“红旗杯”第十五届东北地区大学生程序设计竞赛

The 15th Northeast Collegiate Programming Contest

题解

预期题目难度

very-easy: I

easy: E K M

medium-easy: C J

medium: A B D H

meduim-hard: F

hard: L G

实际上好像B,H,G过的差不多,C题过得有点少了

40分钟被开出来七个题,一度以为要被ak了..

预期榜单情况

冠军:10-11题

金:6题

银:4题

铜:3题

虽然题目难度预测歪了,但是榜单好像还是正的.金银大概要手速快一点或者多一题的样子.

I: Takeaway

热身赛结束后加的签到题,看起来效果不错,不然可能有好多队签不上到?

E: Easy Math Problem

一个明显的构造是 $6p = p + 2p + 3p$.

出题时的构造方式：找到一个大于p的2的幂次2^k. 那么$2^k p = p + p + 2p + 4p + \ldots + 2^{k-1}p$. 将第一个p按照二进制拆分.($13 = (1101)_2 = 2^3 + 2^2 + 2^0$). 由于$2^k \geq p$, 这里用到的所有数字都是$2^k p$的因子.

K: City

很套路的一个问题，将所有询问从小到大排序，建立最小生成树的同时离线回答每个询问即可。

M: Master of shuangpin

按照题意进行模拟，一个思路是，先判断开头有没有声母，根据声母是单个字符还是两个字符（sh，ch，zh）进行拆分。

有声母的话，将声母部分和韵母部分分别转化即可，没有声母的时候要注意一些特殊情况。

挼（ruó，ruá）
啊（à）饿（è）哦（ò）昂（áng）

实际上，将双拼的转移表建出来，根据字符串长度从长到短匹配也可以。

M: Master of shuangpin

出题时,数据是根据domjudge的规范生成的,在赛前一周才通知搬运到hdu.由于hdu不支持python,搬运的过程中错误地使用了带行末空格的C++代码来生成数据.

由于hdu是逐字符比对选手输出与标准输出,所以好像出现了很多pe???

造成的不便请大家谅解.

C: Vertex Deletion

树形dp，要注意每个节点所在子树，这个节点有三种状态：删除、保留并且至少与一个子节点相连，保留并且不与任何一个子节点相连。

将上述三种状态分别记为$dp[x][0/1/2]$

状态转移方程如下:

$$dp[x][0] = \prod_{v\ is\ a\ son\ of\ x} (dp[v][0] + dp[v][1])$$

$$dp[x][2] = \prod_{v\ is\ a\ son\ of\ x} (dp[v][0])$$

$$dp[x][1] = \prod_{v\ is\ a\ son\ of\ x} (dp[v][0] + dp[v][1] + dp[v][2]) - dp[x][2]$$

J: Transform

这题的解法非常多，包括但不限于：

1. 模拟退火、爬山等随机算法
2. 坐标变换
3. 牛顿迭代
4. 几何法

一个简单的几何法可以参考“罗德里格旋转公式”，手推也是非常好推的。

A: Matrix

首先,$1..n$每个数字的位置可以随意调换而不影响$\left|S\right|$的值,$n+1..n^2$也没有影响.因此最终答案需要乘$n!*(n^2-n)!$

接下来,分别考虑$1,2,..,n$每个数的贡献.

考虑数字i提供的贡献,即这一行的其他数字都要比他大,因此共有$C(n^2-i,n-1)$种不同选法.而数字i在这一行中有n种选择.

$ans=\sum_{i=1}^{n}n*C_{n^2-i}^{n-1}*n!(n^2-n)!$

可能有些队伍有TLE的情况? 把式子化简一下可以消去很多项,std只需要300ms就能通过.(如果处理阶乘的时候分段打表还能更快2333)

(upd:还可以直接打答案的表)

A: Matrix

出题人的做法是考虑交集大小为i时的贡献,需要容斥然后推式子再跑NTT...

阶乘可以通过分段打表或者多项式多点求值求出.

又注意到当$n^2-n\geq 998244353$时,由于最后需要乘$(n^2-n)!$.因此答案一定为0.

所以这题最开始的数据范围是$1e7$....

B: Cypher

题目的来源是Steam的Cypher游戏(

难点主要在于理清Enigma内三个部件的功能,需要注意的点在hint里都有标注.

结合给出的图片可以更快的理解题意.

把题意理清楚实现就不难了,std的代码只有不到2k,通过的提交也都不长.

D: Lowbit

注意到每个数字最多需要操作log次,之后的操作就是直接乘2了.

所以只需要在普通线段树的基础上,加上是加$lowbit$还是直接乘2的标记就可以了.

注意对原数的取模需要在这个数变成$100000..000_2$后才能进行,这与它本身是否小于998244353无关.

H:loneliness

一道构造题,出题人觉得这题不难,但是验题时大家都说不好想.

每次向下走时,loneliness的值需要乘2,100次乘2会使得数字变得特别大.有队伍直接输出-1,应该也是注意到这一点.

实际上,只要我们能构造出一个0,就可以解决这些乘2带来的问题了.

如果能注意到,绕$DRUL$走一个环,可以使当前数字减1,绕$RDLU$走一个环,可以使当前数字加1.(其他的环也都是± 1或者± 2,我们称这些环为一个loop)

那么构造出0就很容易了.

注意到,每当我们从倒数第二行走进最后一行,手上的数字一定会是一个偶数,而在最后一行无论怎么左右移动,数字还会是一个偶数,因此可以确定的是,在不能走出去的限制下,所有的奇数都是不能构造出来的.

