第十五届“商汤杯”北京航空航天大学程序设计竞赛预赛题解

北航 ACM 集训队

A 三角形切半

原三角形的面积是 ab，直线右侧的小三角形的面积是 $(x_0+a-c)^2\frac{b}{a}$，$x_0 \le c \le x_0+a$。

令右侧的面积为原三角形的一半，可以解得 $c = x_0 + a - \frac{a}{\sqrt{2}}$。

B 连接美国

用并查集或随便遍历一下获得每个连通块。记连通块的数量为 c。想要得到连通图就需要把连通块连接起来，那最少则是用 $c-1$ 条边把连通块缩点后的图连成一棵树，即最少加入 $c-1$ 条边即可。

简单实现就每个连通块取一个点，将它们连成一条链就行。

时间复杂度 $\mathcal{O}(n)$ 或 $\mathcal{O}(n\alpha(n))$。

C 数码管

容易发现如果确定了第 i 位显示的数字，由于整体转 180 度后还需要读出来的数相同，那么相应的第 $n-i+1$ 位的数字也被确定了。

特别地，当 n 为奇数，中间会有一位数字和自己配对，想要转 180 度后还能读数相同，则只能取 0, 2, 5, 8 这 4 个数。

综上，当 n 为偶数时，有 $\frac{n}{2}$ 个位置能有 6 种选择，答案为 $6^{\frac{n}{2}}$；当 n 为奇数时，有 $\frac{n-1}{2}$ 个位置能有 6 种选择，额外有一个位置有 4 种选择，答案为 $4\times 6^{\frac{n-1}{2}}$。

可惜 n 的长度很大，对 n 除以 2 的复杂度也是 $\mathcal{O}\left(\log n\right)$ 的，快速幂的复杂度就成了 $\mathcal{O}\left(\log^2 n\right)$ 不太行。

为了计算上述模意义下的值，可以使用费马小定理（欧拉降幂）来缩减指数部分。不过模 $\varphi(998244353) = 998244352$ 意义下，虽然可以判断 n 的

奇偶性，但是没有 2 的逆元。因此我们可以将降幂用的模乘 2，在需要除以 2 时，模数和余数同时除以 2 即可。当然也可以实现大整数减 1 及大整数除 2，不过稍麻烦一些。

（最初模用的 10^9+7，但是 6 不是原根，循环节大小最多是 φ 的一半，因此不用上面对指数模意义下除以 2 的处理也能过。）

时间复杂度 $\mathcal{O}(\log n)$。

D 颤弦蝶螈与 PCPC

贪心。容易发现两个性质：

1. debug 时间少的题目严格更优，因此可将所有题目先按照 debug 时间排序。
2. debug 时间大于等于 x 的题目，选择读错题严格更优。

题目要求在 t 分钟内解出尽可能多的题，因此每题应当花费最少的时间（暂不考虑罚时），即 $\min(2x, x+a_i)$，可以在 $\mathcal{O}(n)$ 的时间内求出最多解出的题数，记为 m。

在解出题数为 m 的前提下，考虑最小的罚时。显然颤弦蝶螈做前 m 道题是最优的。不妨枚举读错的题目数量 c_1 和写错的题目数量 c_2，其中 $c_1+c_2=m$。显然写错的题目应该选择那些 debug 时间少的题目，表现在排序后的题目上就是：前 c_2 道题写错，后 c_1 道题读错。注意这里需要判断做题时间是否超过了 t，如超过，则这种情况不合法。所需的做题时间是 $(m+c_1)x+\sum_{i=1}^{c_2} a_i$。此外，若枚举的 c_2 道题中有 debug 时间大于等于 x 的题目，也违反了性质 2，不合法。因而实际做题的顺序恰好与前面排好的序相同（显然先做耗时少的题目更优）。罚时是 $c_2k+\left(\sum_{i=1}^{c_2}\sum_{j=1}^{i} x+a_j\right)+\left(\sum_{i=1}^{c_1}\left(\sum_{j=1}^{c_2} x+a_j\right)+2ix\right)$。这两个式子均可以用前缀和及等差数列求和等技巧 $\mathcal{O}(1)$ 计算。此外，枚举量是 $\mathcal{O}(n)$，因而这部分的复杂度是 $\mathcal{O}(n)$。

