
1.2 What is a Combinatorial Game? We now define the notion of a combinatorial game more precisely. It is a game that satisfies the following conditions.

(1) *There are two players.*

(2) *There is a set, usually finite, of possible positions of the game.*

(3) *The rules of the game specify for both players and each position which moves to other positions are legal moves. If the rules make no distinction between the players, that is if both players have the same options of moving from each position, the game is called **impartial**; otherwise, the game is called **partizan**.*

(4) *The players alternate moving.*

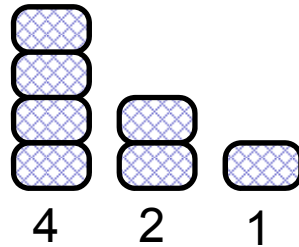
(5) *The game ends when a position is reached from which no moves are possible for the player whose turn it is to move. Under the **normal play rule**, the last player to move wins. Under the **misère play rule** the last player to move loses.*

If the game never ends, it is declared a draw. However, we shall nearly always add the following condition, called the **Ending Condition**. This eliminates the possibility of a draw.

(6) *The game ends in a finite number of moves no matter how it is played.*

Game example and analysis

Game rules



Three piles with 4, 2, 1, pegs(s).

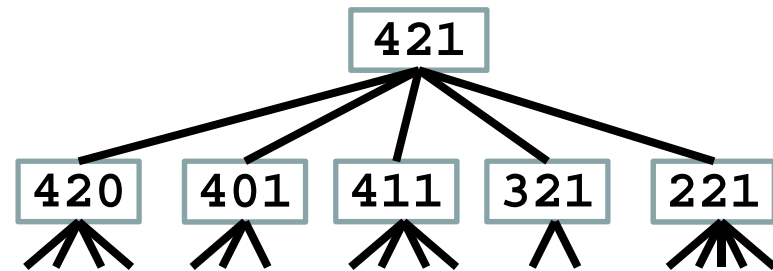
In one move, a player can remove 1 or 2 pegs from any pile(s).

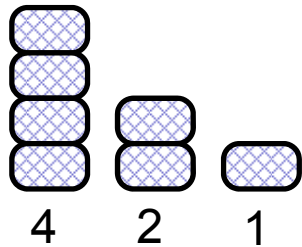
Player who removes the last peg wins.

Game representation

Represent the piles by a triple of integers, number of pegs in the piles, the initial state (position) is then [4,2,1].

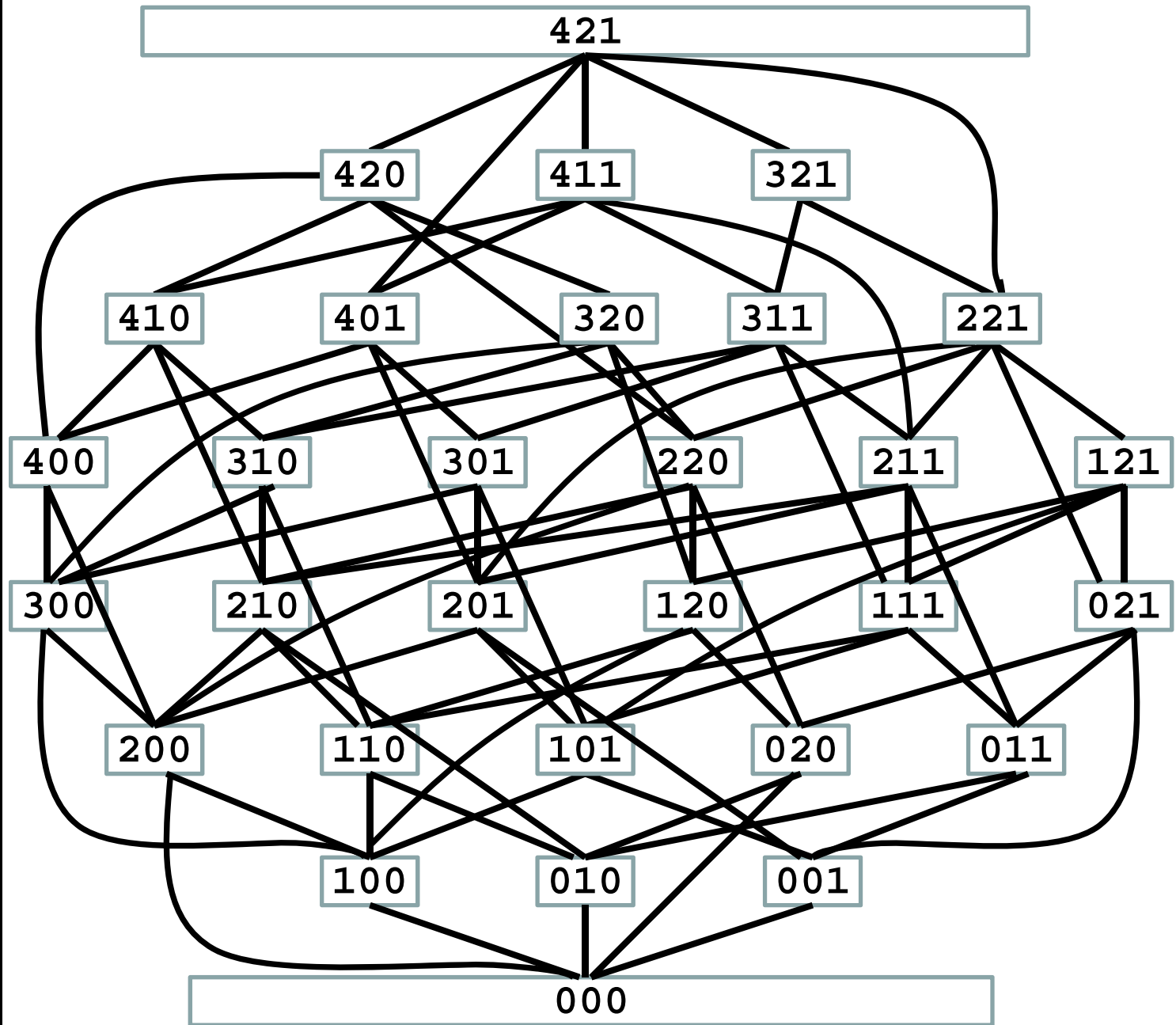
The states (positions) accessible in a single move are connected to the current state by directed edges.
(In the examples, the direction is downwards.)

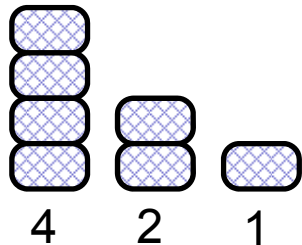




Each player can remove 1 or 2 pegs.

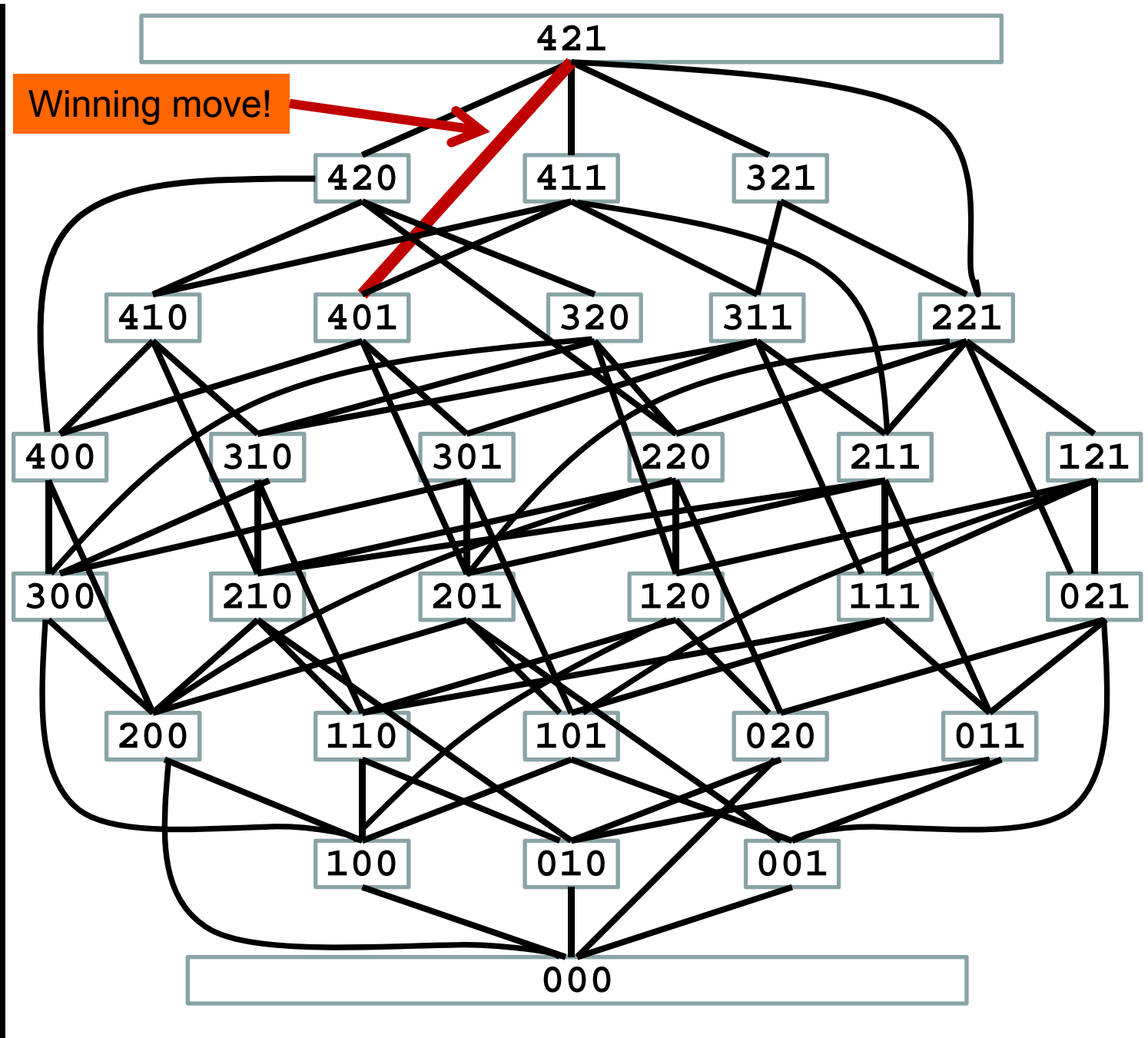
Player who removes the last peg wins.





