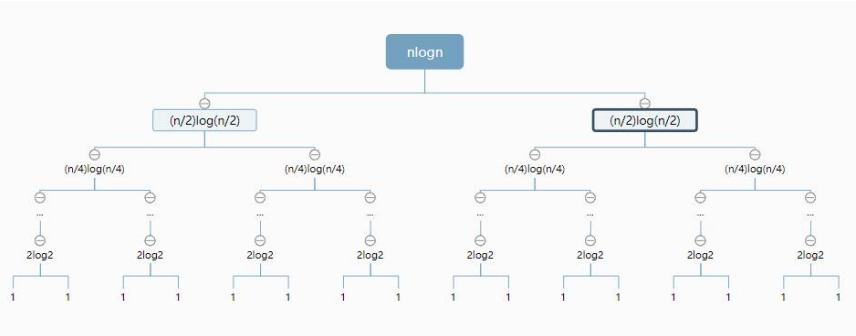
时间复杂度估算即可,一般**忽略低次项和系数;**

- 何为低次项? n 代无穷大进去,小的即为低次项

递推关系式求复杂度:

递归树法:

例:T(N)=2T(N/2)+NlogN　　T[1]=1

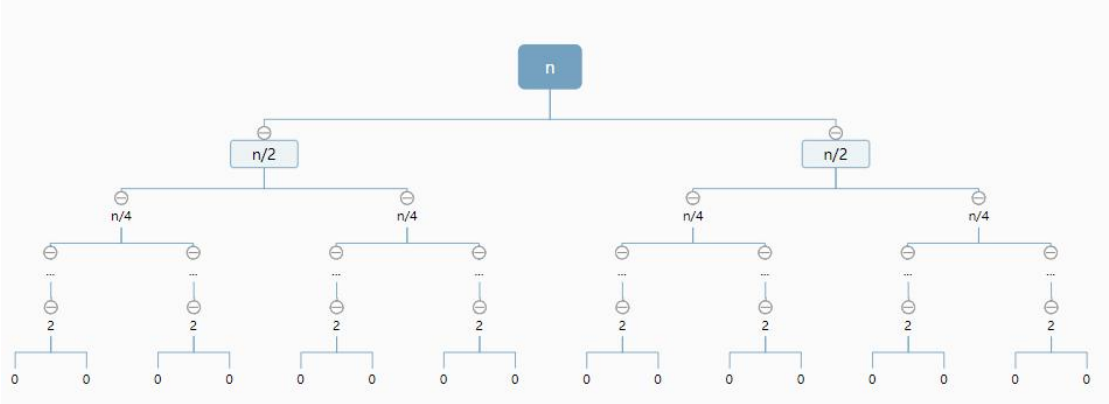


$$n(\log n + \log(\frac{n}{2}) + \log(\frac{n}{4}) + \log(\frac{n}{8}) ..... + \log(2) + 1)$$

$$n(\log n ... + 3 + 2 + 1 + 1)$$

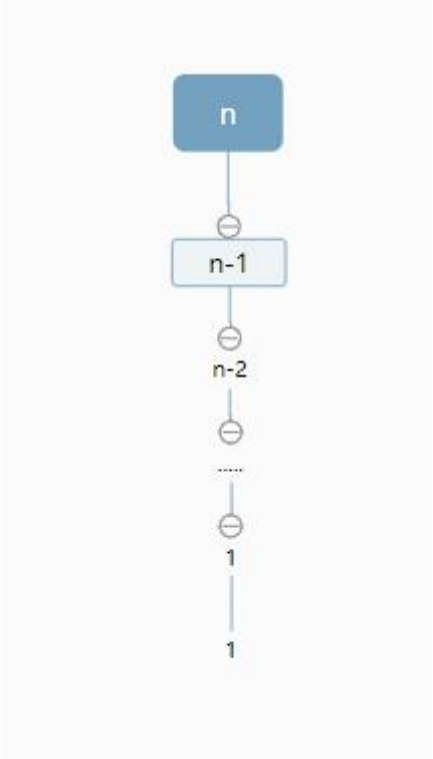
$$n(\frac{\log n + 1}{2} \log n + 1)$$

例:T(N)=2T(N/2)+N　T[1]=0



$$n \log n$$

例: T(n)=T(n−1)+n　　T[0]=1



(n+n−1+n−2+n−3...+1+1)

$\frac{1+n}{2}\mathrm{n}+1$

常见算法的时间复杂度:

① 冒泡,选择,插入排序 o(n^2);

② 归并,sort,快排 o(nlogn);

③ 二分查找(不包括 check) o(logn);

④ 桶排序,链表的遍历与查找 o(n);

常见排序算法的稳定性:

- **稳定性:**两个相同的数字，在排完序以后前后位置不变，就是稳定排序，否则就是不稳定的。
- **记忆方法:情绪不稳定，快些选一堆好友来聊天吧**
- 不稳定的排序算法:
  - 快——快速排序
  - 些——希尔排序
  - 选——选择排序
  - 堆——堆排序
- 稳定的排序算法:
  - 插入排序
  - 冒泡排序
  - 归并排序

五、树

一、树的概念和基本术语

1、线性数据结构与非线性数据结构

线性数据结构（如栈和队列）描述的是一对一的前后相继的逻辑关系，当数据之间的逻辑关系不仅仅限于一对一的线性关系，出现一对多逻辑关系（如学生会结构图）或者多对多逻辑关系（如计算机科学专业基础课程关系图）的时候，就需要用非线性数据结构来描述。

