# 第十八届西南科技大学正式赛赛题题解

## A.花非花

### 题解：

先构造出一个序列 $A_{2*n+1}=0,a_1,0,a_2,0..0,a_n$，用 $manahcer$ 算法处理出以 $i$ 为中心的最长回文串的长度 $d_i$，因为 $A_{i-d_i+1}\sim A_{i+d_i-1}$ 是回文串，所以对于 $0\leq x\leq d_i-1$，$A_{i-d_i+1+x}\sim A_{i+d_i-1-x}$ 也是回文串，显然这样包含了所有以 $i$ 为中心的回文串，也即以 $i-d_i+1\sim i$ 为开头的回文串数量都 $+1$，可以用差分来处理区间加，时间复杂度 $O(n)$。

## B.为欢几何

### 题解：

本场签到题之一。题意是按顺序输出 $n$ 个字符串的第一个字符。

做法很多种，方法之一是开一个 $char$ 数组 $s$。循环 $n$ 次输入字符串 $s$，每次循环过程输出 $s[0]$ 即可。

## C.花空烟水流

### 题解：

可知，长度为 $l$，由 $m$ 种字符组成的字符串共有 $m^l$ 种，而对于字符串 $s$ 来说，连续子串的数量一定小于 $n$。由此可得，$l$ 最长不会超过 $log_m n$。算出每个 $m$ 的最大 $l$，然后直接暴力bfs搜索即可。最终复杂度为 $O(m^{\lceil log_m n\rceil})$。

## D.似花还似非花

### 题解：

如果没有不可选取 $x$ 的这个限制，显然需要的最小操作次数就是排列长度 $n$ 扣掉排列 $A,B$ 的最长公共子序列长度，对于不属于最长公共子序列的元素，至多只需要1次操作就可以把它挪动到恰当位置。

如果限制了元素 $x$ 不可被选取，那么答案就是排列长度 $n$ 扣掉排列 $A,B$ 的包含元素 $x$ 的最长公共子序列长度。

因此，这个题实际上是求解两个排列的包含某一特定元素 $x$ 的情况下的最长公共子序列长度。

两个排列的最长公共子序列长度问题可以转化为最长上升子序列长度问题。定义$idx[i]$表示元素$i$在数列$B$出现的位置，定义$C[i]=idx[a[i]]$，也就是数列$A$中第$i$个元素所在数列$B$当中的位置，我们可以得到一个新数列$C_1,C_2...C_n$。

这种情况下，求数列$A$，$B$的最长公共子序列问题其实就是求数列$C$的最长上升子序列问题。

对于求解某数列$S$的最长上升子序列问题，有一个动态规划+二分的解法，主要步骤如下：

1.定义一个动态数列$Arr$，$Arr$初始为空

2.从左到右遍历所有元素

2.1 获取当前遍历到的元素$S_i$

2.2 如果该元素比$Arr$中所有元素大($Arr$为空时也成立)，则把它放在数列最右端，跳过2.3

2.3 找到数列中大于$S_i$的最左边的元素，并把它替换为$S_i$

(在整个算法过程中，数列$Arr$永远是单调递增的)

3.此时，数列$Arr$的长度就是数列$S$的最长上升子序列的长度。当然了，$Arr$并不一定是$S$的最长上升子序列，只是长度一致。

该算法时间复杂度为 $O(nlogn)$ ，因为步骤2.3可以通过二分查找来进行优化。

至于为何如此，请读者自行思考。

显然，我们可以通过该方法求解最长下降子序列问题，因为它们性质是一样的。

学会了如何求解最长上升子序列长度，就可以求解本题。

我们创建一个长度为$n$的数组$L$,和一个数组$R$。

我们先从左往右遍历数列$C$，求解该遍历顺序下的最长上升子序列长度，但是在算法步骤的2.2之前，需要统计数列$Arr$当中有多少元素小于当前遍历到的元素$C_i$，并把这个统计结果存入数组$L[i]$当中。

