前言

主要内容是: 必胜态与必败态的转移, 几大经典博弈模型(nim, 巴什(应该读shi 二声 哭哭), 威佐夫和扩展威佐夫, 阶梯, multi-nim, anti-nim, 二分图博弈, 斐波那契博弈, 巧克力), SG函数, 先手必胜方案数, 打表找规律技巧.

这里的博弈指的都是平等博弈, 也就是两个人能进行的操作是一样的. 非平等博弈自然就是两个人能进行的操作不一样. 而且一般遇到这种题, 都是假设两个人都非常的聪明, 以最优策略进行游戏. 而且进行一系列操作后, 游戏都能结束, 也就是状态转移图是DAG.

写博弈题一般有三种方法:

第一种: 读懂题意(或者经过一系列巧妙地转化)发现是某个经典模型, 直接秒了.(如果不知道这个模型, 一般就只能第二种)

第二种: 打表找规律, 猜结论瞎搞. 一般这种情况罚时上天.

第三种: 数据范围小或者状态很容易表示, 能够上SG函数. 但可能需要一些小优化.

第四种: 通过一系列严谨的推导和分类讨论, 证明必胜态和必败态的条件.(tql)

论思维灵活性, 博弈不比dp差, 博弈的思维性体现在证明时需要各种讨论.

必胜态与必败态

状态转移指的是某个人进行某次操作后, 局面从状态A转变成状态B了.

定义每个状态为必胜态, 当且仅当先手存在一种策略(无论对手的策略如何)都能赢.

必败态就是先手无论怎样都会输.

那么显然如果有一个状态A能转移到一个后继状态B, B是必败态, 那么只要进行这个转移就可以就可以让对方输了.所以自己是必胜态.

如果转移不到必败态, 也就是他的转移全是必胜态, 那么就没办法只能哭哭了.

总结一下: 若当前状态的所有后继状态全是必胜态, 那么当前是必败态, 若存在一个必败态, 那就是必胜态.

一般来说, 求解必胜态或必败态可以用类似dp的方法来做.(一般肯定会T, 但可以用来打表). 初始化就是如果没办法操作了就算输, 那么把这些没法操作的状态初始化为0, 表示必败态. 建议用记忆化来搜.

$$dp[S] = 1 - \min_{V \in nxt(S)} \{dp[V]\}$$

几种经典博弈模型

Nim博弈

描述:

算是最经典的一种博弈, 给n堆石子, 每堆石子的数量是a_i, 每次的操作是随便从某一堆中取出至少一个石子.

两个人轮流操作, 不能操作就输

结论:

像这种很多堆石子的问题, 很难用dp来做, 因为状态非常难记. 如果只有两三堆石子还可以用两三维的dp来记转移. 而且实际上打表也很难找出规律. 并且也很难从正面推出结论. 像这种只能马后炮证明法(最终状态满足这个结论, 然后对于每一个必胜态都能找到一个后继的必败态, 每一个必败态转移不到另一个必败态).

这不是循环论证! 是一种类似数学归纳法的证明, 只不过这里不是证明若k成立, 则k+1成立, 而是假设S的所有后继状态成立, 然后推出S也成立

但这算是非常众所周知的博弈题了, 结论就是a_i的异或和若是0, 那么先手必败, 否则先手必胜.

引理:

> 若$Xor\ a_i = x$, 那么必定存在一对y和i满足$y \leq a_i$, 且$x \otimes y \otimes a_i = 0$
>
> 首先x的最高位一定来自某个a_i, 那么y把这第i堆石子中的最高位去掉, 前面的位就和a_i一样即可, x的最高位就没了. y的较低的几位可以任意取值, 那么只要取值为$x \otimes a_i$的后几位即可.

知道这个结论后, 证明就很方便了. 首先若没有石子了, 异或和就是0, 显然是必败态.

若异或和不是0, 那么根据上述引理, 可以转移到异或和是0的状态, 也就是必败态.

若异或和是0, 那么因为不能不取, 于是只要取了$x > 0$个石子, 显然$a_i \otimes (a_i - x) \neq 0$, 于是异或和不为0, 也就是只能转移到必胜态.

网上的证明基本都差不多长这样, 也没有直接证明的(博弈基本就这样, 只有知道了结论才能才结论, 至于怎么知道的, 只有一个字)

巴什博弈

描述:

有n个石子, 每次最多拿m个, 至少拿一个.

