後退解析を用いた完全情報七並べの解析

深川大路
同志社大学 文化情報学部
dfukagaw@mail.doshisha.ac.jp

七並べについて

- トランプ５２枚を使ったカードゲーム

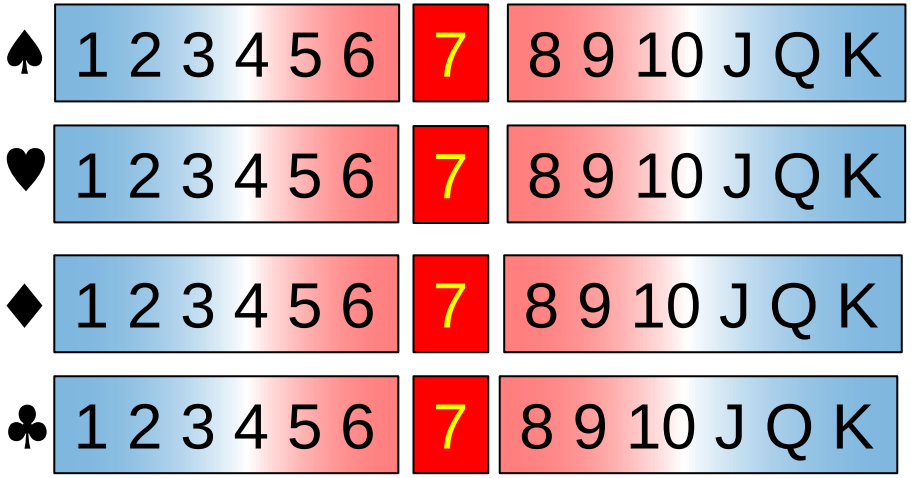
- 日本では非常にポピュラー
- 単純なルールで，子供から大人まで多人数で楽しめる

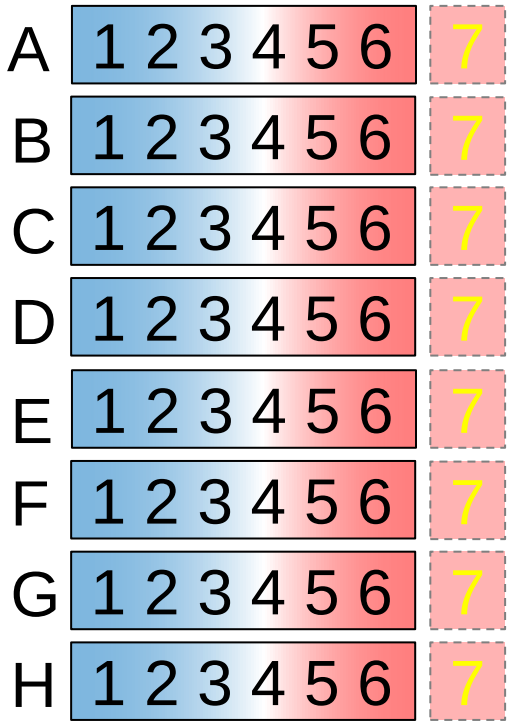
- 基本的なルール
  - 札を均等に配り，各プレイヤーの手札とする
  - １枚ずつ交代で札を場に出し，全ての札を４×１３ に整列させる
  - ただし，既に出ている札と隣り合う数字の札しか出せない
  - 早く手札をなくした者が勝ち

- ローカルルール（パス，トンネル，ジョーカーなど）は
  とりあえず考えないことにする

一般化七並べ

- 通常のトランプは4種のマーク，１～１３の数字
- これを8種のマーク，１～６の数字と考える
  - 7より小さいスペード♠と7より大きいスペード♤は独立
  - 7は最初から置いてあるものとする





