* **清华大学-求最大最小数**

简单题，调用快速排序内置库即可~

* **上海交通大学-日期差值**

给定两个时间，求出日期差值，是个简单题，难度不大，本道题的最佳思路是虚拟设一个基准时间，去计算时间差，利于程序的书写，本道题TL了好多次，其实就是输入设成了死循环，下次要注意这个问题！

* **浙江大学-统计同成绩学生人数**

统计符合查询成绩的人数，是一个简单题，本道题我的方法比较笨，是用了快速排序+二分查找的一个常规方案，因为用例N数目最大只有1000，快排+二分查找只需 n\*log(n)+log(n)的时间复杂度，小于百万级别，所以可以采用~其中本道题采用qsort()函数以及bsearch()函数，对于bsearch()函数它返回的是地址，可以利用地址的相减去求得下标，关于bsearch()函数的使用可以多加记忆！！

* **上海交通大学-Simple Sorting**

排序输出的问题，去除重复的元素输出，较为简单，但为了保证输出格式，又做了些别的处理，很恶心的地方！！

* **哈尔滨工业大学-找x**

水题，本来一开始以为可以快排+二分查找，后来发现不需要，直接暴力就可以了，没意思

* **清华大学-查找学生信息**

查找学生信息问题，是一道排序+二分查找的问题，复杂度为n\*log(n)+m\*log(n)，小于百万级别，关于内容的存储用结构体存储，同时二分查找的返回地址用Node\*得到相应的节点，从这个思路来想，求出原数组的下标是个很简单的问题，用结构体即可。关于结构体的二分查找，有个巧方法！可以不匹配完整的结构体，只匹配一部分时，需要把该值置于结构体第一个元素位（虽然并不知道为什么…）！此道题的解法也颇有意思，可以多多把玩此类题型~

* **FatMouse‘ Trade**

给出总金额，然后给出各个商品总数以及单价，现在有这些金额获取最多的数目，很简单的贪心问题，每次都买最高性价比的商品即可~用结构体去表述问题更容易些~

* **今年暑假不AC**

给出一系列的节目时间数，找出可以独立观看的节目最大数，也是一个贪心问题，就是每次都选择当前结束时间最早的节目观看，这是重心所在！！

* **括号匹配**

用栈操作去判别左括号、右括号位置，并压栈记录位置，按对应位置输出，题目较简单，解法做得很精巧~可以多看看

* **浙江大学-简单计算器**

本道题和POJ上的解题思路颇为一致，难度较小，没有左括号和右括号的判别，本道题仍采用了将中缀表达式转换为后缀表达式的思路，具体解法可以参考POJ的解题报告，关于读入一行字符串，此处采用了C语言形式的gets()函数，很容易使用，可以记忆！

* **清华大学-特殊乘法**

用字符串处理，水题。

* **浙江大学-又一版 A+B**

高精度运算，因为数字位数都没有那么高，还可以用long long型存储，难度不大，进制转换问题，留意此处对于0 0输入时当时没有考虑，是一个思维的漏洞~

* **北京大学-数制转换**

模板题，就是进行两次数制转换，并且涉及了一部分字符串的处理，还是蛮有意思的题，可以多看看了解了解~

* **哈尔滨工业大学-最大公约数**

最大公约数用欧几里得算法思想去解决~本道题解题代码较简单，但是这个思想要了解~

a和b的最大公约数一定也是 b和a%b的最大公约数；如果a和b，b为0，a为正整数，那么最大公约数为a，非常好的算法思路~

* **最小公倍数**

最小公倍数的思想就是，求出a和b的最大公约数，将a和b相乘后，除以这个最大公约数就是a和b的最小公倍数~思想也是极好的思路，要多理解~~

* **哈尔滨工业大学-素数判定**

素数判定的思路很easy，就是找是否含有除1和本身之外的因子，水题~

* **北京航空航天大学-素数**

水题，判定素数，很简单的思路~

* **清华大学-质因数的个数**

素因子分解题，POJ有类似的题目，当时的解法是直接遍历，时间复杂度为O(sqrt(n))，本道题采用了预处理后再运算，其实感觉时间复杂度上更大了…不过最后都能AC，话说在initial()中数组加了初始化后，就没有TL了有是什么奇葩问题？？

* **上海交通大学-整除问题**

蛮有意思的一道题，很有数学韵味的题，难度较大，主要是要知道如何把阶层拆分为素因子幂次相乘形式，其次，还应比对各个幂次的除法求K，很有思想的题目！！值得多多把玩理解！！

