原创 博弈论浅谈

2019-04-02 20:21:33 我是一只计算鸡 阅读数 1880 更多

1：巴什博弈（Bash Game）

只有一堆n个物品，两个人轮流从这堆物品中取物，规定每次最少取一个，最多取m个。最后取光者获胜。

当n=m+1时，无论怎么取，先手都是必败，因为先手一次取不完，而先手无论取多少个，后手一定能取完。

如果先手想要必胜的话，就给后手留m+1个，这样无论后手怎么取先手都能取到最后一个，可以再进一步，给后手留m+1的倍数个，相当于将m+1个重复。

如果用数学公式表示：n=(m+1)*r+s 如果s不等于0，先手就先取s个，给后手留m+1的倍数个，这样先手必胜，反正，s等于0，先手面临m+1的倍数手怎么取都是必败。

下面给出巴什博弈（Bash Game）模板

```
#include<cstdio>
using namespace std;
int main()
{
  int n,m;   // n个物品 至少取一个，最多取m个
  while(scanf("%d %d",&n,&m)==2){
    if(n%(m+1)==0) printf("Lost\n");
    else printf("Win\n");
  }
  return 0;
}
```

2：Fibonacci Nim（斐波那契博弈）

有一堆个数为N的石子，游戏双方轮流取石子

1）：选手第一次不能把石子取完

2）：之后每次可以取的石子数介于1到对手刚取的石子数的2倍之间（包含1和对手刚取的石子数的2倍）。

既然是Fibonacci博弈，肯定和Fibonacci数列有关，1，2，3，5，8...当n=1时，先手不能第一次把石子取完，所以先手必败，当n=2时，先手第一次[illegible]先手必败，所以我们猜测Fibonacci数列构成必败态，下面我们用数学归纳法来证明一下，

当n = 1时，先手必败。

假设n = k时，成立，

当 n = k+1时，f[k+1]=f[k]+f[k-1]，我们把石子看成k堆和k-1堆，一定可以看成k堆和k-1堆，因为如果先手取的石子数大于等于f[k-1]，则后[illegible]由于假设成立，所以对于k-1堆，后手一定能取到最后一颗，我们现在来讨论后手取k-1堆的石子数的情况，如果先手第一次取的石子数y>=1/3[illegible]最后所剩下的石子数小于2y，所以后手可以一次取完且取到最后一颗，此时后手取的石子数为x<=2/3*f[k-1],现在的问题是，在这种情况下，后[illegible]k堆，即2/3*f[k-1]和1/2*f[k]的大小问题，由数学归纳法，2/3*f[k-1]小于1/2*f[k],即是先手不能一次取完k堆，由假设可知后手一定能取到k堆[illegible]手必败。

对于n不是FIbonacci数，则先手必胜，这里需要借助Zeckendorf（齐肯多夫定理），任何一个整数都可以表示为若干个不连续的Fibonacci数之[illegible]的时候尽量选取较大的Fibonacci数，

假设 n=f[a1]+f[a2]+f[a3]+f[a4]+f[a5]+f[ap-1]+f[ap],我们先手取掉f[ap],由于ap+1<ap-1(下标），因为用的是不连续的Fibonacci数，这种情况下，不能取完f[ap-1],所以后手相当于面临这个游戏的子游戏(只有f[ap-1]这一堆石子，且后手先取的必败态),由之前的结论可知，先手一定能取到这堆石子[illegible]同理，对于后面的每一堆石子，先手都能取到最后一颗。

下面给出Fibonacci博弈的模板

```
#include<cstdio>
using namespace std;
long long f[50];  //50个Fibonacci数
int main()
{
   f[1]=1;
   f[2]=1;
   for(int i=3;i<50;i++) f[i]=f[i-1]+f[i-2];
   int n;
   while(scanf("%d",&n)==1){
     int i;
     for(i=1;i<50;i++){
       if(f[i]==n) break;
     }
     if(i<50)  printf("Lost\n");   //Fibonacci数为必败态
     else printf("Win\n");
   }
  return 0;
}
```

3：K倍博弈

两个人取一堆n的石子，先手不能全部取完，之后每人取的个数不能超过另一个人上轮取的K倍。

当k=1时，必败态都是2^i,我们可以借助二进制的思想来理解，将n表示为2进制，先手拿掉最后一个1，后手肯定没法去掉更高位的1，所以后手取完，能拿掉最后一个1，所以先手必胜。当n=2^i时，先手必败，因为此时n的二进制只有一个1，先手第一次不能取完，所以先手取了以后，后手一定能取1，然后先手不能去掉更高位的1，所以先手必败。

