2020牛客暑期多校训练营（第三场）

黄以文

特别致谢

- 感谢图片中的 id 们，尤其是 Q 老师。

1	waynedisonitau123: skylinebaby, edisonhello, waynetuinfor # →to practice →devirtualize	10	1081	+1 00:16	+ 00:25	+ 00:51	+ 01:24	+ 01:05	+ 00:41	+1 02:44		+ 02:53	+ 04:06		+ 02:56
2	NCTU_TaNoShiI: FISHTOBY, hank55663, samsam2310 # →to practice →devirtualize	8	1035	+1 00:05	+3 00:21	+ 00:34	+2 00:49	+1 01:02	+6 01:45	+5 02:37	+1 03:42				
3	BBQube: briansu, icube, baluteshih # →to practice →devirtualize	7	575	+ 00:15	+ 00:28	+ 00:48	+1 01:46	+1 01:16	+ 02:27	+ 01:55	-1	-1			
4	NoName: iaNTU, ltf0501, WillyPillow # →to practice →devirtualize	7	646	+2 00:27	+ 00:20	+ 00:47	+1 02:31	+1 01:28	+ 01:01	+3 01:52				-4	
5	NCTU_Revue: HanaYukii, tracyliu.cs06, marmot0814 # →to practice →devirtualize	6	707	+ 00:45	+1 00:31	+ 01:02	+ 01:56	+4 02:28	-6	+ 03:25					
6	arosusti →to practice	3	363	+ 02:38	+ 01:07	+ 02:18									
	* 300iq	8		+1	+	+	+5	+	+	+	+2				
	* tmt514	5		+2	+	+1	+5	+							
	* wangyenjen	4			+	+		+	+1						
	* tokitsukaze	2		+2	+										
	* arosusti	2					+2	+1							
	* marmot0814	1							+						
	* quailty	1										-1	+		
	* briansu	0									-1				
	* hank55663	0												-2	
	Accepted Tried			10 10	11 11	10 10	9 9	10 10	8 9	7 7	3 5	2 4	3 3	1 3	1 1

L – Problem L is the only Lovely Problem

- 题意：判断一个字符串是否为 lovely 开头，不分大小写。
- 看了下众多测试队伍的比赛结果后，觉得仍缺乏签到题，于是就补上这题了...
- 若有不会做的队伍嘛... 先去把 c 语言的基础学好再来比赛吧 0.0
  没打算在这里教语法 (>__<)

A – Clam and Fish

- **简要题意**
  - 小月有 n 单位的时间都在钓鱼，每个单位时间有四种状态，有蛤蜊/没蛤蜊，有鱼/没鱼。小月事先知道这 n 个时间点的状态。每个时间点有四种可能的动作：
    1. 若该时间点有鱼，则可以直接钓鱼。
    2. 若该时间点有蛤蜊，则可以用该蛤蜊制作一个鱼饵。
    3. 若该时间点身上有至少一个鱼饵，则可以用一个鱼饵钓一条鱼，钓完后就少了一个鱼饵。
    4. 什么事都不做。
  - 请问小月最多可以钓多少条鱼。

A – Clam and Fish

- **灵感来源**
  - 有阵子每天下班都在动物森友会里钓鱼，就想藉由这个游戏当背景出一个相对简单的题。

A – Clam and Fish

- **算法类别：贪心**
- **时间复杂度： $O(N)$**

- **主要观察：有鱼的时候就钓鱼一定是最好的策略之一**
  - 用鱼饵钓鱼会多浪费一个鱼饵。
  - 若该时间拿来制作鱼饵，顶多之后用该鱼饵掉一条鱼，还多占用未来的时间，不如现在就直接钓鱼。
  - 于是我们只需再考虑有无蛤蜊的情形作为子题。

