A. Cut The Wire

std 运行时间：46MS，最快验题提交：46MS.

分两部分计算。

- 偶数部分，是 $n - \frac{n}{2}$。
- 奇数部分，设 $m = \frac{n-1}{3} + 1$，若 m 是奇数，则答案为 $\frac{n-m}{2} + 1$，若 m 是偶数，则答案为 $\frac{n-m+1}{2}$。

这里的除都是整除，下取整。

B. Time-Division Multiplexing

std用时：703ms，最快验题程序：703ms

注意到每个时隙的字符串长度不超过12，因此所有时隙总的循环周期是所有字符串长度的lcm，最大为$5 * 7 * 8 * 9 * 11 = 27720$。

问题就变成了对一个循环周期为$27720n$的字符串求解，这部分可以用简单的双指针解决。

需要注意的是，这个字符串是一个有循环节的无限长的串，因此需要对两倍长度的循环节求解。没有考虑两倍长度的队伍可以试试这个样例，答案应该是11而非14.

$$
\begin{array}{l}
1 \\
10 \\
caa \\
daaa \\
eaaaa \\
faaaaaaaaaa \\
gaaaaaa \\
haaaaaaa \\
iaaaaaaaa \\
jaaaaaaaaaa \\
kaaaaaaaaaa \\
al
\end{array}
$$

C. Pattern Recognition

std 运行时间：2015MS

我们只考虑将查询字符串先从左向右横着放，然后再向下翻折的情况（其他情况类似）：

我们首先将矩阵的行按照从上到下、从右到左的顺序拼在一起，不同行之间用特殊字符隔开，求出后缀数组记为A。再将矩阵的列按照从左到右、从上到下的顺序拼在一起，不同列之间用特殊字符隔开，求出后缀数组记为B。对于后缀数组A中的每一个后缀$A[i]$，若它的起点在矩阵的位置为(x,y)，我们就将$(x+1,y)$对应的后缀在B中的下标赋值给$C[i]$。对于一个查询字符串t，我们枚举它翻折的位置，假设翻折前的部分为$t[1,i]$，翻折后的部分为$t[i+1,|t|]$。这样我们可以用二分找到$t[1,i]$的反串在A中的区间$A[j_1,k_1]$。我们还可以通过二分找到串$t[i+1,|t|]$在B中对应的区间$B[j_2,k_2]$，那么这种形状的串出现的次数就转化为在$C[j_1,k_1]$里有多少个数在区间$[j_2,k_2]$里，用主席树即可解决。

对于其他的情况也是类似的，可以将这些情况合在一起做。时间复杂度$O(nmlognm+qlognm)$，其他类似做法，甚至多一个log的做法也可以通过本题。

D. Depth First Search

std 运行时间：1546MS, 分块： 5078MS

对于原树中的每一个点x，我们将其拆成LCT中的2个点，记为x_1和x_2，权值分别为x和0。如果x是原树中的一个叶结点，记y为x在原树中的广义右兄弟（BFS序中x后面的结点），从x_1和x_2向y_2连一条边。如果x是在原树中有孩子，记y为x在原树中最靠左的孩子，从x_1和x_2向y_1连一条边。LCT中维护结点的权值和、权值最大值、在原树中的深度最大值，这样对于一次查询$3\ x\ d$，我们将x_1 access到根，然后在这个Splay里走，找到在原树中深度$>=d$的第一个位置。这样我们找到了起点和终点，将这条链提取出来就能得到答案，时间复杂度$O(QlogQ)$。

一些要注意的细节：

- 为了能够快速找到一个点的广义右兄弟，我们考虑用链表来维护这棵树的BFS序。
- 我们在所有操作开始之前，先以1为根，在最右侧插入一条长度为Q的链，这样可以保证之后插入的结点一定有广义右兄弟。
- 当向x的孩子里加入y，但是此时x没有孩子结点，这种情况没法在链表里直接找插入的位置。我们可以考虑在LCT中找到在原树中深度$>=dep[x]+1$的第一个点，插入在这个点前面就行，而这个查找操作和查询操作(type 3)是一样的。

