GERALD08 题解

by 吉如一

September 23, 2015

1 题目大意

有一棵 n 个节点的树，树上每一条边的颜色都为红色或者蓝色。现在两个人轮流进行操作，第一个人每次选择一条红色边删除，第二个人每次选择一条蓝色边删除，删除后和树根（1 号点）不连通的部分将被删除，若干轮之后不能操作的人算输。

如果两个人都使用最优策略，问第一个人先手时、第二个人先手时分别是谁赢得游戏。

数据范围 $n \leq 10^5$

2 局面的权值

这是一个博弈问题，可以考虑类比传统的 NIM 游戏来给一个局面定义一个权值。直观上来说，这个权值应该要反应出当前局面是对谁有利，且有利的“程度”是多少。所以我们可以使用权值的正负号来表示对谁有利，权值的绝对值来表示有利的程度。

2.1 最基本的情况

首先先考虑一些最简单的情况。

第一种情况是一个孤立点，这时局面对两个人来说是完全相同的，所以我们定义这个局面的权值是 0。

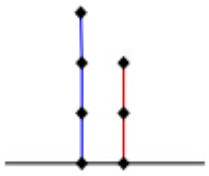
第二种情况是只有两个点且它们之间由一条红边连接，那么这个局面对第一个人有利，所以我们定义这个局面的权值为 $+1$。稍作拓展，我们可以发现如果局面是一棵 $n+1$ 个节点的树，且树上的边都是红边，那么第一个人可以连续进行 n 次删除操作，所以直观上来讲这个局面的有利程度应该是一条红边的 n 倍，所以定义这样的局面的权值是 $+n$

第三种情况是只有蓝边，可以类比第二种情况，可以得到一棵由 $n+1$ 个节点的全是蓝边的树的权值是 $-n$

最后，对于两个完全独立的游戏（即两棵根不同的树），局面的权值应当是两个独立游戏的权值和，因为对这两个游戏来说有利的程度是完全独立的。

2.2 较为复杂的情况

在知道了最基本的情况的权值之后，我们需要考虑如何计算一个较为复杂的局面的权值，考虑这个例子：



由 2.1 的讨论，我们已经知道了这个局面的权值是 -1。现在和 NIM 游戏类似地，我们考虑这个局面的每一种子局面。

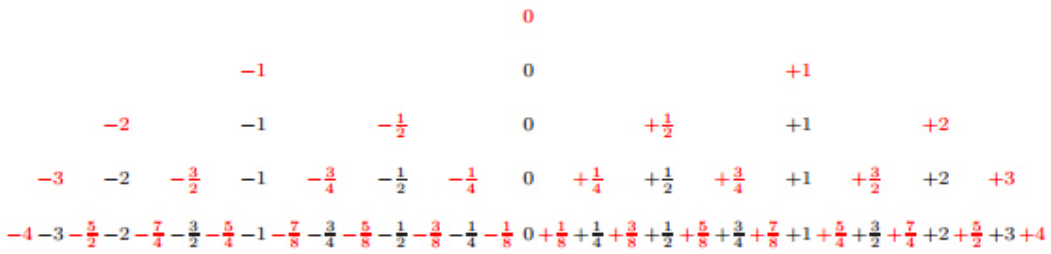
在第一个人操作后，局面的权值变成了 -2 或者 -3，而在第二个人操作之后局面的权值变成了 0 或 1 或 2。可以发现，对于一个局面，当第一个人操作后，权值一定变小，而第二个操作之后，权值一定变大。而双方的最优策略一定是要让操作后的局面对自己的不利程度最小，所以第一个人一定会选择权值最大的后继状态（-2），而第二个人一定会选择权值最小的后继状态（0）。

因此，我们定义一种计算局面权值的运算：

设当前局面的权值为 V，在第一个人操作之后局面权值的最大值为 R，第二个人操作之后局面权值的最小值为 B，那么有 $V = \{R|B\}$，在这个例子中，这个运算就是 $-1 = \{-2|0\}$

2.3 权值的运算

对于这个博弈问题来说，我们可以发现下面这个过程非常有用。假设最开始有一个什么都没有的区间，即 $(-\infty, \infty)$，接着每一次我们选出一个区间的中点，然后把区间分化成两个部分重复下去。这样我们可以得到一个树形结构：



如果我们一直把这个过程进行下去，那么就可以得到所以分母为 2 的幂次的实数。（而在这个游戏中，局面的权值只可能为这些数）

我们定义两个数 $a, b (a < b$ 且 a, b 都出现在这个树形结构中) 之间最简单的数为大小介于这两个数之间且在这棵树中最早出现的那个数（即这两个数在树上的 LCA）。例如 $+\frac{1}{8}$ 和 $\frac{3}{4}$ 之间最简单的数为 $\frac{1}{2}$。而我们在 2.2 中使用的运算就是这个。

接下来是一些这个运算的例子（左侧的空表示负无穷大，右侧的空表示正无穷大）：

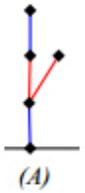
$0 = \{|\} = \{-17/64|47/128\} = \{-1/1024|1000000\}$