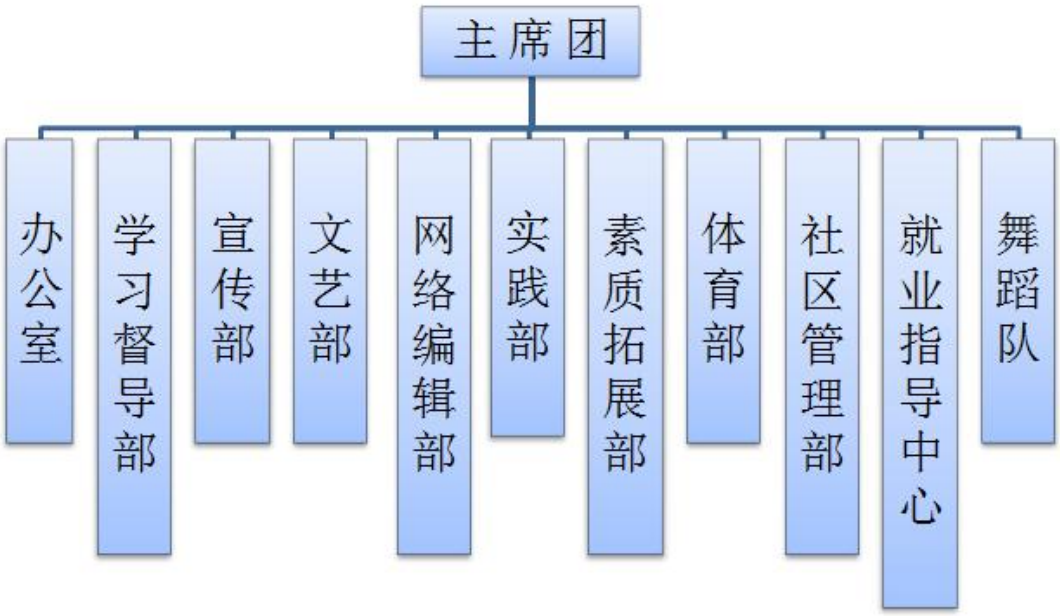


图 1-1

2、树的概念

树是一种重要的非线性数据结构，直观地看，它是数据元素（在树中称为结点）按分支关系组织起来的结构。

一棵树（tree）是由 n（n>0）个元素组成的有限集合，其中：

（1）每个元素称为结点（node）；

（2）有一个特定的结点，称为根结点或根（root）；

（3）除根结点外，其余结点被分成 m（m>=0）个互不相交的有限集合，而每个子集又都是一棵树（称为原树的子树）

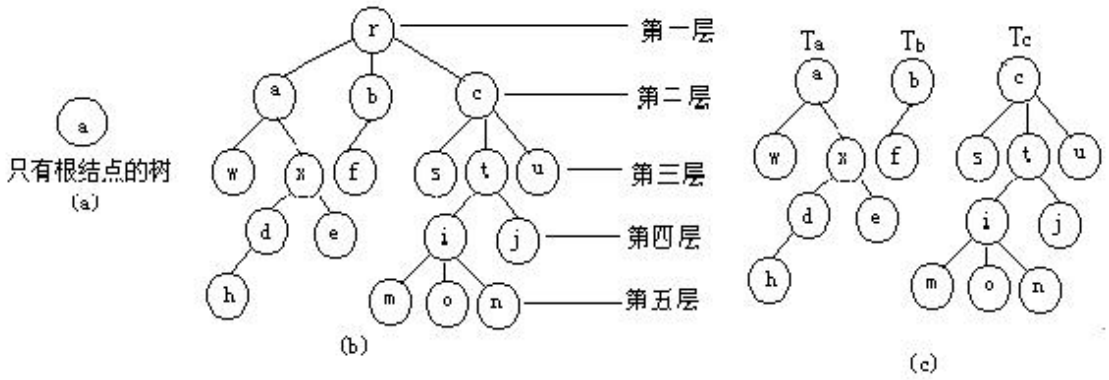


图 1-2

3、结点的分类

结点一般分成三类：

（1）根结点：没有父亲的结点。在树中有且仅有一个根结点。

（2）分支结点：除根结点外，有孩子的结点称为分支结点。b，c，x，t，d，i。分支结点亦是其子树的根；

（3）叶结点：没有孩子的结点称为树叶。w，h，e，f，s，m，o，n，j，u 为叶结点。

根结点到每一个分支结点或叶结点的路径是唯一的。从根 r 到结点 i 的唯一路径为 r-c-t-i。

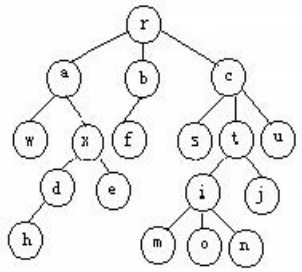


图 1-3

4、有关度的概念

（1）结点的度：一个结点的子树数目称为该结点的度。图中，结点 i 度为 3，结点 t 的度为 2，结点 b 的度为 1。显然，所有树叶的度为 0。

（2）树的度：所有结点中最大的度称为该树的度(宽度）。图中树的度为 3。

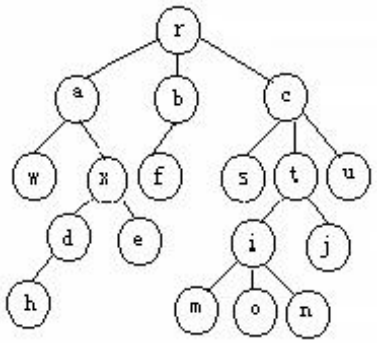


图 1-4

5、树的深度

树是分层次的。结点所在的层次是从根算起的。根结点在第一层，根的儿子在第二层，其余各层依次类推。图中的树共有五层。在树中，父结点在同一层的所有结点构成兄弟关系。树

中最大的层次称为树的深度，亦称高度。图中树的深度为 5。

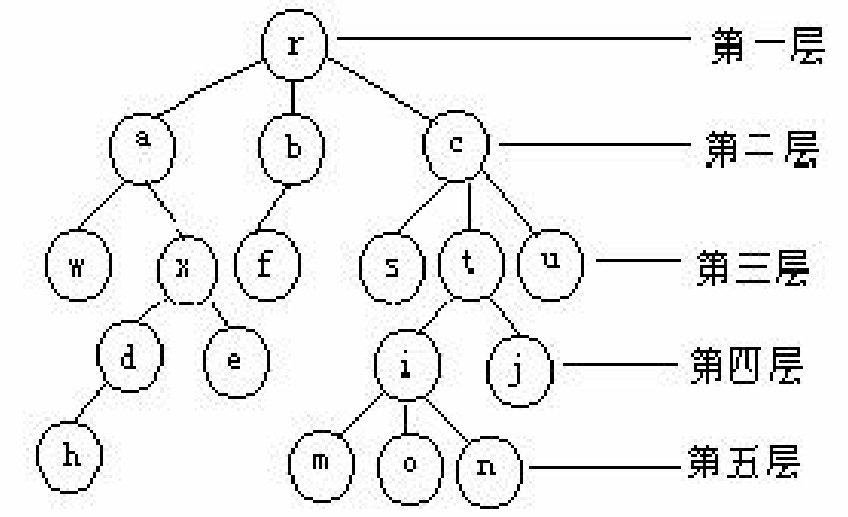


图 1-5

二、二叉树的概念

二叉树是一种很重要的非线性数据结构，它的特点是每个结点最多有两个孩子，且其子树有左右之分（有序树，次序不能任意颠倒）。

1、二叉树的定义

二叉树是以结点为元素的有限集，它或者为空，或者满足以下条件：

（1）有一个特定的结点称为根；

（2）余下的结点分为互不相交的子集 L 和 R，其中 L 是根的左子树；R 是根的右子树；L 和 R 又是二叉树；

前面引入的有关树的一些基本术语对二叉树仍然适用。下图列出二叉树的五种基本形态：

图 2-1

问：二叉树是不是树？二叉树属于树嘛？

二叉树和树是两个不同的概念

（1）树的每一个结点可以有任意多个，而二叉树中每个结点的后继不能超过 2；

（2）树的子树可以不分次序（除有序树外）；而二叉树的子树有左右之分。我们称二叉树中结点的左后件为左儿子，右后件为右儿子。

（3）二叉树可以为空，但树不能为空

2、二叉树的两种特殊形态

（2）满二叉树： 如果一棵深度为 K 的二叉树，**共有**$2^k - 1$**个结点**，即第 L 层**有**2^{L-1}**的结点**，称为满二叉树。（例如下图左边）

