Day1 Solution

A. Miner's Necklace

记 $B(P)$ 为一个新的数字串，如果存在最大的 $j < i, P_j = P_i$，$B_i = i - j$，否则 $B_i = |P|$。

容易发现 $B(Q) = B(P)$ 是 P 和 Q 弱匹配的充要条件。

然后我们从左到右枚举 S 的每一个长度为 $|P|$ 的子串，求出它的 B 的哈希值，和 $B(P)$ 的哈希值去比较统计答案。

如何维护哈希值呢？其实只要考虑删除头，增加尾，B 会如何变化就可以了。

设删去的头是 c，若在第 2 到第 n 个字符中有 c，则其中最前面那个 c 的 B 值要改为 n。其余的 B 值满足 $B'_{i-1} = B_i$。而B'_n可以快速求出，如果有修改的话也只需要修改一处，所以说可以 $O(1)$ 地由上一个哈希值求出下一个哈希值。

id	字符集	大串长度	小串长度	备注
1	2	505	6	随机
2	2	852	2	随机
3	2	1000000	10	随机
4	2	1000000	50000	大部分是5，少量是1
5	1	1000000	500000	全是5，标算应该最慢
6	233	1000000	123456	按照周期233循环
7	1000	1000	2	随机
8	1000	237637	118766	两次长串匹配，中间加杂碎
9	10000	312989	1316	233次短匹配，中间加杂碎
10	3	3	3	"1 2 3", "1 2 1"

B. Monster

题意

给一棵树，一些点是补给点。从 1 号根节点往下走，每次等概率选一个儿子走。走到一个节点上有 P_i 的概率存活，失败了就返回上一个通过的补给点再往下走，总共 K 次复活机会，问走到底的概率。

题解

设 $f_{j,i}$ 为从 i 点出发还剩 j 条命，且还没打过第 i 个点时走到底的概率
$g_{j,i}$ 为从 i 点出发还剩 j 条命，且已经打过第 i 个点时走到底的概率
cnt_i 为 i 点的儿子数，s_i 为 i 点之前的补给点
对于一般节点可得
$f_{j,i} = P_i * \frac{\sum f_{j,son}}{cnt_i} + (1 - P_i) * g_{j-1,s_i}$
$g_{j,i} = \frac{\sum f_{j,son}}{cnt_i}$
如果没有剩余生命了 则 $f_{0,i} = P_i * \frac{\sum f_{0,son}}{cnt_i}$
对于叶子节点
$f_{j,i} = P_i + (1 - P_i) * g_{j-1,s_i}$
$g_{j,i} = 1$
于是从下往上推就可以了
树可能是一条链，DFS 下去会很深，于是先跑一遍 BFS 遍历得到每个点的遍历顺序 然后再对着遍历顺序求 f 和 g
将生命值作为外层循环，注意到需要 g_{j-1,s_i} 时 g_{j,s_i} 是未被修改过的，于是可以将剩余生命那一维去掉，跑 K 次，最后一次的 f_1 就是答案
注意到式中需要的除数小于 N，而给出的 P_i 的除数小于 $1e6$
于是 O(1e6) 预处理逆元即可

C. ZYH's Tree

题意

给定一个n个点的树，每个点都有一个颜色c，定义一条路径的权值为其路径上不同颜色的种数，求所有路径的权值之和。有m次修改操作，每次操作修改一个点的颜色，每次操作后都输出所有路径的权值之和。

题解

LCT

我们首先将所有的颜色分开考虑，现在我们考虑某一种颜色对答案的贡献，也即统计所有经过至少一个该颜色的点的路径个数。由于这个统计较为复杂，所以我们考虑统计更简单的：统计没有经过该颜色的点的路径条数。我们很容易发现，路径条数即删除所有该颜色点后，剩余的联通块大小的平方的和。

我们考虑如何维护这样的一个东西。

我们构造这样一个新图：图上的点与父子关系与原图对应，将当前考虑的颜色的点设为白点，其他颜色的点设为黑点，于是我们需要求的便是所有黑色点构成的联通块的大小的平方和。

为原本的根结点虚拟一个白色父亲节点，并使除了虚拟节点之外的所有点中的所有黑点向父亲连实边，所有白点向父亲连虚边。同时我们在每个点上维护两个值$size, size2$ ，$size$表示该点所在联通块的大小，$size2$表示该点所有的儿子所在的联通块的大小的平方的和。并且我们只对实边进行更新。