$1 = \{0|\} = \{1/2|9/8\} = \{1/16|1000000\}$

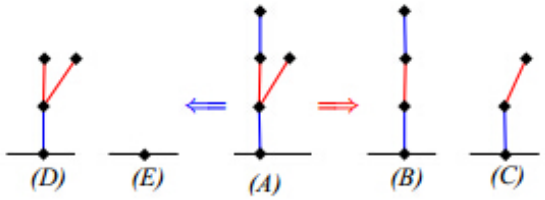
$1/8 = \{0|1/4\} = \{1/64|3/8\}$

2.4 一个例子

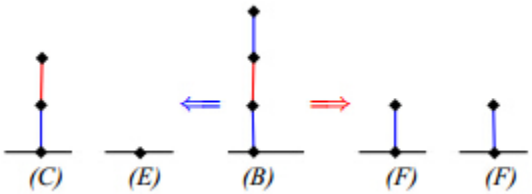
考虑如何计算下列局面的权值：



首先我们枚举所有可能的后继情况：



接着我们要计算所有后继情况的权值，拿局面 B 来举例：



局面 C 的后继只有孤立点和单独的一条蓝边，所以局面 C 的权值等于 $\{-1|0\} = -1/2$

所以对局面 B 来说，第一个人选择的一定是局面 F（-1），第二个人一定选择局面 C 和 E 中权值较小的那个，即 C(-1/2)，所以局面 B 的权值等于 $\{-1|-1/2\} = -3/4$

同理我们可以得到局面 D 的权值是 $-1/4$，所以局面 A 的权值就是 $\{-1/2|-1/4\} = -3/8$

2.5 权值的意义

如果我们得到了一个局面的权值 V，那么就可以得到这个局面的胜负结果：

1.$V > 0$，那么无论谁先手都是第一个人赢。

2.$V < 0$，那么无论谁先手都是第二个人赢。

3.$V = 0$，那么无论谁先手都是后手的人赢。

证明其实很简单，从权值计算式就可以得到。

当权值 $V > 0$ 时，由计算式的定义，无论双方怎么操作，局面的权值一定是非负，且局面权值为 0 只可能出现在第一个人操作结束后。所以到 zh 终止局面时，只可能由两种可能：第一个人操作完后只剩下孤立点，蓝边已经删完而红边还有剩余。无论哪一种情况都是第一个人赢。

同理我们可以得到 $V<0$ 时的情况。

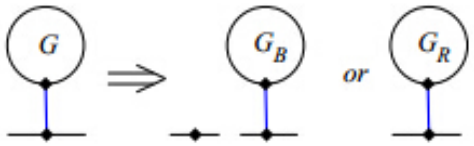
而对于 $V=0$，只有两种可能：1. 为孤立节点，这时必然是先手的输。2. 存在可行的操作，那么如果第一个人先手，权值就会变成负数，如果第二个人先手，权值就会变成正数，这样就可以规约到 $V\neq 0$ 的情况。

3 快速求得权值

现在我们已经知道了如何求一个局面的权值，但是显然这个方法的复杂度对这道题来说是远远达不到要求的。可以发现上述分析并没有使用到局面为一棵树的性质，我们可以从这个方面着手，来寻求一种快速的方法来求得局面的权值。

可以发现对于根的每一个孩子来说，游戏都是完全独立的，所以我们可以把问题简化成根只有一个孩子的情况。

我们接下来的做法基于以下的过程：



即第二个人的操作可能是删除连接根的边或者删除孩子所在子树中的边，而第一个人的操作只可能删除孩子所在子树中的边。这样就有一种把问题简化到孩子所在子树中的可能性。

下面给出一个结论：

设 G 的权值为 x，当根和 G 连接的边为红边时，令 p 为最小的正整数使得 $x+p>1$，那么当前局面的权值为 $\frac{x+p}{2^p-1}$；当根和 G 连接的边为蓝边时，令 p 为最小的正整数使得 $x-p<-1$，那么当前局面的权值为 $\frac{x-p}{2^p-1}$

接下来我们只对连接边为红色的情况进行讨论，因为另外一种情况是类似的。

对于第一个人来说，他有两种选择，第一种选择是删除连接着根和 G 的边，第二种是删除 G 中的红边。如果他选择了第一种，那么局面讲变成一个孤立点，即权值为 0。所以如果他选择了第二种删法，那么他肯定会选择游戏 G 中的最优删法，而且删除之后局面的权值将大于 0。和 2.5 中的证明类似地，可以发现如果他一开始选择了第二种删法，那么在接下来游戏的过程中，局面的权值都不会小于 0，所以在 G 中的红边删完之前，他都不会选择删除连接边。而对于第二个人来说，他的最优删法只可能是游戏 G 中的最优删法。

