A 来点离线做法

考虑根号分治。对于k小于根号n，可以预处理$sum[i][k]=\sum\limits_{j=1}^{i}\frac{a[i]}{k}$，询问时输出$sum[r][k]-sum[l-1][k]$即可。预处理的总复杂度$O(n\sqrt{n})$，询问复杂度$O(1)$。

对于k大于根号n，离线询问，按k从小到大分别处理，用分块维护区间的$\frac{a_i}{k}$的和。当k增大的时候，对于$\frac{a_i}{k}$变小的那些位置进行修改。对于一个a_i，最多有$\sqrt{a_i}$次参与修改，每次修改是$O(1)$的，单点修改并且修改所在块的和即可，因此修改总复杂度是$O(n\sqrt{n})$。询问时暴力枚举小块和中间的整块，询问复杂度也是$O(n\sqrt{n})$。

B 停车难题

是一个求mex的签到题。

由于给定的a_i严格单调递增，所以可以用i从1枚举到n，一旦$a_i\neq i-1$则输出$i-1$。如果$a_i=i-1$全都满足则输出n。具体实现方式不同，时间复杂度可以做到$O(n)$或$O(nlogn)$。

C 数学的大厦崩塌了

是一个比较难写的模拟题。

可以暴力枚举$1\sim n$范围的B，再按位枚举B的子集得到b， 暴力枚举$1\sim b-1$范围里的a，复杂度只有约$2*10^7$。满足$b|Ba$的a进入判定环节，判定环节内部需要判断a,b是否有共同数字（若有共同数字说明还能继续删除，不合法）、判断数字出现次数是否正确（在A出现的次数-在B出现的次数=在a中出现的次数-在b中出现的次数）、a是不是A的子序列（b是B枚举子集得到的，一定满足子序列）。

D 贪玩迷宫

建图后是一个内向基环树森林，求有条件地改变一个边后，能到达终点的所有点的距离的最大值。

设$f[x][y]$表示别的点到(x,y)的最远距离，可以先断开基环，逆向建图做树dp可以求得非基环上的点的$f[x][y]$，基环上的点的$f[x][y]$可能更长，需要断环倍长成链，用环上点的f互相更新。

(x,y)能到达终点时，用$g[x][y]$表示(x,y)到达终点的距离。这个也是反向建图做树dp即可。

设(x,y)相邻的点(u,v)答案是$max(f[x][y]+g[u][v]+1)$，如果(x,y)和(u,v)是祖先关系则不允许更新答案，std在具体实现时使用了dfs序判定。

E 赚的越多，赚的越少

如果暴力模拟，复杂度是$O(n)$的。

分析一下余额的最大值可以发现余额不会超过$2*10^6$，所以每到一个新的余额就记录一下到达时间，如果发现这个余额之前到达过就可以证明进入了循环中，可以利用循环来优化复杂度，优化后复杂度可以做到$O(min(n,x*k))$。

F 阶乘的和

当$n\geq 20$时，$n!$的最后四位会全部变成0，因此以后的$n!$不会影响答案，对于$n\geq 20$直接输出313即可。

对于$n<20$，可以用暴力来做。

G 球

设两球心距离 $d=\sqrt{x^2+y^2+z^2}$，以及半径 r_1, r_2，可以分类讨论：

- **相离**：$d\geq r_1+r_2$，重叠体积为 0。

- **内含**：$d \le |r_1 - r_2|$，重叠体积为较小球的体积 $\frac{4}{3}\pi r_{\min}^3$。
- **相交**：$|r_1 - r_2| < d < r_1 + r_2$，重叠部分是两个球冠的体积之和。

球冠是指一个球被一个平面截取的一部分，其高度为 h，球的半径为 R，则球冠体积为：
$V(R,h) = \frac{\pi h^2}{3}(3R - h)$

设两球心距离 d，两球半径分别为 r_1, r_2。 两球相交的对称面（相交圆所在的平面）到球心1的距离为：
$a = \frac{r_1^2 - r_2^2 + d^2}{2d}$

则球心1到该平面的距离为 a，球心2到该平面的距离为 $d - a$。于是：

- 球1的球冠高度：$h_1 = r_1 - a$
- 球2的球冠高度：$h_2 = r_2 - (d - a)$

这两个球冠合起来就是重叠部分，所以总体积为：

$V(r_1, h_1) + V(r_2, h_2)$

复杂度为：$O(T)$

H 关注火花花喵

由于资源、球数、连续黑球次数范围都很小，可以考虑使用**记忆化搜索**（动态规划）来枚举所有可能的状态。

设$f(n, m, k, r, b, c)$ 表示：

- 当前剩余爆点数n，剩余战技点数m，战技点上限为k（上限在过程中固定不变）；
- 奖池中剩余红球数r（初始3），剩余黑球数 b（初始17）；
- 当前已经连续抽到黑球的次数c（$0 \le c \le 6$，因为一旦达到6下一次必红）。

