**THM** (Sprague-Grundy) Every finite, impartial normal game is equivalent to some nimber.

Proof: by induction on length of game

COR: Let $G = \{G_1, \dots, G_m\}$ where $G_1 \simeq *n_1, \dots, G_m \simeq *n_m$

then $G \simeq *\,\text{mex}(n_1, \dots, n_m)$

↳ minimum excludant = smallest nonneg int not in set

$\text{mex}(\{0,1,4,7,8\}) = 2$

COR: If $G_1 \simeq *n_1$ and $G_2 \simeq *n_2$ then $G_1 + G_2 \simeq *(n \oplus m)$

Proof:

position

$G = \{G_1, \dots, G_m\}$

positions resulting from one move (all smaller than $G$)

$\simeq \{*n_1, \dots, *n_m\}$ ← (apply inductive hypothesis)

Claim $G \simeq *\,\text{mex}(\{n_1, \dots, n_m\})$

$[G + *\,\text{mex}(n_1, \dots, n_m)] \not\simeq *0$ (is a P position)

Consider all moves

moves on $G$: $*n_i + *\,\text{mex}$ [want this is N]

↓ find winning move

2 cases $n_i > \text{mex}$

winning move $*n_i \to *\,\text{mex}$

$n_i < \text{mex}$

similar

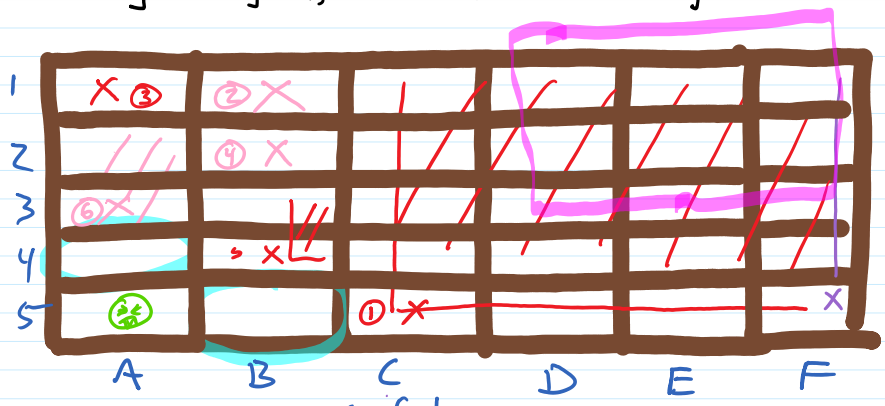
~~$n_i \neq \text{mex}$~~

moves on $*\,\text{mex}$: similar

**Chomp**

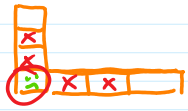
Play on m×n grid. Take turns selecting remaing cell, remove all above and to right

Last move loses.
(misere)

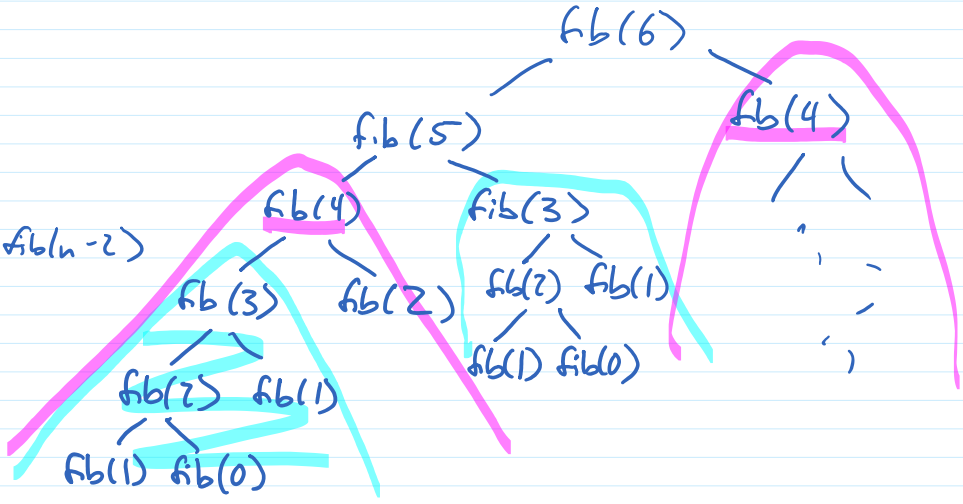


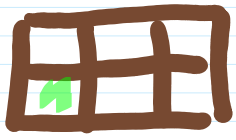

pos in finite impartial game

```
outcome_class(p)
if p is end of game
    return value according to rules   (normal P, misere N)
else
    S ← positions reachable in 1 move from p
    if S contains a P position        ← if outcome_class(x) returns P for any x in S
        return N + corresponding move
    else
        return P
```