A – Clam and Fish

- **只考虑蛤蜊的子题这里提供两种方法**
  1. 从时间早考虑到时间晚
     - 有蛤蜊就做鱼饵，没蛤蜊就就尝试用鱼饵钓鱼。最后若发现还剩 x 包鱼饵，就把制作比较晚的 x / 2 包鱼饵的时间拿来钓鱼。
  2. 从时间晚考虑到时间早
     - 若时间 i 有鱼饵，且时间 i 之后还留有没做鱼饵也没钓鱼的时间点，就在时间 i 做鱼饵，否则该时间就预留着，等着用小于 i 的时间点做的鱼饵钓鱼。

B – Classical String Problem

- **简要题意**
  - 有一个字符串，有两种操作：
    1. 询问第 x 个字符。
    2. 把最左边的 x 个字符搬到最右边或把最右边 x 个字符搬到最左边。

B – Classical String Problem

- **灵感来源**
  - 用字符串及操作来造个脑筋需要稍微转一下的代码很短的题。
  - 相信茫茫题海里是有原题的，就饶了我吧...

B – Classical String Problem

- 算法类别：数据结构 平衡树 指针维护
- 时间复杂度： $O(|S| + Q)$

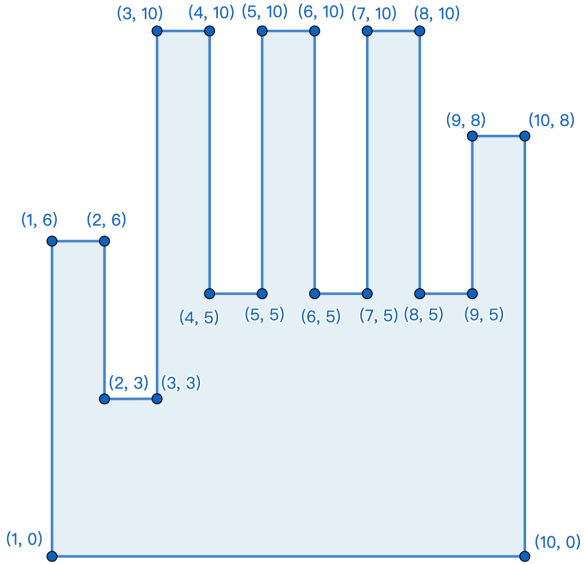
- 我有个好朋友验题时使用了 Treap 花了 50 分钟 AC 。
- 我很友好，不在简单题卡复杂的资料结构的。

B – Classical String Problem

- **参考做法：想像成字符串是头尾接在一起的 ，只需维护一个指针指向当前第一个字符的位置，就能以 O(1) 的时间复杂度进行每个操作。**
- **以 Sample 为例**
  - 初始时指针在 n 的前面 |nowcoder
  - 操作 1 询问指针右边第 1 个字符，答案是 n
  - 操作 2 把 4 个字符搬到右边，相当于把指针向右移 4 个字符 nowc|oder
  - 操作 3 询问指针右边第 6 个字符，答案是 o
  - 操作 4 把右边 3 个字符搬到右边，相当于把指针向左移 3 个字符 n|owcoder
  - …

C – Operation Love

- **简要题意**
  - 下图为某机器人右手手印的形状，左手为右手的对称图形，现在给你一个机器人手印(可能经过平移及旋转)，请判断是左手还右手(保证一定是其中一只手)。



C – Operation Love

- **灵感来源**
  - 现在科技产业很流行各种 AI 应用如人脸辨识，那我就不如搞个题来做简单的手印辨识。

C – Operation Love

- 算法类别：計算幾何 - 顺/逆时钟判断
- 时间复杂度： $O(N)$ (N 是顶点数)

- 若要做正统的两个多边形判断也可以，但由于此题保证输入只可能是题目里的左手印或右手印，于是我们可以先把多边形的点转为逆时针顺序，再判断是否有连续两个边长为 9 和 8 的 pattern 即可。
- 要注意此题输入给的是不精确的福点数，在做数值判断的时候要容许一些误差(例如两条边长度差不超过 0.1 就当作相等) 。