（2）完全二叉树：如果一棵二叉树最多只有最下面两层结点度数可以小于 2，并且最下面一层的结点都集中在该层最左边的若干位置上,则称此二叉树为完全二叉树(例如下图右边)

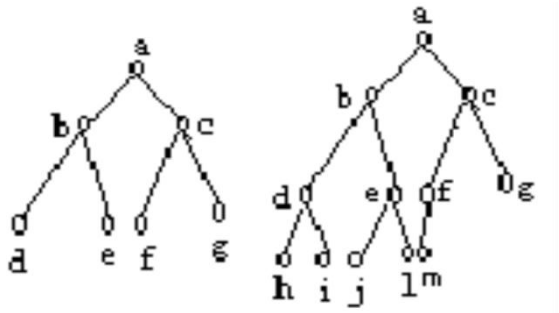


图 2-2

3、二叉树的性质(重点内容)

性质 1：在二叉树的第 i（≥1）层上，最多有2^{i-1}个结点

性质 2：在深度为 k(k≥1)的二叉树中最多有$2^k - 1$ 个结点。

性质 3：在任何二叉树中，叶子结点数总比度为 2 的结点多 1。

其中n_0、n_1、n_2代表度为 0、度为 1 的节点和度为 2 的节点个数

所有结点数目 $n = n_0 + n_1 + n_2$ （1）

所有这些分支同时又为度为 1 和度为 2 的结点发出的。再加上根结点，因此又有

$n = 1 + n_1 + 2 * n_2$ (2)

由上述两个等式得到：$n_0 = 1 + n_2$

4、真题练习

(NOIP2010,5) 如果树根算第 1 层，那么一棵 n 层的二叉树最多有（）个结点。

A. 2^n-1 B. 2^n C. 2^n+1 D. 2^(n+1)

(NOIP2011,7)如果根结点的深度记为 1，则一棵恰有 2011 个叶结点的二叉树的深度最少是（）

A. 10 B. 11 C. 12 D. 13

(NOIP2013,9) 已知一棵二叉树有 10 个节点，则其中至多有（）个节点有 2 个子节点。

A. 4 B. 5 C. 6 D. 7

(NOIP2015)2. 一棵结点数为 2015 的二叉树最多有___个叶子结点。

三、二叉树的存储结构

1、顺序存储

将每个结点依次存放在一维数组中，用数组下标指示结点编号，编号的方法是从根结点开始编号 1，然后由左而右进行连续编号。每个结点的信息包括

（1）一个数据域（data）。

（2）两个指针域，其中有父结点层号、子结点层号。

一棵完全二叉树（满二叉树）如下图所示：

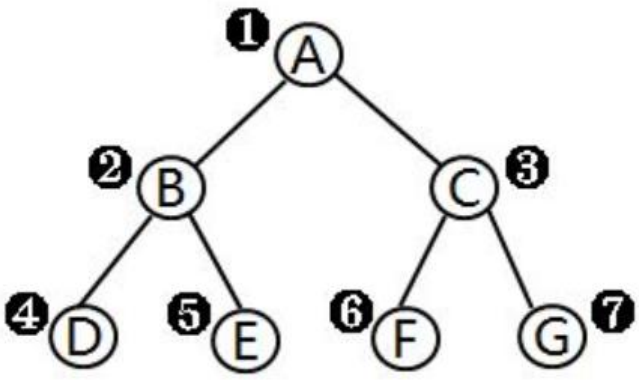


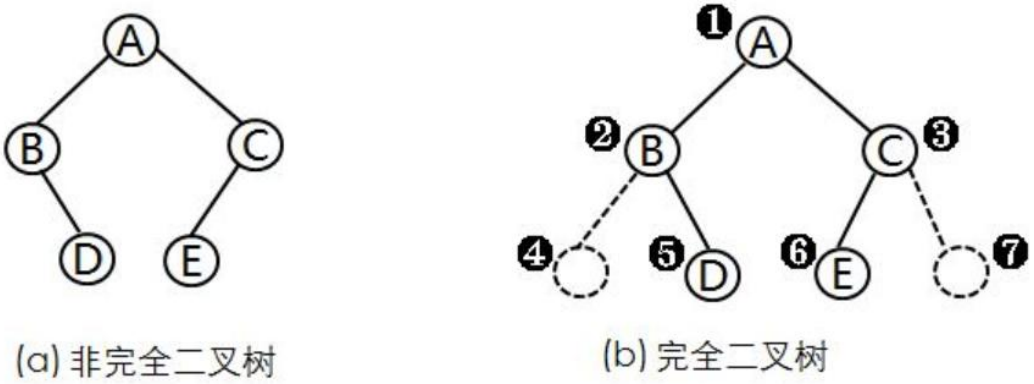
图 3-1

将这棵二叉树存入到数组中，相应的下标对应其同样的位置，如下图所示：

	A	B	C	D	E	F	G
T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]

图 3-2

满二叉树和完全二叉树一般采用顺序存储结构，如果对于一般的非完全二叉树的话，按照这种存储方式，可以通过计算下标反应出每个节点中的逻辑关系，但是这样做会造成浪费。



	A	B	C	^	D	E	^
T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]

图 3-3

从图中可以看到关系，如果根节点是从下标为 1 的地方开始的话，那么任意一个节点的父节点设置为 i 的话，他的两个孩子节点分别是：*左孩子*：$2*i$，*右孩子则为*：$2*i+1$，同样给定一个孩子节点的下标为 i 的话父节点的下标为$\lfloor \frac{i}{2} \rfloor$。

2、链式存储

建立一个单链表,链表的每一个指针变量指向一个记录,每个记录有 2 个域,data 域存储结点信息, lchild,rchild 分别存储结点的左孩子和右孩子的指针。

结点结构如下图所示：

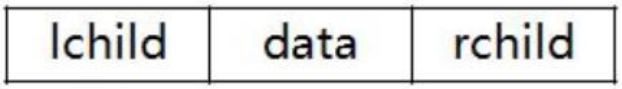


图 3-4

```
struct Tree{
    char root;
    Tree *lchild;
    Tree *rchild;
};
void Create_Tree(Tree **T){
    char c;
    cin>>c;
    if(c=='#'){
        (*T)=NULL;
    }else{
        *T=new Tree;
        (*T)->root=c;
        Create_Tree(&(*T)->lchild);
        Create_Tree(&(*T)->rchild);
    }
}
int main() {
```

```
    Tree *T;
    Create_Tree(&T);
    return 0;
}
```

