Backtracking

```
find_solution(s)
  if s is solved position return []

  for every possible move m
    update s to reflect move m
    solution =  find_solution(s)
    if solution is not NIL
      return [m] + solution
    else
      undo move m in s
  return NIL
```

— returns sequence of moves to solve puzzle starting at position s

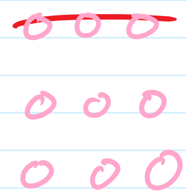
return seq of moves needed to solved

often faster to update/undo than copy + update

(restore original state of s)

current state of game

find_move (s) — returns winning move for current state (or NIL if none)



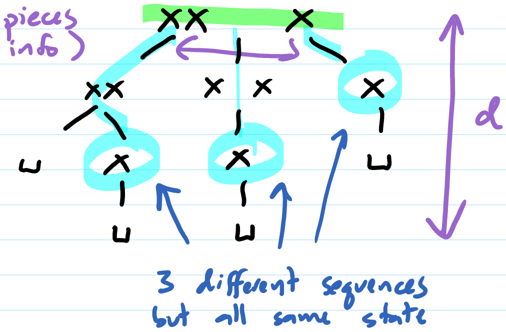
    if s is a terminal position (game over)
        if you won, return WIN (no move necessary)
        else (you lost) return NIL no winning move

    else
        for every possible move m
            update state s according to move m, call result s'
            if find_move(s') == NIL opponent has no winning move at s' so m is a winning move for you at s
                return m

        return NIL no move m was a winning move

Complexity Game complexity - Wikipedia

State Space – # of possible states (configuration of pieces + other info)



Game tree size – # of sequences of moves
$\approx b^d$

Branching factor – # moves possible at each turn (usually an average)
b

Depth – # turns (average)
d

red/black, king/normal

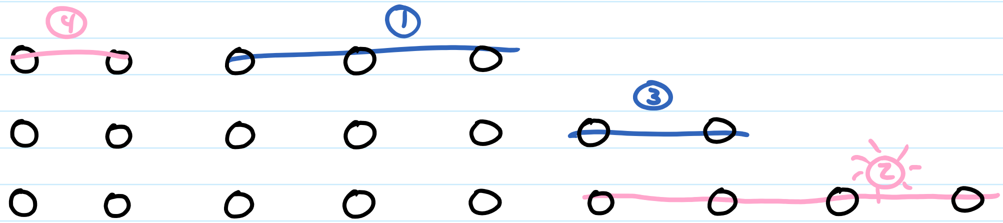
checkers – 32 positions – each can hold one of 4 kinds of pieces or be empty

$$\# \text{states} \lessapprox 5^{32} \approx 10^{20}$$

go – 361 positions holding black/white/nothing

$$\# \text{states} \le 3^{361} \approx 10^{180}$$

Nim



Start with rows of $n_1, n_2, \ldots, n_k$ stones

On each turn, take as many stones as you wish from one row

If no possible moves, you lose (last move wins)

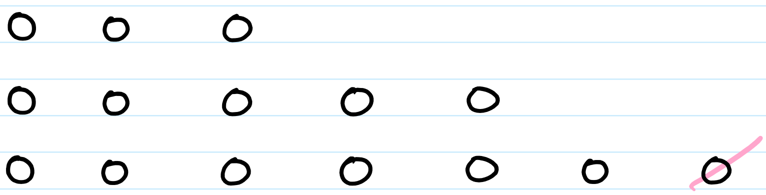
DEF: If player to make next move has a winning strategy, position is an N position (winning); otherwise a P position (losing)

final position in a normal game is a P position (last move wins)

otherwise if there is a move to losing pos, position is N position
else P position

THM: Nim position is P if and only if xor is 0

COR: winning move makes xor 0



011	011
101	101
111	110
001	000

1) compute xor of stones in each row, x
2) find most significant 1 in x
3) find a row r with that bit set
4) remain ← x xor count(r)
5) take count(r) − remain from row r

xor is now = ~~x~~ xor ~~count(r)~~ xor (~~x~~ xor ~~count(r)~~) = 0

x w/o row r — the new count in row r

00000
0000000
000000000

0101
0111
0010
0000

0101
0111
1001 → 0010
1011

remain = 2

Proof: Induction on number of stones (for all n, positions with n stones obey rule)

