A. Kaguya with Ice Creams

算法：二维费用背包问题

这道题是很明显的 0-1 背包问题，可是不同的是选一个雪糕会消耗**两种价值**（生产成本和生产时间）。这种问题其实很简单：方程基本不用变，只需再开一维数组，同时转移两个价值就行了！（完全、多重背包同理）

这时候就要注意，再开一维存放雪糕编号就不合适了，因为容易 MLE。

B. Kaguya with Max Sum

算法：一维数组前缀和

一维数组前缀和的基本处理流程：

把对数组 A 的累加依次放入数组 B 中---B[i] = B[i-1] + A[i]，B[0]=A[0]。

这里要求计算k个连续的和的最大值，计算过程如：

$$ans = f\{max(ans, B[k+i] - B[i]) \mid 0 \leq i < n \; and \; k+i < n\}$$

ans的初值为B[k-1]，因为可能A数组的首项也可能属于符合条件的k个数！

C. Kaguya with Strings

算法：string函数的应用

本题主要考查的是同学们灵活应用string类型函数的能力，这里使用的是**substr**和**rfind**函数，substr用来截取子字符串，rfind函数用来从后面开始寻找子字符串的位置（**注意一下rfind和find的使用区别，想一想为什么这里使用rfind更合适**）。从字符串起始位置 $i(0 \leq i \leq s.size()-1)$ 开始截取一段长度为 $L(0 < L \leq s.size()-1)$ (L 从大到小开始枚举)子字符串 $str1$ ，然后在 i 后面的位置（不能包含 i 位置）寻找是否存在子字符串 $str2$ ，满足 $str1 == str2$ ，如果存在则输出 L ；反之，如果遍历 s 没有找到符合条件的两个子字符串，输出 0 。

D. Kaguya with ⑨

前置知识：类似于能被 3 整除的数，各个数位加起来也能被 3 整除，9 也有相同的性质。 先不考虑最大化数，只考虑最大化位数，因为相同位数下将所选的数从大到小排是显然最优的。 原问题转化为取一些数，使得这些数的和 $\%9=0$ ，显而易见的结论是 0 和 9 全要，但是其他数有太多组合可以组出 9 ，对于不同的情况，他们的优先级并不唯一，比如 1 个 8 和 9 个 1 时，$8+1$ 和 9 个 1 相加，这两种情况的优先级是后者更高。贪心是很容易进也很容易发现的坑。 考虑 dp ，常规思路，加一个数，有两种决策取或不取。并且我们不在意取的是什么数，只需要知道前 i 个数，取一些加起来的余数即可。对 9 的余数只有 $0-8$ 这九种情况，存下每种余数情况下，前 i 个数中最多能取多少个达到这个余数。再试着加一个数进行转移。即最后只需要维护一个 $dp[i][\text{余数}]$，第一维可以滚动。 最后需要一个构造解，在 dp 转移取优时记录前向路径。

E. Kaguya with Lord

3 分钟签到题。 一个循环 01 数列，循环节为 7 ，求使得某连续段的和为 k 时的最小段长。 记下完整的一周会上几天课，得到上课所需的完整周数，小心处理下两端的不完整周即可。

F. Kaguya with Sweet Gem Berry

首先观察得到 x 和 y 两个轴是完全独立互不影响的，分开计算答案。 以 x 轴为例，需要找到一个 X ，使得所有矩形的 $x_l - x_r$ 这条线段，覆盖它。 一条线段 $[l, r]$ ，移动到 X ，若线段在其左边，需要 $r - X$ ，若线段在其右边，需要 $X - l$ ，还有已覆盖的情况。 统一一下移动代价是 $(|r - X| + |X - l| - (r - l))/2$ ，提取出与X相关的部分，对于所有矩形，只需要最小化 $\sum(|r - X| + |X - l|)$ 亦即，l 和 r 在地位上是没有区别的，所有的 $2n$ 个 l 或 r，都需要与 X 作差取绝对值求和。 所有数与中位数的绝对差之和最小，X 取所有 l 和 r 的中位数就是答案。 实际做的时候只需要拍脑袋猜结论，暴力或者手算验证就能快速A掉。

G. Kaguya with 射手座之日II

带权并查集。

维护每艘战舰前面的战舰数、头部、尾部。

前面的战舰数在路径压缩的时候就一起算上，查询的时候类似差分搞一搞就完了。

H. Kaguya with A Simple Problem On Different Numbers

树状数组/线段树。

如果稍微有一点经验就会发现，a_i 的值域太大了，很难在线地回答。(尽管有在线做法)

考虑离线做法，把询问按右端点从小到大排序，依次回答。回答右端点为 r 的询问时，维护好前 r 个数自己是否为从右往左第一个出现的数。

维护操作很简单，因为询问是从左往右回答的，当维护到前 $k - 1$ 个数时，数 a_k 一定是从右往左的第一个 a_k ，如果 a_k 曾经在位置 p 出现过，那么就 $add(p, -1)$ 。然后无论它是否出现过，执行 $add(k, 1)$ 来标记他是从右往左第一个出现的。重复这个操作直到维护完前 r 个。

此时这个询问的答案就是 $sum(r) - sum(l - 1)$。

I. Kaguya with A Simple Problem On Graph

BFS 或 DFS 随便搞。

把图黑白染色，如果相邻两点同色则一定无解。

每个联通块内要删掉的点的个数就是 $min(\text{黑点数}, \text{白点数})$ 。因为每条边的两侧一定是一个黑点一个白点，而且一旦删了黑点就不可能再删白点了，反之亦然。

J. Kaguya with bubbles plus

由于涛哥没把模数设置成质数，导致有位同学把答案的循环节找到了orz。

以下是涛哥的正解。

假设序列长为 L 时答案的数目为 $f(L)$ ，显然有

$f(0)=0, f(1)=2, f(2)=4, f(3)=6, f(4)=9$ 。

当 L 较大时，假设第 L 是 o 那么只要 $L-1$ 符合条件就好，有 $f(L-1)$ 种情况。

如果第 L 个是 O ，由题意，考虑前面两个，总的情况有 OOO, OoO, oOO, ooO 但是 OOO, OoO 不和条件，只考虑 ooO, oOO两种情况

对于 ooO ，只要前面 $L-3$符合条件就好，有 $f(L-3)$ 种情况。

对于 oOO，第 $L-3$ 个只能是 o ，那么只要前面 $L-4$ 个符合条件就好，有 $f(L-4)$ 种情况

综上推出 $f(L)=f(L-1)+f(L-3)+f(L-4)$，由于 n 的范围是 $1e12$ ，用矩阵快速幂实现 $O(logn)$

$$\begin{pmatrix} f(n) \\ f(n-1) \\ f(n-2) \\ f(n-3) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} f(n-1) \\ f(n-2) \\ f(n-3) \\ f(n-4) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}^{n-4} \times \begin{pmatrix} f(4) \\ f(3) \\ f(2) \\ f(1) \end{pmatrix}$$

K. Kaguya with climb

将所有节点编号看成二进制。

若两个节点的值不相等，则将较大节点的编号右移移位。

重复过程直到两个节点相等。

(是不是很像倍增求 LCA)

L. Kaguya with fbgcd

古人的智慧，由更相减损术得 $gcd(x, y)=gcd(y, x+y)$ 。

那么 $gcd(F_n, F_{n+1})=gcd(F_1, F_2)$ 。