A. Kaguya with Spring Outgoing

算法：简单的数学思维

对所有学生的身高进行从小到大进行排序，然后可以从第一个同学 a_1 往后寻找与 a_1 身高差在 t 以内的同学，不断累加这样的同学个数 cnt，第一轮遍历完数组之后，得出 $ans = ans + cnt \times 2$ (因为可以相互交换做队长，所以 cnt 需要乘以 2)。后面依次类推，直到遍历完整个数组为止。

B. Kaguya with LELECHA

算法：三分

咋一看，很多同学可能会使用二分去做，但是三分其实更适合本题。

举个例子：如果有 9 杯奶茶，二分的做法就是分成 $4-5$。因为数量不相等，称出来的结果没有可比性，所以分成 $4-4-1$；第一次称，如果 4 和 4 相等，则多出来的那一杯就是答案（这是比较好的结果），答案是 1；而如果 4 和 4 不相等，则需要把比较重的那 4 杯在分为 $2-2$ ，第二次称出比较重的那 2 杯；然后需要第三次称才能找到需要找到的那杯，所以需要 3 次才必能找到。

如果采用 3 分去做，就可以把 9 杯分为 $3-3-3$ 。第一次称前面两个 3 杯，如果相等，则把另外一个 3 杯奶茶分为 $1-1-1$ ，在称重前面两个 1 ，如果相等，说明第 3 个 1 是要找的答案，如果不相等，则也能找出哪一杯是需要寻找的那一杯，这里一共只要称 2 次就能出答案；如果第一称的时候前面两个 3 杯不相等，则找出比较重的那个 3 杯，分为 $1-1-1$ ，在称一次也能得到结果。所以，无论是上面那种情况，称 2 次必须答案。可见 3 分更适合本题！其实究其原因，就是 2 分每次可以剔除二分之一的数据区间，而 3 分可以剔除三分之二的数据区间，所以比较次数可能会更少。

知识点延伸：

在二分查找的基础上，在右区间（或左区间）再进行一次二分，这样的查找算法称为三分查找，也就是三分法。

三分查找通常用来迅速确定最值。

二分查找所面向的搜索序列的要求是：**具有单调性（不一定严格单调）；没有单调性的序列不是使用二分查找。**

与二分查找不同的是，三分法所面向的搜索序列的要求是：序列为一个凸性或者凹性函数。通俗来讲，拿凸性函数来说，该序列必须有一个最大值，在最大值的左侧序列，必须满足不严格单调递增，右侧序列必须满足不严格单调递减。

三分函数的一般写法：

```
double three_devide(double low,double up)
{
    double m1,m2;
    while(up-low>=eps)
    {
        m1=low+(up-low)/3;
        m2=up-(up-low)/3;
        if(f(m1)<=f(m2))//f**为计算函数**
            low=m1;
        else
```

```
                up=m2;
    }
    return (m1+m2)/2;
}
```

C. Kaguya with Onmyoji

算法：二维数组前缀和

定义一个矩阵 a，可以视为二维数组：

1 2 4 3

5 1 2 4

6 3 5 9

再定义一个矩阵

$$sum_{x,y}=\sum_{i=1}^{x}\sum_{j=1}^{y}a_{i,j}$$

那么这个矩阵长这样：

1 3 7 10

6 9 15 22

12 18 29 45

第一个问题就是递推求 $sum_{x,y}$ 的过程:

$$sum_{x,y}=sum_{x-1,y}+sum_{x,y-1}-sum_{x_1-1,y_2}+sum_{x-1,y-1}+a_{x,y}$$

第二个问题就是如何应用，譬如求 $(x_1,y_1)-(x_2,y_2)$ 的子矩阵的和。 那么，根据类似的思考过程，易得答案为：

$$sum_{x_2,y_2}-sum_{x_2,y_1-1}-sum_{x_1-1,y_2}+sum_{x_1-1,y_1-1}$$

所以题目的推导公式易得：

$ans=max(ans,d[i+r-1][j+c-1]-d[i-1][j+c-1]-d[i+r-1][j-1]+d[i-1][j-1])$

然后对r和c进行枚举（**注意：r和c的最大值不超过n和m**），用 $ans-r\times c\times t$ 更新答案，输出最大值即可。

D. Kaguya with infinity

意思是给你一个奇怪的序列，让你求区间内序列的奇数的个数。

我们不妨用 0 表示偶数，1 表示奇数。

手玩一下这个序列，你会发现k为奇数时序列长这样：

$1,1,1,1,1,1,1,1,1...$

答案显然是 $r-l+1$ 。

当 k 为偶数时，比如 $k=4$:

