Games, geometry, and
[the computational complexity of]
finding equilibria

Bernhard von Stengel

Department of Mathematics
London School of Economics

Overview

An introduction to computational complexity, two open problems and a result on equilibrium computation:

- solving simple stochastic games

- finding a Nash equilibrium of a bimatrix game:

  long Lemke-Howson paths

  [joint with Rahul Savani]

Computational complexity

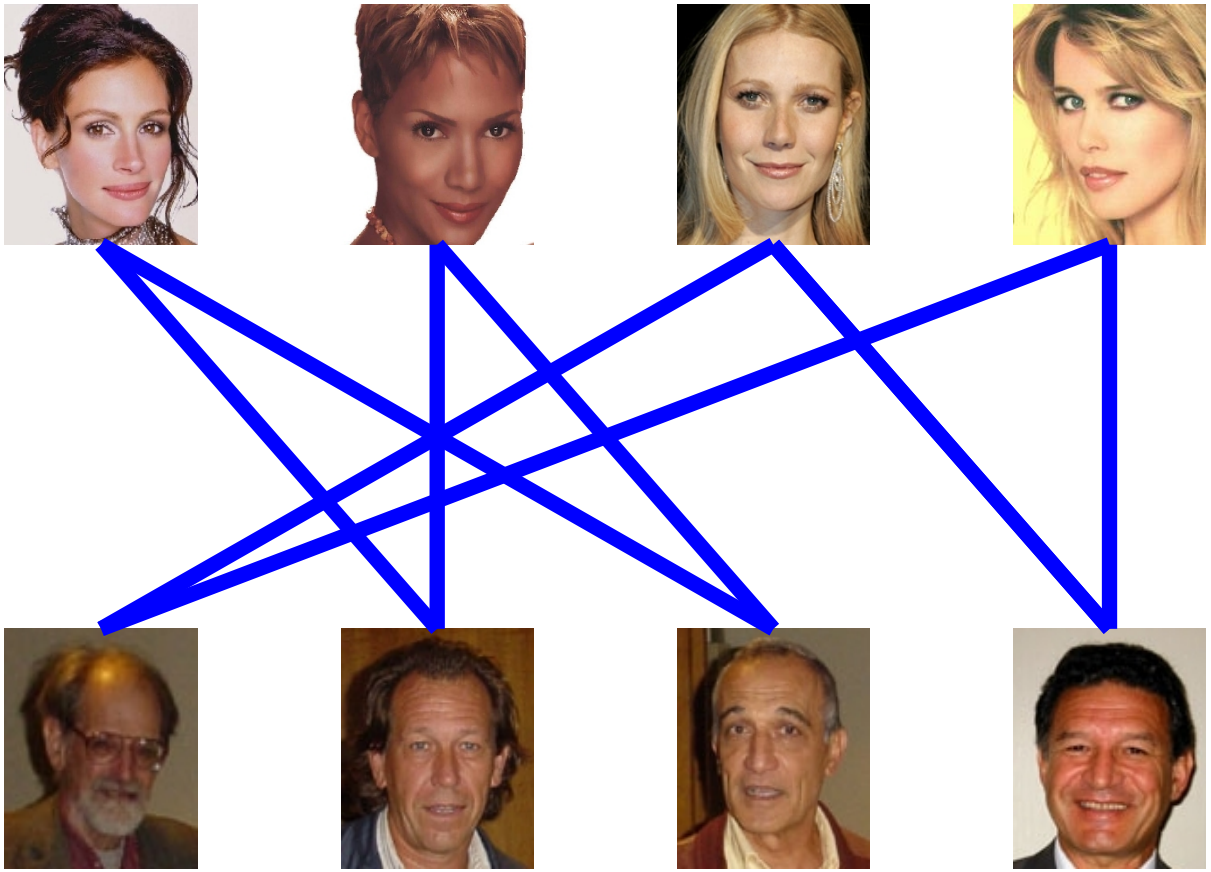
Computational complexity of a problem =
running time as function of input size n
(n = bits required to specify problem instance).

Decision problems = decide "yes" or "no"

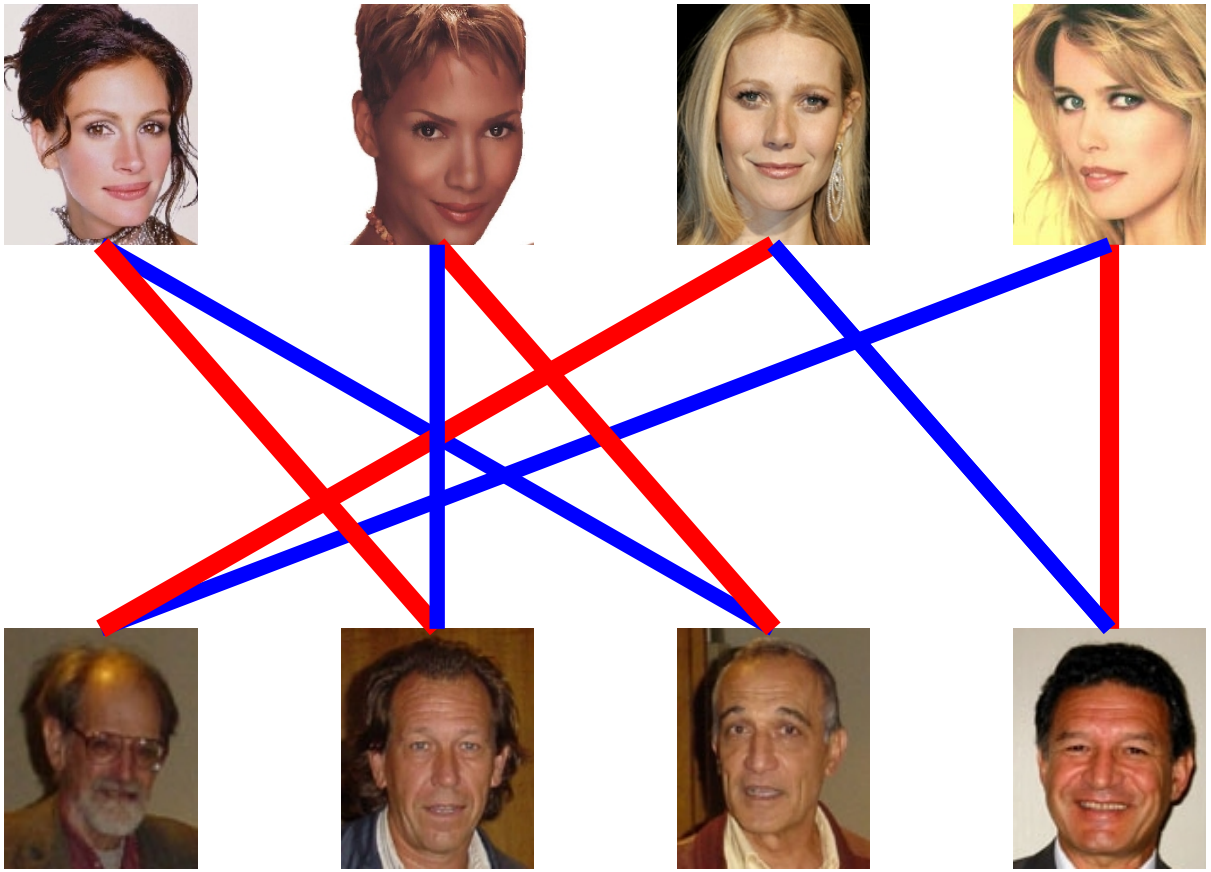
Example:

Perfect matching
Input: Bipartite graph G (list of edges).
Question: Does G have a perfect matching?

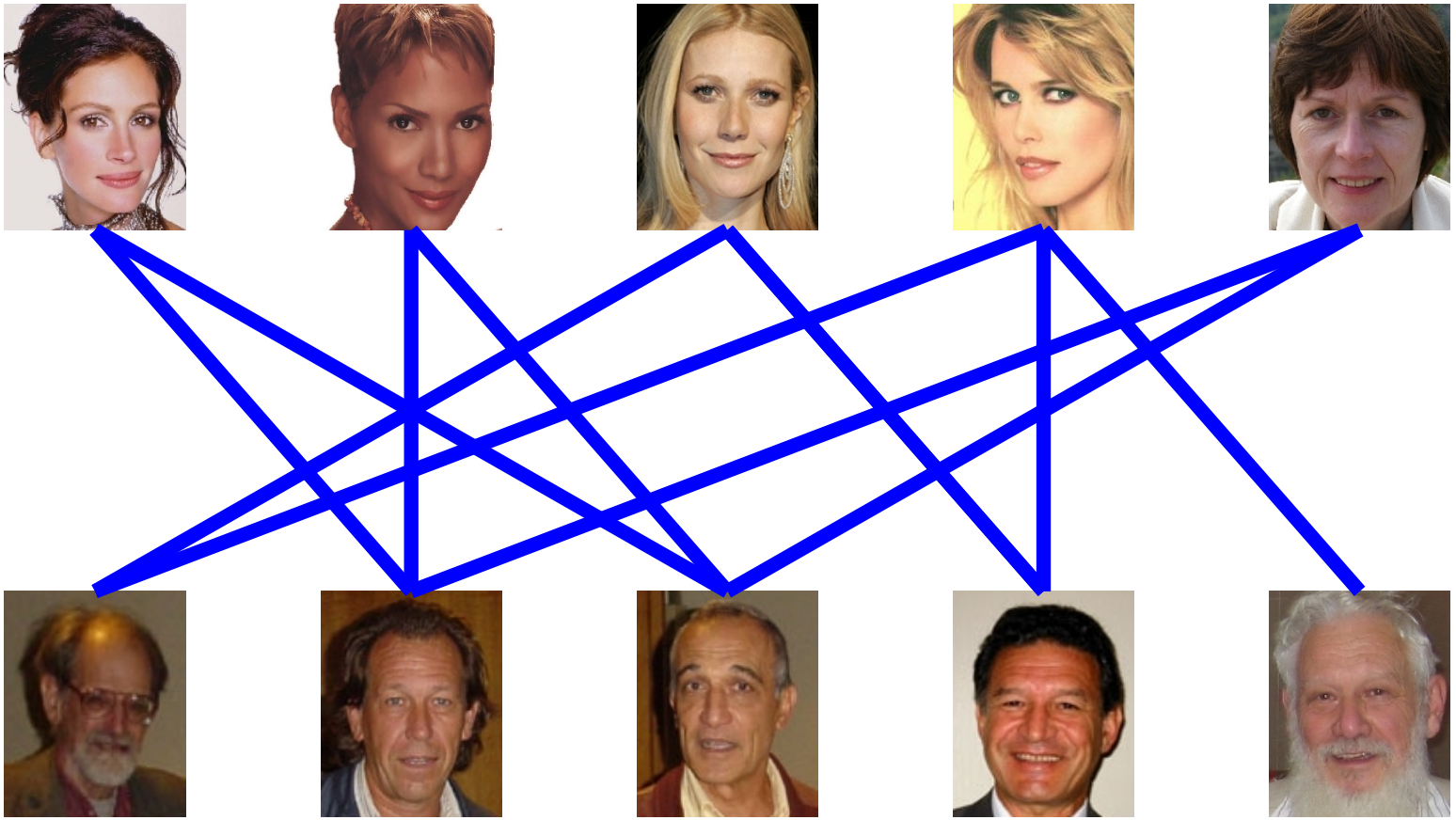
Finding a perfect matching



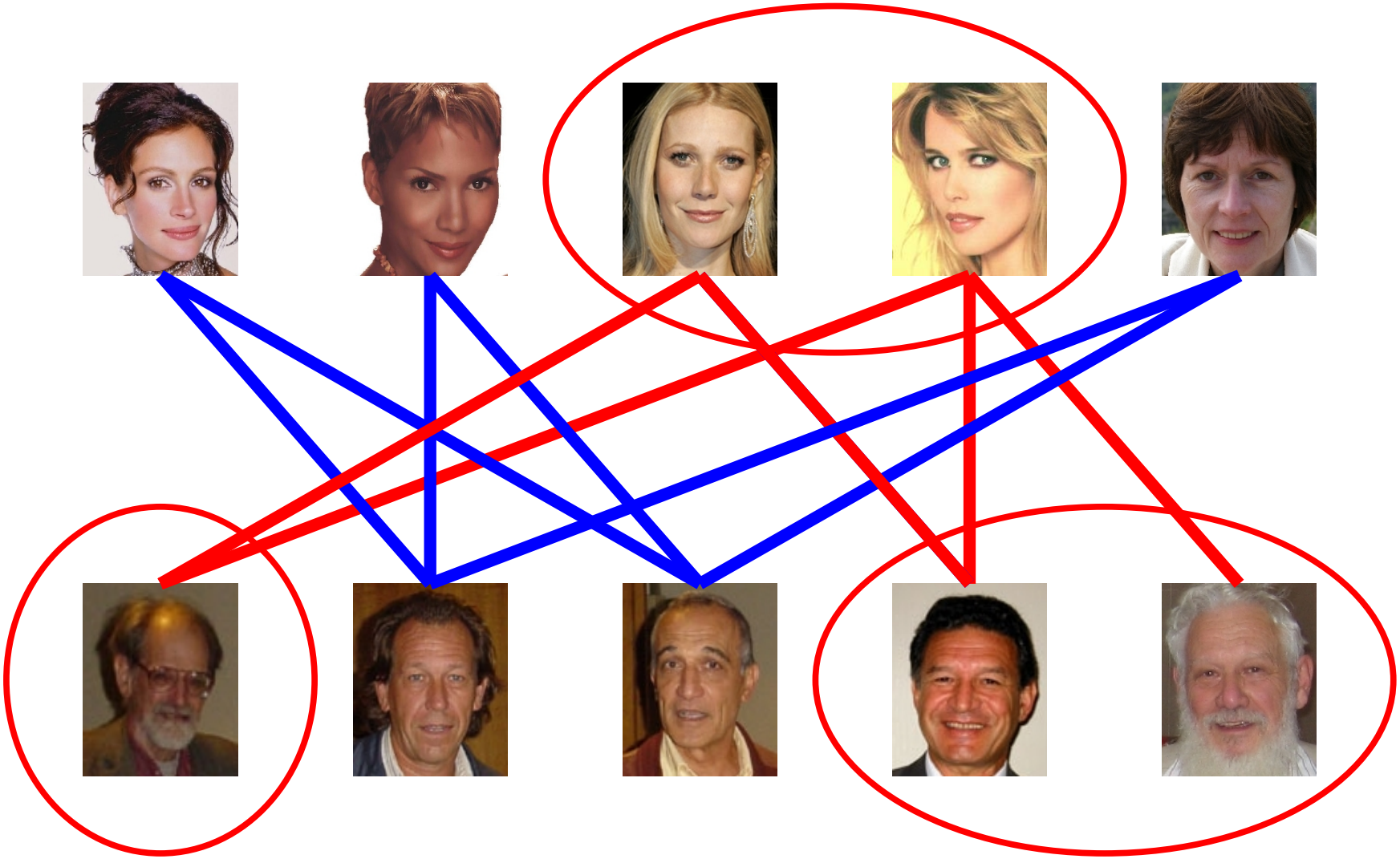
Finding a perfect matching



Perfect matching?



"no" - certificate



The complexity class NP "nondeterministic polynomial time"

A decision problem belongs to **NP**
if a "yes" answer can be verified quickly,
that is, in polynomial time with the help of a
nondeterministically found short "certificate".

Example:

Perfect matching $\in$ **NP**,
("yes"-)certificate = set of matched edges.

The complexity class co-NP

A decision problem belongs to **co-NP**
if a "no" answer can be verified quickly.

Example:

Perfect matching $\in$ **co-NP**,
"no"-certificate = node set A so that

$$|\{\ b : a \in A,\ \ (a,b) \text{ is an edge } \}| < |A| \ ,$$

which exists if the graph has no perfect matching
by Hall's marriage theorem.

The complexity class P
"polynomial time"

A decision problem belongs to **P**

if it can be decided ("yes" or "no") quickly,

that is, in polynomial time $O(n^k)$ for input size n,

for some constant k.

(Typically small k, e.g. $k=3$).

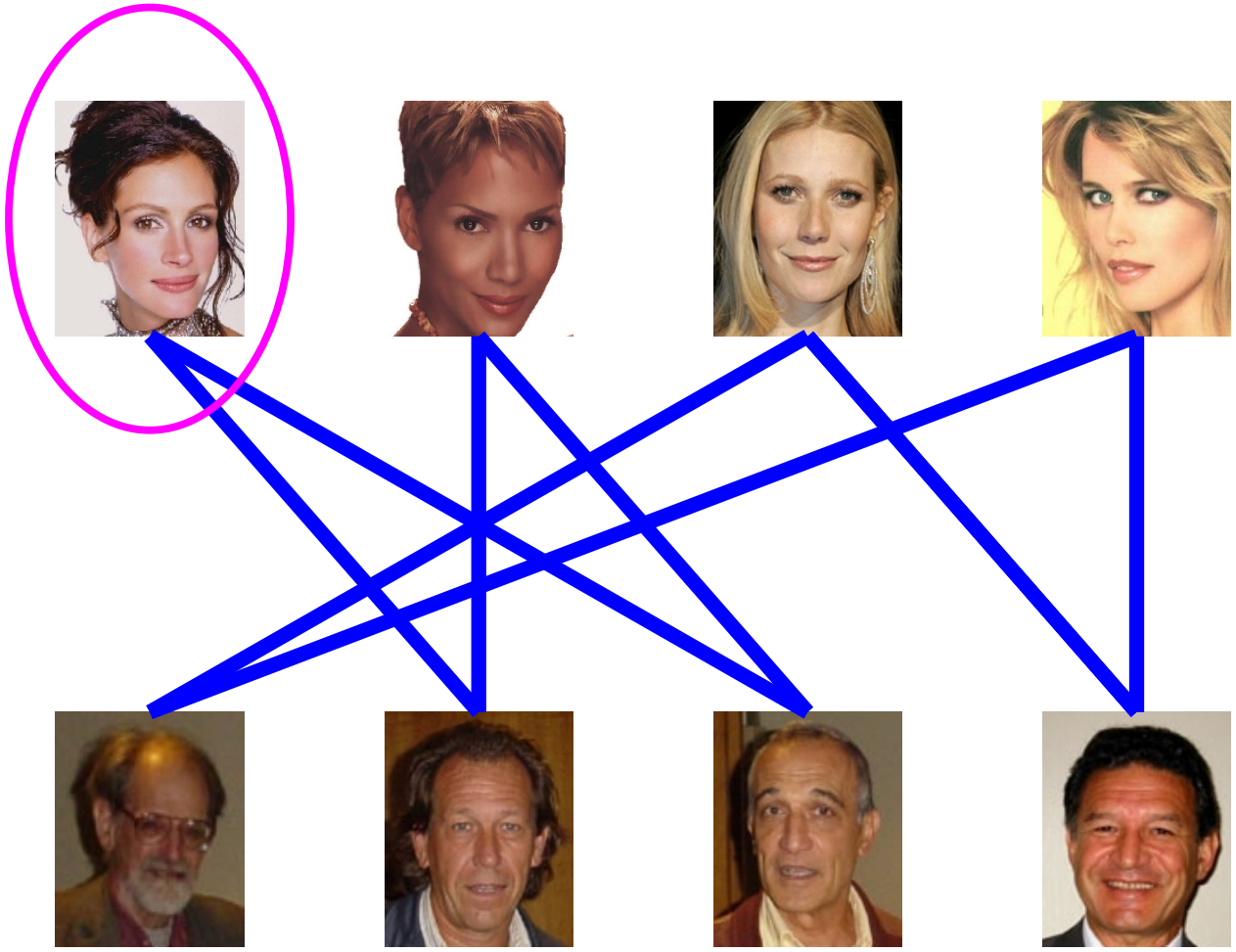
Perfect matching is in P

Input: Bipartite graph with n nodes on each side.
Output: Perfect matching, or "no"-certificate A.

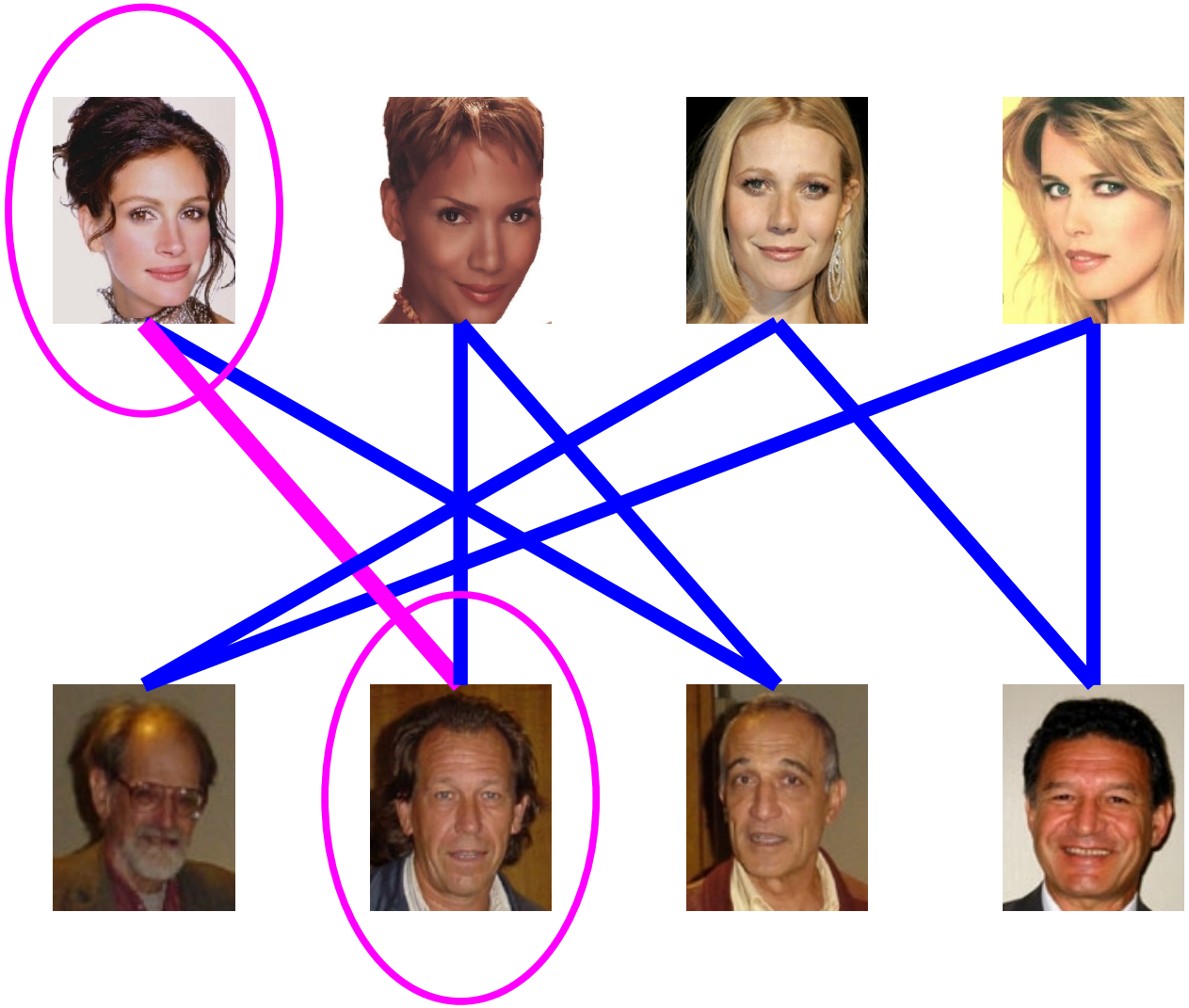
Augmenting-path algorithm:

	time
for each unmatched node v	$\times n$
- find **augmenting path** from v - flip matched and unmatched edges	$O(n^2)$
overall:	$O(n^3)$