3、真题练习

（NOIP2010,19）完全二叉树的顺序存储方案，是指将完全二叉树的结点从上至下、从左至右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置，则第 k 号结点的父结点如果存在的话，应当存放在数组的（ ）

A. 2k B. 2k+1 C. k/2 下取整 D. (k+1)/2 下取整

四、二叉树的遍历

二叉树的遍历是不重复地访问二叉树中的每一个结点。在访问到每个结点时，可以取出结点中的信息，或对结点作其它的处理。

如果用 L、D、R 分别表示遍历左子树、访问根结点、遍历右子树，限定先左后右的次序，三种组合

DLR、 LDR、 LRD

这三种遍历规则分别称为先（前）序遍历、中序遍历和后序遍历（以根为标准）。

1、三种遍历方式

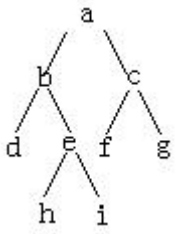


图 4-1

（1）先序遍历：访问处理根结点、前序遍历左子树、前序遍历右子树；a b d e h i c f g

（2）中序遍历： 中序遍历左子树、访问处理根结点、中序遍历右子树；d b h e i a f c g

（3）后序遍历： 后序遍历左子树、后序遍历右子树、访问处理根结点；d h i e b f g c a

2、两种遍历方式确定树结构

可以根据前序遍历或者后序遍历确定根节点，之后放入中序遍历中将树分成两部分。之后可以根据子树根的位置推导出子树的多种形态，根据提供的遍历形式确定形态。

前序遍历：根—左子树—右子树；

中序遍历：左子树—根—右子树；

后序遍历：左子树—右子树—根；

3、真题练习

（NOIP2010,17） 一棵二叉树的前序遍历序列是 ABCDEFG，后序遍历序列是 CBFEGDA，则根结点的左子树的结点个数可能是

A. 2 B. 3 C. 4 D. 5

(NOIP2012,6)如果一棵二叉树的中序遍历是 BAC，那么它的先序遍历不可能是

A. ABC　B. CBA　C. ACB　D. BAC

(NOIP2013,11) 二叉树的（ ）第一个访问的节点是根节点。

A. 先序遍历 B. 中序遍历　C. 后序遍历 D. 以上都是

（NOIP2015,16) 前序遍历序列与中序遍历序列相同的二叉树为（ ）

A. 根结点无左子树的二叉树 B. 根结点无右子树的二叉树

C. 只有根结点的二叉树或非叶子结点只有左子树的二叉树

D. 只有根结点的二叉树或非叶子结点只有右子树的二叉树

五、哈夫曼树和哈夫曼编码

1、哈夫曼树的定义

（1）概念：给定 N 个权值作为 N 个叶子结点，构造一棵二叉树，若该树的带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树(Huffman Tree)。哈夫曼树是带权路径长度最短的树，权值较大的结点离根较近。

（2）带权路径长度：从根节点到各个叶节点的路径长度与相应结点权值的乘积和叫二叉树的带权路径长度 WPL

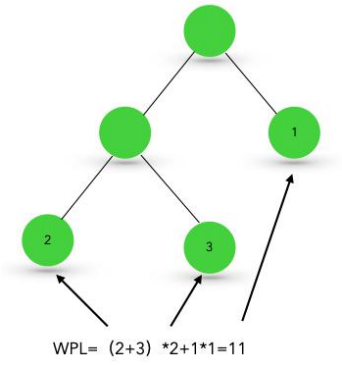


图 5-1

2、构造哈夫曼树

两个原则：

（1）权值越大的叶结点越靠近根节点

（2）权值越小的叶结点越远离根节点

过程：

（1）给定的 n 个权值{W1,W2....}构造 n 棵只有一个叶结点的二叉树，从而得到一个二叉树集合 F={T1,T2,....}。

（2）利用贪心思想从集合 F 中选取根节点的权值最小和次小的两颗二叉树作为左、右子树构造一个新的二叉树，这棵新的二叉树根节点的权值为左右子树根节点权值和。

（3）从 F 集合中删除作为左右子树的的二叉树，新建立的二叉树放入集合 F 中

（4）重复（2）、（3）两步骤，当 F 中剩余一棵二叉树时，这个二叉树就是哈夫曼树。

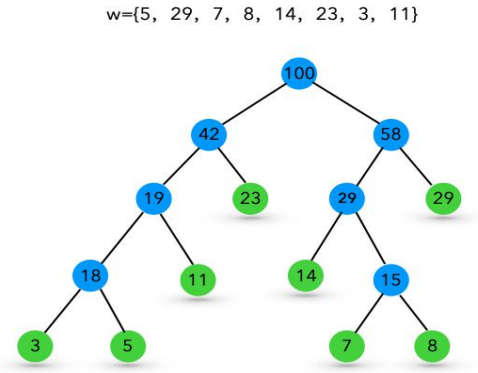


图 5-2

计算 WPL 两种方式：

（1）利用公式，去取叶子节点的和乘以路径

（2）将非叶子结点上的数字全部求和即可。

3、哈夫曼树的特点

（1）没有度为 1 的结点。

（2）有 n0 个叶子结点的哈夫曼树，结点个数有 2*n0-1 个

利用二叉树的性质：n=n0+n1+n2，由于 n1=0，n2=n0-1 所以得到 n=2n0-1

六、哈夫曼编码

1、概念

规定哈夫曼树中的左分支为 0，右分支为 1，则从根节点到每个叶结点所经过的分支对应的 0 和 1 组成的序列便为该结点对应的字符编码，这样的编码称为哈夫曼编码。

根据哈夫曼树的特点可以知道，权值越大的字符编码距离根节点越近自然的编码就越短，反之越长。

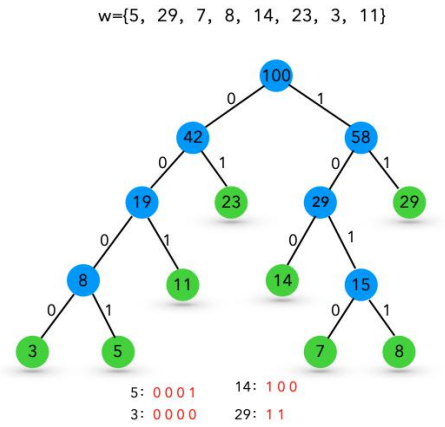


图 5-3

注：在一组字符的哈夫曼编码中，不可能出现一个字符的哈夫曼编码是另一个字符哈夫曼编码的前缀。

如：100，001，0，1 是哈夫曼编码嘛？

不是，因为 1 是 100 的前缀，0 是 001 的前缀。

七、二叉排序树

1、概念

二叉排序树（Binary Sort Tree），又称二叉查找树（Binary Search Tree），亦称二叉搜索树。是数据结构中的一类。在一般情况下，查询效率比链表结构要高。