D – Points Constructive Problem

- **简要题意**
  - 在无限大的格子盘面中，原本所有格子都是白色的，请把其中 n 个格子涂黑，使得满足格子 x 和格子y 是上下左右相邻且 x , y 异色的 (x,y) 组数恰好为k 个((x,y)和(y,x) 视为相同)。

D – Points Constructive Problem

- **灵感来源**
  - 有个经典的数学问题是周长为x的密闭图形可能的最大面积是多少(等周定理)，而我把它改成一个方方正正的版本。

D – Points Constructive Problem

- **算法类别：数学、构造**
- **时间复杂度：** $\mathbf{O}(n+m)$

- ~~是说我有个好朋友用时间复杂度比较高的神秘 dp AC 了此题。~~

D – Points Constructive Problem

无解判断

- **首先观察到 m 是奇数时一定无解**

  考虑从某行无穷远的左端往右端通过，一定是白点->黑点->白点->黑点->白点的顺序交错，且起始和最终都一定是白点，所以每一行一定有偶数个相邻的相异色点对。对于列来说也是同要道理。

- **m > 4 * n 一定無解**

  因为每个黑点至多有 4 个相邻的白点。

- **m 的下界见下页**

D – Points Constructive Problem

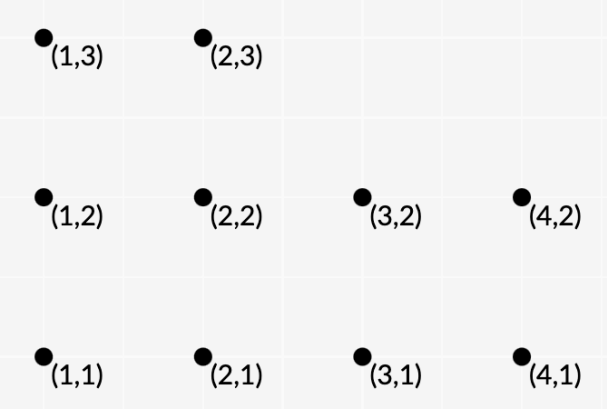
无解判断

- **m 的下界判断**
  - 假设这 n 个黑点共出现在相异的 a 行和 b 列，每行及每列都至少有 2 个异色点对，所以至少有 2(a+b) 个异色点对，且此下界能够构造出来。
  - 这 a 行 b 列的交集至多有 a x b 个黑点。
  - 所以我们要找到 a, b 满足 a x b >= n 且 2(a + b) 最小了
  - 于是暴力枚举 a, b 即可求出 m 的最小可能值
    (有个更强的性质：极值一定发生在 abs(a-b) <= 1) 。

D – Points Constructive Problem

无解判断

- **m 的下界判断**
  - 以 n = 10 为例，a = 3, b = 4 时满足 a x b = 12 ≥ 10 且 2(a+b) = 14 的值最小，故 m 最小可能值为 14。
  - n = 10, m = 14 的其中一组解如下：



D – Points Constructive Problem

解的构造

- **m 恰为下界**
  - 找出符合下界的 a, b 后，选定任意连续的 a 行 b 列，先由左而右，再由下往上依序涂成黑点即可。
- **m > 下界**
  - 当目前相邻相异色点对数仍小于 m，就把最右上角的黑点丢到远处，相邻相异色点对数 就会增加 2 or 4，一直做此操作。
  - 若全部操作完时相邻相异色点对数超过 m，必定只超过 2，此时只要移动某个四周都是白点的黑点到恰与另一个黑点相邻的位置即可。

E - Two matchings

- **简化过的题意：**
  - 给 2n 个点的完全图，每个点有个数值，每条边的权重是相连两个数值的差的绝对值，请找出两个边没有交集的完全匹配使得这两个完全匹配的所有边的权重和最小。

