組み合わせゲーム理論 — Grundy数の周辺 —

B3 mikoshiba

唐突な例題：取りつくしゲーム

- 二人で次のようなゲームをする．
- 二人の目の前にはコインの山がひとつある．コインの枚数は13枚．二人で順番に1枚か２枚か３枚のコインを取る．最後にコインをとりきったら勝ち(ルールは最後に取れなくなったら負け).
- 二人が最適な戦略を取るとき先手必勝か？後手必勝か？

- ゲームの終わりから考えてみる.

- コインが0枚で手番がきたら負け，コインが1, 2, 3コインで手番が来たら全部コインをとって勝ち

- コインが４枚で手番がきたら次は1,2,3枚にするしかない．すると相手が残りを取り尽くして負け

- コインが5, 6, 7枚で手番がきたら次は４枚にして相手に回せば勝ち.

- コインが８個で手番が来たら次は5, 6, 7枚にするしかない．すると相手が４枚に調整してきて負ける

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
勝負	負	勝	勝	勝	負	勝	勝	勝	負	?	?	?	?	?

- 以下同様に表をかいてみるとコイン13枚で対面したときは勝ちとわかる
- 実際，自分の手番でコインを取って12, 8, 4枚に調整し続ければ最後に取り切って勝ち

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
勝負	負	勝	勝	勝	負	勝	勝	勝	負	勝	勝	勝	負	勝

- 一般化してコインがx枚のとき先手必勝か？
- 取っていいコインの数が1, 2, 4だったら先手必勝か？
- コインの山が１つじゃなくてN個の山だったら先手必勝か？
- プレイヤーの行動に「コインの山を２つに分ける」を追加したら先手必勝か？

- これらのゲームは先手必勝か？をメインに扱う

代表的なゲーム「Nim」の説明

- コインの山から取っていくゲーム．山の数が複数なのがポイント．毎ターン山を一つ選んで 1 枚以上いくらでもとってOK
- 最後のコインを取ったら勝ち（コインを取れなくなったら負け）
- 例えばコインの山が 3 つで枚数がそれぞれ $(x_1, x_2, x_3) = (1, 2, 6)$だったら先手必勝なのか？

スライド概略

- 与えられた正規形の組み合わせゲームは先手必勝なのか後手必勝なのかをどう判断するか？をテーマとして扱う．特に不偏ゲームをメインに解析.
- 1章）組み合わせゲームのある局面は先手必勝か後手必勝かは再帰で判断できる.
- 2章）しかしながら複数の組み合わせゲームを合わせてプレイするときには再帰による解析のみでは計算機上では時間がかかりうる．そこでNim和，ゲームの和とGrundy数を導入することで解決できることを説明（ただし不偏ゲームで正規形に限る）.

1章　組み合わせゲームの基礎とN-位置，P-位置

- 1.1節　ゲームの分類とルール
- 1.2節　N-位置，P-位置

2章：効率的に先手必勝を判断するアルゴリズム

- 2.1節　NimとNim和
- 2.2節　Grundy数とゲームの和

1章　組み合わせゲームの基礎とN-位置，P-位置

1章　組み合わせゲームの基礎とN-位置，P-位置

- **1.1節　ゲームの分類とルール**
- 1.2節　N-位置，P-位置

2章：効率的に先手必勝を判断するアルゴリズム

- 2.1節　NimとNim和
- 2.2節　Grundy数とゲームの和

1.1.1項　組み合わせゲームと不偏ゲーム

- 組み合わせゲーム・・・二人で交互に手を打てなくなるまで打ち勝ち負けがあるゲームで，確定性と完全情報性があるもの．ふつう有限の手数でゲームは終了する．　例）◎将棋，囲碁　×ポーカー，麻雀

※　確定性　　・・・　サイコロを振る等のランダム要素を含まない
　　完全情報性　・・・　ゲームの進行や二人の手の内がともにわかっている

- 不偏ゲーム・・・組み合わせゲームのうち二人の動かせる局面集合がいつ如何なるときも同じゲーム
例）◎取りつくしゲーム，Nim　×将棋，オセロ

1.1.2項　ゲームのルール

- 正規形・・・ルールの分類の１つ．最後の手を打ったプレイヤーの勝ち．ゲームの局面を動かせなくなった局面(**terminal position**)で手番になったら負け．ほとんどのゲームがこのルール．

- （逆形・・・ルールの分類の１つ．最後の手を打ったプレイヤーが負け．あんまりこのゲームはない．）

- 今回のスライドでは正規形の不偏ゲームのみを扱う
- 勝負がついて両プレイヤーの操作が毎回同じゲームなんだなぐらいの認識でOK

1章　組み合わせゲームの基礎とN-位置，P-位置

- 1.1節　ゲームの分類とルール
- **1.2節　N-位置，P-位置**

2章：効率的に先手必勝を判断するアルゴリズム

- 2.1節　NimとNim和
- 2.2節　Grundy数とゲームの和

1.2節 N-位置とP-位置　その局面を見て勝敗が決まっている！？

- 不偏ゲームの状態（局面）一つ一つに対して決まる．

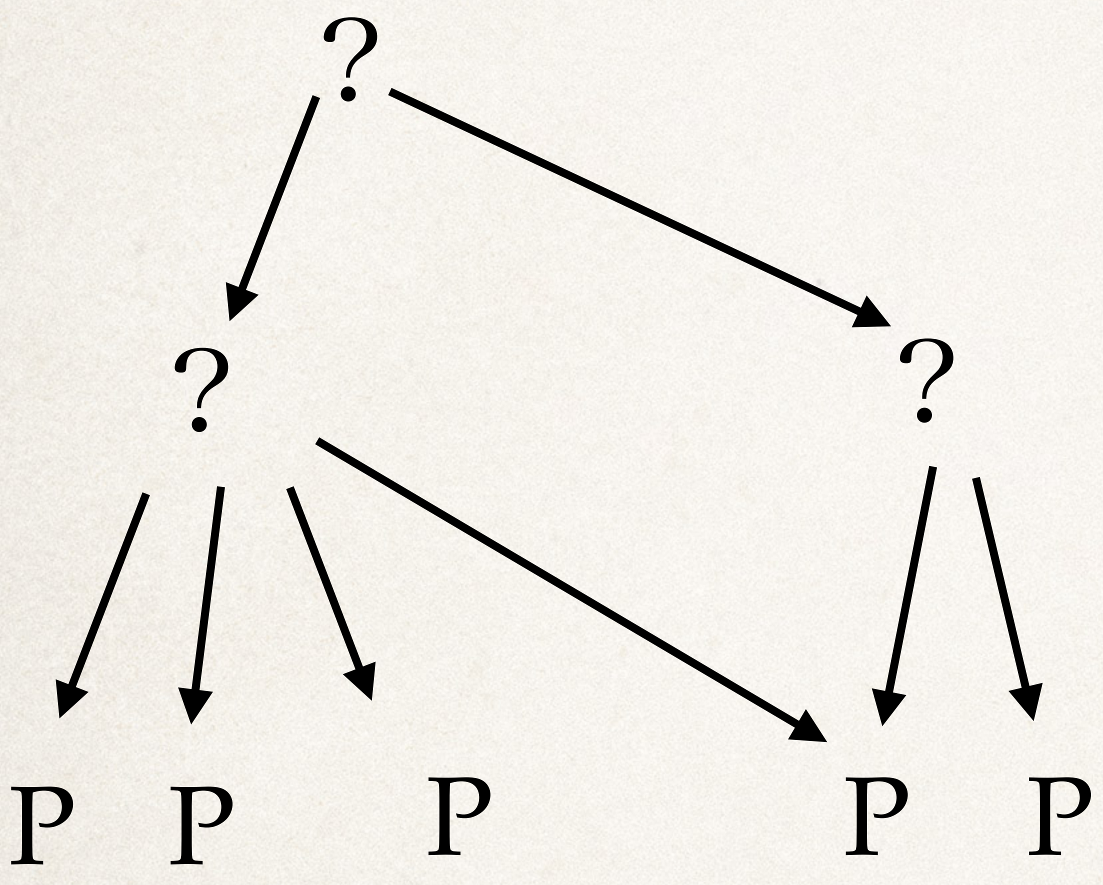
- ゲームの状態が**N-位置(Next position)**である　とはこの局面を見たときにこの局面**を，今から**次に動かすプレイヤーが必勝の場面

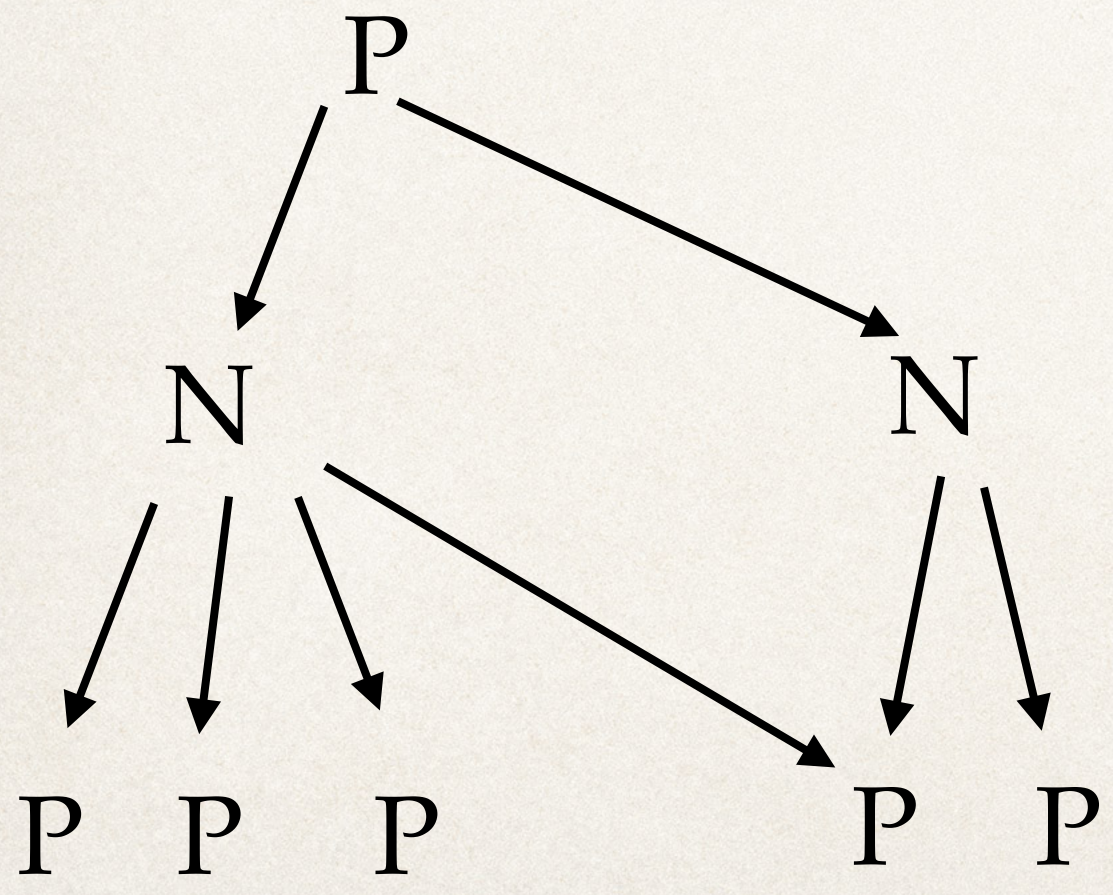
※この局面を見たプレイヤーは上手い局面を選べば必勝（うれしい）

- ゲームの局面が**P-位置(Previous position)**である　とはこの局面を見たときにこの局面**に，ついさっき**動かしたプレイヤーが必勝の場面

※この局面を見たプレイヤーはどうあがいても必ず負ける（かなしい）

1.2.1項　P-位置とN-位置　帰納的な定義と性質

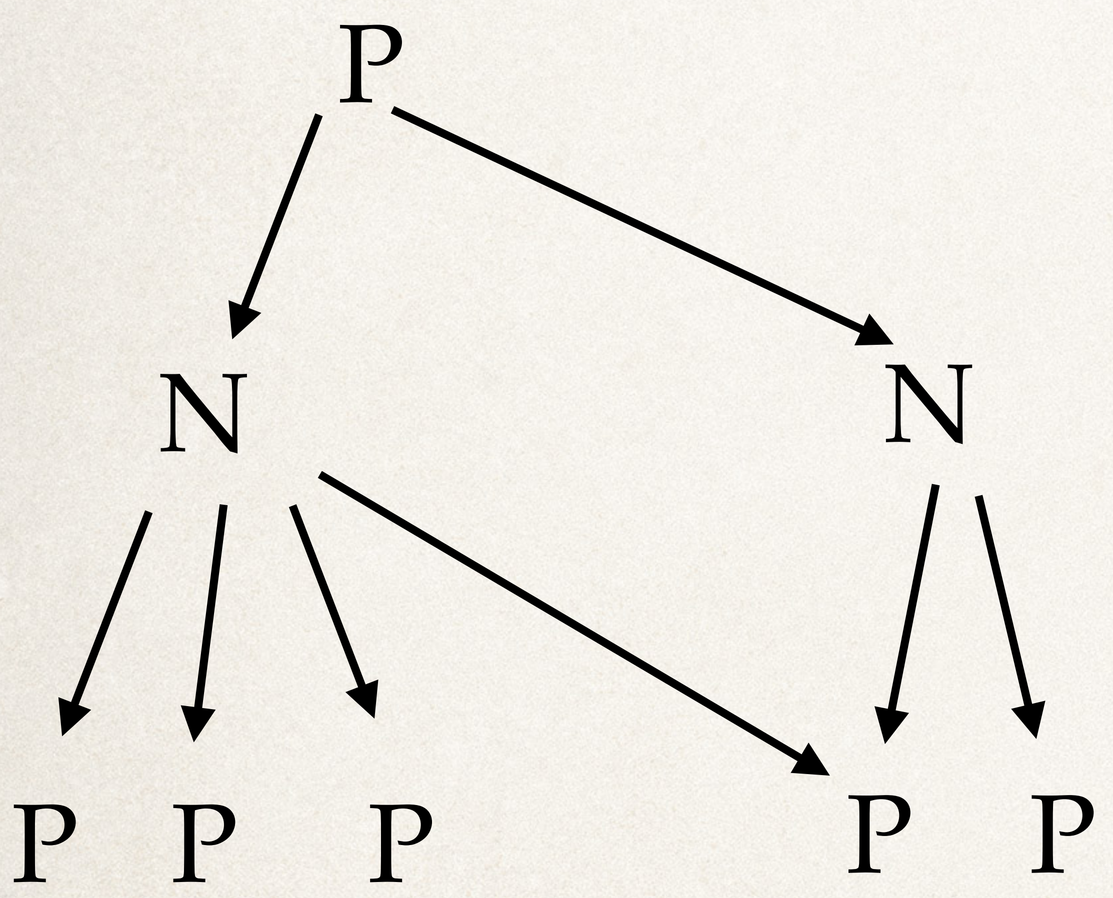




定義

- すべてのterminal positionはP-位置
- 局面がN-位置のとき，一手で動かせる局面のうちP-位置であるものが存在
- 局面がP-位置のとき，一手で動かせる局面はすべてN-位置

性質



- terminal positionを見たら負け
- 局面がN-位置のとき，**次**の局面の選択肢-集合に，対面したら必敗の局面が存在．うまい動きをして次はその必敗の局面を相手におしつけられる
- 局面がP-位置のとき，**次**の局面の選択肢-集合は，すべて対面したら必勝の局面のみ．必勝の局面にどうあがこうとしても動かざるを得ない

ある局面がP-位置かN-位置かどうかの判断法

- 局面に対してbooleanなり0/1で管理してメモ化再帰すればわかる.