然后从右往左遍历数列$C$，求解该遍历顺序下的最长下降子序列长度，同样地，在步骤2.2之前，统计有多少元素大于当前的$C_i$，将结果存入$R[i]$当中

这样两遍处理完所需要的数据之后，在数列$C$的某个元素$C_i$被固定选取的情况下，数列$C$的最长上升子序列长度就是$L[i]+R[i]+1$ (它本身)。

这样，就可以以 $O(2nlogn)$ 的效率进行预处理，对于每个询问，以 $O(1)$ 的效率查询答案。

总体时间复杂度为 $O(2nlogn+q)$ 。

# E.西楼暮，一席疏雨

## 题解：

一个简单的排列组合问题。

选定一个数，剩下的位置就是一个 $n-1$ 的排列，所以每个元素在式子中出现 $A_{n-1}^{n-1}$ 次，用快速幂求即可，注意欧拉降幂，即 $a^b\%c=a^{b\%\varphi(c)+\varphi(c)}\%c$，其中 $\varphi(c)$ 是欧拉函数；根据排列的对换性质，逆序数为奇数的排列的个数和逆序数为偶数的排列个数一定相等，所以 $\Pi(\sigma)$ 产生的 $2$ 的个数是 $\frac{A_n^n}{2}$。时间复杂度为 $O(n^2\log a)$。

特别的，当 $n$ 等于 $1$ 时特判，此时逆序数为 $0$，所以答案为 $2\times a_{11}\%mod$。

# F.青山隐隐，败叶萧萧

## 题解：

本题属于简单题。题意简化后如下：任意多次操作后，能否使得一个序列任意相邻两数之和为是质数，且每个数都是质数。

前置知识：奇数 $+$ 奇数 $=$ 偶数，奇数 $+$ 偶数 $=$ 奇数，偶数 $+$ 偶数 $=$ 偶数； $2$ 是最小的质数，也是唯一的偶质数。

又有无穷多个数满足：自身是质数，自身 $+2$ 仍是一个质数。这样的例子有很多，例如：$\{3,5,11,17,29,\dots\}$ 这些数都满足该性质。

所以题目一下就简单起来了：只要该序列是 "奇数偶数交替" 排列的数即可构造出满足题意的序列。

从第二个数开始遍历序列，判断是否满足 $(a_i+a_{i+1})$ 为奇数即可。

只有一个数的时候，则一定可以满足。

总时间复杂度为 $O(t\times n)$ 。

# G.几番烟雾，只有花难护

## 题解：

由题意可知，本题要求的是 $\sum_{i=1}^{n} i^2\lceil\frac{n}{i}\rceil$ 。

正常暴力跑，每一次单独计算答案，则时间复杂度为 $O(t\times n)\approx 10^{11}$ ，显然会超时。

考虑使用整除分块的思想处理该问题。

设一共有 $x$ ( 表示 $n$ 的因数个数 ) 个数满足 $n\%i=0$ ，我们令 $D_n$ 为这 $x$ 项之和，即令 $a_i$ 表示正整数 $n$ 的第 $i$ 个因子，则 $D_n=\sum_{i=1}^{x} a_i^2$ 。我们将原式的每一项的 ( $\lceil\frac{n}{i}\rceil$ ) 部分均看成不能整除的结果，则每一项的该部分均可以改写成 ( $\lfloor\frac{n}{i}\rfloor+1$ )，故答案只需要在此基础上减去 $D_n$ 即可。

根据先前推理，则所求原式可做以下变式：

原式 $=\sum_{i=1}^{n} i^2\lceil\frac{n}{i}\rceil$

$=\sum_{i=1}^{n} i^2(\lfloor\frac{n}{i}\rfloor+1)-D_n$

$=\sum_{i=1}^{n} i^2+\sum_{i=1}^{n} i^2(\lfloor\frac{n}{i}\rfloor)-D_n$

我们将该式子化为三个部分：

①： $D_n$ 部分，已经推得 $D_n=\sum_{i=1}^{x} a_i^2$ ，计算 $D_n$ 的一种方法是， $O(sqrt(n))$ 暴力遍历 $n$ 的所有因子，计算所有因子的平方和。最后 $D_n$ 注意要边累加边求余。