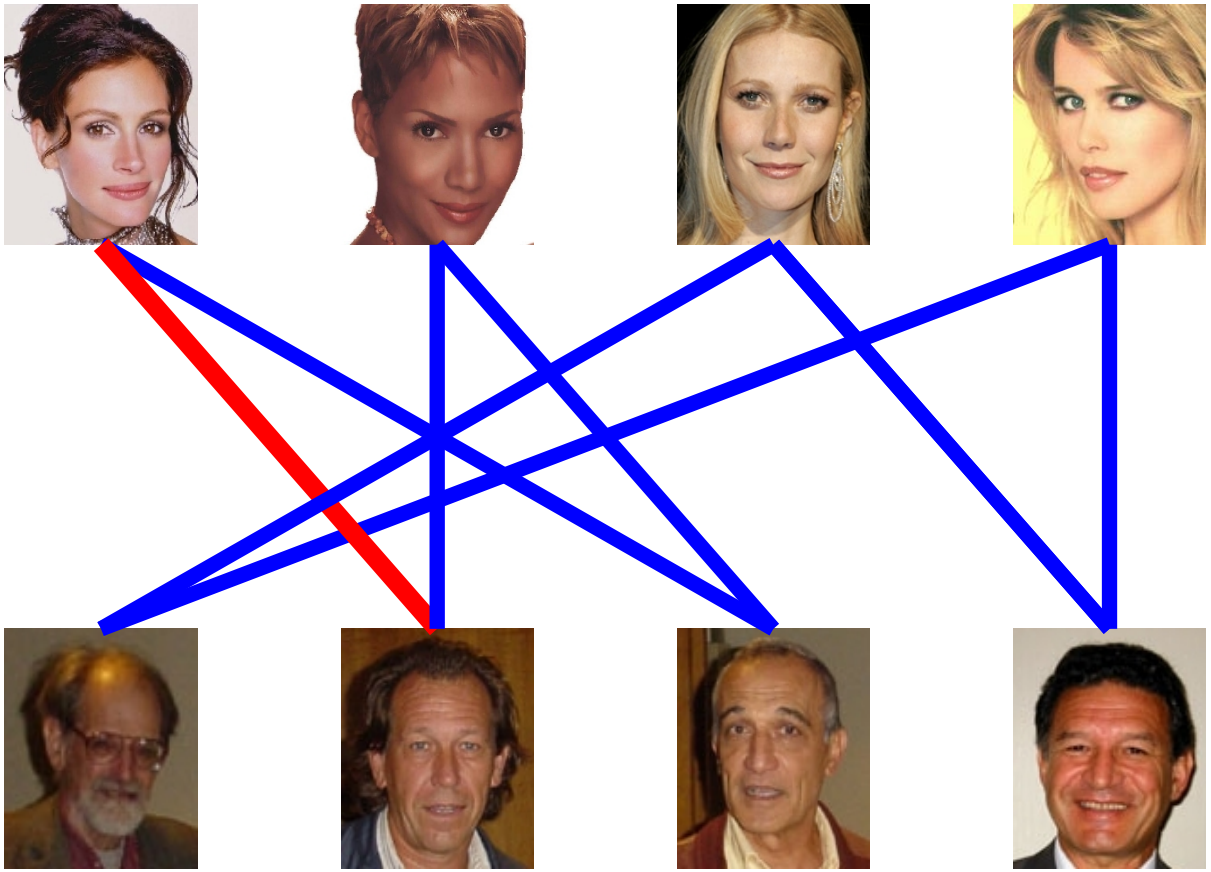
Start with unmatched node



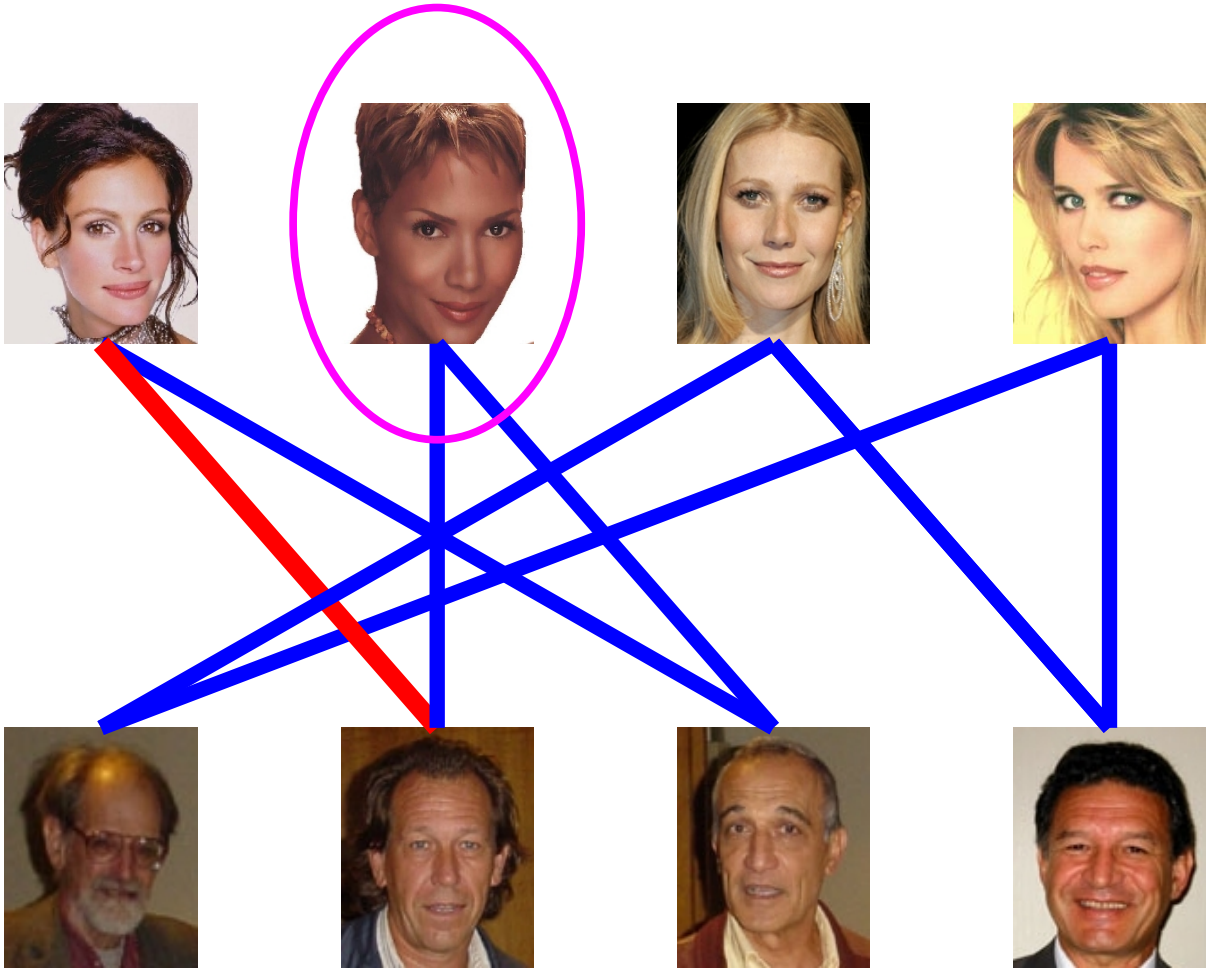
Find unmatched node



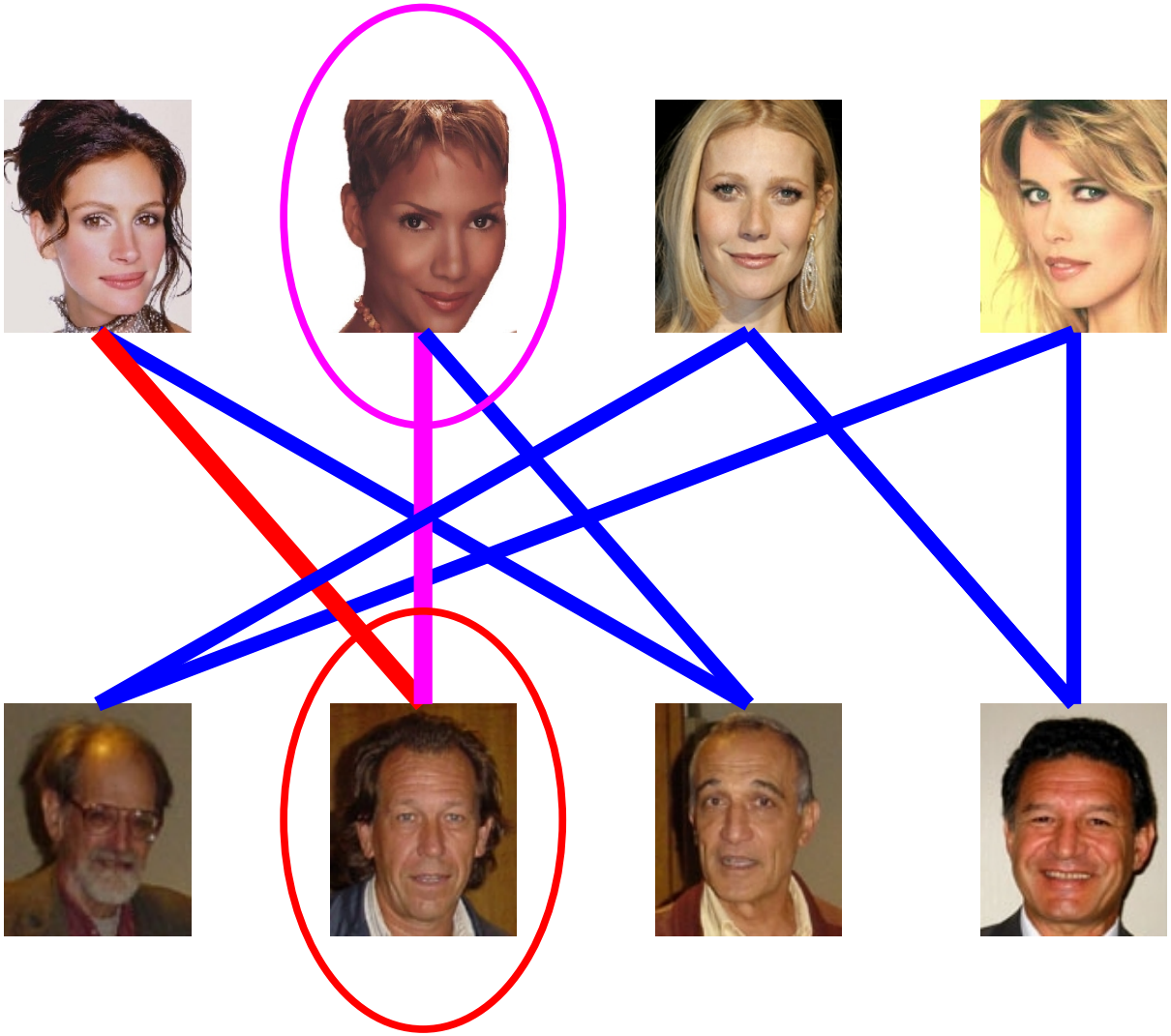
Get matched edge



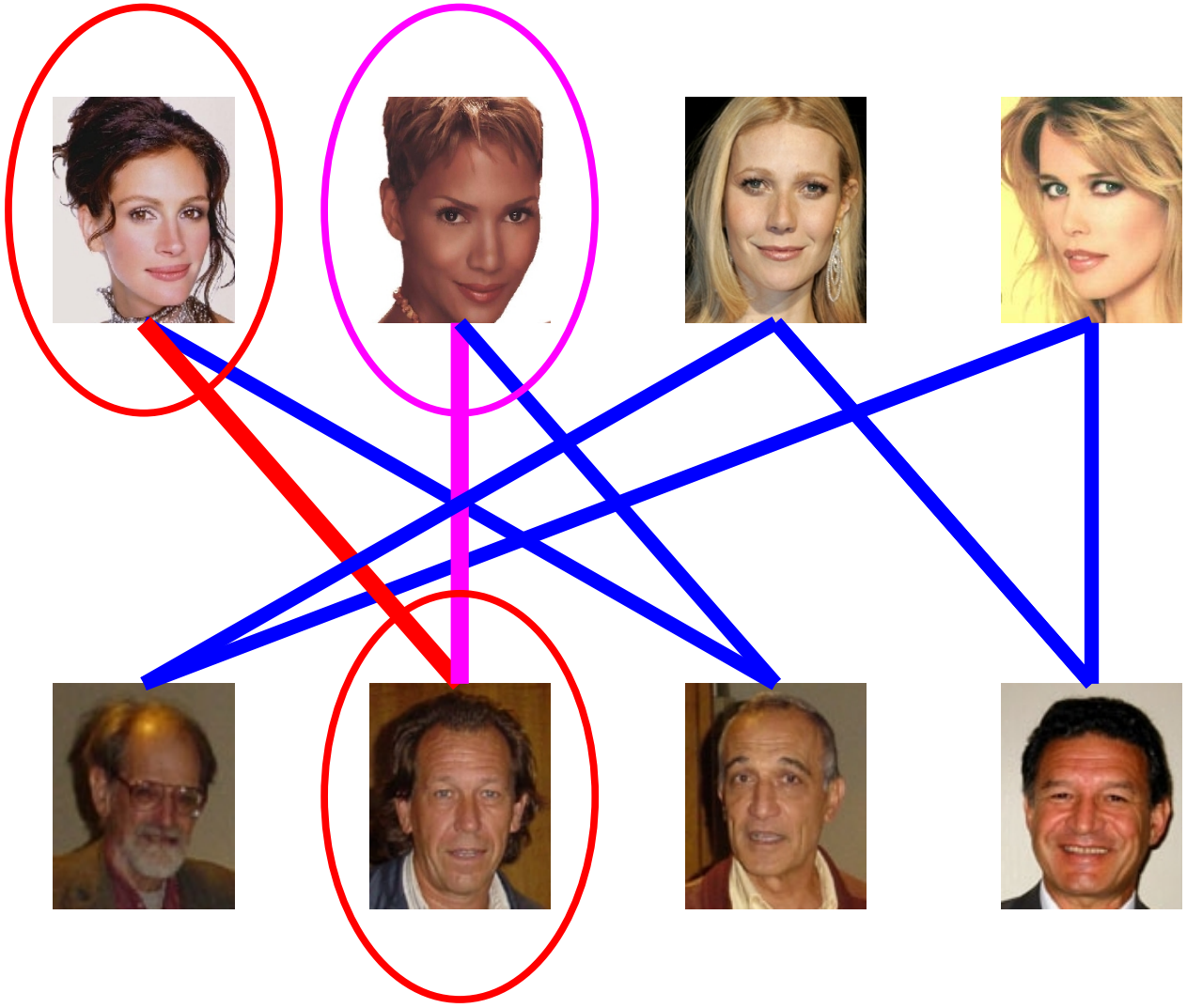
Next unmatched node



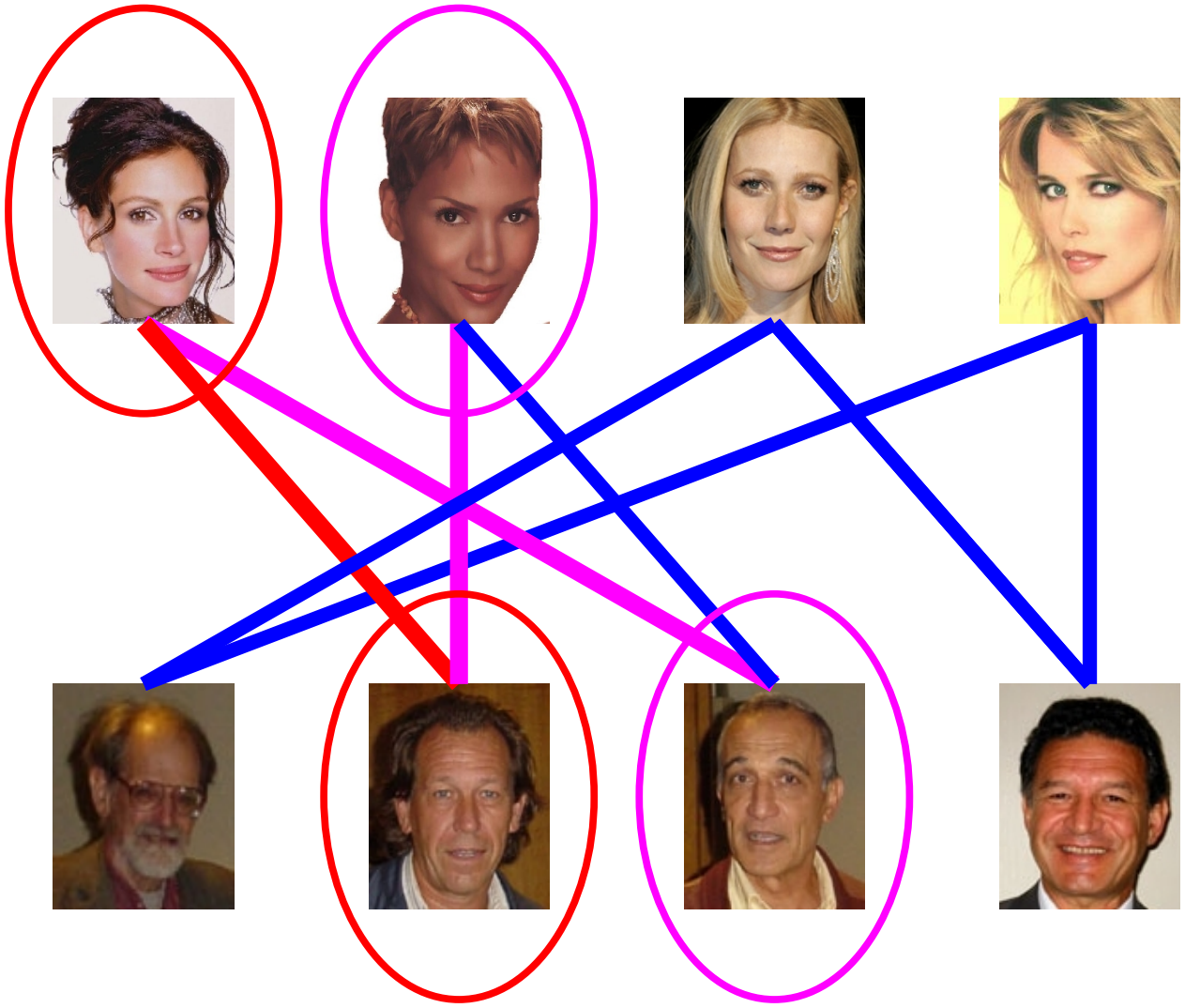
Find matched node



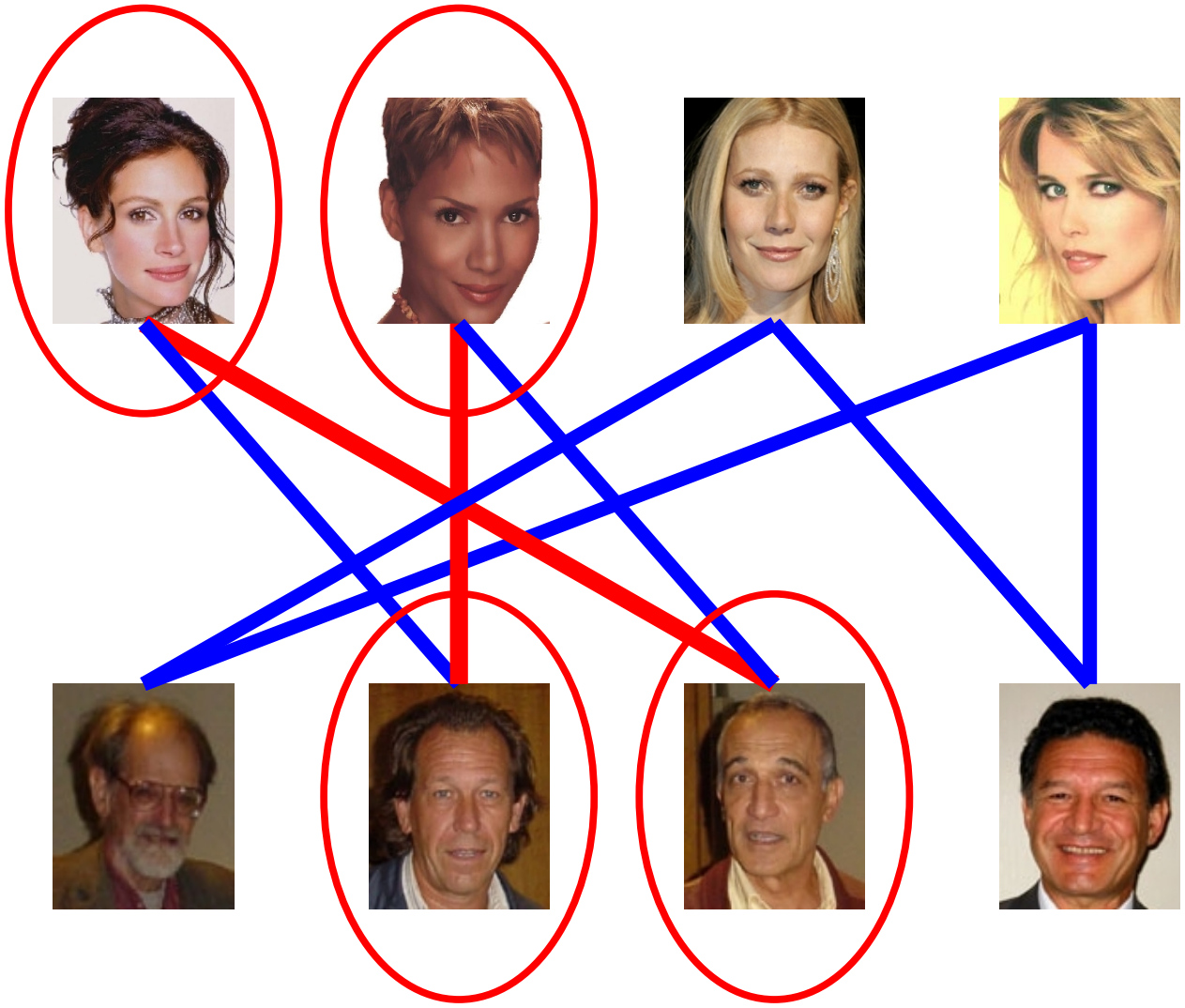
Go back along matched edge



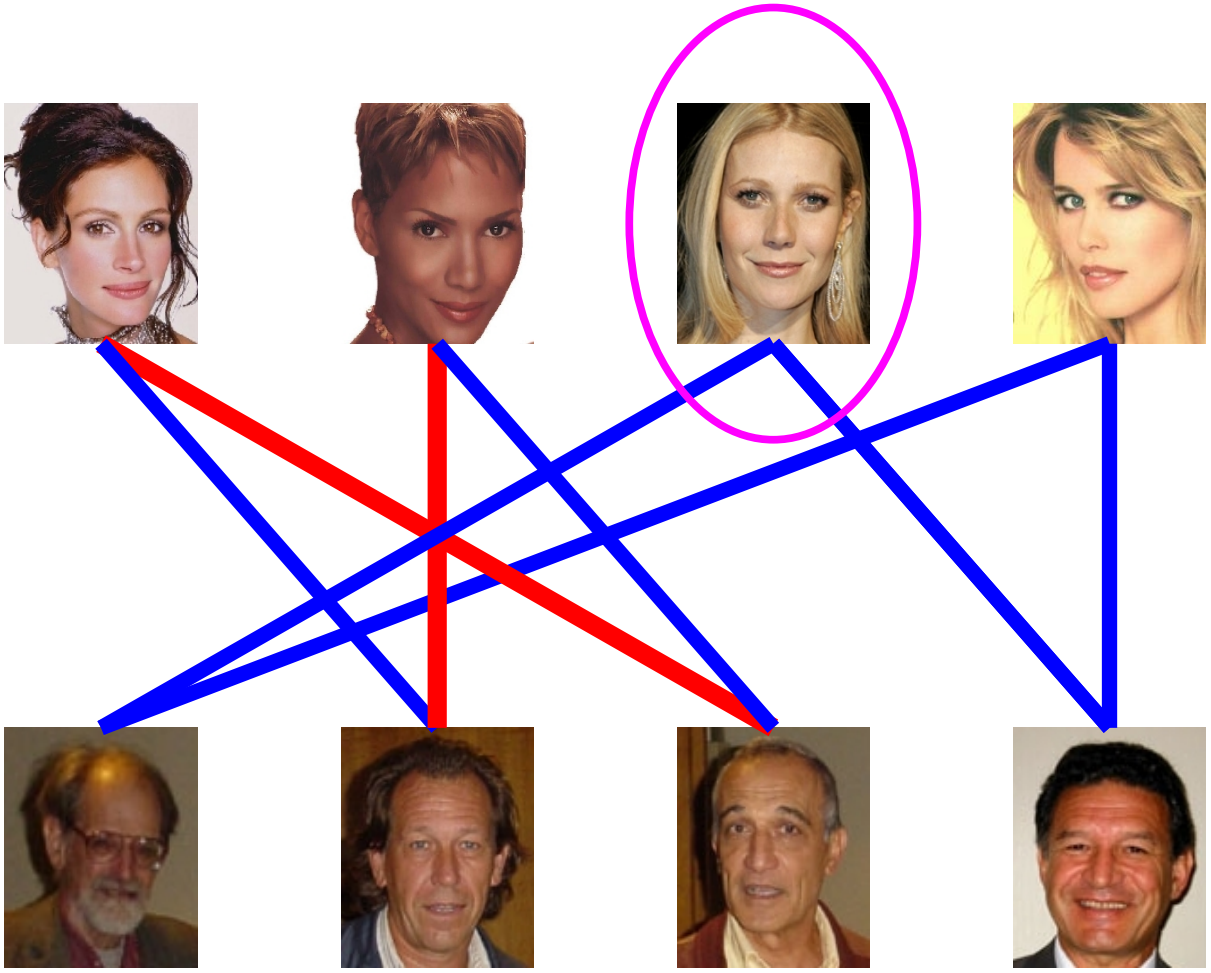
Find unmatched node



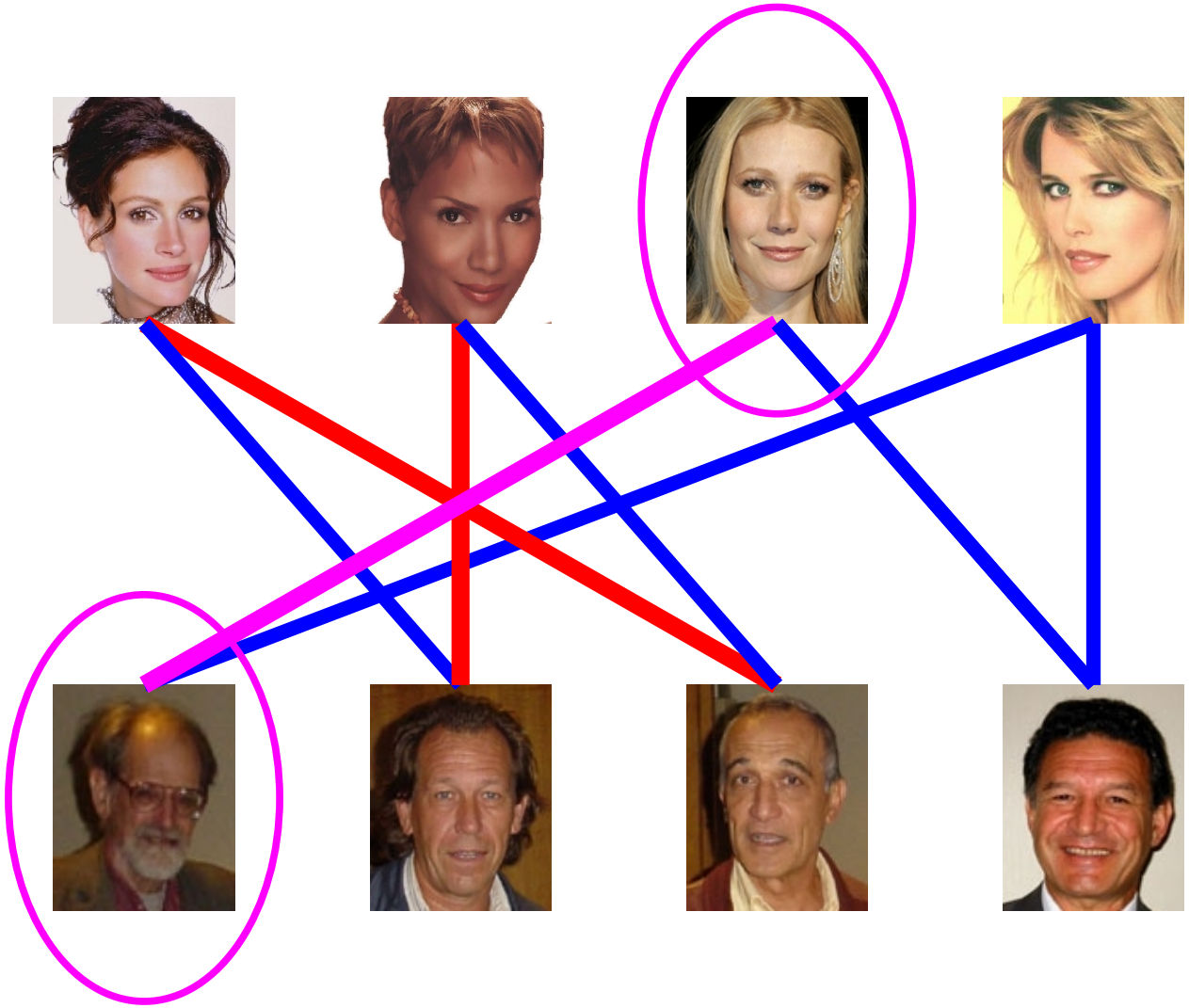
Flip edges



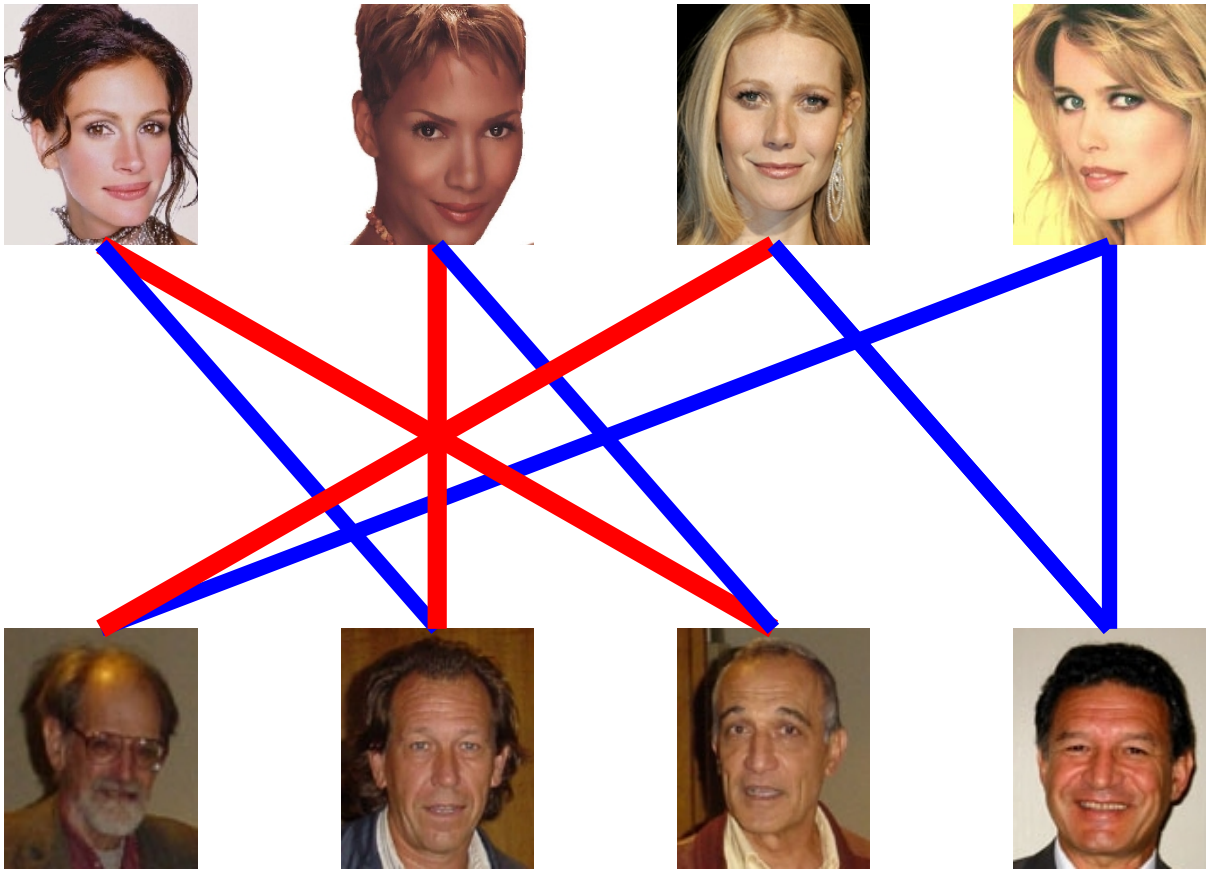
Next unmatched node



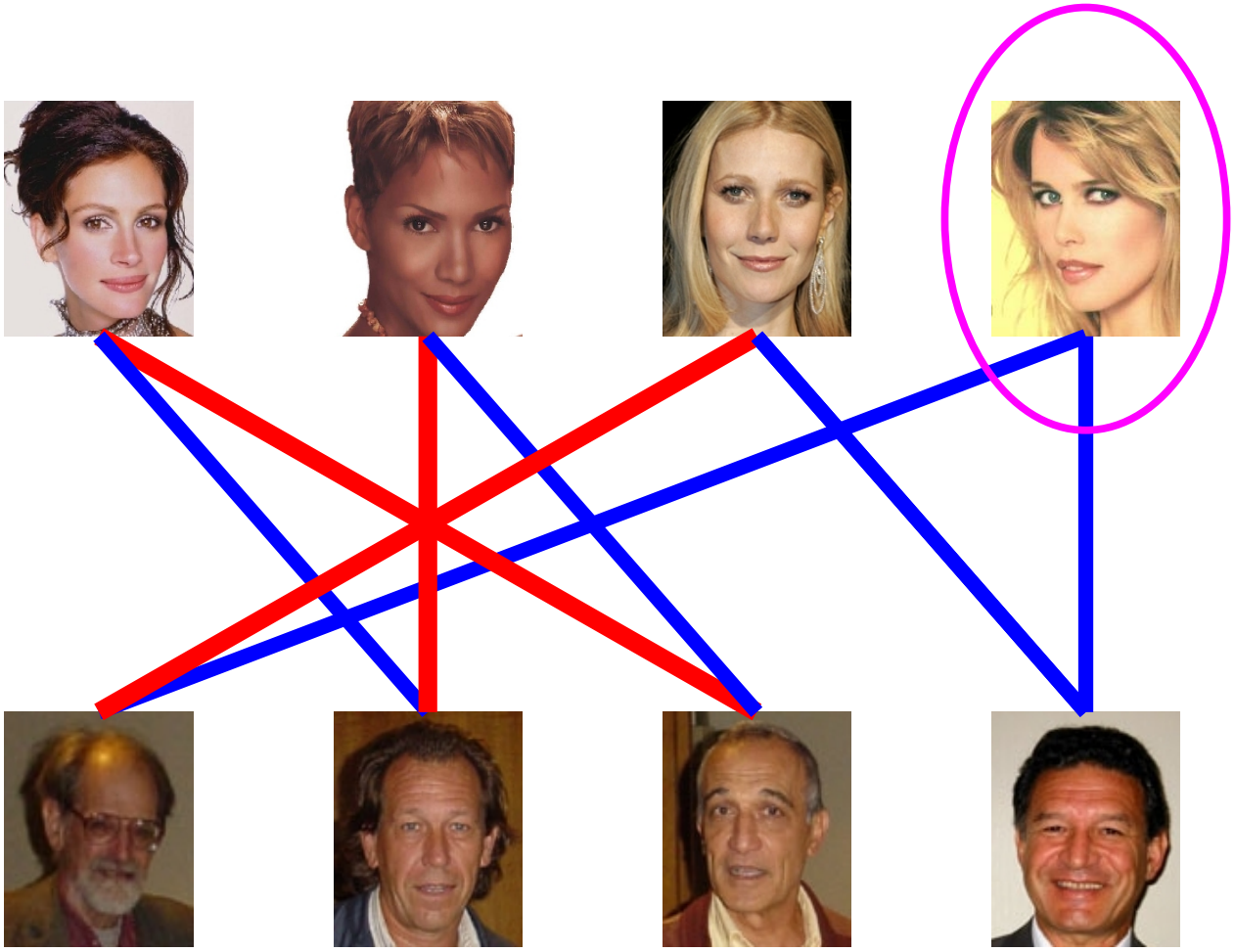
Find unmatched node



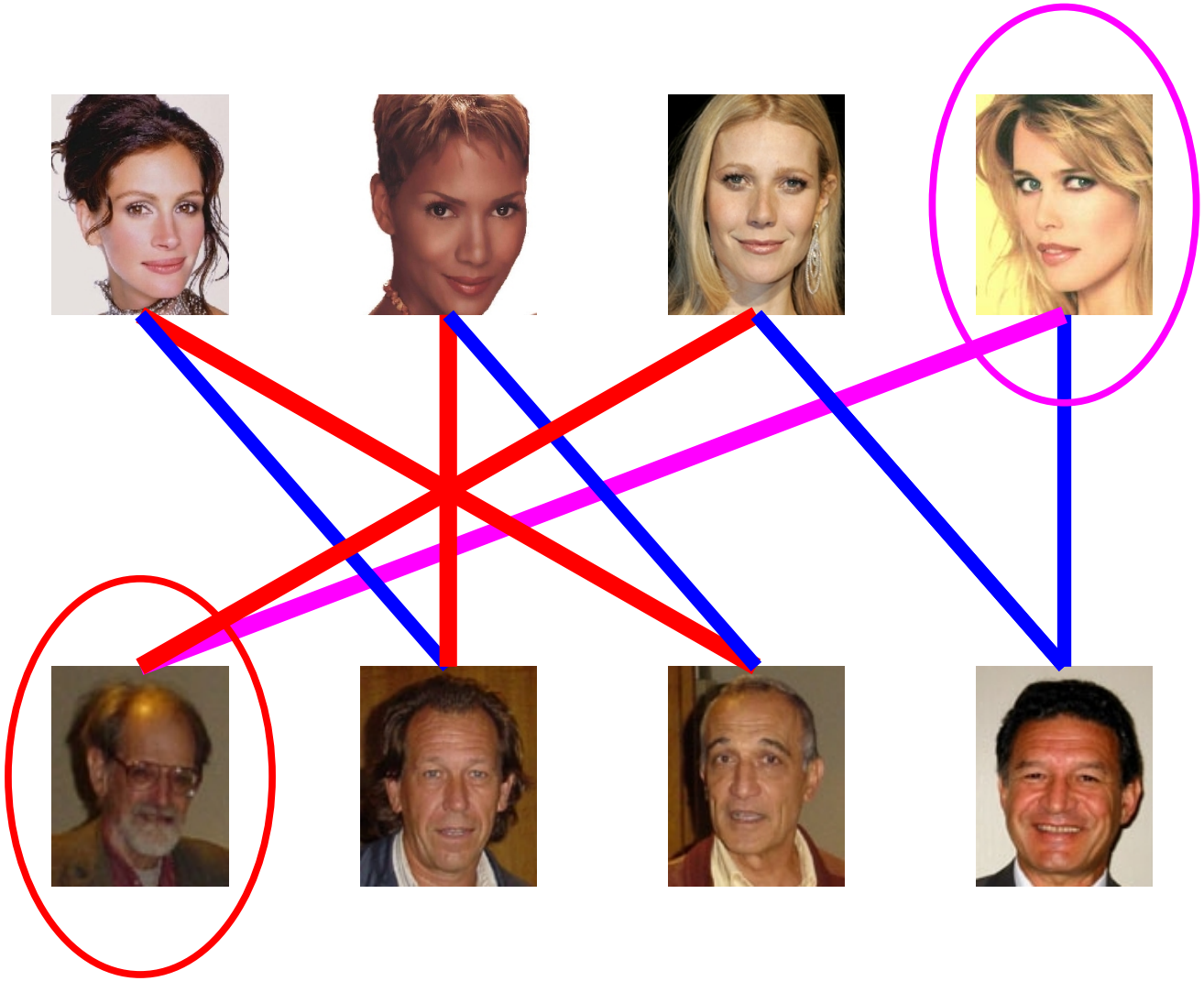
Flip edge



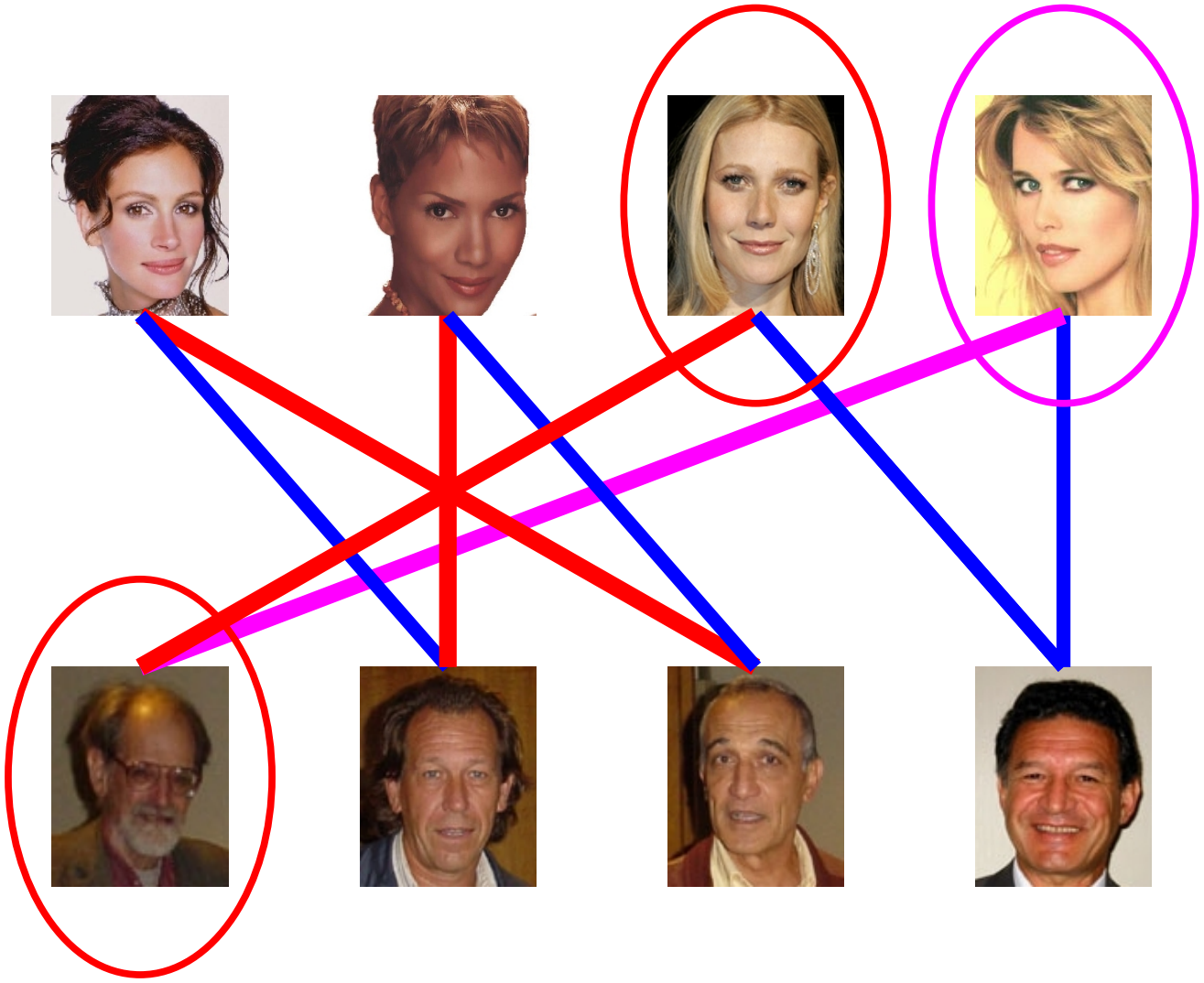
Next unmatched node



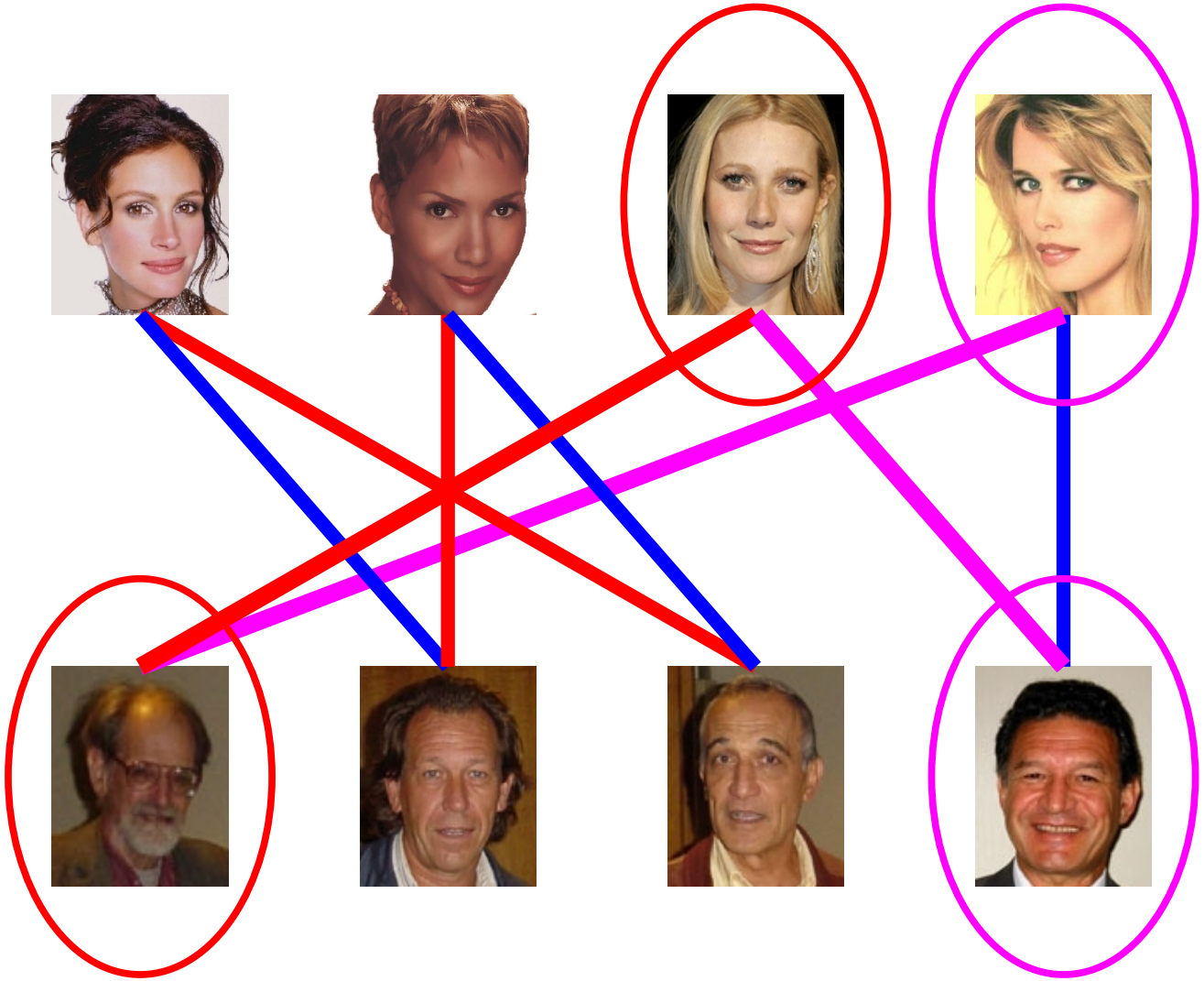
Find matched node



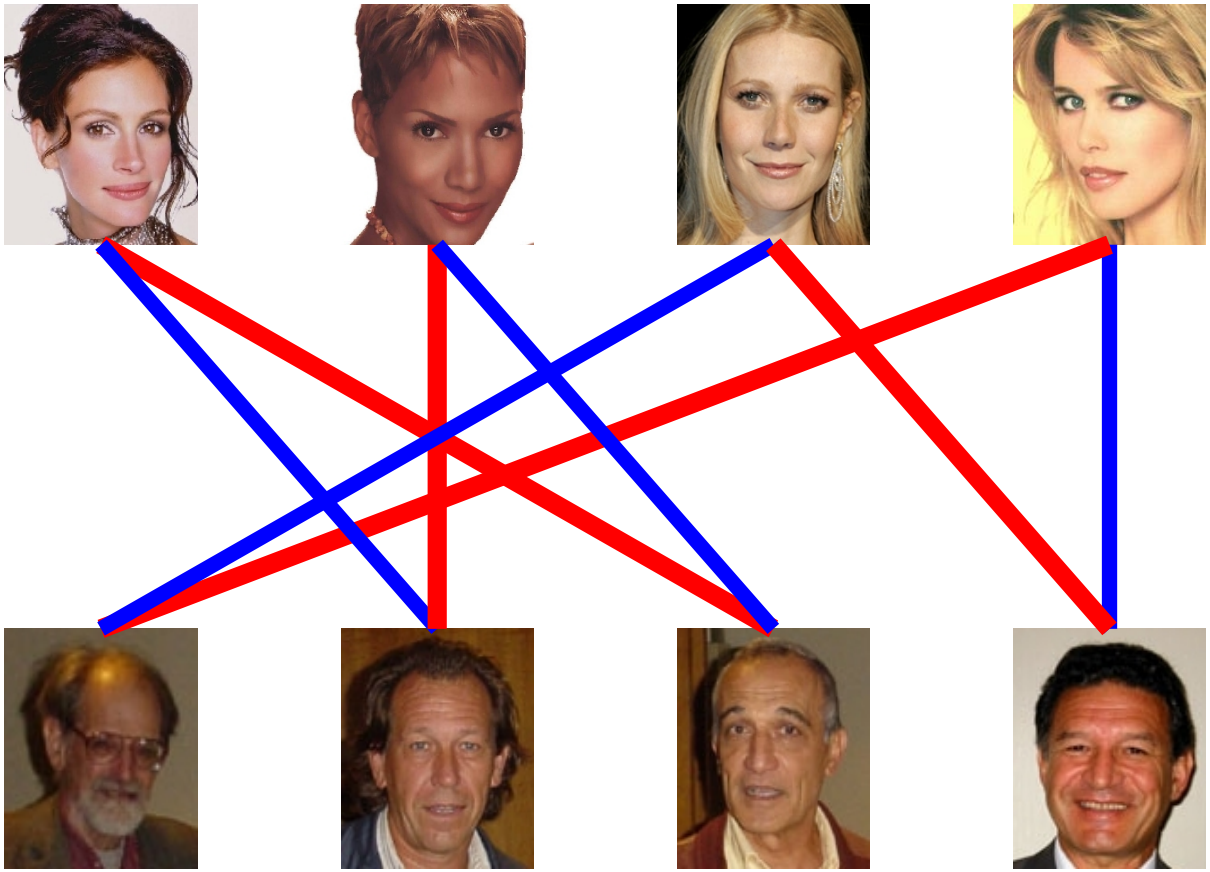
Go back along matched edge



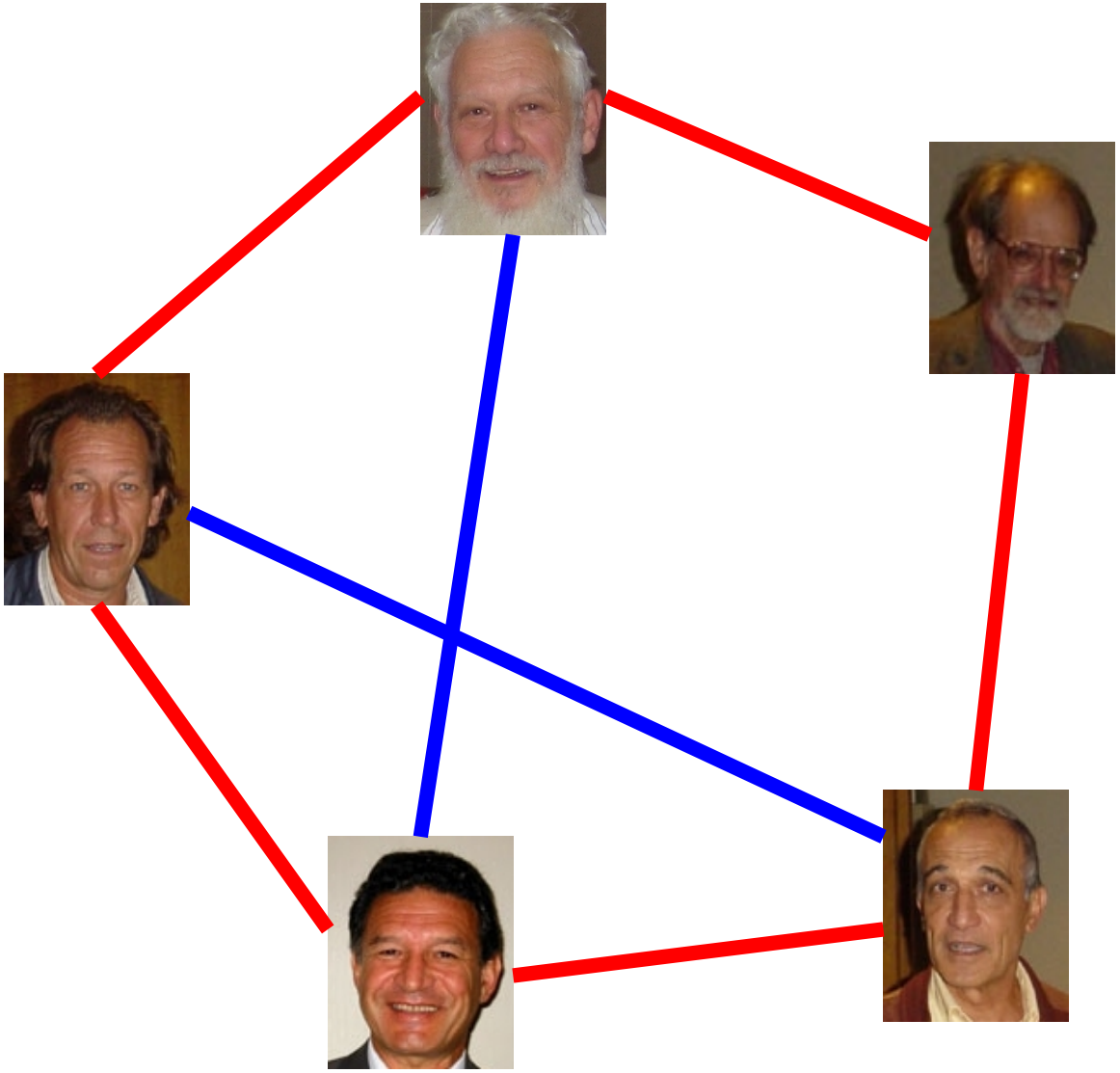
Find unmatched node



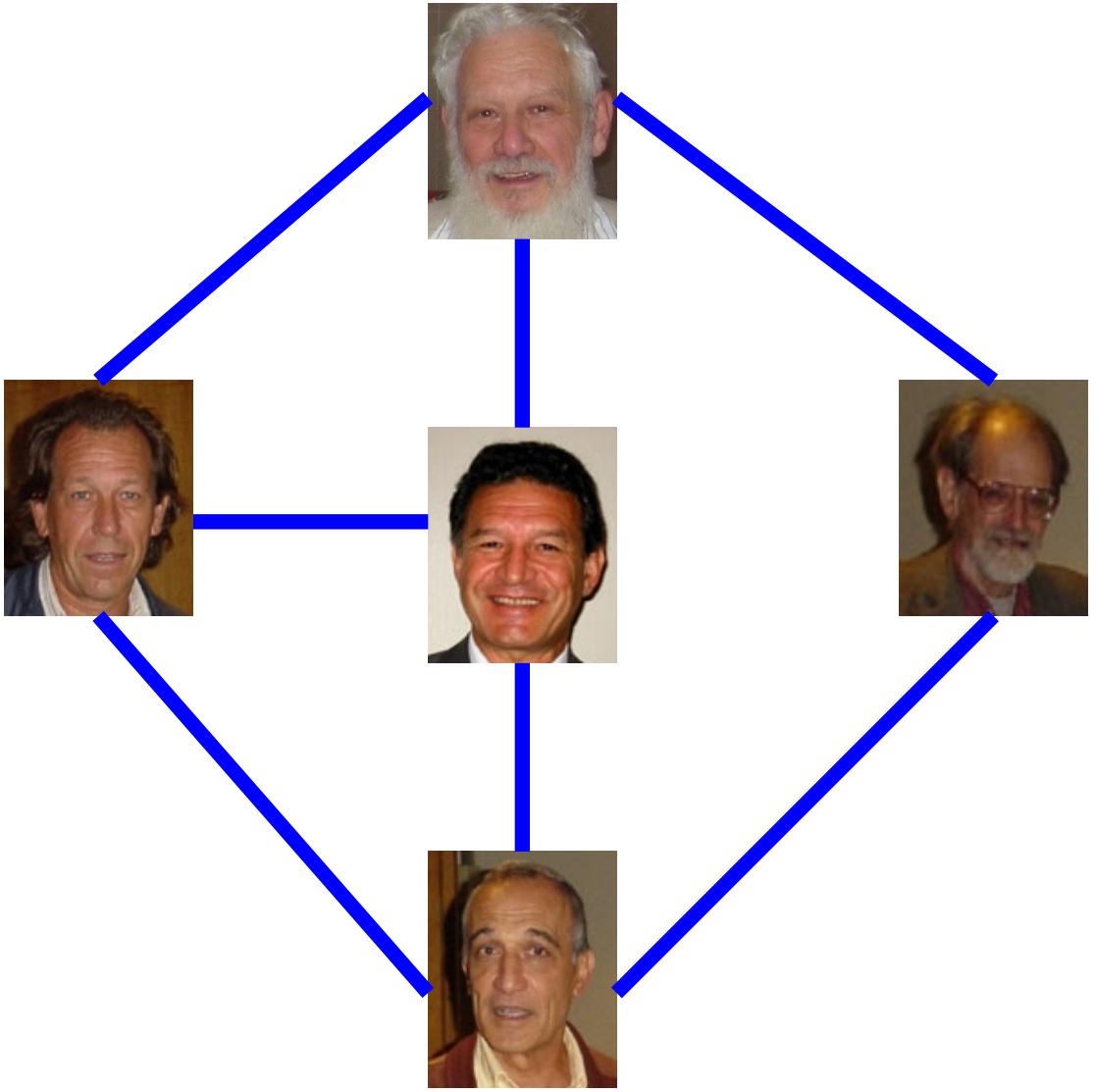
Flip edges, done



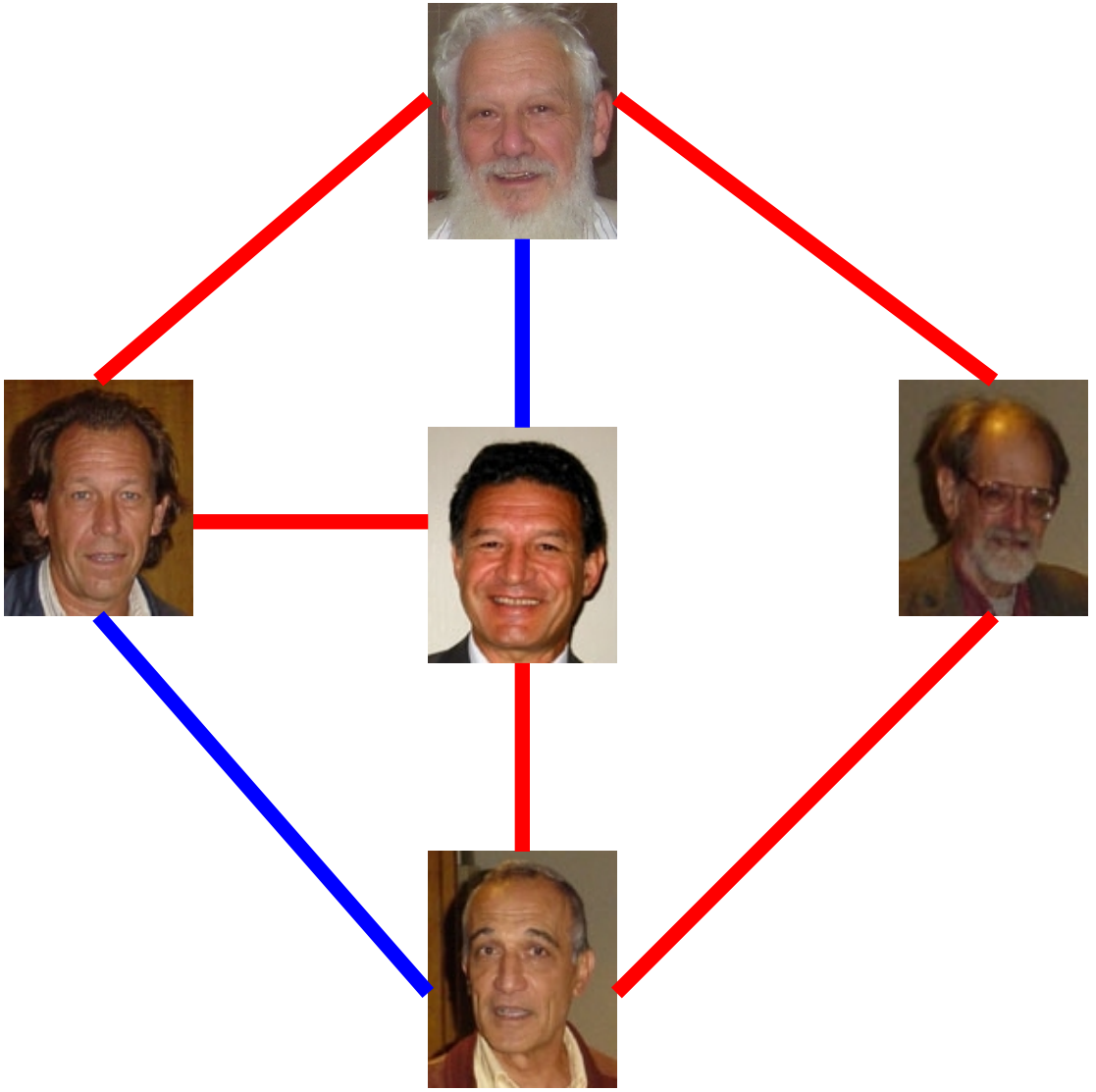
Sitting friends around a table



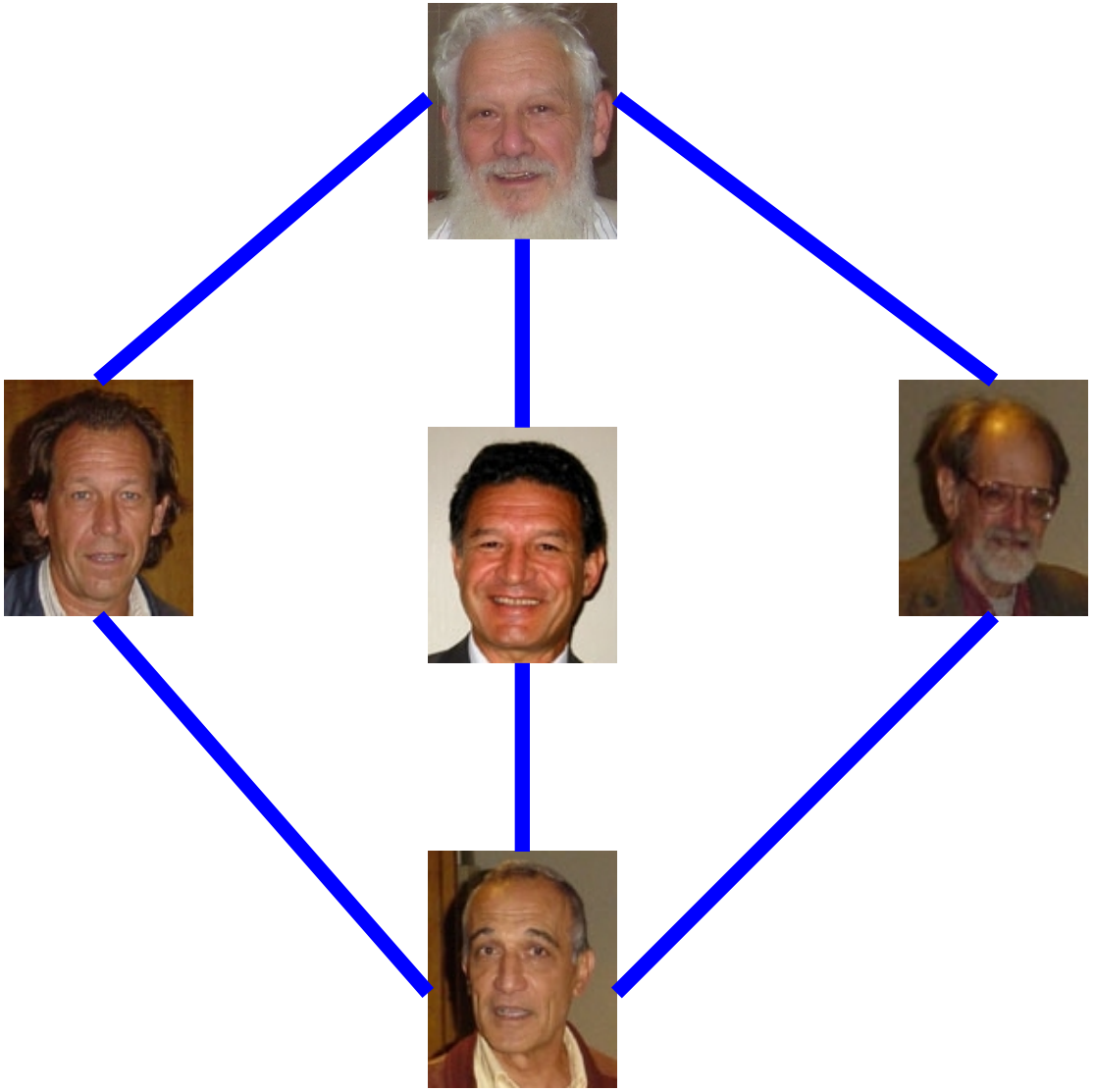
= finding a Hamiltonian cycle



Hamiltonian cycle, $\in$ NP



Hamiltonian cycle?



Complexity of Hamiltonian cycle

Input: Graph G

Question: Does G have a Hamiltonian cycle?