总时间复杂度 $\mathcal{O}(n\log n)$。

E 游戏分组

考虑这 n 个人，$\binom{n}{ka_m}$ 选 ka_m 个人来玩第 m 个游戏，那么如果知道了剩下的人玩其他游戏的方案数，再乘上 ka_m 个人分成 k 个人数相等的组的方案数，即 $\binom{ka_m}{a_m,\cdots,a_m}\cdot\frac{1}{k!}$，就有了 k 个人玩最后一个游戏的方案数。

那么有 DP：

$$f_{m,n} = \sum_{k=0}^{\lfloor n/a_m \rfloor} \frac{n!}{(n-ka_m)!k!(a_m!)^k} f_{m-1,n-ka_m}$$

直观上就有一个 $\mathcal{O}\left(n^2 m\right)$ 的 DP 做法了，已经能通过了。

由于 a_i 各不相同，这个做法实际上的复杂度是 $\mathcal{O}\left(\sum_{j=0}^{n}\sum_{i=1}^{m} j/a_i\right) = \mathcal{O}\left(\sum_{j=0}^{n} j\sum_{i=1}^{n} 1/i\right) = \mathcal{O}\left(n^2 \log n\right)$。

E.1 进一步思考

实际上把上述 DP 展开 m 次，去掉递推后，容易发现需要求的就是：

$$\sum_{k_1a_1+k_2a_2+\cdots+k_ma_m=n} \frac{n!}{k_1!(a_1!)^{k_1}\cdots k_m!(a_m!)^{k_m}}$$

也就是多项式 $f_u(x) = \sum_{i\ge 0}\frac{x^{ia_u}}{i!(a_u!)^i}$ 的乘积的 n 次项系数的 $n!$ 倍。

$$f_u(x) = \sum_{i\ge 0}\frac{1}{i!}\left(\frac{x^{a_u}}{a_u!}\right)^i = \exp\left(x^{a_u}/\left(a_u!\right)\right)$$

$$\prod_u f_u(x) = \exp\left(\sum_u \frac{1}{a_u!}x^{a_u}\right)$$

用多项式 Exp，n 次项系数乘 $n!$ 就是答案。

时间复杂度 $\mathcal{O}(n\log n)$。

F 推箱子

地图上只有 $n\times m\le 225$ 个位置，其中人和箱子一共有三个不断变化的位置，记 $f_{i,j,k}$ 表示从初始状态出发，达到人在 i 号位置、箱子 A 在 j 号位置、箱子 B 在 k 号位置这一状态的最少移动步数。使用 bfs 算法可求得每个状态的最优解（如果别的题中相邻格子的距离不为 1，可以采用 dijkstra 等最短路算法解决）。根据 bfs 算法的性质，只要走到一个合法目标状态（箱子在传送点），它的 f 就是最优解。当然也可以跑完整个 bfs 后，枚举合法目标状态取最小值。

时空间复杂度 $\mathcal{O}((nm)^3)$。

G easy segment problem

n 个线段各选一个点，坐标求和，能用这个过程得到的点的集合实际就是这 n 个线段的闵可夫斯基和，最终是一个凸包。

一种思路是分治 + 闵可夫斯基和，直接求出来这个凸包，然后旋转卡壳来或枚举 + 二分求凸包上的最远点对。时间复杂度 $\mathcal{O}(n\log n)$，但是很难实现。

由于需要求的是凸包上的最远点对，凸包相对原点的位置没有用。也可以想到，每个线段实际是 $\overrightarrow{a_ib_i}, \overrightarrow{b_ia_i}$ 包成的一个面积为 0 的凸包，每次加入一个线段，相当于这个凸包照着这个线段的方向、移动线段长度的距离，所有经过的点组成的集合就是加入这个线段后的闵可夫斯基和。

所以这个凸包也就是所有的 $\overrightarrow{a_ib_i}, \overrightarrow{b_ia_i}$，按极角序排序，然后任选一个开始求向量的前缀和，前缀和最后的一个元素必然是零向量。