定义一个联通块的顶点为该联通块深度最小的那个点的父亲，那么我们发现，答案即为所有黑色联通块的顶点的$size2$的值的和。

接下来我们考虑颜色的修改操作，对应到对于单个颜色的考虑，即为：将一个黑点变成白点、或将一个白点变成黑点。

1. 将一个黑点变成白点：首先我们将该点连向父亲的边变为虚边，删除该节点对父亲的贡献。然后我们考虑答案的变更，该联通块之前对于答案的贡献为为联通块的大小的平方，之后的贡献为两个联通块的大小的平方的和。
2. 将一个白点变成黑点：首先我们将该点连向父亲的边变为实边，将该节点对父亲的贡献计入父亲节点。然后我们考虑答案的变更，该联通块和即将连接的联通块对答案的贡献之前是分别的平方的和，连接之后是对答案贡献连接后的联通块的答案的和。

我们可以用LCT维护所有实边构成的森林来维护上述过程

对于每个节点维护$size$与$size2$，这样每一次的节点颜色的变更操作我们都可以用LCT来维护

1. 将一个黑点变成白点：由于我们使用LCT维护实边组成的森林，所以在这里进行cut操作，并维护相关值的变化
2. 将一个白点变成黑点：对应的，在这里进行link操作，维护相关值的变化

将一开始n个点具有的颜色转化为n次在时间为0时进行的操作，每次求出答案的变化值。

单次操作复杂度$O(\log n)$，总复杂度$O((n+m)\log n)$

D. ADS

这其实不纯粹是一道爆搜题目，而是要你自己去慢慢发现你原来写的爆搜会被什么样的数据卡掉然后再去做修改。

1. 首先考虑暴力，枚举一个元素的01状态，枚举完之后利用边的关系再确定一部分一部分元素的01关系。
2. 确定一个点的01状态后，显然有一些边的约束就没用作用了，将其暂时删去，并且当一个点无边连出的时候，不去枚举该点，直接将答案乘2.
3. 对于多个连通块，答案显然满足乘法原理。 满足上述简单三条其实已经可以通过大部分数据了，接下来需要感受一下还有类似什么样的数据能对你造成麻烦，首先直观感觉有用的约束太多的图显然是可以通过的因为这样均摊下来 所需枚举的点的个数<=n/2,而边过于稀疏时例如一个圈，一条链，或是全部孤立点，我选取几个点之后整张图就会变成很多的连通块似乎也还可以解决。 但是对于一张

图的约束既不稀疏又不稠密（大概每个点连有3条左右的约束的时候）这样似乎也不太行（而且每次枚举完点都重新更新连通块好像也很麻烦） 考虑一个稍微优雅一点的解决方案（一点也不优雅），对于当前枚举到的点如果他已经在一棵树里面了，那么直接树形dp，这样就可以跑的非常优越了。

E. Wheat

考虑$1,\ldots,i$这个前缀。

每次从前缀里面拿走的都是$max\{a_1,a_2,\ldots,a_i\}$。

那么当拿走的数比a_i大时，i位置上的数是不会变的。

不妨设a_i是第k大，那么前k次拿之前都不会变。

第$k+1$次拿之前，a_i上面就变成了第$k+1$大，并且之后每次a_i都会变小1名。

也就是说，对于询问x,y，答案就是$min\{a_y,xthmax(1,\ldots,y)\}$

然后就是带修改第k大

复杂度$O(n\log^2 n)$

F. Company

首先我们设$s_i=\sum_{i=0}^{i}a_i$，$x=\lfloor\frac{l}{n}\rfloor,y=\lfloor\frac{r}{n}\rfloor$，那么每个约束的形式就是$s_j-s_{i-1}+(y-x)s_{n-1}\le w$或者$s_j-s_{i-1}+(y-x)s_{n-1}\ge w$，那么对于$0,\ldots,n-1$这些点显然是一个差分约束系统。

那么暴力就是可以去枚举s_{n-1}然后判断约束系统是否矛盾，即是否有负环。

但是这样显然不太行。

注意到所有的约束都是一次不等式的形式，也就是说每个环型约束非负的条件一定是$[l,\infty]$或者$[-\infty,r]$的形式，那么所有约束成立的条件一定是$[l,r]$的形式。

我们就可以二分去求这个左右边界，然后注意到我们每次是找到一个负环，之后对于s_{n-1}的改变要根据负环上面s_{n-1}前的系数和$\sum k$来决定s_{n-1}应该更小还是更大，如果$\sum k=0$，就看$\sum b$，如果$\sum b>0$说明满足的s_i有无数种，否则就为0。

这样不断操作每个环都会被满足，所以就能满足所有约束。

复杂度$O(nm\log V)$