这道题还可以使用块状链表进行维护。 考虑在进出都做记录的 DFS 序上进行维护，那么每个查询即为从左到右在 DFS 序上扫描子树，如果当前节点的深度为第一次遇到，那么就将该节点做记录，如果超过查询深度那么就可以返回了。 如果使用块状链表进行维护，按照 $\sqrt{Q}$ 进行分块。那么一个块内的深度变化不会超过 $\sqrt{Q}$。对于每一个块内，维护该块内每个深度第一次出现的点。然后做后缀和和后缀 $\max$ 。 对于插入和删除操作，直接进行删除和插入，然后重建块。 对于一个查询，从左往右暴力，记录当前已经处理的深度。对于最左边块外的，直接暴力。然后对于块内部分，如果块最大深度小于查询的深度，那么可以通过后缀和和后缀 $\max$ 直接进行处理。否则说明块内就可以处理完，直接暴力处理块就可以了。对于右侧的也直接暴力。 这样一来就可以在 $O(\sqrt{Q})$ 的时间内完成查询和插入删除。总复杂度 $O(Q\sqrt{Q})$。验题人的未经任何优化的代码跑了 5078MS。

E. Easy Math Problem

std 运行时间：1250MS，最快验题提交：718MS

首先看到，题目中涉及递推式，考虑将两个不同的递推写成矩阵 a 和 b，那么只需要求出

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\binom{i+j}{i}\cdot a^i b^j$$

到这一步其实就可以使用 FFT 或者 NTT 拆解矩阵进行计算了，复杂度是 $O(Tn\log_2 n)$，时间可能会比较卡，需要一个常数非常好的板子才能通过。

下面来介绍 std 的做法。

可以从组合数的角度入手。如果只计算一行的话还是可以 $O(N)$ 算出来的。那么考虑第 i 行和第 $i+1$ 行的关系。令 $f(i)=\sum_{j=1}^{n}\binom{i+j}{i}\cdot a^ib^j$，那么 $f(i+1)=\sum_{j=1}^{n}\binom{i+j+1}{i+1}a^{i+1}b^j$。考虑到 $\binom{i+j+1}{i+1}=\binom{i+j}{i+1}+\binom{i+j}{i}$，那么错位搞一搞可得

$$f(i)+a^{-1}f(i+1)b=a^{-1}f(i+1)+\binom{i+n-1}{i+1}a^ib^{n-1}$$

这样我们就能得到一个 $f(i)$ 关于 $f(i+1)$ 的递推式，预处理两个矩阵的幂就可以递推了。复杂度 $O(Tnw^3)$，其中 w 为矩阵的大小，这里为 2。

F. Power Sum

std用时：171ms,最快验题程序：15ms。

一个显然正确的做法:$(x+1)^2-(x+2)^2-(x+3)^2+(x+4)^2=4$. 根据$n\pmod 4$的结果对$1,2,3,4$进行构造，然后每次通过上述恒等式进行加四操作即可。

另外，用k以内的完全平方数，能构造出2^k个结果，排除掉超出边界的和重复的数字，也能有很多不同的结果，因此本题也有很多其他乱搞的做法。

验题人的一种做法是，从第一个满足$\sum_{i=1}^{k}i^2\geq n$的k开始枚举，先令所有元素都为正，再贪心地将一些正数改为负数，也能找到合法的解。

由于采用类似std的做法输出量很大，因此需要直接输出字符串而非一个int一个int地输出，后者需要几分钟才能跑完。

G. Function

std 运行时间：109MS，最快验题提交：46MS

$\forall x\in[1,10^6],g(x)_{max}=g(999999)=54$，也就是说$g(x)$只有54种不同的值。我们枚举$g(x)$的值令其为$i$，当$i$确定之后$f(x)$就是一个二次函数了，剩下的就是一个二次函数求最值的问题了。但是对称轴处的x，不一定满足$g(x)=i$，这时我们要找对称轴左侧和右侧最近的满足$g(x)=i$的x，这可以通过预处理出满足$g(x)=i$的所有x，然后在这个数组上二分来实现。注意讨论二次函数的开口方向以及二次项系数为0的情况。

H. GCD on Sequence

std 运行时间：2203MS，最快验题提交：1656MS

本题定义 $v(l,r)=\max\limits_{l\le i<j\le r}\gcd(a_i,a_j)$。问每一种不同的 x 有多少个区间 $[l,r]$ 的值与 x 相等。

