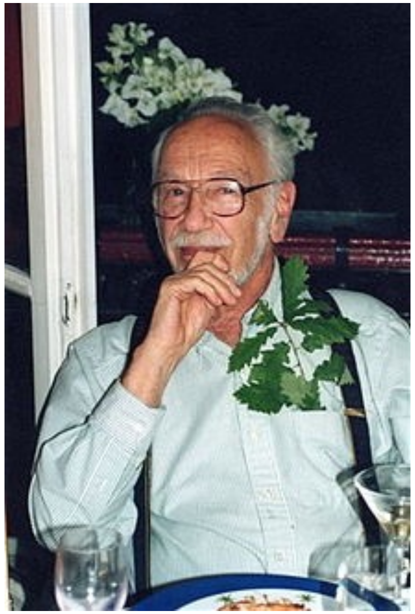
15-251
Great Theoretical Ideas in Computer Science

Lecture 14:
Graphs IV: Stable Matchings



March 2nd, 2017

From Last Time

Bipartite maximum matching problem

Bipartite maximum matching problem

Input: A *bipartite* graph $G = (X, Y, E)$.

Output: A maximum matching in G.

Important Definition: Augmenting paths

Let **M** be some matching.

An ***augmenting path*** with respect to **M** is an alternating path such that:

- the first and last vertices are **not** matched by **M**

Algorithm to find maximum matching

Theorem:

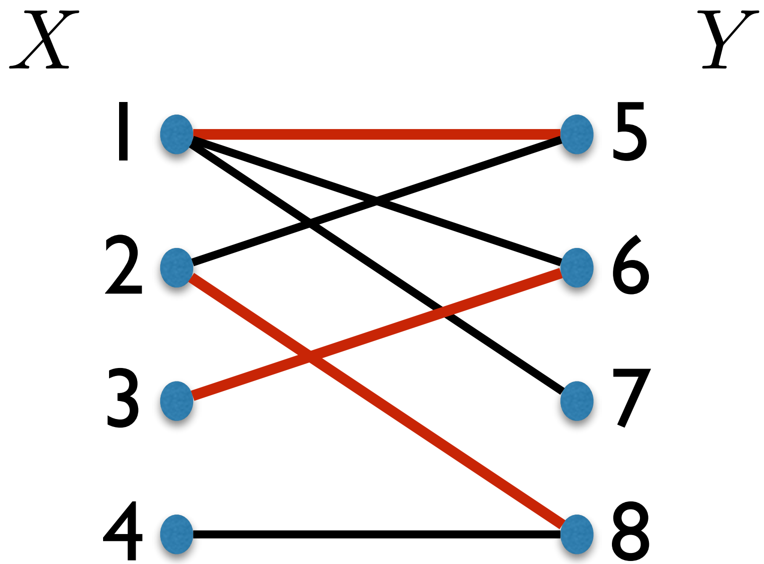
A matching **M** is maximum **if and only if** there is **no** augmenting path with respect to **M**.

Algorithm:

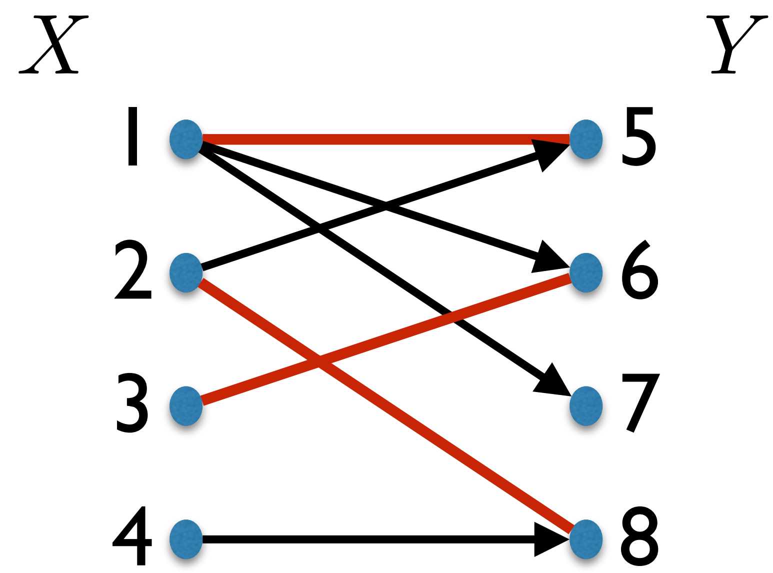
- Start with a single edge as your matching **M**.
- Repeat until there is no augmenting path w.r.t. **M**:
    - Find an augmenting path with respect to **M**.
    - Update **M** according to the augmenting path.

OK, but how do you find an augmenting path?