練習：再帰でN-位置，P-位置を求めてみる

- 二人の目の前にはコインの山がひとつある．コインの個数は13枚．二人で順番に1~3枚のコインを取る．最後にコインをとりきったら勝ち（ルールは最後に取れなくなったら負け）．

- 二人が最適な戦略を取るときコインの枚数に対してN-位置かP-位置か求める

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P														

- コインが０枚の時はP-位置　∵一手でいける状態が存在しない（terminal position）
- コインが1枚のときはN位置　∵一手でいける局面∈{0} の中にP-位置が存在
- コインが2枚のときはN位置　∵一手でいける局面∈{0, 1} の中にP-位置が存在
- コインが3枚のときはN位置　∵一手でいける局面∈{0, 1, 2} の中にP-位置が存在

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P	P	N	N	N	?	?	?	?	?	?	?	?	?	?

- コインが4枚のときはP位置　∵一手でいける局面∈{1, 2, 3} はすべてN-位置
- コインが5枚のときはN位置　∵一手でいける局面∈{2, 3, 4} の中にP-位置が存在

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P	P	N	N	N	P	N	?	?	?	?	?	?	?	?

- 以下同様の議論をくりかえして次の表を得る

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P	P	N	N	N	P	N	N	N	P	N	N	N	P	N

- 必勝かどうかと対応している

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
勝負	負	勝	勝	勝	負	勝	勝	勝	負	勝	勝	勝	負	勝

例題：再帰でN-位置，P-位置を求める

出典：EDPC K問題

- 二人でK枚のコイン山ひとつからコインを取っていく．
- とっていいコインの枚数は自然数の集合$A = \{a_1, a_2, \ldots, a_N\}$の元に限る
- 行動できなくなったら負け二人が最適な戦略を取るときどちらが勝つか？
- 制約）$1 \leq N \leq 100,\ 1 \leq K \leq 10^5,\ 1 \leq a_1 < a_2 < \ldots \leq K$
- 例）$N = 2,\ K = 4,\ A = \{2, 3\}$のとき先手が3枚取ると後手は操作できなくなり先手必勝
- 例）$N = 2,\ K = 5,\ A = \{2, 3\}$のとき先手が3枚（２枚）取ると後手は２枚（３枚）取って後手必勝

K枚の山をこれから取るときN-位置？P-位置？

- 一種のDPと考えられる．$dp[x] :=$ x枚のコインを目の前にしたときの位置とおく
- x枚のコインから取ろうとしても$x < a_1$なら遷移してくることができずterminal positionなので $dp[x] =$ P
- コインがx枚のとき$x - a_1, x - a_2, \ldots, x - a_N$の中にP-位置があればN-位置，全部N-位置ならP-位置

コイン	0	…	a_1 - 1	…	x-a_N	…	x-a_2	…	x-a_1	…	x	…	K
N/P	P	…	P	…	N/P	…	N/P	…	N/P	…	?	…	?

- 実装. $O(NK)$時間.

```
#include <iostream>
#include <string.h>

//入力
int N, K;
int a[1000010];

int dp[100010]; //dp[x] := P-位置/N-位置 <-> 0/1

//x枚のコインから取っていく局面はP-位置かN-位置かを再起的に求める
int DP(int x){
    if(dp[x] != -1) return dp[x];

    //次の局面 x - a_i のなかにP-位置が存在すればxはN-位置
    for(int i = 0; i < N; i++){
        if(x - a[i] >= 0 && DP(x - a[i]) == 0) return dp[x] = 1;
    }

    //次の局面 x - a_i がすべてN-位置ならxはP-位置
    return dp[x] = 0;
}

void solve(){
    memset(dp, -1, sizeof(dp)); //初期化

    if(DP(K) == 1) printf("First\n"); //K枚で対面してN-位置なら先手必勝
    else printf("Second\n");            //K枚で対面してP-位置なら後手必勝

    return;
}
```

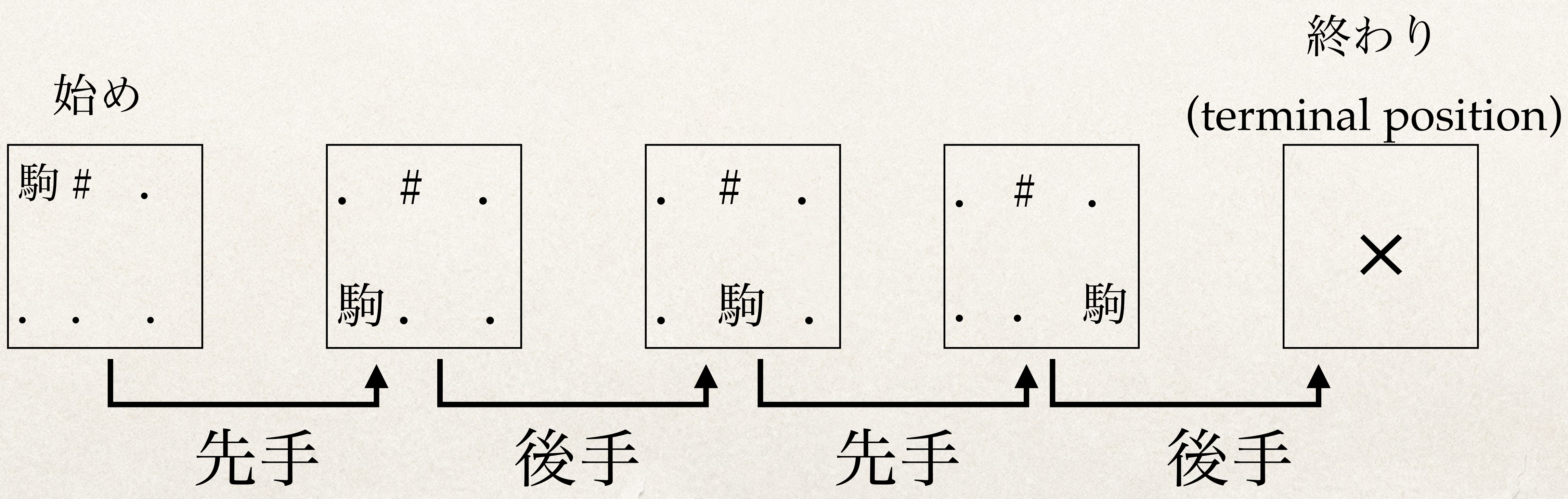
例題：再帰

ARC 038 B問題

- 二人でH行W列のマス目Sで駒を１つ使ったゲームをする．マス目上には障害物がある
  $(1 \leqq H \leqq 100,\ 1 \leqq W \leqq 100)$
- はじめ(1, 1)に駒があって二人は駒を毎回→，↓，↘のいずれかにひとマス分だけ動かす．障害物上と盤外には動かせない．ルールは動かせなくなったら負け
- 二人が最適に行動するとき先手と後手どちらが勝つか？

例）$H = 2,\ W = 3,\ S =$

```
. # .
. . .
```



先手が最適に動き後手を詰ませられる
よって先手必勝・・・（答え）

方針

(x, y)

$$
\begin{tikzcd}
N \arrow[r] \arrow[d] \arrow[dr] & N \\
P & P
\end{tikzcd}
$$

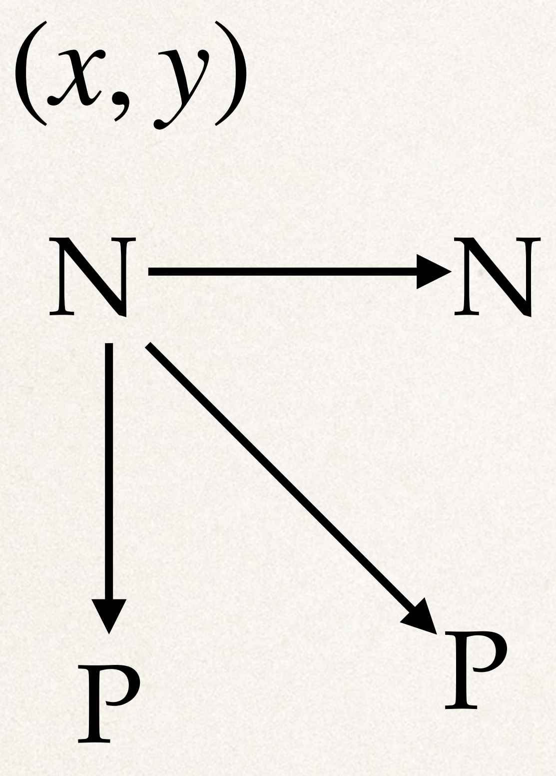
- $(x,\ y)$がN-位置のとき$(x+1,\ y),\ (x,\ y+1),\ (x+1,\ y+1)$の中にP-位置が存在

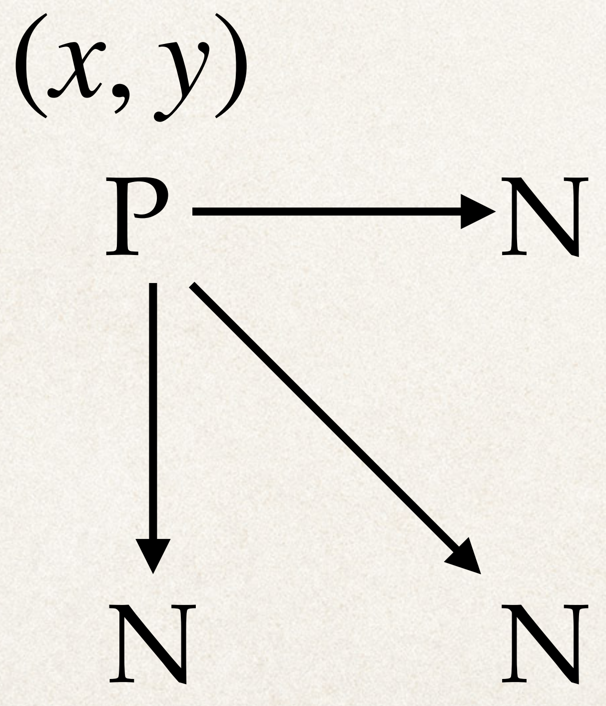
(x, y)

$$
\begin{tikzcd}
P \arrow[r] \arrow[d] \arrow[dr] & N \\
N & N
\end{tikzcd}
$$

- $(x,\ y)$がP-位置のとき$(x+1,\ y),\ (x,\ y+1),\ (x+1,\ y+1)$はすべてN-位置

実装. $O(HW)$時間.






```cpp
#include <iostream>
#include <string.h>

//入力
int H, W;
char S[110][110];

int memo[110][110];

//position(x, y) = (x, y)に駒があるときP-位置/N-位置かを0/1で再帰で得る
int position(int x, int y){
    if(memo[x][y] != -1) return memo[x][y];     //メモ化してあるならその値を返しておわり
    if(S[x][y] == '#')   return memo[x][y] = -1;  //障害物があるなら位置が定義されない


    //次の局面にP-位置が存在したら今の局面はN-位置
    if(x + 1 <= H && y     <= W && position(x + 1, y)         == 0) return memo[x][y] = 1;
    if(x     <= H && y + 1 <= W && position(x, y + 1)         == 0) return memo[x][y] = 1;
    if(x + 1 <= H && y + 1 <= W && position(x + 1, y + 1)     == 0) return memo[x][y] = 1;

    //次の局面がすべてN-位置なら今の局面はP-位置
    return memo[x][y] = 0;
}

void solve(){
    memset(memo, -1, sizeof(memo));

    if(position(1, 1) == 1) printf("First\n"); //(1, 1)がN-位置だったら先手必勝
    else printf("Second\n");                   //(1, 1)がP-位置だったら後手必勝

    return;
}
```