首先可以注意到，对于一个固定的左端点，如果右端点向左移动，答案会逐渐变小。然后如果考虑一个区间的值要大于等于 x，那么这个区间必定横跨两个 x 的倍数。

那么考虑从大到小枚举 gcd 的值，去维护每个左端点还未确定答案的右端点。假设当前枚举的 gcd 为 g，处理出所有的 g 的倍数的位置 $\{p_i\}$。从左到右处理，对于 p_i 和 p_{i+1}。那么可以将所有小于 p_i 的所有点的未确定答案

的右端点与 $p_{i+1}-1$ 取最小值。这样一来，处理 g 之前里所有位置的和减去处理之后的和即为 v 为 g 答案。

这样一来只需要维护一个区间取 $\min$。这样其实已经可以使用 Segment Tree Beats 进行维护。

不过其实注意到，所有未确定答案的右端点一定是单调递增的，所以可以直接在线段树上二分确定需要区间覆盖哪个区间。维护区间覆盖和区间 $\max$ 以及区间求和即可。

由于 1 到 n 一共 $n\log n$ 个因数，因此时间复杂度为 $O(Tn\log^2 n)$。

其实这道题使用线段树上二分这种做法还可以做到离线之后区间查询，不过考虑到题目难度的原因所以没有加上。

I. Command Sequence

std 运行时间：78MS，最快验题提交：78MS.

定义 (a_i,b_i)，其中 $a_i=\sum_{j=1}^{i}[s[j]==U]-\sum_{j=1}^{i}[s[j]==D]$ ，
$b_i=\sum_{j=1}^{i}[s[j]==R]-\sum_{j=1}^{i}[s[j]==L]$。

(ai,bi) 可以通过前缀和来求。

那么 $ans=\sum_{i=0}^{n}\sum_{j=i+1}^{n}[(a_i,b_i)==(a_j,b_j)]$， 这可以通过排序或者 map 来求。

复杂度 $O(n)$ 或者 $O(nlogn)$。

J. Random Walk

std 运行时间：1750MS，最快验题提交：1453MS

我们首先建立一个超级源s，令从s走到点x的概率为$\frac{w_x}{\sum_{i=1}^{n-1}w_i}$，这样问题就转化成了从$s$开始随机游走，一直走到$n$时期望捡到的硬币个数。一条边上的硬币数量最多会在$\lfloor\log_2 100\rfloor=6$ 秒之后变成定值b_i。我们考虑一条边对答案的贡献，设它的两个端点分别为A和B，从s走到n的过程中在6秒之后经过A的期望次数为$E[A]$，经过B的期望次数为$E[B]$，则在6秒之后的时间里这条边对答案的贡献为$(\frac{E[A]}{degree(A)}+\frac{E[B]}{degree(B)})\times b_i$，其中$degree(x)$表示点$x$的度数。$E[x]$可以通过一次高斯消元得到，注意到高斯消元时系数矩阵不会因为从s走到x的概率改变而改变，所以我们可以先求出矩阵逆，或者离线把所有类型2的修改拿出来一起消元。对于6秒之前的情况可以类似的处理，具体的，$\forall 1\le t\le 6$，我们可以通过一个DP来计算出在时刻t到达点x的概率$P(x,t)$，这样对于一条边(A,B)在时刻t对答案的贡献就是$(\frac{P(A,t)}{degree(A)}+\frac{P(B,t)}{degree(B)})\times max\{\lfloor\frac{a_i}{2^{t-1}}\rfloor,b_i\}$。这样对类型$1$的修改，我们只需要重新算一遍这条边对答案的贡献，时间复杂度为$O(1)$。对类型2的修改，我们要算一次矩阵乘，还要重新算一遍DP，时间复杂度为$O(n^2+m)$。

K. Shooting Bricks

std 运行时间：234MS，最快验题提交：203MS

经典背包问题.

$dp[i][j][0]$ 为前 i 列，用了 j 颗子弹，前 i 列，不存在一列打的最上面的砖块是无奖励的。

$dp[i][j][1]$ 为前 i 列，用了 j 颗子弹，前 i 列，存在一列打的最上面的砖块是有无奖励的。

转移就是背包的转移方式。

我们考虑假设我们打的一个方案中，不存在一列打的最上面的砖块是无奖励的则，则我们需要有 $j+1$ 颗子弹。若存在一列打的最上面的子弹是无奖励的，则需要 j 颗子弹即可，因为我们可以将这列放在最后去打。