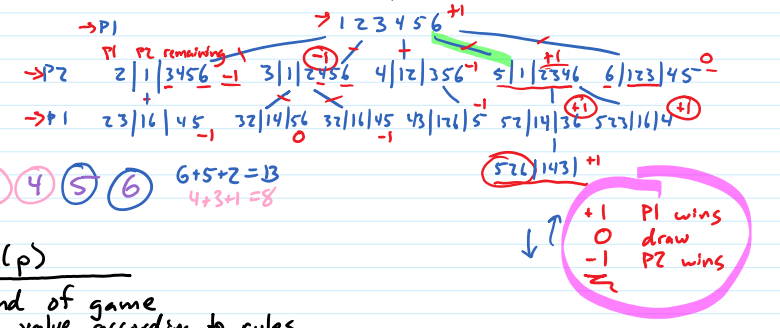


```
def fib(n):
    if n<2:
        return n
    else:
        return fib(n-1) + fib(n-2)
```




Divisors: Start with 1...n, players take turns taking a number with remaining divisors; opponent gets all the remaining divisors. Game is over when no moves remain; winner is player with higher sum (draw if =)




1 2 3 4 5 6 6+5+2=13 4+3+1=8

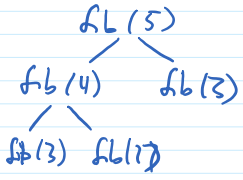
Minimax(p)

if p is end of game
  return value according to rules
else
  S ← positions reachable in 1 move from p
  if P1 move at p  return max(Minimax(S))
  else  return min(Minimax(S))

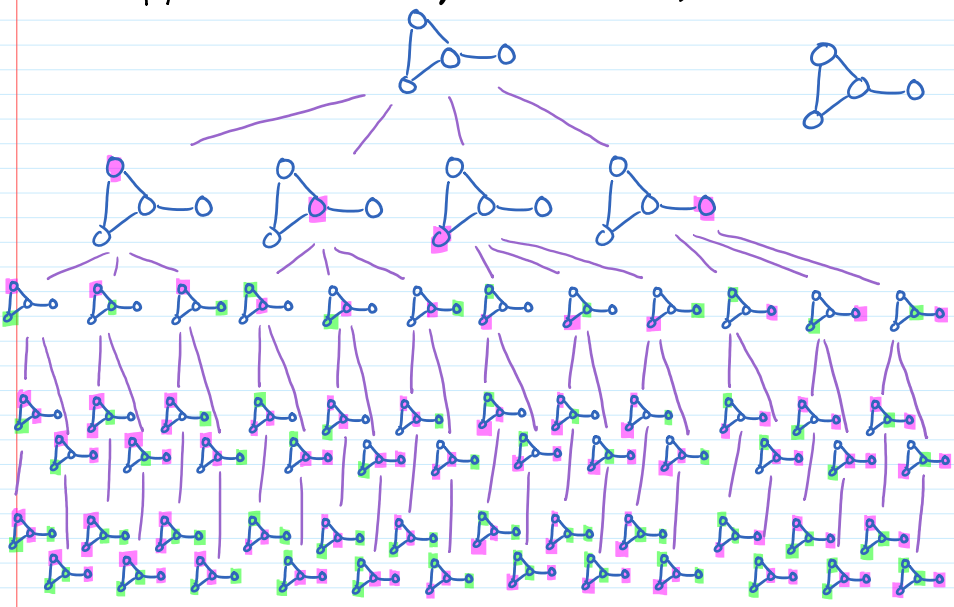
Minimax(p)

if p is end of game
  return value according to rules
else
  let S = positions reachable in 1 move from p
  if p is P1's turn
    return max Minimax(p')
           p'∈S
  else return min Minimax(p')
           p'∈S

```
def fib(n):
    if n<2: add (n,n) to memo
        return n
    else:                          check if n-1, x is in memo
        return fib(n-1) + fib(n-2)   if so, use value in memo
                                   check if (n-2,y) in memo
```



Graph: take turns coloring a vertex in a graph with your color
player who covers the most edges wins (draw if =)



```
def fib(n):
    if n<2:
        return n
    else:
        return fib(n-1) + fib(n-2)
```