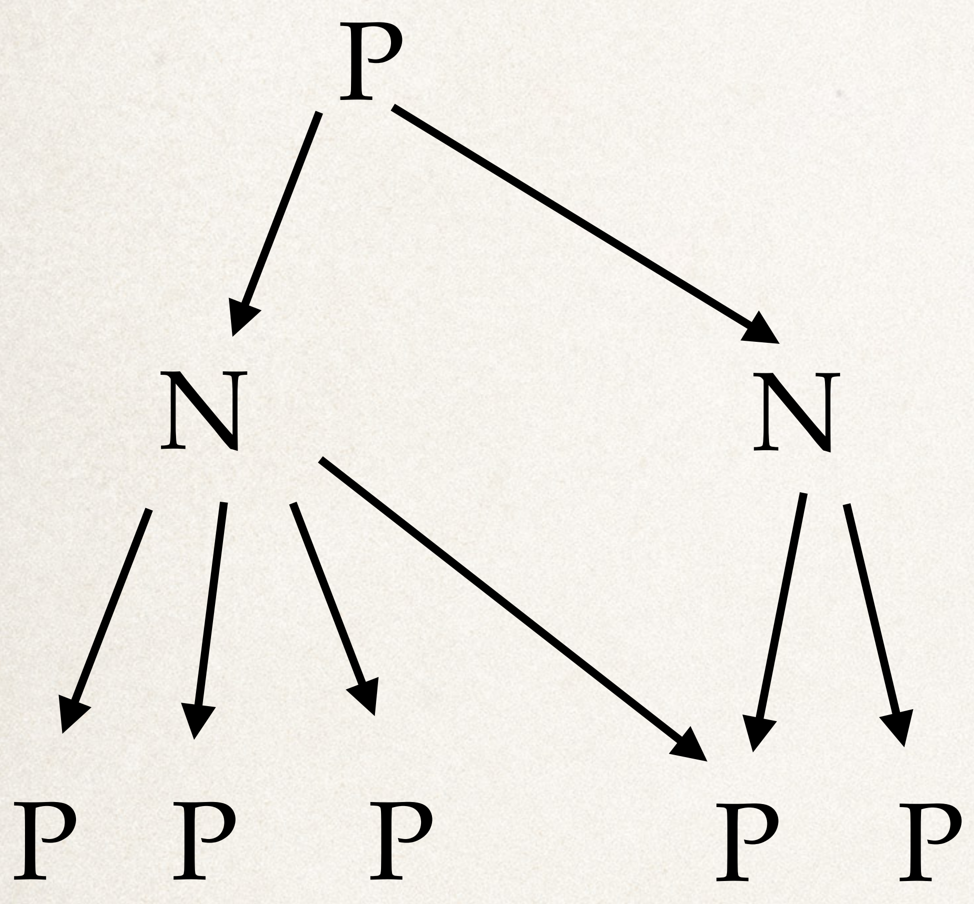
1.2.2項　再帰による決め打ち方の問題点

- ある局面がN-位置かP-位置かどうかの愚直な判断は再帰的に求めることができるが，ゲームの局面数が大きいと計算が大変.

- 例えばN個の山でプレイするNimははじめ各山にx_i個の石が積まれているとして，先手必勝かの再帰での判断は$O(x_1 \times x_2 \times \cdots \times x_N)$.

- そこでNimに帰着させたりゲームの和（後述）と見抜いて，ゲームの局面のGrundy数という値を解析することで先手必勝か後手必勝かを高速に判断できる（↑のNimだと$O(N)$でわかる），不偏ゲームに限るが.

1章のまとめ



- N-位置，P-位置の定義：

  すべてのterminal positionはP-位置

  ゲームの状態がN-位置のとき，１手で動ける状態のなかにP-位置が**存在**

  ゲームの状態がP-位置のとき，１手で動ける状態は**すべて**N-位置

- 局面xが先手必勝$\Longleftrightarrow$$x$がN-位置，後手必勝$\Longleftrightarrow$$x$がP-位置

- 位置は再帰で分かるがゲームの状態数が膨大なとき現実的に不可能

第２章：効率的に先手必勝を判断するアルゴリズム

1章　組み合わせゲームの基礎とN-位置，P-位置

- 1.1節　ゲームの分類とルール
- 1.2節　N-位置，P-位置

2章：効率的に先手必勝を判断するアルゴリズム

- **2.1節　NimとNim和**
- 2.2節　Grundy数とゲームの和

2.1.1項　Nim

問題）コインの山がN個与えられる．各山のコインの枚数はx_iである．

二人でコインのある山を一つだけ選び一枚以上のコインを好きなだけとるという動作を繰り返す.

最後にコインがとれないときが負け．先手必勝かどうか$O(N)$時間で判断せよ．

- 次に説明するNim和を使うとカンタンに判断できる

2.1.2項　Nim和の定義と性質

- 非不整数a, bに対してNim和$a \oplus b$を次のように定義する

a, bを２進数で書いたときに，各桁の繰り上がりナシ足し算

- 例）$5 \oplus 3 = 6$

$$\begin{array}{r} 101_{(2)} \\ \oplus\ \ 11_{(2)} \\ \hline 110_{(2)} \end{array}$$

$11 \oplus 14 = 5$

$$\begin{array}{r} 1011_{(2)} \\ \oplus 1110_{(2)} \\ \hline 101_{(2)} \end{array}$$

- Nim和は各桁ごとの排他的論理和（xor）と同じ

性質

- $a \oplus b = b \oplus a$
- $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- $a \oplus 0 = a$
- $a \oplus a = 0$

(再掲)

- 問題）石の山がN個与えられる．各山の石の個数はx_iである．二人で，石のある山を一つだけ選び一個以上の石を好きなだけとるというプレイを繰り返す．最後に石がとれないときが負け．先手必勝かどうか$O(N)$時間で判断せよ．

2.1.3項　Nim和とBoutonの定理

Boutonの定理：　$x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0$ならN-位置すなわち先手必勝

$x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0$ならP-位置すなわち後手必勝

Boutonの定理の気持ち

- $x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0$のときN-位置．この局面を見たプレイヤーは１手で遷移できる局面の中に$x_1 \oplus \cdots \oplus x_i \oplus \cdots \oplus x_N = 0$となるものが存在するはずなので，この局面に遷移すればよい．

- $x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0$のときP-位置．この局面を見たプレイヤーは$x_1 \oplus \cdots \oplus x_i \oplus \cdots \oplus x_N \neq 0$となる局面に遷移せざるを得ない

- これを繰り返すとやがて$0 \oplus 0 \oplus \cdots \oplus 0 = 0$となる局面(terminal position)をつくって相手に押し付けることができるので勝ち

※証明

x_1

x_2

x_3

$\oplus$ x_4

$d = 00101010010101$

← → この位置が「1」のx_iが存在

$d = x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0$のとき$d$のうちビットが立っているものがある．そのうち一番左にビットがたっているとき，同じ位置のビットが立っているx_iが存在する．このビットだけ0にして011…1にしてしまう．あとはのこりの011….1を適当に操作して$x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0$とできる． $d = 1$なら簡単に$d = 0$にできる．

x_1

x_2

x_3

$\oplus$ x_4

$d = 00000000000000$

$d = 0$のときx_1を減らしてニム和を0にする$x_1^{'}$にできたと仮定する．つまり

$x_1 \oplus x_2 \oplus \cdots \oplus x_{N-1} \oplus x_N = x_1^{'} \oplus x_2 \oplus \cdots \oplus x_{N-1} \oplus x_N (= 0)$.

この両辺の右から$\oplus x_N$を作用させると

$x_1 \oplus x_2 \oplus \ldots \oplus x_{N-1} = x_1^{'} \oplus x_2 \oplus \ldots \oplus x_{N-1}$となる．

さらに両辺の右から$\oplus x_{N-1}, \oplus x_{N-2}, \ldots, \oplus x_2$,を作用させると$x_1 = x_1^{'}$．しかるに$x_1 > x_1^{'}$であるので矛盾．つまり今の局面において$x_i$のNim和がゼロのとき，どうやってもニム和を 1 以上にした局面にして相手に差し出すしかない．（証明終）

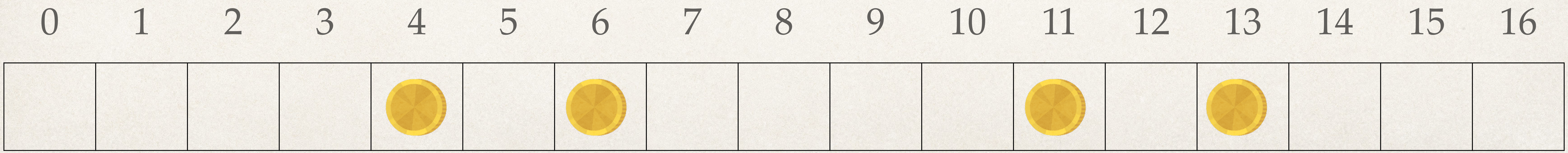
- 不偏ゲームはしばしばNimに帰着できる

2.1.4項　Nimに帰着させる練習

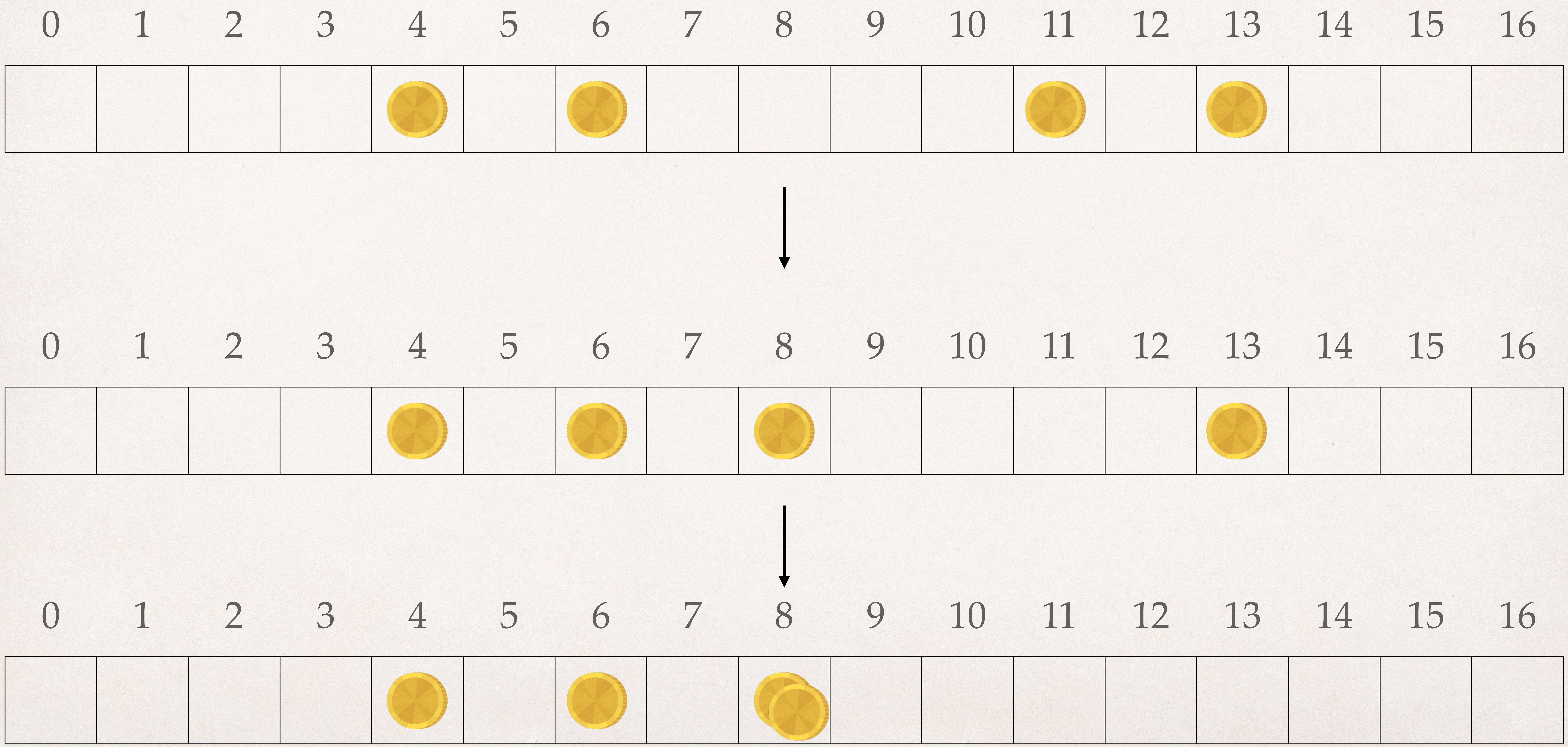
- 図のように一列のマス目列の上でコインをスライドさせるゲームをプレイする．コインは初めN枚あり，初期配置でそれぞれ位置が$x_1, x_2, \ldots. x_N$である．

- ゲームの操作：コインの位置が0ではないものを一枚選んで左にスライドする．

- スライドするコインが無かったらその時点で負け．言い換えれば最後にすべてコインを位置0にしたときに勝ち．



スライド移動の例


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

- 次のようなNimと同じと考えられる
- N山のコインの山で，それぞれの山の枚数はx_i
- したがって$x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0$のとき先手必勝．$x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0$のとき後手必勝

練習：Nimに帰着させる（難）

- 一列のコイン列を１枚か２枚表か裏に返すゲーム．コインはN枚あり，初期配置でこのコインの状態はそれぞれ表かウラである．

1	2	3	4	5	6	7	8	9	10	11	12	13
ウラ	表	ウラ	ウラ	表	ウラ	ウラ	ウラ	表	表	ウラ	表	ウラ

- ゲームの操作：表コイン１枚を表から裏にひっくり返す．望むなら今ひっくり返したコインの左にあるコイン１枚を選んでひっくり返しても良い，このひっくり返すコインは表でも裏でも良い．
- 表にするコインが無かったらその時点で負け．言い換えれば最後にすべてコインを裏にしたときに勝ち．
- ひっくり返すコインのうち一番右は表であることに注意して，先手必勝かNim和の考えで判断せよ．

- 動作例

- ９の位置にあるコインを表から裏にひっくり返し，さらに４の位置にあるコインを裏から表にひっくり返す

1	2	3	4	5	6	7	8	9	10	11	12	13
ウラ	表	ウラ	**ウラ**	表	ウラ	ウラ	ウラ	**表**	表	ウラ	表	ウラ

1	2	3	4	5	6	7	8	9	10	11	12	13
ウラ	表	ウラ	**表**	表	ウラ	ウラ	ウラ	**ウラ**	表	ウラ	表	ウラ

- 実は位置x_iが表のコインならx_i枚のNimの積み重ねと見なすことができ$x_1 \oplus x_2 \oplus x_3 \ldots \oplus x_N \neq 0$なら先手必勝である.