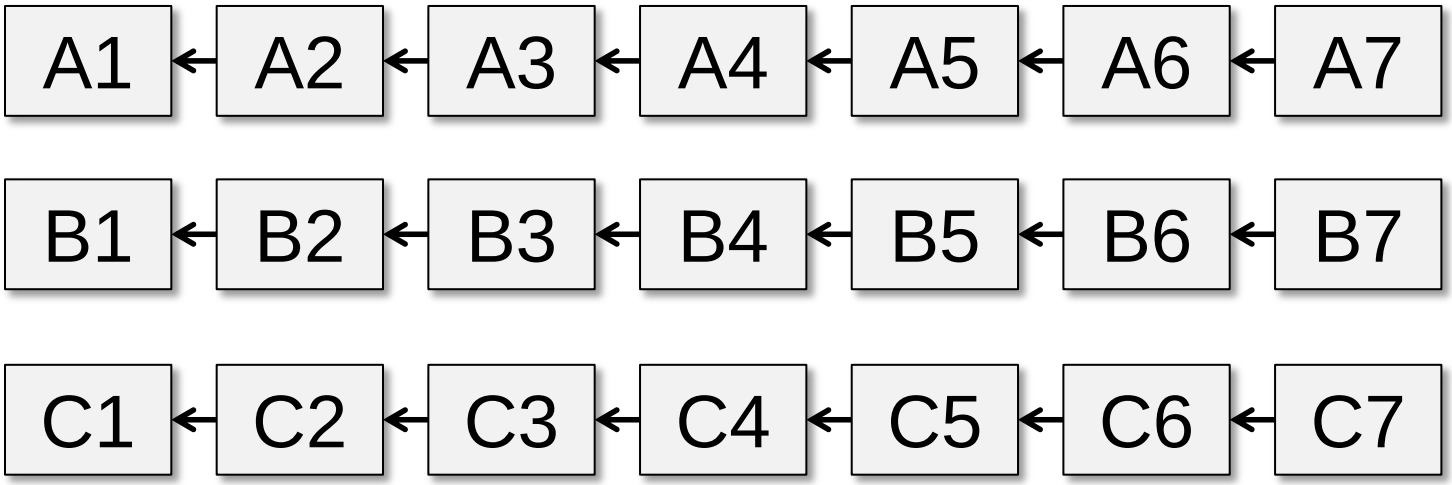
完全情報七並べ

- 通常の七並べ
  - 初期状態： 場は空である（7は出ており，１～６が空き）
  - 手番のプレイヤーは，手札から場に1枚出す
  - 手札が空になったら勝ち

- 完全情報七並べ
  - 札は全て場に出ている
  - 各札は色 $c \in \{1, \ldots, p\}$ を持っている （ただし p はプレイヤーの人数）
  - プレイヤーは自分の色の札を回収する
  - 回収できる条件は同じ（各列に残っているうちの最大の札）
  - 自分の色の札をすべて回収すれば勝ち