②： $\sum_{i=1}^{n} i^2$ 部分，设 $S_n=\sum_{i=1}^{n} i^2$ ，这显然是一个平方数前 $n$ 项和公式。由多种方法均可推导出 $S_n=\frac{n(n+1)(2n+1)}{6}$ ，故直接计算值即可。但由于模运算中并不存在除法的性质，所以注意使用乘法逆元。

③： $\sum_{i=1}^{n} i^2(\lfloor\frac{n}{i}\rfloor)$ 部分，设 $T_n=\sum_{i=1}^{n} i^2(\lfloor\frac{n}{i}\rfloor)$ ，对于每个块均有：

$\sum_{i=l}^{r} i^2(\lfloor\frac{n}{i}\rfloor)$

$=\sum_{i=l}^{r} i^2(\lfloor\frac{n}{l}\rfloor)$

$=\lfloor\frac{n}{l}\rfloor\sum_{i=l}^{r} i^2$

$=\lfloor\frac{n}{l}\rfloor\times(\frac{r(r+1)(2r+1)}{6}-\frac{(l-1)((l-1)+1)(2(l-1)+1)}{6})$

该计算过程中，仍需要注意边累加边求余，且也需要使用乘法逆元，也需要注意括号内可能为负数，使用减法求余。

综上所述，可得原式 $=S_n+T_n-D_n$ 。令 $ans=S_n+T_n-D_n$ ，仍需要注意的是，该项有可能为负数，使用减法求余，应输出 $(ans+mod)\%mod$ 。

计算因子和整除分块的时间复杂度均为 $O(sqrt(n))$ ，故总时间复杂度为 $O(t\times(sqrt(n)))$ 。

最后注意：需要预处理 $6^{mod-2}$ 。若把他写在 $cal$ 函数中，则会被调用 $sqrt(n)$ 次，其本身时间复杂度就达到 $log$ 级别，会造成超时，总时间复杂度 $O(t\times(sqrt(n))\times\log mod)$ 。

# H.岸风翻夕浪，舟雪洒寒灯

## 题解：

$Hint\ 1$：序列 $S_n$ 的长度为 $2^n-1$；

$Hint\ 2$：对于 $N$ 位二进制数，$000...000 \sim 011...111$ 和 $100...000 \sim 111...111$ 的后 $N-1$ 位是相等的；