- 位置xの表コインをウラにするとxを０にすることに相当

- 位置xの表コインをウラにし，位置$x'(<x)$のウラのコインを表にするとxをx'に減らすことに相当

- 位置xの表コインをウラにし，位置$x'(<x)$の表コインをウラにするとxをx'に減らすことに相当　以下に説明

実際はxが０になりx'も０になる．Nim和の観点では$\oplus 0 \oplus 0$を作用させるがこれは$\oplus x' \oplus x'$を作用させているのと同じ

これはxをx'にしてx'はそのままにしているのと同じ.

2.1節のまとめ

- N山のNimで各山がx_i枚のとき：

先手必勝$\Longleftrightarrow x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0$, 後手必勝$\Longleftrightarrow x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0$

- ゲームをNimに帰着できることもある.

1章　組み合わせゲームの基礎とN-位置，P-位置

- 1.1節　ゲームの分類とルール
- 1.2節　N-位置，P-位置

2章：効率的に先手必勝を判断するアルゴリズム

- 2.1節　NimとNim和
- 2.2節　**Grundy**数とゲームの和

2.2.1項　不偏ゲームと有向非巡回グラフ（DAG）

- （有限）不偏ゲームはDAGと見なすことができる．ゲームの状態を頂点とおけばループ無しなので．
- ここで有向グラフの数学的定義を書いてみる．

空でないゲームの状態集合（頂点集合）Xと，ある状態（頂点）$x \in X$から次に動かせる状態集合を$F(x)$と書ける写像Fを用いて

$$G = (X, F)$$

と表す．ここに$F(x)$はXの部分集合．

$F(x)$はxからプレイヤーが動かせる状態なので「xのfollower」という．

2.2.2項　Grundy数：すべての不偏ゲームは一山崩しと同じ

- 一山崩しというゲームを考える．コインがz枚つまれた一山を互いに**一つ以上いくらでも取ってよい**．このとき$z \neq 0$なら先手必勝，$z = 0$なら後手必勝.

（コインが積まれていれば全部とることで勝利するなんてつまらないゲームでしょう！）

- 実は$z = 0$のときをちゃんと考えてあげえるのが大事

2.2.2項　Grundy数：すべての不偏ゲームは一山崩しと同じ

- 一山崩しというゲームを考える．コインがz枚つまれた一山を互いに**一つ以上いくらでも取ってよい**．このとき$z \neq 0$なら先手必勝，$z = 0$なら後手必勝.

（コインが積まれていれば全部とることで勝利するなんてつまらないゲームでしょう！）

- 実は$z = 0$のときをちゃんと考えてあげえるのが大事

- 不偏ゲームのある局面をz枚の一山崩しと同じだと見なすときにこの最小値をGrundy数という

一山崩しと見なしてみる

- 二人の目の前にはコインの山がひとつある．コインの個数は13枚．二人で順番に**1~3**枚のコインを取る．最後にコインをとりきったら勝ち（ルールは最後に取れなくなったら負け）．
- 二人が最適な戦略を取るときコインの枚数xに対してz枚の一山崩しとみてみよう

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
z														

一山崩しと見なしてみる

- 二人の目の前にはコインの山がひとつある．コインの個数は13枚．二人で順番に**1~3**枚のコインを取る．最後にコインをとりきったら勝ち（ルールは最後に取れなくなったら負け）．
- 二人が最適な戦略を取るときコインの枚数xに対してz枚の一山崩しとみてみよう
- 考えてみると下の表のようになる．

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
z	0	1	2	3	0	1	2	3	0	1	2	3	0	1

$$z = x (\mathrm{mod}\ 4)$$

- ３枚のコインを見たら，３枚の一山崩しと同じと見なすことができる．
- ５枚のコインの山を見たら，１枚の一山崩しと同じと見なすことができる．実際１枚とれば相手は４枚からコインを取り去り，その後先手は残り全てを取り勝利
- ４枚のコインを見たら負けなので０枚で一山崩しをするのと同じ．見た瞬間に後手必勝が決まっている．
- 44444444444444444444枚のコインを目の前にしても0枚の一山崩しと同じで後手必勝だとすぐわかる

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
z	0	1	2	3	0	1	2	3	0	1	2	3	0	1

- 実はすべての不偏ゲームはz枚１山の一山崩しと見なすことができる

- 例えば３山のNimだろうが，二次元グリッドで駒を動かすゲームだろうが一山崩しと同じと見なすことができる．

- 以上の背景から

$$g\left(\text{複雑なゲームの局面}\right) = z \ \text{（整数）}$$

となる関数gがあれば便利．　この関数をSprague-Grundy関数という．

2.2.3項　Sprague-Grundy関数—すべての不偏ゲーム一山崩しと同じ

- ゲームの状態$x \in X$に対してSprague-Grundy関数$g : X \to \mathbb{Z}$を次のように再帰的に定義する(Xはゲームの局面の集合).

$$g(x) = \mathrm{mex}\{g(y) : y \in F(x)\}$$

あるxに対する$g(x)$をSprague-Grundy値とかGrundy数などという

次ページでmexと$F(x)$について説明

- ゲームの状態xに対して，$F(x)$は１手で動かせる状態の**集合**（１手で動かせる状態は「xのfollower」）．$F(x)$が空集合のときxはterminal position

例）一山のコイン取り去りゲームで，毎回1~3枚取っていくルールのときコインの枚数をxとおくと

$$F(13) = \{10,\ 11,\ 12\},\ F(3) = \{0,\ 1,\ 2\},\ F(1) = \{0\},\ F(0) = \{\}$$

- mex S：minimal excludantの略．非負整数の**集合**Sに含まれないのなかで，最小の非負整数

例）
$$\mathrm{mex}\{0,\ 1,\ 2,\ 3\} = 4,\ \mathrm{mex}\{0,\ 1,\ 2,\ 3,\ 4\} = 5,\ \mathrm{mex}\{0,\ 1,\ 2,\ 4\} = 3,$$
$$\mathrm{mex}\{1,\ 2,\ 3\} = 0,\ \mathrm{mex}\{1,\ 1,\ 2,\ 3\} = 0,\ \mathrm{mex}\{\} = 0,$$

再掲）ゲームの状態$x \in X$に対してSprague-Grundy関数 $g : X \to \mathbb{Z}$ を次のように再帰的に定義する　(Xはゲームの局面の集合).

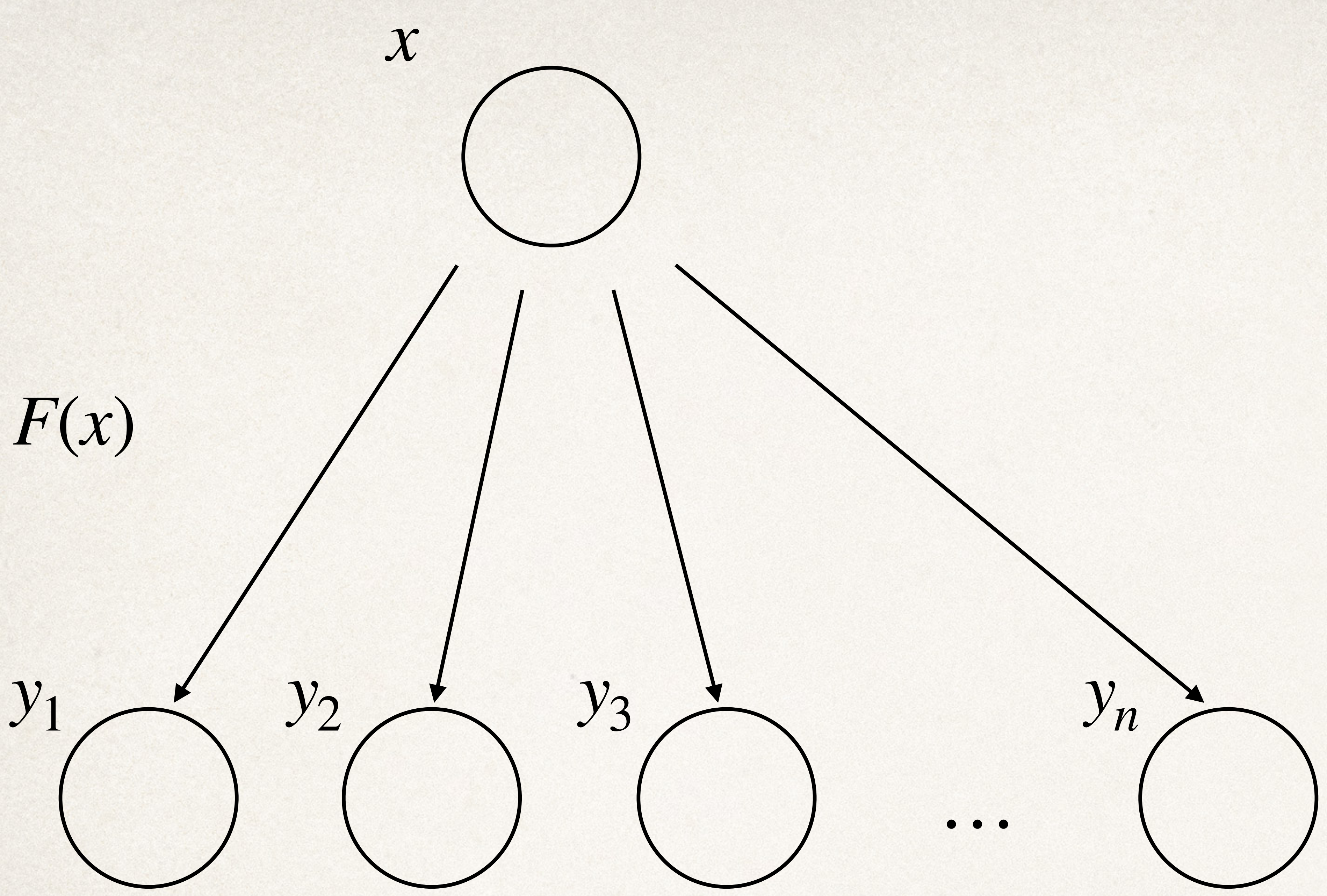
$$g(x) = \mathrm{mex}\{g(y) : y \in F(x)\}$$

次に動かせる局面がn個，いわば$F(x) = \{y_1, y_2, y_3, \ldots, y_n\}$のとき

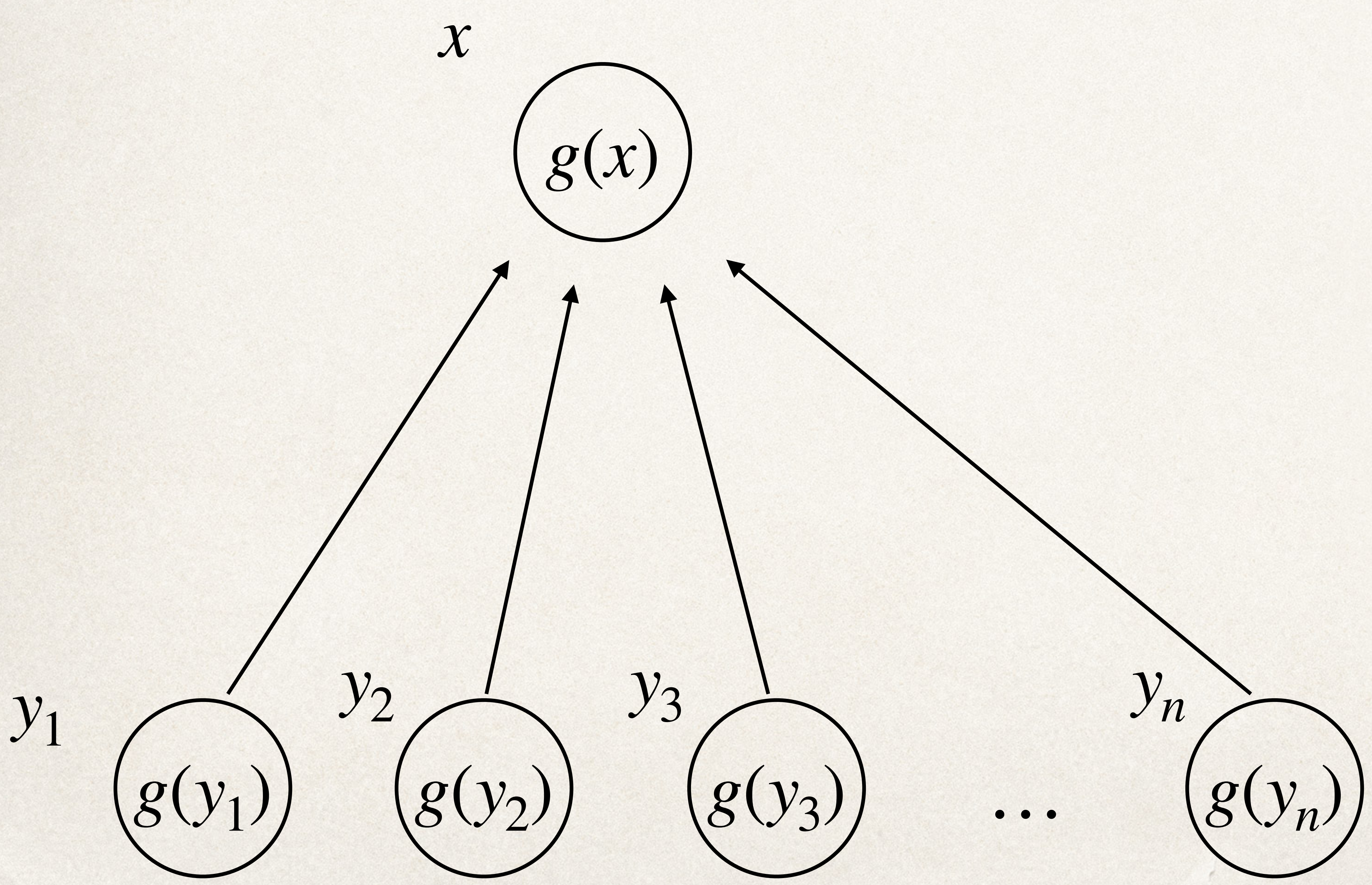
$$g(x) = \mathrm{mex}\{g(y_1), g(y_2), g(y_3), \ldots, g(y_n)\}$$