(= cycle that visits each node once)

Hamiltonian cycle is [widely believed to be]
not in **co-NP**.

[Karp 1975]:
Hamiltonian cycle is **NP-complete**.

NP-complete problems

A decision problem is **NP-complete**
if it is in **NP**, and if every problem in **NP**
can in polynomial time be reduced to it.

If an **NP**-complete problem is in **P**, then

$$NP = P$$

(the $1,000,000 question,

[widely believed: $NP \neq P$, $NP \neq$ co-NP]).

See: Garey / Johnson, *Computers and Intractability: A Guide to Theory of NP-completeness* [1979].

Exponential algorithms

The only known algorithms to decide

NP-complete problems of size n are exponential,

with running time c^n for $c > 1$.

Why not use brute force?

E.g. try out all $n!$ permutations of nodes for Hamiltonian path?

Why polynomial time?

Polynomial time algorithms **scale well**,

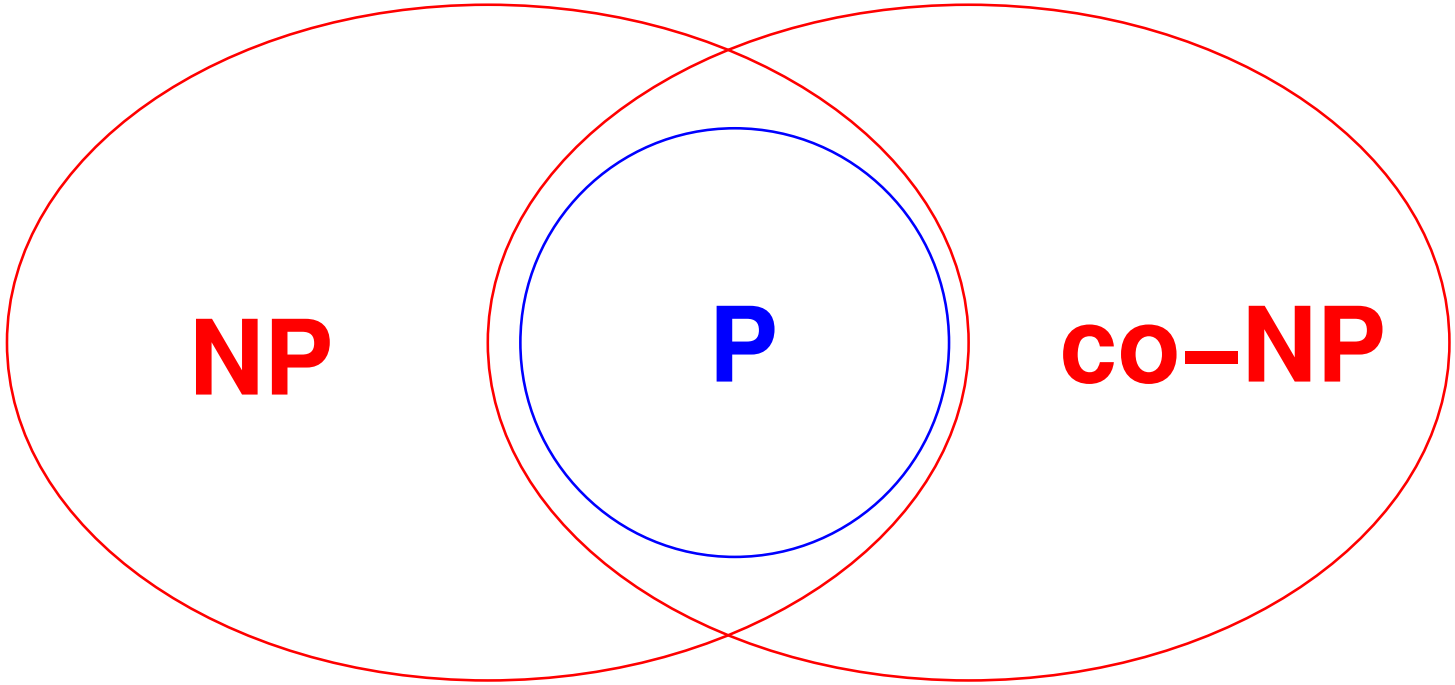
e.g. O(n^3) algorithm:

input size increases from n to $2n$

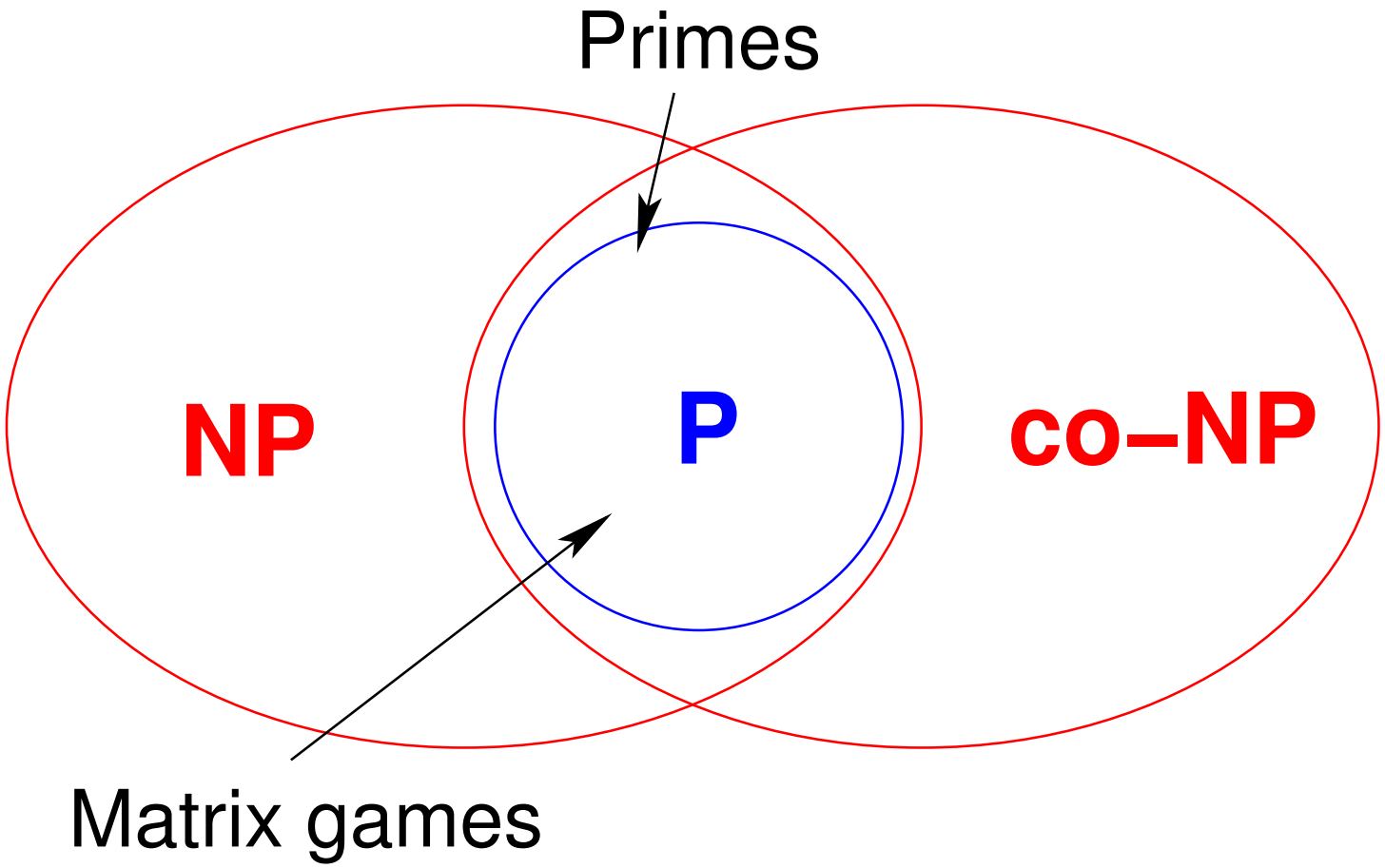
$\Rightarrow$

running time increases from cn^3 to $8cn^3$

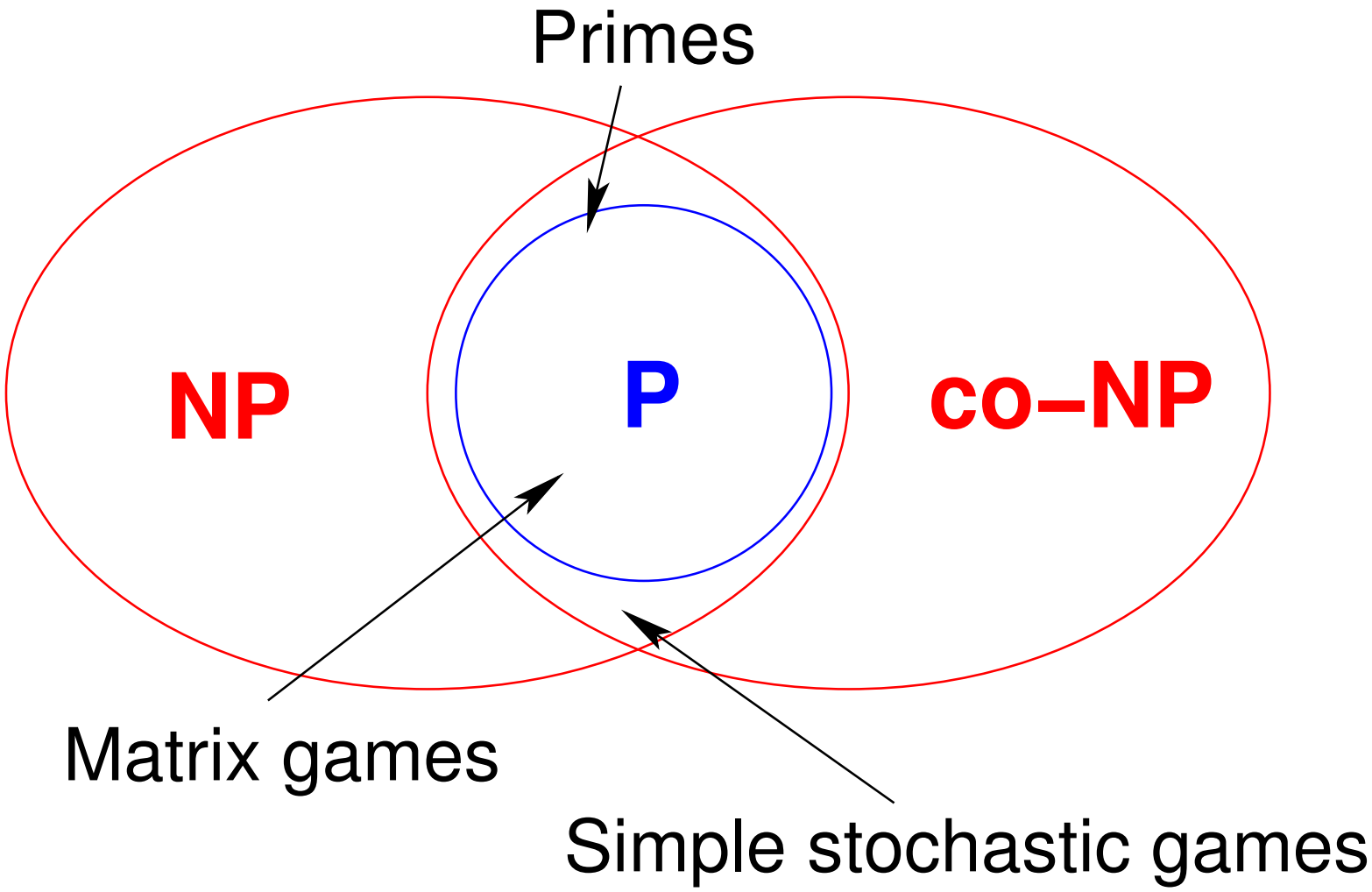
The complexity classes



The complexity classes



The complexity classes



Matrix games in NP ∩ co-NP

Input: Integer matrix A, rational number q

Question: Does the zero-sum matrix game A have value $\geq q$?