E - Two matchings

- **灵感来源：**
  - 想要出个在数线上的一些点连来连去的问题，就编出了这个。

- ~~网上哪里有个原题也不奇怪。~~

E - Two matchings

- 算法类别：图论、贪心、动态规划
- 时间复杂度： $O(n)$

E - Two matchings

- 观察一：此题等价于找一些长度为偶数的环使得这些环恰通过每个点一次，且所有边的总权重最少。
- 观察二： 2k 个点所构成的环的权重和的最小值为最大的权重减最小的权重。
- 观察三：先把所有点的按照权重排序，最佳解一定是出现在每个环都是由排序后连续的点构成。
- 观察四：若某个环长度 ≥ 8，总是可以拆成一些长度为 4 或 6 的环且总边权更小。
- 于是我们就可以把点的权重排序后， 对于前 2k 个点当作子题来动态规划啦，转移时只要枚举最后一个点所在的环长度是 4 还是 6 即可。

F - Fraction Constructive Problem

- **简要题意**
  - 给定一个分数 $\frac{a}{b}$ $(1 \le a,b \le 2\times10^6)$ 请构造出两个分数 $\frac{c}{d}$, $\frac{e}{f}$ 满足 $\frac{c}{d}-\frac{e}{f}=\frac{a}{b}$ 且 $d,f<b,c,e \le 4\times10^{12}$。

F - Fraction Constructive Problem

- **灵感来源：**
  - 想要出一个和分数有关的构造题就变出了这题了。

F - Fraction Constructive Problem

- 算法类别：数学、扩展欧几里得算法
- 时间复杂度：预处理: $O(V)$ ，每个询问 $O(log\ V)$ (V 是 a,b 的上界)

F - Fraction Constructive Problem

- **状况 1：a 和 b 不互质**
  - 令 g 为 a 和 b 其中一个大于 1 的公因数，$\frac{a}{g}+1, \frac{b}{g}, 1, \frac{b}{g}$ 就是答案。
- **状况 2： b 的相异质因数不超过 1 个**
  - 无解 (为什么呢?下页再解释) 。
- **状况 3：b 的相异质因数个数超过 1 个**
  - 先找到 d, f， 满足 $d \times f = b$ 且 d, f 互质。
  - $\frac{c}{d}-\frac{e}{f}=\frac{a}{b}$ 通分后可推得 $\frac{c\times f-e\times d}{d\times f}=\frac{a}{b}$，，也就是要找到 c 和 e 满足 $c\times f - e\times d = a$。
  - 这不就是扩展欧几里得算法在解的问题吗？

F - Fraction Constructive Problem

- **Codeforces 上差不多感覺的構造題：1366D**
- **a, b 互质且 b 的相异质因数不超过 1 个时无解的证明**
  - 假设 $b = P^K$，由于 a, b 互质，所以 $\frac{c}{d} - \frac{e}{f}$ 化为最简分数后必须是 $\frac{a}{b}$，但 d 和 f 的质因数分解中 p 的指数都小于 k，得到矛盾。

G – Operation on a Graph

- **简要题意**
  - 给一个 n 个点的 Graph，第 i 个点一刚开始是第 I 种颜色，接着有 k 次操作，第 i 次操作有个参数 o_i 代表颜色 o_i 会侵略所有和自己相邻的颜色，于是所有和 o_i 相邻的颜色全都变成 o_i (若已没有颜色o_i 已被侵略，则该次操作无效)，求最终每个点的颜色。