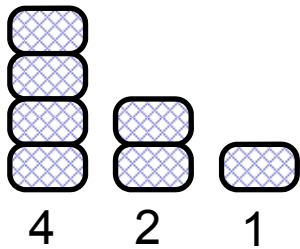
Each player can remove 1 or 2 pegs.

Player who removes the last peg wins.



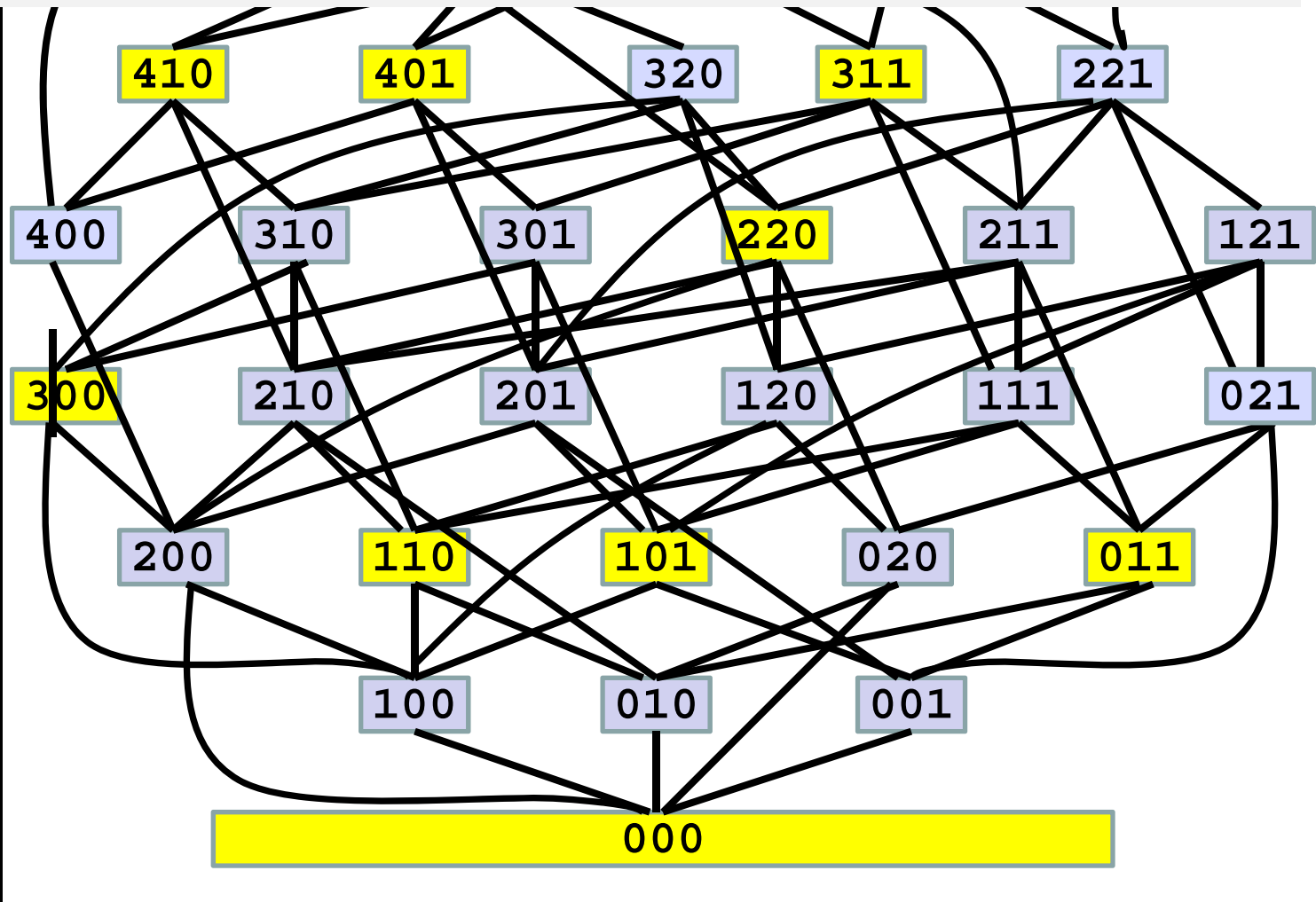
P positions are positions that are winning for the Previous player (the player who just moved to the position)

N positions are positions that are winning for the Next player (the player who will move to some next position).



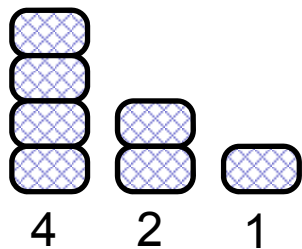
Each player can remove 1 or 2 pegs.

Player who removes the last peg wins.



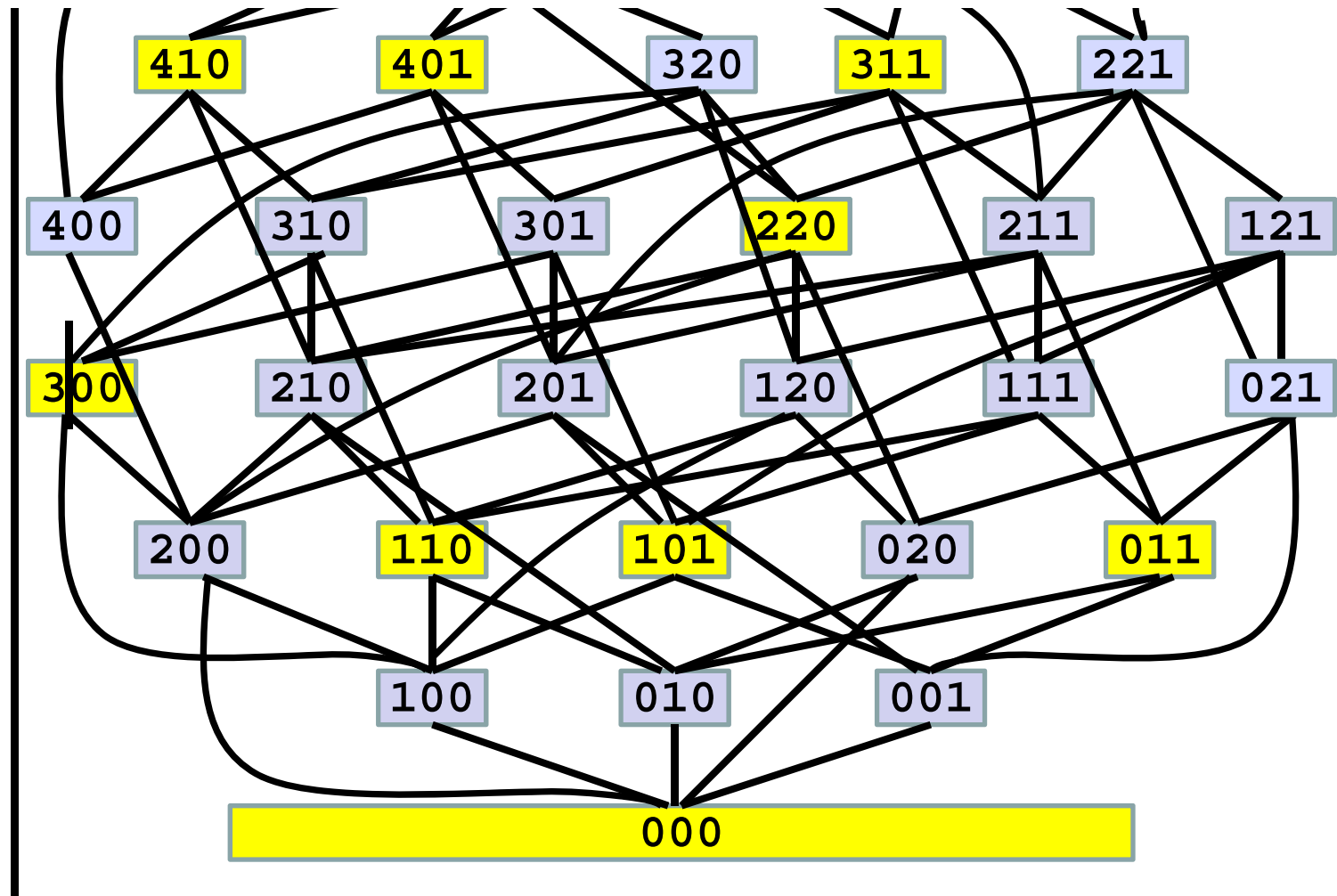
Characteristic Property. *P*-positions and *N*-positions are defined recursively by the following three statements.

- (1) All terminal positions are *P*-positions.
- (2) From every *N*-position, there is at least one move to a *P*-position.
- (3) From every *P*-position, every move is to an *N*-position.



Each player can remove 1 or 2 pegs.

Player who removes the last peg wins.