一棵空树，或者是具有下列性质的二叉树：

（1）若左子树不空，则左子树上所有结点的值均小于或等于它的根结点的值；

（2）若右子树不空，则右子树上所有结点的值均大于它的根结点的值；

（3）左、右子树也分别为二叉排序树；

2、构建排序树

过程：拿到一组序列以后，第一个点作为根节点，依次拿元素，如果这个元素小于或等于根节点，那么放左子树；如果这个元素大于根节点，那么放在右子树；之后再去跟左子树（或右子树）的根结点比较，直到没有比较以后，作为叶子结点放好。之后进行中序遍历，得到的序列即为从小到大的顺序。

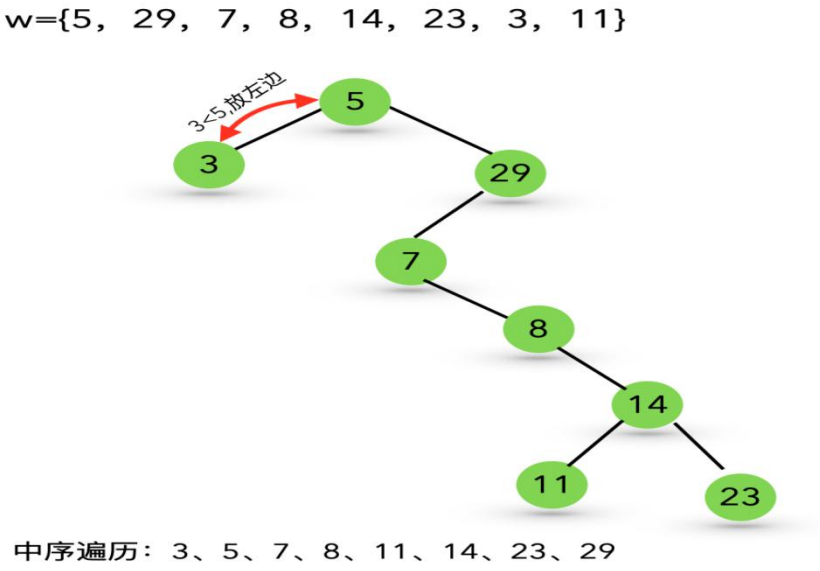


图 6-1

六、图

一、图的基本内容

1、图的概念

G(V,E) V 代表顶点（结点） E 代表边的集合

（1）有向图：图的边有方向，只能按箭头方向从一点到另一点

（2）无向图：图的边没有方向，即可以双向

（3）结点的度：无向图中与结点相连的边的数目

（4）结点的入度：有向图中，以这个点为终点的有向边的数目

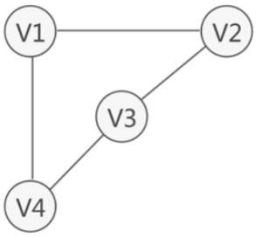
（5）结点的出度：有向图中，以这个点为起点的有向边的数目

（6）权值：边的费用（价值），可以理解为边的长度

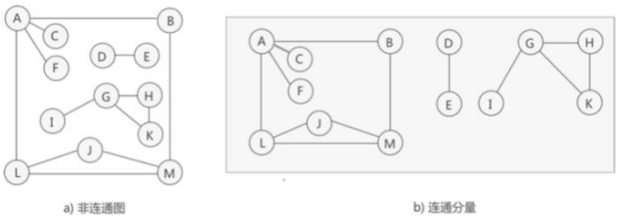
（7）在无向图中，如果从顶点 v 到顶点 w 有路径，则称顶点 v 和顶点 w 是连通的。

（8）如果无向图 G 中任意两个顶点都是连通的，则称图 G 是连通图。

（9）如果一个无向图不是连通图，则其中的每个极大连通子图称为无向图的连通分量。



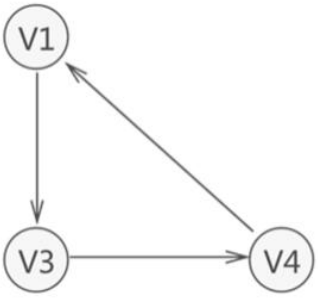
（8）连通图



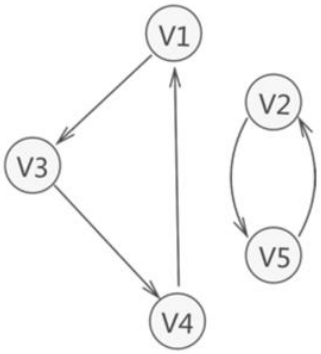
（9）连通分量

（10）如果有向图 G 中任意两个顶点间都存在有向路径（对任意两个顶点 v 和 w，既存在 v 到 w 的有向路径，也存在 w 到 v 的有向路径），则称有向图 G 是强连通图。

（11）其各个极大强连通子图称作它的强连通分量。 如果不考虑有向图中边的方向所得到的无向图是连通图，则有向图称为弱连通图



（10）强连通图



（11）强连通分量

（12）回路：起点和终点相同的路径，又称作“环”

（13）完全图：一个 n 阶的完全无向图含有 n*(n-1)/2 条边，一个 n 阶的完全有向图含有 n*(n-1)条边

注：完全图和连通图的区别在于，完全图是边边相连，连通图是存在路径即可

（14）稀疏图与稠密图：边数远远少于完全图的图/边数接近完全图的图

（15）拓扑排序：对于有向无环图来讲，每次选择入度为 0 的点，将其出度都删除，继续找入度为 0 的点即可。所以拓扑排序的结果并不唯一。

（16）简单图：在无向图中，关联一对顶点的无向边，如果多于 1 条，则成这些边为平行边。平行边的条数称之为重边。在有向图中，关联一对顶点的有向边如果多于 1 条，并且终点和起点一样（方向一致的情况），也称为平行边。含有平行边的图称为多重图。既不含平行边也不包含自环的图称为 简单图