G – Operation on a Graph

- **灵感来源**
  - 过去已经出过各种在格子上各种涂颜色的题(例如 Codeforces 1329A)，那这一次就出在 Graph 上涂颜色的题吧。

G – Operation on a Graph

- 算法类别：数据结构(并查集、链表)
- 时间复杂度：$\mathbf{O}(\mathbf{N}\alpha(N)+M)$

G – Operation on a Graph

- **重要观察：在所有操作过程中，对于每个点，至多只会有一次把相邻的点和自己变为同一种颜色的操作，经过该次操作后，就永远和相邻的点同色了。**
- **参考做法：**
  1. 对于每个颜色都维护一个点的链表，储存该颜色中尚未把所有相邻点变为自己颜色的点。
  2. 用并查集纪录每个点属于哪个颜色。
  3. 每次操作就会把颜色 o_i 的链表中所有相邻的点所属的颜色都变为 o_i ，就直接把对应的链表合并以及在并查集上做对应的操作。

H – Sort the Strings Revision

- **简要题意**
  - 有一个长度为 n 的字符串s_0, 第 i 个字符是数字 i mod 10， 给定一个 0~n-1 的排列 p_i 和 序列 d_i，接着还有 n 个字串 s_1~s_n, s_i 是把 s_{i-1} 的第 p_i 个字符改成 d_i 的结果。请把这 n+1 个字符串按字典序排序后，输出每个字符串在排序后的位置。 ($n \le 10^7$，会给定随机数据生成器的种子产生读入)

H – Sort the Strings Revision

- **灵感来源**
  - 自己曾经出过一个有关字符串排序且自以为很厉害的题，这次原本想出个类似题，可是不知怎么搞的就变成这个样子了(~~反正都是和字符串以及排序有关www~~)
  - 大家会不会困惑为什么题目名称还有个单词 Revision 呢？这是因为原本这套题里还有个版本是 p_i 并没有规定是 permutation，但是 300iq 验题后发现该版本和 csacdemy 上的某个题作法一模一样，就忍痛割爱了。

H – Sort the Strings Revision

- 算法类别：栈、笛卡尔树、动态规划
- 时间复杂度：$O(N)$

H – Sort the Strings Revision

- **参考做法：**
  - 令 s_i 排序后的位置为 pos_i，方便起见，这里都假设 d_i != p_i mod 10
  - 先考虑所有字串的第一个字符，不妨假设是 p_k = 0， 那么 s_0~s_k 都是 0，而 s_{k+1}~s_n 都是 d_k (假设 d_k 不为 0) 。于是我们就知道s_0~s_k 和 s_{k+1}~s_n 之间的大小关系。于是我们就可以把所有字符串分成 s_0~s_k 和 s_{k+1}~s_n 两组，把两组都视为子问题递归求出答案，由於 0 < d_k, pos_0~pos_k 的值會和 s_0~s_k 的子问题一样，pos_{k+1}~pos_n 的值會是 s_0~s_k 的子问题答案中每个值都加上k+1。
  - 如果有某些行满足 d_i = p_i mod 10，直接忽略那些行即可。

H – Sort the Strings Revision

- **参考做法：**
  - 递归的过程中，每次对pos 数组的操作都是把连续某段加上某个值，且我们并不需要在过程中知道某个子题的实际答案，所以就可以用连续和的 dp 技巧以 O(n) 的时间复杂度维护答案！
  - 但递归时，我们还需要在每个子问题中(假设是 s_l~s_r在做排序)，找到 k 满足 p_k 是 p_l~p_r 间值最小的数。
  - 哎，好像有个东西叫做笛卡尔树就是在解这个问题。
  -

I – Sorting the Array

- **简要题意**
  - 有一个长度为 n，0~n-1 的排列 b[0..n-1]，给定一个正整数 m, 对于 i 从 0~n-m 依序执行 sort(b,b+m)，给你执行完的结果，并且再给你一个正整数 k，请问数组 b 的所有可能中，字典序第 k 小的是哪个。