前缀和表示的点组成了一个凸包，这个凸包刚好是题目所给 n 个线段的闵可夫斯基和（丢掉和原点关系的信息）。

时间复杂度 $\mathcal{O}(n\log n)$。

H 还原神作

不妨将 p_i 排序。

H.1 最大值

对于最大值，可以证明最优解的任意两对设计中，一定有一对包含另一对。若两对设计 (l_1,r_1) 和 (l_2,r_2) 相离，即 $l_1\le r_1\le l_2\le r_2$，那么 $r_1-l_1+r_2-l_2\le r_2-l_1+l_2-r_1$，那么将它们替换成 (l_1,r_2) 和 (r_1,l_2) 更优。若 (l_1,r_1) 和 (l_2,r_2) 相交，即 $l_1\le l_2\le r_1\le r_2$，那么 $r_1-l_1+r_2-l_2=r_2-l_1+r_1-l_2$，即替换成 (l_1,r_2) 和 (l_2,r_1) 也不会变差。因此存在一种最优解，它的所有设计对形成了一个区间套。显然应当选取 $(p_1,p_n),(p_2,p_{n-1}),\cdots,(p_k,p_{n+1-k})$ 最优。

H.2 最小值

对于最小值，类似地可以证明最优解中的任意两对设计相离，那么显然只会选择相邻的设计对。问题转化为在 $n-1$ 条线段中，选择 k 条不相邻的线段使得权值和最小。下面介绍两种解法：

解法一 贪心。记第 i 条线段的长度为 d_i，将所有 d_i 插入一个小根堆，每次选取堆顶线段 d_t，将 d_t 加到答案中，并将 d_{l_t}, d_{r_t} 删除，将 d_t 修改为 $d_{l_t} + d_{r_t} - d_t$，代表的含义是，放弃选择 d_t，选取左右两边的线段。这样操作 k 次即可。时间复杂度 $\mathcal{O}(n \log n)$。

解法二 wqs 二分（带权二分）。不妨先考虑一种 dp 解法，设 $dp_{i,j,0/1}$ 表示前 i 条线段中已经选了 j 条线段，第 i 条线段没取/取了的最小代价。那么答案为 $f(k) = \min(dp_{n-1,k,0}, dp_{n-1,k,1})$，转移时只需枚举第 $i+1$ 条线段取不取即可，时间复杂度为 $\mathcal{O}(nk)$。为了优化这一 dp，可以证明 $f(k)$ 是关于 k 的下凸函数，图像为一个右下凸壳。考虑函数 $g(k;a) = f(k) - ak$，显然它仍然是一个凸函数，且它的最小值点随斜率 a 的增加而向右移动。二分斜率 a，即可找到最小值恰在题目给出的 k_0 的 $g(k_0;a_0)$，此时有 $f(k_0) = g(k_0;a_0) + a_0k_0$。$g(k;a)$ 最小值的求解与上面的 dp 类似，只不过第二维线段数量 j 变成了 dp 要求的优化目标（二分时需要使用，在代价最小的前提下，段数最少），且转移的时候每条线段的代价需要减去斜率 a。这个 dp 的复杂度为 $\mathcal{O}(n)$，因而总复杂度为 $\mathcal{O}(n \log A)$，其中 A 表示坐标的值域。

请学习 wqs 二分以更好地理解，如这篇博客。

I 随机游走

I.1 采用 min-max 容斥的做法

为了方便说明，我们视 n, m 同阶。

记 $\min(S)$ 为点集 S 中最早到达的点到达的期望时间，$\max(S)$ 为点集 S 最后到达的点到达的期望时间，则由 min-max 容斥可知：

$$\max(S) = \sum_{T \subseteq S} (-1)^{|T|+1} \min(T)$$

由于 $K_{n,m}$ 是个完全二分图，所有左侧的点与所有右侧的点各自等价，因此可以设 T 中有 a 个点来自二分图左侧，b 个点来自二分图右侧。枚举初始点在左侧还是右侧后，可以直接按照数学期望定义计算 $\min(T)$ 的值。具体来说，假设时间 t 第一次到达 T 中的某个点，则概率是 t 时刻前随机游走没有到达过 T 中的点，而 t 时刻恰好到达的概率。

对所有 $t = 0, 1, 2, \ldots$ 分别计算 t 时间第一次到达 T 中点的概率与 t 的乘积并求和，结果就是所求的数学期望，且是个无穷级数形式。将其转化为

