# 前言

难度分布：

Easy：A、C、G

Mid：B、H

Hard：D、I、J

AK：E、F

Unknown：D、E

Easy对应低于区域赛难度的题目，或者对应部分区域赛的第一题

Mid对应区域赛签到题

Hard对应区域赛的铜~银难度

AK对应区域赛金难度

下面重点说一下Unknown：

实际上，D的定位是一个Mid-Hard过渡的题目，但是内测时某橙名WA了13发，并经过与std的多次对拍后才AC，某金牌WA了3发跑路，红名WA了1发跑路，黑红WA了9发后AC（中间有些在测数据），另一位黑红WA了4发后AC。

E作为博弈论，橙名选手和金牌选手都顺利通过，但是黑红选手却被卡了。

关于时间和空间：

所有题目时间均为两倍及以上std时间，且std常数较大，如果TLE大概率是时间复杂度不对；

所有题目空间均已开到OJ可以支持的最大空间，并且都有四倍及以上的std空间。

# Easy

## A 烤羊

**枚举**

每种调料选或不选，共8种方案，直接手动枚举或二进制即可。

时间复杂度 $O(T)$ 。

## C 抹茶

**枚举、双指针、前缀和**

首先需要发现， $a$ 数组中只有正整数，因此肯定是选择越长的区间越好。

因此，我们可以从前往后枚举，如果第 $i$ 个位置的 $a_i + b_i$ 与第 $i-1$ 个位置的相等，我们选择的区间长度就加 1 ，否则就重新选择第 $i$ 个位置作为新区间的起点，每次选择区间后对答案取一次 max 即可，选择的区间之和可以用前缀和或者一个变量记录。

时间复杂度 $O(n)$ 。

## G 贪心

**排序、模拟、贪心、双指针**

一个贪心的思路就是每次选择可选的最大值。

我们可以分颜色对每个格子进行排序，然后先取红色格子的最大值，再取黑色格子的最大值，...，以此类推。

时间复杂度 $O(n \times logn)$ 。

# Mid

## B 英逃

**枚举、二分、前缀和、ST表、单调队列**

首先需要发现：以同一个位置作为左端点时，右端点越往右可以使得数组权值变得越小。

因此我们可以枚举每一个点作为左端点，二分找出右端点的最左位置，取最小值即可。

记枚举的左端点为 $l$ ，二分出的右端点为 $r$ ，如何check呢？

我们可以发现，在 $l$ 左边、 $r$ 右边的数组权值是不会受影响的，这两部分用前缀和、后缀和提前预处理后可以 $O(1)$ 得到。

区间 $[l, r]$ 内部都是区间中的最大值，权值为 0 ，记最大值为 $x$ ，此时我们还需要再加上 $|a_{l-1} - x|, |a_{r+1} - x|$ 。