假设 G 的权值为 x，且 x 是由 $\{x_R|x_B\}$ 得到的，现在我们要寻找游戏 G 和当前局面（G 加上一条连接 G 的根和当前根的红边）的权值之间的联系，我们把当前局面的权值即为 $f(x)$。值得注意的是，两个权值相同的游戏是完全等效的，所以在讨论过程中，对于一个权值 x，我们可以挑选权值为 x 的一个特殊的局面来代表它，这可以降低分析的难度。

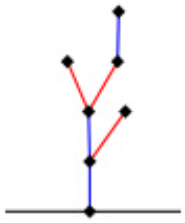
首先当 $x\geq 0$ 的时候，我们选择只包含红边的局面来代表 G，那么这时，显然有 $f(x)=x+1$。

当 $x<0$ 的时候，我们可以讨论一些特殊情况，第一种是 $x=-1$，此时可以计算得 $f(x)=\frac{1}{2}$，接着是 $x=-2$，此时可以计算得 $f(x)=\frac{1}{4}$，以此类推，我们可以得到一张表格：

$x=$	-5	-4	-3	-2	-1	0	1	2	3	4	5
$f(x)=$	$\frac{1}{32}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	1	2	3	4	5	6

由此，我们可以得到 x 为整数时的所有函数值。看起来当 x 为负数的时候，$f(x)=2^x$，其余情况 $f(x)=x+1$。但是显然这是无法应用到实数情况下的，因为当 x 为负实数时，$f(x)$ 就有可能变成无理数，而用最初的计算方法不可能得到无理数的答案。所以对于实数来说我们需要重新尝试一些值，才能得到正确的结论。在进行一些数值的打表之后可以发现所有满足 $i \le x \le i+1(i$ 为整数 $)$ 的点 $(x, f(x))$ 连成了一条线段（即一次函数）。

综上，通过这种方法，我们就能找到 x 和 $f(x)$ 之间的函数关系，应用这种方法，就能以一种较快的速度计算一个树形局面的权值。接下来举一个计算的例子：



首先对于所有叶子节点，它的权值为 0。而将一个叶子节点连上一条红边的权值为 +1，连上一条蓝边的权值为 -1。

接着对于那个连向两个孩子的边都是红边的点，其中一个孩子的贡献是 +1，而另一个孩子的贡献是 $\frac{-1+3}{2^2}=+\frac{1}{2}$，又因为这两个孩子是完全独立的，所以这一个点的权值就是 $+\frac{3}{2}$

接着对于根节点的孩子，红边的孩子的贡献是 +1，而另一个孩子的贡献是 $\frac{\frac{3}{2}-3}{2^2}=-\frac{3}{8}$，所以这一个点的权值就是 $+\frac{5}{8}$

由此，我们可以得到这个局面的权值为 $\frac{\frac{5}{8}-2}{2}=-\frac{11}{16}$

4 代码实现

既然已经知道了求得函数的方法，那么接下来就肯定要用代码来实现它了。但是在实现之前，还有一个非常重要的问题，那就是权值的位数可能很多，即权值可能需要被精确到 2^{-n}，显然不能使用 double 或者 long double 来存储。

4.1 高精度

对于每一个权值，我们可以记录一个数组 A 以及两个整数 I 和 w，且保证数组中的每一个数都小于 $-w$，假设数组 A 的长度为 m，那么表示这一个数为 $I+\sum_{i=1}^{m} 2^{w+A_i}$。这样就可以精确地保存每一个数并进行运算，但是不难发现，这样时间复杂度就变成了 $O(n^2)$，这样是难以接受的。

4.2 数据结构优化

为了降低时间复杂度，我们可以使用以下的方法：

对于每一个高精度数都使用一棵平衡树来维护数组 A（实现的过程中可以使用 STL 的 set）。可以发现，一棵大小为 m 的子树，其小数的位数是 $O(m)$ 的。证明很简单，对于每一个节点，函数值的整数部分至多会增加 1，所以整数部分的最大值是 $O(m)$ 的。而从 3 的计算式可以发现，除以 2 的次数是和整数部分大小同阶的，所以小数的位数是 $O(m)$ 的。

所以就可以使用启发式合并的方法。可以发现在整个过程中，高精度数总共就只会进行以下两种操作：

1. 除以 2^m 次方，我们可以把整数部分的后 m 位中为 1 的位数给插入平衡树中（注意根据 w 来调整插入的值），最后再把 w 减去 m 就好了。

2. 加上一个数，因为是使用启发式合并的方法，就可以枚举较小的高精度数数中的每一位，然后暴力插入另一个高精度数中，至于进位也可以暴力进行，即查找当前插入的权值在平衡树中存不存在，如果存在那么把它删掉然后把插入的数加一。因为每一次进位所有高精度数中 1 的个数一定减少了 1，而一共至多有 $O(n)$ 个 1，所以暴力进位的时间复杂度是 $O(n \log n)$ 的。最后再把整数部分相加就好了。

综上，我们以 $O(n \log^2 n)$ 的时间复杂度，$O(n \log n)$ 的空间复杂度解决了这题。