封闭形式后，可以在 $\mathcal{O}(\log M)$（计算逆元的复杂度；或 $\mathcal{O}(1)$，如果能很好地预处理所有用到的逆元）时间内计算结果，因此也可以求出 $\min(T)$。

枚举 $\mathcal{O}(n^2)$ 个可能的 (a,b) 二元组并代入 min-max 容斥式计算，即可在 $\mathcal{O}(n^2 \log M)$（或 $\mathcal{O}(n^2)$）内解决问题。

I.2 采用 DP 的做法

为了方便说明，我们视 n,m 同阶。

令 $f(x,S)$ 为当前在 x，已经走过了点集 S，接下来遍历全图需要的期望。假设现在在点 u，考虑沿着一条相邻边走到点 v，那么有

$$f(u,S) = \frac{\sum_v f(v, S \cup \{v\})}{\deg(u)} + 1$$

式子含义就是决策下一步怎么走。显然最后条件是对任意 x 有 $f(x,U) = 0$，其中 U 为全集，因此可以反过来递推，从 x 出发的期望步数就是 $f(x,\{x\})$。注意上面的转移可能还会回到当前的状态，因此需要把当前状态都整理到同一侧才能得到递推式。

由于 $K_{n,m}$ 是个完全二分图，所有左侧的点与所有右侧的点各自等价，因此 S 可以直接用二元组 (a,b) 代替，表示有 a 个点来自二分图左侧，b 个点来自二分图右侧，且我们不关注当前的点具体是哪个点，而是只关心它在哪一侧。

这样表示后，DP 方程就只有 $\mathcal{O}(n^2)$ 个状态，转移是 $\mathcal{O}(\log M)$ 的（或 $\mathcal{O}(1)$，如果能很好地预处理所有用到的逆元），因此总时间复杂度为 $\mathcal{O}(n^2 \log M)$（或 $\mathcal{O}(n^2)$）。

I.3 采用母函数的做法

记当前已经访问过左部 a 个点、右部 b 个点，并且访问时起点在 $u \in \{\mathrm{L},\mathrm{R}\}$（即起点在左部或右部），当前停在 $v \in \{\mathrm{L},\mathrm{R}\}$，并且此时是第一次访问 v 侧最后一个被访问到的点（即访问到 v 侧的一个新的点时就停下）。

有母函数

$$P_{a,b,u,v}(x) = \sum_{j=0}^{\infty} Pr\left\{\text{has passed through the edges } j \text{ times}\right\} x^j$$

题目所求即 $\sum_{u,v} P'_{n,m,u,v}(1)$。

初始有 $P_{1,0,L,L}(x) = \frac{n}{n+m}x^0$, $P_{0,1,R,R}(x) = \frac{m}{n+m}x^0$；其余的当 $a+b \le 1$ 时，$P_{a,b,u,v}(x) = 0$。

记有 $Q_{u,v,a,b}(x)$，能满足下列递推关系：

$$P_{a,b,u,L}(x) = P_{a-1,b,u,L}(x)Q_{L,L,a,b}(x) + P_{a-1,b,u,R}(x)Q_{L,R,a,b}(x)$$

$$P_{a,b,u,R}(x) = P_{a-1,b,u,L}(x)Q_{R,L,a,b}(x) + P_{a-1,b,u,R}(x)Q_{R,R,a,b}(x)$$

即 $Q_{u,v,a,b}(x)$ 表示，左部 a 个点、右部 b 个点，在 u 一侧少一个点的情况下，经过若干条边从最后一次访问到的点（在 v 侧），走到 u 一侧的某个新点，关于经过边的次数的母函数。

$$Q_{L,L,a,b}(x) = \frac{b}{m}\frac{n-a+1}{n}\sum_{j=0}^{\infty}\left(\frac{b}{m}\frac{a-1}{n}\right)^j x^{2j+2}$$

$$Q_{L,R,a,b}(x) = \frac{n-a+1}{n}\sum_{j=0}^{\infty}\left(\frac{a-1}{n}\frac{b}{m}\right)^j x^{2j+1}$$

$$Q_{R,L,a,b}(x) = \frac{m-b+1}{m}\sum_{j=0}^{\infty}\left(\frac{b-1}{m}\frac{a}{n}\right)^j x^{2j+1}$$

$$Q_{R,R,a,b}(x) = \frac{a}{n}\frac{m-b+1}{m}\sum_{j=0}^{\infty}\left(\frac{a}{n}\frac{b-1}{m}\right)^j x^{2j+2}$$