Determining P and N positions

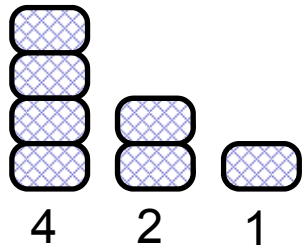
Step 1: Label every terminal position as a P-position.

Step 2: Label every position that can reach a labelled P-position in one move as an N-position.

Step 3: Find those positions whose only moves are to labelled N-positions; label such positions as P-positions.

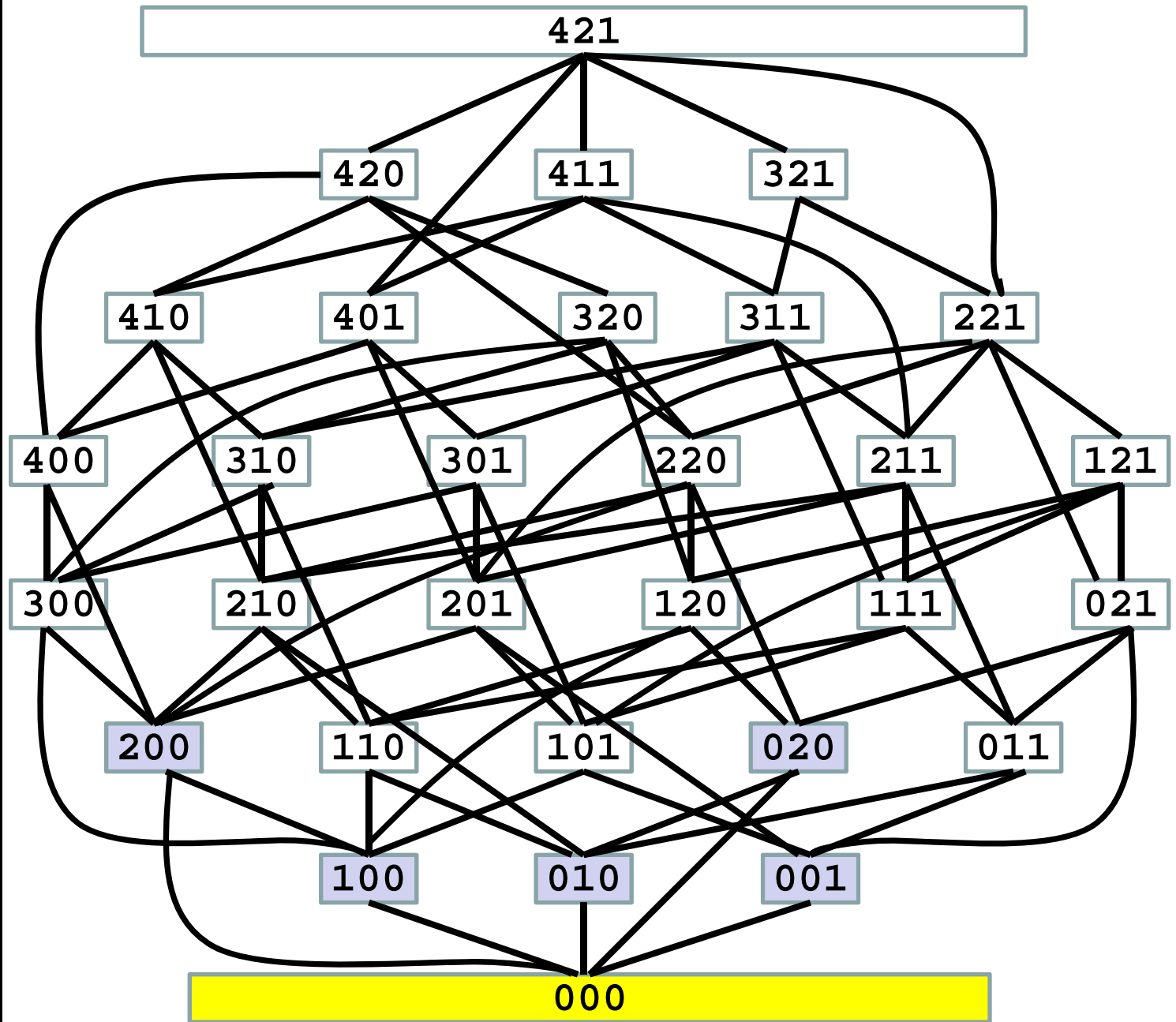
Step 4: If no new P-positions were found in step 3, stop; otherwise return to step 2.

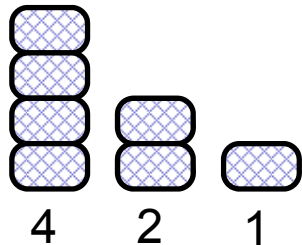
It is easy to see that the strategy of moving to P-positions wins. From a P-position, your opponent can move only to an N-position (3). Then you may move back to a P-position (2). Eventually the game ends at a terminal position and since this is a P-position, you win (1).



Each player can remove 1 or 2 pegs.

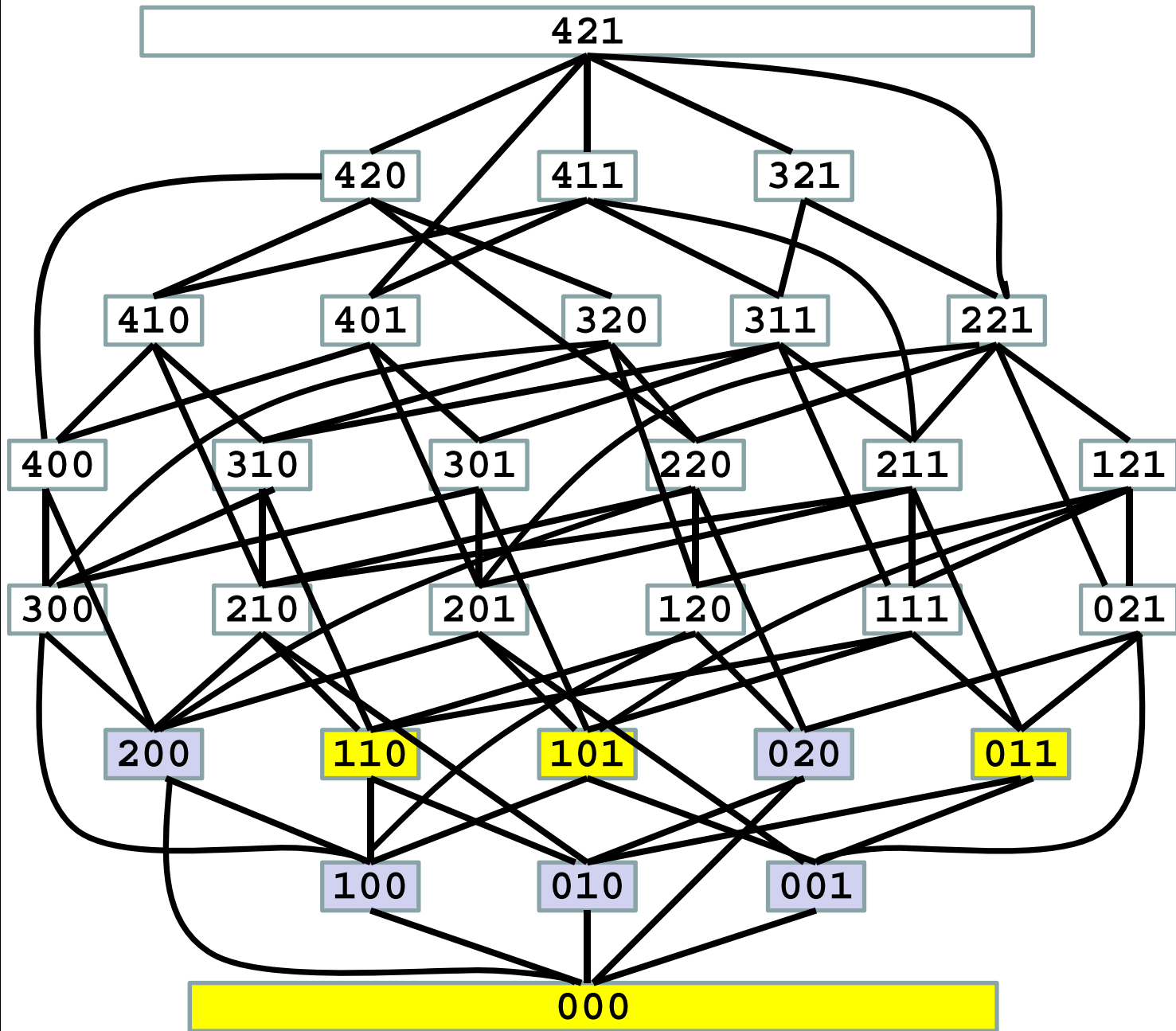
Player who removes the last peg wins.

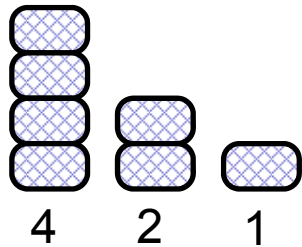




Each player can remove 1 or 2 pegs.

