1001

题解

首先，要求生成的图甚至不要求联通，所以通过1与$2,3,4$连边，5与$6,7,8$连边等，容易构造出$n/3$鸡爪的图。难点在如何让字典序最小。

直觉的想法是让与1连的点尽可能多。如果n模3有余数，则直接与1连接。下面仅考虑n为3的倍数的情况。

对于一个给定图，给每个结点标号，为了让输出的边字典序最小，发现度数越多的点，标号越小。

为了生成一个字典序最小的图，我们要让结点1的度数尽可能多，相同时要结点2的度尽可能多，以此类推。因此，一个贪心的想法是，让$1,2,\ldots,n/3$为鸡爪中心点。

$4,5,\ldots,n/3$的中心点分别与$1,2,3$相连，形成鸡爪。而$1,2,3$为中心点的鸡爪，为了让与1连的点尽可能多，1需要找$n/3+1,n/3+2,n/3+3$的点相连，2需要找$1,n/3+1,n/3+2$相连，3需要找$1,2,n/3+1$相连。

最终得到$1,\ldots,n$的度数序列，可以证明不存在比该度数序列更优的序列：起码有$n/3$个点度数≥ 3，只需要考虑$n/3+1,n/3+2,n/3+3$的点度数能否更小，以$n/3+1$为例，其目前和$1,2,3$连接，而这三个点都已经和$4,\ldots,n/3$都连接，所以无法换成更小的点。所以该序列已为最优。

最后，特殊判断$n<=6$的情况，因为3不形成中心点。

1002

简要题意

在一张 $n\times m$ 大小的地图里（左下角为 $(1,1)$，右上角为 (n,m) ），有 K 个怪物。

对于每个怪物有这样三个属性，价值 A_i ，攻击力 B_i ，攻击距离 C_i 。

主角的初始生命值为1，在进入地牢前可以贷款 x 个金币（ x 为任意正整数）来获得 x 点生命值。主角初始出现在地图的 (sx,sy) 位置，保证该位置上没有怪物。

每回合的开始，主角可以进行下面两个操作中的一个操作。

1.离开地牢，获得当前击杀怪物的金币并进行结算。

2.瞬移到同行/同列上与当前位置距离不超过 d 的没有怪物的地图内的位置，并且消灭瞬移起点和终点相连的线段上的所有怪物。（消灭怪物可以瞬间获得怪物的价值数量的金币）这里的距离可以用曼哈顿距离来理解。

在每回合结束时，会结算存活的怪物对主角造成的伤害。

对于每个怪物而言，如果主角和怪物之间的曼哈顿距离小于等于怪物的攻击距离 C_i ，那么主角就会受到 B_i 点伤害。在任意时刻主角的生命值小于等于0角色就会死亡，并且失去所有获得的金币。

试问主角最多能获得多少金币。

$(2\leq n,m\leq 40,\ 1\leq K\leq 10,\ 1\leq d\leq 8,\ 1\leq C_i\leq 8)$

题解

首先看到K的数据范围很小，我们可以联想到可以用状态压缩的方式来表示怪物的存活情况。

然后对于一开始的生命值贷款这个机制，你可以这样想，你可以直接忽略生命值小于等于**0**角色就会死亡这个限制，然后把怪物的攻击看成扣金币，这样算出一个最大金币收益的方案，其实就是最终的答案了。（因为你知道了这种情况下的方案之后，你就知道了会受到多少怪物的攻击，那你一开始贷款那么多买血量就可以了）

然后可以设立状态$dp[i][j][k]$表示当主角在点(i,j)，怪物的存活状态用二进制表示为k时，主角最多能获得多少金币，然后我们枚举在当前状态下可以进行的操作之后就可以得到他的后继状态。

接着我们发现，如果我们把(i,j,k)看成是一个抽象点，把这些操作都看成是单项边，边权就是做这个操作所能获得的金币（这里有可能是负数，就是失去金币）。然后我们就得到了一张有权的有向图。那么我们只要做一遍单源的最长路就可以找到我们最终的答案。