该状态返回从当前状态开始，继续抽奖直到无法再抽（无资源或无球）所能获得的**期望笑点**。

- 若$r = 0$且$b = 0$，无球可抽，期望为0。
- 若$n = 0$且$m = 0$，无资源可用，期望为0。

每次抽奖优先使用爆点，因此分为两种情况：

1. **有爆点**（$n > 0$）：
   - 若$c = 6$（已连续6黑），则下一次必红。消耗1爆点，获得2笑点，战技点增加2（不超过上限），红球数减1，连续黑球计数清零，转移：
     $f = 2 + f(n-1,\ \min(m+2, k),\ k,\ r-1,\ b,\ 0)$
   - 否则，正常随机抽取。红球概率$p_r = \frac{r}{r+b}$，黑球概率$p_b = \frac{b}{r+b}$。
     - 抽到红：消耗1爆点，得2笑点，战技点增加2，r减1，c清零。
     - 抽到黑：消耗1爆点，得1笑点，战技点不变，b减1，c加1。
       转移：
     $f = p_r \cdot \big(2 + f(n-1,\ \min(m+2, k),\ k,\ r-1,\ b,\ 0)\big) + p_b \cdot \big(1 + f(n-1,\ m,\ k,\ r,\ b-1,\ c+1)\big)$
2. **无爆点但有战技点**（$n = 0,\ m > 0$）：
   - 若$c = 6$，必红。消耗1战技点，得2笑点，战技点增加2（注意消耗后原战技点为m，消耗后变为$m-1$，再增加2得$m+1$，需与上限取最小值），转移：
   - $f = 2 + f(0,\ \min(m+1, k),\ k,\ r-1,\ b,\ 0)$
   - 否则，正常随机抽取。
     - 抽到红：消耗1战技点，得2笑点，战技点变为$\min(m+1, k)$，r减1，c清零。
     - 抽到黑：消耗 1战技点，得1笑点，战技点变为$m-1$，b 减 1，c加1转移：
     $f = p_r \cdot \big(2 + f(0,\ \min(m+1, k),\ k,\ r-1,\ b,\ 0)\big) + p_b \cdot \big(1 + f(0,\ m-1,\ k,\ r,\ b-1,\ c+1)\big)$

实现的时候使用记忆化数组$f[16][16][16][4][18][8]$储存，所求答案为$f(n, m, k, 3, 17, 4)$。

时间复杂度约为：$O(500nmk)$

I 收益

每个人从起点向下走，相当于选择一条从起点出发到叶子的路径，路径上可能存在一些点没有物品。

考虑从叶子向上处理：每个节点u维护一个大根堆，里面存放经过u并且还能继续向上走的若干条可能被选择的路径。 先递归处理u的所有子节点，将子节点的大根堆合并到u（启发式合并）。如果u的大根堆非空，取出最大值，将其加上u物品的价值放回大根堆里。若u有x人，则从u所在的大根堆取出前x大的路径累加到答案里（若没有x个路径则取完即可）。

总复杂度：$O(n\log^2 n)$。

J 神经网络图染色

设$f[i][0]$表示第i层颜色全一样的方案数，$f[i][1]$表示第i层恰有两个颜色相同，$f[i][2]$表示第i层三个节点颜色都不一样。

对于$i=1$，有

$f[1][0]=m*(m-1)$

$f[1][1]=m*(m-1)*(m-2)*3$

$f[1][2]=m*(m-1)*(m-2)*(m-3)$

对于$1<i\le n$，有

$f[i][0]+=f[i-1][0]*(m-1)$

$f[i][0]+=f[i-1][1]*(m-2)$

$f[i][0]+=f[i-1][2]*(m-3)$

$f[i][1]+=f[i-1][0]*(m-1)*(m-2)*3$

$f[i][1]+=f[i-1][1]*[m-2+(m-2)*(m-3)*3]$

$f[i][1]+=f[i-1][2]*(m-3)*(m-3)*3$

$f[i][2]+=f[i-1][0]*(m-1)*(m-2)*(m-3)$

$f[i][2]+=f[i-1][1]*[(m-2)*(m-3)+(m-2)*(m-3)*(m-4)]$

$f[i][2]+=f[i-1][2]*[(m-3)*(m-4)*(m-5)+(m-3)*(m-4)*3+(m-3)*3+1]$

这是线性齐次递推，使用矩阵快速幂优化是一个经典的技巧。由于本题主要考察组合计数和动态规划，就不再考察矩阵快速幂了，直接$O(n)$递推即可。

最后的答案是$f[n][0]*(m-1)+f[n][1]*(m-2)+f[n][2]*(m-3)$。

K 取石子

设b_i表示把a_i质因数分解后，2的幂次是多少。问题转变成每次操作可选若干个非0的b_i，将每个被选中的b_i拆分成两个b_i-1，b_i全为0时无法操作，问先手胜负。