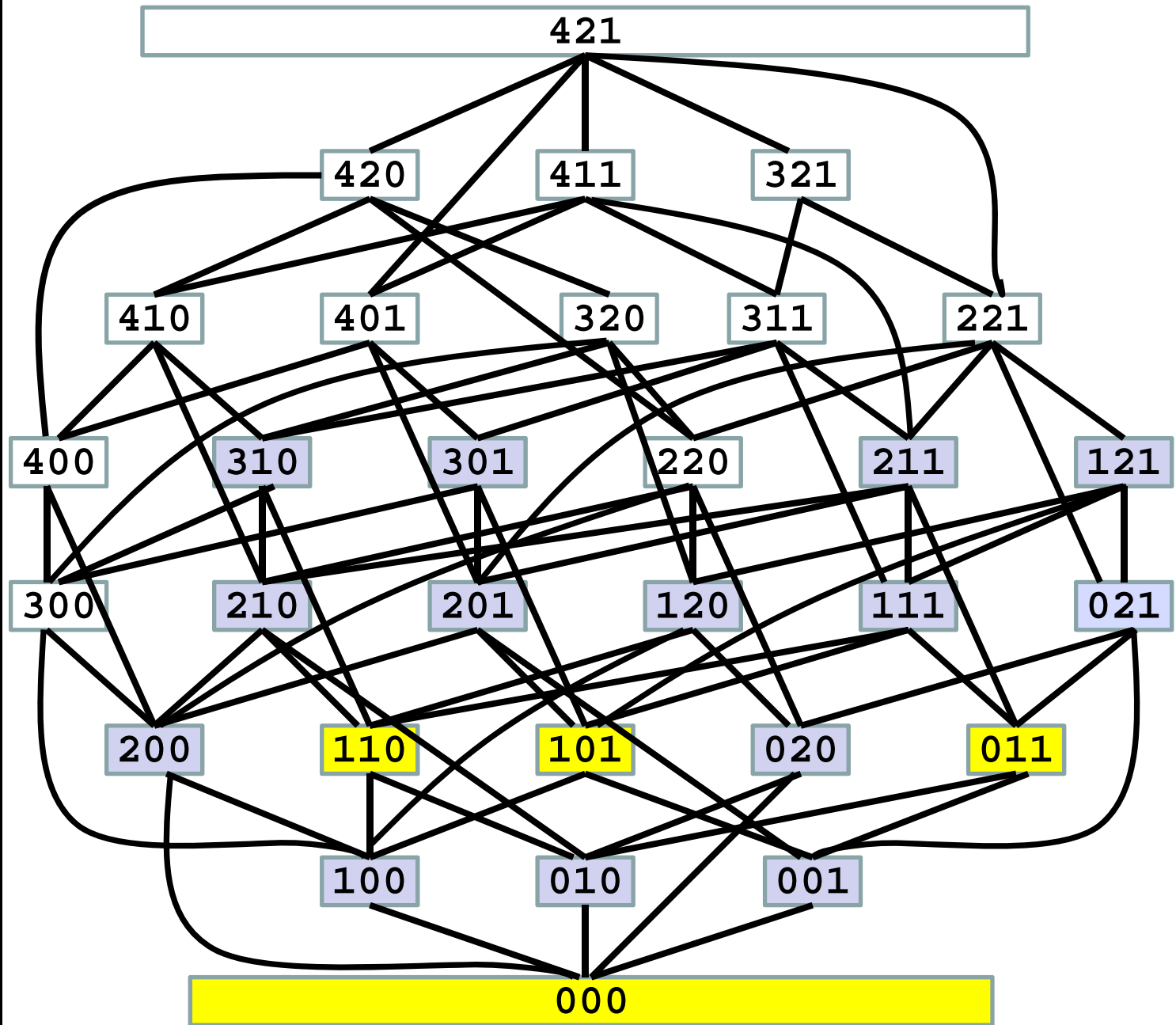
Player who removes the last peg wins.

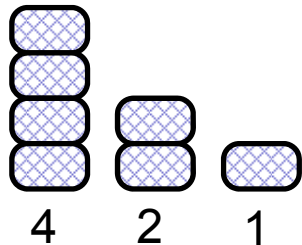




Each player can remove 1 or 2 pegs.

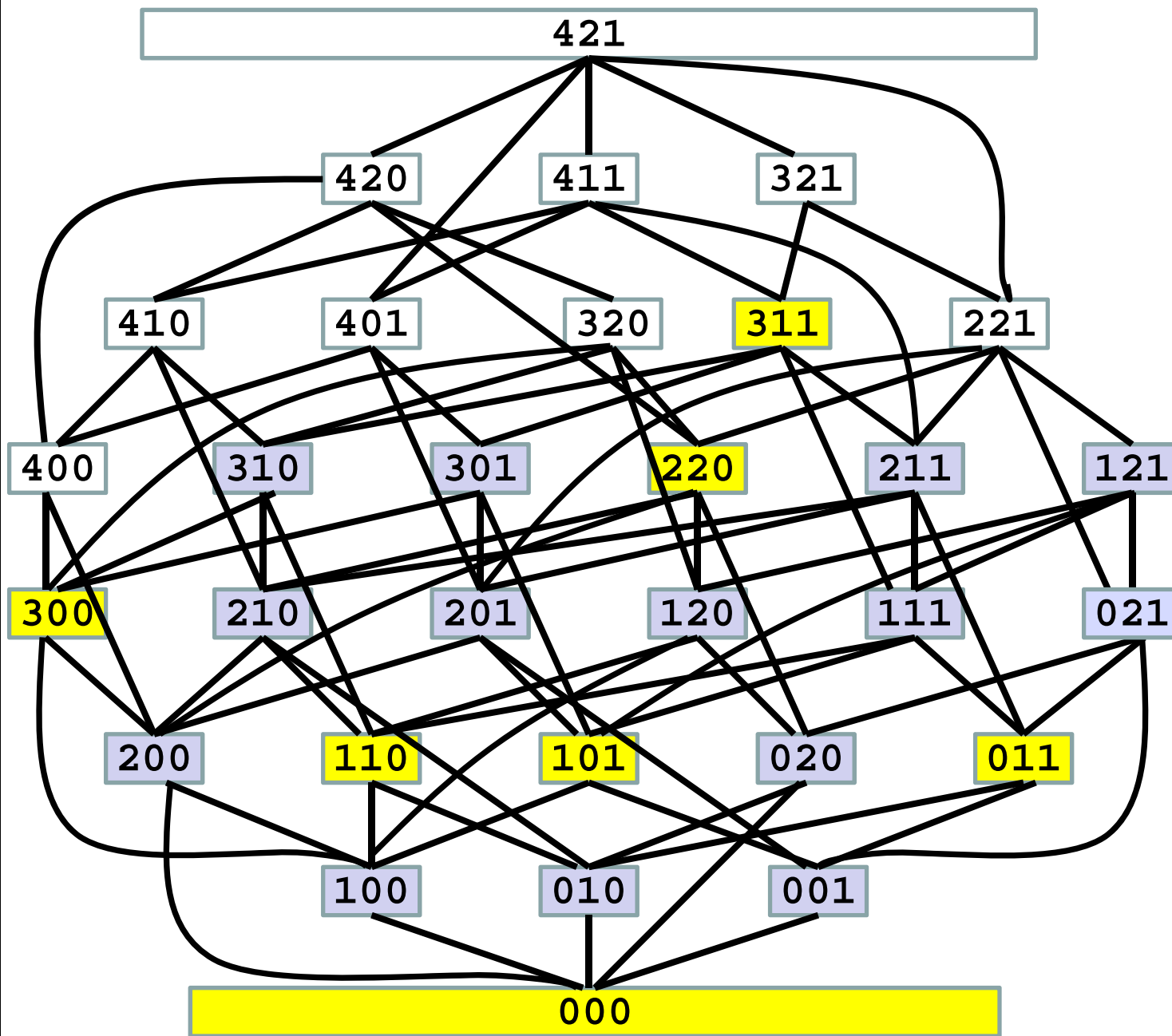
Player who removes the last peg wins.

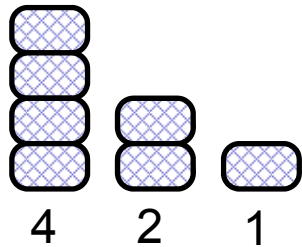




Each player can remove 1 or 2 pegs.

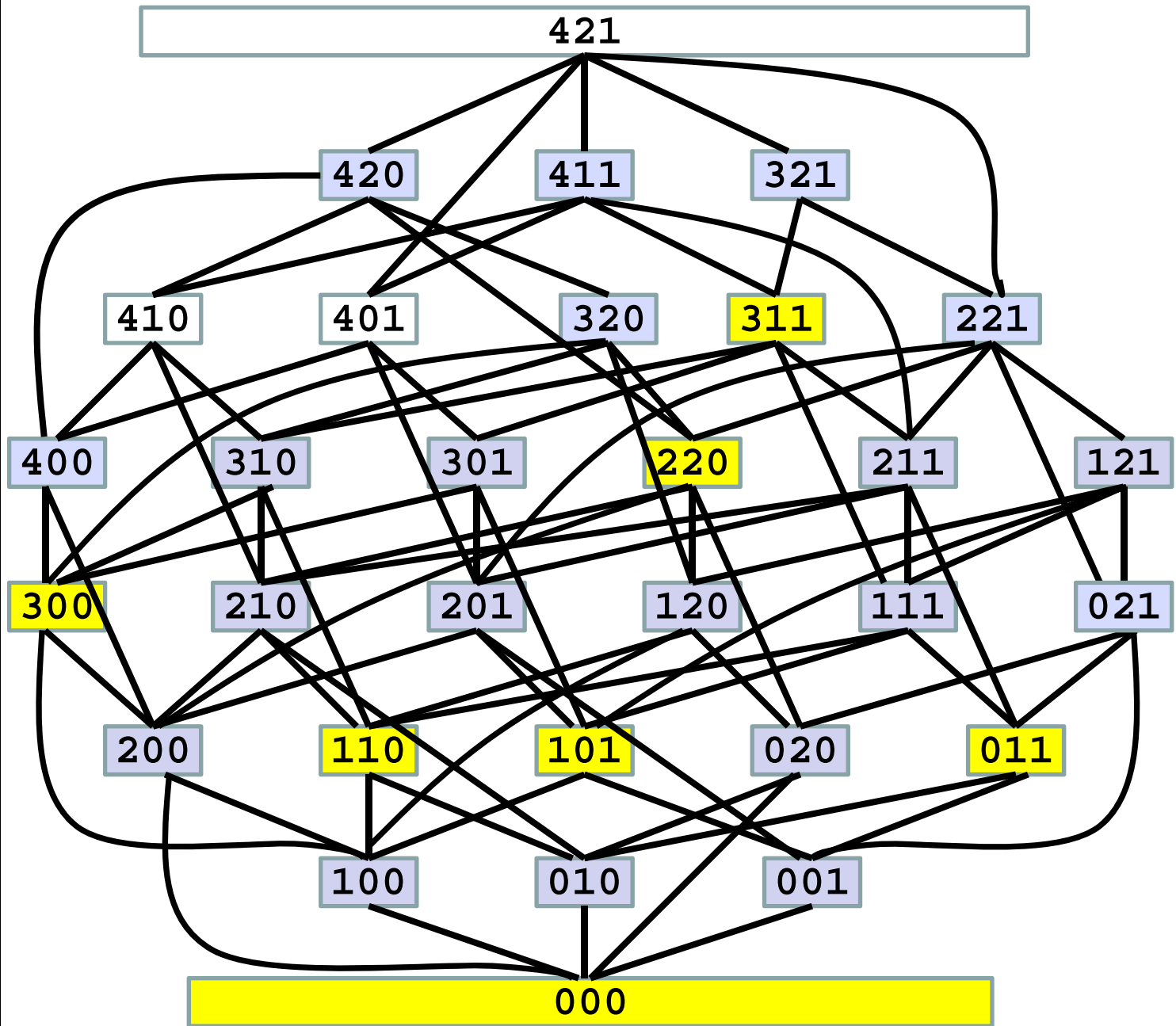
Player who removes the last peg wins.