通常の七並べ

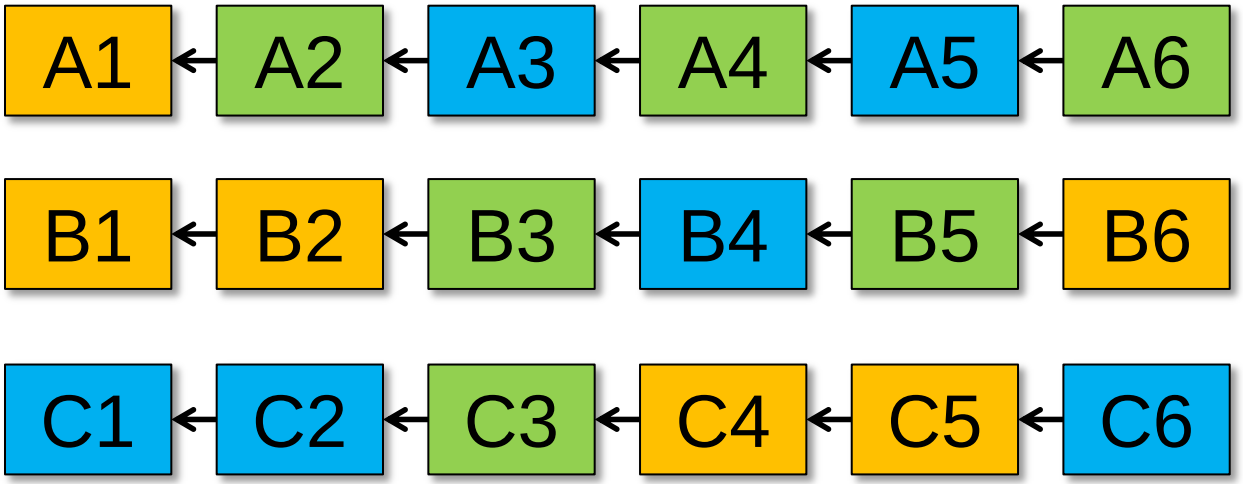
- 7は最初から置いてあるものとする



- プレイヤーは順番に1枚ずつ札を置く
- 先に手札をなくせば勝ち

完全情報七並べ

- 札は最初から置いてある（7は考えない）
- 各札はプレイヤーを表す色を持つ

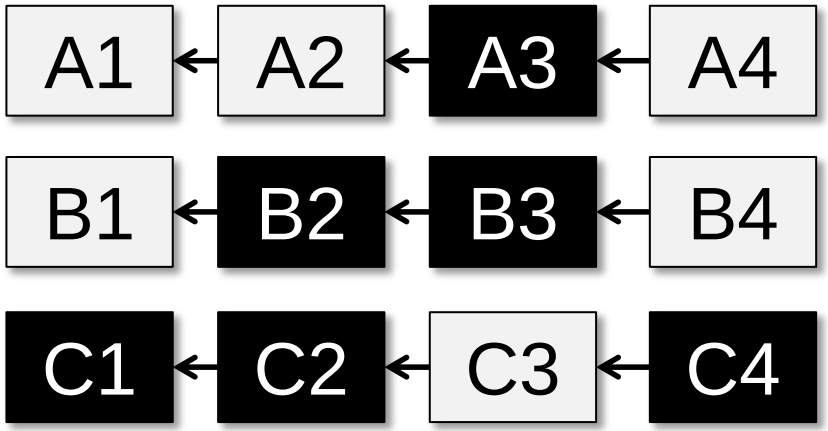


- 手番のプレイヤーは自分の色の札を回収する
- 先に自分の色の札をすべて回収した者が勝ち

完全情報一般化七並べの定式化

- パラメータ
  - M: 数字の種類数
  - N: マークの種類数
  - p: プレイヤーの人数

図. $(M, N) = (4,3)$ の例



- 以下の発表では，$p = 2$ に限定して考える
  - 黒札が先手，白札が後手

- 枚数は必ずしも均等でなくてよい（パスなどの影響も考慮）

決定問題としての七並べ

- 入力
  - パラメータ M, N, p
  - 札の色 $c_{mn} \in \{1, \ldots, p\}$
- 質問
  - 与えられたインスタンスは先手必勝か？
- どんなインスタンスも先手必勝もしくは後手必勝であることは明らか

完全情報一般化七並べの困難性

- 手数制限のある二人ゲームなので，高々 PSPACE 完全
- 予想： 多項式時間で解ける（？）
- かなり楽観的な予想
  - あまり理論的解析をせずに，計算機を回して意外な発見があるとよい
  - 計算機実験向きの問題規模で嬉しい

状態の数え上げ (1/3)

- 全部で何通りの状態があるか？

- とても単純に見積もると，全部で 3^{MN} 通り

  各位置ごとに {黒, 白, 空き}

- もう少し丁寧に，列ごとに考える
  - 各列は $L = 2^0 + 2^1 + \cdots + 2^M = 2^{M+1} - 1$ 通り
  - ある配置の任意の 2 列を交換して得られる配置は本質的に等価
  - 重複組合せより，全部で ${}_L H_N = \binom{L+N-1}{N}$ 通り

状態の数え上げ (2/3)

- 列ごとの重複組合せによる数え上げ
- N: 列の数, M: 各列の最大長

N＼M	1	2	3	4	5	6
1	3	7	15	31	63	127
2	6	28	120	496	2,016	8,128
3	10	84	680	5,456	43,680	349,504
4	15	210	3,060	46,376	720,720	11,358,880
5	21	462	11,628	324,632	9,657,648	297,602,656
6	28	924	38,760	1,947,792	109,453,344	6,547,258,432
7	36	1,716	116,280	10,295,472	1,078,897,248	124,397,910,208
8	45	3,003	319,770	48,903,492	9,440,350,920	2,083,664,995,984

状態の数え上げ (3/3)

- いわゆる七並べは $(M, N) = (6,8)$ のとき
- 簡潔に表現でれば 41 ビットで表す事が出来る
  - $2^{41} \approx 2 \times 10^{12} > 2{,}083{,}664{,}995{,}984$

- 実際には，通常のデータ構造で表現
  - 各状態を $(M+1)N$ で表現
  - $(M, N) = (6,8)$ のとき 56 ビット

データ構造

- 長さ M 以下の札列を $M+1$ ビットのブロックで表現
  - 黒を 0, 白を 1 で表現
  - 長さを表すために M+1 ビット目を 1 に固定
  - [黒,白,白,黒] → $10110_{(2)}$
  - [黒] → $10_{(2)}$
  - [] → $1_{(2)}$
- 札列を表すブロックを N 列分ならべて各状態を表現
- 各状態を $(M+1)N$ ビット整数で表現
- 札列を任意に入れ替えても（例えばクラブとスペード）インスタンスは等価であるため，正規型のみを扱えばよい