对于比较小的偶数(小于200),我们可以通过0一直走到最下面一行,在向右走的过程中,不断地通过-2的loop抵消影响就好了.

对于更大的偶数,由于对字符串长度有限制,不能通过$+2$的loop来构造.

这时可以通过前面的乘法操作来快速的让数字变大,只需要在合适的时候将0变回1,每次向下走之后可以选择走一个$+1$的loop或者不走,我们便有了$f(x)=2x$和$f(x)=2x+1$两种操作,有了这两种操作很容易想到通过二进制的方式进行构造.

可以通过二进制的方式,在左下角构造出一个$k-198$,再一路向右走就可以了.

注:

构造过程只会用到左边两列和下面两行,不过验题的时候似乎也有沿着对角线走的方法.

本题原本的题意的数字全程不能超过k(现在是$2k$),在这个限制下,数字2是无法被构造出来的.不过由于很多构造方法中会出现$k+1$或者$2k-2$等中间结果,因此把限制放宽到了$2k$.

F: Permutation

解法一：我们要对下标维护一个Splay，值维护一个Splay，对于每一个类型3的操作，我们新建两个结点分别对应下标结点和值结点，并且在这两个结点之间**互相保存一个指向对方的指针**，然后将下标结点插入到下标Splay，值结点插入到值Splay。注意不管是下标还是值，都是由在树中的相对位置决定的（中序遍历第几个访问的）。这样对于5个操作分别有：

1. 直接在下标Splay里翻转就行了（打reverse标记）
2. 直接在值Splay里翻转就行了（打reverse标记）
3. 新建两个结点，建立指针关系，分别插入到对应的Splay中

4. 在下标Splay中走到第i个位置，由指针得到对应在值Splay中的结点，将这个结点伸展到根，左子树大小+1就是所求的值了。
5. 在值Splay中走到第v个位置，由指针得到对应在下标Splay中的结点，将这个结点伸展到根，左子树大小+1就是所求的下标了。

时间复杂度$O(m \log (n+m))$

这两个Splay是一样的，而且指针在新建结点的时候设置好以后也不需要维护，所以是不是贴个带区间翻转的Splay板子就能过？？！！

解法二：块状链表，需要维护的标记与Splay一样，时间复杂度$O(m\sqrt{n+m})$

G: Ball

多个验题人都表示不会这道题,就把他的难度估计放在了这里.

所有落到斜面外的球,一定都是从一个连续的填满的前缀里放下去的.

枚举从下往上第一个没有球的洞,就可以将整个区间分成两半,所有落到斜面外的球都在下面一半.

记录$f[i][j]$为在长度为i的槽里扔j个球,没有球落出去的方案数.

记录$g[i][j]$为在长度为i的槽里扔j个球,能够完全填满i个槽的方案数.

之前的区间被分成了x和$n-x-1$两段.

就会有$g[x][x+k]*f[n-x-1][m-x-k]*C_m^{x+k}$种方案.

考虑如何求解f和g.

对于$f[i][j]$,共扔出了j个球,所以最后的答案需要乘$j!$.

考虑共有k个球在第i个位置落下,有一个落入第i个槽中,另外$k-1$个会填到前面的槽里.
由于这k个球之间没有下落的先后顺序,因此方案数需要除以$k!$.

即:$f[i][j]=j!\times\sum f[i-1][j-k]\times\frac{1}{k!}$

对于$g[i][j]$,同样考虑共有k个球在第i个位置落下,在$g[i-1][j-k]$,即前$i-1$个槽被全部占满的情况下,会有$k-1$个球落到斜面外.

$$g[i][j]=j!\times\sum g[i-1][j-k]\times\frac{1}{k!}$$

两个状态转移方程是相同,但在实际计算的时候,需要根据状态的实际意义确定各自的枚举范围.

$O(n^3)$的时间复杂度将所有f和g预处理之后,就可以$O(n)$回答每一组询问了.

注意到两个状态转移方程都可以通过FFT进行优化,因此这道题可以做到$O(n^2logn+Tn)$

L: Kth Smallest Common Substring

后缀自动机：

先将所有的字符串翻转，对第一个字符串建后缀自动机，其他的串在上面跑，记下来每个结点在多少个字符串里出现。

然后对所有出现了n次的结点按照字典序排序，对结点所能表示的字符串个数求前缀和，每一次询问就是在前缀和数组上二分。

题目要求找最左侧的出现位置，翻转之后就是最右侧的出现位置，在parent树上向上合并取最右的位置即可。

后缀数组:

首先将所有字符串拼接起来连成一个长串，记录下每个字符都是哪个字符串的。对这个串建立后缀数组。

那么对于排好序的所有后缀，如果区间 $[l, r]$ 内 1 到 n 所有字符串都出现了至少一次，那么这一段的 LCP 的所有前缀便都属于相同公共子串的集合。

由于后缀数组中所有的后缀是按照字典序排序的，所以很容易可以解决字典序第 K 的问题。

考虑使用双指针来维护 1 到 n 所有字符串都出现了至少一次。当左指针 l 移动到 $l+1$ 时，右指针移动到 r'，那么这两次的 LCP 有可能会有相同的前缀。那么显然相同的部分即为 l 到 r' 的 LCP 。因此可以很容易的统计出个数。

此时将询问离线，将所有询问按从小到大顺序排序，那么在双指针移动的时候即可算出满足要求的一个串的位置。

但题目要求的是在第一个串中第一次出现的位置，而上述求的可能是任意一个串的任意一个位置。因此需要在后缀数组上二分，找到包含此子串的完整区间，在其中进行 RMQ 即可找到第一个字符串中第一次出现的位置。