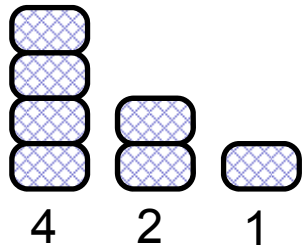




Each player can remove 1 or 2 pegs.

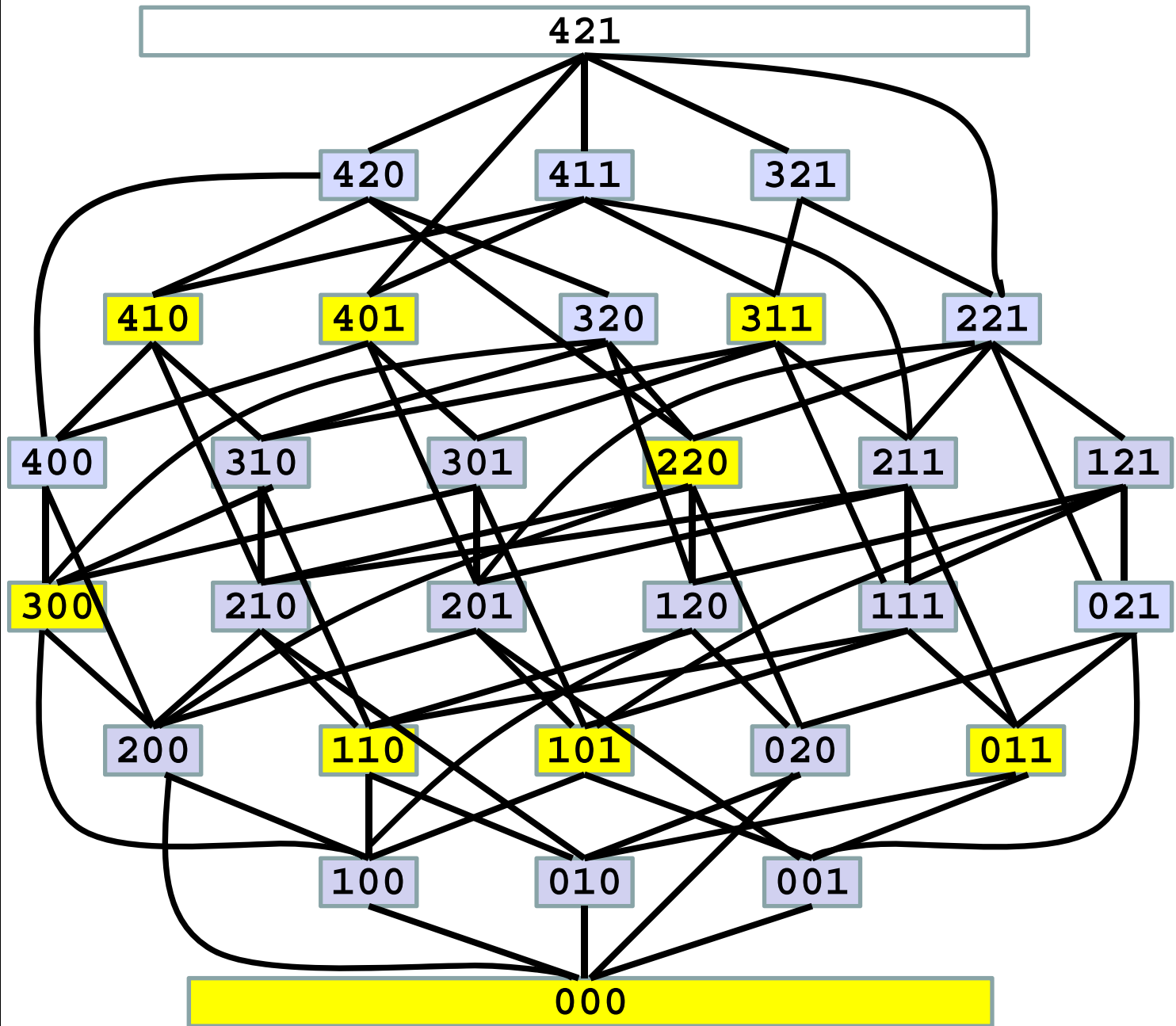
Player who removes the last peg wins.

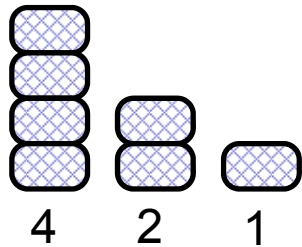




Each player can remove 1 or 2 pegs.

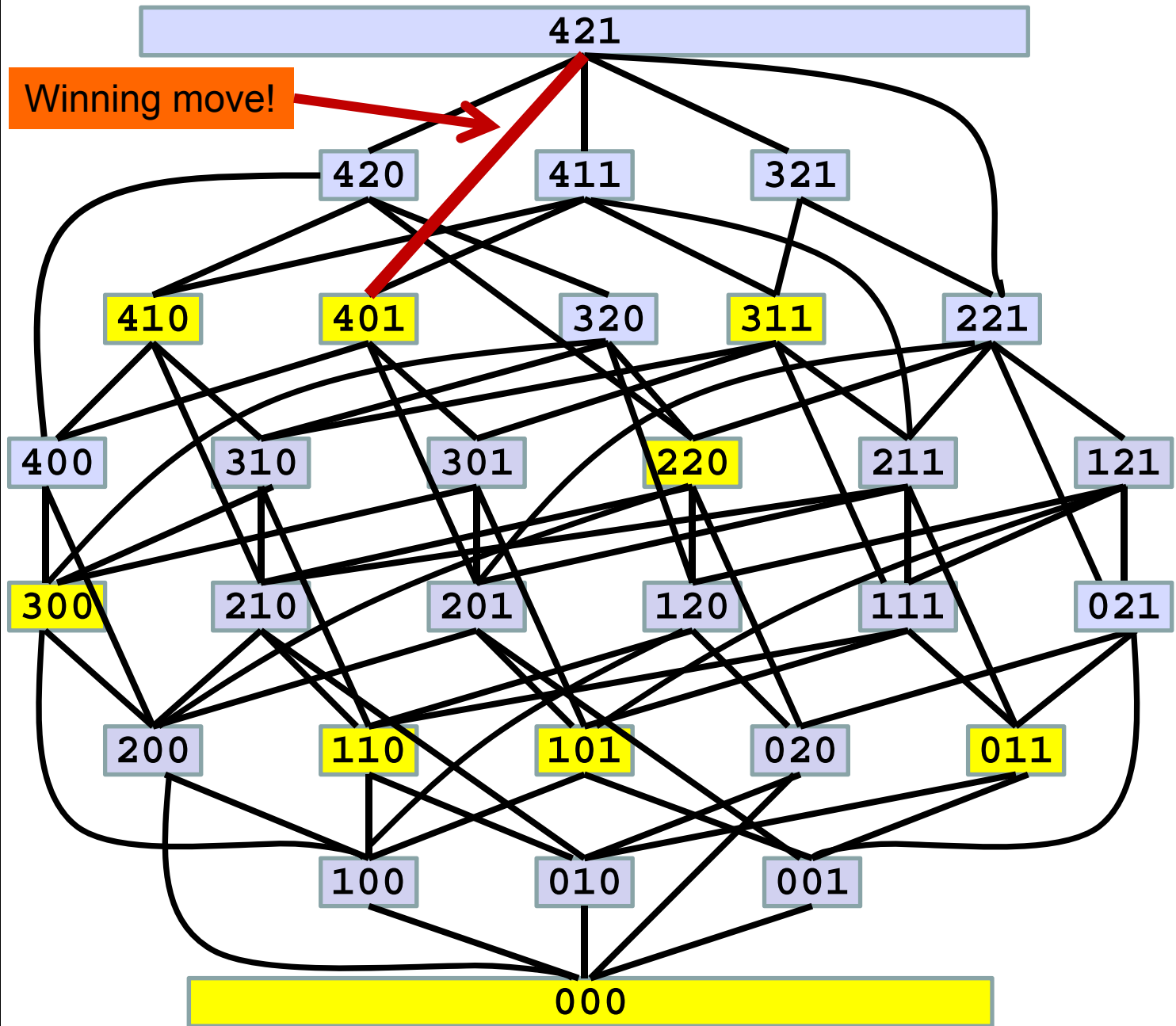
Player who removes the last peg wins.