In **NP** / **co-NP** with mixed strategies for maximizer / minimizer as certificates.

In **P** via polynomial-time linear programming algorithm [Khachiyan 1979]

Primes in NP ∩ co-NP

Input: Integer x with n digits

Question: Is x prime?

In **co-NP**: factors as certificate

In **NP**: [Pratt 1975]

In **P**: [Agrawal / Kayal / Saxena 2004]

This is *not* factoring in polynomial time!

Game theoretic-problems in NP ∩ co-NP but not known to be in P

Parity games:
Used in logic and verification, equivalent to model checking for modal mu-calculus.

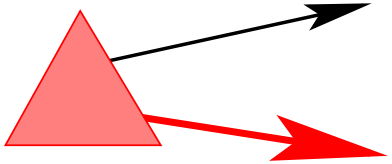
Mean-payoff games:
Used in competitive analysis of online algorithms.

Simple stochastic games:
More general than both.

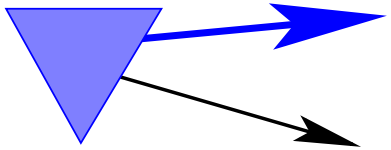
Simple stochastic game

= directed graph with

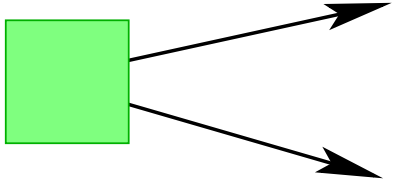
out–degree–2 nodes:



controlled by
max



controlled by
min



prob 1/2

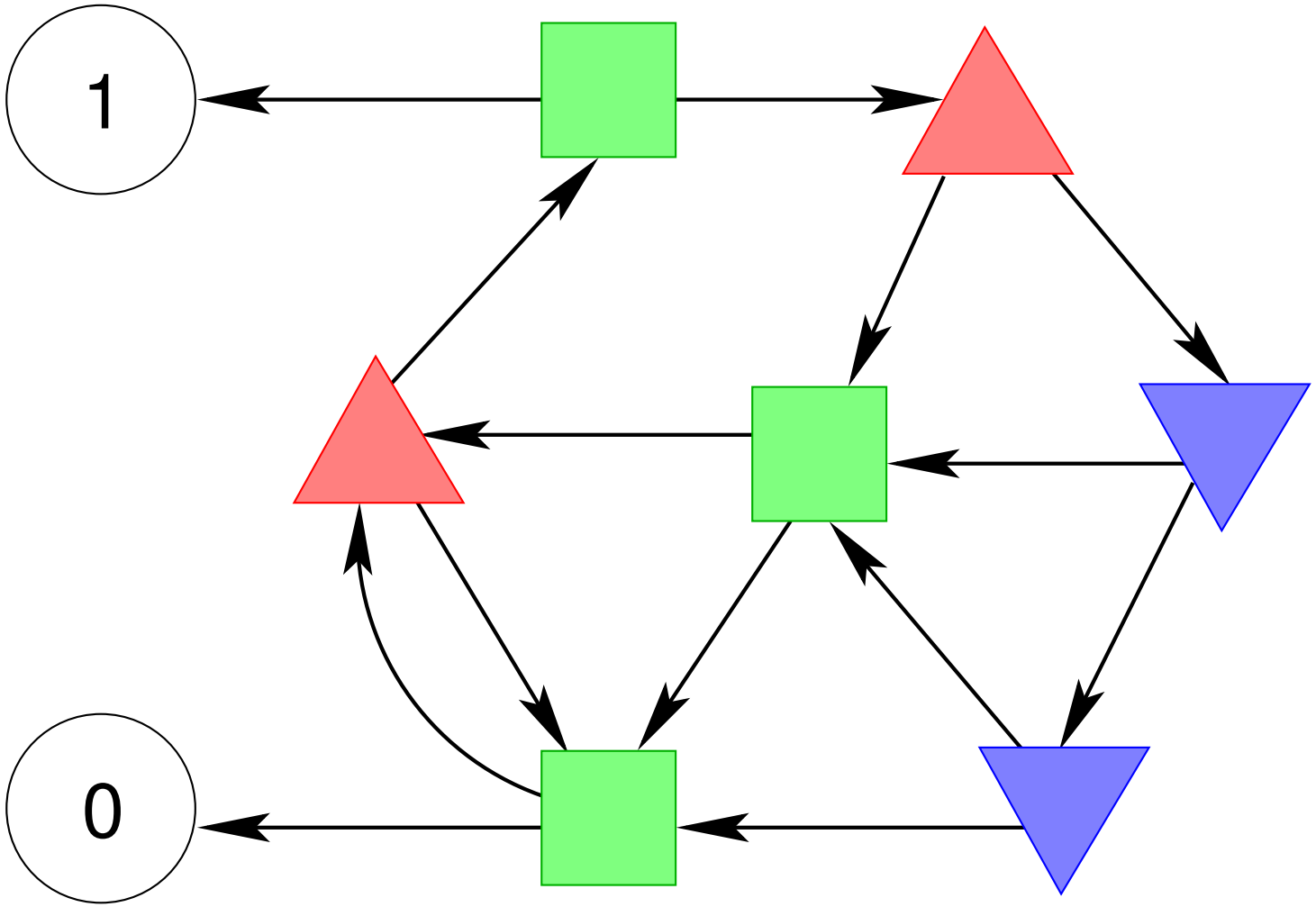
averaging
node

prob 1/2

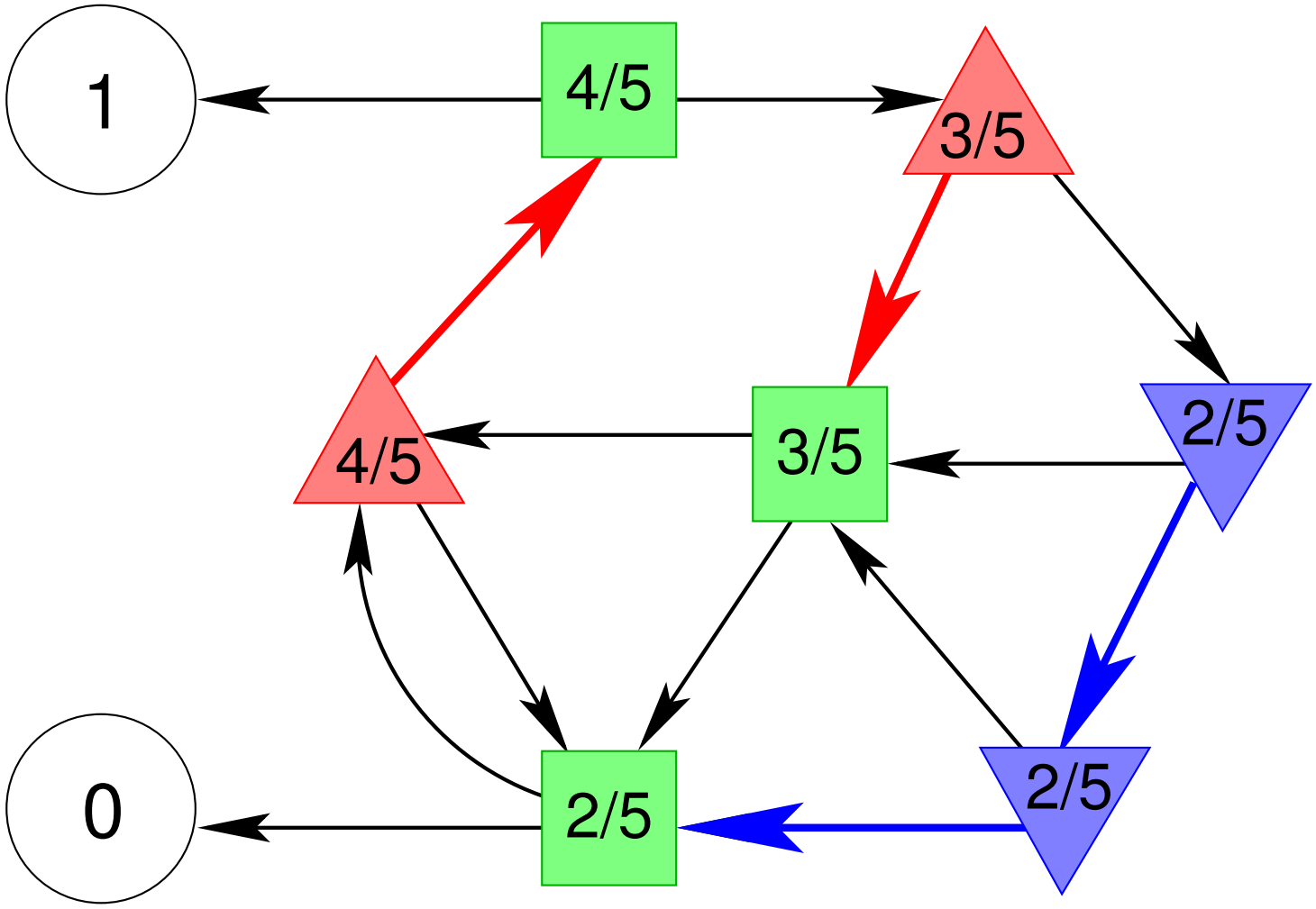
two sinks: 0 1

w.l.o.g. [Condon 1992]: stopping game (ends with prob. 1)

Simple stochastic game



Certificate: strategies and payoffs



Solving simple stochastic games

Input: Simple stochastic game, node u, rational number q

Question: When started at u, does the game have value $\geq$ q?