Proof: Induction on number of stones (for all n, positions with n stones obey rule)

Base case: n = 0 → already lost so P

Induction: Let n > 0 and assume positions with m < n stones are P if and only if xor is 0

If xor is 0

any move makes xor non-zero → changes some bits in one row and the same bits in the xor

so is to an N position by ind. hyp.

so original is a P position

If xor is non-zero

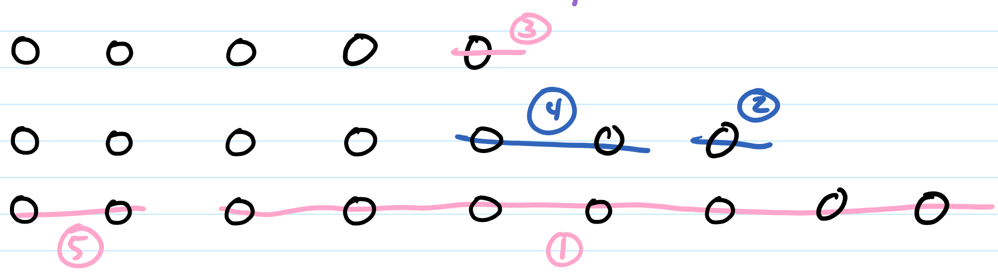
then there is a move that makes the xor 0

ind. hyp. applies → move is to a P pos

so original is a P position

100
110
010

000



100
100
000

000

Game Positions

Game position = set of positions you can move to

take any number

In traditional 1-row Nim

$$\sqcup = \{\} = \emptyset$$

$$0 = \{\sqcup\}$$

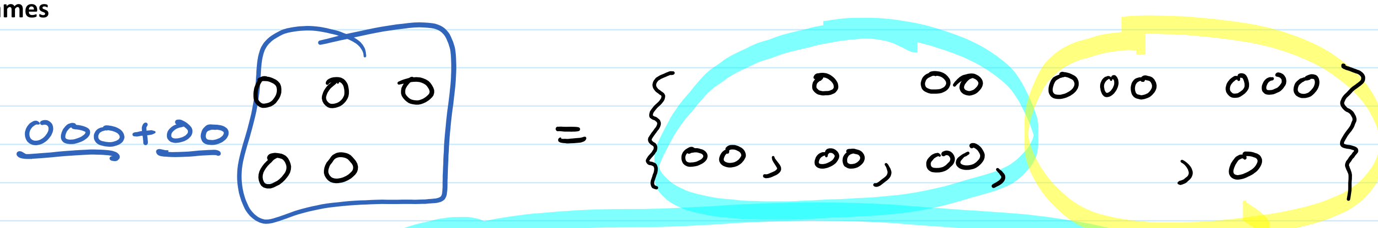
$$00 = \{\sqcup, 0\}$$

$$000 = \{\sqcup, 0, 00\}$$

$$0000 =$$

$$\vdots$$

Sums of Games



$$G + H = \{ G' + H \mid G' \text{ is an } \underset{\text{move}}{\underline{\text{option}}} \text{ of } G \}$$

$$\cup$$

$$\{ G + H' \mid H' \text{ is an option of } H \}$$

Equivalence of Games

G is equivalent to G′

For impartial, normal games G, G', say $\underline{G \approx G'}$ if and only if

for any game H, $G+H$ and $G'+H$

are both $\mathcal{N}$ positions
or both $\mathcal{P}$ positions
(so replacing G with G′ doesn't change outcome)

Is $*2 \approx *1$? $\quad *2 + \_\_\_ \quad *1 + \_\_\_$
Nimbers

Is $*5 \approx *3$? $\quad *5 + \_\_\_ \quad *3 + \_\_\_$

Conjecture: $\forall m, n \in \mathbb{N}$, $m \neq n \rightarrow$

Is $*2 + *1 \approx *3$

$*2 + *1 + \underline{*0}$ N $\qquad *3 + \underline{*0}$

$*2 + *1 + \underline{*1}$ N $\qquad *3 + \underline{*1}$

$*2 + *1 + \underline{*2}$ $\qquad *3 + \underline{*2}$

$*2 + *1 + \underline{*3}$ $\qquad *3 + \underline{*3}$

000

$*2 + *1 + \underline{*4}$ $\quad$ $*3 + \underline{*4}$

Conjecture: $*n + *m \approx$