2、图的一些性质

（1）有向图中，所有顶点的入度之和等于所有顶点的出度之和。

（2）有向图中，每个顶点的度等于该顶点的入度和出度之和。

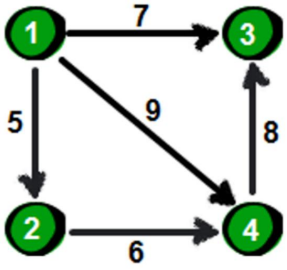
（3）有 n 个顶点的图，其对应的最小生成树有 n-1 条边

（4）有 n 个顶点的图，其对应的树有 n-1 条边。

3、图的存储方式

最常见的两种存储方式是邻接矩阵和邻接表。

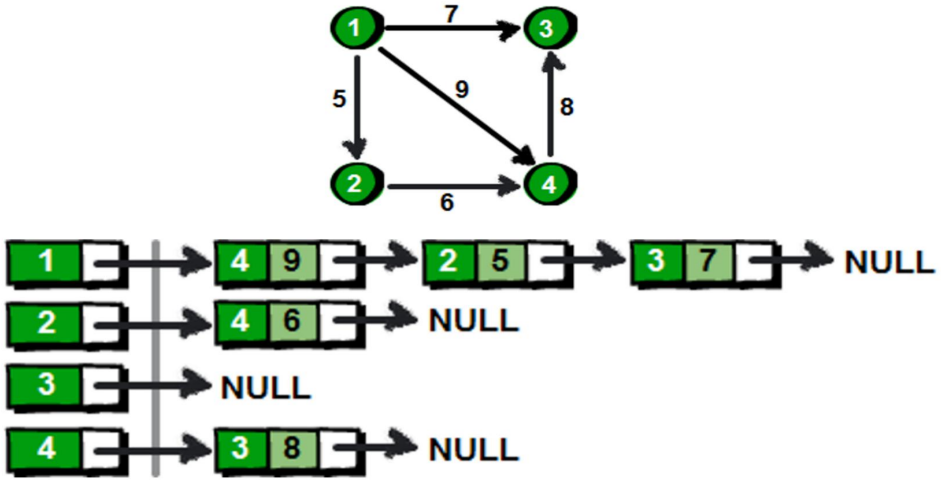
（1）邻接矩阵 a[i][j] 表示起点 i 终点 j 值可以用来表示权重。



点编号	1	2	3	4
1	0	5	7	9
2	-1	0	-1	6
3	-1	-1	0	-1
4	-1	-1	8	0

（2）邻接表

是图的一种顺序存储与链式存储相结合的存储方式，当想找某一个点为起点 的所有边时，不需要遍历所有的边或所有的点



4、最小生成树

（1）概念：一个有 n 个结点的连通图的生成树是原图的极小连通子图，且包含原图中的所有 n 个结点，并且有保持图连通的最少的边。

（2）有 n 个结点的图要变成一个数，最小需要 n-1 条边。

5、真题练习

（2010）18. 关于拓扑排序，下面说法正确的是（ ）。

A. 所有连通的有向图都可以实现拓扑排序

B. 对同一个图而言，拓扑排序的结果是唯一的

C. 拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点前面

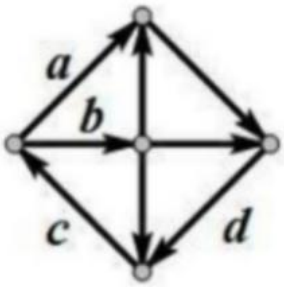
D. 拓扑排序结果序列中的第一个结点一定是入度为 0 的点

（2011）5. 无向完全图是图中每对顶点之间都恰有一条边的简单图。已知无向完全图 G 有 7 个顶点，则它共有（ ）条边。

A. 7 B. 21 C. 42 D. 49

（2011）19. 对一个有向图而言，如果每个节点都存在到达其他任何节点的路径，那么就称它是强连通的。例如，右图就是一个强连通图。事实上，在删掉边（ ）后，它依然是强连通的。

A. a B. b C. c D. d



（2017）10. 设 G 是有 n 个结点、m 条边（n≤m）的连通图，必须删去 G 的（ ）条边，才能使得 G 变成一棵树。

A. m－n+1 B. m-n C. m+n+1 D. n－m+1

（2016）15. 设简单无向图 G 有 16 条边且每个顶点的度数都是 2，则图 G 有（）个顶点。

A. 10　B. 12　C. 8　D. 16

阅读代码理解专题

一、要点分析

题型：一般不太出现指针，就算是链式结构，往往也是数组模拟

要点：认真、仔细，注意输出格式！拿满分！

分类：

① **简单送分题**

② **莽夫题**：一般没什么特别的规律，即使有也能莽出来，一步一步在草稿纸上写出变量的变化，逐步模拟，都能得到答案。循环次数一般不超过 30 次。(大部分都是这种类型的模拟题）

③ **规律题**：循环次数一看就不可莽，需要总结出规律，根据规律运算得到结果。一般 0~1 题。

④ **递归题**：画表格，反过来从递归边界开始逐个填充表格所有元素。切记递归题千万不能当莽夫！

注意：本题型比赛中严重拉分，主要因为不仔细算错数、看错代码。

细节：

大括号会省略，注意看清语句的归属，可以自己手动把括号补齐

一些简略的书写格式——

- if (!a) ☾ if (a == 0) //类似写法也会出现在循环条件里
- if (a) ☾ if (a != 0) //while (!a) // for (int i = 0; i ; i ++)
- scanf(“%d”, d + i) ; ☾ scanf(“%d”, &d[i]);

递归画表格法和分类讨论法（思维导图），表格中可能还能简化递推关系式。

求准确不求快，尽可能拿满分

耐心打草稿，记录每一个变量值的变化过程

如果有相似熟悉的代码，类比学过的内容

对熟悉的专题代码保持熟练度

4 道判断题部分：

A. 举反例

B. 改语句查结果有无变化，对比修改语句前后循环部分的区别手动模拟观察

C. 改语句查运行错误☾下标越界，除数为 0 等

D. 阅读代码不提供题面，很难直接从代码推敲出实际功能，更多直接从代码本身结构，条件语句直观入手分析

二、具体题型结合整体分析

1、模拟类:

循环技巧:(不要傻傻的模拟循环运行,太慢太繁琐容易炸脑)

技巧 1:快速判断循环次数:把条件想象成一个门卫,判断哪些人被放进门

技巧 2:循环次数多的一般都有规律,只需算前几项,后面推规律即可

例 1:

```
int a=123;
while(a>0){
    a-=10;
}
```

进门的 a 依次为　123 113 103.....13 3　　总共 13 个人进门

例 2:

```cpp
#include <iostream>
using namespace std;
int main(){
    int n,m,x=1,y=1,dx=1,dy=1,cnt=0;
    cin>>n>>m;
    while(cnt!=2){
        cnt=0;
        x+=dx,y+=dy;
        if(x==1||x==n){
            ++cnt;
            dx=-dx;
        }
        if(y==1||y==m){
            ++cnt;
            dy=-dy;
        }
    }
    cout<<x<<" "<<y<<endl;
    return 0;
}
```

输入：2017 1014

输出：_____

1.注意循环**什么时候停**:x 和 y 同时满足才停.

2.找出 x 的变化规律 1 2 ....2017 2016 .....1 2 .....2017 2016 .....1 2 ...