I – Sorting the Array

- **灵感来源**
  - 过去也出过给定一段和 sort 有关的代码然后随意地问一些事情的问题(如右)，这次就依样画葫芦变出了这题。

由於題目敘述什麼的向來都不是重點，大家直接看 dreamoon 原本寫的 TLE 程式碼來猜測題意，寫出能 AC 該題的程式吧！

```
#include<bits/stdc++.h>
using namespace std;
set<vector<int>>arrays;
void add_partial_sorted_array(vector<int> array,int i,int j){
    sort(array.begin()+i,array.begin()+j);
    arrays.insert(array);
}
int main(){
    int N;
    vector<int>input;
    cin>>N;
    for(int i=0;i<N;i++){
        int x;
        cin>>x;
        input.push_back(x);
    }
    for(int i=0;i<N;i++)
        for(int j=i+1;j<=N;j++)add_partial_sorted_array(input,i,j);
    printf("%d\n",(int)arrays.size());
    return 0;
}
```

I – Sorting the Array

- 算法类别：数学、数据结构
- 时间复杂度：$O(log\ k \sum n)$

I – Sorting the Array

- **各种观察**
  - 根据 sort 的性质，我们可以知道，ret[i] 是是 b[i.. min(i+m-1,n)] 中扣除 ret[0] ~ ret[i-1] 中最小的数。
  - 对于一个 j，若存在某个 ret[i] > ret[j] 且 i < j，那么 b[j+m-1] 一定是 ret[j]
  - 扣除上一个观察中满足条件的所有 ret[j] 后，假设只剩下 x 个数字，我们要解的问题相当于输入的 m 和 k 值不变，n 改成 x，而 ret = [0,1, ...,x-1]。

I – Sorting the Array

- **现在我们就当作 input 中的 ret 数组中 ret[i] = I**
- **先考虑一个相对简单的问题：求出数组 b 的可能个数**
- **再一个观察，按照以下算法可以产生出所有可能的数组 b：**
  - 依序从 0 至 n - 1 枚举 i ，在 b[0] ~ b[min(n-1,i+m-1)] 中，选一个尚未给定数值的数设为 i。
- **根据上述算法，决定 i 的在数组 b 中的位置的时候恰有 min(m,n-i) 个位置可以选择，所以可能的数组 b 的个数就是 $\prod_{i=0}^{n-1} min(m, n-i)$。**
- **又一个观察：我们可以找到一个尽可能大的数 v，满足 $\prod_{i=v}^{n-1} min(m, n-i) \geq k$，则字典序第 k 小的 b 中，对于所有 i < v 必定有 b[i] = i。**

I – Sorting the Array

- 最后数组 b 只剩下至多 min(n, log k) = s 个位置还不确定数值，我们可以对每个位置枚举要放那个数字，并按照类似上页所列的排组方式以 O(s) 的时间计算该位置摆该数字时 b 数组含有多少种可能性，有O(s) 个位置，每个位置至多枚举 O(s) 个数，每次花 O(s) 的时间计算，于是我们就得到一个单笔数据 $O(s^3)$ 的方法。而所有数据实际是时间复杂度分析出来会是$O(n \log^2 k)$。
- 若常数写的不高，此方法已经能够通过此题。
- 实际上，仔细分析排列组合公式就会发现，枚举一个位置所有可能摆放的数时，从一个数换成另外一个数，可以只用 O(1) 的时间复杂度算出答案。所以此题可以做到 O(n log k)。

J – Operating on a Tree

- **简要题意**
  - 给一个 n 个点的 tree 定义一个 permutation 的 cost 为，把该 permutation 当作 Graph 这题的 a 序列，有多少次侵略行为中 tree 上至少有一个是第 o_i 种颜色。请计算所有 0 ~ n-1 的排列的 cost 总和 mod 998244353。

J – Operating on a Tree

- **灵感来源**
  - 就如题目叙述里所说，灵感是来自 G 题，其实原本想了各种 G 题的推广版本，例如给定 output 请构造一组 input 之类的，但改来改去，只有这个版本想的到做法...