あるxに対する$g(x)$をSprague-Grundy値とかGrundy数などという

式の解釈：$\underbrace{\text{今の局面のGrundy数}}_{g(x)}$は，$\underbrace{\text{１手で動かせる全ての局面に対するGrundy数の集合}}_{\{g(y_1), g(y_2), g(y_3), \ldots, g(y_n)\}}$ $\underbrace{\text{に含まれない最小の非負整数}}_{\mathrm{mex}}$である



- 状態xからの遷移するイメージ



- xのGrundy数$g(x)$を得るイメージ

- Grundy数は定義そのものが再帰的で初めは捉えづらい．次ページから具体的に問題を解いてイメージを捉えてみる．

練習：Grundy数を求めてみる

- 二人の目の前にはコインの山がひとつある．コインの個数は13枚．二人で順番に1~3枚のコインを取る．最後にコインをとりきったら勝ち（ルールは最後に取れなくなったら負け）．

- 二人が最適な戦略を取るときコインの枚数xに対してGrundy数を求めてみる

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)														

$g(x) = \mathrm{mex}\{g(y) : y \in F(x)\}$

- コインが０枚の時は$F(0) = \{\}$だから$g(0) = \mathrm{mex}\{\} = 0$
- コインが1枚のときは$F(1) = \{0\}$だから$g(1) = \mathrm{mex}\{g(0)\} = \mathrm{mex}\{0\} = 1$
- コインが2枚のときは$F(2) = \{0, 1\}$だから$g(3) = \mathrm{mex}\{g(0), g(1)\} = \mathrm{mex}\{0, 1\} = 2$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2											

$g(x) = \mathrm{mex}\{g(y) : y \in F(x)\}$

- コインが3枚のときは$F(3) = \{0, 1, 2\}$だから$g(3) = \mathrm{mex}\{g(0), g(1), g(2)\} = \mathrm{mex}\{0, 1, 2\} = 3$
- コインが4枚のときは$F(4) = \{1, 2, 3\}$だから$g(4) = \mathrm{mex}\{g(1), g(2), g(3)\} = \mathrm{mex}\{1, 2, 3\} = 0$
- コインが5枚のときは$F(5) = \{2, 3, 4\}$だから$g(3) = \mathrm{mex}\{g(2), g(3), g(4)\} = \mathrm{mex}\{2, 3, 0\} = 1$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2	3	0	1								

- 以下同様の議論をくりかえして次の表を得る．$g(x) = x(\mathrm{mod}4)$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2	3	0	1	2	3	0	1	2	3	0	1

- 以下同様の議論をくりかえして次の表を得る．$g(x) = x(\mathrm{mod}4)$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2	3	0	1	2	3	0	1	2	3	0	1

- N-位置/P-位置と対応している．$g(x) \neq 0$ならN-位置，$g(x) = 0$ならP-位置

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P	P	N	N	N	P	N	N	N	P	N	N	N	P	N

- 以下同様の議論をくりかえして次の表を得る．$g(x) = x(\mathrm{mod}4)$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13
g(x)	0	1	2	3	0	1	2	3	0	1	2	3	0	1

- N-位置/P-位置と対応している．$g(x) \neq 0$ならN-位置，$g(x) = 0$ならP-位置

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
N/P	P	N	N	N	P	N	N	N	P	N	N	N	P	N

- 必勝かどうかと対応している．$g(x) \neq 0$なら必勝，$g(x) = 0$なら必敗

Grundy数 $\neq 0$なら必勝

コイン	0	1	2	3	4	5	6	7	8	9	10	11	12	13
勝負	負	勝	勝	勝	負	勝	勝	勝	負	勝	勝	勝	負	勝

Grundy数を求める練習

- 二人の目の前にはコインの山がひとつある．コインの個数はx枚．二人で順番に$S = \{1,3,4\}$の元ひとつを選んでその枚数のコインを取る．最後にコインをとりきったら勝ち（ルールは最後に取れなくなったら負け）．

- 二人が最適な戦略を取るときコインの枚数xに対してGrundy数を求めてみる

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
g(x)															

$\boxed{S = \{1,3,4\}}$

- terminal positionは$x = 0$なので$g(0) = 0$
- $x = 1$は$x = 0$にしか動けないので$g(1) = \mathrm{mex}\{g(0)\} = \mathrm{mex}\{0\} = 1$
- $x = 2$は$x = 1$にしか動けないので$g(2) = \mathrm{mex}\{g(1)\} = \mathrm{mex}\{1\} = 0$ ２枚でスタートしたら先手は負け
- $x = 3$は$x = 0,2$に動けるので$g(3) = \mathrm{mex}\{g(0), g(2)\} = \mathrm{mex}\{0,0\} = 1$
- $x = 4$は$x = 0,1,3$に動けるので$g(4) = \mathrm{mex}\{g(0), g(1), g(3)\} = \mathrm{mex}\{0,1,1\} = 2$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
g(x)	0	1	0	1	2										

- これを続けると次の表を得る

$\boxed{S = \{1,3,4\}}$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
g(x)	0	1	0	1	2	3	2	0	1	0	1	2	3	2	0

- よって次のようになる

$$g(x) = \begin{cases} 0 & (x(\mathrm{mod}7) = 0,2) \\ 1 & (x(\mathrm{mod}7) = 1,3) \\ 2 & (x(\mathrm{mod}7) = 4,6) \\ 3 & (x(\mathrm{mod}7) = 5) \end{cases}$$

- ここまで具体的に書けなくても，実際は与えられたxに対して再帰で求めれば十分だと思う（$O(x)$）

- 実際にGrundy数を再帰的に求めてみる

```cpp
#include <iostream>
#include <set>

const int X = 14;
const std::set<int> SubstractionSet = {1, 3, 4};

int memo[20]; //メモ化 memo[x] = g(x)

int g(int x){
    if(memo[x] != -1) return memo[x];

    std::set<int> S;
    for(auto e: SubstractionSet) if(x - e >= 0) S.insert(g(x - e));

    /*mex S*/
    int res = 0;
    while(S.count(res)) res++;
    return memo[x] = res;
}

int main(){
    for(int x = 0; x < 20; x++) memo[x] = -1;
    for(int x = 0; x <= X; x++) {
        printf("g(%d) = %d\n", x, g(x));
    }
    return 0;
}
```

- 出力

```
g(0) = 0
g(1) = 1
g(2) = 0
g(3) = 1
g(4) = 2
g(5) = 3
g(6) = 2
g(7) = 0
g(8) = 1
g(9) = 0
g(10) = 1
g(11) = 2
g(12) = 3
g(13) = 2
g(14) = 0
```

補足

- 取っていいコインの枚数の選択肢が毎回固定されているとき（今回は$S = \{1,3,4\}$）のとき実はSG関数は周期的

- 今の局面のGrundy数が$z(\neq 0)$のとき$0,1,\ldots,z-1$に移動する手がある．0にしてしまえば勝ち．まさしくz枚の一山崩し．

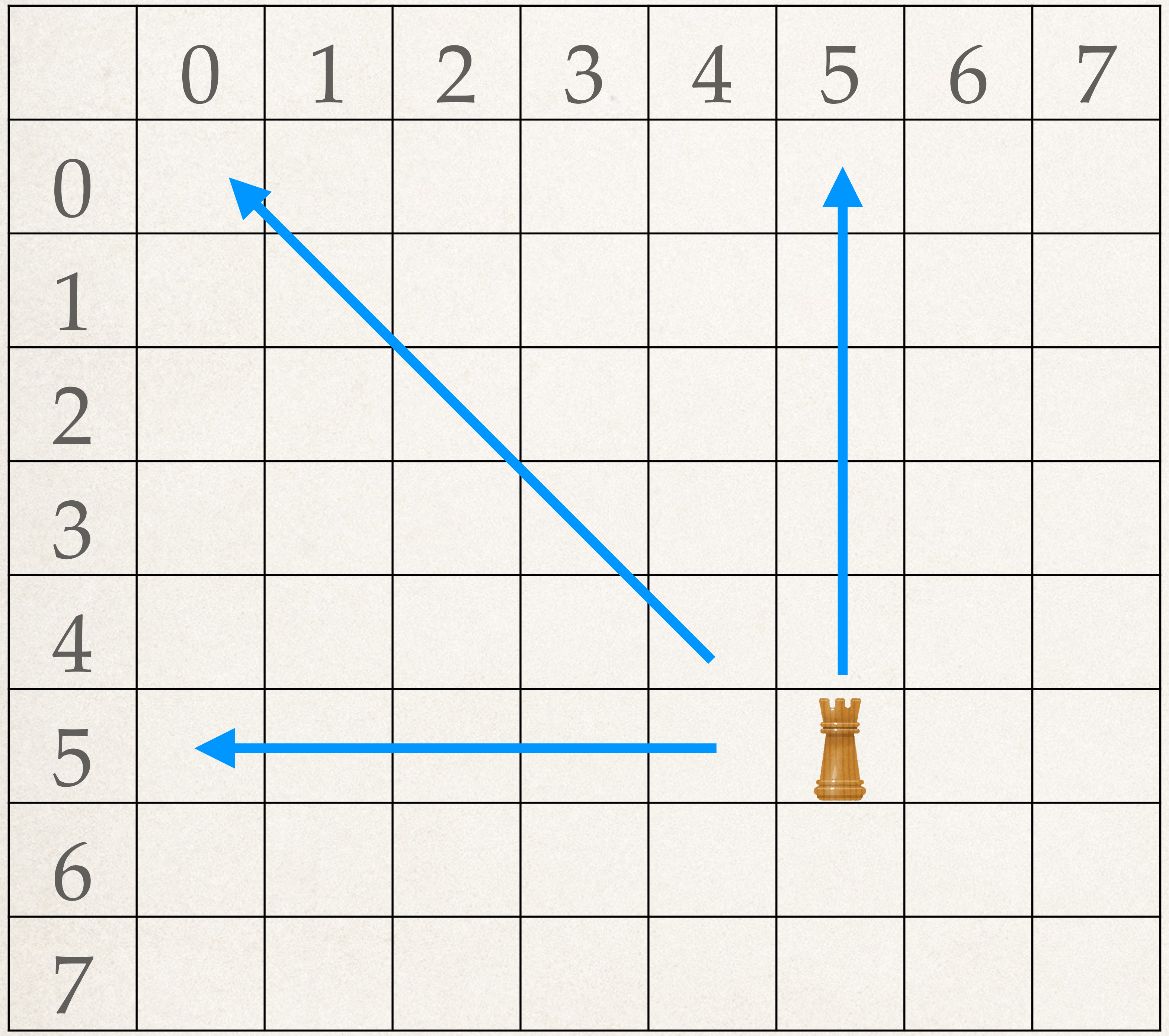
- $z = 0$のときGrundy数が0でない局面に移動せねばならず負け．

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
g(x)	0	1	0	1	2	3	2	0	1	0	1	2	3	2	0

↑５枚の山と対面したら２枚にすれば勝ち．

こうすれば$g(2) = 0$

Grundy数を求める練習



- 8×8のマス目上で駒１つを互いに動かすゲームをする．
- (x, y)から(0,0)に近づくように駒を動かす．(0,0)になったら動かせない．
- 許される動かし方: i)水平方向　ii)垂直方向　iii)対角線上 のいずれかで(0,0)に近づく方向のみ
- (x, y)に駒があるときのGrundy数を求めてみる

- Grundy数の定義より

$$g(x,y) = \mathrm{mex}\{g(x-1,y),\ g(x-2,y),\ \ldots\ g(0,y),\\ g(x,y-1),\ g(x,y-2),\ \ldots\ g(x,0),\\ g(x-1,y-1),\ g(x-2,y-2),\ \ldots\}$$

- この再帰関数を実行して次の表を得る

```
#include <iostream>
#include <set>

const int X = 7, Y = 7;
int memo[X + 1][Y + 1]; //実際は−1で初期化する

int g(int x, int y){
    if(memo[x][y] != -1) return memo[x][y];

    std::set<int> S;
    for(int dx = -1; x + dx >= 0; dx--) S.insert(g(x + dx, y));
    for(int dy = -1; y + dy >= 0; dy--) S.insert(g(x, y + dy));
    for(int d = -1; x + d >= 0 && y + d >= 0; d--) S.insert(g(x + d, y + d));

    int res = 0;
    while(S.count(res)) res++;
    return memo[x][y] = res;
}
```

g(x, y)

x\y	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	0	4	5	3	7	8
2	2	0	1	5	3	4	8	6
3	3	4	5	6	2	0	1	9
4	4	5	3	2	7	6	9	0
5	5	3	4	0	6	8	10	1
6	6	7	8	1	9	10	3	4
7	7	8	6	9	0	1	4	5

Grundy数を求める練習

- 一山x枚のコインを取るゲームをする．ただし取っていいコインの枚数は多くてもその時点のコインの枚数の半分まで.