$Hint\ 3$：对于 $N$ 位二进制数，$000...000 \sim 011...111$ 共有 $2^n-1$ 个数；

结论：第 $K$ 位上的值即为 $K$ 二进制位最低位 $1$ 的位置。

假设对于 $S_n$ 结论成立，那么对于$S_{n+1}$

①：序列前 $2^n-1$ 项与 $S_n$ 相等，满足结论；

②：序列后 $2^n-1$ 项等于前 $2^n-1$ 项，由 $Hint\ 2$ 可知满足结论；

③：序列第 $2^n$ 项的值为 $n+1$，$2^n$ 的最低位 $1$ 的位置为 $n+1$，满足结论；

(1)：由 ①②③ 可知，即当 $S_n$ 满足结论时，$S_{n+1}$ 也满足结论；

(2)：对于 $S_1$，第 $1$ 位的值为 $1$，满足结论；

由 (1)，(2) 可得结论成立；

可以通过位运算或者 `__builtin_ffsll()` 计算 $K$ 的最低位 $1$ 的位置，时间复杂度 $O(logK)$。

# I.醉漾轻舟，信流引到花深处

## 题解：

这是一道二分答案 $+$ 折半搜索的题型。折半搜索的部分是板子，但是这个算法比较冷门。二分答案倒是不难想到。

首先是二分答案，我们假定一个答案 $w$，如果得到的方案数未达到 $k$，那么往小调整答案区间，否则往大调整。

关键在于如何在限定时间内统计出可行购买方案数：

至多有 $30$ 个物品，可以考虑把这 $30$ 个物品分成较为平均的两部分$A$、$B$，然后购买情况有四种：

$1$ .$A$ 中什么都不买，$B$中也什么都不买，显然只有 $1$ 种方案；

$2$ .$A$ 中购买若干件物品，$B$中购买若干件物品；

$3$ .$A$ 中购买若干件物品，$B$中什么都不买；

$4$ .$A$ 中什么都不买，$B$中购买若干件物品；

于是，我们可以对$A$、$B$ 两个部分，分别进行 $DFS$，枚举所有的购买情况下花费的金额，每个部分至多只有$2^{15}=32768$ 种方案。

那么我们可以很快统计出情况 $3$ 和情况 $4$ 的方案数。至于方案 $2$，我们先对$A$，$B$ 枚举得到的金额花费列表进行排序，然后我们枚举$A$ 中的购买方案，再根据剩余的金额去二分搜索出 $B$ 中有多少种购买方案，把方案数计入统计结果。枚举完之后得到的就是情况 $2$ 的所有方案数。

最终将4种情况的方案数加起来就是对于这个 $w$ 得到的最多方案数。

折半搜索部分，复杂度为 $O(2^{\frac{n}{2}} \times log2^{\frac{n}{2}}) = O(2^{\frac{n}{2}} \times \frac{n}{2})$。

因为外层还有一个二分答案，$w$ 的取值最大可能取到 $10^9$，因此总体时间复杂度 $O(log10^9 \times 2^{\frac{n}{2}} \times \frac{n}{2})$。

# J.满城烟水月微茫，人倚兰舟唱

## 题解：

我们可以将牌堆看成 $n$ 个队列，然后直接模拟即可。总时间复杂度为 $O(m \times n)$ 。

# K.对潇潇暮雨洒江天，一番洗清秋

## 题解：

最短路问题，关键在于如何存这个图，另外还需要注意到的可能有些圆上没有点。

假设所有点都在前 $r = 5$ 的圆内，且每层均匀分布，那么建边的复杂度为 $O(n^2)$，空间就已经爆了。可以发现，一般的，同一层的所有点都可以通向邻层上的所有点和次邻层上的所有点，而这些边显然都是冗余的，不难想到创建一个层结点，层结点到该层的所有点的花费为 0，一层上的所有结点（不包括层结点）到邻层、次邻层建边，注意与层结点相连的边都是有向的。

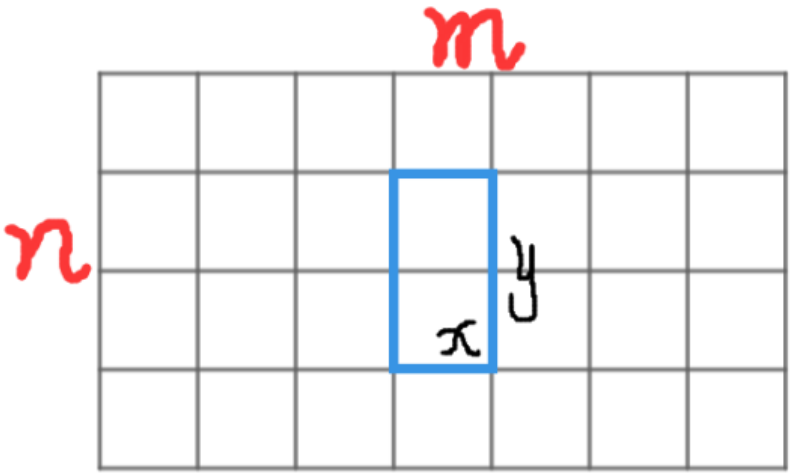
$r = 1$ 上的所有点只有出边，$r = n$ 上的所有点只有入边。假如“始元”上有 $k$ 个结点，这些点都是源点，那么就需要跑 $k$ 次最短路，假设最短路用优先队列优化的 Dijstra 算法，那么时间复杂度为 $O(k \times E \log V)$。最后所求的答案是所有源点的最短路的最小值，可以发现，对于最终的最短路径，不可能有从一个源点经过另外一个源点的最短路。所以可以保存所有"始元"上的点到其他"元"上的点的花费，省去"始元"中的其他点，即把这些 $k$ 个点"压缩"成一个点，保留这个点到其他层的点的边。这样时间复杂度为 $O(E \log V)$。

