初赛复习

南开信竞教练组 董又铭

题型分析

初赛题可能考什么？

初赛有哪些题型？

- 选择题（30分）
  - 15道，每题2分
- 阅读程序题（40分）
  - 3份程序
  - 判断题，每题1.5分
  - 选择题，每题3分
- 完善程序题（30分）
  - 2份程序
  - 选择题，每题3分

- 选择题均为4个选项，单选
- 其他有些“早年题型”，不考虑

单选题 考哪些知识？

- 计算机基础知识、常识
- 算法常识、经典算法、常用数据结构
- 计数问题、计算题

- 注意事项
  - 大部分题目是送分，也有些是故意留坑
  - 题目之间没有关联性，不要因为前面的题搞心态而影响后面的题
  - 如果觉得选项全都不对，赶紧重新审题
    如果真的觉得选项都不对，选最接近正确答案的
  - 计数题和计算题，尽量采用两种不同方法/顺序，确保自己算对

阅读程序 考哪些知识和能力？

- 阅读程序，分析出它在解决什么问题
- 考察对程序细节的理解
  - 当n时某个值时，数组是否会越界？
  - xxxx这句话如果删除，程序会不会改变？
  - 如果把xxxxx这句话改成yyyyyyy，程序会输出什么？
- 时间复杂度的计算
  - 最好/最坏/平均 的时间复杂度
  - 特殊输入数据下的时间复杂度

阅读程序 考哪些知识和能力？

- 注意事项
  - 每个小题一般是独立的
  - 程序可能特别长，但也要耐心读完再做题
    先看变量名、函数名有没有暗示，再看主函数内容，再看其他函数
  - 一边读，一边勾画重点和写批注
  - 递归函数如果难理解，手动设定几个参数算答案，找规律
  - “如果输入xxx，程序会输出什么”这种题，先理解整个程序，再计算答案
    不要盲目的直接手动模拟程序运行，容易算错

完善程序 考哪些知识和能力？

- 会给出一段文字，描述程序解决的问题
  - 甚至可能会给出程序使用算法的介绍
- 做题时
  - 每个空，仔细阅读文字提示和程序上下文
  - 不一定按顺序填每个空，有把握的空先填
  - 填空时，先直接按自己习惯填，再对照选项
  - 没有想法时，可以尝试将选项逐个代入，分析选项之间的差异
  - 多个空连在一起，且关系紧密时，要反复检查
  - 仔细思考所有可能的特殊情况
  - 填完后再把整个程序通读一遍
- 注意事项
  - 每个小题可能有一些关联性，做错一个可能就连续错多个
  - 一边读，一边勾画重点和写批注

基于往年题目复习

往年都考了些什么？

今年有没有可能考类似的？

计算机基础知识

- 操作系统：
  - Windows, Linux, MAC OS, Android, IOS, ...
  - 功能：管理计算机硬件和软件资源
- Linux命令：
  - pwd　　显示目录路径
  - cd　　更改目录路径
  - ls　　查看目录中的文件和文件夹

  - mkdir　　创建文件夹
  - cp　　复制文件或文件夹
  - rm　　删除文件或文件夹
  - mv　　移动文件或文件夹

计算机基础知识

- Linux命令（续上一页）：
  - time ./xxx
    - 运行xxx这个程序，同时测量运行时间
    - real time　　程序开始到结束的总时间
    - user time　　进程xxx用户态占用CPU的时间
    - sys time　　进程xxx内核态占用的CPU时间
    - 理论上 real = user + sys
  - gdb调试
    - break（或者b）行号或者函数名　　设置断点
    - continue（或者c）　　连续执行（调用其他函数时直接运行至返回当前函数）
    - step（或者s）　　单步执行（调用其他函数时进入函数内）
    - display 变量名　　跟踪查看某变量，每次停下都显示它的值