所以答案即为 $ans = max(dp[n][k-1][0], dp[n][k][1])$。

复杂度 $O(nmk)$。

L. Remove

std 运行时间：453MS，最快验题提交：406MS。

该题是一道贪心题目。

首先我们可以知道，当 $i >= lcm(pi)$，时无解$ans_i = 0$。

然后在有解的情况下，ans_i 是单调不减的，可以用归纳法证明。

所以每次我们选择的 p 应该是使 $m \bmod p$ 最大的 p。

考虑一个 x，若有一个因子 p，在所给的模数选择集合中，则 x 可以转移到 $[x+1, x+p-1]$之间的位置，所以我们在所给的模数集合中的素因子找一个最大的 p，则为 x 可以转移到的最大范围。

有了这个我们倒着做一遍贪心，即可算出所有的 ans_i。

求最大的素因子，可以用线性筛来求，复杂度 $O(n)$。

因为本题给的模数集合中的数字都为素数，也可以直接枚举素因子，然后枚举倍数，来求最大素因子，复杂度是 $O(nloglogn)$。

其实这道题还可以这样，对于每一个质数 p，维护其当前最大值。从 1 开始往大扫，将所有的质数及其可以转移到的点的答案放入一个小根堆维护。对于一个 i，取出堆顶的答案，如果该答案的质数为 i 的因子，那么将其pop出来。如果该质数的值已经发生了改变，那么也将其 pop 出来，并用其正确的答案更新 i 的答案。如此操作直到无法 pop 为止。那么假如堆顶有东西，那么可以将 i 的答案和其转移过来的答案取 $\min$。然后再将所有 pop 出来的质数的答案更新再放回堆。

这样乍一看是 $n \log n$的。但是其实并非如此。可以发现，答案序列一定是单调递增的。而转移只能是 $+1$ 转移。那么可以得到 $ans[i] \leq ans[i-1] + 1$所以假如最大的素数为 p，那么有 $ans[p-1] = 1$，$ans[p] = 2$，因此有 $ans[2p-1] \leq 3$，则 $ans[2p] \leq 4$。

所以对于 N 的答案 $ans[N]$ 有 $ans[n] \leq 2\lceil \frac{n}{p} \rceil$。

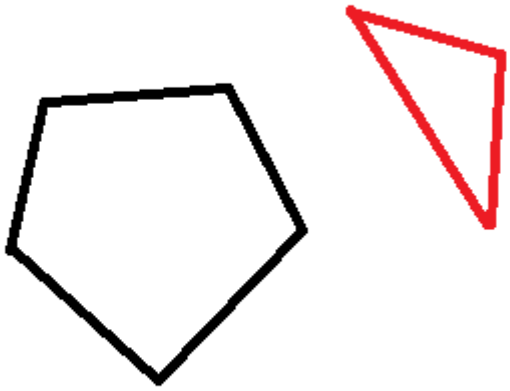
也就是说每个质数最多被 pop 出来 $2\lceil \frac{n}{p} \rceil$ 次。所以复杂度为 $O(2\lceil \frac{n}{\max P} \rceil |P| \log |P|)$。由于 P 为质数集，因此时间复杂度也是合理的。