例）６枚の山から取れるコインの枚数は1,2,3枚．7枚のとき1,2,3枚まで取れる

- Grundy数を考えてみる．

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
g(x)																	

- 0,1枚のときこれ以上コインを取れない(terminal position)．従って$g(0) = g(1) = 0$

- 2枚のとき1枚取って1枚の山にできる（$F(2) = \{1\}$）ので$g(2) = \mathrm{mex}\{g(1)\} = \mathrm{mex}\{0\} = 1$

- $x = 3$のとき$F(3) = \{2\}$なので$g(3) = \mathrm{mex}\{g(2)\} = \mathrm{mex}\{1\} = 0$．実際，自分が１枚取ったら山は２枚になり相手が１枚取り山は１枚（terminal position）になり自分の手番になる（負け）．

- $x = 4$のとき$F(4) = \{2,3\}$なので$g(4) = \mathrm{mex}\{g(2), g(3)\} = \mathrm{mex}\{0,1\} = 2$

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
g(x)	0	0	1	0													

- 以下，同様にして次の表を得る

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
g(x)	0	0	1	0	2	1	3	0	4	2	5	1	6	3	7	0	8

- $$g(x) = \begin{cases} \dfrac{x}{2} & (\text{if } x \text{ is even}) \\ g\left(\dfrac{x-1}{2}\right) & (\text{if } x \text{ is odd}) \end{cases}$$

練習：Grundy数を求める訓練

- 1山の一山崩しを考える．山にx枚のコインがあるとき$g(x)=?$

※一山崩しは一枚以上いくらでもとって良い

- Grundy数は不偏ゲームをz枚の一山崩しと見なしたとき，最小のzのこと．
  従って今回は定義より$g(x) = x$　・・・（答え）

- （別解）具体的にGrundy数を書いてみる．

$g(0) = \mathrm{mex}\{\} = 0,\ g(1) = \mathrm{mex}\{g(0)\} = \mathrm{mex}\{0\} = 1,$

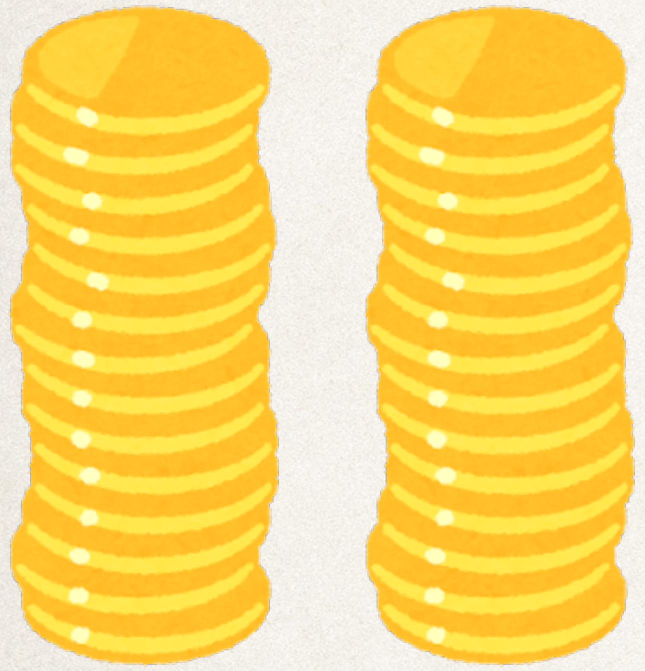
$g(2) = \mathrm{mex}\{g(0), g(1)\} = \mathrm{mex}\{0,1\} = 2,$

よって$g(x) = x$

これは数学的帰納法で示せる．

Grundy数を求める練習

- 2山のNimを考える．毎ターンプレイヤーは山を一つ選んで 1 枚以上のコインを取り去る．山にx_1, x_2枚のコインがあるとき $g(x_1, x_2) = ?$



※一山崩しは1枚以上いくらでもとって良い

Grundy数の定義より

$$g(x_1, x_2) = \mathrm{mex}\{g(x_1, y_2), g(y_1, x_2) : 0 \leq y_1 < x_1,\ 0 \leq y_2 < x_2\}$$

$$g(x_1, x_2) = \text{mex}\{g(x_1, y_2), g(y_1, x_2) : 0 \le y_1 < x_1,\ 0 \le y_2 < x_2\}$$

g

x_1/x_2	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	0	3	2	5	4	7	6	9	8
2	2	3	0	1	6	7	4	5	10	11
3	3	2	1	0	7	6	5	4	11	10
4	4	5	6	7	0	1	2	3	12	13
5	5	4	7	6	1	0	3	2	13	12
6	6	7	4	5	2	3	0	1	14	15
7	7	6	5	4	3	2	1	0	15	14
8	8	9	10	11	12	13	14	15	0	1
9	9	8	11	10	13	12	15	14	1	0

$$g(x_1, x_2) = \mathrm{mex}\{g(x_1, y_2), g(y_1, x_2) : 0 \le y_1 < x_1,\ 0 \le y_2 < x_2\}$$

g

x_1/x_2	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	0	3	2	5	4	7	6	9	8
2	2	3	0	1	6	7	4	5	10	11
3	3	2	1	0	7	6	5	4	11	10
4	4	5	6	7	0	1	2	3	12	13
5	5	4	7	6	1	0	3	2	13	12
6	6	7	4	5	2	3	0	1	14	15
7	7	6	5	4	3	2	1	0	15	14
8	8	9	10	11	12	13	14	15	0	1
9	9	8	11	10	13	12	15	14	1	0

実は

$$g(x_1, x_2) = x_1 \oplus x_2$$

である（！）

- 小ネタ：Nim和を次式で定義する考えもある.

$$x \oplus y := \mathrm{mex}\{x \oplus b, a \oplus y : 0 \le b < y, 0 \le a < x\}$$

Grundy数を求める練習

- 2つのゲームG_1, G_2を同時にプレイすること考える．毎ターンゲームG_1かG_2のどちらかを選んで局面を動かす.
- 今，ゲームG_1，G_2のGrundy数がそれぞれ$g_1(x_1), g_2(x_2)$であることがわかっているとする.
- このとき，同時プレイするときのGrundy数　$g(x_1, x_2) = ?$

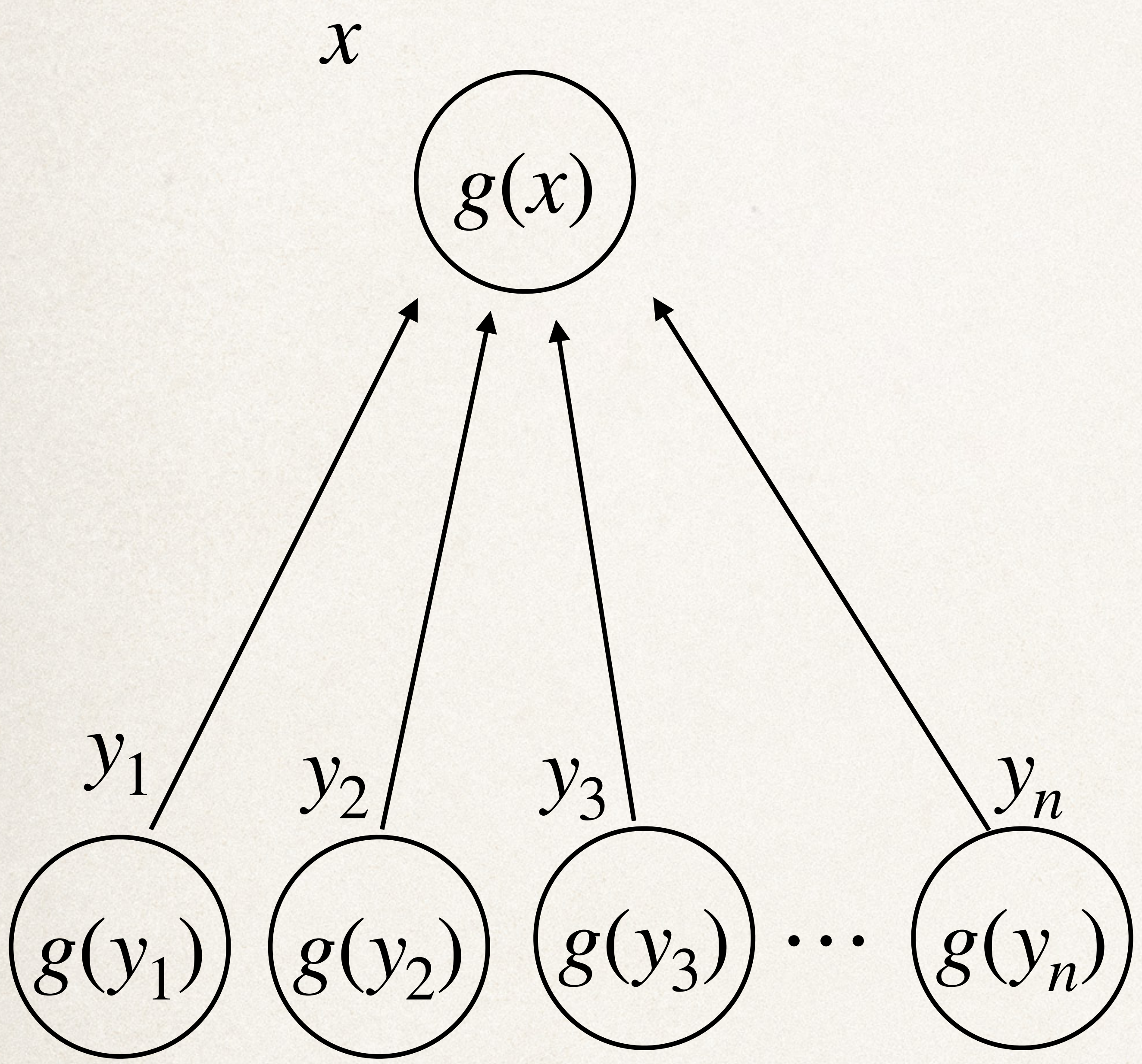
　コインの枚数が$g_1(x_1), g_2(x_2)$枚である二山のNimをしているのと同じ．ゲームG_1を選んで局面動かすときコインの枚数を$0 \sim g_1(x_1) - 1$にできる．

すると２山Nimの問題より

$$g(x_1, x_2) = g_1(x_1) \oplus g_2(x_2)$$

である．

ここまでの2章まとめ



- N山のNimで各山がx_i枚のとき：

$$x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0 \Longleftrightarrow \text{先手必勝},$$
$$x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0 \Longleftrightarrow \text{後手必勝}$$

- ゲームをNimに帰着できることもある.

- 不偏ゲーム$G = (X, F)$の局面xについてGrundy数
$g(x) = \mathrm{mex}\{g(y) : F(x)\}$に対して

$$g(x) \neq 0 \Longleftrightarrow \text{先手必勝}, \quad g(x) = 0 \Longleftrightarrow \text{後手必勝}$$

1章　組み合わせゲームの基礎とN-位置，P-位置

- 1.1節　ゲームの分類とルール
- 1.2節　N-位置，P-位置

2章：効率的に先手必勝を判断するアルゴリズム

- 2.1節　NimとNim和
- **2.2節**　Grundy数とゲームの和

2.2.4項　ゲームの和

- ゲームGの局面xから一手で動ける局面の集合を$F(x)$と書くことにする(follower)．このときゲーム$G_1 = (X_1, F_1)$, ゲーム$G_2 = (X_2, F_2)$,．．．．，ゲーム$G_N = (X_N, F_N)$に対してゲームの和$G = G_1 + G_2 + \ldots + G_N$を定義する．

2.2.4項　ゲームの和

- ゲームGの局面xから一手で動ける局面の集合を$F(x)$と書くことにする(follower)．このときゲーム$G_1 = (X_1, F_1)$, ゲーム$G_2 = (X_2, F_2)$, ．．．，ゲーム$G_N = (X_N, F_N)$に対してゲームの和$G = G_1 + G_2 + \ldots + G_N$を定義する．

- ゲームGの頂点の集合Xは直積$X = X_1 \times X_2 \times \ldots \times X_N$である．つまりすべての$x_i \in X_i$に対する$x = (x_1, x_2, \ldots, x_N)$の集合

- ゲームの和Gの頂点$x = (x_1, x_2, \ldots, x_N) \in F(x)$のfollower集合を次のように定める．

$$
\begin{aligned}
F(x_1, x_2, \ldots, x_N) = {} & F_1(x_1) \times \{x_2\} \times \ldots \times \{x_N\} \\
& \cup \{x_1\} \times F_2(x_2) \times \ldots \times \{x_N\} \\
& \cup \ldots \\
& \cup \{x_1\} \times \{x_2\} \times \ldots \times F_N(x_N)
\end{aligned}
$$

2.2.4項　ゲームの和

- ゲームGの局面xから一手で動ける局面の集合を$F(x)$と書くことにする(follower)．このときゲーム$G_1=(X_1,F_1)$, ゲーム$G_2=(X_2,F_2)$，．．．，ゲーム$G_N=(X_N,F_N)$に対してゲームの和$G=G_1+G_2+\ldots+G_N$を定義する．

- ゲームGの頂点の集合Xは直積$X=X_1\times X_2\times\ldots\times X_N$である．つまりすべての$x_i\in X_i$に対する$x=(x_1,x_2,\ldots,x_N)$の集合

- ゲームの和Gの頂点$x=(x_1,x_2,\ldots,x_N)\in F(x)$のfollower集合を次のように定める．

$$
\begin{aligned}
F(x_1,x_2,\ldots,x_N) = & F_1(x_1)\times\{x_2\}\times\ldots\times\{x_N\}\\
& \cup\{x_1\}\times F_2(x_2)\times\ldots\times\{x_N\}\\
& \cup\ldots\\
& \cup\{x_1\}\times\{x_2\}\times\ldots\times F_N(x_N)
\end{aligned}
$$

1つだけゲームG_iを動かすことしか許されていない が必ず一個動かせねばならない　同値　このゲームは和である．

Nimはゲームの和である

- N個のコイン山であるNimはゲームの和である．コインの枚数はそれぞれx_iであるとする．

- $G_i = x_i$枚の一山崩し　とおく．このとき問題文のゲームは$G = G_1 + G_2 + \ldots + G_N$とおくことができる

- ゲームの和とおける理由：Gの状態$x = (x_1, x_2, \ldots x_N)$から動ける状態は，ゲーム$G_i$をひとつだけ選び$x_i$を動かす＆他のゲーム$G_j$ $(j \neq i)$の状態x_jはそのまま

2.2.5項　ゲームの和のGrundy数

$G = G_1 + G_2 + \cdots + G_N$のGrundy数は何だろう？

ゲームG_iのGrundy数が$g_i(x_i)$であるとする.