计算机基础知识

- 编译器
  - C++的编译器是g++，C语言的编译器是gcc
  - 功能：把源程序（高级语言）翻译成机器语言（低级语言）
  - 基本格式　　g++ akioi.cpp -o akioi.exe
  - 编译选项
    -O0 / -O1 / -O2 / -O3　　编译时的优化等级
    -lm　　使用数学库时需要，高版本g++默认已添加
    -std=c++14　　使用c++的版本
    -Wall　　显示所有警告
    -g　　生成调试信息
    -Wl,--stack=256000000　　扩大系统的递归栈空间
    （仅windows有效，linux扩大的方法不同）
  - 用法就是串烧，顺序可以随意
    g++ -lm -O2 akioi.cpp -g -Wall -o akioi.exe -std=c++14

计算机基础知识

- 硬件
  - 中央处理器(CPU)
    - 计算机的运算和控制核心
    - 控制器、运算器、缓存(Cache)
  - 内存(Memary)
    - 内存储器，也叫主存储器
    - 正在运行的程序的指令和数据
    - CPU与外存沟通的桥梁
  - 其他设备
    - 机箱里：显卡、声卡、网卡、硬盘……
    - 机箱外：鼠标、键盘、耳机、显示器……

计算机基础知识

- 冯·诺依曼(John von Neumann)
  - 现代计算机之父
- 图灵(Alan Mathison Turing)
  - 提出图灵机、可计算性理论
- 克劳德·艾尔伍德·香农(Claude Elwood Shannon)
  - 引入热力学的熵理论，提出信息熵
- 林纳斯·本纳第克特·托瓦兹(Linus Benedict Torvalds)
  - Linux之父
- 丹尼斯·里奇(Dennis Ritchie)
  - C语言之父、Unix之父
- 本贾尼·斯特劳斯特卢普(Bjarne Stroustrup)
  - C++语言之父

算法常识

- 枚举
- 模拟
- 搜索
  - 还是枚举，但不一定要完全枚举
  - 深搜、广搜、双向广搜、A*、迭代加深、剪枝……
- 贪心（也叫贪婪）
  - 求解问题是总是选择当前最优的决策
  - 一些问题不能贪心
- 动态规划
  - 划分阶段、设定状态、状态转移方程
  - 与递推的区别在于转移时是否有决策
- 倍增

算法常识

- 分治
  - 把问题划分规模更小的一个或多个子问题，分别求解
- 随机化算法
  - 解决问题时引入了随机
  - 数值随机算法：求数值的近似解，时间越长精度越高
    例：蒲丰投针
  - 蒙特卡罗算法：求精确解，但不一定正确；时间越长，正确概率越大
    例：Miller-Rabin算法
  - 拉斯维加斯算法：可能找不到解，但一旦找到解一定是正确的，时间越长，找到解的概率越大
    例：Pollard-rho算法、Cipolla算法
  - 舍伍德算法：总能找到解，且一定是正确的
    例：快速排序、快速选择、Treap

算法常识

- 图论基本概念
  - 点（也叫节点、顶点）和边（也叫弧）
  - 邻接矩阵、邻接链表（也叫邻接表、链式存图、链式前向星）
  - 有向图、无向图、简单图、连通图、完全图、竞赛图、二分图
  - 独立集、团
  - 拓扑序、传递闭包、有向无环图（DAG）
  - 路径、回路、简单路、简单回路、欧拉路和欧拉回路
  - 树、森林、基环树
  - 二叉树、完全二叉树、满二叉树、前序/中序/后续遍历、多叉转二叉
  - 强连通、弱连通、点双连通、边双连通、割点、割边（也叫桥）
  - 流、割

经典算法、常见数据结构

- 太多
- 都比较熟悉
- 就不细说了

计数问题

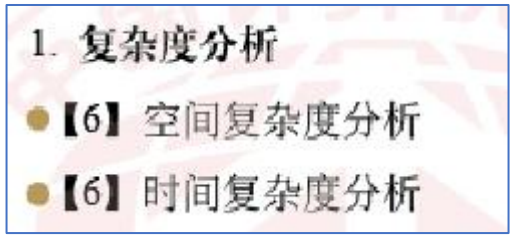
- 初赛题可能用到的方法
  - 排列组合
  - 递推法
  - 容斥原理
- 不会算时的备用方法
  - 暴力枚举
  - 排除某些选项
  - 考虑n规模较小时找规律
- 理论上不会用到的方法
  - 行列式
  - 生成函数
  - 反演
  - ……