~~还有~~比较阴间的是，存在一种情况，点分布在 $1\ 2\ 4\ 5 \cdots 9995\ 9997\ 9998\ 10000$，，这样就需要 $p \times 3333 \times 3$ 点体力值，$1061109567$ 在这个范围内，也就是卡了初始化无穷大的值。

# L.夜暗方显万颗星，灯明始见一缕尘

## 题解：

计算一个 $n \times m$ 的矩形网格中所有矩形的数目，即计算 $(n + (n-1) + (n-2) + \ldots + 1) \times (m + (m-1) + (m-2) + \ldots + 1) = \frac{n(n+1)}{2} \frac{m(m+1)}{2}$ 的值。(每一种长度的边都能有一次贡献，长和宽相乘即为答案) 。

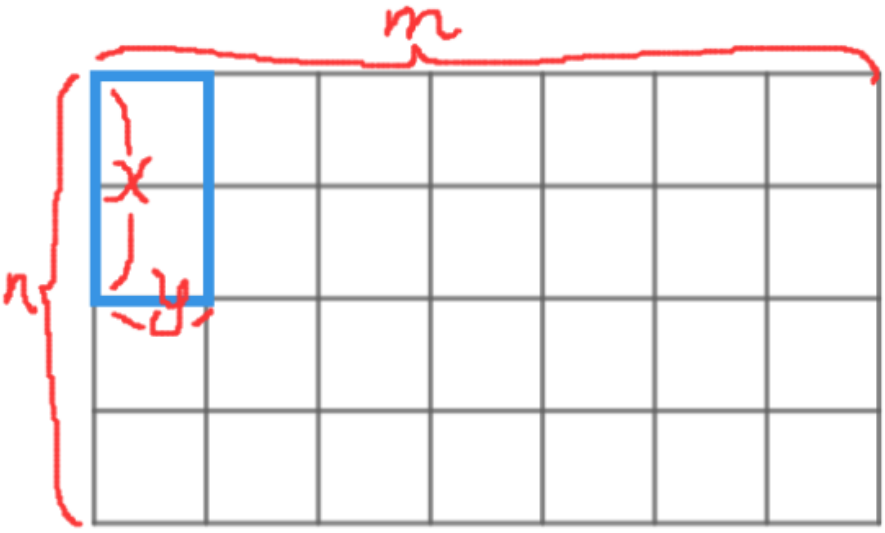



设挖去一个坐标为 $(x, y)$ 的单位格子，包含点 $(x, y)$ 的矩形数目就是：$x \times (n - x + 1) \times y \times (m - y + 1)$ ，由均值不等式可得，当 $x$ 和 $y$ 越靠近中心时，这个值越大。同理可得，把被挖去的单位格子拓展为矩形，很容易得出，包含以 $(L, D)$ 为左下角，$(R, U)$ 为右上角的矩形的方案数 $= L(n - R + 1) \times D(m - U + 1)$ 。对这个挖去矩形的左侧共有 $L$ 个点，右侧共有

$(n-R+1)$ 个点。设左右相加的值为 $Z$，$a=L, b=(n-R+1)$，可得
$Z=(n+1)-(R-L)$ 为一个常量。设 $a+b=M$，则 $b=M-a$。可得

$ab=a(M-a)=aM-a^2=-(a-\frac{M}{2})^2+\frac{M}{4}$。再次使用均值不等式可得，当且仅当 $a=b$ 取最大值为 $\frac{a^2+b^2}{2}$。由于要让被挖去矩形的影响最小，所以 $a$ 和 $b$ 的差距要尽量大。左右如是，上下亦同理可得。综上所述，将被挖去矩形放在角落是最佳选择。