* **人见人爱 A^B**

二分求幂的思路，关于二分求幂，在刷POJ时也有碰到过此类题型，但当时理解得不够深刻，甚至理解得出现了一些问题，现在再更正下思路。对于a^b，可以将b转换为2的幂次，如12^6🡪6=1\*2^2+1\*2^1+0\*2^0，那么12^6可以看为：12^(1\*2^2+1\*2^1+0\*2^0) 于是有12^(2^2)=12^(2^1)\*12^(2^1)、12^(2^1)=12^(2^0)\*12^(2^0)这样的思维转换，从最底层开始乘起，不断彼此相乘，若某位有1，那么乘上该值，这是一个很好的求幂次的思路，尤其对于幂次并不在long long 型范围内时，常见的套路题就是求出低多少位的值~

* **华中科技大学-a+b**

高精度运算，大整数加法，简单题型，把住核心思路即可，其实可以采用int 型数组去从存储各个位的进位，也免去了char型存储转换的不便~~

* **清华大学-N的阶乘**

大数乘法，题目本身难度不大，就是稍微有点繁琐，此处在cstring库的memset()函数的调用上出了一些问题！函数原型是：void \*memset(void \*s, int ch, size\_t n); 于是可以知道第三个参数是指数组的大小，于是调用函数时出了些问题！要记住这种简单问题不可再错！其次是N的阶乘用大数运算表示时需要开较大的数组存储，本处一开始开了2000位的数组但是开小了，改开10000位的数组成功AC~

* **华中科技大学-大整数排序**

排序问题，较为简单，利用qsort()函数可以轻松解决~此种题可以转化为字符串排序问题，用结构体存储再加上快速排序便可以轻松AC！

* **北京大学-进制转换**

写出一个程序，接受一个十六进制的数值字符串，输出该数值的十进制字符串。这是一道典型的进制转换问题，一开始猜测是高精度问题，在心中默默推演之后发现难度不小…但是之后惊奇地发现这是一道非高精度问题，就是道水题…

（附上自己关于如果是高精度的大数运算的思路：逐位进行高精度乘法，并且由于有大数幂次，所以幂次的求值可以采用快速幂算法，当然也需要用到高精度乘法，粗略一想就感觉时间复杂度很大…但是看一些高校机试题，发现清华竟然灭绝人性地出了这种题，极其复杂..）

* **浙江大学-畅通工程**

给出城镇之间彼此连接的节点，要求求出使得城镇之间直接或间接连接的最短额外搭建的边数。此道题可以由树的角度出发，是典型的并查集问题，先要构建一个针对于并查集的树，然后求出独立的集合数目，减一后即为最终求得的边数，这是道经典的模板题以及模板题解，一定要熟练掌握！！

* **More is better**

典型的并查集问题，只不过加了一个计数的功能，程序也是一个模板示例，一定要熟练掌握！！

* **浙江大学-还是畅通工程**

给出城镇之间路径的长度，要求搭建公路，保证所有城镇之间可以彼此到达，并且铺设的公路长度最短。这是典型的最小生成树问题，利用Kruskal算法，每次找到当前最短边，并判别其是否属于最小生成树集合中，于是题目转换为了快速排序以及并查集问题，本道题用到了结构体的快速排序+基本的并查集算法。

* **北京大学-Frackles**

给出各个点坐标，试将所有点都可以连接起来，并且总共的距离值是最小的。

这道题和浙大的还是畅通工程思路是相似的，只需要先把各个点之间彼此的距离值算出来，然后用结构体存储，之后就转换为了还是畅通工程这道问题。

本道题用到了iomanip库的setprecision()函数以及fixed；cstring库的memset()函数；cmath库的sqrt()函数。并且调用了qsort()函数进行快速排序。

* **最短路-Floyd**

给出所有结点之间彼此的距离，试求出各个节点之间的最短路径。

这是一道最基本最原始的最短路问题，就是套用Floyd算法模板做题，本程序是一个模板程序，一定要记住！！

* **最短路-Dijkstra**

Dijkstra算法适合于单源求最短路径问题，程序是非常经典的实现流程，一定要多加记忆并且熟练掌握！！程序本身的思路就是模拟Dijkstra我们学习时用的那种表格模拟流程，所以携带着一同记忆更有利于深入理解~

* **浙江大学-最短路径问题**

给你n个点，m条无向边，每条边都有长度d和花费p，给你起点s终点t，要求输出起点到终点的最短距离及其花费.如果最短距离有多条路线，则输出花费最少的。典型的最短路径问题，只不过在最短路径上又加了最小花费的约束，不过只是在最短路径源代码上稍作修改就可以解决~Dijkstra程序极其经典，一定要多加理解！！