大体思路就是如果是一侧出发到同一侧停止，就是走了偶数条边；否则走了奇数条边。算一下这样走的概率后，和少一个点的情况乘一下就可以递推过来。

维护上面母函数及其导数在 $x=1$ 处的取值，就可以得到最终答案所求的值。

时间复杂度 $\mathcal{O}(n^2)$。

J 期望步数

对所有串建立 AC 自动机，在 AC 自动机上按照长度顺序 dp。考虑 AC 自动机的某个节点，需要计算两种贡献，一种是当前节点所有后缀的答案和 $dp_{1,u}$（即当前节点 fail 链上所有“有串的”节点的答案和），一种是当前

节点所有前缀的 dp_1 之和 $dp_{2,u}$。如果当前节点有串，这个串的期望步数是 $\frac{dp_{2,u}}{\frac{(n(n+1)}{2}-1} = \frac{dp_{2,u}}{\frac{(n-1)(n+2)}{2}}$。

时间复杂度可以做到 $\mathcal{O}(n)$，但是 $\mathcal{O}(n\Sigma)$，或者 log 求逆元也可以接受。

K 抽鬼牌，打伤害

这个题需要会用 FWT 算异或以及二进制与的逻辑卷积，或者会构造差集的 FWT。

我们用二进制的第 $0 \le i < m$ 位来表示集合中点数 $i+1$ 有无的情况。记 $w(x) = \prod_{k=0}^{m-1}(k+1)[x \wedge 2^k]$。这个用 lowbit 能 DP 出来。

假设已知不同的开局情况下，计算伤害前 Alice 手牌集合为 i 的方案数 p_i，以及 Bob 的 q_i，那么有：

$$d_A = \sum_{i=0}^{2^m-1}\sum_{j=0}^{2^m-1} p_i q_j w(i \wedge \neg j)$$

记 $r_k = \sum_{i \wedge \neg j = k} p_i q_j$，有 $d_A = \sum_k r_k w(k)$。r_k 可以用 FWT 对 p_i 和 $q_{\neg i}$ 做二进制与的逻辑卷积来获得（或者手动构造算差集的变换也可以）。

同理我们也能用 $p_{\neg i}$ 和 q_i 获得 d_B。

那么问题是 p_i 和 q_i 怎么算。

记 $u_i = 2^{a_i-1}$，容易发现 Alice 取 l 到 r 之间牌，手牌的二进制表示就是 u_i 中下标在区间 l 到 r 的异或和，记 u'_i 是 u_i 的异或前缀和，区间 l 到 r 的手牌最终情况也即 $u'_r \oplus u'_{l-1}$。同理也可以有 Bob 的。

记 c_i 为序列 u'_j 中等于 i 的数量。那么有：

$$p_k = \sum_{i<j, i\oplus j=k} c_i c_j$$

即手牌集合为 k 时的方案数，也就是有多少对 u'_r 和 u'_{l-1} 异或为 k。

所以 p_i 可以用 FWT 通过 c_i 和 c_i 自己做异或的逻辑卷积来获得，不过需要减掉掉 $i=j$ 的方案数，并将结果除以 2。同理也能算出 q_i。

对于模 2^{32} 的同时，还要能支持 FWT 除以 2 和处理答案时除以 2 的运算，只需要考虑在 64 位整数下进行计算即可。因为这相当于一直在模 2^{64} 下进行运算，每次对余数除以 2 时，就会让模也同时除以 2，也就是说高若干位的信息就不再正确。但最多会除以 2^{21}，所以并不会影响低 32 位的正确性，直接正常做除法，最后再输出 32 位的整数即可。

时间复杂度 $\mathcal{O}(n + m2^m)$。

L 子集大小和

树上莫队维护颜色个数。

莫队的实现可以考虑用包括退栈信息的欧拉序，然后 u 到 v 的链也就是根到 u 的链、根到 v 的链、$lca(u,v)$ 中出现奇数次的点的集合，根到 i 的链也就是欧拉序的某个前缀。