找出 y 的变化规律 1 2 ....1014 1013 .....1 2 .....1014 1013 .....1 2...

3.~~此题不要想周期,公共周期(直接炸脑),~~

x 到达边界的规律:1 1+2016 1+2016*2 1+2016*3....

y 到达边界的规律:1 1+1013 1+1013*2 1+1013*3....

假设同时达到边界:2016*n=1013*m 很明显最小公倍数了.....

2、递归类

技巧:盯死边界!,然后打表(或画思维导图,打表通常更简单,打不出来再画吧)

例 1:

```cpp
#include <iostream>
using namespace std;
int n,m;
int findans(int n,int m) {
    if(n==0) return m;
    if(m==0) return n%3;
    return findans(n-1,m)-findans(n,m-1)+findans(n-1,m-1);
}
int main(){
    cin>>n>>m;
    cout<<findans(n,m)<<endl;
    return 0;
}
```

输入 5 6

n\m	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	0	3	2	5	4	7
2	2	-1	4	1	6	3	8
3	0	1	2	3	4	5	6
4	1	0	3	2	5	4	7
5	2	-1	4	1	6	3	8

注意此题两条边界(m==0 和 n==0)

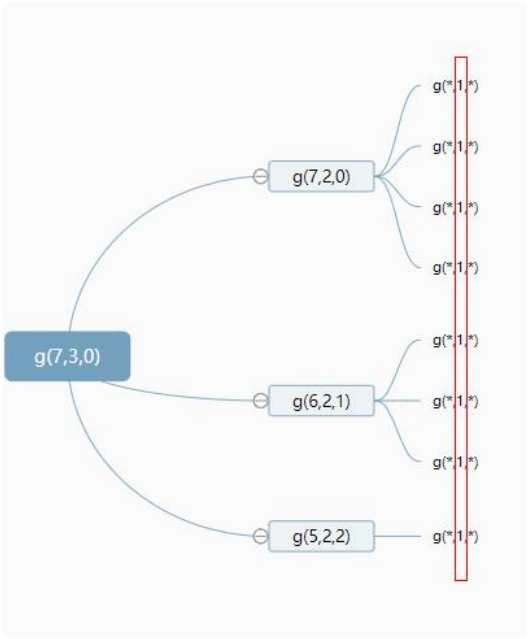
此题可以几何化:我=我的上面-我的左边+我的左上

例 2:

```cpp
#include <iostream>
using namespace std;
int g(int m,int n,int x){
    int ans=0;
    if(n==1) return 1;
    for(int i=x;i<=m/n;i++)
        ans+=g(m-i,n-1,i);
    return ans;
}
int main(){
    int t,m,n;
    cin>>m>>n;
    cout<<g(m,n,0)<<endl;
    return 0;
}
```

输入 7 3

此题 3 参数,打表三维,难度较大,改画思维导图



注意此题边界,当 n==1 时,m 和 x 已经不重要了.

三、历年真题分类

2016、规律模拟

2016、规律数组交换

2016、字符串

2017、字符串

2017、递归

2017、规律模拟

2018、字符串

2018、规律取模运算

2018、递归题

2019、字符串大小写转换

2019、规律题推导

2019、树+递归

2020、字符串，规律题

2020、进制转换、逐位推导

2020、等差数列，根据题目意思推导

完形填空专题

一、基本要点分析

第四大题要点：

- 理解每一个变量，每一个函数的作用，不存在无用的变量和函数
- 带着题目的需求，从 main 函数出发，沿着函数的运行过程去理解整个代码的思路，不一定要一遍看完代码马上全部填完，不会的可以先空着跳过，第一遍的目的重在理清代码的思路。理清思路再进行第二遍，第三遍，直到全部填完。
- 如果有结构相似代码，学会类比，譬如 gcd 的应用
- 要对熟悉的专题代码保持熟练度，经常会有基础算法的变型考验
- 二分答案多多重视这个模板和变型
- 不确定的地方半蒙半做半猜，不要留空
- 注意补全的语句的完整性，要仔细
- 总分数：28 分，2 题，严重拉分。

二、技巧部分

半蒙半做半猜——前言：

- 如果能分析出确定的结果，那就去正常做，不要依赖于技巧。
- 如果有几个空不明所以，甚至看不懂整道题在干嘛，请认真阅读以下内容。

补全程序中的许多空是有多个答案的，理论上来说只要可以正确让程序运行即可。考试中填空时最好写普通一些的答案，如果太偏，可能会被误判。

解题格式：（注意格式上的陷阱）

(1) 例如题目为：if(①) 答案应该是小括号里面的条件，而不是连带着小括号一起的内容。即答题纸上应该填写 a>1，而不是（a>1）

(2) 例如题目为：s= ① 答案应该加上 ；才能让程序变得完整。如果题目是 s= ① ；答案就不应该加上 ；

(3) 正常解题思路：

- ✧ 读题目，标注好题目要求做哪些事情。最重要的是要知道题目最终要去求什么！
- ✧ 将程序大致划分成几个模块，标注好每个模块的大致功能。
- ✧ 标注每个变量、函数的含义、功能。尤其是带有英文单词的变量或函数名，单词意思≈实际功能
- ✧ 细致分析每个空可能实现的功能，填写相应语句。

(4) 当毫无头绪的情况下，提供以下几个思考方向：

- ✧ 每个变量几乎都会用到，所以向没用过的变量的方向去思考。
- ✧ 每个使用过的变量都会有用！☾ 程序没有废语句，不可能赋值以后就不再使用了。
- ✧ 变量数值突然变化！例如初始化某变量 a 为 0，后面突然判断 a 是否为 1。而且，初始化的变量很多时候的作用：累加/计数/判断。

- ✧ 联系上下文，相同或者相似结构中的代码一定会有联系。但是要注意相似代码的区别。
- ✧ 观察有没有哪个变量没赋值过就直接参与了运算，如果有，这个变量的前面一点会需要写对应的赋值或者输入语句。
- ✧ 如果有残缺的输入输出内容，在适当的位置上添加。

谨记：变量含义+题目求解内容+技巧 >> 透彻的理解代码的实现原理

算法类技巧:

此部分必须要有算法基础,各种算法技巧太多,不能一一陈诉.