计算题

- 进制转换、特殊进制的加减法、位运算
  - 送分题，但一定认真仔细
  - 原码、反码、补码，三者的关系
  - 熟记ASCII码表
  - 常数：八进制开头写个0，十六进制开头写个0x。
- 前缀/中缀/后缀表达式
  - 前缀=符号在前，后缀=符号在后
  - 波兰表达式=前缀，逆波兰表达式=后缀
  - 前缀/后缀表达式，不需要打括号就能确定运算顺序
- 仔细关注选项间的差异！

计算题

- 数论类型的计算
  - 灵活使用各个定理
    （欧拉、费马、威尔逊、卢卡斯、中国剩余……）
  - 必要时使用暴力
- 图论类型的
  - 手算最短路、生成树、拓扑序之类的
  - 送分题，不要急，稳扎稳打
- 时间复杂度相关的
  - 给个公式，例如T(n)=2*T(n/2)+O(n)，例如T(n)=T(n-1)+O(n^2)
    - 回想学过的哪个算法符合它
  - 给一小段代码
    - 先认真读懂，再进行分析

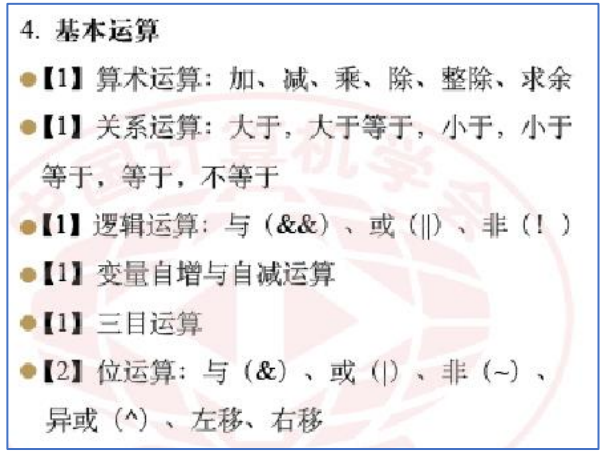


基于考纲复习

考纲里有什么平时讲的少的？

哪些内容适合在初赛考？

运算符优先级


4. 基本运算

- 【1】算术运算：加、减、乘、除、整除、求余
- 【1】关系运算：大于，大于等于，小于，小于等于，等于，不等于
- 【1】逻辑运算：与（&&）、或（||）、非（！）
- 【1】变量自增与自减运算
- 【1】三目运算
- 【2】位运算：与（&）、或（|）、非（~）、异或（^）、左移、右移

- 单目运算
  - 右侧运算符　`a++`　`a[i]`　`a.x`　`a->x`　`a::x`
  - 左侧运算符　`++a`　`!a`　`~a`　`-a`　`&a`　`(int)a`
- 双目运算
  - 乘、除、模　`a*b`　`a/b`　`a%b`
  - 加、减　`a+b`　`a-b`
  - 移位　`a<<b`　`a>>b`　`cin>>a`
  - 比较运算　`a<b`　`a>b`　`a<=b`　`a>=b`　`a!=b`　`a==b`　（后俩更低）
  - 按位与、异或、或　`a&b`　`a^b`　`a|b`　（这仨依次递减）
  - 逻辑与、或　`a&&b`　`a||b`　（这俩依次递减）
- 三目运算　`a ? b : c`
- 赋值运算　`a=b`　`a+=b`　`a^=b`　`a<<=b`

switch语句

- 略

●【2】if 语句，switch 语句，多层条件语句

输入输出格式控制

【2】cin 语句，scanf 语句，cout 语句，printf 语句，赋值语句，复合语句

- cout浮点数
  - 输出8位小数
但末尾默认补的是空格 `cout << setprecision(8) << x;`
  - 切换到末尾补零 `cout << fixed << setprecision(8) << x;`
- printf
  - 保留3位小数 `printf("%.3lf", x);`
  - 保留3位小数，整数高位补空格凑足6位 `printf("%6.3lf", x);`
  - 保留3位小数，整数高位补0凑足6位 `printf("%06.3lf", x);`
  - 输出整数，高位补0 `printf("%06d", x);`
- scanf字符串
  - 只接受指定字符 `scanf("%[a-zA-Z]", str);`
  - 不接受指定字符 `scanf("%[^\n]", str);`