Algorithm to find augmenting path

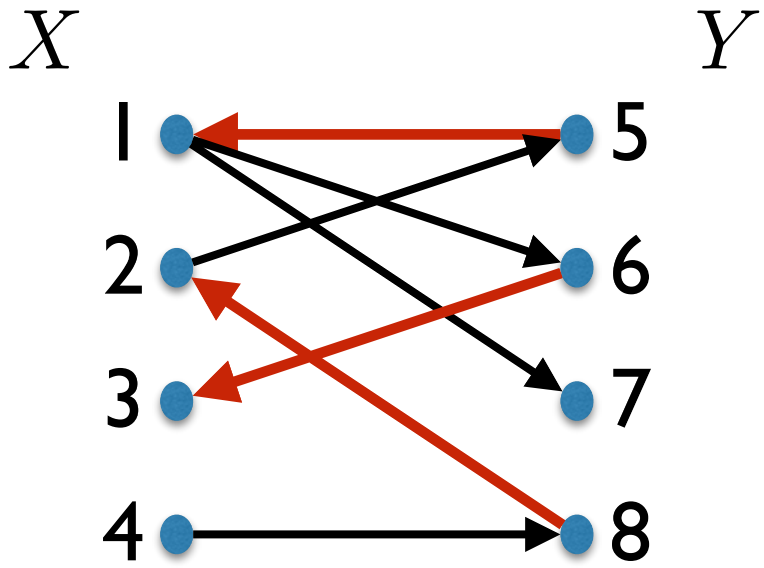


Algorithm to find augmenting path



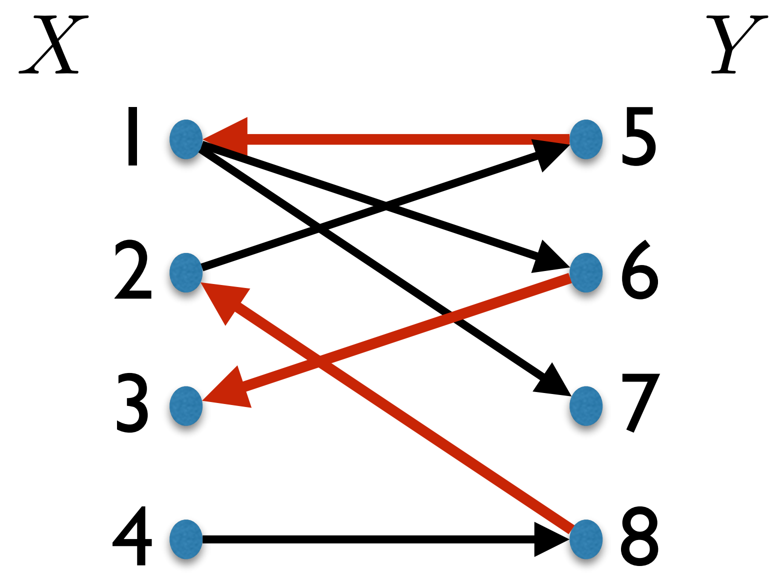
- direct edges **not** in **M** from left to right (X to Y).

Algorithm to find augmenting path



- direct edges **not** in **M** from left to right (X to Y).
- direct edges in **M** from right to left (Y to X).

Algorithm to find augmenting path

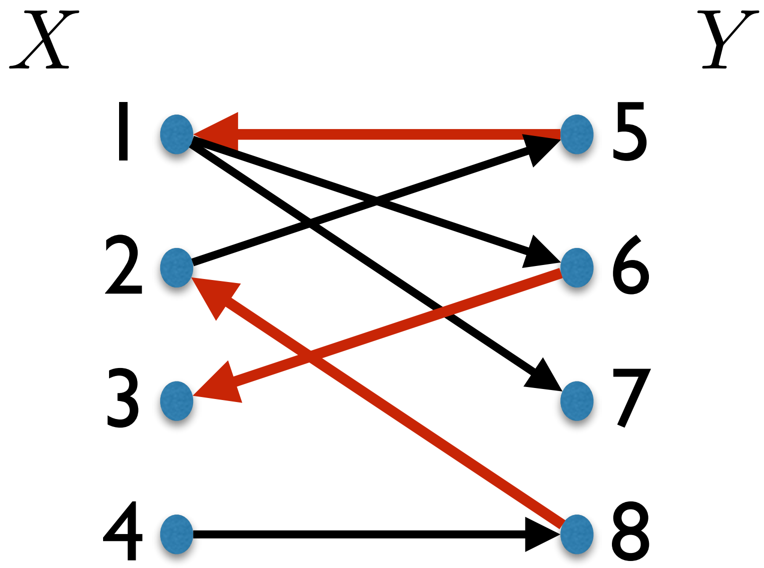


- direct edges **<u>not</u>** in **M** from left to right (X to Y).
- direct edges in **M** from right to left (Y to X).

<u>Observation:</u>

There is an augmenting path **iff** there is a directed path from an *unmatched* $x \in X$ to an *unmatched* $y \in Y$.