In **NP** / **co-NP** with pure Markov strategies for

max / min and a value for each node as certificates.

But: no polynomial-time algorithm known!

Strategy improvement algorithm

[Hoffman / Karp 1966]

Start with arbitrary strategy s of max

loop:

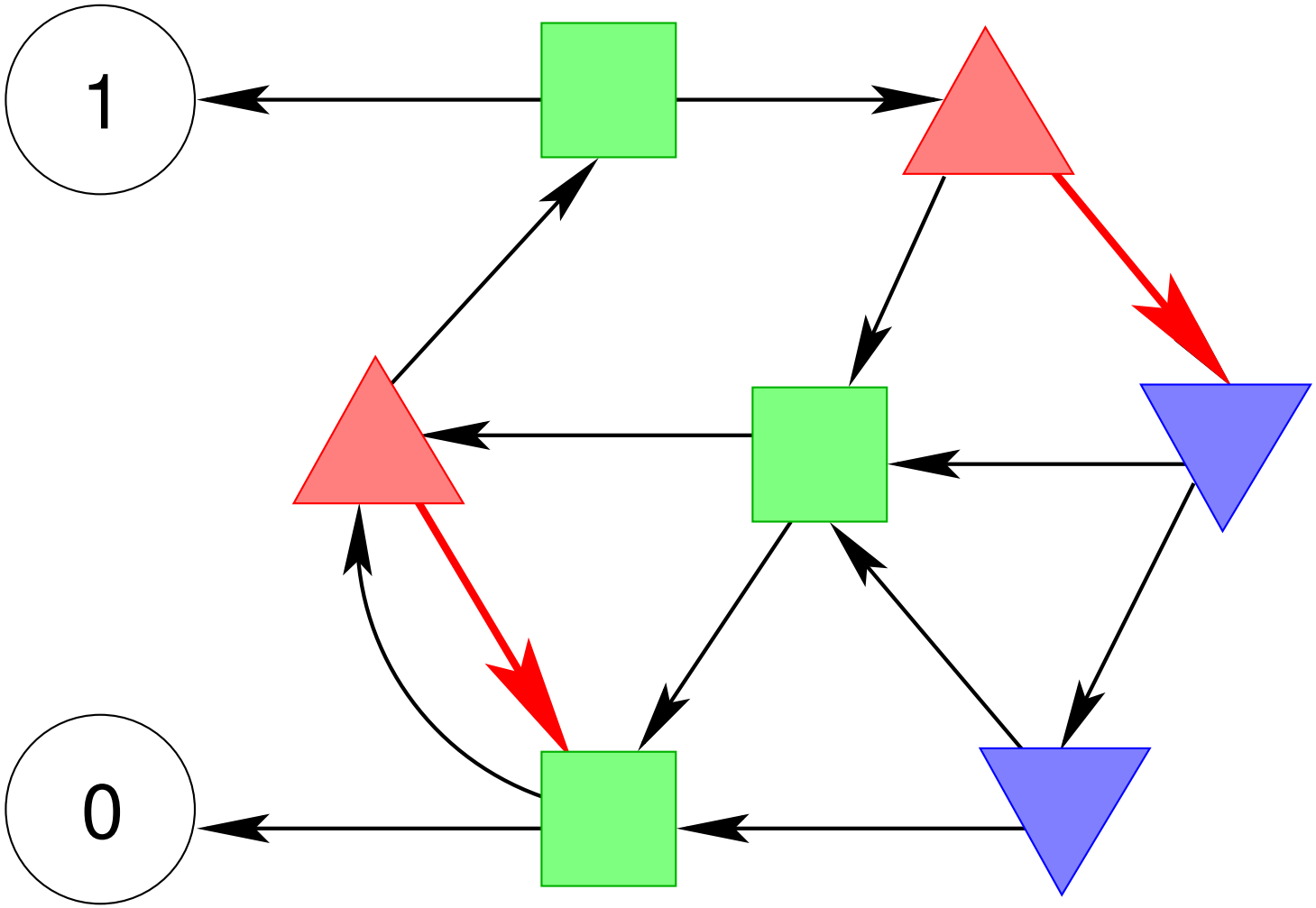
Let t be best response of min to s .

Let s' be obtained from s by switching all max-choices with better value(s,t).

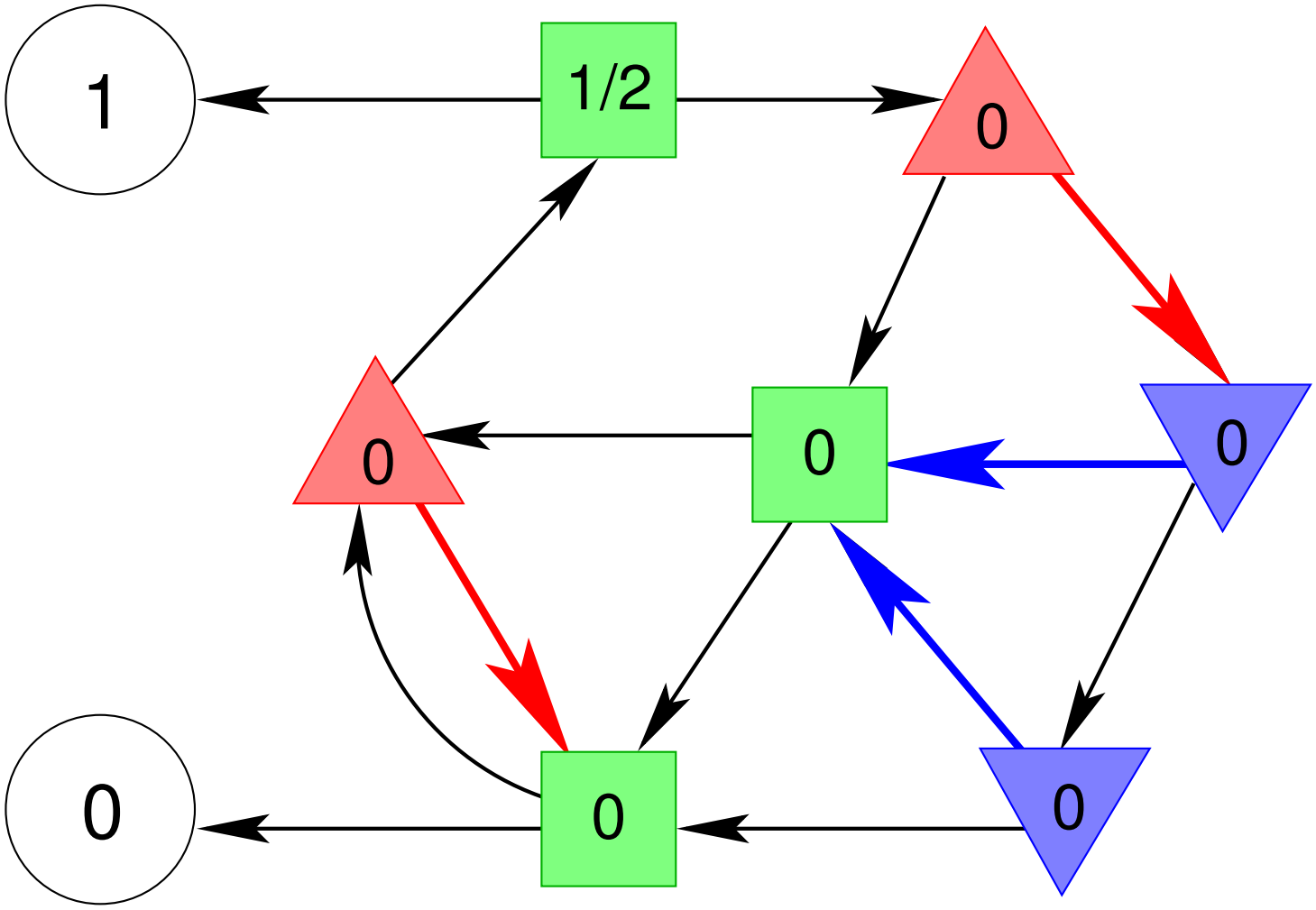
if $s \neq s'$ **repeat** with $s \leftarrow s'$

Each iteration quick, but number of iterations not known to be polynomial (nor exponential).

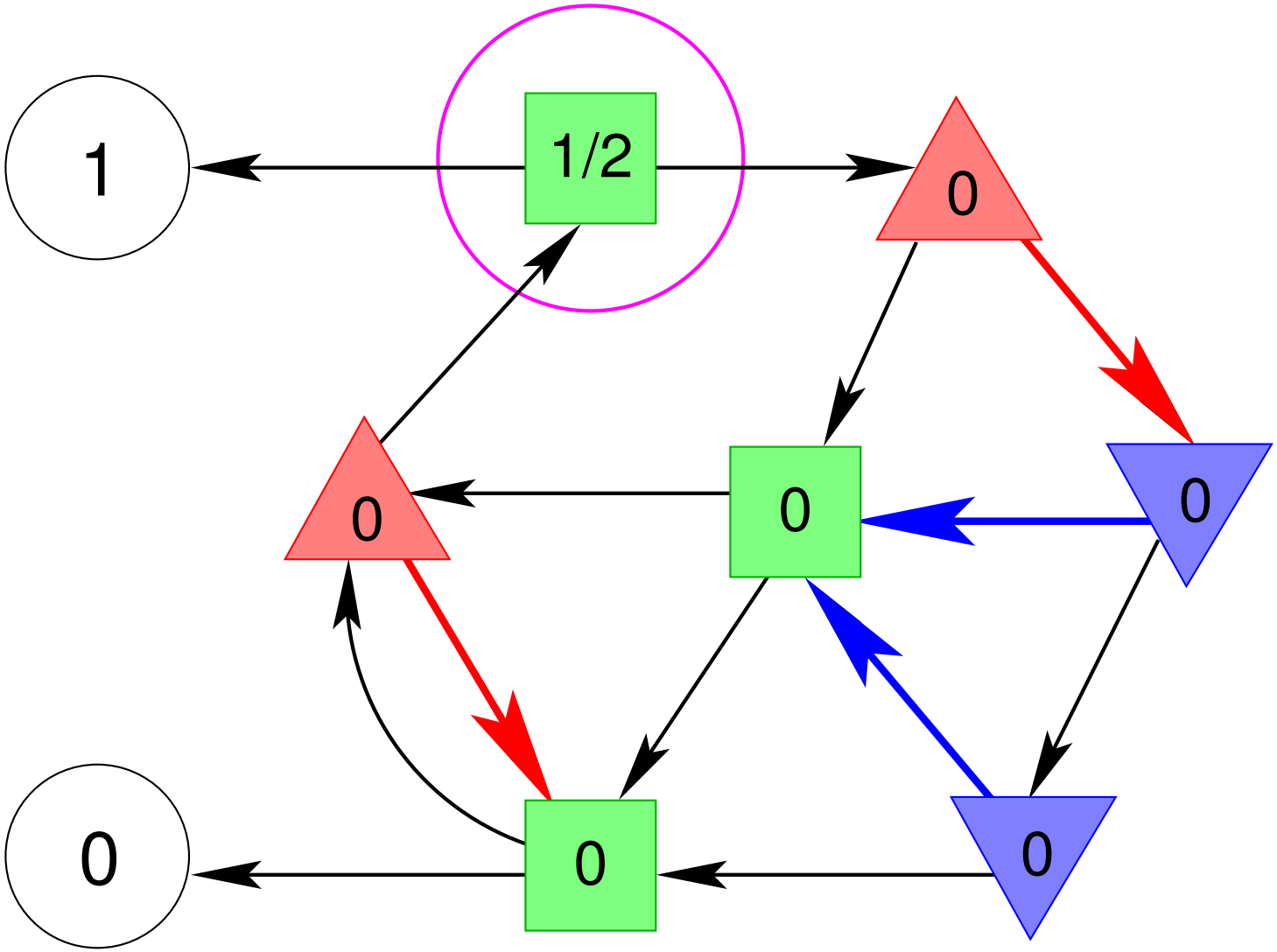
start with some max strategy s



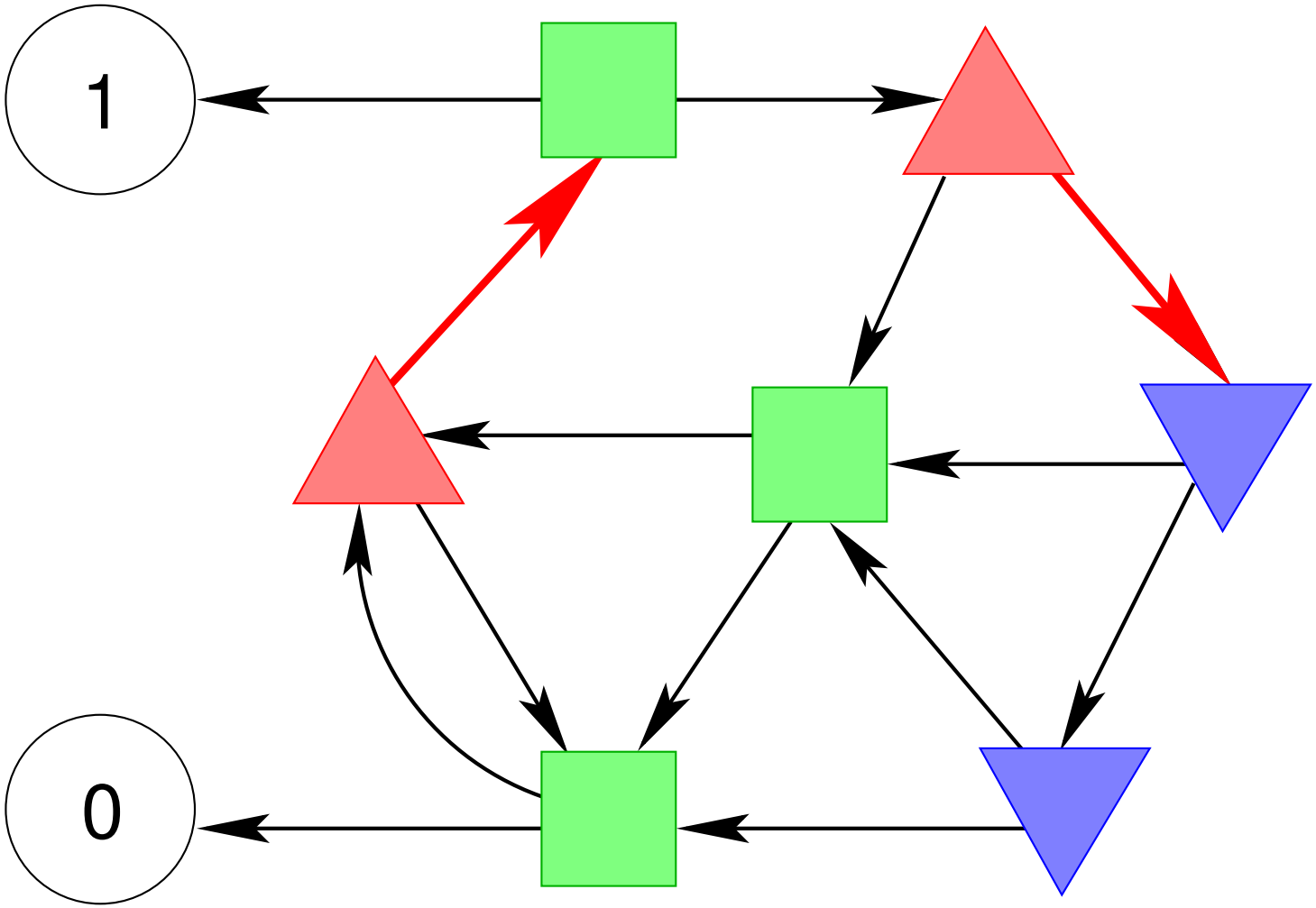
best response t



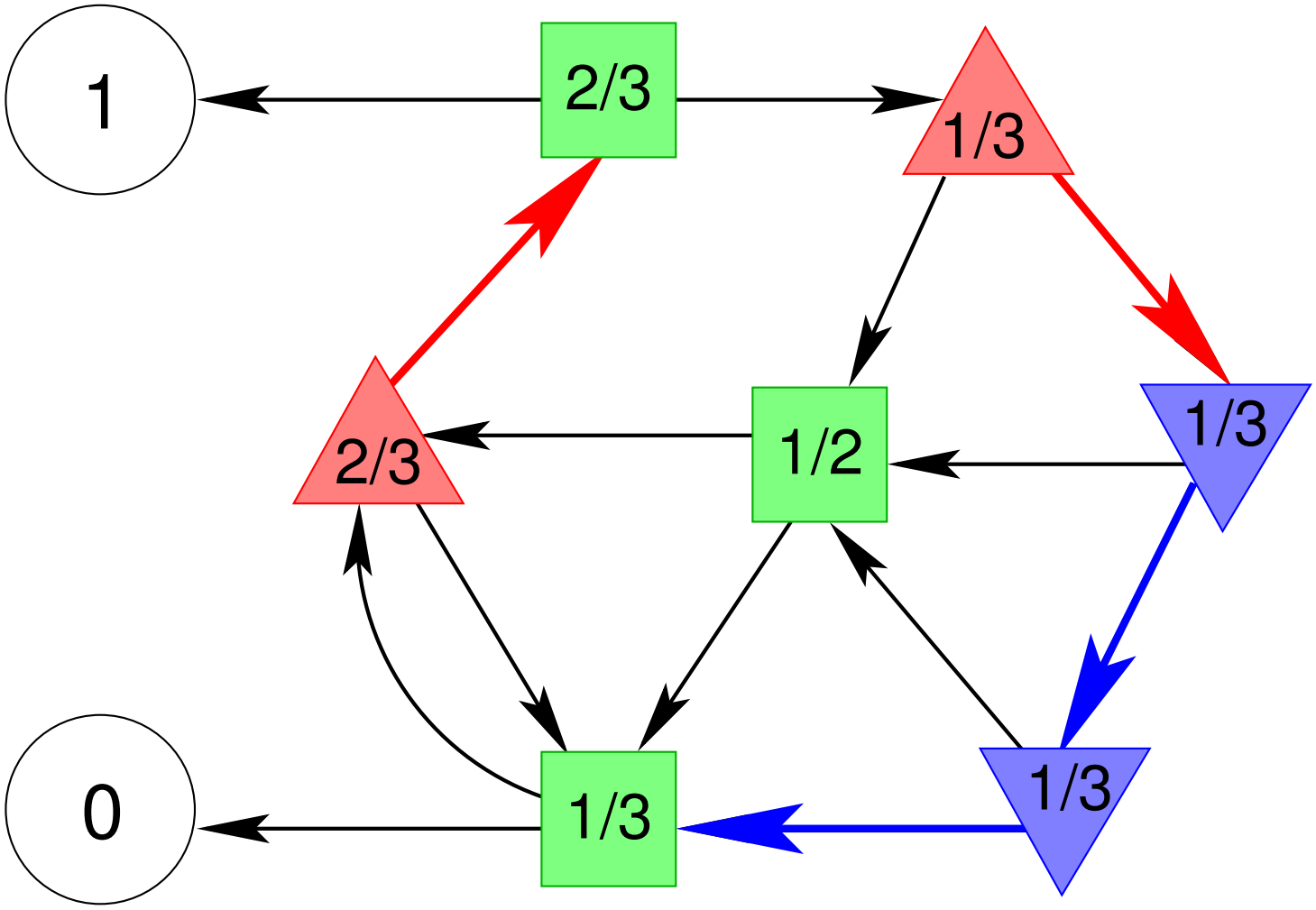
switch for max



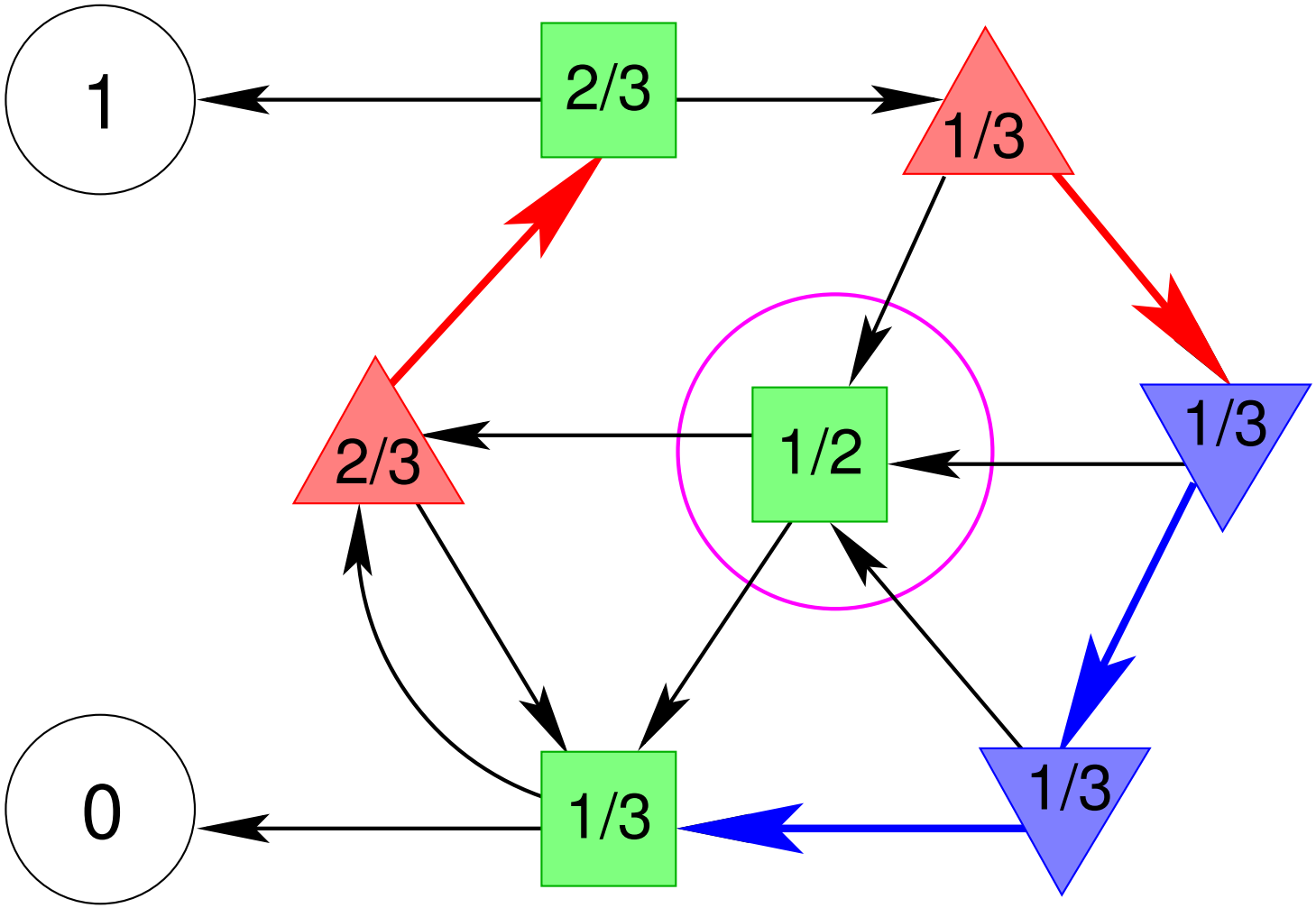
change to s’



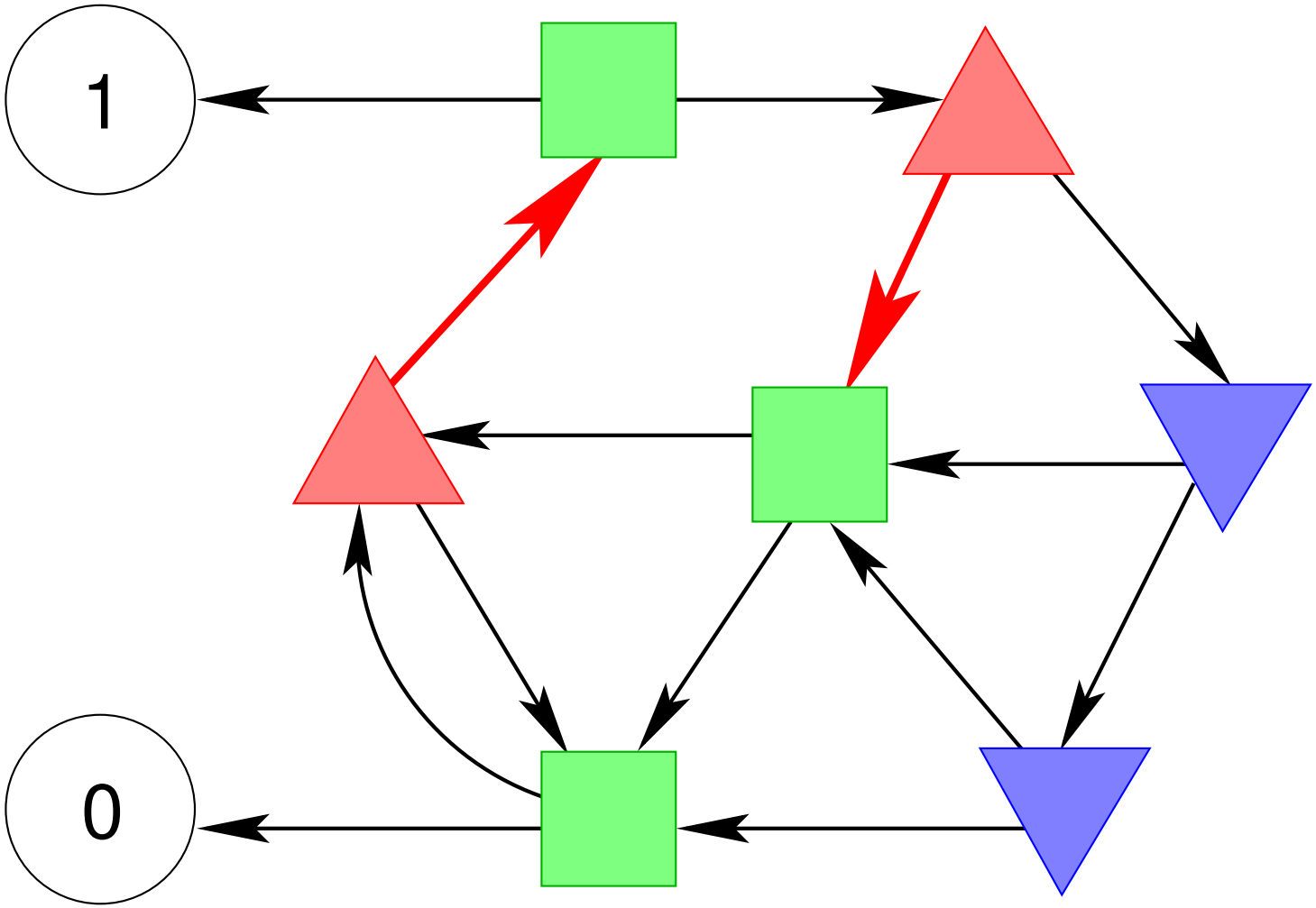
best response t’