两个人轮流操作, 不能操作就输.

结论:

还是马后炮证明, 先给出结论, 若$n \equiv 0 \mod (m+1)$, 则必败, 否则必胜.

首先$n = 0$, 显然必败, 符合结论.

若$n \neq 0 \mod (m+1)$, 则显然$n \equiv x \mod (m+1)$, $(1 \leq x \leq m)$, 那么取走x个石子即可转移到必败态.

若$n \equiv 0 \mod (m+1)$, 那么就不得不转移到必胜态.

威佐夫博弈与扩展威佐夫博弈

描述:

有两堆石子, 每次可以拿其中一堆石子至少一个, 或者在两堆石子中拿相同数量个(至少1个). 石子数10^9

两个人轮流操作, 不能操作就输.

扩展版: 第二种操作变成在第一堆拿x,第二堆拿y, 满足$|x - y| \leq d$.

结论:

首先有个很显然的$O(N^3)$打表, 稍微观察一下实际上可以优化到$O(N)$, 因为如果状态(a,b)是必败态, 那么$(a+x,b),(a,b+y),(a+k,b+k)(x>0,y>0,k>0)$都是必胜态, 也就是说在之前的必败态中, 没有$(a,*),(*,b)$或$(x,x+b-a)$ 形式的, 那么(a,b)就是必败态.

于是枚举差值k, 然后找到最小的还没出现过的数字a, 然后$(a,a+k)$就是k的必败态, 显然$a+k$之前肯定没有出现过, 再把a和a+k放到出现过的数字中. 然并卵, 还是做不了.

开始瞎搞或者看博客, 发现如果$(a,b)(a<b)$如果是必败态, 那么一定满足$a=\lfloor\frac{\sqrt{5}+1}{2}(b-a)\rfloor$

于是判断一下这个是否满足条件即可(可以家中常备高精度威佐夫).

至于证明, 这里不是数学课. 想知道可以看博客(还是不建议去学, 因为如果能利用Betty定理或它的思想来做题, 至少出线水平哭哭)

扩展的话也就是个式子:
$a=\lfloor\frac{1-d+\sqrt{d^2+2d+5}}{2}(b-a)\rfloor,b=\lfloor\frac{d+3+\sqrt{d^2+2d+5}}{2}(b-a)\rfloor$

如果要求所有的必败态, 可以通过枚举$(b-a)$得到, (a,b)也称为第$b-a$个必败态哭哭.

阶梯博弈

描述:

有n堆石子, 每次可以选第$i(i>0)$堆中至少1个石子, 放到$i-1$堆里.

两个人轮流操作, 不能操作就输.

结论:

这题既不能打表找规律, 正经推也推不出来, 于是马后炮证明.

首先偶数堆石子对答案不影响, 因为你从$2i$堆拿石子, 那么后手可以把相应的石子数从$2i-1$堆里面拿出相应数目的石子, 还是影响. 但是奇数堆会有影响, 因为从第1堆石子拿石子, 后手就不能复制这个操作来抵掉影响. 而偶数堆没影响直接视作黑洞, 所有从奇数堆来的石子都直接吃掉, 可以理解为从奇数堆拿石子就真的被拿掉了. 于是做一次奇数堆的nim即可.

看上去很不严谨, 实际上还是很严谨的, 请细品.

例题:

有n个石子在坐标轴上, 每次可以选一个在x的石子放到y上, 满足$0\le y<x,[y,x-1]$上没有石子.

两个人轮流操作, 不能操作就输.

做法:

第一个石子可以左移$can_1=a_1$个位置, 第二个可以$can_2=a_2-a_1-1$个位置, 第i个可以$can_i=a_i-a_{i-1}-1$个位置

若你移动第i个石子x个位置, 那么$can_i-=x,can_{i+1}+=x$. 若把can_i看做第i堆石子数, 相当于把第i堆的石子移动到第$i+1$堆里, 相当于下标反向的阶梯博弈哭哭.

multi-nim和nim-k

哭哭这个讲到SG函数再来吧.

Anti-nim

描述:

给n堆石子, 每堆石子的数量是a_i, 每次的操作是随便从某一堆中取出至少一个石子.

两个人轮流操作, 不能操作就赢(或者拿走最后一个石子输哭哭).