Each player can remove 1 or 2 pegs.

Player who removes the last peg wins.



Properties of P- and N- positions

- Any edge from any P-position ends in an N-position.
- From any N-position, there exists (at least one) edge to a P-position.

Graph kernel DAG G , Set of nodes S .

- Any edge from any node in S ends in $V(G) - S$ (outside S).
- From any node in $V(G) - S$ (outside S) there exists (at least one) edge to a P-position.

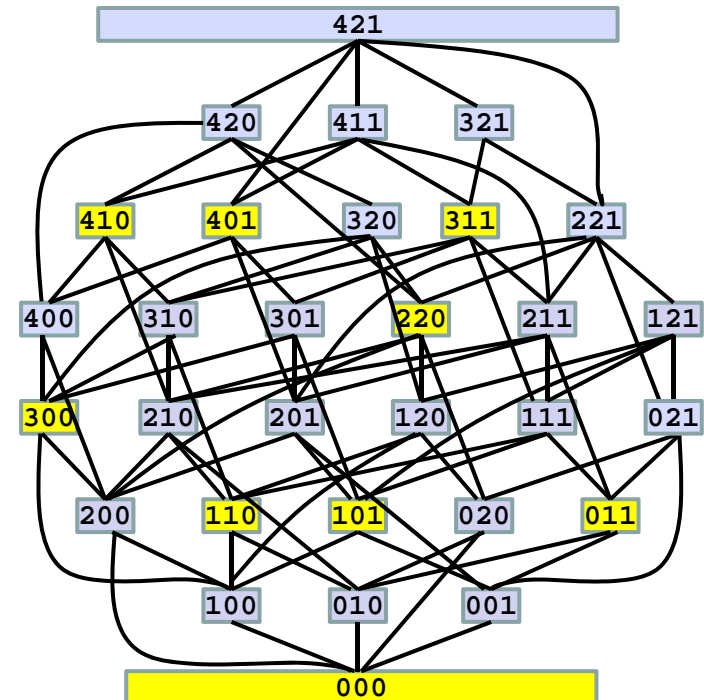
The set of all P-positions is the **kernel** of the DAG.

Fundamental fact

Each DAG has unique kernel.

Fundamental fact

The sets of P-positions and N-positions are uniquely specified in a DAG..



Jørgen Bang-Jensen, Gregory Z. Gutin

Digraphs Theory, Algorithms and Applications

Springer Monographs in Mathematics (2nd ed.), Springer-Verlag, ISBN 978-1-84800-997-4.

(also downloadable from web)

Subtraction Games

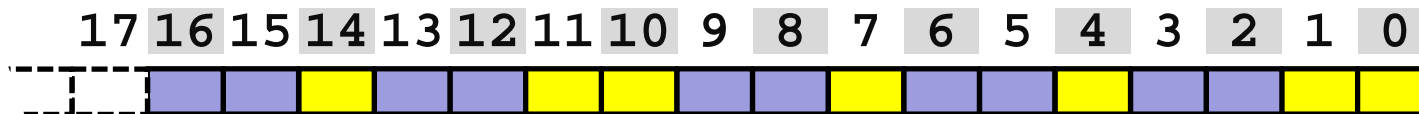
Let S be a set of positive integers.

The subtraction game with subtraction set S is played as follows.

From a pile with a large number, say n , of chips, two players alternate moves. A move consists of removing s chips from the pile where $s \in S$.

Last player to move wins.



Example



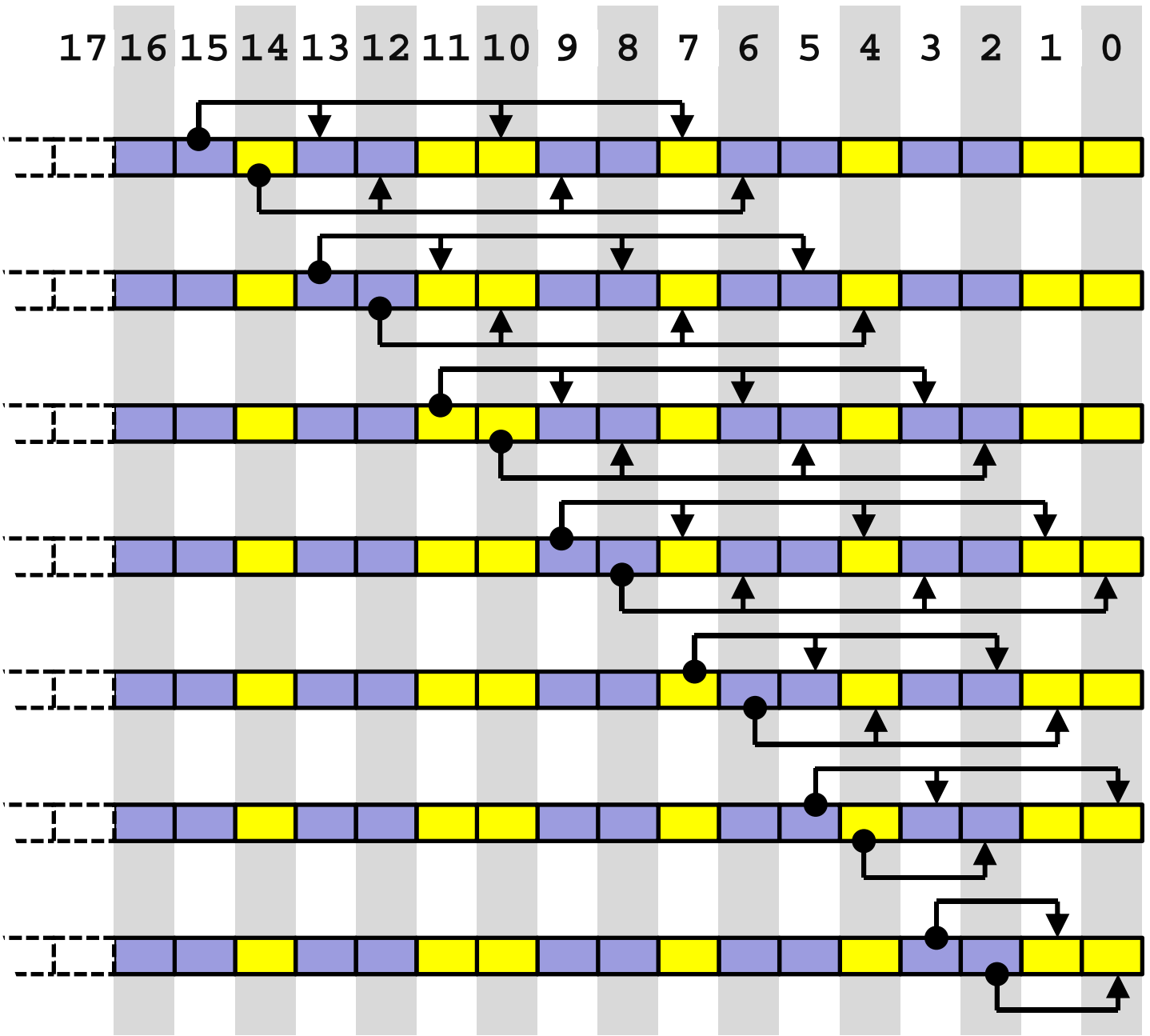
P- and N- positions in subtraction game with subtraction set $\{2, 5, 8\}$.



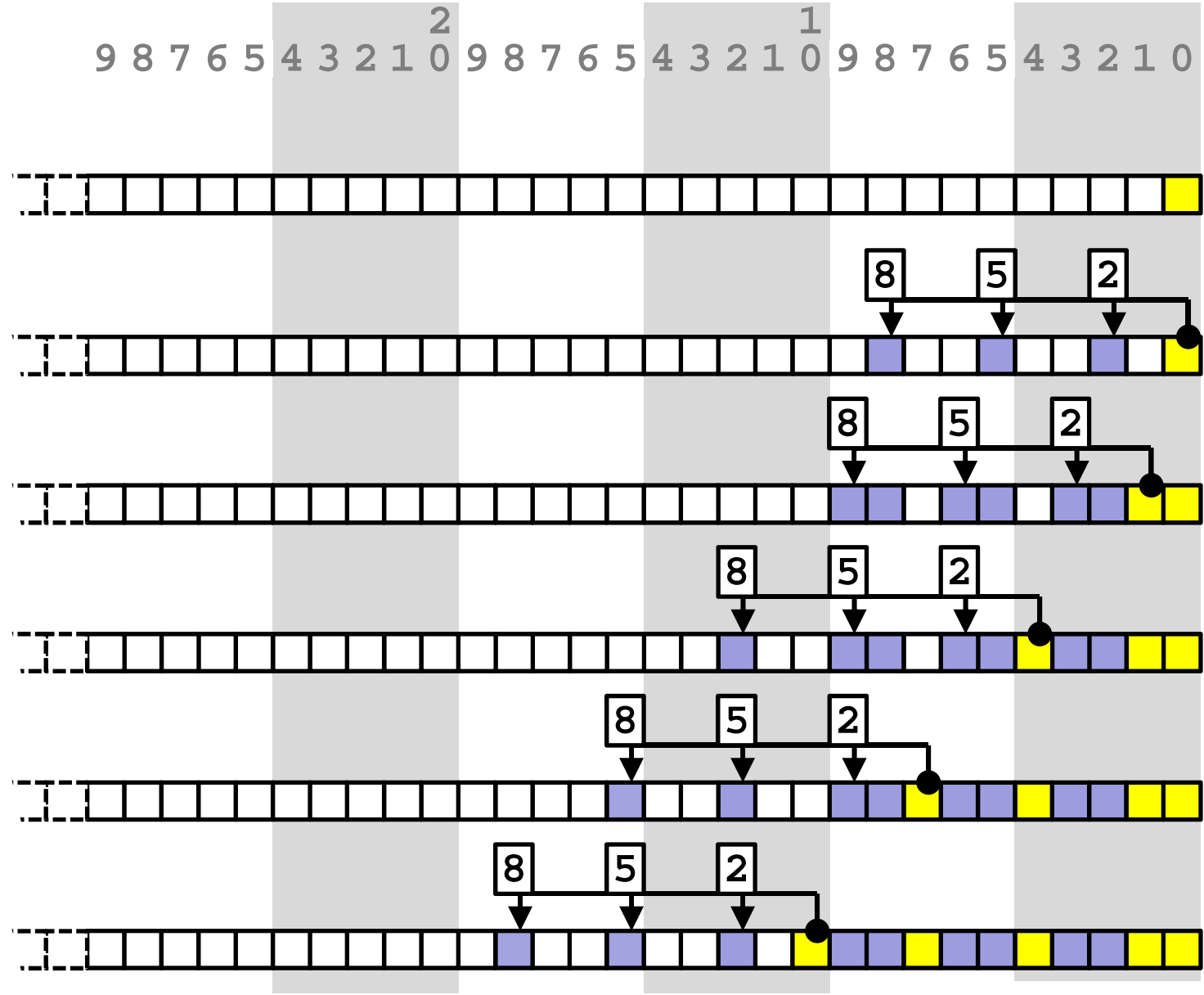
From state K there is a transition to the states $K-2$, $K-5$ and $K-8$
(if those are non-negative).