因为只有消灭怪物的时候金币才会变多（也就是正权边），所以这张图不会有正环，故我们直接使用传统的最长路算法即可。点数为$n \cdot m \cdot 2^K$，边数为$4n \cdot m \cdot 2^K \cdot d$，复杂度为$O((n \cdot m \cdot 2^K \cdot d)\log(n \cdot m \cdot 2^K))$

1003

题解

保证了魔方被扭动的次数最多为3，所以可以直接从一个完好的魔方开始暴搜，看是否能找到一个状态与所给魔方相同或只差一个角。

同时容易发现我们只在意魔方角上的块，所以可以转换为2阶魔方，降低代码难度。

正解则不需要考虑这个性质，假设这个魔方是一个二阶魔方。

正常使用魔方能得到的状态与交换一次贴纸能得到的状态一定是不重叠的，但同时还存在将魔方拆散再任意重组的状态，这些状态包含正常使用魔方的所有状态，且仍与交换一次贴纸能得到的状态不重叠。

对这类状态的处理十分简单，可以使用打表或预处理的方法处理出使用这种方法能得到的所有24种角的形态，并与当前8个角的形态进行对比，至多只有1个角不在这24种形态中，找到的这个角即为答案。

1004

简要题意

$x_1^{t_1}x_2^{t_2}...x_k^{t_k} \le n$，$gcd(x_1,x_2,...,x_k)=1$

$1 \le n \le 10^{10}$，$1 \le k \le 10^5$，$1 \le t_i \le 33$

求满足以上条件的有序数对 $(x_1,x_2,...,x_k)$ 数量

题解

令$f(n)$表示$x_1^{t_1}x_2^{t_2}...x_k^{t_k}=n$ 且 $gcd(x_1,x_2,...,x_k)=1$ 数对数量，$F(n)=\sum_{i=1}^n f(i)$，则答案为$F(n)$

$gcd(x_1,x_2,...,x_k)=1$等价于每个质因子在$x_1,x_2,...,x_k$中次数的最小值都是0，易知各个质因子之间独立，f为积性函数，问题在于$f(p^c)$怎么求

$c=1$时，$f(p)=num_1$，其中num_1是$t_1,...,t_k$中1的数量

$c>1$时，$f(p^c)$等价于将c划分给k个数，第i个数必须是t_i的倍数，且k个数里至少有一个为0的方案数

考虑枚举某个集合的元素均为0，剩余部分必须不为零，形如2w+2x+3y+4z=c（w,x,y,z>=1）

如果已知2x+3y+4z在每个c处的方案数，那么可以$O(maxc)$的时间内推出2w+2x+3y+4z=c在每个c处的方案数，就是一个稍加修改的完全背包

发现对于值相同的t_i，性质是一样的，可以直接用t_i这个值有几个来代替，而不需要2^k枚举，由于w,x,y,z>=1，且$c\le 33$，故搜索的次数是有限的，经计算不超过24w

此时发现$f(p)$为关于p的零次多项式，$f(p^c)$可快速求，可用min25筛快速求得

总复杂度$O(\frac{n^{\frac{3}{4}}}{logn}+24w\cdot maxc)$

1005

简要题意

你需要构造一个长度为n的数列A，满足每个A_i是0到K的一个整数。

然后一共有若干个规则来评估你这个数列A的分数，规则可以由一个整数七元组(op,i,j,a,b,d,v)构成。

当 op 为 0 时，代表如果 $a\times A_i+b\times A_j\le d$那么你就可以获得 v 的得分。

当 op 为 1 时，代表如果 $a\times A_i+b\times A_j\ge d$ 时你就可以获得 v 的评分。

如果A_i和A_j出现在一组规则中，我们定义属性i和属性j有关系，题目数据满足属性之间的关系是一个树的结构，并且树的直径小于等于2。并且还满足任意一对属性对(i,j)之间不会有超过20个规则。