区间最大值是一个经典的问题，使用ST表预处理后可以 $O(1)$ 得出。

还有另一个结论：最终的区间长度（答案）具有二分性。

因此我们可以二分得出区间长度，再枚举每一个点作为左端点，右端点也可以直接得出，按上述方法进行同样的check即可。

由于我们枚举的过程中可以从左到右，并且区间长度是固定的，因此这部分也可以使用单调队列处理区间最大值。

时间复杂度 $O(n \times logn)$ ，多一个log似乎过不了。

## H 天使

**数学**

实际上，任意一种接触方式最终得到的破坏值之和都相同，因此使用任意一种接触方式计算即可。

最简单的接触方式是：将第一个使徒作为中心，后续的使徒一直与这个中心进行接触。

接触方式的数量为：每次接触后使徒数量都会减少 1 ，每次在剩余的使徒中任选两个的方案数相乘，$\prod_{i=2}^{n} C_i^2$ 。

时间复杂度 $O(n)$ 。

# Hard

## D 剪切

**树形DP**

首先有一个容易想到的结论：如果两根喜欢的黄瓜在树上相邻，必然是无解的。

接下来就是比较难的DP部分，首先定义状态：

（下面说的子树实际指的都是DFS过程中考虑完第 $i$ 个节点之后的一个连通块，并不是严格意义上的子树）

$dp[i][0][0]$ 为第 $i$ 根黄瓜不被剪切，子树中没有喜欢的黄瓜的最大剩余价值。

$dp[i][0][1]$ 为第 $i$ 根黄瓜不被剪切，子树中恰好有 1 根喜欢的黄瓜的最大剩余价值。

$dp[i][1][0]$ 为第 $i$ 根黄瓜被剪切，子树中没有喜欢的黄瓜的最大剩余价值。

$dp[i][1][1]$ 为第 $i$ 根黄瓜被剪切，子树中恰好有 1 根喜欢的黄瓜的最大剩余价值。

（在剪切一根黄瓜后，连通块应当为空，因此第四种状态实际上是不合法的，不需要考虑）。

接下来考虑转移， $i$ 为枚举到的节点， $j$ 为 $i$ 的孩子，并用 $inf$ 表示负无穷：

```
如果第i根黄瓜是喜欢的:
S = sum of max(dp[j][0][0], dp[j][1][0])
dp[i][0][0] = inf
dp[i][0][1] = a[i] + S
dp[i][1][0] = inf
如果第i根黄瓜不是喜欢的:
dp[i][0][0] = a[i] + S
dp[i][0][1] = max({-max(dp[j][0][0], dp[j][1][0]) + dp[j][0][1]} | all(j)) + a[i]
+ S
dp[i][1][0] = sum of max(dp[j][0][1], dp[j][1][0])
```

最后的答案就是 $max(dp[root][0][1], dp[root][1][0])$ ，对于最终状态 $dp[root][0][0]$ 是不合法的。

时间复杂度 $O(n)$ 。

## I 键帽

**DP、计数、逆元、DP优化**

首先有一个比较暴力的做法：

定义 $dp[i][j]$ 表示长度为 $i$ ，最后 $j$ 个字母为元音的字符串种数，转移如下：

```
dp[i][0] = sum(dp[i - 1][0], ... , dp[i - 1][i - 1]) * 21
dp[i][j] = dp[i - 1][j - 1] * 5
```

然后我们可以发现从 $t$ 到 $i$ 之间若都为元音，且第 $t-1$ 个字母为辅音，则：

```
len = j - i + 1
dp[i][len] = dp[t - 1][0] * pow(5, len)
```

此时我们可以修改状态定义为：

```
dp[i][1] 表示长度为 i 且第 i 个字母为元音的合法字符串种数
dp[i][0] 表示长度为 i 且第 i 个字母为辅音的合法字符串种数
dp[i][1] = sum(dp[i - t][0] * pow(5, t)) | t in [1, k]
dp[i][0] = (dp[i - 1][0] + dp[i - 1][1]) * 21
```

现在的瓶颈在 $dp[i][1]$ 的转移上，一种优化方式是记 $f_i = dp[i][0] \times 5^{n-i+1}$ ，对 $f$ 数组求前缀和。

转移到 $i$ 时，使用 $\frac{f_i - f_{i-k-1}}{5^{n-i+1}}$ 即可，由于涉及到除法，因此需要使用逆元。

时间复杂度 $O(n)$ ，多一个log似乎也能过，验题人中还有人写了树状数组，因此将时限放小了，但是树状数组还是能过。

# J 章鱼

**计数、换根DP、容斥原理**

这题其实不需要真的使用换根DP。

考虑一棵树以 $i$ 为根，$j_1, j_2, \ldots, j_k$ 为 $i$ 的孩子，以 $t$ 为根的子树大小为 $a_t$ ，有多少对节点的LCA为 $i$ 呢？

答案是 $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ 两两相乘再加上 $n$ （ $i$ 和任意节点的LCA都为 $i$ ），这一部分用简单的方式即可优化成线性。

不以 $i$ 为根的情况下呢？有一个比较显然的结论是以 $j_t$ 子树中任意一个节点 $u$ 为根的 $f(u, i)$ 都是相等的。

尝试枚举以 $j_t$ 为根的情况， $f(j_t, i)$ 就是 $a_{j_1}, \ldots, a_{j_{t-1}}, a_{j_{t+1}}, \ldots, a_{j_k}$ 两两相乘再加上 $n - a_{j_t}$ （ $i$ 和 $j_t$ 子树外节点的LCA都为 $i$ ）。

暴力枚举显然是不行的，需要思考如何 $O(1)$ 得出 $f(j_t, i)$ ，思考一下 $f(i, i) - f(j_t, i)$ 是什么东西？（容斥原理）

实际上少掉的这部分就是 $a_{j_t}$ 与 $a_{j_1}, \ldots, a_{j_{t-1}}, a_{j_{t+1}}, \ldots, a_{j_k}$ 的乘积再加上 $a_{j_t}$ ，等价于 $a_{j_t} \times (n - a_{j_t})$ 。