するとゲームの和$G = G_1 + G_2 + \cdots + G_N$のGrundy数は

$$g(x_1, x_2, \ldots x_N) = g_1(x_1) \oplus g_2(x_2) \oplus \cdots \oplus g_N(x_N)$$

であるという強力な定理がある(Sprague-Grundyの定理).

これは山のコインが$g_1(x_1), g_2(x_2), \ldots, g_N(x_N)$枚のNimと解釈できる

- 証明

ゲーム$G_1 + G_2$のGrundy数は$g_1(x_1) \oplus g_2(x_2)$であることを前提とする.

ゲーム$G_1 + G_2 + G_3$は$g_1(x_1) \oplus g_2(x_2)$枚と$g_3(x_3)$枚の２山崩しと同じだから
Grundy数は$g_1(x_1) \oplus g_2(x_2) \oplus g_3(x_3)$

同じ議論を繰り返すことで不偏ゲーム$G_1 + G_2 + \cdots + G_N$のGrundy数は

$$g_1(x_1) \oplus g_2(x_2) \oplus \cdots \oplus g_N(x_N)$$

である（証明終）

$G = G_1 + G_2 + \ldots + G_N$のGrundy数（再掲）

ゲームG_iのGrundy数が$g_i(x_i)$であるとする．

するとゲームの和$G = G_1 + G_2 + \cdots + G_N$のGrundy数は

$$g(x_1, x_2, \ldots x_N) = g_1(x_1) \oplus g_2(x_2) \oplus \ldots \oplus g_N(x_N)$$

であるという強力な定理がある．

今までは全探索でgを見つけた$(O(x_1 \times x_2 \times \cdots x_N))$が上の定理を使うと独立に求めてNim和を取ることでカンタンに求まる．

さらに$g_1 = g_2 = \cdots = g_N$のとき（全く同じゲームをN個プレイするとき）もっとラク．

ゲームの和のGrundy数を求める練習

出典：HackerRank

- 二人でN個のコイン山から互いに一つ山を選んでコインを減らすゲームをする．コインの枚数は最初h_i枚である．

- コインの山の枚数をx枚からy枚に減らすときyはxの約数であるという条件つき．

例えば$x = 36$なら$y = 1,2,3,4,6,9,12,18$に減らせる

- 減らせなくなったら負け．先手必勝なら「１」，後手必勝なら「２」を出力せよ

- 制約）$1 \leq N \leq 100,\ 1 \leq h_i \leq 10^6$

ゲーム$G_i := h_i$枚のコイン１山崩しでh_iの約数のみに減らせる　とすると問題のゲームは$G = G_1 + G_2 + \cdots + G_N$とおける．

ゲームの和とおける理由はゲームを一つ選んでx_iを減らすが他のゲームのx_jはそのまま，かつ少なくともゲームの局面を動かさねばならないから．

一山h_i枚のときのGrundy数を$g(h_i)$とすると$G = G_1 + G_2 + \cdots + G_N$のGrundy数は

$$g(h_1) \oplus g(h_2) \oplus \cdots \oplus g(h_N)$$

である．

- C++によるコード
- （元の提出サイトだと日本語コメント書くと提出がはじかれる）

$$g(h_1) \oplus g(h_2) \oplus \cdots \oplus g(h_N)$$

```cpp
#include <stdio.h>
#include <string.h>
#include <set>
#include <vector>
#include <algorithm>

int N;

const int MAX_N = 110, MAX_H = 1000010;
int h[MAX_N], memo[MAX_H];

std::vector<int> divisor(int n) {                    /* 自然数nの約数列挙  O(√n) */
    std::vector<int> res;
    for (int i = 1; i * i <= n; i++) {
        if (n % i == 0) {
            res.push_back(i);
            if (i * i != n) res.push_back(n / i);
        }
    }
    std::sort(res.begin(), res.end());
    return res;
}

int g(int h){                                    /* 枚数hのGrundy数を取得 */
    if(memo[h] != -1) return memo[h];
    if(h == 1)           return memo[h] = 0;

    std::vector<int> divisor_list = divisor(h);      /* hの全約数 */

    std::set<int> S;                             /* hの約数dに対してdのGrundy数−集合 */
    for(auto d: divisor_list) if(d != h) {
        S.insert(g(d));
    }

    int res = 0;                                     /* mex */
    while(S.count(res)) res++;
    return memo[h] = res;
}

void solve(int N, int h[]){
    memset(memo, -1, sizeof(memo));

    int nimSum = 0;
    for(int i = 0; i < N; i++) nimSum ^= g(h[i]);

    if(nimSum > 0) printf("1\n");
    else           printf("2\n");

    return;
}
```

ゲームの和のGrundy数を求める練習

出典：AtCoder 典型90問-031

N個の白い石山と青い石山がある．石の個数ははじめW_i, B_i個.

行動：i番目の白と青い石山のペアを選び次の①か②を行う．このときの石の個数を各々w, bとする.

① 選んだ青石にw個の青石を加えて白石を一つ取り去り$w-1$個にする．この選択は$w \geq 1$のときのみ行える.

② 選んだ山の青い石山から1以上$b/2$個以下の整数k個だけ取り去る．この選択は$b \geq 2$のときのみ行える.

行動できなくなったら負け.

制約）$1 \leq N \leq 10^5$, $0 \leq W_i, B_i \leq 50$, $(W_i, B_i) \neq (0, 0)$

ゲーム$G_i :=$ i番目の白い石山と青い石山でルールに基づきプレイするゲーム　とおく

すると問題文のゲームは$G_1 + G_2 + \ldots + G_N$とおける.

なぜならば毎ターンどれかのゲームG_iを選び白い石と青い石(w_i, b_i)を動かすが他のゲームの(w_j, b_j)は放置，かつ局面は必ず進めなければならないから.

ゲーム$G_i :=$ i番目の白い石山と青い石山でルールに基づきプレイするゲーム　とおく

すると問題文のゲームは$G_1 + G_2 + \cdots + G_N$とおける．

なぜならば毎ターンどれかのゲームG_iを選び白い石と青い石(w_i, b_i)を動かすが他のゲームの(w_j, b_j)は放置，かつ局面は必ず進めなければならないから．

よって白い石，青い石がw, b個あるときのGrundy数を$g(w, b)$として

$g(W_1, B_1) \oplus g(W_2, B_2) \oplus \cdots \oplus g(W_N, B_N)$が正なら先手必勝．0なら後手必勝

$$g(w,b) = \mathrm{mex}\{g(w-1, b+w),\ g(w,b-1),\ g(w,b-2),\ \dots\ g(w,b-b/2)\}$$

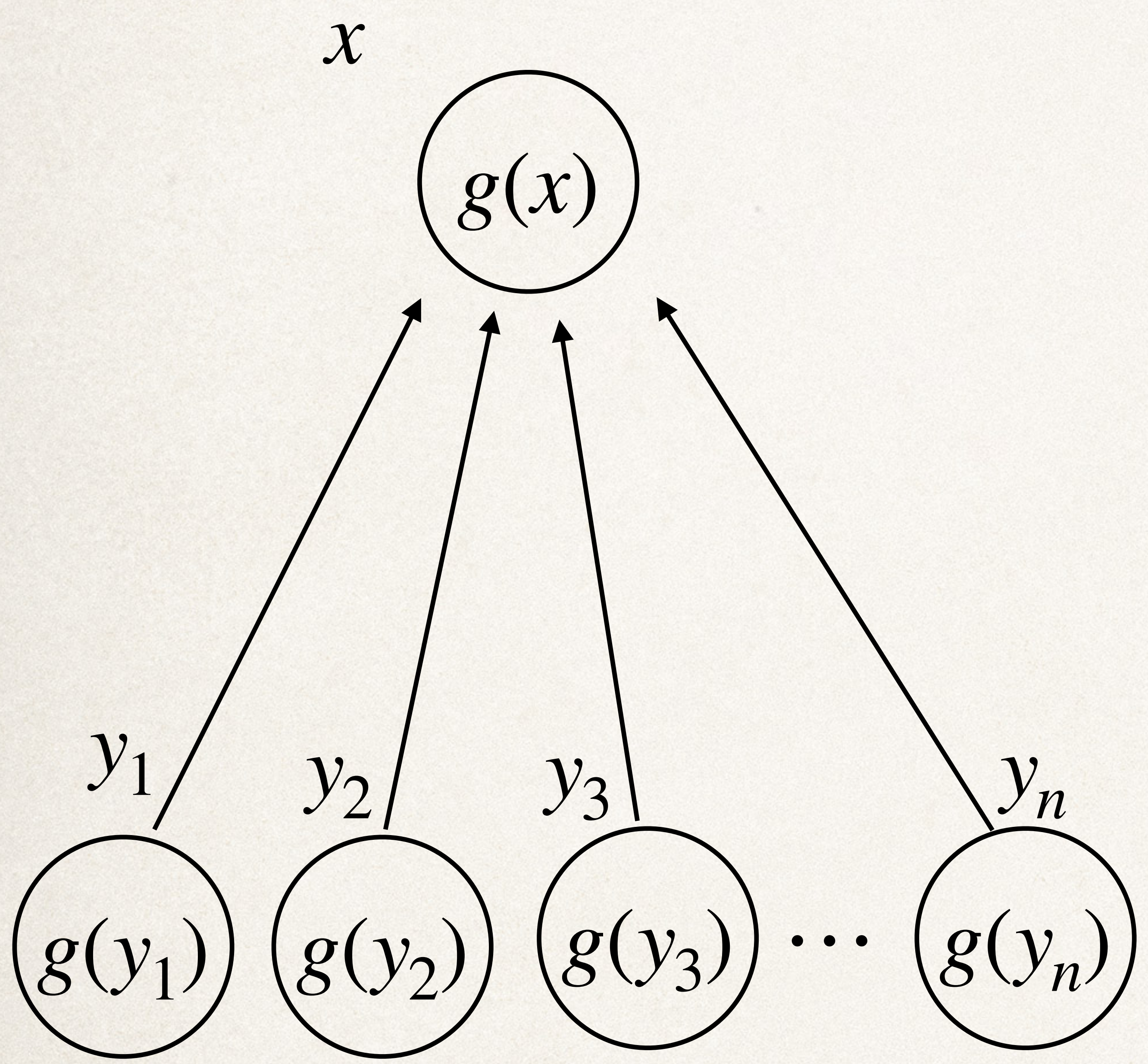
$A := \max(W_i, B_i)$とすると状態数が$O(A^3)$で遷移が$O(A^2)$なので計算量は$O(A^5)$．全体のゲームのGrundy数はNim和をとることで$O(A^5+N)$でわかる．

解答

$$g(w,b) = \mathrm{mex}\{g(w-1, b+w),\ g(w, b-1),\ g(w, b-2),\ \ldots\ g(w, b-b/2)\}$$

```cpp
#include <iostream>
#include <string.h>
#include <set>

int N;
const int MAX_N = 100010;
int W[MAX_N], B[MAX_N];

const int MAX_W = 60, MAX_B = 60;
int memo[MAX_W][MAX_W * MAX_B];

int g(int w, int b){
    if(memo[w][b] != -1) return memo[w][b];

    std::set<int> S;
    if(w >= 1) S.insert(g(w - 1, b + w));
    for(int k = 1; k <= b/2; k++) S.insert(g(w, b - k));

    int res = 0;
    while(S.count(res)) res++;
    return memo[w][b] = res;
}

void solve(){
    memset(memo, -1, sizeof(memo));

    int nimSum = 0;
    for(int i = 0; i < N; i++) nimSum ^= g(W[i], B[i]);

    if(nimSum) printf("First\n");
    else       printf("Second\n");
}
```

2章まとめ



- N山のNimで各山がx_i枚のとき：

$$x_1 \oplus x_2 \oplus \cdots \oplus x_N \neq 0 \Longleftrightarrow \text{先手必勝}，\ x_1 \oplus x_2 \oplus \cdots \oplus x_N = 0 \Longleftrightarrow \text{後手必勝}$$

- ゲームをNimに帰着できることもある．

- 不偏ゲーム$G = (X, F)$における局面xでGrundy数$g(x) = \mathrm{mex}\{g(y) : F(x)\}$に対して

$$g(x) \neq 0 \Longleftrightarrow \text{先手必勝}，\ g(x) = 0 \Longleftrightarrow \text{後手必勝}$$

- 不偏ゲーム$G = G_1 + G_2 + \ldots + G_N$における局面$x = (x_1, x_2, \ldots, x_N)$のGrundy数は$g(x_1, x_2, \ldots, x_N) = g_1(x_1) \oplus g_2(x_2) \oplus \cdots \oplus g_N(x_N)$である．また，これに対して

$$g(x) \neq 0 \Longleftrightarrow \text{先手必勝}，\ g(x) = 0 \Longleftrightarrow \text{後手必勝}$$

発展テーマ：ゲーム数が増える選択

ちょっと捻った問題だとゲーム数が増えるものがある．

わかりやすいのがコイン山の取り去りゲームの操作でコイン山を**分割**することもOKとするゲーム．

コインを減らすとき，例えば6枚から3枚に減らす

コインを分割するとき，例えば6枚から(2,4)に分割

ではこのときのGrundy数は何か求めよ，と問われたらどうしようか

ゲームが増えるときのgrundy数を求める練習

- 二人でコインの山でゲームをする．
- 毎ターン各プレイヤーに許される行動は次の２パターンのみ

  コインの１山から１枚もしくは２枚のコインを取る

  コイン山から１枚取って残った山を２山に分割する

- コイン山に対して行動できないとき負け．最後のコインを取ったら勝ち．
- 山にx枚のコインがあるときGrundy数を$g(x)$を求めよ．$O(x^2)$でよい．

x枚あるとき$x-1,\ x-2,\ (1,\ x-2),\ (2,\ x-3),\ (3,\ x-4),\ \ldots\ (x-1,\ 1)$に動かせる.