然后，查询元素个数之和的话，也就是问每个元素所在集合的个数之和。因为每个元素只在它存在的集合中对答案有贡献。

考虑一种颜色，这种颜色每个元素的贡献为，其他颜色的个数加一的乘积。

所以做法是：维护每个颜色的个数，每次更新先把这个颜色的元素全都拿出来，然后对剩下的颜色的贡献进行查询（先除原来这里的元素个数加一，这里元素个数加一，然后乘这里元素个数加一）。然后算自己的贡献。也就是其他元素个数加一后的乘积。除法在总体答案中乘个逆元就好。

时间复杂度 $\mathcal{O}(n^{3/2})$。

M 普通的集合

M.1 推式子

记 $G(n)$ 为 n-普通的集合的个数，则有：

$$G(n) = 1 + \sum_{d|n,\ d<n} G(d) \tag{1}$$

$$S(n) = n \cdot G(n) + \sum_{d|n,\ d<n} S(d) \tag{2}$$

显然 G, S 都不是积性函数，但是可以用杜教筛的求和技巧求一些非积性函数的前缀和。对 (2) 式移项：

$$2S(n) - n \cdot G(n) = \sum_{d|n} S(d)$$

两边求前缀和：

$$
\begin{aligned}
2\sum_{i=1}^{n} S(i) - \sum_{i=1}^{n} i \cdot G(i) &= \sum_{i=1}^{n} \sum_{d|i} S(d) \\
&= \sum_{d=1}^{n} \sum_{d|i} S\left(\frac{i}{d}\right) \\
&= \sum_{d=1}^{n} \sum_{i=1}^{\lfloor n/d \rfloor} S(i) \\
&= \sum_{i=1}^{n} S(i) + \sum_{d=2}^{n} \sum_{i=1}^{\lfloor n/d \rfloor} S(i)
\end{aligned}
$$

记 $T(n) = \sum_{i=1}^{n} S(i)$，$R(n) = \sum_{i=1}^{n} i \cdot G(i)$，整理得

$$
T(n) = R(n) + \sum_{i=2}^{n} T\left(\left\lfloor \frac{n}{i} \right\rfloor\right)
$$

这个式子非常套路，只要数论分块后递归就行。然而里面有个 $R(n)$ 很碍事，我们来考虑一下和它相关的 G 的性质。式 (1) 移项可得：

$$
2G(n) - 1 = \sum_{d|n} G(d)
$$

两边乘 n：

$$
\begin{aligned}
2n \cdot G(n) - n &= n \sum_{d|n} G(d) \\
&= \sum_{d|n} d \cdot G(d) \cdot \frac{n}{d}
\end{aligned}
$$

求前缀和：

$$
\begin{aligned}
2R(n)-\sum_{i=1}^{n} i &= \sum_{i=1}^{n}\sum_{d|i} d\cdot G(d)\cdot \frac{i}{d} \\
&= \sum_{d=1}^{n}\sum_{d|i} d\cdot G(d)\cdot \frac{i}{d} \\
&= \sum_{d=1}^{n}\sum_{d|i} \frac{i}{d}\cdot G\left(\frac{i}{d}\right)\cdot d \\
&= \sum_{d=1}^{n} d \sum_{i=1}^{\lfloor n/d\rfloor} i\cdot G(i) \\
&= R(n)+\sum_{d=2}^{n} d\cdot R\left(\left\lfloor\frac{n}{d}\right\rfloor\right)
\end{aligned}
$$

整理得：

$$
R(n)=\sum_{i=1}^{n} i+\sum_{i=2}^{n} i\cdot R\left(\left\lfloor\frac{n}{i}\right\rfloor\right)
$$

形式和 T 非常相似。至此，我们的问题就顺利地被解决了。

M.2 时间复杂度分析

下面是对时间复杂度的分析。

假定我们预处理了前 k 个 $S(n), G(n)$ 的值，那么类似筛法枚举每个数作为约数做的贡献计算，时间复杂度是 $\mathcal{O}(k\log k)$ 的。