P- and N-
 
 positions in
 subtraction
 game with
 subtraction
 set {2, 5, 8}.

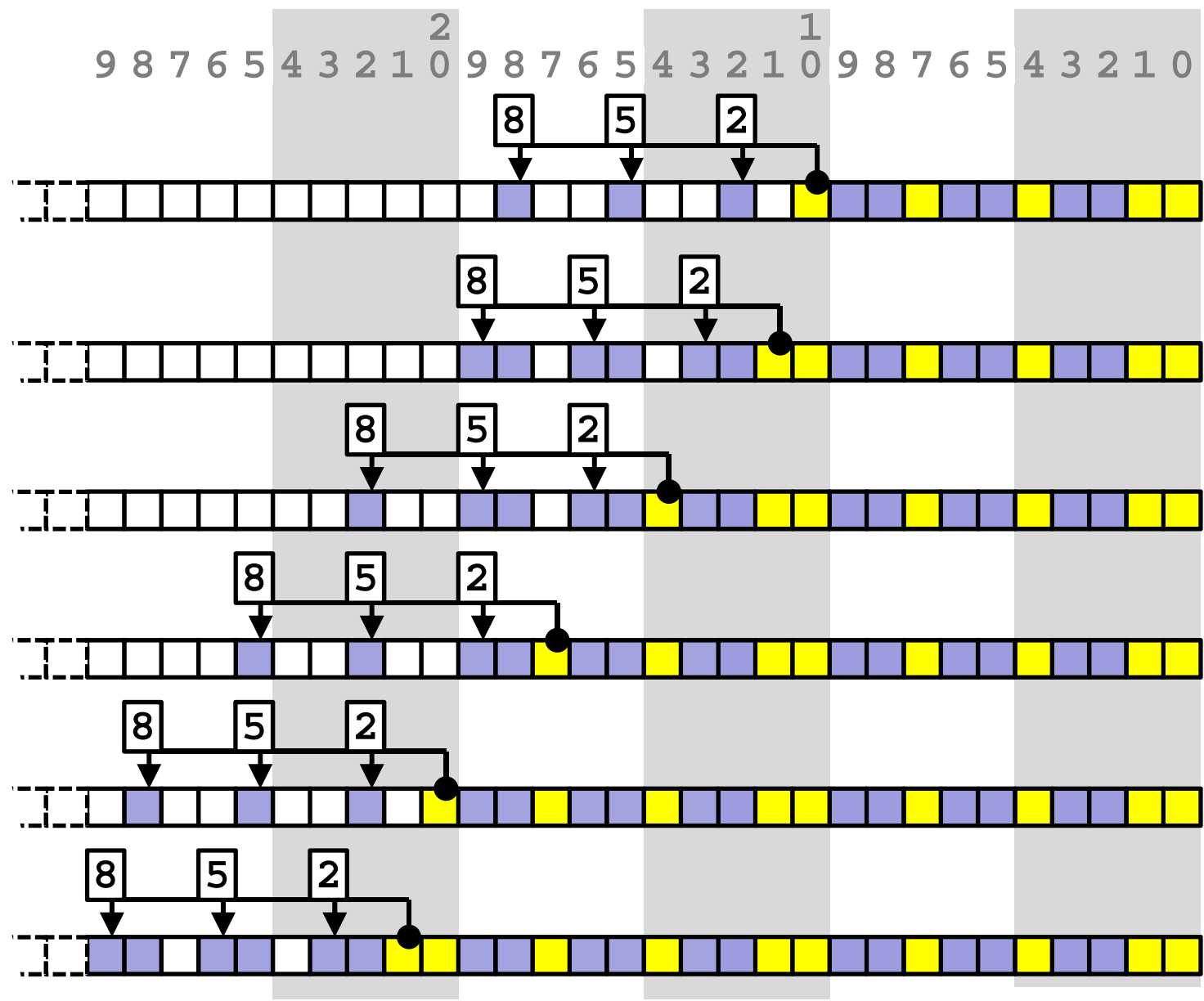
From state K
 there is a
 transition
 to the states
 $K-2$, $K-5$ and
 $K-8$
 (if those are
 non-negative).



Generate P and N positions in the subtraction game with subtraction set {2, 5, 8}.



Generate P and N positions in the subtraction game with subtraction set {2, 5, 8}.

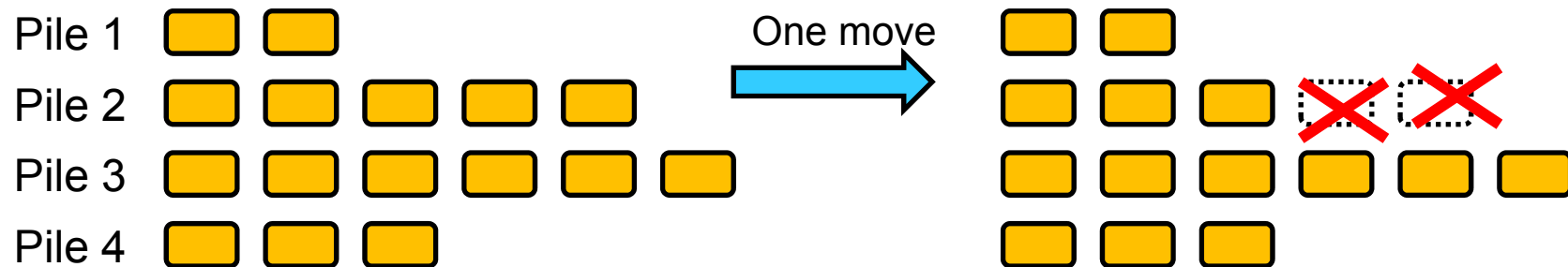


etc...

Nim:

- Objects (tokens, stones, matchsticks, ...) in a few piles.
Number of piles is not limited, number of objects in a pile is not limited too.
- One move: Take one or more objects from any one pile.
Also, an entire pile can be taken.
- Players take turns.
- Normal game -- player taking the last object wins
misère game -- player taking the last object loses

Example with 4 piles:



Try it yourself:

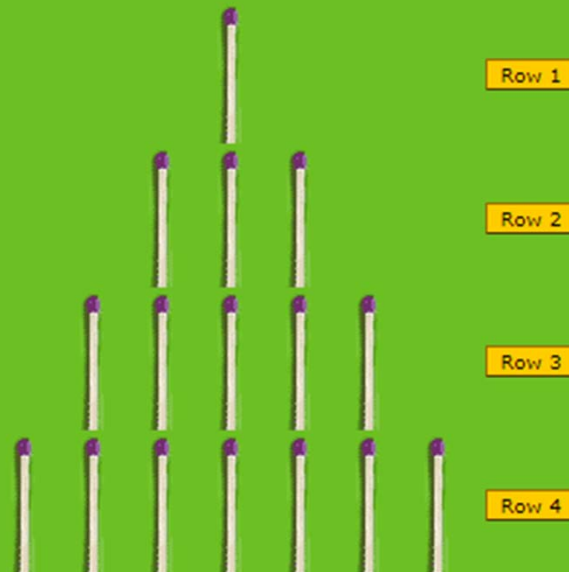
https://www.archimedes-lab.org/game_nim/play_nim_game.html

Online Game by Archimedes' Lab
Play Nim against your computer!

In one move, you can remove any number of matches but only from one row. Select any 'Row button' and click it to REMOVE matches. After each move click 'PC move' to make the computer play. You win if you leave the LAST match for the computer.

Who Moves First

At the start of a game, you have the first move, unless you allow the computer to play first by pressing the 'PC move' button.