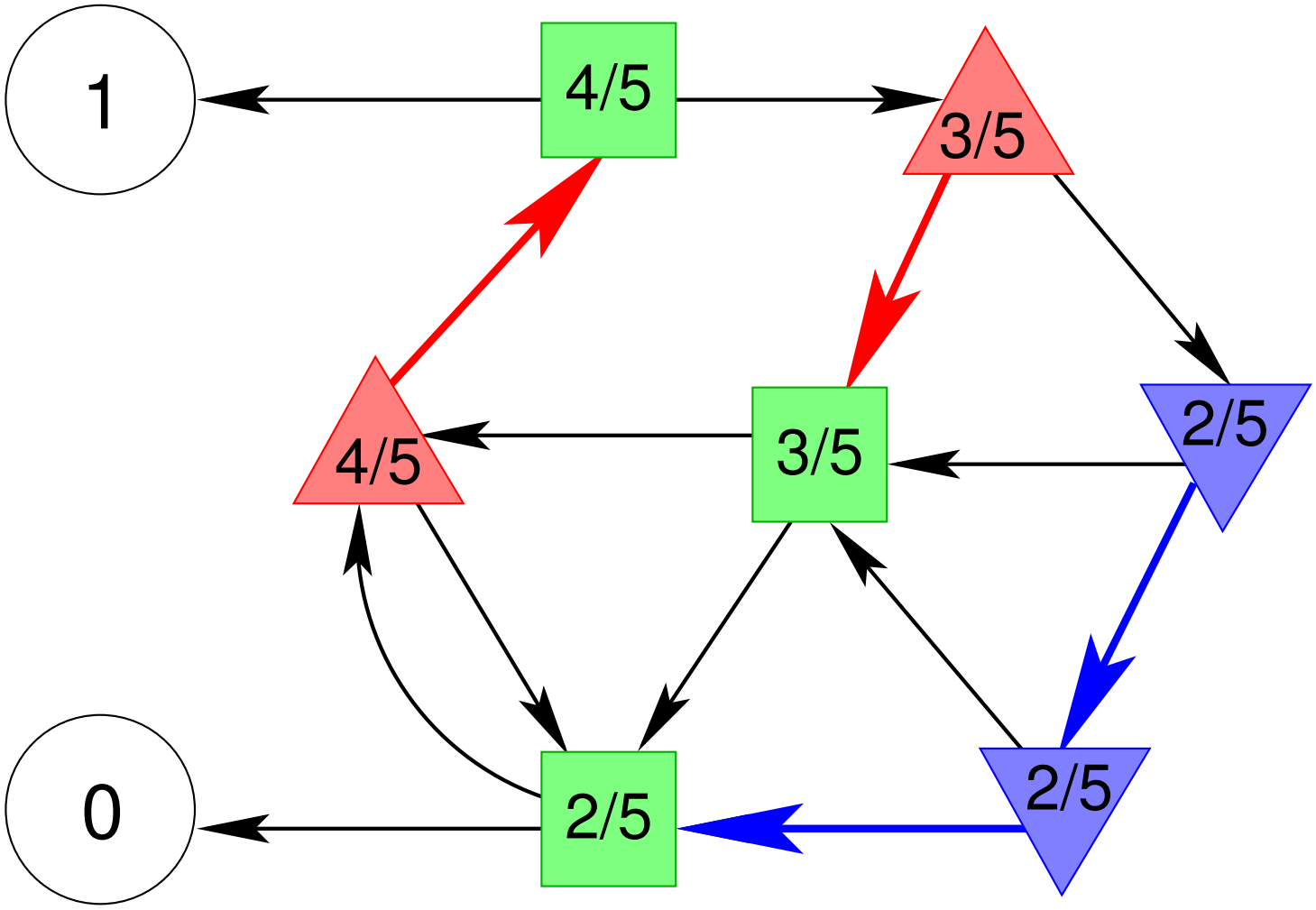
switch for max



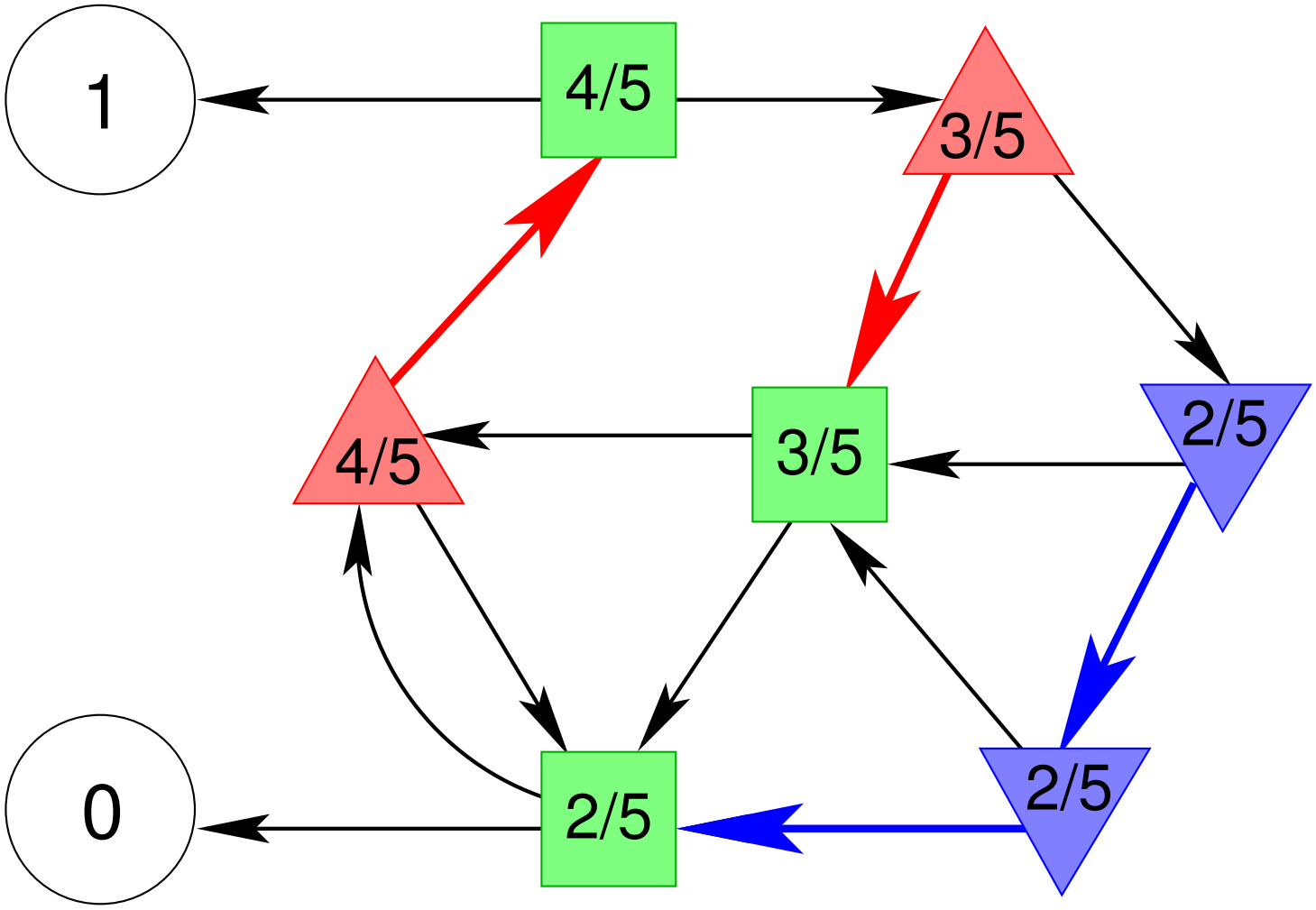
change to s”



(same) best response t’



no change for max, done



Recent progress

[Gärtner / Rüst 2005]

A simple stochastic game can be expressed as a P-matrix LCP.

LCP = Linear Complementarity Problem

P-matrix: LCP has always unique solution.

Polynomial-time interior-point methods for some P-matrix LCPs known [Ye 2002].

Finding a Nash equilibrium

Input: Bimatrix game (A,B).

Output : A Nash equilibrium (x,y).

This is a (**NP**) **search** problem.

The decision problem is trivial!

Finding a Nash equilibrium

Input: Bimatrix game (A,B).

Output : A Nash equilibrium (x,y).

This is a (**NP**) **search** problem.

No polynomial-time algorithm known!

Decision problems for Nash equilibria

[Gilboa / Zemel 1989]

The following problems are **NP-complete**:

Input: Bimatrix game, rational number q.

Question: Does the game have a Nash equilibrium with **payoff** $\geq$ q to player 1?

Input: Bimatrix game.

Question: Does the game have a **unique** Nash equilibrium ?

Long Lemke-Howson paths

[Savani / von Stengel 2006]

The Lemke-Howson algorithm for finding **one** Nash equilibrium of a bimatrix game may take exponential time.

Nash equilibria of bimatrix games

$$A = \begin{bmatrix} 3 & 3 \\ 2 & 5 \\ 0 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 4 & 3 \end{bmatrix}$$

Nash equilibrium =

pair of strategies x , y with

x best response to y and
y best response to x.

Mixed equilibria

$$A = \begin{bmatrix} 3 & 3 \\ 2 & 5 \\ 0 & 6 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 4 & 3 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 \\ 1/3 \\ 2/3 \end{bmatrix} \qquad y^T = \begin{bmatrix} 1/3 & 2/3 \end{bmatrix}$$

$$Ay = \begin{bmatrix} 3 \\ 4 \\ 4 \end{bmatrix} \qquad x^TB = \begin{bmatrix} 8/3 & 8/3 \end{bmatrix}$$

only **pure best responses** can have probability > 0

Best response polytope Q for player 2

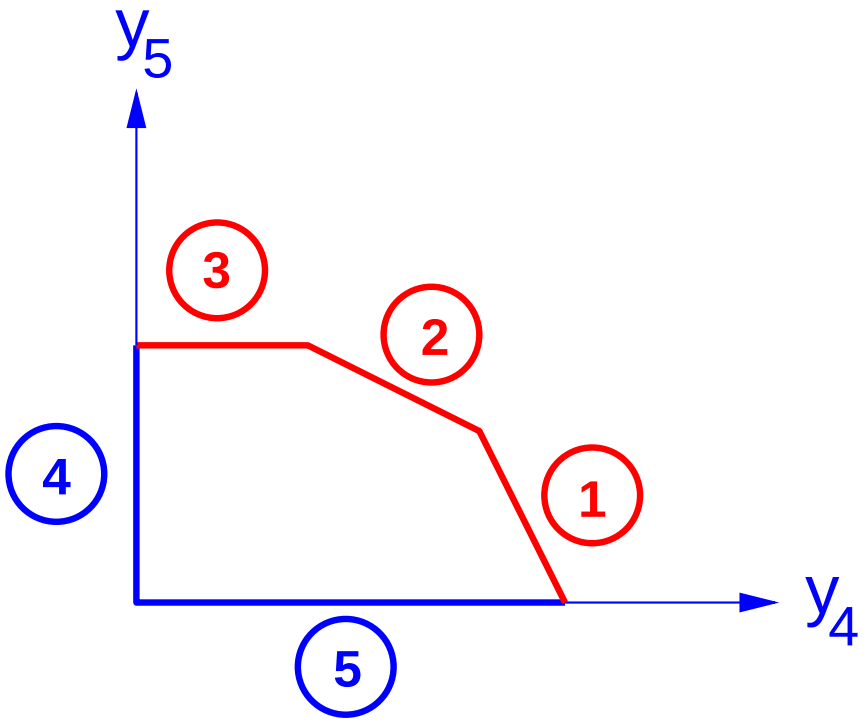
	y_4	y_5
①	3	3
②	2	5
③	0	6

$= A$

$$Q = \{ y \mid Ay \leq 1,\ y \geq 0 \}$$

$$Q = \{ (y_4, y_5) \mid$$

$$\begin{aligned}
① &: \quad 3y_4 + 3y_5 \leq 1 \\
② &: \quad 2y_4 + 5y_5 \leq 1 \\
③ &: \quad\quad 6y_5 \leq 1 \\
\\
④ &: \quad y_4 \geq 0 \\
⑤ &: \quad y_5 \geq 0 \ \}
\end{aligned}$$



Best response polytope Q for player 2

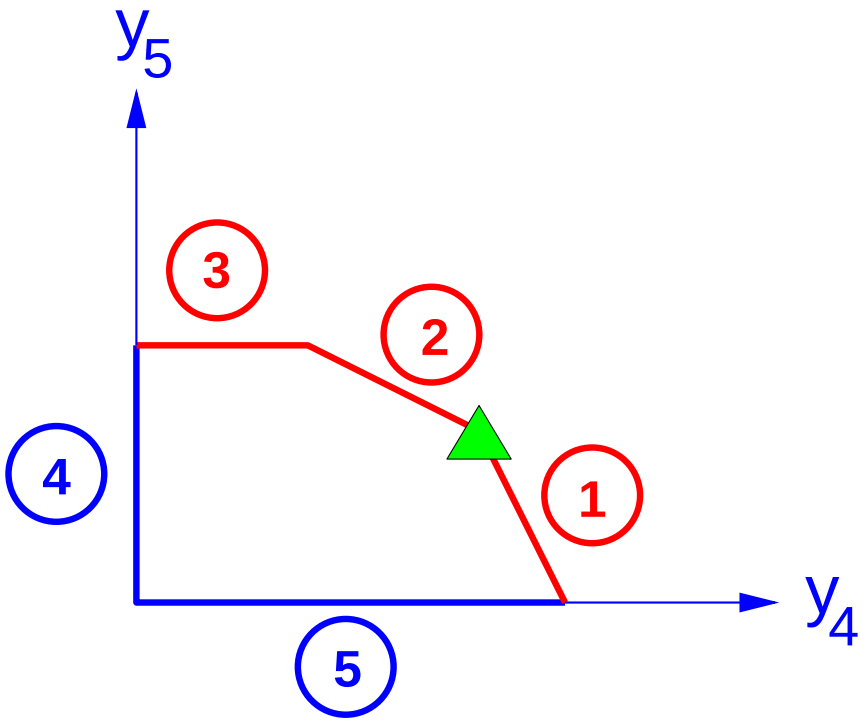
	y_4	y_5
①	3	3
②	2	5
③	0	6

$= A$

$$Q = \{ y \mid Ay \leq 1,\ y \geq 0 \}$$

$$Q = \{ (y_4, y_5) \mid$$

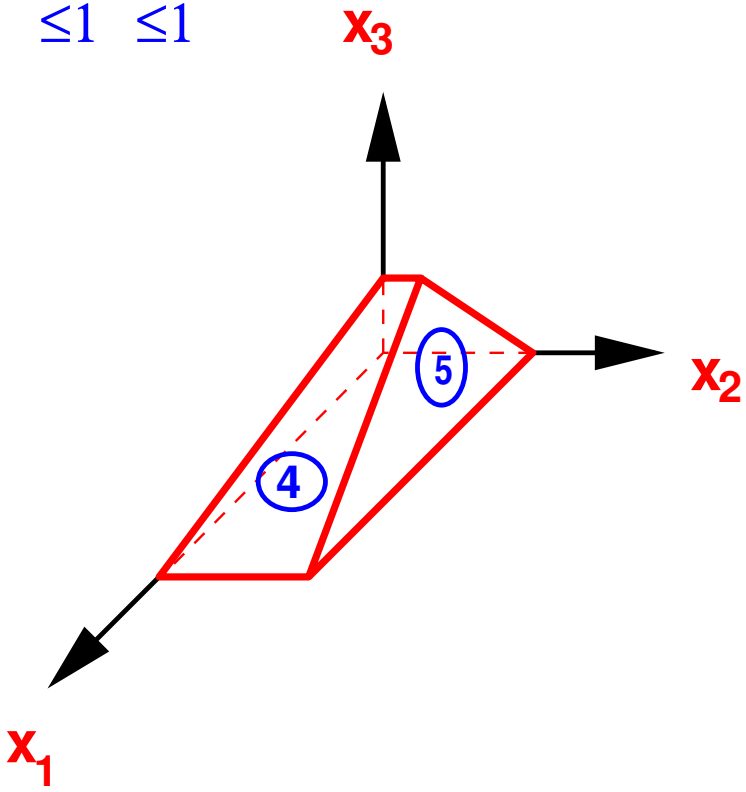
$$\begin{aligned}
① &: \quad 3y_4 + 3y_5 \leq 1 \\
② &: \quad 2y_4 + 5y_5 \leq 1 \\
③ &: \quad\quad 6y_5 \leq 1 \\
\\
④ &: \quad y_4 \geq 0 \\
⑤ &: \quad\quad y_5 \geq 0 \ \}
\end{aligned}$$

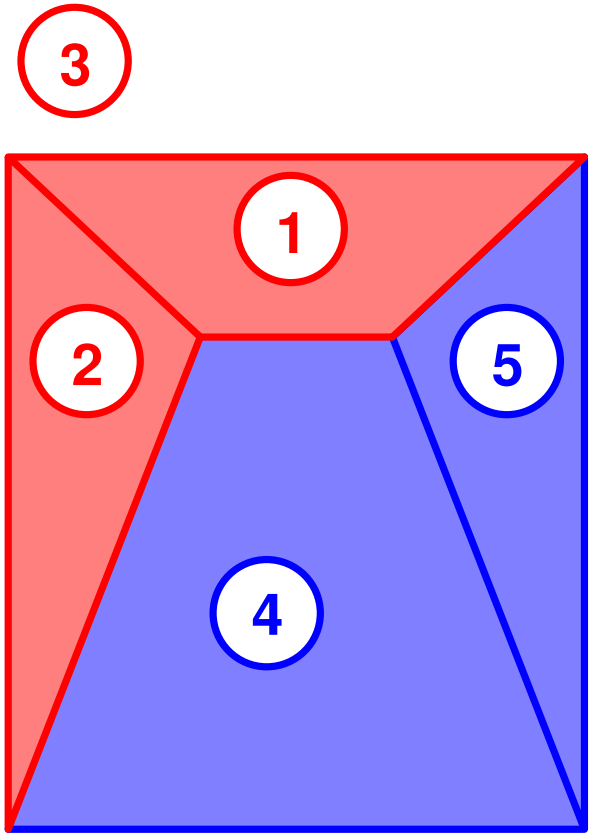


Best response polytope P for player 1

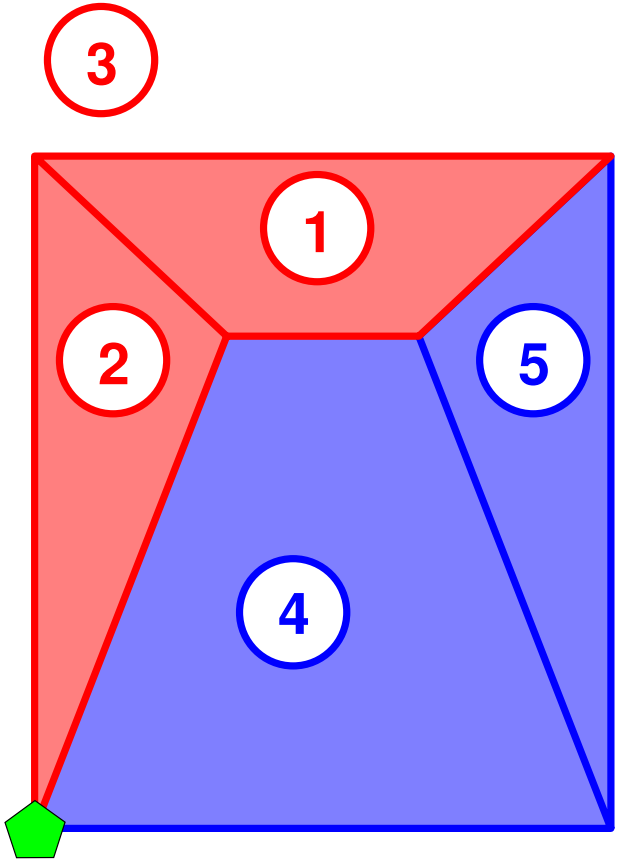
x_1	1	0	
x_2	0	2	= B
x_3	4	3	
	≤1	≤1	

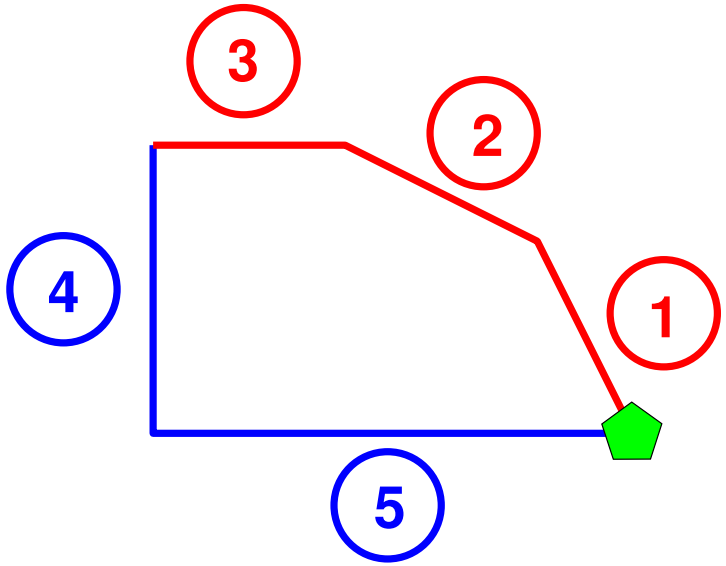
$$P = \{ x \mid x \geq 0,\ x^{\top}B \leq 1 \}$$