询问你能最多能获得多高的评分。只用输出得分不用输出具体方案。

$(2\le n\le 200,1\le K\le 10^6)$

注意$-1\le a,b\le 1$

题解

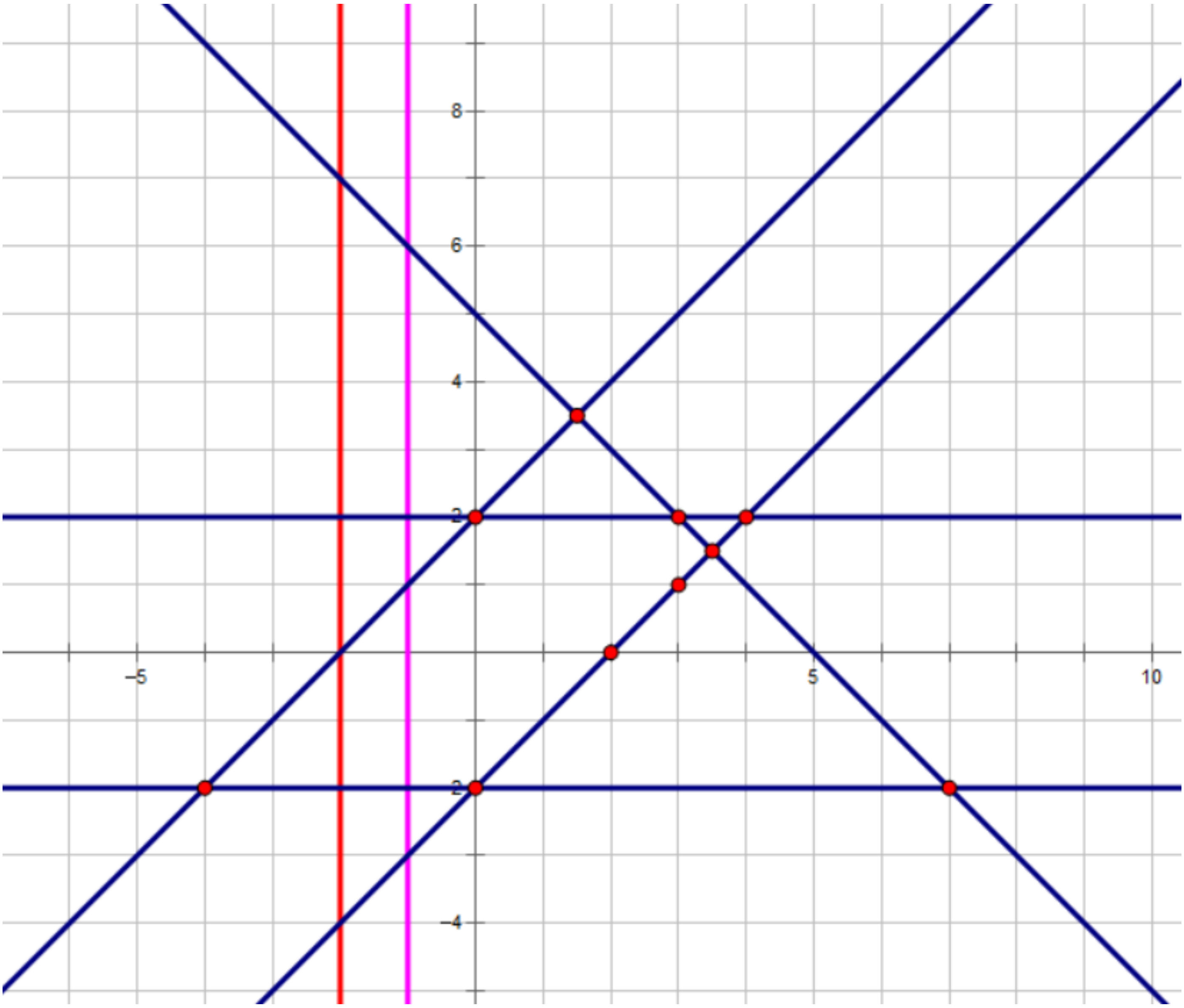
首先对于$v<0$的情况，我们可以通过先给自己的分数加上v，然后把规则取反一下。