结论:

具体证明可以看这个, 反正也是马后炮证明+大分类讨论:https://www.cnblogs.com/linzhuohang/p/15046448.html

总之结论是:

必胜态是:　　(所有石子堆都是1且异或和=0)或(存在一个石子堆大于1且异或和!=0)

否则就是必败态.

二分图博弈:

描述:

给一个二分图G和起点s, 每次可以走到一个未走到过的相邻的点.

两个人轮流操作, 不能操作就输.

结论:

马后炮: 若所有可能的二分图最大匹配都必须包含s, 那么先手必胜,否则先手必败.

进行移动后相当于把之前的点删了.

若存在一个二分图最大匹配不包含s, 那么随便他怎么走到v, 都会转移到一个状态$(G\setminus\{s\},v)$, 而此时v肯定属于最大匹配, 反证法: 假设可以不属于, 那么在G里为什么v和s不匹配? 所以一定会转移到必胜态. 所以当前状态是必败态.

若二分图最大匹配必须包含s, 那么他走到和他匹配的v, 就会转移到一个状态$(G\setminus\{s\},v)$, 而此时v肯定可以不属于最大匹配, 因为之前G的最大匹配把(s,v)这对匹配删掉后, 一定是$(G\setminus\{s\})$的最大匹配, 反证法: 假设还不是, 那么只有可能是v还能连别的边, 那么最大匹配数居然没变, 这不是说明s可有可无吗? 和最大匹配必须包含s矛盾. 于是可以转移到必败态, 当前状态是必胜态.

实际上不是二分图应该也能做, 因为上述证明中没有用到他是二分图的性质, 只不过要用比较冷门的带花树求无向图的匹配了哭哭.(没试过, 可能是假的)

斐波那契博弈

描述:

只有一堆石子n个, 第一次可以取至少一个但不能拿完, 之后每一次都至少取1个, 最多取上一次取的2倍.

两个人轮流操作, 不能操作就输.

结论:

首先可以用个二维dp来打表$dp[i][j]$表示还剩i个石子, 当前最多取j个的胜负状态. 转移较为显然.

打表之后发现规律很明显, 当n是斐波那契数时就是必败态, 遂A.

马后炮证明:

显然1,2必败.

若n是大于2斐波那契数, 记为$f[k]=f[k-1]+f[k-2](k\geq 3)$, 显然如果拿了大于等于$f[k-2]$个石子就G了, 因为$f[k-1]=f[k-2]+f[k-3]\geq f[k-2]+f[k-2]$, 也就是对方可以一次性拿走剩下的.

那么现在先手不能一次性拿走$f[k-2]$个, 那么相当于玩一个$n=f[k-2]$的游戏, 这个游戏根据假设先手必败, 即后手必能拿走最后一个的, 而且如果中间过程先手可以拿超过$f[k-2]$个的石子, 那么$n=f[k-2]$就先手必胜了(可以拿为什么不拿). 因此, 无论先手怎么拿, 后手一定尽量会把$f[k-1]$个石子传给先手.

既然$f[k-1]$和$f[k-2]$必输, 如果两个都输了, $f[k]$显然也输了. 现在唯一有可能赢的方法就是后手在把$f[k-1]$个石子转移过来时, 你能一次性全拿走. 我们考虑后手在转移到$f[k-1]$时拿的石子个数最多可能是多少. 假设之前拿了$l(0\leq l\leq f[k-2]-2)$个, 然后最后第二次先手拿了$1\leq x<f[k-2]-l$个, 然后后手拿了$1\leq y\leq 2x$个, 并且:

$$
\begin{aligned}
x+y+l&=f[k-2]\\
x+y&=f[k-2]-l\geq \frac{3}{2}y\\
y&\leq \frac{2f[k-2]-2l}{3}
\end{aligned}
$$

当$l=0$, y可能取到最大值$\frac{2f[k-2]}{3}$, 那么你下一次就可以取$\frac{4f[k-2]}{3}$个, 如果大于等于$f[k-1]$你就**有可能**赢啦!

$$
\begin{aligned}
&\frac{4f[k-2]}{3}-f[k-1]\\
=&\frac{4f[k-2]-3f[k-1]}{3}\\
=&\frac{4f[k-2]-3f[k-2]-3f[k-3]}{3}\\
=&\frac{f[k-2]-3f[k-3]}{3}<0
\end{aligned}
$$

很可惜, 哭哭, 连有可能赢的可能都没有.

