# 1、捉迷藏（game.cpp）解题报告

【本体考察知识点】

LCA，思维。

【前置知识】

可以用 LCA 树上求两点之间的距离，下面给出公式：

$$dis_{a,b}=depth_a+depth_b-2\times depth_{lca}$$

其中，$depth_x$ 表示结点 x 的深度，lca 表示 a、b 两点的最近公共祖先。

【算法分析】

先给结论：谁步长大谁就获胜，步长一样就平局。

证明：步长大的那一个总能移动一定距离，使自己可以抓到对方而对方抓不到自己。最后可以把对方逼到一个死角然后抓住。

特殊情况：先手有可能第一步就抓到对方，这时输出 Zayin。

这里判断先手第一步是否能抓到对方用倍增 LCA 实现。

不要用 memset 清空，上次用多少就清多少，否则会超时。

时间复杂度为：$O(T(n\log_2 n+q\log_2 n))$，卡一卡常也能通过本题。

# 2、读书解题（book.cpp）解题报告

【本体考察知识点】

线段树（或分块）。

【算法分析】

可以发现，每次优先取 $a_i$ 较为小的章节读，是我们的主要贪心思路。

然后就有了思路：

1. 找 $a_i$ 较小的章节
2. 把他读了
3. 不断重复以上步骤

由题可得，所花费的天数顶多是 $n$ 天，超出了这个天数就不可能读完。

然后以这个思路我们就可以打出一个优秀的 $O(n^2)$暴力，得到 $40pts$ 的好成绩。

然后考虑优化一下,哪里能优化呢？

- 找 $a_i$ 较小的章节

每次找较小之后都需要修改，然后接着找 …… ，这不就是线段树吗？

通过线段树我们可以把找最小值这一步优化成 $O(\log n)$，总的复杂度为 $O(n\log n)$。

但是最小值可能很多，我们需要把线段树 update 部分修改一下。用一个变量 $cnt$ 记录读了多少书，在 update 递归的过程中，如果该区间的最小值小于 $cnt$ ，那就递归进去将里面所有 $a_i$ 小于 $cnt$ 的章节读了，修改为 INF。

最后，如果整棵树的最小值就是 INF，说明读完了，输出答案即可。

# 3、正方形扩展（sqrex.cpp）解题报告

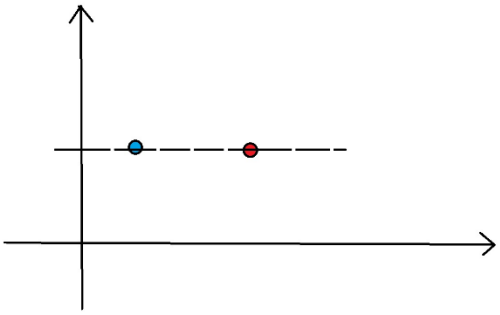
【本题考察知识点】

思维分析。如果用树套树或者权值线段树实现，会被卡掉一部分分！

【算法分析】

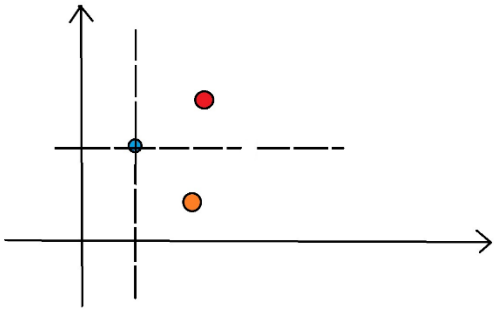
考虑一个点怎样会被围住而无法扩展，显然是四面都被阻挡了。接着可以探究一下一个点在一个方向什么会被阻挡，有以下两种情况：

1．被一个点挡住，当且仅当这两点平行于横轴或者纵轴。



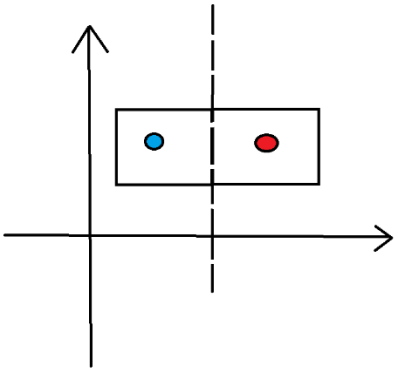
如图，红点的左侧被蓝点挡住，蓝点的右侧被红点挡住。

2．被两个点一起卡住，当且仅当这两点在被堵住点的一个方向的两端，且三点不共线。

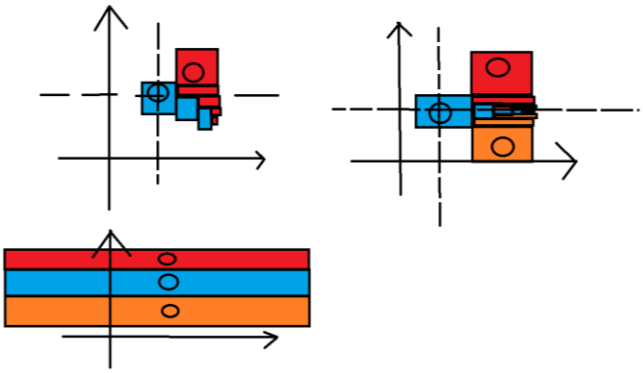


如图，蓝点的右侧被红点和橙点卡住了，其中蓝点的右上角有红点，右下角有橙点。

首先情况 1 很容易证，这两点所扩展的区域相交时，由于时间相同，所以相交时所接触的面也相同，所以它们会互相卡住，如下图的蓝点和红点互相挡住。



情况 2 比较麻烦，有很多细节。我们可以发现，任意一个点都会把另一个点的扩展向一个方向压缩，如左上图，所以当两个点把另一个点往相反的方向压缩，最后使被压缩的点的扩展空间越来越小，最后无限趋近于 0，如右上图，还有一种特殊情况是三点共线，此时中间的那个点无法被有效地在我们理想的方向压缩，如左下图。