如果 op 为 0，$a\times A_i+b\times A_j\le d$，翻转之后就是$a\times A_i+b\times A_j\ge d+1$，如果符合这个规则就获得$-v$的评分。op 为 1 的时候类似。

我们先考虑$n=2$的时候的情况，会发现这就是一个线性规划。如果这个题的A数组可以允许实数，那么我们有一个很有用的结论——我们把每个规则看成一条直线，那么我们只用枚举这些直线的交点，然后对于每个交点求出其对应的分数就可以找到最终的答案。

对于本题有A数组的数都是整数的限制，我们不能直接使用以上结论，但是也能从以上结论收获一些思路。

我们设函数$F(x)$表示当A_1为x时，我们能通过构造A_2获得最高多少的评分。然后我们会发现$F(x)$是一个分段函数，这个分段点其实就是上面我们提到的所有规则对应的直线的交点对应的x。（如果这个交点的x不是整数，那么我们取$\lceil x\rceil$和$\lfloor x\rfloor$作为分段点即可）因为我们限制了满足任意一对属性对之间的规则不超过20条，所以这个交点不会超过800个（$20\times 20\times 2$）。那么我们只要求出这些交点对应的F函数的值，就可以表示出这个分段函数了，也就是能求出答案了。这个过程有点像扫描线，拿一条$x=t$的竖线从左向右扫。



至于为什么是分段函数，你可以观察上图（上图中蓝色的先都是规则对应的直线，红色点是直线之间的交点），在两个交点之间有两条竖线（就是颜色不一样的那两条）。然后你会发现红色的那条对应的$F(x)$的值其实是和紫色那条一样的，但是如果紫色的线再往右移一点移过他右边的交点，他的$F(x)$的值就和红色的不一样了。

然后再看我们如何对于每个交点的x求出这个$F(x)$，因为x已经确定，也就是A_1已经确定，那么规则就会变成如果$A_2 \leq w$那么就可以获得v的得分这样的形式，就变成了一个区间加减的问题，这个时候我们可以用前缀和后缀和、差分相关的思想或者直接用离散化+线段树、树状数组之类的数据结构维护一下就可以求出这个$F(x)$了。

这一部分比较难理解的话可以动手画一下图，会很好地帮助到理解。

然后对于n比较大的情况，首先由于这个树的直径不超过2（也就是说是一颗菊花树），那么我们可以先找到度数最大的点作为树的根，假设为A_1，然后**依次**对于A_2，A_3，$\cdots$，A_n求出他们的$F(x)$，最后把这些分段函数用二路归并的方式归并到一起就得到了$G(x)$，表示当A_1为x时，我们可以通过构造剩下的A数组的值来获得的最大的评分。因为A_1确定的时候，我们可以发现剩下的A_i之间其实是独立的，所以才可以用二路归并的方式合并。最后有了$G(x)$我们通过求分段函数的最值的方法就可以求出最后的答案了。

复杂度$O(n * cnt^3)$

1006

简要题意

有一个n个结点的有根树，你需要从1号结点走到任意一个叶子节点，每天你有$\frac{p_i}{15}$的可能性可以向任意一个**儿子节点**走一步（i为你当前所在的节点编号），询问你期望最多可以在这颗树上走多久。

$n \leq 10^5$

题解

简单分析一下一下题意就是每个点有一个期望的停留时间（可以用分数类来维护），然后我们只用找到一条期望时间之和最长的一端为根节点的链即可。

1007

题解

可以直接通过url parse的规则，提取出字符串中的每一个部分：scheme/netloc/path。对于path中的每一个部分只需要判断是否有=，然后按=位置划分成两部分A和B。按照题意输出即可。

1008

题解

如果树的直径经过由x成长后的点，则必然可以经过所有由 x 成长为的点。

所以将x权值设为成长出的点数，直接对修改权值后的树求直径即为答案。

考虑如何快速算出成长后的点数，可以发现，在进行一次成长后的树度数最大为3，对于一个2度点，成长一次变为2个2度点，对于一个3度点，成长一次边为2个2度点和1个3度点，对于一个d度点，成长一次为2个2度点和$d-2$个3度点。