现在我们可以枚举每一个 $i$ 作为根， $i$ 的孩子个数为 $k$ ，线性得到 $f(i, i)$ ，再线性得到所有的 $f(j_t, i)$ ， $n$ 个节点 $k$ 的总和就是 $2n - 2$ 整体是线性的。至于每次如何求 $a_{j_1}, a_{j_2}, \ldots, a_{j_k}$ ，此处不再赘述。

时间复杂度 $O(n)$ 。

# AK

## E 挺准

**博弈论**

一个比较显然的结论是：最终状态下， $a_1=1,a_2=2,\ldots,a_{n-1}=n-1,a_n=?$ ， $a_n$ 似乎并没有什么用。

其实我们可以直接将 $a_n$ 剔除，然后把题意转化：一条线上有许多空格，我们要移动空格。

可以发现假如移动最后一个空格，对方无法复刻我们的操作，移动倒数第二个空格同样如此 。

但是移动倒数第三,第四个空格，就可以分别通过移动最后一个，倒数第二个空格来抵消掉这个操作的影响 。

同理可以递推得到，有效的操作其实只有倒数 1,2, 5,6, 9,10... 这些空格。

实际上这是两个阶梯博弈的拼接，根据阶梯博弈的结论，取这些部分异或即可。

时间复杂度 $O(n)$ 。

# F 双子

**字符串、后缀数组、马拉车**

有一个暴力的 $O(n^4)$ 算法：

枚举 $s$ 中的两个位置 $i,k(i<k)$ ， $t$ 中的一个位置 $j$ ，使得 $s_i=t_j$ 且 $s$ 的子串 $[i+1,k]$ 是回文串，然后我们可以将 $i$ 往左、 $j$ 往右移动，直到 $s_i\neq t_j$ 时 break 掉。同理我们还要在 $t$ 中进行类似的操作。

可以发现我们不论 $k$ 取什么， $i,j$ 移动的次数都是一样的，这样就可以减少一次枚举，时间复杂度降至 $O(n^3)$ 。

枚举 $k$ 的过程实际上是判断有多少个以 $i$ 为左端点的回文串，这部分我们可以暴力枚举一个字符（或两个相邻字符）作为回文中心，往左右扩张看最远能延申到哪里，提前进行预处理，这部分时间复杂度为 $O(n^2)$ 。

枚举 $i,j$ 以及移动次数我们可以用一个DP进行预处理：如果 $s_i=t_j$ 时， $dp_{i,j}=dp_{i-1,j+1}+1$ ，这部分的时间复杂度也为 $O(n^2)$ 。

现在整体的时间复杂度就降到了 $O(n^2)$ 。

回文中心、左右扩张，这一部分其实就是一个比较典型的马拉车，然后再用马拉车得到的数组求一下差分就行了，这部分的时间复杂度就是 $O(n)$ 。DP部分可以使用后缀数组进行处理，这部分时间复杂度是 $O(n\times logn)$ 。

时间复杂度 $O(n\times logn)$ 。

# 后话

这场代码量有点小，除了最后一题几乎都不怎么需要写代码，因此前排选手都做的飞快。

而D题似乎确实到了一个非常抽象的位置，而且几乎没有一发AC的选手。

对于题目难度的判断，除了D题之外，似乎都是正确的。D题在出题、思考和造题的时候感觉都还好，不过确实也是和暴力对拍了两遍才写好，后来在验题中发现了不对劲。首先是橙名WA了13发并与std经过多次对拍后才AC，然后是金牌、红名WA完就跑，再到黑红WA9发，以及hdu验题人WA了6发，最终将这一道看似简单的题目变成了一个有一定难度的题目。正式赛中，也有不少选手非常红温，WA了很多发才过，甚至有WA了十多发还没过的选手。

rk1应该是开黑，甚至是多机位，两发不同题目的AC提交只差1min还勉强可以解释，两个不同题目的两发WA只差1min是解释不了的，并且名字本人很大概率根本没有参赛。现在我很好奇中之人会是谁，北交金牌强队似乎只有一个，但是看码风应该又可以排除某一位选手。

rk2是很容易就能盒到的出线选手，估计是本人在正常单打，码风完全一致。

rk3、rk4分别是初一、初二的杭二oi爷，翻了一下前几场的记录，其中一位基本只做了部分题目，有一场做了后6题，没做前4题，本质上相当于也是AK了。对比oidb中的数据后发现，名字和参赛的人很大概率是对不上的，不是很清楚这两位选手为什么在这场都按非常正常的顺序写完了。

rk5感觉有点预料之外，不过也是一位金牌选手。