到这里已经证明了一半, 另一半就是不是斐波那契数则必胜. 这里就通过数学归纳法+构造必胜策略来证明.

有一个定理,就是可以把n唯一的写成若干个不相邻的斐波那契数的和, 假设有相邻, 设最大的那两个是$f[k],f[k-1]$, 此时一定没有$f[k+1]$因为是最大的两个, 于是用$f[k+1]$代替那两个即可, 可能会产生新的相邻, 重复此操作即可, 唯一性的话可以自己搜齐肯多夫定理.

于是把n写成$f[i_1]+f[i_2]+\cdots+f[i_x](i_1<i_2<\cdots<i_x),x>1$, 那么先手拿$f[i_1]$个就能赢, 因为:

$$
\begin{aligned}
i_2&\geq i_1+2\\
f[i_2]&\geq f[i_1+2]=f[i_1]+f[i_1+1]>f[i_1]*2
\end{aligned}
$$

也就是后手怎么也拿不了$f[i_2]$个, 因为斐波那契数必败, 所以先手(后手的后手)必定能拿走$f[i_2]$的最后一个石子, 而且就算先手最后一次拿了$f[i_2]$个, 接下来的后手还是不能一次拿$f[i_3]$个, 继续陷入斐波那契数必败的陷阱, 直到最后一个都不能翻身. 然后后手就被玩死了哭哭.

可能中间某一个地方会卡住, 但是想通了会觉得很妙. 不过当你证完比赛也应该结束了.

巧克力博弈

描述:

有一个$n*m$的巧克力, 左下角是(1,1), 坏掉了哭哭. 每次你必须选一个还存在的巧克力(i,j), 然后把他的右上方所有的巧克力全吃掉, 换句话说若之前选过(x,y), 那么你现在就不能选$(i,j)(i\geq x\wedge j\geq y)$

两个人轮流操作, 吃掉坏掉的巧克力就输.

结论:

这东西不是很好用dp来打表, 要稍微维护一下轮廓线, 但是通过打表发现, 只有$1*1$必败, 否则必胜.

这次是玄学证明:

首先你就吃(n,m)这个巧克力, 就是玩儿.然后后手随便怎么吃了一口(a,b), 必定会包含(n,m), 相当于你这一步什么都没做, 然后假设他经过一系列操作必胜了.那么现在你悔棋, 直接第一步就把他的(a,b)吃了, 然后你就把他的必胜方案偷过来了.

如果吃了(n,m)后对方必败, 那显然你已经赢了哭哭.

例题:

有$1,2,\ldots,n$这么n个数, 每次可以选一个还存在的数, 然后把他的约数全删了.

两个人轮流操作, 不能操作就输.

结论:

先手必胜, 第一步就选1, 若对方接下来选x, 然后经过一系列操作赢了, 然后悔棋, 第一步就选x, 然后就赢了哭哭.

green博弈

一种树上的博弈, 不是很会哭哭.

不知道怎么分类的博弈1

描述:

进行k次游戏, 每次都是同一个游戏, 除了第一次以外, 后面每一次的先手是上一局游戏输的人. 全局的输赢只看最后一句, 问谁会赢.

结论:

先讨论一下单次游戏的状态. 设第一局先手的人为A, 后手的为B, 假设后手必胜, 那么第一局B能打赢, 第二局还是A先手, B还是能打赢, 一直到第k局都是, 所以若后手必胜, 则B必胜.

现在后手想赢也赢不了, 问题就在于先手能不能控分, 于是还有两种情况

假设先手能必胜, 也能故意输掉, 那么A前k-1局一直输, 然后最后一局赢就行了.

如果打不了假赛, 先手只能赢, 那么要看k的奇偶性了.

至于怎么求先手是否能故意输掉, 考虑在dp时多加一个0/1状态, $dp[S][0]$表示状态S先手是否能必败, $dp[S][1]$表示先手是否能必胜. 那么在转移的时候, $dp[S][1]$还是正常的转移, $dp[S][0]$的转移就是看是否能转移到$dp[T][0]=0$的, 也就是让对方输不了.

字符串博弈

描述:

一开始有一个空串, 每次添加一个字母在末尾, 要保证当前串满足[是一些串中的某个串的前缀/是一些串中的某个串的子串].