结构体、类

1. 类（class）
- 【6】类的概念及简单应用
- 【6】成员函数和运算符重载

- 区别
  - 结构体(struct)成员默认是public
  - 类(class)成员默认是private
  - 也可以给每个成员单独设置，所以struct和class可以相互转化，本质相同
- 成员变量、成员函数的访问权限
  - public：外部可以访问
  - protected：派生类可以访问，外部不能访问
  - private：仅内部可以访问
- 一些写法
  - 构造函数
  - 友元函数
  - 重载运算符
  - 强制类型转换
  - 继承

2.【8】面向对象的程序设计思想 (OOP)

指针

- 地址与指针
- new和delete

- 【4】指针的概念及调用
- 【4】指针与数组
- 【4】字符指针与 string 类
- 【4】指向结构体的指针

STL

- 大家都熟悉的
  - string
  - vector
  - pair
  - stack, queue, deque
  - priority_queue
  - set, multiset
  - map, multimap
- 可能不太熟悉的
  - list
  - tuple
  - iterator
  - ……

13. STL 模板应用

- 【3】<algorithm> 中 sort 函数
- 【4】栈（stack）、队列（queue）、链表（list）、向量（vector）等容器

2. STL 模板

- 【5】集合（set）
- 【5】列表（list），双端队列 (deque)，优先队列（priority_queue）
- 【5】多重集合 (multiset)
- 【5】映射（map），多重映射 (multimap)
- 【5】对（pair），元组（tuple）

1.【8】STL 模板: 容器 (containers)、迭代器 (iterators)、空间配置器 (allocators)、配接器 (adapters)、算法 (algorithms)、仿函数 (functors)

2.【8】面向对象的程序设计思想 (OOP)

伪代码、算法流程图

- 【2】算法描述：自然语言描述、流程图描述、伪代码描述

- 伪代码
  - 没有固定的语法，有时需要猜测它符号的含义
  - 举几个例子

`a := b`	一般表示赋值
`a ← b`	可能表示赋值，也可能表示更新答案
`a ↔ b`	可能表示交换数值
`FOR a := 1 TO n`	不也就是个for循环嘛
`CALL xxx`	一般表示调用某个函数

- 流程图
  - 比较容易看懂
  - 认真仔细+手动模拟

排序算法

5. 排序算法
- 【3】排序的基本概念（稳定性等）
- 【3】冒泡排序
- 【3】简单选择排序
- 【3】简单插入排序

- 稳定性？
  - 看相等的数的相对位置是否可能改变
  - 稳定的：冒泡排序、插入排序、归并排序、桶排序、基数排序
  - 不稳定的：选择排序、快速排序、希尔排序、堆排序、锦标赛排序、猴子排序
- 基于比较/基于分组？
  - 基于比较：每次把两个元素拿来比较
    例：选择、冒泡、插入、快速、归并、希尔……
  - 基于分组：根据元素的某些特征进行划分
    一般情况下，只能用来对整数、字符串进行排序
    例：桶排序、计数排序、基数排序……

3. 排序算法
- 【5】归并排序
- 【5】快速排序
- 【6】堆排序
- 【6】树形选择排序（锦标赛排序）
- 【5】桶排序
- 【6】基数排序

锦标赛排序

- 也叫“树形选择排序”
  - （从某种意义上说，叫做“线段树排序”也很有道理）
- 特点：
  - 比大小的次数最少
  - 额外空间O(n)
  - 不稳定
- 举个例子：
  - 找出n个数的最大和次大。最少进行多少次比大小？
  - ……