状態の列挙と勝敗判定

- 全ての列が空である状態 [[],[],…,[]] から開始して，１枚ずつ札を追加する操作を繰り返すことで全ての状態を列挙できる
- 空の状態を「勝ち型」として，後退解析によって全ての状態の勝敗を判定できる
- (M,N)=(6,8) のとき
  - 全状態を記憶するには 16TB
  - 勝敗表を記憶するには 2TB
  - 実際は，正規型のみ記憶すればよいため，もっと少なく済むが・・・
- 実は，七並べに特化した系統的な列挙も可能で，そちらの方がかなり高速
- 残り枚数 X 枚の勝敗判定に必要なのは残り枚数 X-1 枚の勝敗表のみ（パスは別途扱う）であるから，メモリもかなり削減できる

必勝法の導出に向けて

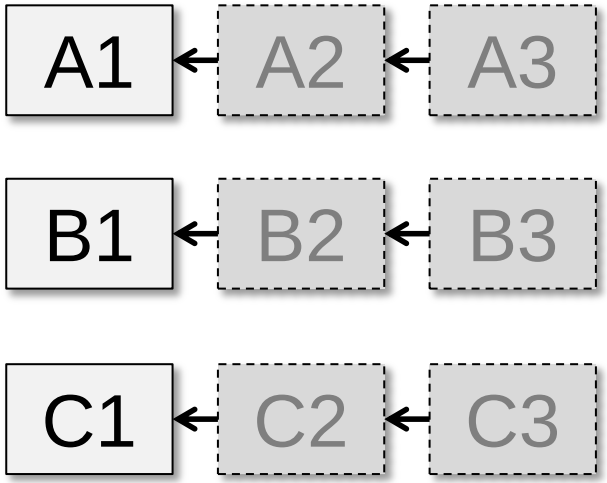
- 勝敗表を作成できるようになった
  - ディスク不足により，(M,N)=(6,8) は保留
- いわゆる「必勝法」とよばれている戦略を評価できる
- 実験結果を眺めて活用できないか？

考察

自明な場合（1）

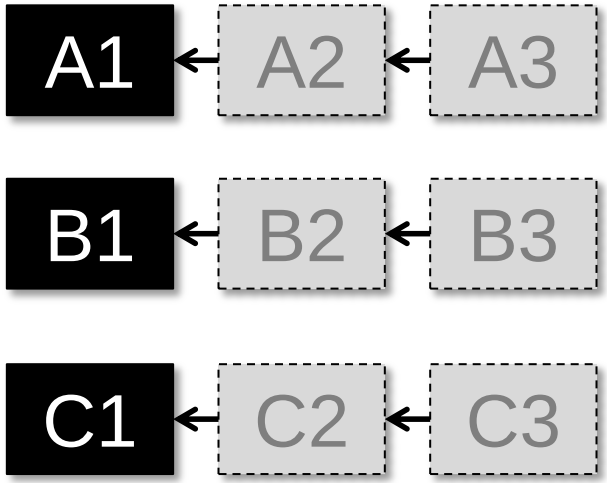
- 各列の最後のカードを片方のプレイヤーが持っている場合は，そのプレイヤーの負け

Case 1:



各列最後のカードは後手
⇒**先手（黒）勝**

Case 2:



各列最後のカードは先手
⇒**後手（白）勝**

自明な場合(2):残り1列

- 残り 1 列しかない場合

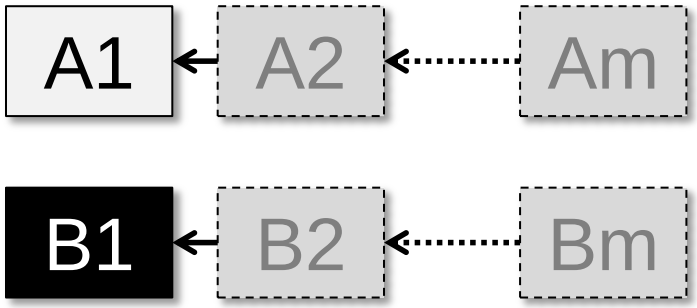
最後のカードは後手
⇒**先手(黒)勝**

最後のカードは先手
⇒**後手(白)勝**

自明な場合(3):残り2列

- 残りがちょうど 2 列の場合

- 自明な場合を除去すると，最後のカードを持つプレイヤーは先手と後手に分かれる

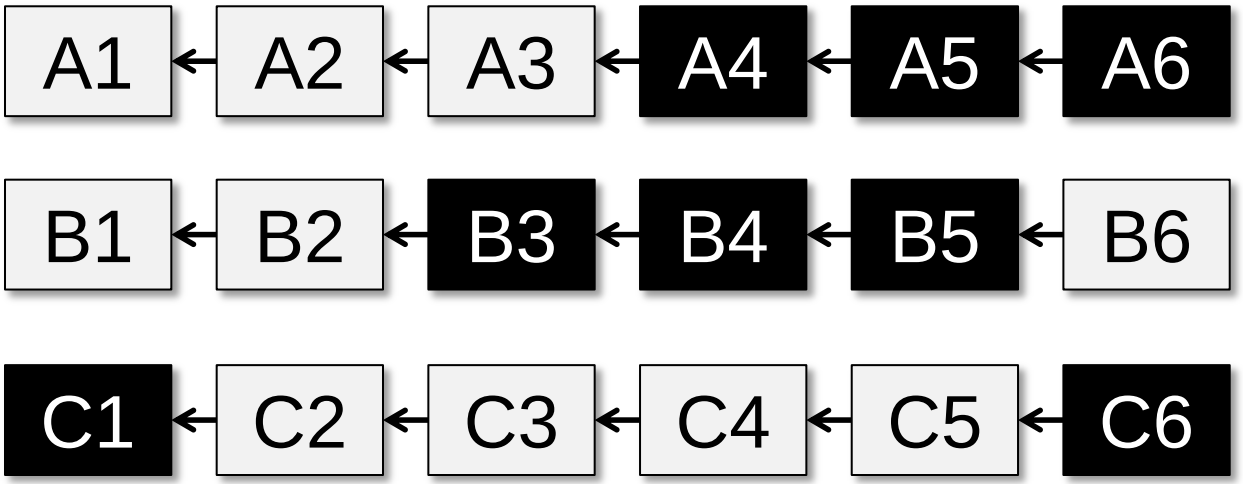


- **自分が最後のカードを持っている方の列を優先して進めるのがベストな戦略(要証明) → 「自列優先」戦略とよぶ**

列数による場合分け (3/*)

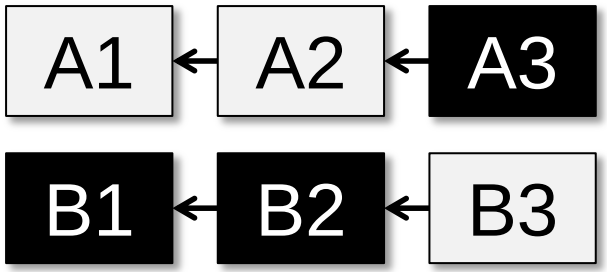
- 残りがちょうど 3 列の場合
- 2 列の場合と同じ自列優先戦略は通用しない
- 反例あり（次頁）

自列優先戦略が最適でない例



Zugzwang (ツークツワンク)

- 相互ツークツワンク



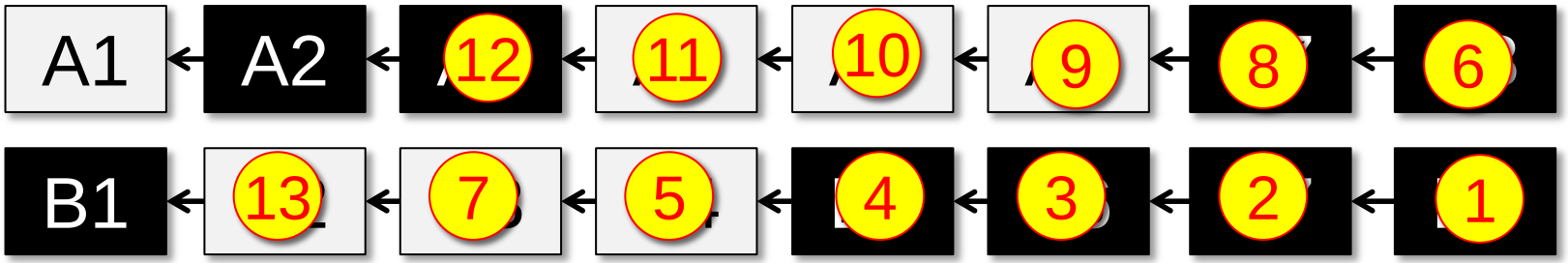
プレイヤー1 = {B1, B2, A3}
プレイヤー2 = {A1, A2, B3}