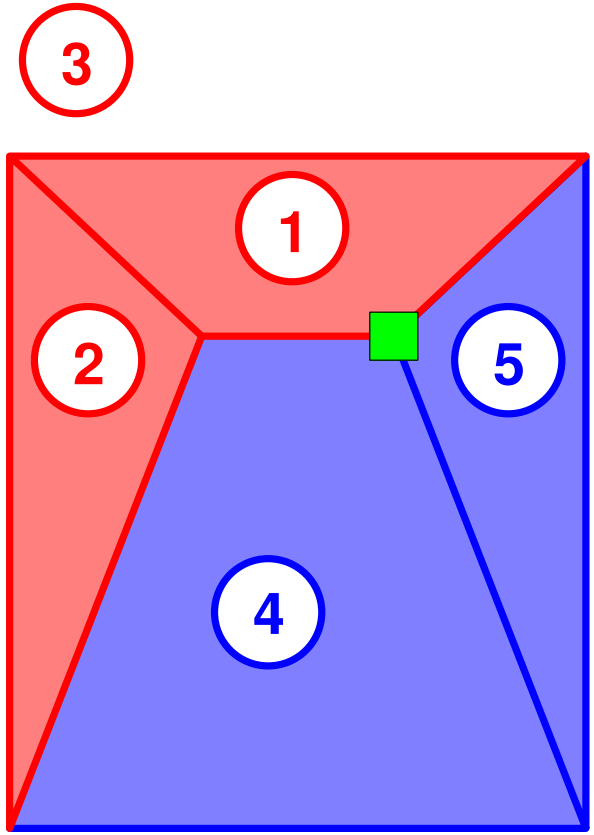
Equilibrium = completely labeled pair

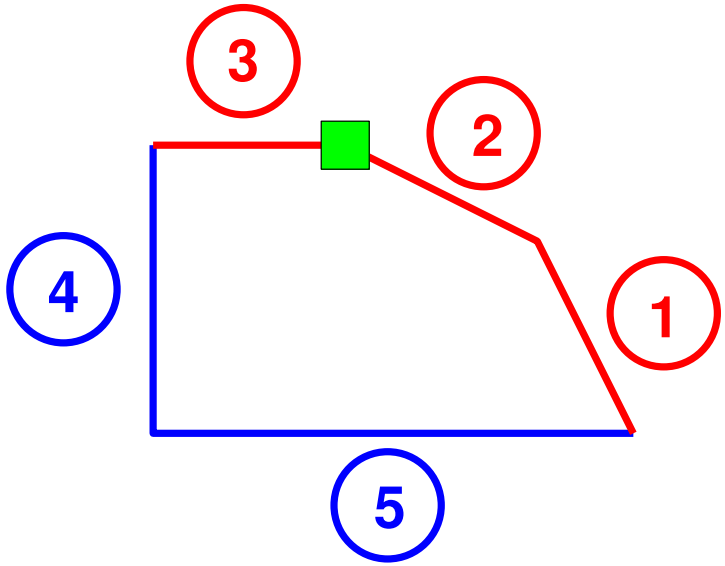




pure equilibrium

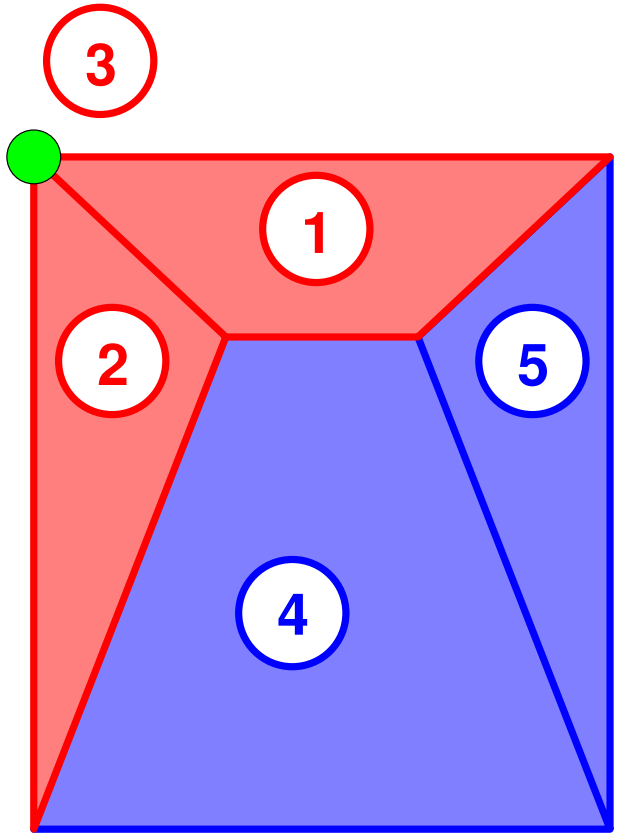
Equilibrium = completely labeled pair

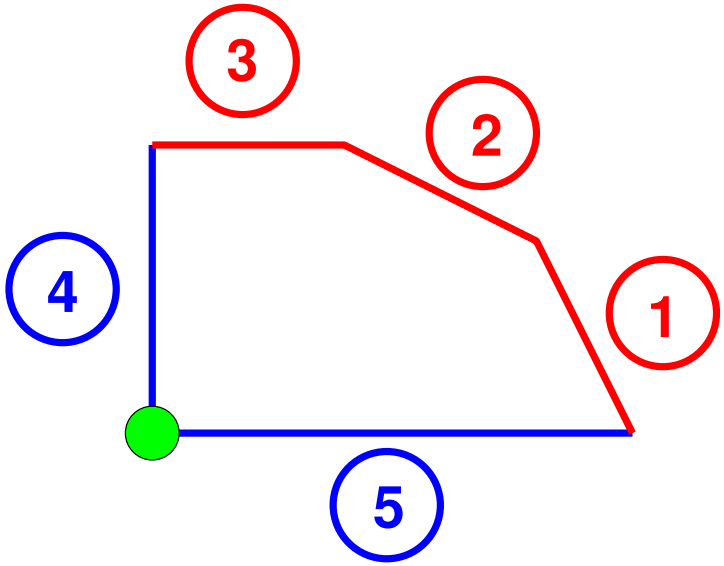




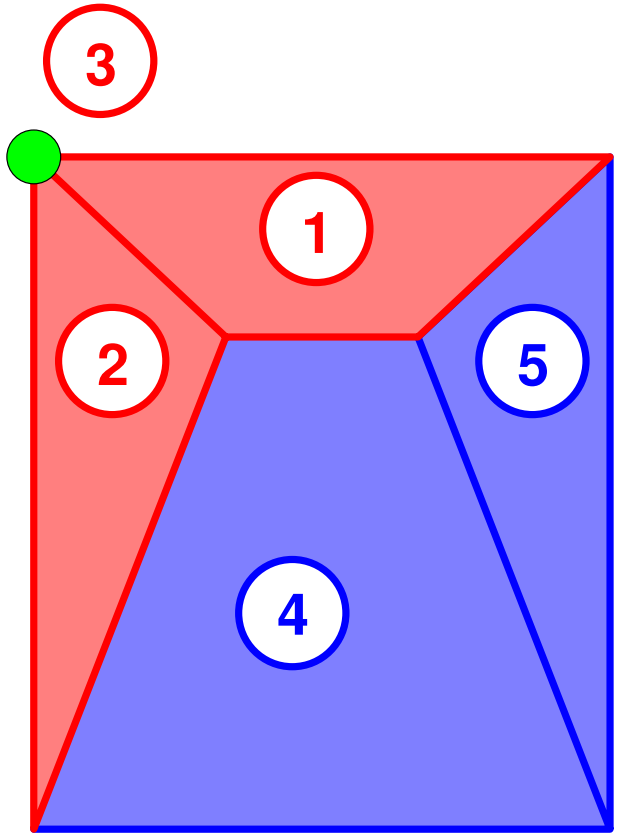
mixed equilibrium

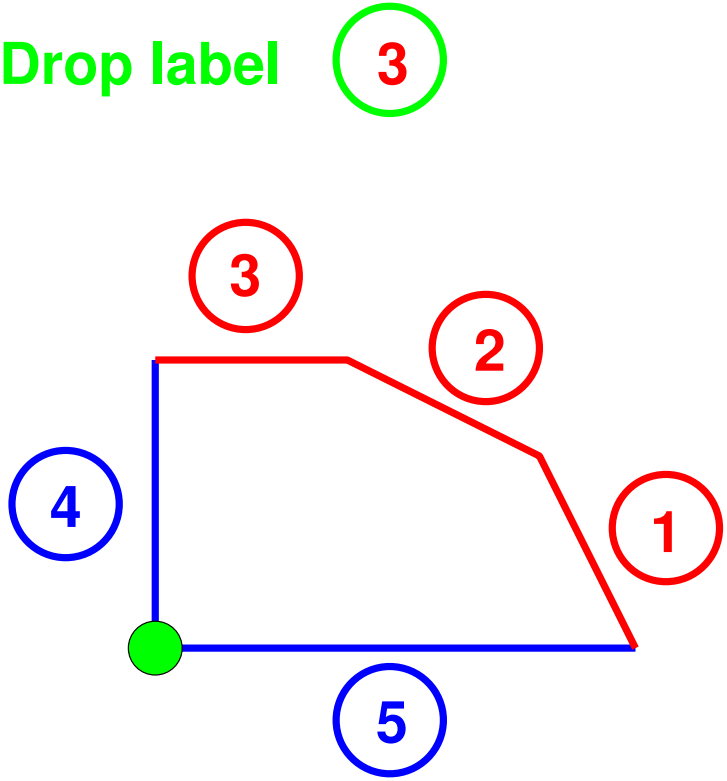
The Lemke–Howson algorithm



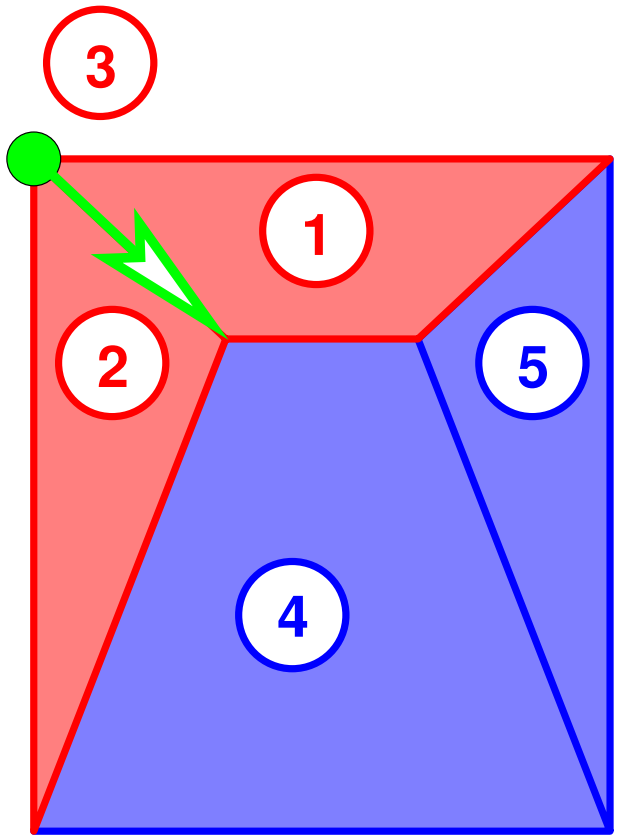


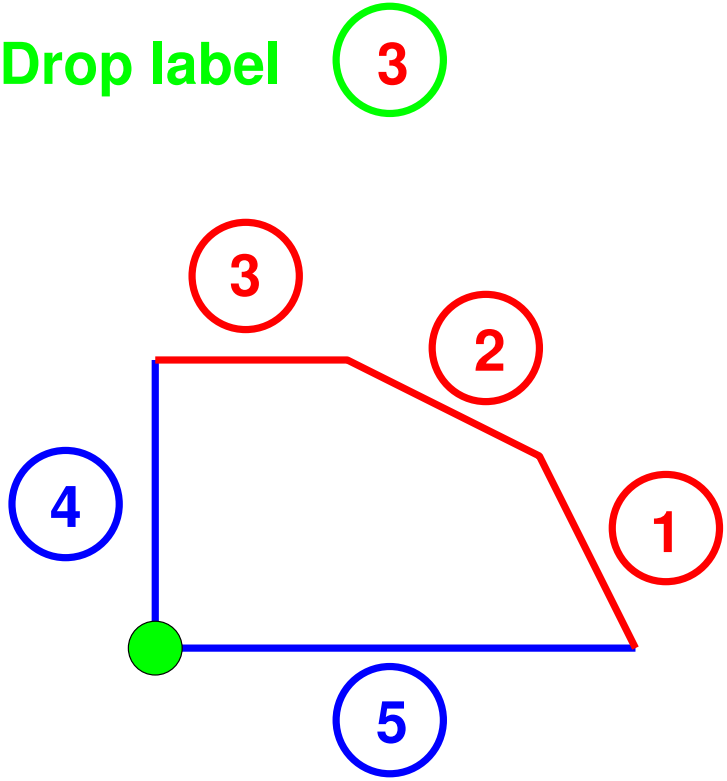
The Lemke–Howson algorithm



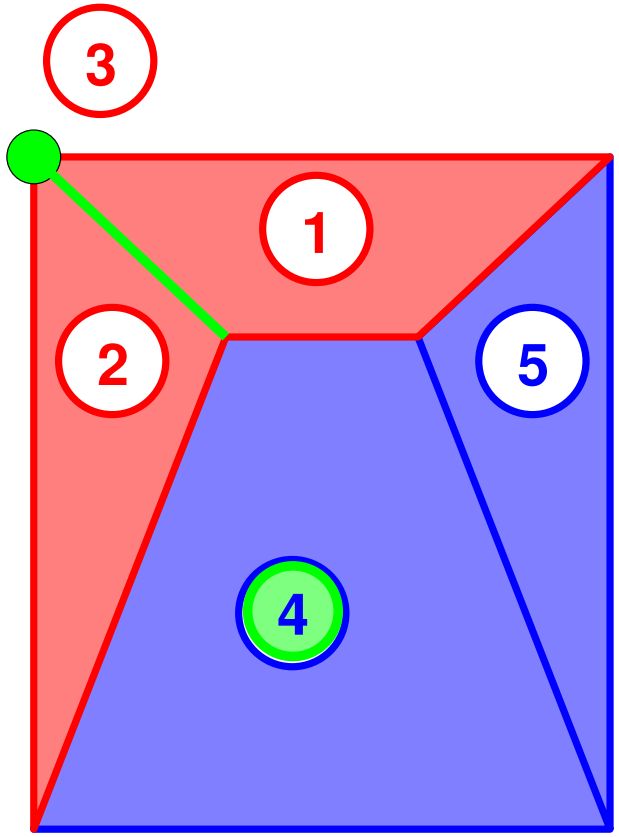


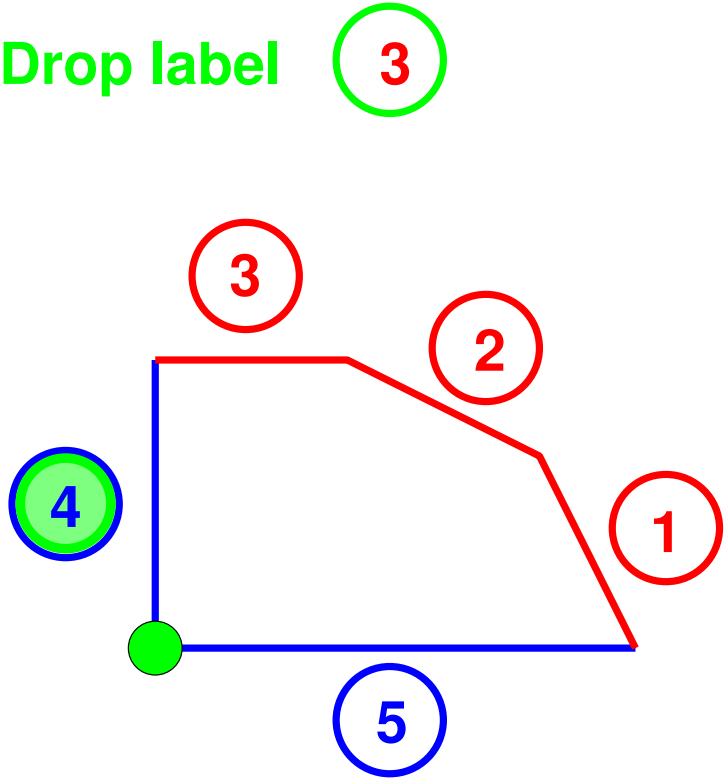
The Lemke–Howson algorithm



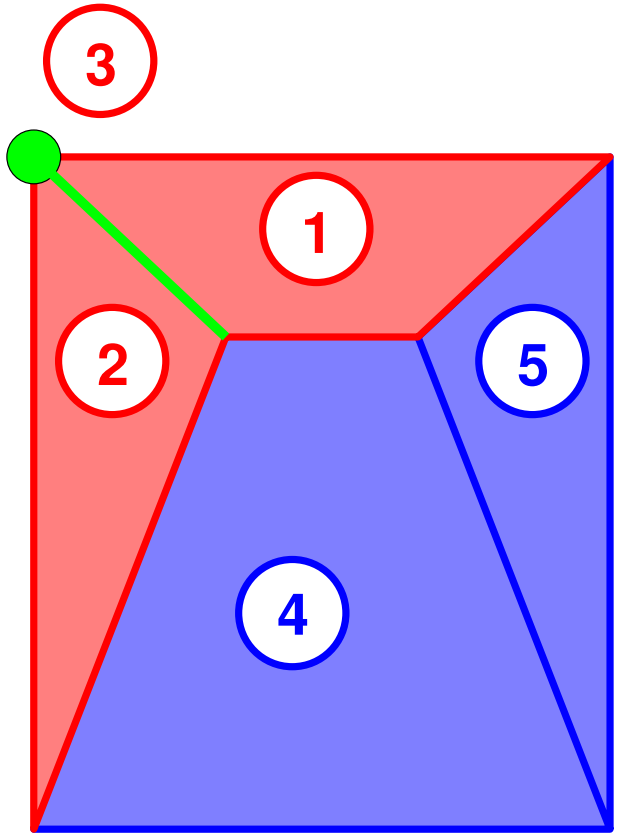


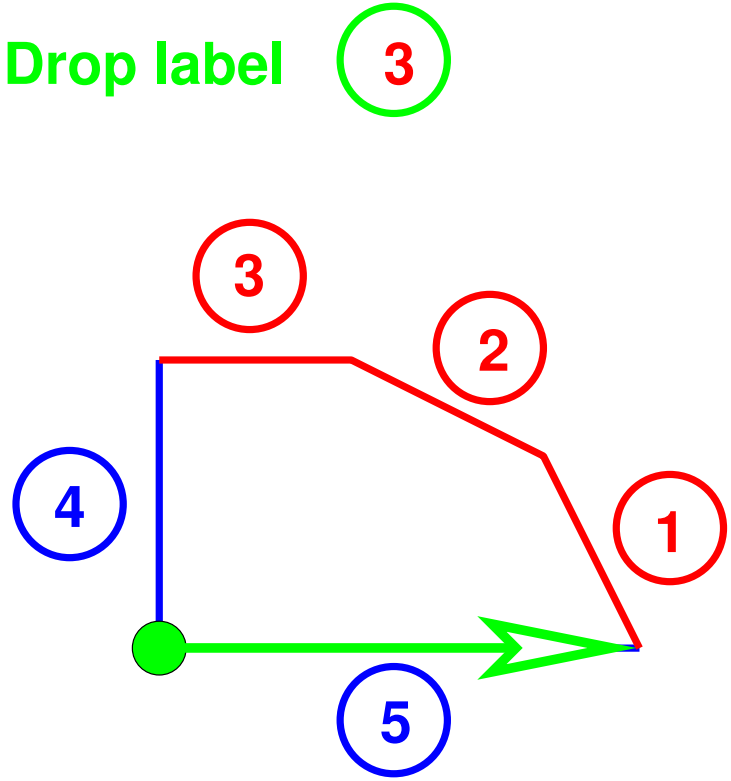
The Lemke–Howson algorithm



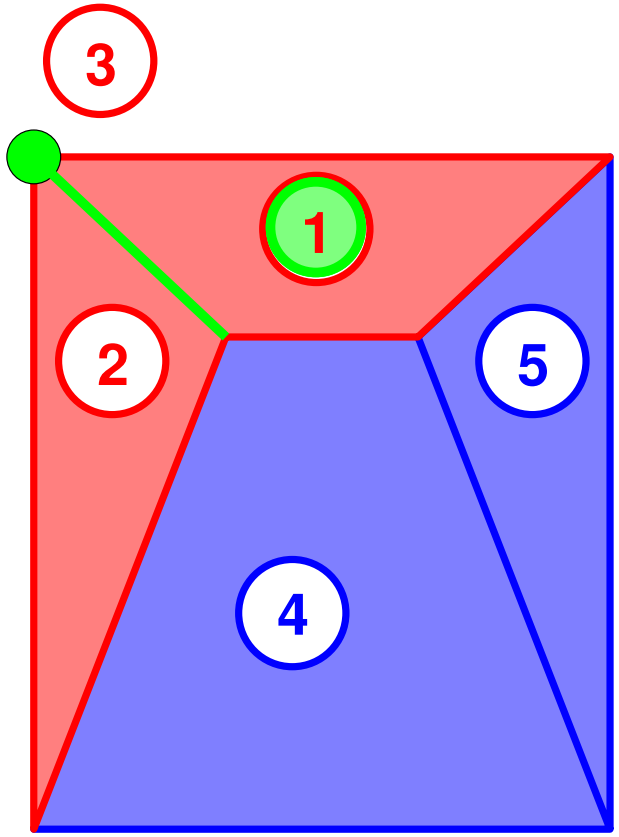


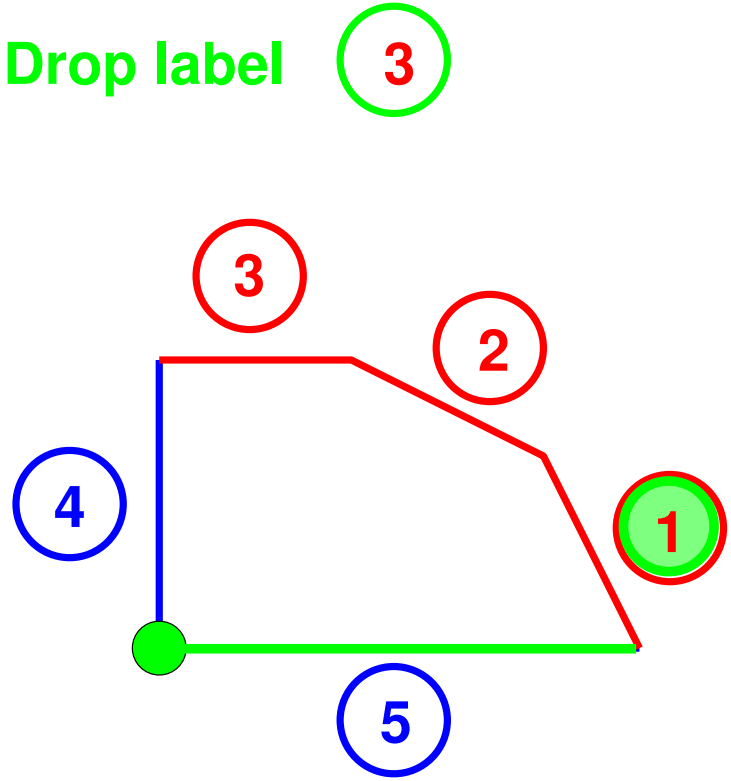
The Lemke–Howson algorithm



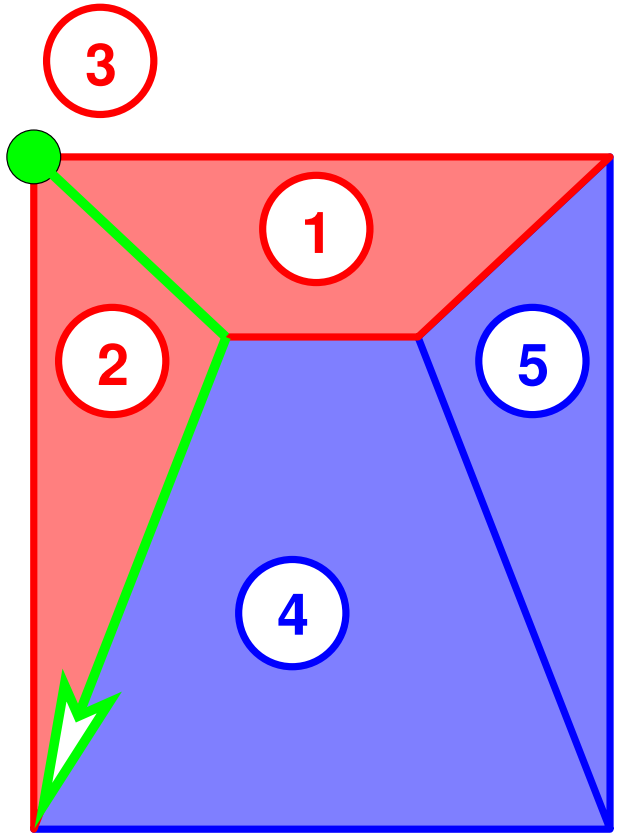


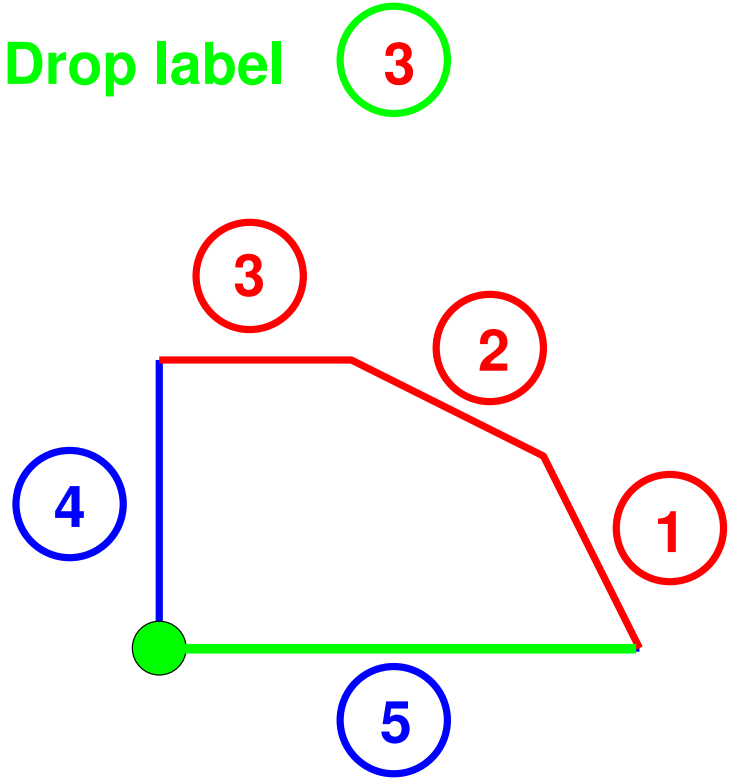
The Lemke–Howson algorithm



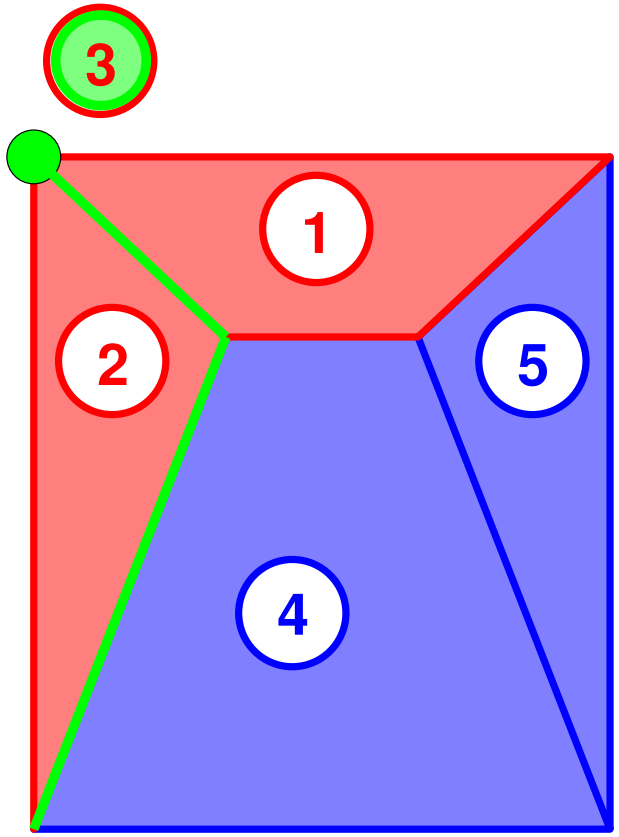


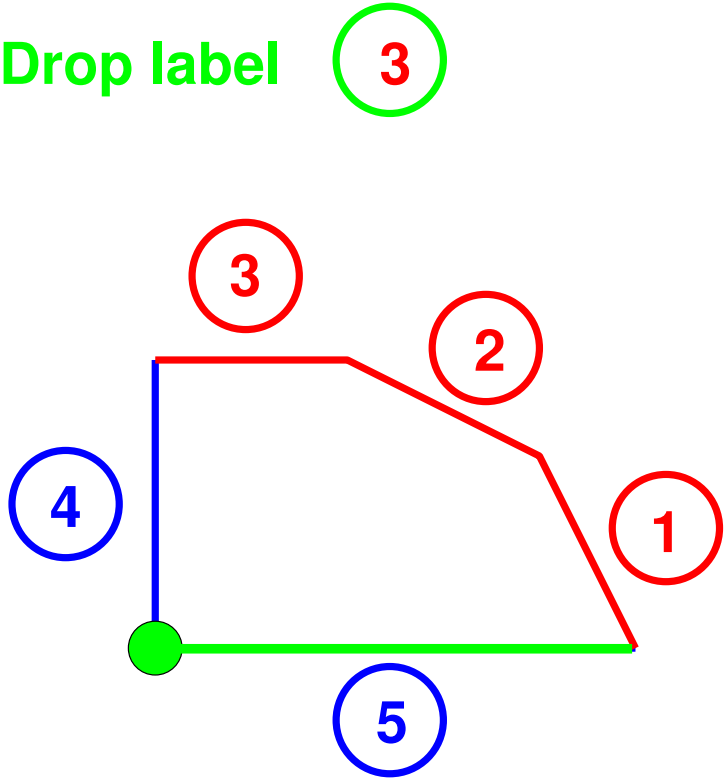
The Lemke–Howson algorithm



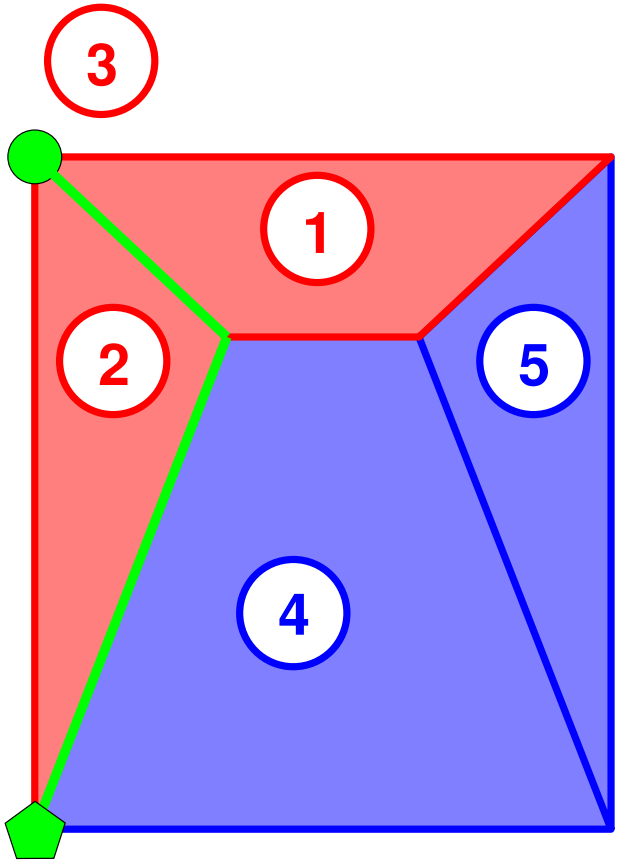


The Lemke–Howson algorithm

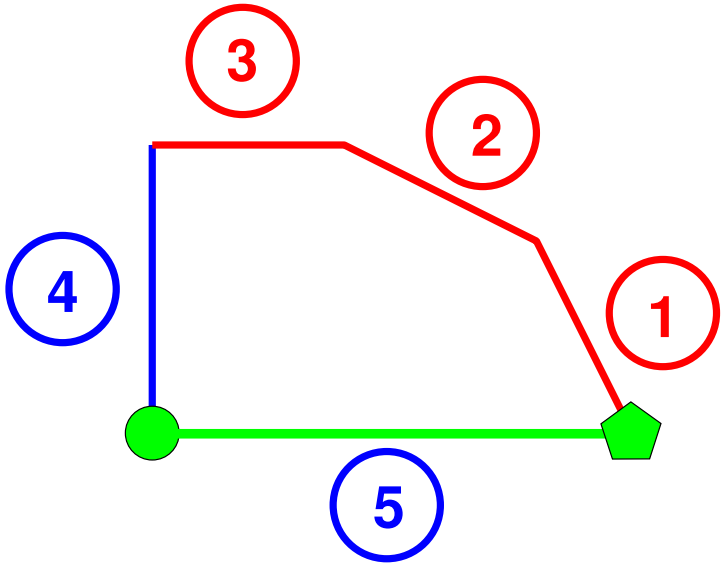




The Lemke–Howson algorithm



Drop label 3



Why Lemke-Howson works

LH finds at least one Nash equilibrium because

- finitely many "vertices"

for nondegenerate (generic) games:

- **unique** starting edge given missing label
- **unique** continuation

$\Rightarrow$ precludes "coming back" like here:

Complexity of Lemke-Howson

- finds at least one Nash equilibrium,
  pivots like Simplex algorithm for linear programming

- Simplex may be exponential [Klee-Minty cubes]

- exponentially many steps of Lemke-Howson
  for **any** dropped label?

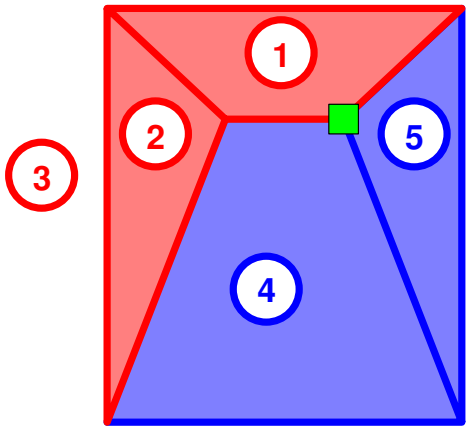
- **Yes**! This is our result.