* **上海交通大学-最短路径**

这是一道较难的题，虽然是典型的最短路径问题，但是因为有2的高次幂存在，所以用大数运算去处理的话费时又费力，于是用了一个巧妙的思路-最小生成树的思想！本道题是典型的最短路径问题，但因为有2的N次方存在，所以要么用大数运算，不过大数运算求解思路并没有彻底理清楚...故而采用了最小生成树的算法，分析可知2^n>Σ2^k，所以可以知道 对于新的路径，如果两个节点均在一个集合下，那么这两个点之间的路径长度肯定有比这个新的路径小的存在；对于两个不在同一集合的端点，说明该两个集合内的点，它们之前彼此没有路径连接，那么此时两个集合下的点之间的距离一定都是最短距离，所以需要遍历两个集合，将所有彼此二者关联的点计算出路径~

本道题思路较难，对于图论的考查也是比较深入，一定要明白意图，对症下药！！

* **拓扑排序**

拓扑排序面向有向无环图的形式，若节点U经过若干条有向边后能够到达节点V，则在求得的序列中U必排在V之前，这样便可以得出一定得为有向无环图，按照这种性质可以判定一张图是否为有向无环图。

具体步骤：选择一个入度为0的节点，作为序列的第一个节点，将该点以及以该点为弧尾的所有边删除获得一个新图；再在新图上选择一个入度为0的节点，重复以上操作，知道所有的节点和边均被从图中删除，如果操作完成后仍存有入度不为0的节点，那么说明原图中有环路，故而有原图不为有向无环图。

关于这种问题，难度不是很大，主要是熟悉stl库的操作，如stack、vector的基本操作！！以及拓扑排序的原理~

* **北京邮电大学-哈夫曼树**

题目要求输出节点的值与权值的乘积，仔细分析可以发现就是所有的非叶节点上的权值之和，那么只需要模拟建树的过程后输出非叶节点的权值之和就可以~采用了两种排序方法解决问题，一个是快速排序，一个是优先队列（利用了堆排序），每一次找出当前顶点权值最小的两个节点然后合并为一个树，如此来看，便可以模拟建树的过程去求解问题，题目难度不大，自己细细推演就可以理解其中原理~

* **华中科技大学-二叉树遍历**

给出前序、中序遍历后的结果，输出后序遍历的结果，这是典型的二叉树遍历问题，重点是要熟悉这些方法，以及指针的调用，整体难度不大，但是程序的处理方式很经典，要熟悉~

* **清华大学-二叉树遍历**

给出标准的前序遍历结果，利用前序遍历的结果搭建出二叉树之后后序遍历二叉树输出结果，这是一个标准的遍历题，关于二叉树的遍历问题，难点在于递归的理解，此处一定要多阅读程序，理解其中蕴含的思想！！

* **华中科技大学-二叉排序树**

二叉排序树的构建也有它自己的套路，所以在构建二叉排序树的程序书写上，本道题的程序书写可以说是非常经典的书写方式！！一定要多加看多加理解~

* **华中科技大学-二叉排序树（新）**

在构建出二叉排序树后，还要求得到每个节点的父节点的权值，关于父节点，此处我采取了比较笨的遍历的方法，虽然较笨，但也算是侥幸AC过去！

* **浙江大学-二叉搜索树**

二叉搜索树是指二叉排序树的比对，题目本身其实和之前的二叉排序树思路相仿，但是中间需要用到比较多的字符串比对函数进行比对。首先需要清楚的是二叉搜索树指的是 二叉排序树，对于二叉树的遍历只要中序+任意一个序列遍历两个遍历结果都一致可以说明此二者树是一致相同的。本道题难点在于如何存储状态，此处我采用数组存储，对于树形的建立，其节点每次都会清空，主要存储的是遍历后的结果，本道题用到了一些字符串比对函数，如strcmp()用于字符串的比对，strlen()用于计数字符串的长度，memset()用于字符串的初始化，memcpy()用于字符串的复制，注意这些常用的函数一定要熟悉，否则机试时会大大耗费时间...