用数学归纳法可以证明，当b_i**存在**奇数时为先手胜状态，当b_i**全为**偶数时为先手输状态。时间复杂度为$\sum log_2(a_i)$。

L 最小生成图

题外话：本题有更优秀的FFT和模拟退火等做法，各位选手可以在$O(n^2)$做法的基础上继续思考。

如果已经决定选择若干条边，肯定是从小到大依次选择这些边连接能得到最小的总代价。要求最终方案图联通，即选择的方案的边集一定能包含某个最小生成树的边集。

提前把所有边排序，求出来一个最小生成树（任意一个即可不会影响答案），其中一个最优方案所选择的边集一定是包含这个最小生成树的边集和x条（x可以为0）非最小生成数边。考虑枚举x的值，对于每个x一定是选择前x小的非最小生成数边最优，把前x小的非最小生成数边和最小生成树边放一起从小到大统计总代价更新答案即可。

时间复杂度：$O(n^2)$

M 蜗牛养殖

由于$k=4$，可以暴力找到蜗牛所在点的lca，把影响可达性的关键点找出来建立虚树，这样最多涉及到七个点六条边，接下来3^6暴力枚举每条边的方向（x指向y/y指向x/无法互相到达），判定是否能从一个蜗牛走到另一个蜗牛。如果不能走到说明是一个合法方案，累加方案数，虚树内部的每条边对应着树上的若干条边，进行方案数累乘，设虚树外部有cnt条边，这些边的方向不影响可达性，所以乘以2^{cnt}即可。

容斥做法不太好做，出题人没想明白，所以这里不进行讨论。

另一种做法是考虑将蜗牛视为关键点，进行动态规划。

设$f[x][0/1][0/1]$表示x为根的子树的方案数。其中第一个$0/1$表示是否存在子树里的一个蜗牛向上移动到x的路径，第二个$0/1$表示是否存在x向下移动到子树里的一个蜗牛的路径。
$f[x][0][0]+f[x][0][1]+f[x][1][0]+f[x][1][1]$即为以$x$为根的子树内部给边定向后的**合法**方案数。

如果x是关键点

此时只关心$f[x][1][1]$即可，不允许$f[x][0][0]/f[x][1][0]/f[x][0][1]$有值

先令$f[x][1][1]=1$，接下来开始转移：

```
    f[x][1][1]=f[x][1][1]*(f[y][0][0]*2+f[y][0][1]+f[y][1][0])
```

如果y的状态是00，则(x,y)这条边可以任选方向。

如果y的状态是01，则(x,y)这条边必须定向为$x\leftarrow y$。

如果y的状态是10，则(x,y)这条边必须定向为$x\rightarrow y$。

（如果y也是关键点，则方案数会变成0。但是不用特判，会在代码中自然地使得方案数为0）

如果x不是关键点

此时必须要求$f[x][1][1]=0$，因为x不是关键点，且状态为11说明存在
蜗牛$\rightarrow\dots y1\rightarrow x\rightarrow y2\rightarrow\dots\rightarrow$蜗牛的路径存在。

由于要对$f[x][0][0],f[x][0][1],f[x][1][0]$，推荐建立一个辅助数组$g[0/1][0/1]$

```
g[0][0] = f[x][0][0] * (f[y][0][0]*2 + f[y][0][1] + f[y][1][0]);
g[0][1] = f[x][0][1] * (f[y][0][0]*2 + f[y][0][1]*2 + f[y][1][0] + f[y][1][1])
        + f[x][0][0] * (f[y][0][1] + f[y][1][1]);
g[1][0] = f[x][1][0] * (f[y][0][0]*2 + f[y][1][0]*2 + f[y][0][1] + f[y][1][1])
        + f[x][0][0] * (f[y][1][0] + f[y][1][1]);
```

第一个转移式表示要想x的状态为00，需要原来就是00状态。此时的定向方案和上面类似：

如果y的状态是00，则(x,y)这条边可以任选方向。

如果y的状态是01，则(x,y)这条边必须定向为$x\leftarrow y$。

如果y的状态是10，则(x,y)这条边必须定向为$x\rightarrow y$。

第二个转移式表示要想x的状态为01：

1、原来x就是01状态。此时y可以是00状态，随意定向；y也可以是01状态，随意定向；y也可以是10状态，必须$x\rightarrow y$；y也可以是11状态，必须$x\rightarrow y$

2、原来x是00状态。此时需要y能够向下到达蜗牛也即需要y为$01/11$状态，且(x,y)这条边必须定向为$x\to y$

第三个转移式和第二个类似。

执行g的转移后再将g的值重新赋值给f即可完成子树y的转移：

```
    f[0][0]=g[0][0];
    f[0][1]=g[0][1];
    f[1][0]=g[1][0];
```

最后输出$f[1][0][0]+f[1][0][1]+f[1][1][0]+f[1][1][1]$即可。

可以发现这种做法不要求$k\le 4$，出题人令$k\le 4$是为了~~误导选手（bushi）~~给虚树做法留个路。