递归过程只用到了 n 的除法分块点处的函数值，因此只需要计算一次复杂度。不难得到 $k>\sqrt{n}$ 时杜教筛的复杂度为 $\mathcal{O}\left(\frac{n}{\sqrt{k}}\right)$，取 $\frac{n}{\sqrt{k}}=k\log k$ 并视 $\log k$ 与 $\log n$ 同阶，得 $k=\left(\frac{n}{\log n}\right)^{2/3}$，此时有预处理复杂度和递归复杂度同为 $\mathcal{O}(n^{2/3}\log^{1/3}n)$。

综上，单次计算时间复杂度为 $\mathcal{O}(n^{2/3}\log^{1/3}n)$。

M.3 微小的优化

预处理时使用高维前缀和 dp（将每种质因子看作一维）。以 G 的预处理为例，记 $dp_{u,p}=\sum_{d|u,\mathrm{prop}(u,d,p)} G(d)$，其中 prop 是指这样的性质：对于 u 的每种小于等于 p 的质因子，d 含有的数量小于等于 u 含有的数量；对于 u

的每种大于 p 的质因子，d 含有的数量等于 u 含有的数量。举例来说：

$dp_{12,1}$	$G(12)$
$dp_{12,2}$	$G(3)+G(6)+G(12)$
$dp_{12,3}$	$G(1)+G(2)+G(3)+G(4)+G(6)+G(12)$

设 u 的唯一因子分解为 $u=p_1^{s_1}p_2^{s_2}\cdots\cdot p_t^{s_t}$，那么 $G(u)=1+\sum\limits_{i=1}^{t} dp_{x_{u,i},p_i}$，其中 $x_{u,i}=u/p_i$。举例来说：

$$\begin{aligned}G(12)&=1+dp_{4,3}+dp_{6,2}\\&=1+(G(1)+G(2)+G(4))+(G(3)+G(6))\\&=G(12)\end{aligned}$$

dp_u 的计算方式为：

$$\begin{aligned}dp_{u,1}&=G(u)\\dp_{u,p_i}&=dp_{u,p_{i-1}}+dp_{x_{u,i},p_i}\end{aligned}$$

可以发现，对于每个 u，计算 $G(u)$ 的复杂度为 $\mathcal{O}(\omega(u))$，其中 $\omega(u)$ 表示 u 的不同质因子数。根据 Mertens' second theorem，预处理的时间复杂度为 $\sum_{i=1}^{k}\mathcal{O}(\omega(i))=\mathcal{O}(k\log\log k)$。

因而总时间复杂度为 $\mathcal{O}(n^{2/3}\log\log^{1/3}n)$。

N 风与牧场与集市

N.1 题解

方便起见，先暂时不考虑潜在关联的限制。

记 $d_i\in\{0,1\}$ 表示产品 i 是否选中，则题目要求最大化

$$\begin{aligned}\frac{(\sum_{i=1}^n x_id_i)^2}{\sum_{i=1}^n x_i^2d_i^2}=&\frac{\sum_{i=1}^n x_i^2d_i^2+\sum_{i=1}^n\sum_{j=1}^n[i\neq j]\cdot x_id_i\cdot x_jd_j}{\sum_{i=1}^n x_i^2d_i^2}\\=&1+\frac{\sum_{i=1}^n\sum_{j=1}^n[i\neq j]\cdot x_ix_j\cdot d_id_j}{\sum_{i=1}^n x_i^2\cdot d_id_i}\\=&1+\frac{\sum_{i=1}^n\sum_{j=1}^n[i\neq j]\cdot x_ix_j\cdot p_{ij}}{\sum_{i=1}^n x_i^2\cdot p_{ii}}\end{aligned}$$

其中 $p_{ij}=d_id_j$，故只需最大化后面的式子。注意到后者实际上是一个含有 n^2 个变量的 01 分数规划，且 $\sum_{i=1}^{n} x_i^2 \cdot p_{ii}$ 的值恒大于等于 0，故可以二分这个式子的答案 ans。若存在某个合法 p 序列的取值，使得

$$
\begin{aligned}
&\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}[i\neq j]\cdot x_ix_j\cdot p_{ij}}{\sum_{i=1}^{n} x_i^2\cdot p_{ii}} > ans\\
\Leftrightarrow\ &\sum_{i=1}^{n}\sum_{j=1}^{n}[i\neq j]\cdot x_ix_j\cdot p_{ij} - ans\sum_{i=1}^{n} x_i^2\cdot p_{ii} > 0\\
\Leftrightarrow\ &\sum_{i=1}^{n}\sum_{j=1}^{n}([i\neq j]\cdot x_ix_j)\cdot p_{ij} - \sum_{i=1}^{n}(x_i^2\cdot ans)\cdot p_{ii} > 0
\end{aligned}
$$