现分析将矩形"横"放还是"竖"放。计算 "L"型矩形网格的总矩形个数方法有很多，如图所示：




该"L"型矩形网格中矩形总数

$=\frac{(n-x)(n-x+1)}{2}\frac{m(m+1)}{2}+\frac{(m-y)(m-y+1)}{2}\frac{n(n+1)}{2}-\frac{(n-x)(n-x+1)}{2}\frac{(m-y)(m-y+1)}{2}$。

$n+m-x-y$ 是一个定值。二次均值不等式及简要证明后可得以下结论：

$(n-x)$ 和 $(m-y)$ 的差的绝对值越大，总矩形数越多。例如上图"竖"放，
$(n-x)=2,(m-y)=6$。若是"横"放，则 $(n-x)=3,(m-y)=5$。将其带入上式，答案分别为 $231$ 和 $228$。

故最终只需要比较 $(n-x)$ 和 $(m-y)$ 的差的绝对值，取大的那种组合，带入式子即可得出答案。

注意要使用 $longlong$ 哦~

# M.劝君终日酩酊醉，酒不到刘伶坟上土

## 题解：

一个很简单的签到题。计算过程中注意要使用 $long\ long$ 防止数据溢出。计算答案的方法若选择暴力累加，则造成超时 (时间复杂度为 $(O(t\times(2min(n,m)-1)))$)，需要使用等差数列求和公式减少运算量，时间复杂度为 $(O(t))$。

可以发现，决策只和第一次拿行还是列有关，与第几行，第几列都无关。如果行数大于等于列数，则先拿列，反之先拿行。

假设 $k$ 很大，易证得，只有前 $2\times min(n,m)-1$ 次拿取是有效的，后面的拿取获得的酒均为 $0$。

设酒桌大小的两个参数中 $n\le m$ (即默认列 $\ge$ 行，只是为了方便书写，毕竟参数顺序不影响解题)

设 $k$ 次操作 $(k\le 2n-1)$ 里，奇数编号 (拿行) 的操作次数为 $k_1$，偶数编号 (拿列) 的操作次数为 $k_2$。则有

$$\begin{cases} k_1 + k_2 = k & (\text{通}) \\ k_1 - k_2 = 1 & (k\%2 == 1) \\ k_1 - k_2 = 0 & (k\%2 == 0) \end{cases}$$

拿行获取的酒的数量 $w_1 = (m + (m-1) + \ldots + (m - k_1 + 1))$。可以看出，这是一个首项为 $((m - k_1) + 1)$，末项为 $m$，公差为 $1$ 的等差数列的总和。由等差数列求和公式可得，$w_1 = \frac{2mk_1 + k_1 - k_1^2}{2}$。

若 $k_2 = 0$，则拿列获取的酒的数量 $w_2 = 0$。

若 $k_2 > 0$，则拿列获取的酒的数量 $w_2 = ((n-1) + (n-2) + \ldots + (n - k_2))$。可以看出，这是一个首项为 $(n - k_2)$，末项为 $(n-1)$，公差为 $1$ 的等差数列的总和。由等差数列求和公式可得，$w_2 = \frac{2nk_2 - k_2^2 - k_2}{2}$。(而实际上也并不需要特判，推理出的式子 $w_2$ 当 $k_2 = 0$ 时其值也为 $0$，为了保持逻辑清晰度分类讨论)

所以最终输出结果即为 $w_1 + w_2$。由于 $k1, k2$ 一定都不会不大于 $min(n, m)$，所以运算过程并不会溢出 $long\ long$ 的范围。

附带检验 (**非严谨检验，只是代值计算**)：当 $k_1 = n, k_2 = n - 1$ 时，此时所有的酒都会被拿完，令 $K = n - 1$。

$\therefore n \times m = w_1 + w_2$

$= \frac{2m(K+1) + (K+1) - (K+1)^2}{2} + \frac{2nK - K^2 - K}{2}$

$= -K^2 + (m + n - 1)K + m$

把 $K = n - 1$ 带入得：

原式 $= -(n-1)^2 + (m + n - 1)(n - 1) + m$

$= -n^2 + 2n - 1 + n^2 - 2n + mn - m + 1 + m$

$= mn = n \times m$，代值检验完毕。