两个人轮流操作, 吃掉坏掉的巧克力就输.

做法:

第一种问题, 前缀:

只要建一棵字典树, 那么当前字符串可以表示为字典树上的某个节点, 并且下一个字符也必须是字典树的某个儿子. 相当于另一个问题: 一开始在根节点, 每次只能往儿子走. 这就一个经典的dp就完事了.

第二种问题, 子串:

只要建一个广义后缀自动机, 这个自动机就是个DAG, 那么当前字符串可以表示为字典树上的某个节点, 并且下一个字符的转移也必须是该节点的出边. 相当于另一个问题: 一开始在DAG上的1号点, 每次只能走一条出边. 这也是一个经典的dp就完事了, 具体看后面的例题.

SG函数

$SG[S]$的定义就是状态S能转移到的所有状态T的$SG[T]$的值的mex, 或者说:

$$SG[S] = mex_{T \in nxt(S)}(SG[T])$$

mex就是之前讲座说过很多次的"集合中最小的没有出现过的非负整数"

SG的求法一般有两种:

1. 没有规律的转移, 就只能暴力枚举所有转移来求mex了.
2. 打表或者证明, 直接得出SG的通项公式.

例1

一堆石子n个, 一次可以拿任意个石子, 求SG[n].

首先$SG[0] = 0$, 因为他不能进行任何转移, 空集的mex就是0

然后假设$SG[i] = i, i \leq k$, 证明$SG[k+1] = k+1$

因为$SG[k+1]$能转移到所有的$0 \leq i \leq k$, 所以

$$SG[k+1] = mex(SG[0], SG[1], \ldots, SG[k]) = mex(0, 1, .., k) = k+1$$

于是$SG[n] = n$

例2

一个DAG, 从S点开始, 每次可以任意走一条出边, 不能操作就输, 求他的SG[S].

显然这东西不可能用数学归纳法之类的来证明. 只能暴力, 一般都用反向图拓扑排序或记忆化搜索来做.

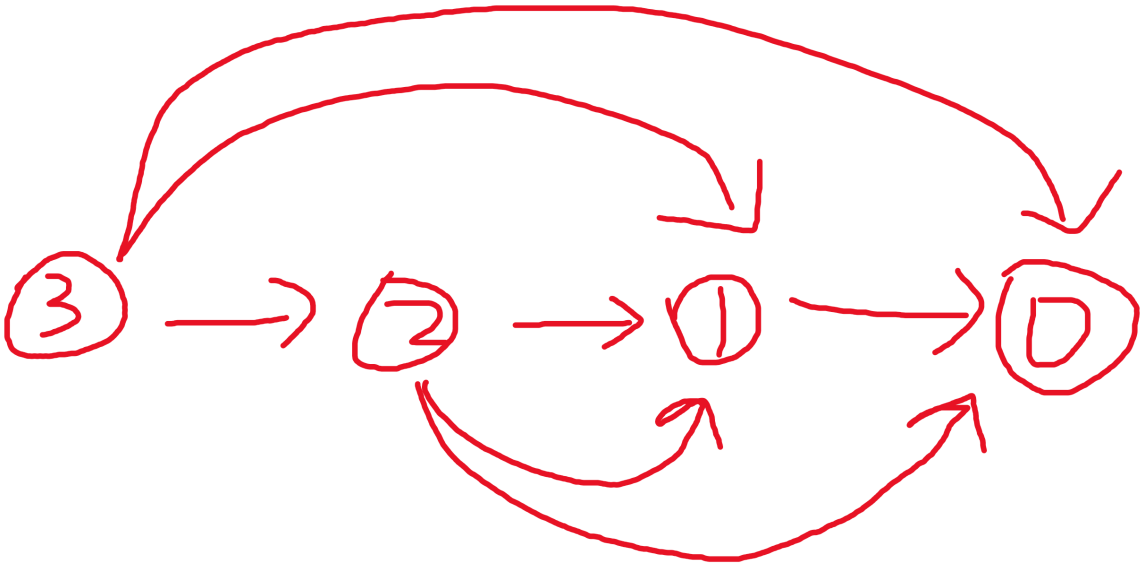
首先出度为0的点显然SG值为0.