New game

Close window

PC move

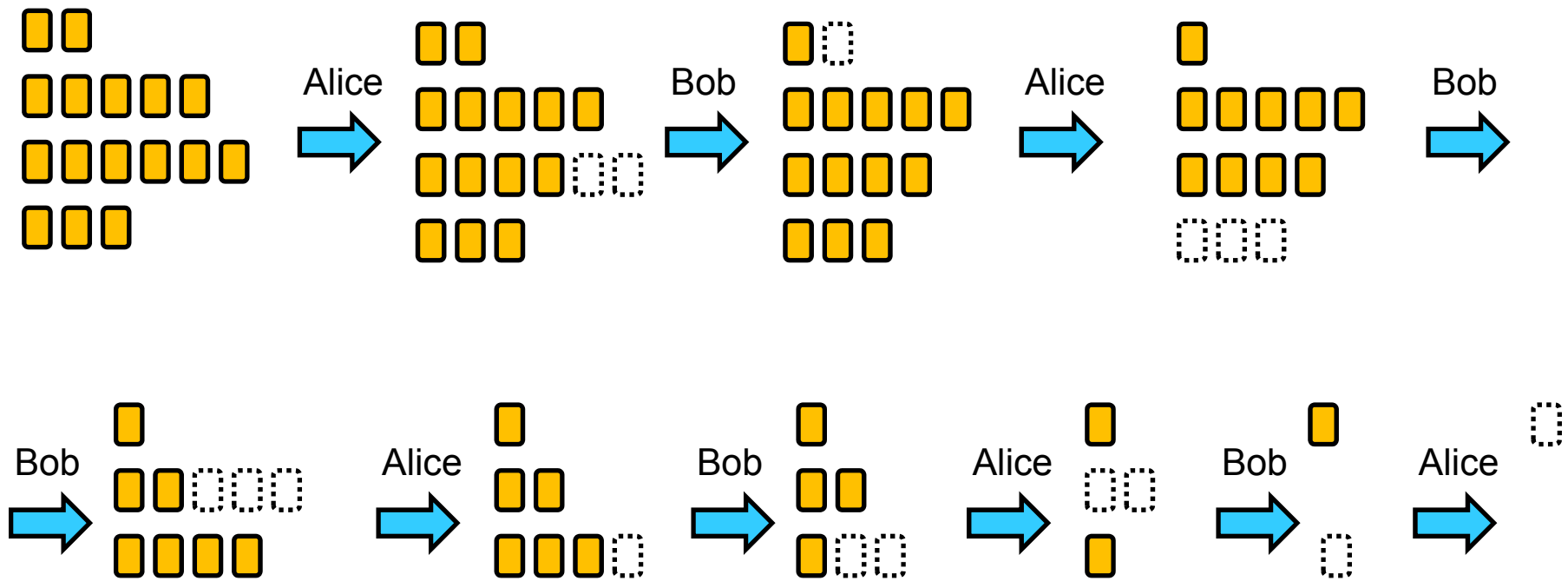
Why [the computer beats you](#) repeatedly at Nim game...





You can copy the javascript program of this **Nim game**... Feel free to modify or improve it! We would appreciate receiving any [comments](#) or suggestions concerning this script.

Try it yourself:

https://www.archimedes-lab.org/game_nim/play_nim_game.html

Normal game example, 4 piles, with 2, 5, 6, 3 stones.
 Alice plays the winning strategy.



Piles	Objects count	Expressed in binary
Pile 1 	13	1 1 0 1
Pile 2 	20	1 0 1 0 0
Pile 3 	27	1 1 0 1 1
Pile 4 	7	1 1 1

Nim-sum (Grundy value):

Bitwise XOR,
also, bitwise addition modulo 2
for each order (column) separately:

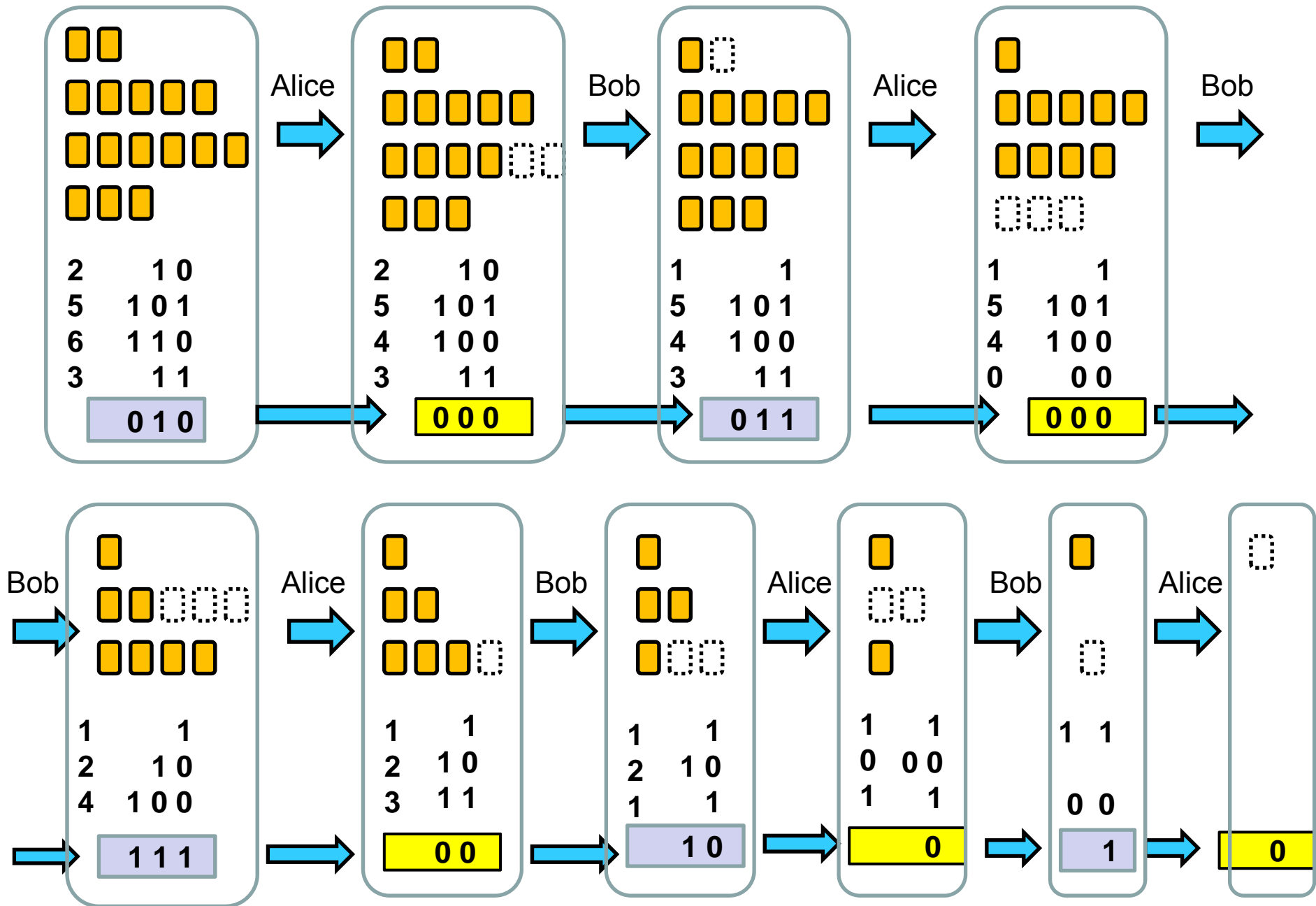
	1	1	0	1
1	0	1	0	0
1	1	0	1	1
		1	1	1
0	0	1	0	1





Fundamental Fact:

In normal game,
P-positions are exactly those with nim-sum = 0,
N-positions are exactly those with nim-sum ≠ 0.

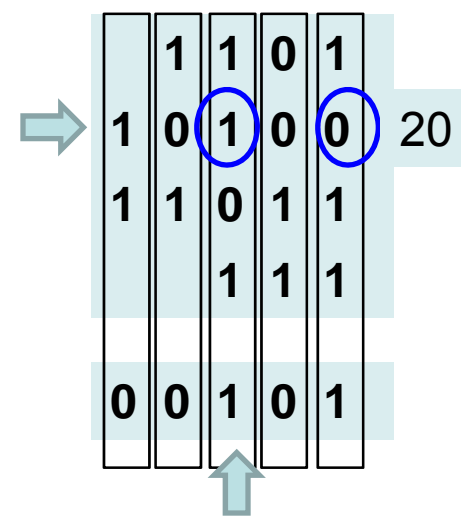
Winning strategy: Always move to the position with nim-sum = 0.

Normal game example repeated, with nim-sums.

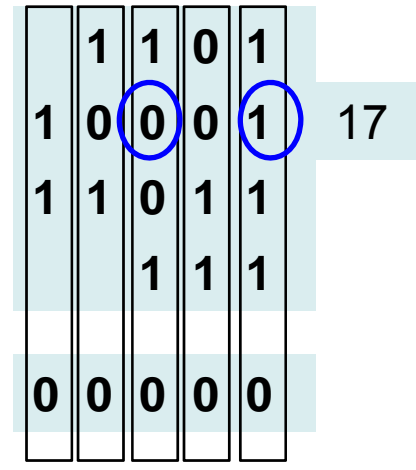


Piles	Objects count	Expressed in binary
Pile 1 	13	1 1 0 1
Pile 2 	20	1 0 1 0 0
Pile 3 	27	1 1 0 1 1
Pile 4 	7	1 1 1

Finding the right move to a P-position (= to the kernel) .



Take the column (order) corresponding to the highest order 1 in the nim-sum. Choose any pile which binary representation has 1 in that column. Flip those bits in this pile bin. repr. which order is the same as the orders of 1's in the nim-sum. (= XOR the pile with the nim-sum)



Remove 3 objects from pile 2

The new size of the chosen pile is smaller (it may even become empty). The difference between the old size and the new one is the number of objects to be removed from that pile in the current move.