M. Start Dash ! !

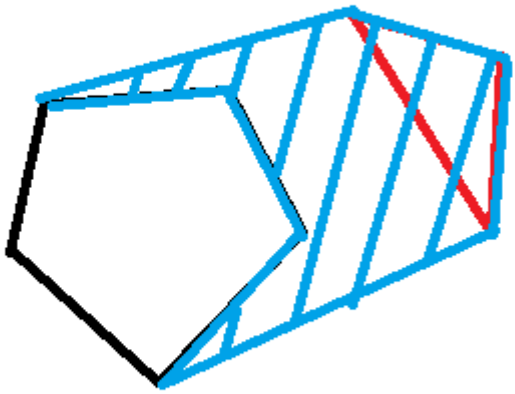
std 运行时间：1437MS，最快验题提交：828MS。

首先明确一下问题，这道题求的是在已知一个凸包的前提下，每次查询给出一个三角形，求平面内可以和三角形内的点构成射线穿过凸包但构成线段不穿过凸包的点的面积。

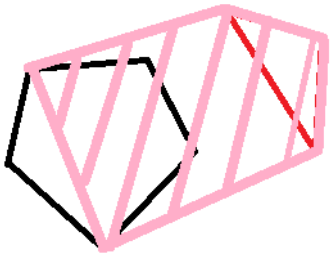
形状大概是这样：

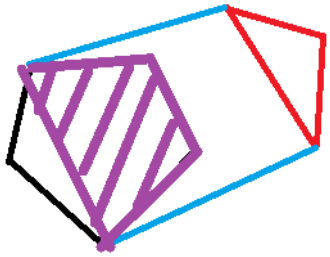


要求的区域大概是这样：



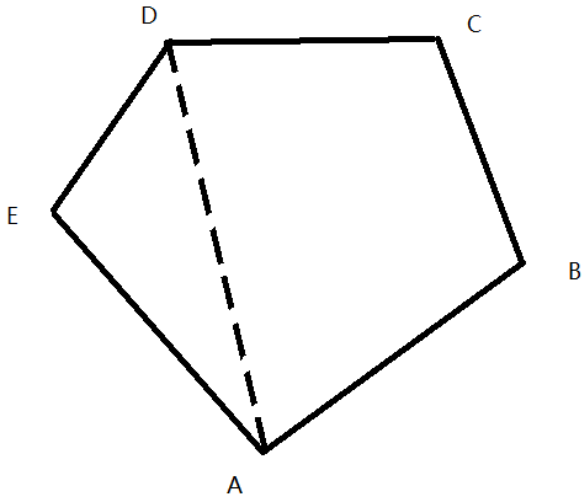
那么答案可以看作是粉色区域减去紫色区域。





首先可以考虑如何求解粉色区域的面积，注意到凸包上的那两个点(也有情况是一个点)是三角形和凸包公切线的切点，只要找到这俩点那么粉色区域的面积就求到了。由于三角形只有三个点，因此有一个很显然的办法是拿三角形三个点分别去求切点然后拿这9个点求个凸包拿到面积。或者枚举两点近点，求这两个近点和原凸包的新凸包，再合并上远点最后取max都行，方法很多。至于求切点的方法，常规的二分或者三分法都可以，都能通过，本题的时限很宽只要复杂度读了应该都行。std使用的方法是记录原凸包的上下链，根据每次询问的点的位置进行讨论，大致分为两个切点都在上凸壳，两个切点都在下凸壳和上下凸壳各一个点的情况，每个切点用一次二分来求。该问题比较经典，可以参照网上的各类解法，本题解中不详细阐述。

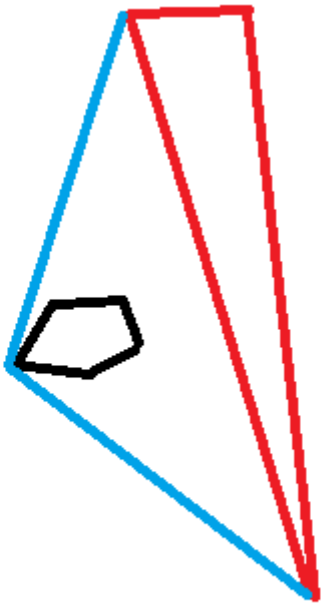
接下来考虑粉色区域的面积，这个区域就是凸包上一段连续的点构成的凸多边形面积。这个部分观察一下求这部分面积的式子，发现可以分解成叉积的前缀和来求，这样基本就做完了。



如上图，假设要求$ABCD$区域的面积，那么可以分解为$(B-A)\times(C-A)+(C-A)\times(D-A)$，由于$A\times A=0$最后可以化简为$B\times C+C\times D-(B+C)\times A-A\times(C+D)$，仅需要维护相邻两个点的叉积的前缀和和凸包上点向量的前缀和即可。

以上做法的复杂度是$O(q*log(n))$

最后把上述的一个点的情况贴一下：



这种情况的答案就是得到新凸包的面积（也就是上文中粉色区域的面积）减掉原凸包的面积。

这道题std在杭电的比赛机子上跑了1437ms，验题人最快跑了828ms，也没有什么很坑的数据，考虑到今年多校出现了求凸包切线的题所以本来觉得这道题应该不会很难，毕竟本质板子题的求切线和叉积前缀和。在最开始的设想里多边形不是凸的，后来考虑到题目整体难度改成了凸包，通过率低于预期可能是今天hduoj崩了大家都不想做计算几何吧XD。

最后提一句$\mu' sic\ forever$，祝大家今后计算几何顺利。