$x-1,\ x-2$枚のGrundy数は$g(x-1), g(x-2)$である

a, b枚の２山あるときこの状態のGrundy数は何だろうか.

x枚あるとき$x-1,\ x-2,\ (1,\ x-2),\ (2,\ x-3),\ (3,\ x-4),\ \ldots\ (x-1,\ 1)$に動かせる.

$x-1,\ x-2$枚のGrundy数は$g(x-1), g(x-2)$である

a, b枚の２山あるときこの状態のGrundy数は何だろうか.

a枚の山かb枚の山どちらかのみ指定して動かすのでゲームの和となる.

したがってこのときのGrundy数は$g(a) \oplus g(b)$である.

以上より

$$g(x) = \mathrm{mex}\{g(x-2),\ g(x-1),\ g(1) \oplus g(x-2),\ g(2) \oplus g(x-1),\ \ldots\ g(x-1) \oplus g(1)\}$$

$$g(x) = \mathrm{mex}\{g(x-2),\ g(x-1),$$
$$g(1) \oplus g(x-2),\ g(2) \oplus g(x-1),\ \ldots\ g(x-1) \oplus g(1),\ \}$$

を実装して$x = 0 \sim 24$まで出力して眺めてみる

```cpp
#include <iostream>
#include <string.h>
#include <set>

const int MAX_X = 25;
int memo[MAX_X + 10];

int g(int X){
    if(memo[X] != -1) return memo[X];

    std::set<int> S;
    if(X - 1 >= 0) S.insert(g(X - 1));
    if(X - 2 >= 0) S.insert(g(X - 2));
    for(int y = 1; X - 1 - y >= 1; y++) S.insert(g(y) ^ g(X - 1 - y));

    int res = 0;
    while(S.count(res)) res++;
    return memo[X] = res;
}

void print_g(int X = MAX_X){
    memset(memo, -1, sizeof(memo));

    for(int x = 0; x < X; x++) {
        printf("g(%d) = %d\n", x, g(x));
    }

    return;
}

int main(){
    print_g();
    return 0;
}
```

出力

```
g(0) = 0
g(1) = 1
g(2) = 2
g(3) = 3
g(4) = 0
g(5) = 1
g(6) = 2
g(7) = 3
g(8) = 0
g(9) = 1
g(10) = 2
g(11) = 3
g(12) = 0
g(13) = 1
g(14) = 2
g(15) = 3
g(16) = 0
g(17) = 1
g(18) = 2
g(19) = 3
g(20) = 0
g(21) = 1
g(22) = 2
g(23) = 3
g(24) = 0
```

$$g(x) = \mathrm{mex}\{g(x-2),\ g(x-1),$$
$$g(1) \oplus g(x-2),\ g(2) \oplus g(x-1),\ \ldots\ g(x-1) \oplus g(1),\ \}$$

を実装して$x = 0 \sim 24$まで出力して眺めてみる

```
#include <iostream>
#include <string.h>
#include <set>

const int MAX_X = 25;
int memo[MAX_X + 10];

int g(int X){
    if(memo[X] != -1) return memo[X];

    std::set<int> S;
    if(X - 1 >= 0) S.insert(g(X - 1));
    if(X - 2 >= 0) S.insert(g(X - 2));
    for(int y = 1; X - 1 - y >= 1; y++) S.insert(g(y) ^ g(X - 1 - y));

    int res = 0;
    while(S.count(res)) res++;
    return memo[X] = res;
}

void print_g(int X = MAX_X){
    memset(memo, -1, sizeof(memo));

    for(int x = 0; x < X; x++) {
        printf("g(%d) = %d\n", x, g(x));
    }

    return;
}

int main(){
    print_g();
    return 0;
}
```

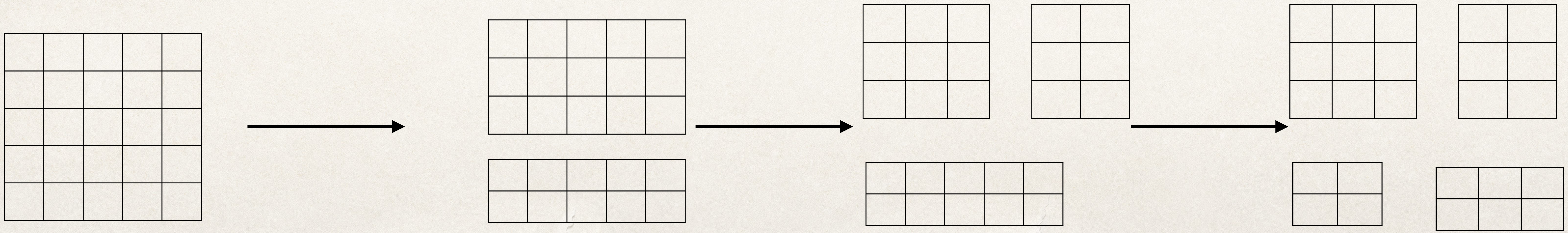
```
g(0) = 0
g(1) = 1
g(2) = 2
g(3) = 3
g(4) = 0
g(5) = 1
g(6) = 2
g(7) = 3
g(8) = 0
g(9) = 1
g(10) = 2
g(11) = 3
g(12) = 0
g(13) = 1
g(14) = 2
g(15) = 3
g(16) = 0
g(17) = 1
g(18) = 2
g(19) = 3
g(20) = 0
g(21) = 1
g(22) = 2
g(23) = 3
g(24) = 0
```

$$g(x) = x \pmod 4$$
である．帰納法で示せる．

ゲームが増える状態のGrundy数を求める練習

出典：POJ 2311

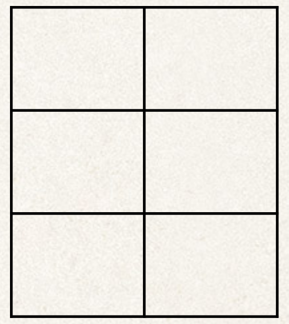
- 二人で横と縦の長さがW, Hである長方形の紙を長方形に切って分割するゲームをする．毎ターン分割してよい長方形は一つのみ．（切るまえにn個の長方形があったら，どれかを切った後は$n+1$個になる）
- 先に1×1の紙を切り出したら勝ち．
- 先手必勝なら「WIN」，負けなら「LOSE」を出力せよ．
- 制約）$2 \leq H, W \leq 200$



縦h横wの紙の状態を(h, w)と書くことにする．このときのGrundy数を$g(h, w)$とする．この値はどのように求められるだろうか．次に移れる局面を考えてそのGrundy数から求まる．

縦に切ると(h, i)と$(h, w-i)$に分かれる．

(h, i)と$(h, w-i)$のとき，この二つをまとめた状態のGrundyは何だろう．

$g($  ,  $) = ?$

(h, i)と$(h, w-i)$のとき，この二つをまとめた状態のGrundy数は何だろう．

ゲームのルールから(h, i)と$(h, w-i)$の**片方**だけ選んでゲームを進めるからこれはゲームの和となっている．よって$g(h, i) \oplus g(h, w-i)$

以上より

$$
\begin{aligned}
g(h, w) = \mathrm{mex}\{&g(h,2) \oplus g(h, w-2),\ (h,3) \oplus g(h, w-3),\ \ldots g(h, w-2) \oplus g(h,2),\\
&g(2,w) \oplus g(h-2,w),\ (3,w) \oplus g(h-3,w),\ \ldots g(h-2,w) \oplus g(2,w)\}
\end{aligned}
$$

$$g(h,w) = \operatorname{mex}\{g(h,2) \oplus g(h,w-2),\ (h,3) \oplus g(h,w-3),\ \ldots g(h,w-2) \oplus g(h,2),$$
$$g(2,w) \oplus g(h-2,w),\ (3,w) \oplus g(h-3,w),\ \ldots g(h-2,w) \oplus g(2,w)\}$$

```cpp
#include <iostream>
#include <string.h>
#include <set>

const int MAX_H = 210, MAX_W = 210;
int memo[MAX_H][MAX_W];

int g(int h, int w){
    if(memo[h][w] != -1) return memo[h][w];


    std::set<int> S;
    for(int i = 2; h - i >= 2; i++) S.insert(g(i, w) ^ g(h - i, w));
    for(int i = 2; w - i >= 2; i++) S.insert(g(h, i) ^ g(h, w - i));

    int res = 0;
    while(S.count(res)) res++;
    return memo[h][w] = res;
}



void solve(int h, int w){

    if(g(h, w) != 0) printf("WIN\n");
    else             printf("LOSE\n");

    return;
}
```

おまけ：２次元コインゲーム

- ２次元の格子状ボード上にコインがすべて埋まっているとする．コインの座標を(x, y)で表すことにする．ただし$(0,0)$からはじまる．

- 動作は(x, y)が表のコインならウラにひっくり返し，さらに同じ行の１枚(x, b)か同じ列の１枚(a, y)をひっくり返せる．この２枚コインの座標は制限があってそれぞれ

$$0 \leq b < y, \; 0 \leq a < x$$

- Grundy数$g(x, y)$を求めよ．

$$g(x, y) = \mathrm{mex}\{g(x, b),\ g(a, y) : 0 \leq b < y,\ 0 \leq a < x\}$$

これは$x \oplus y$と同じ（前の方で扱った）

２次元コインゲーム

- ２次元の格子状ボード上にコインがすべて埋まっているとする．コインの座標を(x, y)で表すことにする．ただし(0,0)からはじまる．

- 動作は(x, y)が表のコインならウラにひっくり返し，さらに同じ行の１枚(x, b)と同じ列の１枚(a, y)，加えてもう１枚(a, b)をひっくり返す．この3枚コインの座標は制限があってそれぞれ

$$0 \leq b < y,\ 0 \leq a < x$$

- Grundy関数$g(x, y)$を求めよ．

$$g(x, y) = \operatorname{mex}\{g(x, b) \oplus g(a, y) \oplus g(a, b) : 0 \leq b < y,\ 0 \leq a < x\}$$

$$g(x, y) = \text{mex}\{g(x, b) \oplus g(a, y) \oplus g(a, b) : 0 \le b < y,\ 0 \le a < x\}$$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	2	3	1	8	10	11	9	12	14	15	13	4	6	7	5	32
3	0	3	1	2	12	15	13	14	4	7	5	6	8	11	9	10	48
4	0	4	8	12	6	2	14	10	11	15	3	7	13	9	5	1	64
5	0	5	10	15	2	7	8	13	3	6	9	12	1	4	11	14	80
6	0	6	11	13	14	8	5	3	7	1	12	10	9	15	2	4	96
7	0	7	9	14	10	13	3	4	15	8	6	1	5	2	12	11	112
8	0	8	12	4	11	3	7	15	13	5	1	9	6	14	10	2	128
9	0	9	14	7	15	6	1	8	5	12	11	2	10	3	4	13	144
10	0	10	15	5	3	9	12	6	1	11	14	4	2	8	13	7	160
11	0	11	13	6	7	12	10	1	9	2	4	15	14	5	3	8	176
12	0	12	4	8	13	1	9	5	6	10	2	14	11	7	15	3	192
13	0	13	6	11	9	4	15	2	14	3	8	5	7	10	1	12	208
14	0	14	7	9	5	11	2	12	10	4	13	3	15	1	8	6	224
15	0	15	5	10	1	14	4	11	2	13	7	8	3	12	6	9	240
16	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	24

- 小ネタ

$$g(x, y) = \mathrm{mex}\{g(x, b) \oplus g(a, y) \oplus g(a, b) : 0 \le b < y,\ 0 \le a < x\}$$

を$x \otimes y$という演算で表すことができる．$g(x, y) = x \otimes y$

ニム積　$x \otimes y := \mathrm{mex}\{(x \otimes b) \oplus (a \otimes y) \oplus (a \otimes b) : 0 \le b < y,\ 0 \le a < x\}$

(分配法則や結合法則がある)

- 小ネタ

より一般的に１次元コインゲームG_1でコインの場所xに対して場所$x_1, x_2, \ldots, x_m$のコインをひっくり返し，同じく１次元コインゲームG_2でコインの場所yに対して場所$y_1, y_2, \ldots, y_n$のコインをひっくり返すとする．このとき次のような２次元ゲーム$G_1 \times G_2$のルールを定める．

(x, y)を表→ウラ，加えて$(x_i, y_j)(1 \leq i \leq m, 1 \leq j \leq n)$のコインをひっくり返す

１次元ゲームG_1, G_2のSGの関数を各々$g_1(x), g_2(x)$とする．２次元ゲーム$G_1 \times G_2$のSG関数$g(x, y)$は次の通り．

$$g(x, y) = g_1(x) \otimes g_2(y)$$

発表内容はここまで！

スライドで使った問題

- 1章
- K-Stones　(EDPC K)
- マス目と駒　(ARC038 B)
- 2章
- Tower Breakers, Revisited! (HackerRank)
- VS AtCoder (典型90問 031)
- Cutting Game (POJ 2311)

問題集

- ARC038-C 「茶碗と豆」
- ABC206-F 「Interval Game 2」
- HackerRank Grundy

  Grundy数まわりの問題が15問程度ある

参考資料

- GAME THEORY

  元々はUniversity of California, Los Angelsで使われた講義テキスト.

  演習問題が豊富. 邦訳と解答あり.