本道题我之前踩的坑主要是字符串的结尾处应该加上’\0’结束符，这样字符串的比对才会无误！在某种程度上，这算是一个套路的处理方式，一定要内心清楚！！

* **哈尔滨工业大学-百鸡问题**

水题，暴力求解就好了。

* **胜利大逃亡-广度优先搜索**

迷宫问题，有阻碍有路，这是典型的深搜题，本道题在深搜问题中难度属于较大的，程序写得特别规范，适合模板记忆~ 以构造搜索树的思路去处理深搜问题，用mark标记，用队列存储当前节点状态，不断地遍历，利用了遍历树节点时最先到达的时间是最优的，所以可以每次到达时可以mark下。所以对于此类题，一定要多加记忆！！

* **非常可乐-广度优先搜索**

给出三个杯子容量，和可乐的容量，要求平分可乐

使用四元组（x，y，z，t）来表示一个状态，其中 x、y、z 分别表示三个瓶子中的可乐体积，t 表示从初始状态到该状态所需的杯子间互相倾倒的次数。状态间的相互扩展，就是任意四元组经过瓶子间的相互倾倒而得到若干组新的四元组的过程。这样，当平分的状态第一次被搜索出来以后，其状态中表示的杯子倾倒次数即是所求。

本道题实际上还是脑海中有搜索树的概念，然后利用BFS求解状态，难度不大，注意倾倒必须每次倒满这个前提的调节~

这算是一个模板题型，可以熟悉队列的使用，它在广度优先搜索中有这不错的效果！

* **Prime ring problem**

给出一系列数，将数字按环形排列，要去相邻数之和为素数，这是一个经典的深搜题，就是从第一个节点开始依次遍历，直至找到一个环并输出。对于深搜问题，是一个短板，因为对于递归理解得并不是很好，所以此处还是应该多加注意理解！！细心把玩~

* **Temple of the bone**

典型的深搜题目，就是从一个节点一站到底，只要达到要求即返回，要加上返回条件才能保证深搜过程顺利进行！！是个经典题型，也要多加记忆、理解！！

此处有则细节：主人公每走一步时，其所在位置坐标中，只有一个坐标分量发生增一或者减一的改变，那么两个坐标分量和的奇偶性将发生变化。这样，当主人公走过奇数步时，其所在位置坐标和的奇偶性必与原始位置不同；而走过偶数步时，其坐标和的奇偶性与起点保持不变。若起点的坐标和的奇偶性和终点的坐标和不同，但是需要经过偶数秒使其刚好达到，显然的这是不可能的，于是我们直接判定这种情况下，整棵解答树中都不可能存在我们所需的状态，跳过搜索部分，直接输出 NO。

* **华中科技大学-N阶楼梯上楼问题**

简单的递归问题，就是n阶楼梯的上楼方式个数=F[n-1]+F[n-2]，这在高中也有考查过此类问题，就是整理出逻辑~

* **北京大学-拦截导弹**

简单的最长递增子序列的变种题，难度不大，重要的是理解这种思路！

* **上海交通大学-Coincidence**

最长公共子序列的典型题目，此处基本是原始题型，主要应该注意几个转移条件的原理，知道背后的工作机制！

* **搬寝室**

典型的动态规划问题，给出搬寝室的要求，本处的动归方程不是特别容易想出来，主要是基于两点：

1.选定的最优方案中，任选两对组合，其配对情况必为重量最大物品和重量次大物品为一对、重量最小物品和重量次小物品为一对，不存在交叉组合的情况

2.对于每对组合，配对必须是相邻重量的物品配对，否则属于交叉组合没有意义

基于以上二点可以去构造递归方程，首先选择一个二元方程组去表示递归的状态，本处dp[i][j]的设置含义也很有意思，指前j个物品中选出i对物品时最小的疲劳度

如果j=2\*i,那么j和j-1可以配对；而如果 j>2\*i，那么错过就错过了，dp[i][j]=dp[i][j-1]

故综上，动归方程为 dp[i][j]=min{dp[i][j-1],dp[i-1][j-2]+(weight[j]-weight[j-1])^2

某程序用到了qsort()函数用于快速排序。

* **Greedy Tino**

挑担问题，要求担左右重量一致并且一担的总重量最大，这是典型的动态规划题，但题目本身难度较大！本题只是大致了解了下作者的思路，orz…

因为本题要求解的是重量相等的两堆柑橘中每堆的最大重量，并且在堆放过程中，由于新的柑橘被加到第一堆或者第二堆，两堆之间的重量差会动态发生改变，所以我们设状态 dp[i][j]表示前 i 个柑橘被选择后（每个柑橘可能放到第一堆或者第二堆）后，第一堆比第二堆重 j 时（当 j 为负时表示第二堆比第一堆重），两堆的最大总重量和。