假设n=6(110)，我们先去掉最后一个1，变为4（100），此时如果对手取两个，那么我们直接去两个就能取完，如果对手取一个，还剩3个，我们能取

当k=2时，这就是一个Fibonacci博弈，可知先手必胜当且仅当n不为Fibonacci数，还是利用，先手取掉最后一个1，后手无法去掉更高位的1，所以后至少还能拿掉最后一个1。Fibonacci数列有一个很好的性质就是，任何一个整数都可以表示为若干项不连续的Fibonacci数，所以我们先去掉最后一个x，后手肯定无法去掉更高的数2x，小于高两位的1，所以后手无法取完。

假设n=11=7+3+1，表示为10101，我们先手去掉最后一个1，后手无法去掉高两位的1，所以后手取完，我们至少还能去掉最后一个1。

当k的时候，想办法构造数列，使得数列的任意两项之间的倍数大于k。

就像Fibonacci博弈一样，我们还是想要构造一个想Fibonacci一样的数列，我们用a数组，表示要构造的数列，b[i]表示a[1..i]所能组成的最大数，为了们还是用Fibonacci数列举例子，显然a[ ]={1,2,3,5,8...},b[3]=4,因为5本身就是Fibonacci数，而6=1+2+3，相邻两项的倍数根本就不大于2，6=1+5，以b数组中的数时我们要构造的数列中的一些满足要求的数的和，a[i]=b[i-1]+1,为什么呢，因为a[i]中的数是不可构造的，因为取到它就是必败。而b[i所能构造的最大数，那么加1，就是无法被前面的数列构造出来，所以只能另外开一项。

关于b[i]的构造，由于b[i]是a[1..i]中的数构造出来的，所以我们一定会用到a[i],不然就成了b[i-1]了，所以我们先要按递减顺序找到a[t]*k<a[i],那么b[i]果前面找不到那么b[i]=a[i],为什么呢，因为前面的数没有k项或者说构造出来太小了，所以只能选取一个，那么肯定选取最大的哪一个，前面a[1...i-1]b[i-1]小于a[i],所以这种情况下b[i]=a[i]。所以我们先手能不能必胜就看n在不在这个a[ ]数组里面。

下面给出K被博弈的模板


```
#include<cstdio>
using namespace std;
const int maxn=1000000+7;
int a[maxn];   //构造的数列
int b[maxn];    //b[i]为a[1..i]所能凑出的最大数
int main()
{
  int n,k;
  while(scanf("%d %d",&n,&k)==2){
    int i=0,j=0;
    a[0]=b[0]=1;
    while(a[i]<n){
      i++;
      a[i]=b[i-1]+1;    //a[i]为a[1..i-1]所能构造出的最大的那个数+1
      while(a[j+1]*k<a[i]) j++;  //寻找临界点
      if(a[j]*k<a[i])  b[i]=b[j]+a[i];   //a[1..j]所能构造出来的最大值加上a[i]
      else b[i]=a[i];   //相邻  小于了K倍 自然构造的最大的数就是a[i]了
    }
    if(a[i]==n)  printf("Lost\n");   //如果数列a中有n则先手必败
    else printf("Win\n");
```

```
    }
    return 0;
}
```

4：威佐夫博弈（Wythoff Game）

有两堆各若干物品，两个人轮流从某一堆或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者获胜。

这要引入一个新的概念奇异局势（ak,bk),前面几个奇异局势 (0,0)，(1,2)，(3,5)，(4,7)，ak时前面奇异局势里面没有出现过的最小的整数，bk=ak+k，成必败态，换句话说，当谁面临奇异局势时，就必败。

奇异局势可以变成非奇异局势，当我们改变某一个变量时，另外一个变量不可能出现在其他的奇异局势中。

非奇异局势可以变成奇异局势。假设当前非奇异局势为（a,b)，要变成的奇异局势为(ak,bk),由前面的结论不难得出，a和b至少有一个变量出现在某个奇
下面分别来讨论一下情况：

- 如果b=a,那么在两堆中同时取走a个变成奇异局势(0,0)。
- 当a=ak，b>bk时，这时我们只要把b变成bk就能变为奇异局势，所以把b减去b-bk即是取走b-bk个，就能得到奇异局势(ak,bk)。
- 当a=ak,b<bk时，这时我们必须在两个数里面同时去掉同一个数才能变为奇异局势，减去ab-2*ak，数学公式怎么推来的，笔者也不清楚。
- 当a>ak,b=bk时，这时我们把a变为ak就能变为奇异局势，所以把a减去a-ak，就能得到奇异局势(ak,bk)。
- 当a<ak,b=bk时，如果a=aj （j<k) ,这时我们把b变为 bj 就能得到奇异局势，所以将 b减去b-bj,就能得到奇异局势(aj,bj)，如果a=bj(j<k),那么直接
面拿走b-aj。