我比较喜欢记忆化搜索, 但是反向图拓扑排序有个小细节就是队列搜到当前点了才开始算SG值(否则要开n个桶), 所以给出两种代码(没跑过):

```cpp
/**
记忆化搜索:
*/
int SG[MAXN];
vector<int>to[MAXN];
int tak[MAXN];
int getSG(int u)
{
    if(SG[u]!=-1)return SG[u];
    for(auto v:to[u])getSG(v);// 必须先算好v, 再更新桶, 否则桶会乱.
    for(auto v:to[u])tak[getSG(v)]=1;//可以写++, 但是只需要知道是否存在即可
    for(SG[u]=0;tak[SG[u]];++SG[u]);// 算SG
    for(auto v:to[u])tak[getSG(v)]=0;//不能用maxsg来清空会被卡. 这样才是理论O(N+M)的哭哭
    return SG[u];
}
```

```cpp
/**
反向图拓扑排序:
*/
vector<int>to[MAXN],rto[MAXN];// rto是反向边
int tak[MAXN],du[MAXN];
void getSG()
{
    queue<int>q;
    for(int i=1;i<=n;i++)if(to[i].size()==0)q.push(i);// 把没有出度的点放进去.
    for(int i=1;i<=n;i++)du[i]=to[i].size();
    while(!q.empty())
    {
        int u=q.front();q.pop();
        for(auto v:to[u])tak[SG[v]]=1;//可以写++, 但是只需要知道是否存在即可
        for(SG[u]=0;tak[SG[u]];++SG[u]);// 算SG
        for(auto v:to[u])tak[SG[v]]=0;// 清空
        for(auto v:rto[u])
        {
            du[v]--;
            if(du[v]==0)q.push(v);//拓扑排序.
        }
    }
}
```

根据SG的定义, 我们可以发现SG[x]的后续转移满足大小为SG[x]+1的完全有向图(把SG相同的点合并成一个点):



于是SG值如果是3, 相当于有3个石子, 每次可以拿任意多个.

这里还可以注意到一个性质, 就是最大的SG值是$O(\sqrt{M})$级别的. 这不止是对于DAG, 可以说所有的博弈, 如果所有的转移有M种(即有M对不同的(S,T), 使得S能转移到T), 那最大的SG就是$O(\sqrt{M})$级别的.

例3:

有K个DAG, 每个DAG有一个棋子在1点, 有n_i个点, m_i个边, 每次可以选一个棋子走一条边.
$\sum_{i=1}^{k} n_i \leq 10^5 \sum_{i=1}^{k} m_i \leq 10^6$

两个人轮流操作, 不能操作就输.

做法:

求出每个图的$SG_i[1]$, 于是问题相当于第i堆石子有$SG_i[1]$个石子, 每次选一堆取走任意个,两个人轮流操作, 不能操作就输.

经典nim博弈, 异或和一下就好?

这就是把SG和nim结合起来了, 这也是SG函数的经典用法. 就是把一个游戏分解成若干个子游戏, 当然子游戏间不能影响(一次只能玩其中一个). 这样的话所有子游戏的SG值的异或和就是当前游戏的值了.这些子游戏甚至可以不是一类游戏, 你甚至可以一边玩DAG, 一边玩取石子, 一边吃巧克力(如果你能求出他的SG值的话).甚至可以搞Anti-nim.

例4: nim-k

题意

有n堆石子, 每次可以从一堆石子中取至少1个不超过k个.

两个人轮流操作, 不能操作就输.

做法:

首先还是求单堆石子的SG函数, 打表可以发现$SG[x] = x \mod (k+1)$.

数学归纳法, 首先显然$SG[i] = i, 0 \leq i \leq k$

假设对于所有的$i \le X$, $SG[i] = i \mod (k+1)$,

$SG[X+1] = mex_{i=1}^{k}(SG[X+1-i]) = mex_{i=1}^{k}(X+1-i \mod (k+1))$

若$X+1\%(k+1)=0$, 则$SG[X+1] = mex(k, k-1, \dots, 1) = 0$ 成立.

若$X+1\%(k+1)=y$, 则$SG[X+1] = mex(y-1, \dots, 0, k, k-1, \dots y+1) = y$, 也成立.

所以$SG[n] = n\%(k+1)$

然后把SG异或起来即可.