初始时，dp[0][0]为 0，即不往两堆中加任何柑橘时，两堆最大总重量为 0；dp[0][j]（j 不等于 0）为负无穷，即其它状态都不存在。根据每一个新加入的柑橘被加入到第一堆或者第二堆或者不加到任何一堆，设当前加入柑橘重量为 weight[i]，这将造成第一堆与第二堆的重量差增大 weight[i]或减小 weight[i]或者不变，我们在它们之中取最大值，其状态转移为：dp[i][j]=max(dp[i-1][j-weight[i]]+weight[i],dp[i-1][j+weight[i]]+

weight[i],dp[i-1][j])

当根据该状态转移方程求出所有的状态后，状态 dp[n][0] / 2 即是所求

* **Piggy-Bank**

典型的完全背包问题，在0-1背包问题上，对于每个物体的数目不加限制，可以为任意个。

由于每个钱币的数量都可以有任意多，所以该问题为完全背包问题。但是在该例中，完全背包有两处变化，首先，要求的不再是最大值，而变为了最小值，这就要求我们在状态转移时，在 dp[i]和 dp[i-weight[j]]+value[j] 中选择较小的转移值；其次，该问题要求钱币和空储蓄罐的重量恰好达到总重量，即在背包问题中表现为背包恰好装满，为了解决背包必须恰巧装满的处理，我们一开始背包每个容量出都初始化为一个无穷大数，只有查询至该容量即改容量对应有组合时才会更改数值.

本道题引入了最大数，我觉得int型可能最大值为 0X7FFFFFFF!关于INF数值的表述要熟记！

* **珍惜现在，感恩生活**

这是经典的多重背包问题，介绍多重背包问题，其介于 0-1 背包和完全背包之间：有容积为V 的背包，给定一些物品，每种物品包含体积 w、价值 v、和数量 k，求用该背包能装下的最大价值总量。与之前的背包问题都不同，每种物品可选的数量不再为无穷或者 1，而是介于其中的一个确定的数 k。与之前讨论的问题一样，我们可以将多重背包问题直接转化到 0-1 背包上去，即每种物品均被视为 k 种不同物品，对所有的物品求 0-1背包。

1. 简单版；当题目的数据量较小，所以此处这个简单版本就是直接把k件物品视为k种物品展开得到的，如本道题
2. 如果数据量很大的话，那么按照1中的处理方式，会大大增加时间复杂度，于是可以采用改进版本的：我们采用一种更为有技巧性的拆分。将原数量为 k 的物品拆分为若干组，每组物品看成一件物品，其价值和重量为该组中所有物品的价值重量总和，每组物品包含的原物品个数分别为：为：1、2、4...2^c、k-(1+2+4+..+2^c)，其中 c 为使 k-(1+2+4+..+2^c)大于 0 的最大整数，接着就是最后拆出来的一个数是 k-(1+2+4+..+2^c)

这种类似于二进制的拆分，不仅将物品数量大大降低，同时通过对这些若干个原物品组合得到新物品的不同组合，可以得到 0 到 k 之间的任意件物品的价值重量和，大大减少了时间复杂度！

* **北京航空航天大学-字符串的查找删除**

字符串处理问题，本题用到了string stl相关的函数操作，使得程序更加简洁，更容易书写，是很不错的一道题，解法上也很精巧！

string库的find()函数以及 erase()函数，要多了解~

也应该留意gets函数的使用，它可以读入空格，只有遇到换行符才会终止！

了解gets的输入特点，当程序运行至gets语句后，它将依次读入遗留在输入缓冲中的数据直到出现换行符，并将除换行符外的所有已读字符保存在字符数组中，关于gets()函数因为它读取的机制所以要注意它和cin、scanf函数混用时容易发现一些工作机制上的冲突！

* 产生冠军

如果 A 打败了 B，B 又打败了 C，而 A 与 C 之间没有进行过比赛，那么就认定，A 一定能打败 C。如果 A 打败了 B，B 又打败了 C，而且，C 又打败了 A，那么 A、B、C 三者都不可能成为冠军。根据这个规则，无需循环较量，或许就能确定冠军。你的任务就是面对一群比赛选手，在经过了若干场撕杀之后，确定是否已经实际上产生了冠军。

分析可知，这是一个拓扑排序衍生出的题，可以产生冠军的情况即为全图中入度为零的点唯一，本题中比较恶心的地方是只给了字符串去替代人名，我们需要将人名和对应的数字匹配上，才方便进行数组操作。采用了map将选手的姓名映射为结点的编号。C++中，将字符串映射为数字这种典型的映射操作，应该了解map的使用！map有很多高效的处理方式完成这些映射。