那么说明当前的 ans 还不够优，继续向上二分即可。

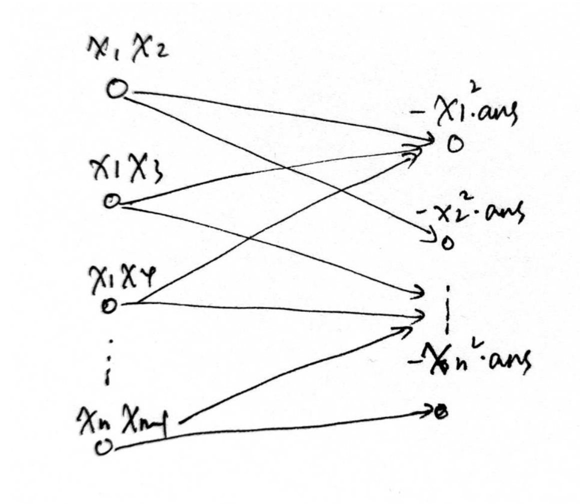
上式等价于：有 n^2 个物品可以选或不选，选中物品 p_{ij} 的权值是 x_ix_j，选中物品 p_{ii} 的权值是 $-x_i^2\cdot ans$，且若选中 p_{ij}，则也需要选中 p_{ii} 与 p_{jj}，判断是否能让选中物品的权值和大于 0。

显然，这是一个最大权闭合子图问题，使用最大流算法就可以解决。现在考虑潜在关联带来的性质，则限制 (u,v) 等价于选中 p_{uu} 就要选中 p_{vv}，因此仍然是一个最大权闭合子图上的限制，解法相同。关于最大权闭合子图的正确性证明与网络流时间复杂度的分析，将在后文叙述。

综上，我们只需要二分 ans，并每次验证对应建图的最大权闭合子图权值和是否大于 0，即可完成对原始问题的求解。

N.2 最大权闭合子图的正确性证明

最大权闭合子图的选取可能存在这样的问题：某个 x_i^2 与 x_j^2 的项被选择了，x_ix_j 的项却没被选择（这是满足最大权闭合子图的要求的）。接下来我们会证明这种情况不存在。



显然，左侧的点权值一定大于 0，右侧的权值一定小于 0，因此如果出现选了某个 x_i^2 与 x_j^2 的项却没选 x_ix_j 的项，那么选中 x_ix_j 项一定会让结果更优。因此对于所有选定的平方项，它们两两对应的交叉项也一定会被选中。

N.3 时间复杂度分析

按照上述方式建图，图中的点数 V 与边数 E 都是 $\mathcal{O}(n^2)$ 级别的。

为了证明采用 Dinic 方法的最坏时间复杂度是 $\mathcal{O}(n^4)$，我们只需证明重建层次图这一步骤只会进行 $\mathcal{O}(n)$ 次，寻找增广路的复杂度是 $\mathcal{O}(n^3)$。

设源汇点分别是 S,T，对于增广过若干次的图，从某个 x_i^2 出发，它合法的后继都只有 $x_i^2 \to x_j^2$、$x_i^2 \to x_ix_j \to x_j^2$、$x_i^2 \to T$ 三种选择。同时分层图是个 DAG，故每个 x_i^2 项只会经过最多一次，因此 $S \to T$ 的最短路长度最坏是 $\mathcal{O}(n)$ 级别的。

寻找增广路的复杂度为 DFS 的复杂度与修改增广路上流量的复杂度。对于前者，在当前弧优化下为 $\mathcal{O}(E)$；对于后者，由于最多找 $\mathcal{O}(E)$ 次增广路，每次增广路最坏长度为 $\mathcal{O}(n)$，故复杂度为 $\mathcal{O}(En) = \mathcal{O}(n^3)$。

综上，原问题的总时间复杂度为 $\mathcal{O}(n^4 \log \dfrac{1}{\varepsilon})$，其中 ε 为二分的精度。