由此，我们就知道了如何判断一个点是否能无限扩展。

我们可以直接排序和通过维护最值来判断一个区域是否有点，判断一个点的四边是否被挡住，具体实现见代码。

其实也可以用树套树或者权值线段树实现，但是会被卡掉一部分分。

# 4、EX 中缀表达式（exmid.cpp）解题报告

【本题考察知识点】

本题是一道难度较大的暴力大模拟，细节非常多。

【前置知识】

后缀表达式（逆波兰式）、表达式树、扩展欧拉定理。

【赛场插曲】

先说一下赛场上一般容易出现的情况：那时选手非常智慧，看到这道题一眼模拟直接无脑开始狂打。像这种表达式求值的题很明显的做法是将中缀表达式转成后缀表达式，然后进行求值。因为打熟悉了中缀表达式转成后缀表达式，所以很快打完，并且 Error 判断的也比较好，最后一直过不了样例，甚至叫来老师问是不是样例出错了（因为样例有次方运算，选手又把次方给进行了模运算，导致直接暴力算都是错的），最后也是不明不白地离开了考场。出来和同学讨论以后才发现次方运算的次方不可以直接模。结果还怀疑人家题出错了。最后居然还能拿没有次方运算的 16 分。加上 Error 的 12 分（有两个没有判断到）。

【算法分析】

**一、判断不合法的 Error 情况：**

首先判断不合法的中缀表达式有以下几种：

- 出现了不是数字、+、*、^、(或)的字符；
- 括号不匹配；
- 运算符没有数字在前表示优先级或者数字不在 1 到 $n$ 的范围内；
- 两个括号之间为空；
- 每个字符没有对应运算的两个数字。

**二、将中缀表达式转成后缀表达式**

个人认为这是比较简单的一步，只要你学过中缀表达式转成后缀表达式。在这里还是讲一下步骤吧：

我们用一个字符栈来维护这个过程：

- 如果遇到数字，直接加入后缀表达式。
- 如果遇到字符，先判断是否有优先级更高的字符在栈中，如果高就弹出并加入后缀表达式，直到遇到优先级比它低的符号。如果在弹的过程中遇到左括号，就不再弹出；
- 如果遇到左括号，直接加入字符栈；
- 如果遇到右括号，一直弹运算符直到左括号，同时也弹掉左括号；
- 那怎么处理同优先级的左右运算呢？题目中确实有这样的规定。左结合就是从左往右运算，左边先加入的优先级更高，右结合代表从右往左运算，右边的优先级更高；
- 如果字符栈里还有剩余就全部弹出即可。

这时我们就得到了一个后缀表达式。

**三、将次方取模时的处理（乘方运算在取模意义下的取值）**

对于次方来说确实不好办的，不可以直接取模，所以这应该是最难的一个点了，我们需要用到一个数学推论：扩展欧拉定理。

这里给出式子：

$$a^b \equiv \begin{cases} a^{(b \bmod \varphi(p))}, & \gcd(a,p)=1 \\ a^b, & \gcd(a,p)\neq 1, b<\varphi(p) \\ a^{(b \bmod \varphi(p)+\varphi(p))}, & \gcd(a,p)\neq 1, b\geq \varphi(p) \end{cases}$$

我们发现不能仅仅靠后缀表达式，我们需要建立一颗后缀表达式树，因为我们需要先算出 $a$，然后根据 $a$ 是否与 $p$ 互质分类（$p$ 指的是当前的模数），然后再递归 $b$ 改变它的模数为 $\varphi(p)$。对于每个数字或字符直接动态开点记录左右儿子即可。

接下来的操作对于加法和乘法是比较简单的，我们直接将它们取模返回值即可。

当我们遇到次方运算时，我们需要计算 $a^b$ ，$a$ 是很容易得到的，直接递归左边的子树（$a$ 的子树下的值是可以直接 $\bmod p$ 的），得到 $a$ 的值以后我们看 $\gcd(a,p)$ 是否为 1（是否互质）。

如果互质也就是第一种情况就非常简单，求得 $\varphi(p)$ 后把它当成 $b$ 的子树的模数下传得到之后返回快速幂计算即可。

发现不互质的情况还是很麻烦，需要判断 $b$ 与 $\varphi(p)$ 的大小。但 $b$ 下到子树下可能还会多次改变模数从而改变 $b$ 原来有的值，所以我们为了判断 $b$ 的大小再预处理一遍整颗后缀表达式树，算出 $\varphi(p)$ 后每次在 $b$ 运算时及时判断它什么时候超过了 $\varphi(p)$ ，超过了我们记录一个数组 $q$ 的值为 -1 ，没超过就直接记录 $b$ 的值，这样就可以对 $b$ 的大小分类，如果是 -1 或者当前的 $q$ 值比当前的 $\varphi(p)$ 要大表示应该是上述第三种情况，否则已经得到了 $b$ 的值（在 $q$ 里面）直接运算即可。

这样就能发现获得了 $84pts$ 。

最后发现输入的数字还有可能是一个很多位数的数字，必须用高精度来存储，只用在刚开始的数字中计算即可（因为上传的时候会模数）。**建议高精度用结构体存储，并只在表达式树下层叶子节点数字计算高精度时使用，上层取模后可以直接存储。**

代码有点小多，要坚持下去才能打败这种大模拟题！