例5: multi-nim

题意:

有n堆石子, 每次可以从一堆石子中取至少1个, 或者把一堆石子分裂成两堆石子(每堆至少有1个). $a_i \le 10^9$

两个人轮流操作, 不能操作就输.

做法:

a_i很大, 求SG没什么办法, 只能打表找规律.

发现分裂这个操作不是很好打表. 要打就得用map<vector,int>的方法来暴力记忆化搜索了(也不是不行, 应该能打到15左右吧, 也许能看出规律).

考虑怎么更快的求SG.

首先我们证明一个nim游戏本身的SG函数就是石子数的异或和.

假设某个状态$a_1 \otimes a_2 \cdots \otimes a_n = A$, 那么某一堆石子拿走了$x(0 < x < a_i)$个, 设是第一堆石子被拿走了, 显然$(a_1 - x) \otimes a_2 \cdots \otimes a_n = (a_1 - x) \otimes a_1 \otimes A \neq A$, 也就是说异或和是A的状态转移不到一个SG是A的状态.

接下来只要证明该状态一定能转移到SG值小于A的状态: 还是和nim类似的证明方法, 假设要转移到SG值是x的状态, 那么因为$A > x$, 所以A和x从高位到低位第一个不一样的位, 一定是A为1, x为0这种情况, 那么A的1一定是由某堆石子贡献的, 那把这堆石子的1去掉, 后面几位随便取, 只要取的和x后面几位一样就行了. 因此:

$$SG[S] = mex(0, \dots, A-1, [A+1, \dots]) = A$$

有了这个, 我们就能$O(N^2)$的求出SG了, 代码:

```
int sg[MAXN];
int getsg(int n)
{
    if(sg[n]!=-1)return sg[n];
    set<int>st;
    for(int i=1;i<=n;i++)
    {
        st.insert(getsg(n-i));/// 取走i个石子
    }
    for(int i=1;i<n;i++)
    {
        st.insert(getsg(i)^getsg(n-i));// 分裂
    }
    for(sg[n]=0;st.count(sg[n]);sg[n]++);
    return sg[n];
```

```
16 }
```

```
sg[0]= 0
sg[1]= 1
sg[2]= 2
sg[3]= 4
sg[4]= 3
sg[5]= 5
sg[6]= 6
sg[7]= 8
sg[8]= 7
sg[9]= 9
sg[10]= 10
sg[11]= 12
sg[12]= 11
sg[13]= 13
sg[14]= 14
sg[15]= 16
sg[16]= 15
sg[17]= 17
sg[18]= 18
sg[19]= 20
sg[20]= 19
sg[21]= 21
```

找完规律直接异或和一波带走.

如果要证明这个规律需要数学归纳法+大分类讨论. 闲得无聊可以看看https://www.cnblogs.com/sdlang/p/13649774.html 还是那句话, 等你证完比赛已经结束了, 而且得先有结论才能证.

这样下次遇到各种奇葩限制比如必须分裂成大小大于等于2的两堆石子之类的, 也能很快的打表了哭哭.

先手必胜方案数:

一般形式就是给一个状态, 问你先手第一步有多少种不同的操作方法使自己必胜.

还有比较像的题型就是初始状态不明确, 都有可能, 问有多大概率先手能必胜.

在做这类题时, 你必须要把先手必胜必败的策略给搞懂了. 像之前的巧克力博弈, 证明先手必胜时并没有给出必胜策略, 比较瞎搞, 像这种就做不了了哭哭.

例1:

题意:

唉我都懒得复制粘贴了, 就nim游戏先手有多少种第一步操作的方案能必胜.

做法:

先手必须把必胜态转移给对方, 也就是异或和是0的状态转移给对方, 否则自己就输了. 相当于问有多少种方案选一个a_i减x, 使得异或和为0.

显然对于每一个i, 只有一个x可能满足条件. 设当前异或和为S, 这个x必须满足:

$$0 < x \le a_i$$
$$a_i \otimes S \otimes (a_i - x) = 0$$

即:

$$a_i - x = a_i \otimes S$$
$$0 < x = a_i - (a_i \otimes S) \le a_i$$

对于每个i判断一下即可.

例2:

题意:

只有一堆石子$n(n \le 10^7)$个, 第一次可以取至少一个但不能拿完, 之后每一次都至少取1个, 最多取上一次取的2倍.

两个人轮流操作, 不能操作就输.