如果面对非奇异局势，先手必胜，反之，后手必胜。

那么对于任意一个局势我们怎么判断是否为奇异局势呢，可能是某个伟大的数学家推出来的一个公式。

ak=[k(1+sqrt(5)/2],bk=ak+k。所以对于任意一个局势，我们先反解k,然后判断ak+k是否等于bk，但是由于k整数，反解的时候int会向下取整，所以
我们还得判断一下k+1。由于2/(1+sqrt(5)=(sqrt(5)-1)/2，所以k=(a(sqrt(5)-1)/2)。

下面给出威佐夫博弈的模板

```
#include<cstdio>
#include<cmath>
using namespace std;
const double ep1 = (sqrt(5.0)-1.0)/2.0;  //黄金分割率  判断奇异局势
const double ep2 = (sqrt(5.0)+1.0)/2.0;
int main()
{
  int n,m;
  while(scanf("%d %d",&n,&m)==2){
    if(m>n){   //交换使n最大
      int tmp=n;
      n=m;
      m=tmp;
    }                 //ak=[k(1+sqrt(5)/2)]  bk=ak+k
                      // k=ak*2/(1+sqrt(5))=ak*(sqrt(5)-1)/2
    int id=m*ep1;   //反解下标k
    int temp1=ep2*id;    //带入下标，求解ak
    int temp2=id+temp1;
    int temp3=ep2*(id+1);    //因为int向下取整 所以要判断下一个下标是否满足
    int temp4=(id+1)+temp3;
    if(temp1==m&&temp2==n) printf("Lost\n");  //面临奇异局势必输
    else if(temp3==m&&temp4==n)  printf("Lost\n");
    else printf("Win\n");
}
  return 0;
}
```


5：尼姆博弈（The Game of Nim）

有多堆物品两个人轮流从某一堆物品取任意多的物品，规定每次至少取一个，多者不限，最后取光者获胜。

我们关心的是对于一个初始局面先行者是否有必胜的策略。尼姆博弈和二进制有密切的联系。我们仅需要将n堆石子的数量逐个异或，如果为0，则为
以先手必败。

下面给出尼姆博弈的模板。

```
#include<cstdio>
using namespace std;
int main()
{
  int n,m;
  while(scanf("%d",&n)==1){
    int cnt=0;
    for(int i=0;i<n;i++){
      scanf("%d",&m);
      cnt=cnt^m;    // 逐个异或
    }
    if(cnt==0)  printf("Lost\n");
    else printf("Win\n");
  }
  return 0;
}
```

6：寻找必败态

必败态就是，在对方使用最优策略时，无论做出什么决策都会导致失败的局面。此类博弈的精髓就是让对手永远面临必败态。

规则1：一个状态是必败态，当且仅当它的所有后继状态都是必胜态，因为无论怎么操作它的后继都是必胜态，所以它是必败态。

规则2：一个状态是必胜态，当且仅当它的后继状态至少有一个必败态，因为后继状态至少有一个状态是必败态才能保证当前状态是必胜态。

拓扑排序逆序递推每个结点是必胜态还是必败态，在判断某个结点是，它的后继都已经判断过了。

7：SG博弈

给定一个有向无环图和一个起使顶点上的一枚棋子，两名选手交替的将这枚棋子沿着有向边进行移动，无法移动者判负。

SG博弈说起来太麻烦了，笔者下次有时间再写吧，orz。

下面浅谈一下笔者对博弈论的感悟。

其实笔者认为，绝大多数博弈，都存在一个均衡点，如果我们可以很好的利用这个均衡点就会得出必胜的策略，怎么利用呢，最简单的就是模仿。考虑题，在大海上航行的有两艘完全相同的船A船和B船，A船在B船前面，谁先到终点谁就胜，但是风向会影响船的行驶，如果你是B，你肯定会调整船帆和更快的速度，才可能获胜。那么如果你是A呢，你会怎么做，当然是模仿B了，因为反正A在B前面，B怎么做，A怎么做。这样A始终都会在B的前面。这用均衡点的一种方式罢了。很多题都可以这样想，2018年ACM—ICPC南京站，就有一道博弈，有N块排好序号的东西，每次只能取连续的任意个数块取完，问你先手是否能必胜，这样的话，我们先手取掉中间的，让剩下的关于这个中间对称，对手怎么取其中一边的，我们就模仿取另外一边的，这样最后一块。