Our result

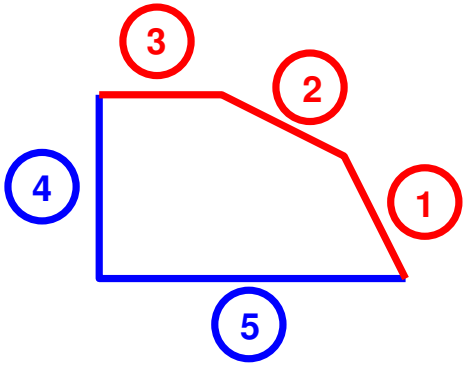
There are $d \times d$ games with exactly one Nash equilibrium, for which the Lemke-Howson algorithm takes $\geq \phi^{3d/4}$ many steps for **any dropped label** (with Golden Ratio $\phi = (\sqrt{5} + 1) / 2 = 1.618...$)

We will show this using **dual cyclic polytopes**.

Vertices as bit patterns



P

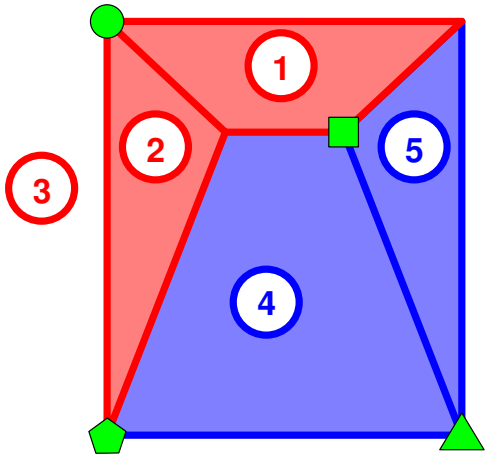


Q

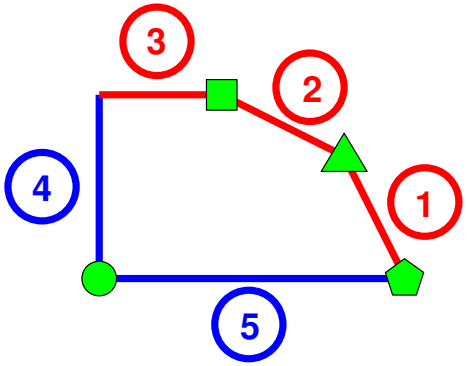
	1	2	3	4	5
	1	1	1	0	0
	1	1	0	1	0
	1	0	1	0	1
■	1	0	0	1	1
	0	0	1	1	1
	0	1	1	1	0

1	2	3	4	5
0	0	0	1	1
0	0	1	1	0
0	1	1	0	0
1	1	0	0	0
1	0	0	0	1

Vertices as bit patterns



P



Q

	1	2	3	4	5
●	1	1	1	0	0
	1	1	0	1	0
	1	0	1	0	1
■	1	0	0	1	1
▲	0	0	1	1	1
⬟	0	1	1	1	0

1	2	3	4	5
0	0	0	1	1
0	0	1	1	0
0	1	1	0	0
1	1	0	0	0
1	0	0	0	1

Dual cyclic polytopes

- vertices = strings of **n** bits with **d** bits "**1**",
- **no** odd substrings 010, 01110, 0111110, . . .
  [Gale **even**ness]

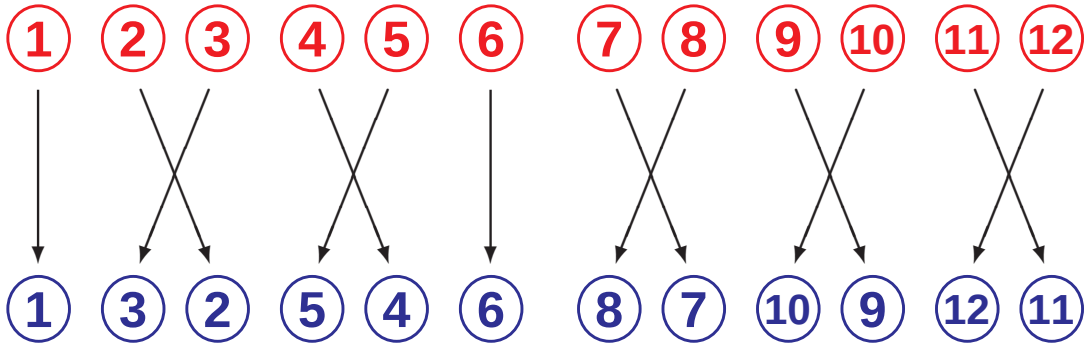
Example: d=4, n=6 d=2, n=6 (4×2 game)

d=4, n=6	d=2, n=6
111100	000011
111001	000110
110110	001100
110011	011000
101101	110000
100111	100001
011110	
011011	
001111	

Permuted labels

P = dual cyclic polytope in dimension **d** with **2d** facets

with facets labeled



Q = P

with facets labeled

$\Longrightarrow$ only **one** non-artificial equilibrium:

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

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

Lemke–Howson will take long to find it!

Lemke-Howson on dual cyclic polytopes

P								Q							
①	②	③	④	⑤	⑥	⑦	⑧	①	③	②	④	⑥	⑤	⑧	⑦
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1

Lemke-Howson on dual cyclic polytopes

P								Q							
1	2	3	4	5	6	7	8	1	3	2	4	6	5	8	7
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0								

Lemke-Howson on dual cyclic polytopes

P								Q							
①	②	③	④	⑤	⑥	⑦	⑧	①	③	②	④	⑥	⑤	⑧	⑦
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1

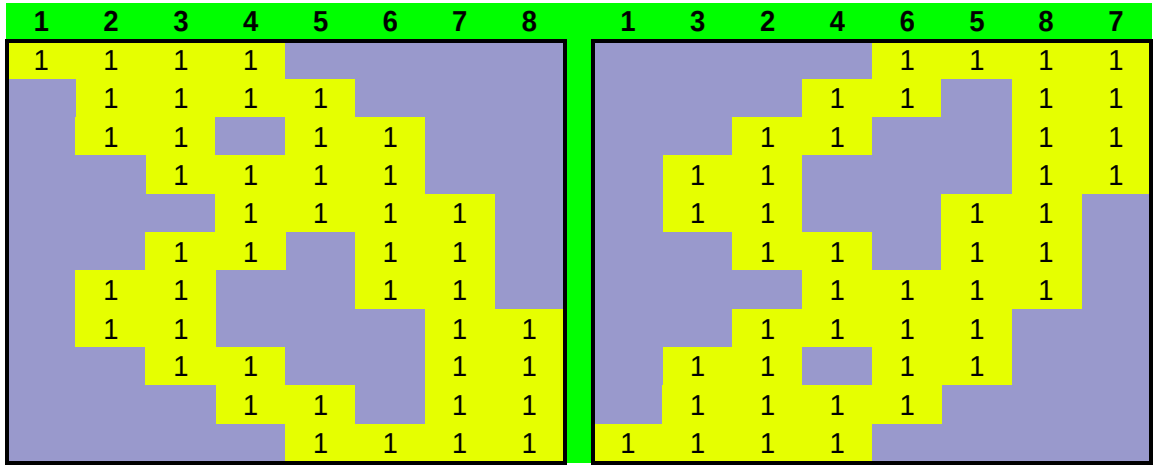
Lemke-Howson on dual cyclic polytopes

P								Q							
①	②	③	④	⑤	⑥	⑦	⑧	①	③	②	④	⑥	⑤	⑧	⑦
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0								

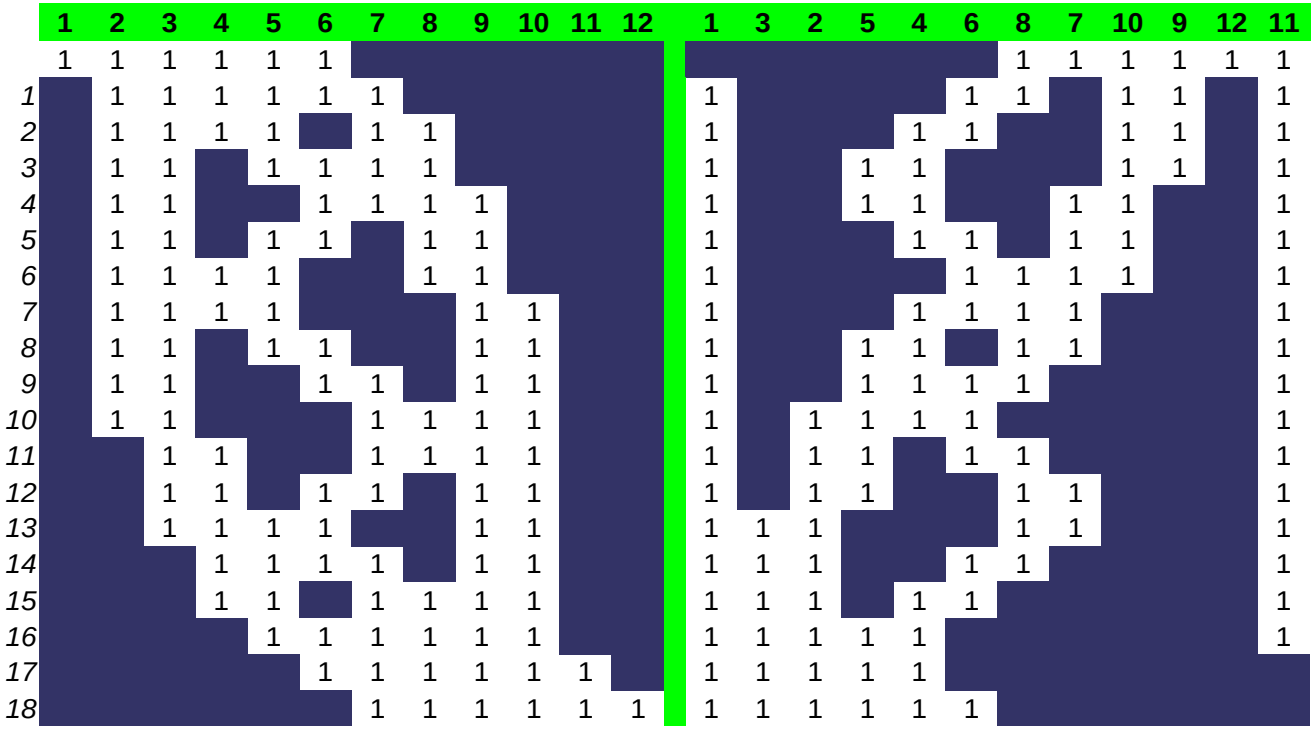
Lemke-Howson on dual cyclic polytopes

P								Q							
①	②	③	④	⑤	⑥	⑦	⑧	①	③	②	④	⑥	⑤	⑧	⑦
1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
0	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1
0	1	1	0	1	1	0	0	0	0	1	1	0	0	1	1
0	0	1	1	1	1	0	0	0	1	1	0	0	0	1	1
0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0
0	0	1	1	0	1	1	0	0	0	1	1	0	1	1	0
0	1	1	0	0	1	1	0	0	0	0	1	1	1	1	0
0	1	1	0	0	0	1	1	0	0	1	1	1	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	1	1	0	0
0	0	0	1	1	0	1	1	0	1	1	1	1	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0

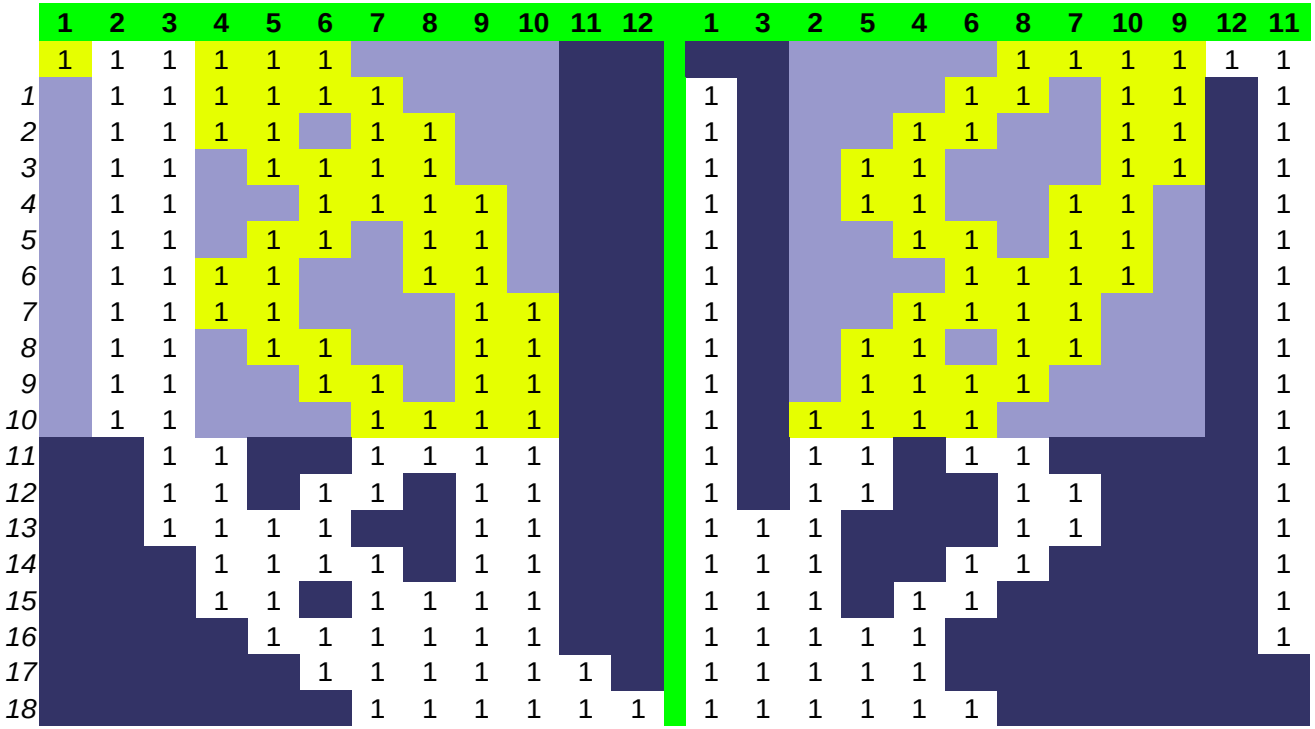
A(4) = path for d=4, label 1



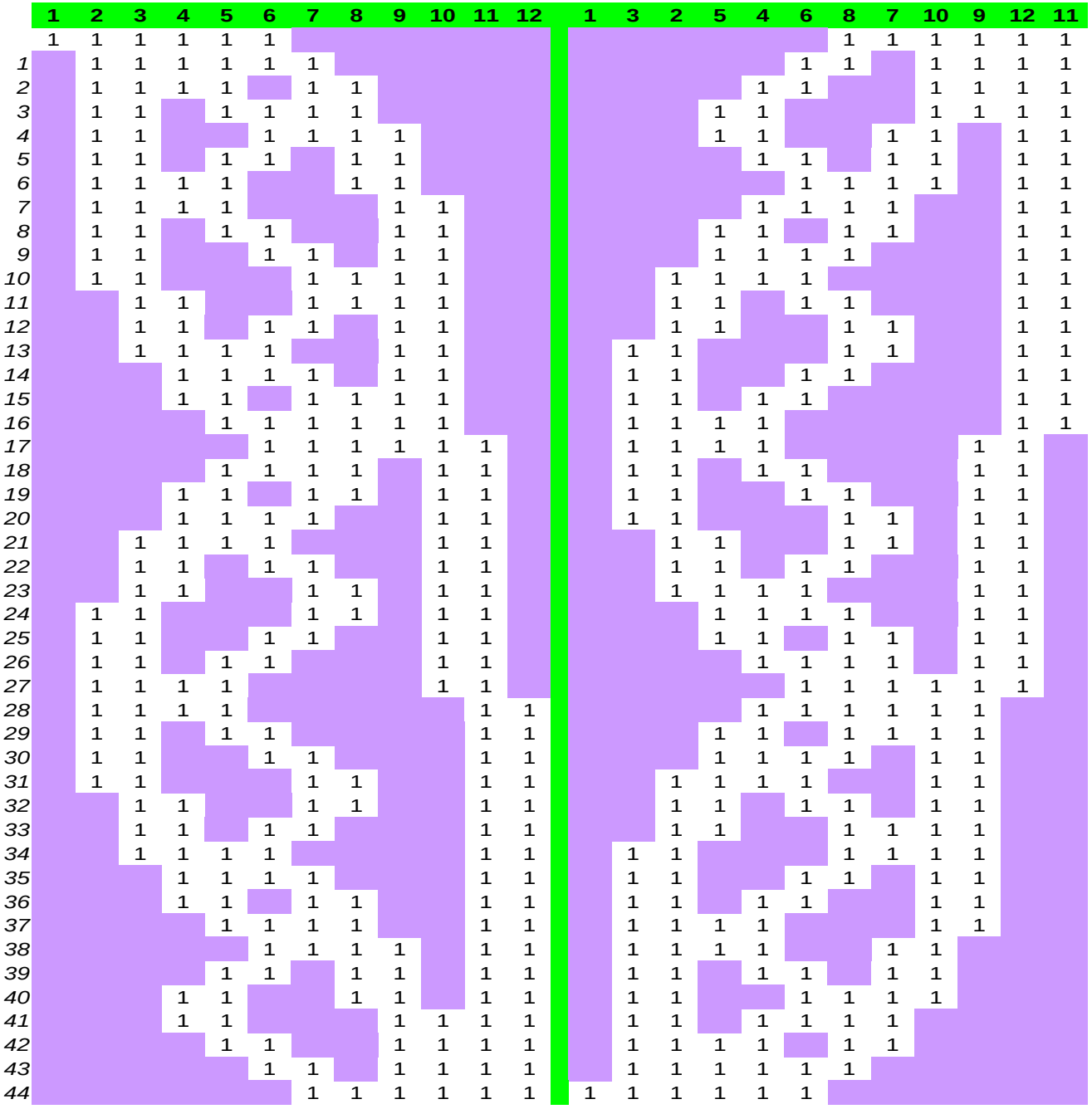
B(6) = path for d=6, label 12



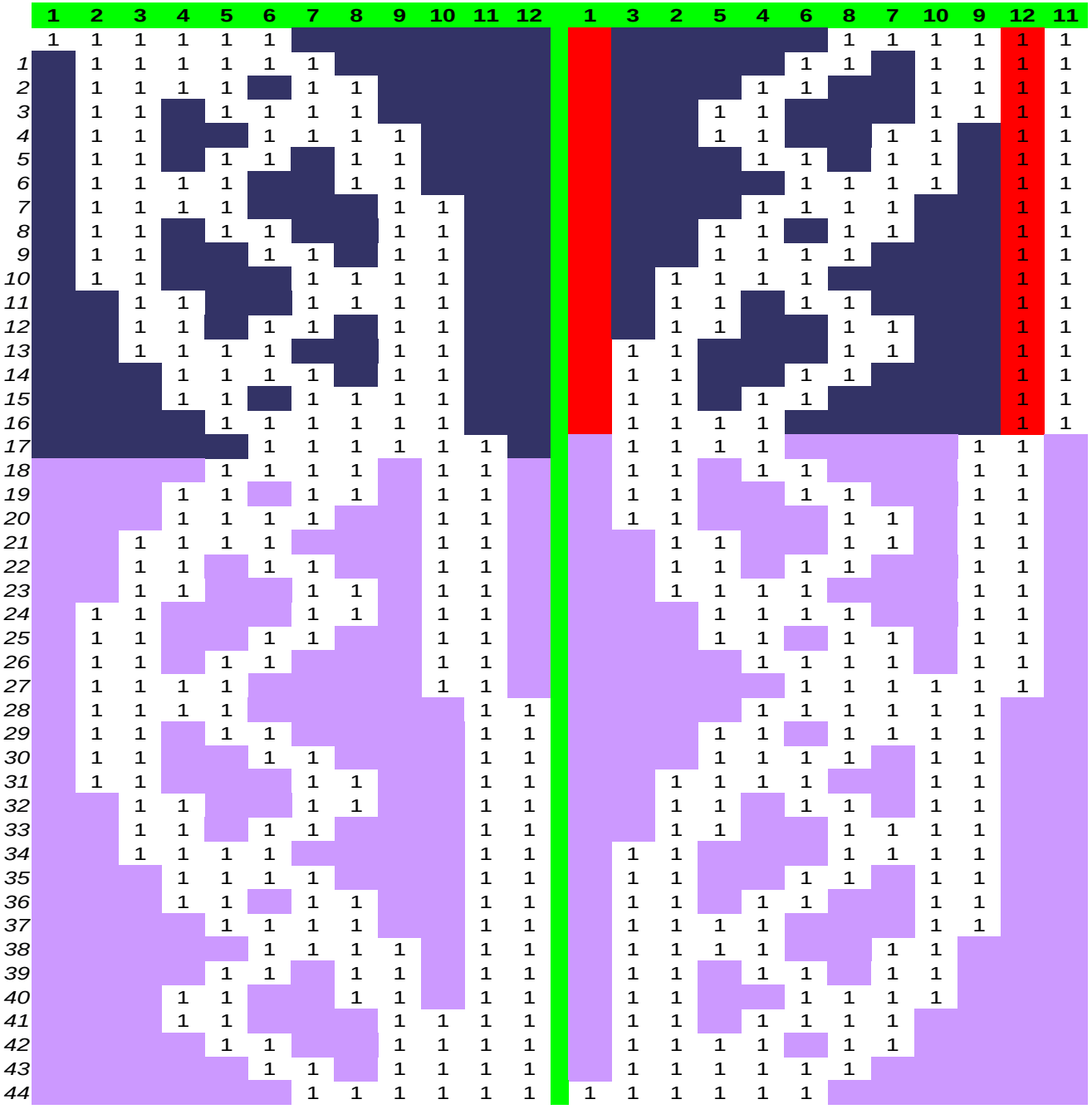
A(4) is prefix of B(6)



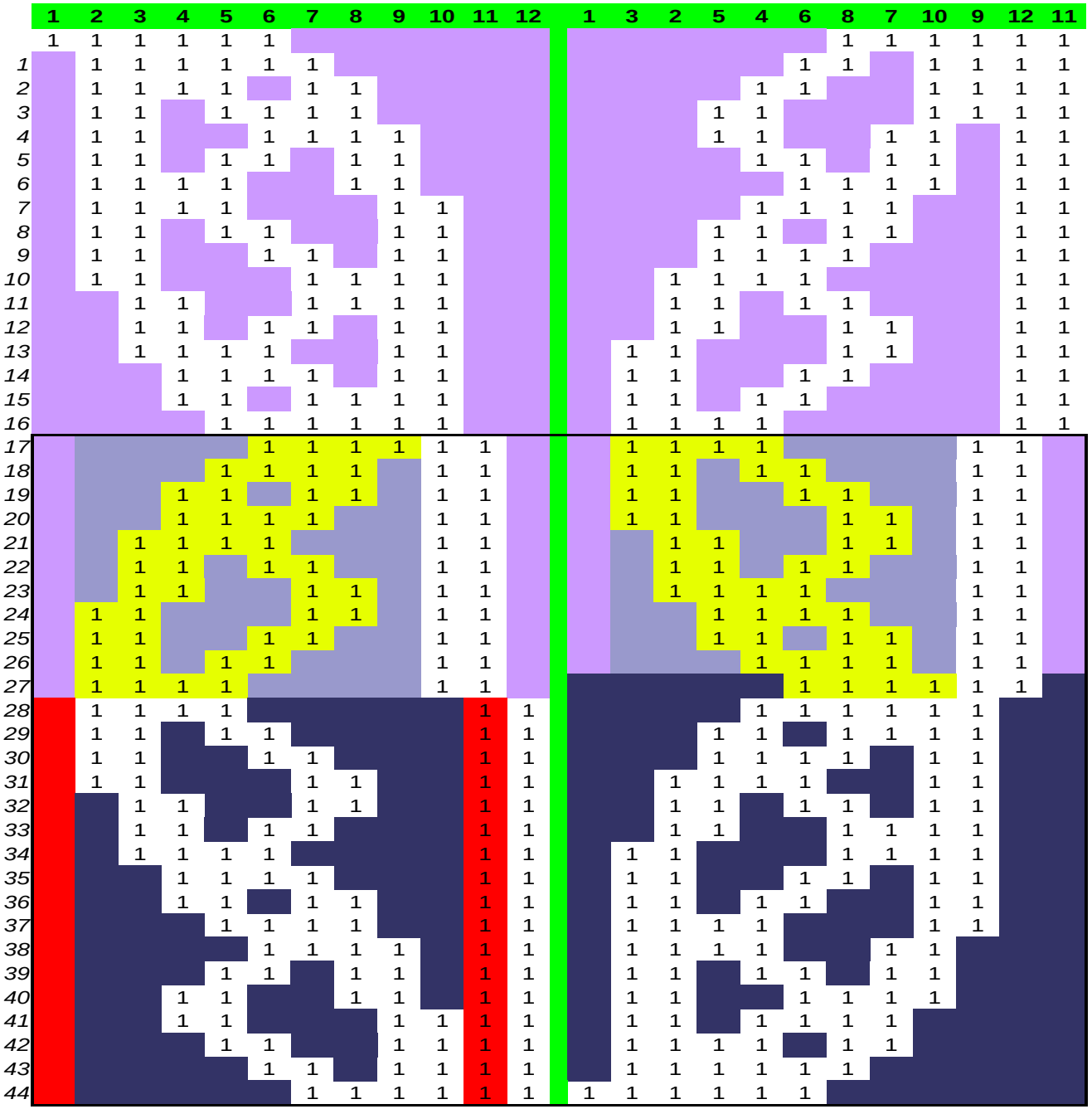
A(6) = path for d=6, label 1



B(6) is prefix of A(6)



Suffix of A(6) = C(6) = A(4)+B(6)



Recurrences for longest paths

A(d) = LH path dropping label 1 in dim d

B(d) = LH path dropping label 2d in dim d

C(d) = suffix of A(d)

lengths of

B(2) C(2) **A**(2) B(4) C(4) **A**(4) B(6) C(6) **A**(6) . . .

are the **Fibonacci** numbers

2 3 5 8 13 21 34 55 89 . . .

Exponential path lengths

longest paths: drop label 1 or 2d, paths A(d), B(d)

path length $\Omega(\phi^{3d/2})$

with Golden Ratio $\phi = (\sqrt{5} + 1) / 2 = 1.618...$

shortest paths: drop label 3d/2, path B(d/2)+B(d/2+2)

path length $\Omega(\phi^{3d/4}) = \Omega(\mathbf{1.434...}^{d})$

Summary and extensions

- NE of a bimatrix game = combinatorial polytope problem
- label dual cyclic polytopes,
  equilibrium at end of **exponentially long** paths
- **but**: fully mixed equilibrium easily guessed
  by support enumeration algorithms
- can extend to $d \times 2d$ games with hard-to-guess
  support (exponentially many guesses on average)
  and exponentially long paths

The following song is a cover version of an original by Billy Joel, "The longest time".

This version by Daniel Barrett, who wrote it as a graduate student at Johns Hopkins University, "on May 1, 1988, during a difficult Algorithms II final exam", and subsequently recorded it.

Woh oh-oh-oh find the longest path

Woh oh-oh find the longest path.

If you say P is NP tonight
there would still be
papers left to write
I have a weakness
I'm addicted to completeness
and I keep searching
for the longest path.

The algorithm I would like to see
is of polynomial degree
but it's elusive
nobody has found conclusive
evidence that we can find
the longest path.

I have been
hard working for so long
I swear it's right
and he marks it wrong
somehow I feel
sorry when it's done
GPA 2.1
is more than I hope for

Garey, Johnson,
Karp and other men (and women, too)
try to make it order N log N
am I a mad fool
if I spend my life in grad school
forever following the longest path

Woh oh-oh-oh find the longest path
Woh oh-oh find the longest path
Woh oh-oh find the longest path.