问先手第一步有多少种第一步操作的方案能必胜.

做法:

斐波那契博弈, 根据之前的构造性证明, 若$n = f[i_1] + f[i_2] + \cdots + f[i_x], x > 1, i_1 < i_2 < \cdots < i_x$, 先手拿$f[i_1]$就能赢. 问题在于是否还有别的赢法呢?

假设取了x个, 若$n - x = f[j_1] + f[j_2] + \cdots + f[j_y]$, 并且$f[j_1] \le 2x$, 那么后手直接把这个$f[j_1]$取走你接下来就只能被玩死. 若$f[j_1] > 2x$, 那么根据结论可得, 对方必定取不完这$f[j_1]$个石子, 然后和之前一样必赢.

于是只要检查每个x即可, 重点在于怎么快速分解$n - x$, 可以发现每次取当前最大的能取的斐波那契数就一定能保证不会相邻, 假设会相邻, 请问你上一次为什么不取更大的那个斐波那契数呢? 1e7以内只有35个斐波那契数.

但$O(35N)$还是有点慢, 有很多常数优化的点(剪枝).

首先显然$x \le \frac{n}{3}$, 还可以$O(N)$尺取预处理每个数n能取的最大斐波那契数, 因为不相邻所以最多跳17次. 稍微卡卡能过(虽然并没有这一题)

例3:

题意:

给一个二分图G和起点s, 每次可以走到一个未走到过的相邻的点.$N < 2000, M < 5000$

两个人轮流操作, 不能操作就输.

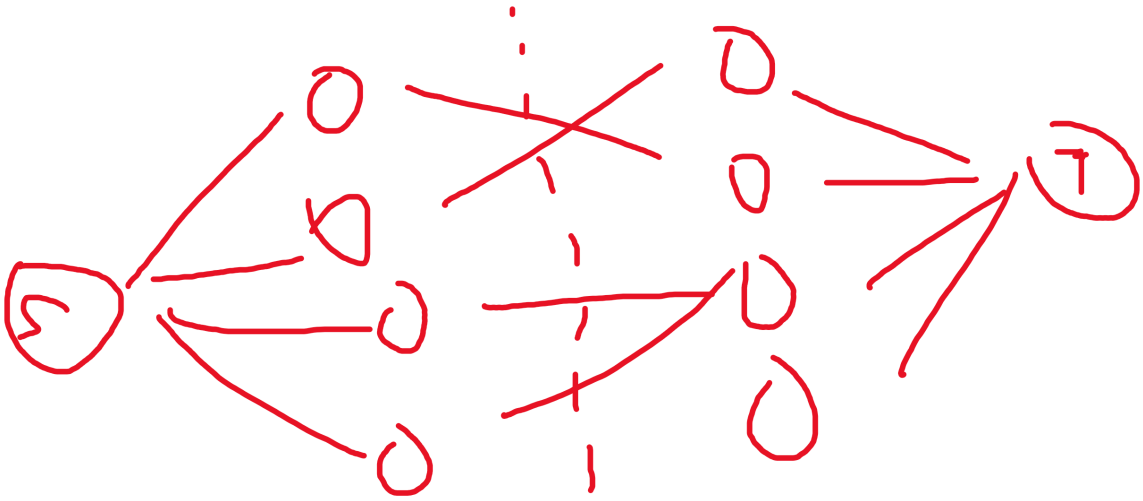
若先手必胜, 求第一步有多少种移动方案先手必胜

做法:

首先二分图博弈的结论就是若当前点在最大匹配中是可有可无的, 那么当前状态就是必败态.

枚举走到一个新点x, 此时s不能再走可以直接当做删掉了, 此时的二分图最大匹配数是p. 我们再看看是否x是必败态即可, 那么就要把x删掉再跑一次最大匹配. 时间复杂度$O(N \cdot M\sqrt{N})$.

考虑跑dinic的时候, 最后没有流量的节点就是当前这个最大匹配没用上的点, 即可有可无, 那么走到这些点就是转移到必败态了, 再考虑当前有流量的点是否可有可无, 那么就把当前的流退回去, 然后再把这个点的流量修改为0, 再跑dinic, 这样$O(M)$就能得到一次最大流, 最多退N次, 于是复杂度$O(NM)$



非常规博弈题

看心情讲吧哭哭.