$1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,1...$

答案是区间长度减掉区间内模 $k+1$ 为 k 的数的个数。

若 $x \equiv k(mod\ (k+1))$ 则 $x+1 \equiv 0(mod\ (k+1))$ 。

个数是

$$\lfloor \frac{r+1}{k+1} \rfloor - \lfloor \frac{l}{k+1} \rfloor$$

那么答案就是

$$r-l+1-\lfloor \frac{r+1}{k+1} \rfloor + \lfloor \frac{l}{k+1} \rfloor$$

E. Kaguya with Oak apples

线段树单点修改板子题，你只需要同时维护区间和以及区间最值就做完了。

由于值域很小，你也可以开值域个树状数组去维护每个值域的情况。

F. Kaguya with Frogs

只有形如 $ABXBXBX \cdots BXB.$ 这样的才可能有解。

X 内可以是 . 也可以是 B

不妨设字符串的长度为 l 。

那么最少需要的 B 的个数是 $\lfloor \frac{l}{2} \rfloor$ ，最多是 $l-2$ 个。

G. Kaguya with Zero to Ichi

做这道题你需要一个前置芝士。

堆优化 $Dijkstra$ 求单源最短路

(非负权图跑 $SPFA$? $SPFA$已经死了)

(但 $SPFA$ 仍然能过这题)

如果你对它不熟悉，建议先补一下模板题：

https://www.luogu.org/problemnew/show/P4779

X 矩阵的定义很奇怪，我们可以考虑从这方面入手。

可以发现如果把X看做类似邻接矩阵，三个条件可以规约为：

- 1 节点的出度为 1
- n 节点的入度为 1
- 其他节点的出度等于入度

  其他节点的出度等于入度

那么原问题就变成了，我们从完全图当中选一些边使其满足上述三个条件，使得边权和最小。

1° 如果 1 和 n 是联通的，那么我们选出来的边一定是不包含环的简单路径。

证明：反证法。如果包含环且它是边权和最小且唯一的，那么我们显然可以去掉环上的一些边，使 1 和 n 仍然联通，且边权和不变大，矛盾。

2° 如果 1 和 n 不联通，那选出来的边使 1 和 n 各自成一个简单环(不能是自环，因为自环对答案没有贡献)。

证明与上面类似。

那么我们可以得出结论

$$ans = min(1\text{到}N\text{的最短路}，1\text{的最短简单环} + N\text{的最短简单环})$$

怎么求 1 的最短简单环？

我们只需要在 dij 的过程中，用 1 的出边更新完 dis 之后，把 $vis[1]$ 表示成未访问过，跑完的 $dis[1]$ 就是最短简单环了。

对 n 和 1 分别跑一次 dij，这道题就做完了。

H. Kaguya with QodeForces

前置芝士：单调队列

模板题 https://www.luogu.org/problemnew/show/P1886

模板题可以帮我们解决 $maxrating_i$ 。

正着跑好像解决不了 $count_i$ 。

我们发现倒着跑一遍单调队列的时候，还活在队列里的个数就表示区间内比队首元素单调小，且在队首序号前的元素个数，即 cnt_i 。

I. Kaguya with 树成生小最

前置芝士：$Kruskal$ 求 MST

模板题 https://www.luogu.org/problemnew/show/P3366

我们重新跑一遍生成树的生成过程。

设我们当前要连的边的两端分别是 a 和 b，其权值是 val，a 所在的集合为 A ，b 所在的集合为 B ，设这两个集合的大小为 cnt_A, cnt_B 。

合并 A 和 B 时，两个集合中两两构成的边一共有 $cnt_A \times cnt_B$ 条，除了我们要连的这条边以外，每条边的权值最少为 $val+1$ 。

证明：假如有某条边的权值小于(等于) val ，那么我们把要连的边换成这条，依然不影响连通性，但最小生成树就不同(不唯一)了。

那么把这些边都连上，答案就增加了 $(cnt_A + cnt_B - 1) \times (val + 1)$ 。

按这样重建完 MST ，最终的和就是答案，注意不要爆 int 就行了。

J. Kaguya with building blocks

可以线段树，但没必要。

注意到加操作是全局加，我们不需要对每个点都加，只要记录全局加了多少就行了，不妨记为 add 。

单点 set 的时候我们只需要令 $a[i] = v - add$ 以此来抵消它之前受到的全局加操作就行了。

单点 $query$ 的答案就是 $a[i] + add$。

K. Kaguya with cxk123

鸡你太美

只需要模拟一下每次在比赛的坤坤就行了~

L. Kaguya with Binary

算法：简单的进制转换

考察进制转换，把一个 01 串当成不同进制下的数，然后相加，答案不会超过 $long\ long$ 范围！