对以上规律进行推导，可以得到一个度数为d的节点，成长m次后，总点数为$(d-1)*(2^m-1)+1$。

对于求改变后的树的直径，在m较小时，可以直接将改变后的点权计算出来；在m较大时，可以把总度数减去总点数作为第一关键字，总点数作为第二关键字，最后再通过总的度数和点数得到直径的长度。

1009

题解

增量与初始速度无关，直接按初始速度为零进行计算，速度与时间构成的图上的面积为路径的增量。1秒内的面积分为加速，匀速，减速三种情况，可以进一步分为整块的和半块的进行统计。

对于每一次目标速度的区间加，可以拆成一次后缀加和一次后缀减。可以发现，当前速度的变化只与目标速度的差有关，当速度和目标速度同时进行修改时，后续的速度折线形状不会发生变化，只会向上或向上移动。

所以对于一次后缀修改，当前速度变化的位置只会存在一个，变化位置前的速度与目标速度的差进行区间的加减，速度折线的移动可以直接统计进整块的答案中。

考虑如何找到速度变化的位置，对于一个增加的操作，速度变化的位置在目标速度增加后，也就是s之后的，第一个匀速或减速的位置，这个减速的位置要满足减速前的速度刚好与目标速度差1。对于一个减少的操作，变化的位置也是类似的。

可以将以上的对于速度与目标速度的差的部分抽象为维护一个数据结构，满足支持区间的加减1，查询区间内第一个出现的-1,0,1的位置。

一个较为简单的实现方法是通过分块维护，直接在块内进行排序，通过平衡块的大小，可以做到$O(m\sqrt{nlogn})$的时间复杂度，在常数较为优秀时可以通过此题。

也可以在块内维护一个值域范围为$[-\sqrt{m},\sqrt{m}]$桶，对于答案可能超出桶的范围的情况直接重构块，做到均摊$O(m\sqrt{n}+n\sqrt{m}))$的时间复杂度。

还有一种时间复杂度更为优秀的做法是分别维护查询-1和1的线段树，在线段树上维护最值和最值的位置，可以做到$O(mlogn)$的时间复杂度。

1010

题解

直接按题意模拟即可，也可以稍微找一下规律。

可以发现除了女神的睿智外，合成出的颜色取决于左侧，所以2个大结晶的颜色等于第1,5个碎片的颜色，女神的睿智的颜色取决于第1,5个碎片的颜色和与其同色的碎片的数量。

1011

题解

首先可以通过 KMP/Hash 等字符串匹配判断B'中是否包含 C，筛选出合法的 B。

之后就是在字符串 A 中找到所有出现过的 B。这个过程可以通过构建B的AC自动机，在这个自动机上跑 A 串的匹配。

在A跑匹配过程中经过的所有点，沿着AC自动机的fail边往上跳所经过的节点就表示了所有在A中出现过的字符串。

我们只需要建出fail树，统计子树中是否存在A串匹配时经过的点即可。最后判断B串在AC自动机上对应的节点，即可知道B串是否合法。

1012

题解

考虑判断两个点是否在任意两张图上都互相联通，可以转换成看两个点在每一张图中并查集所在集合的根是否相同。

如何快速的判断两个集合相等，我们可以对这个集合进行哈希。给每一张图的每一个点随机一个long long范围内的数字作为哈希值，然后将所有点的哈希值求和（或者做异或之类的操作），作为集合的哈希值。

判断两个集合是否相等，只需要看两个集合的哈希值是否相等即可。

于是我们只需要维护每一张图的并查集，同时对于每一个点，维护其所在集合的哈希值。由于不同的合并对于不同的点对之间的影响是不同的，所以在合并并查集的时候我们还需要启发式合并，以动态修改待合并点的哈希值。

最后我们只需要通过哈希表/map来统计总共有多少个点对的哈希值相同，这个答案即为所求在任意两张图上都互相联通的点对数量。