玄学类技巧:

1.程序对称性~~(骗分)~~;

2.找程序没被维护的变量,数组,多半是在填空处维护~~,至于如何维护,还是得理解算法~~.:

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

const int N = 100010;
int n;
int L[N], R[N], a[N];

int main() {
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        int x;
        cin >> x;
        ① ;
    }

    for (int i = 1; i <= n; ++i) {
        R[i] = ② ;
        L[i] = i - 1;
    }

    for (int i = 1; i <= n; ++i) {
        L[ ③ ] = L[a[i]];
        R[L[a[i]]] = R[ ④ ];
    }

    for (int i = 1; i <= n; ++i) {
        cout << ⑤ << " ";
    }

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

很明显a数组没维护

很明显的对称性

很明显的对称性

此处不懂算法没法做

三、历年真题分类

完善程序：

2016、字符串

2016、二分

2017、快速幂

2017、二分

2018、最大公因数

2018、链表

2019、矩阵、递归

2019、排序

2020、质因数分解

2020、排序+数组

四、总结：熟知模板和认真推论很重要！！！

考察内容：

变量方向的填空

循环方向的填空

分支转移方面的填空

主程序和子程序关系方面的填空

输入输出方面的填空

方法：

从总体上通读程序，大致把握程序的目的和算法；

猜测变量的作用，跟踪主要变量值的变化（列表），找出规律；

将程序分段，理清每一小段的作用和目的（灵感+关键表达式和语句的领会）；

按照算法逻辑填空

整体思路：

通读 -> 假定 -> 填写 -> 验证 -> 调整 -> 检查验证

五、基础算法模板

1、排序算法

1.1 选择排序

```
1. void selectsort(int a[], int n) {
2.     for (int i = 1; i < n; i ++) { //n - 1 次排序
3.         int minn = i; // minn —— 最小值的下标(位置)
4.         for (int j = i + 1; j <= n; j ++)
5.             if (a[j] < a[minn]) minn = j;
6.         if (minn != i) swap(a[i], a[minn]);
7.     }
8. }
```

1.2 冒泡排序

```
1. void bubblesort(int a[], int n) {
2.     for (int i = 1; i < n; i ++) { //n - 1 次排序
3.         bool flag = true;
4.         for (int j = 1; j <= n - i; j ++)
5.             if (a[j] > a[j + 1]) {
6.                 swap(a[j], a[j + 1]);
7.                 flag = false; //如果发生交换，说明排序没有完成
8.             }
9.         if (flag == true) return ; //没有返回值
```

```
10.         }
11.     }
```

1.3 桶排序

```
1. void countingsort(int a[], int n) {
2.      int b[maxnum] = {0}, minn = a[1], maxn = a[1]; //maxnum 数字出现的最大值
3.      for (int i = 1; i <= n; i ++) {
4.          b[a[i]] ++;
5.          minn = min(a[i], minn);
6.          maxn = max(a[i], maxn);
7.      }
8.      for (int k = 0, i = minn; i <= maxn; i ++)
9.          for (int j = 1; j <= b[i]; j ++)
10.                 a[++ k] = i;
11.     }
```

1.4 快速排序

```
1. void quicksort(int a[], int start, int end) {
2.     int mid = a[(start + end) / 2], i = start, j = end;
3.     while (i < j) {
4.         while (a[i] < mid) i ++;
5.         while (a[j] > mid) j --;
6.         if (i <= j) swap(a[i ++], a[j --]);
```

```
7.     }
8.     if (start < j) quicksort(a, start, j);
9.     if (i < end) quicksort(a, i, end);
10.    }
```

1.5 归并排序

```
1. void mergearray(int a[], int temp[], int first, int mid, int last) {
2.     int i = first, j = mid + 1, k = 0;
3.     int m = mid, n = last;
4.     while (i <= m && j <= n) {
5.         if (a[i] <= a[j]) temp[k ++] = a[i ++];
6.         else temp[k ++] = a[j ++];
7.     }
8.     while (i <= m) temp[k ++] = a[i ++];
9.     while (j <= n) temp[k ++] = a[j ++];
10.        for (i = 0; i < k; i ++) a[first + i] = temp[i];
11.    }
12.    void mergesort(int a[], int temp[], int first, int last) {
13.        if (first < last) {
14.            int mid = (first + last) >> 1;
15.            mergesort(a, temp, first, mid);
16.            mergesort(a, temp, mid + 1, last);
17.            mergearray(a, temp, first, mid, last);
```

```
18.         }
19.     }
```

2、基础数学算法

2.1 单个质数判断

```
1. bool is_prime(int num) {
2.     if (num < 2) return false;
3.     for (int i = 2; i <= sqrt(num); i ++)
4.         if (num % i == 0) return false;
5.     return true;
6. }
```

2.2 最大公约数递归算法

```
1. int gcd(int a, int b) {
2.     if (b == 0) return a;
3.     else return gcd(b, a % b);
4. }
5. int lcm (int a, int b) {
6.     return a * b / gcd(a, b);
7. }
```

2.3 埃式筛法

```
1. void Eratosthenes_Sieve(int maxtmp) { // 筛选范围从 2 ~ maxtmp
   里的所有质数
```

```
2.     memset(flag, false, sizeof flag); // 这一步为了让范围以外(最主要是小于 2)的数字都标记为不是质数的状态
3.     for (int i = 2; i <= maxtmp; i ++) // 首先初始化从 2 开始的所有都是质数的状态
4.         flag[i] = true;
5.     for (int i = 2; i <= maxtmp; i ++) { // 筛选开始
6.         if (flag[i] == true) // 如果当前数字 I 是质数
7.             for (int j = i * 2; j <= maxtmp; j += i) // 那么筛除数字 I 的所有倍数(从 2 倍开始)
8.                 flag[j] = false; // 筛除即更新对应数字的 flag 状态值
9.     }
10.    }
```

2.4 欧拉筛法

```
1. void Euler_Sieve(int maxtmp) { //对比埃式筛法，避免了同一个数字被重复筛选判断
2.     memset(flag, false, sizeof flag); // 这一步为了让范围以外(最主要是小于 2)的数字都标记为不是质数的状态
3.     for (int i = 2; i <= maxtmp; i ++) // 首先初始化从 2 开始的所有都是质数的状态
4.         flag[i] = true;
5.     for (int i = 2; i <= maxtmp; i ++) { // 筛选开始
```

```
6.          if (flag[i] == true) prime[++ tot] = i; // 如果当前数字 I 是质数，那么添加到素数表里
7.          for (int j = 1; j <= tot; j ++) { // 遍历素数表的全部元素
8.              if (i * prime[j] > maxtmp) break; // 如果筛选的数字超出最大范围，直接结束
9.              flag[i * prime[j]] = false; // 将倍数 i 倍*某个质数筛除，标记状态为非质数
10.             if (i % prime[j] == 0) break; // 保证只筛到以 prime[j]为最小质因数的数，退出内层循环
11.         }
12.     }
13. }
```

3、二分答案

```
1. int binarysearch(int left, int right) {
2.     int ans = -1;
3.     while (left <= right) {
4.         int mid = (left + right) / 2;
5.         if (check(mid) == true) {
6.             ans = mid;
7.             left = mid + 1;
8.         }
```

```
9.          else right = mid - 1;
10.         }
11.         return ans;
12.     }
```