- 黒先なら A3 にしか出せず，黒の負け
- 白先でも同様（対称性より）

- Zugzwang の存在→局所的な「負けるが勝ち」の存在

例題1

- 黒が先手

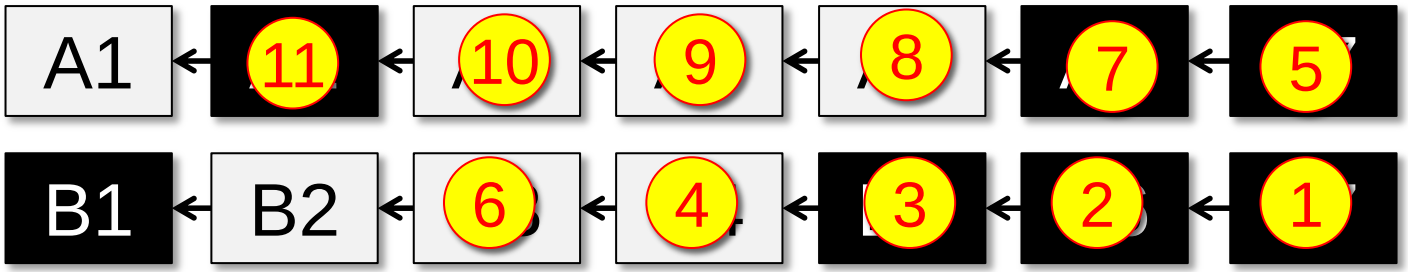


2列の場合は自列優先戦略が最適

- 先手（黒）の勝ち

例題2

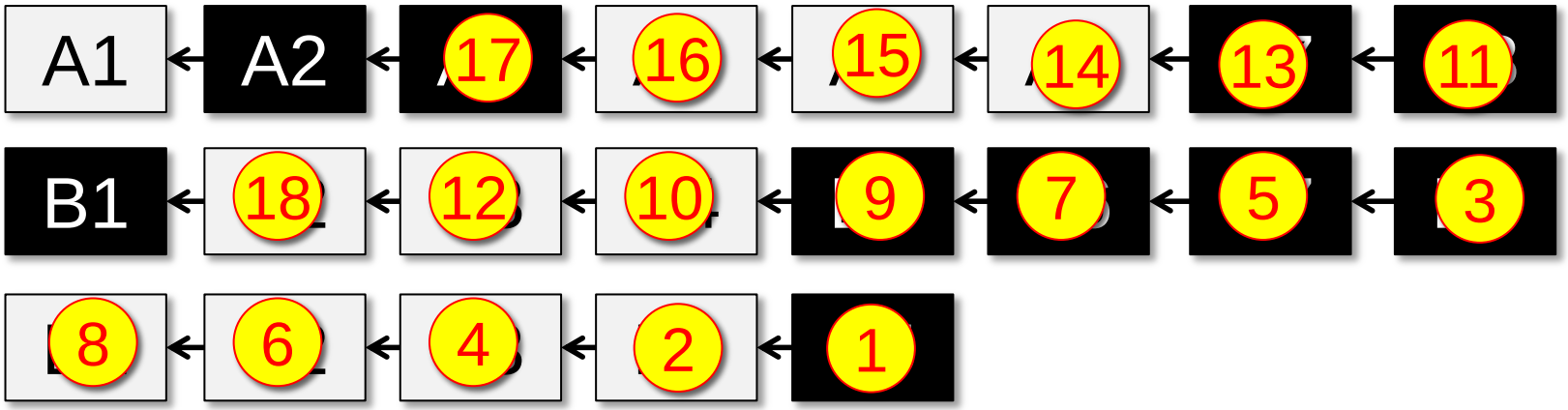
- 黒が先手



2列の場合は自列優先戦略が最適

- 後手（白）の勝ち

例題3



「相手が出せる札の数を増やさない」は
必ずしも正しい戦略ではない

まとめと今後の課題

- まとめ
  - 計算機を用いて完全情報一般化七並べの勝敗表の作成を試みた
    - (M,N)=(6,8) は未完（列挙はできても保存する領域が無い）

- 今後の課題
  - 残り3列（以上）の場合を解決したい
  - 興味深いインスタンスを探して考察
  - データマイニングによる必勝法の導出
    - 勝敗表から必勝法を機械的に導出
  - 高速化（必勝法が必要なインスタンスの割合はかなり少ない）