J – Operating on a Tree

- 算法类别：树型动态规划，各种动态规划技巧
- 时间复杂度：$\mathbf{O}(N^2)$

J – Operating on a Tree

- 定义一个点 x 比点 y 当且仅当 permutation 中 x 在 y 的前面
- 对每个点我们再多一个属性：好点/坏点。
- 好点就是在一个排列中会对答案贡献 1 的点，没有贡献的点则是坏点。
- 好点和坏点有一下一些性质，且满足这些点的属性组合就是合法的：
  - 不存在两个相邻好点
  - 每个坏点至少和一个比他大的好点相邻

J – Operating on a Tree

- **相信不少人能有这是个树形 dp 题的直觉。于是我们先试着设计所需的各种 dp 状态如下：**
  - dp_num[子树树根编号][子树里有多少点比树根大]
    [树根是好点/坏点但尚未有比它大的好点相邻/坏点已有比它大的好点相邻]
  - dp_sum[子树树根编号][子树里有多少点比树根大]
    [树根是好点/坏点但尚未有比它大的好点相邻/坏点已有比它大的好点相邻]
  - num 是储存组合个数, sum 是储存所有组合的 cost 和

J – Operating on a Tree

- 剩下的就是辛苦的把转移式列出来了...

- 转移式中可能要用到连续和的技巧。
- 这题也用到乍看是 O(n^3) 实质为 O(n^2) 的某常见树形 dp 原理。
- 转移式这里就不多说了... 照念会像是在念咒文一样...
- 我自己在列 dp 式時就是枚舉一個子樹接在另一個子樹下時，想想看 [树根是好点/坏点但尚未有比它大的好点相邻/坏点已有比它大的好点相邻] x [树根是好点/坏点但尚未有比它大的好点相邻/坏点已有比它大的好点相邻] 共九总组合各会发生什么事，列完就没了， 就是个脑力苦工活。

K – Eleven

- **简要题意**
  - 给一个长度为 n 由数字和问号 '?' 组成的字串， Alice 和 Bob 玩游戏，如果问号的数目是奇数， Alice 是先手，否则 Bob 是先手，每人轮流把某个问号填上某个数字，若最终全部问号填完后，把字串视为可能有前导零的数字 x，若x 是 11 的倍数，此游戏的最终 score 为 x，否则为 -x， Alice 希望最终的 score 尽可能地大，而 Bob 则希望 score 尽可能的小，请问在两个玩家都绝顶聪明的情况下，最终游戏的 score 是多少。

K – Eleven

- **灵感来源**
  - 原本这题只是要问答案是不是非负，但后来整理完要出的题单后，觉得都没有难题，于是就试着把其中几题变难，J, K 两题都是这样的产物 ~

K – Eleven

- 算法类别：博弈论、数学
- 时间复杂度：$O(N)$

K – Eleven

- **先考虑判断答案是否非负的版本**
- **大家可能知道，判断一个奇数是否为 11 的倍数有一个方法是：把奇数位置所有数的和减去偶数位置所有数的和是否为 11 的倍数。**
  - 所以把奇数的位置填 0~9 对 mod 11 的余数的贡献就是 -1 + x，其中 x 的范围是 1 ~ 10
  - 而把偶数的位置填 9~0 对 mod 11 的余数的贡献就是 -10 + x，其中 x 的范围也是 1 ~ 10
  - 所以可以把问题转换为：给定一个初值 x，两个人轮留把此数增加一个范围介于 1 至 10 的整数，共 q 轮，若末家仍把最终数字变为 11 的倍数，就是末家赢，否则末家输。
- **这其实就是抢三十游戏的另一种版本！**

K – Eleven

- 上頁問題的結論會是若初值是 11 的倍數，後手必勝，否則先手比輸，因為對於兩人來說，都是若能讓完成該輪後數值變為 11 的倍數就必勝。

- 若改回问原题呢？原问题的做法就请见直拨中讲解吧～

Thanks