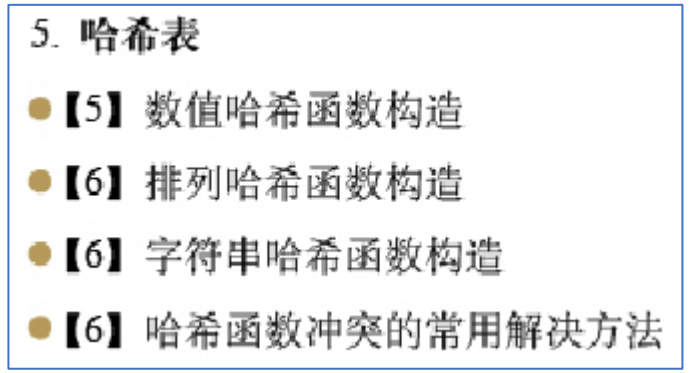
哈夫曼树、哈夫曼编码

- 【2】编码：ASCII 码，哈夫曼编码，格雷码
- 【4】哈夫曼树的定义、构造及其遍历

- 解决的问题：
  - 对不同频率的若干元素进行二进制编码
  - 在无前缀包含关系的情况下使平均编码量最小
- 基本思想：
  - 堆维护贪心　　或者　　排序+队列维护贪心
- 举个例子：
  - 一篇文章中a,b,c,d,e,f分别出现了10,7,8,2,17,2次，求最优编码长度
  - ……

哈希与冲突

- 数值哈希、哈希函数、冲突处理策略
  - 题目会描述得很清楚
  - 举个例子(csp2020提高)
  - ……



笛卡尔树

- 性质
  - 中序遍历=原序列
  - 数值满足堆性质（类似Treap）
- 构建算法
  - 单调队列跑一遍
- 作用
  - 序列转化为树
    - RMQ转化LCA
    - 一些DP或者计数题
  - SA的height数组+笛卡尔树
    - 等价于后缀树
  - Treap的O(n)建树
    - 有一点点优化效果，但似乎也没啥必要

解析几何、立体几何

- 平面解析几何
  - 点、向量、向量运算
  - 直线表示方法（两点、点+方向向量、直线方程）
  - 线段、射线
  - 投影的计算
  - 圆
  - 线与线、线与圆、圆与圆的位置关系
  - 多边形、简单多边形、凸多边形
  - 以上图形的周长、面积计算
  - 圆锥曲线

解析几何、立体几何

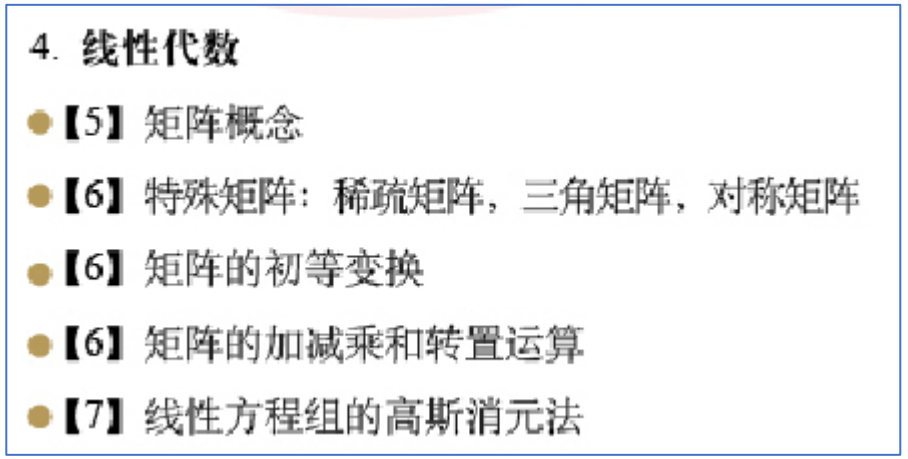
- 立体几何
  - 点、三维向量、向量运算
  - 平面（三点、点+法向量、平面方程）
  - 距离、投影
  - 正方体、长方体、棱柱、棱锥
  - 圆柱、圆锥、球
  - 以上几何体的表面积和体积公式

解析几何、立体几何

- 常用数学函数
  - 三角函数
  - 反三角函数
  - 正弦定理、余弦定理
  - ……
- 注意事项
  - 认真打草稿
  - 多画图，尽量画得精确点
  - 立体的问题，可以画平面图、截面图

线性代数

- 矩阵
  - 加、减、乘、转置
  - 变种的矩阵乘法
- 高斯
  - 解方程组
  - 求逆矩阵
  - 求行列式



祝大家顺利AK