Algorithm to find augmenting path



Algorithm:

- for each *unmatched* $x \in X$:
    - do DFS(x), stop when you find *unmatched* $y \in Y$.

Running time: $O(n+m)$

Important Note

Theorem:

A matching **M** is maximum **if and only if** there is **no** augmenting path with respect to **M**.

This theorem holds for all graphs.

The algorithm works for bipartite graphs.

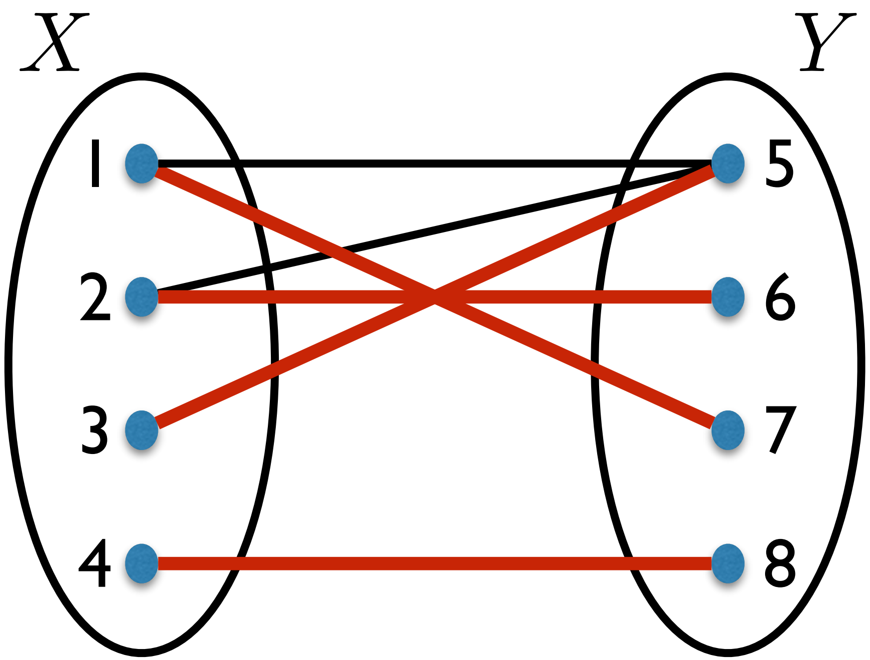
How do you solve a problem like this?

1. Formulate the problem
2. **Ask**: Is there a trivial algorithm?
3. **Ask**: Is there a better algorithm?
4. Find and analyze

Hall’s Theorem

Characterization for perfect matchings

Often we are interested in perfect matchings.

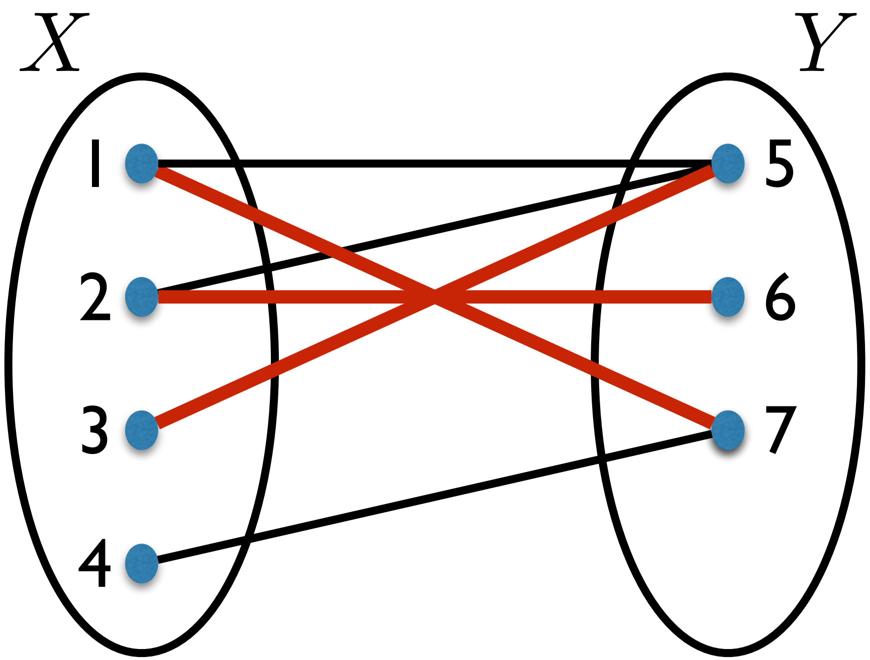


<u>An obstruction:</u>

$$|X| \neq |Y|$$

Characterization for perfect matchings

Often we are interested in perfect matchings.



An obstruction:

If $|X| > |Y|$, we cannot "cover" all the nodes in X.

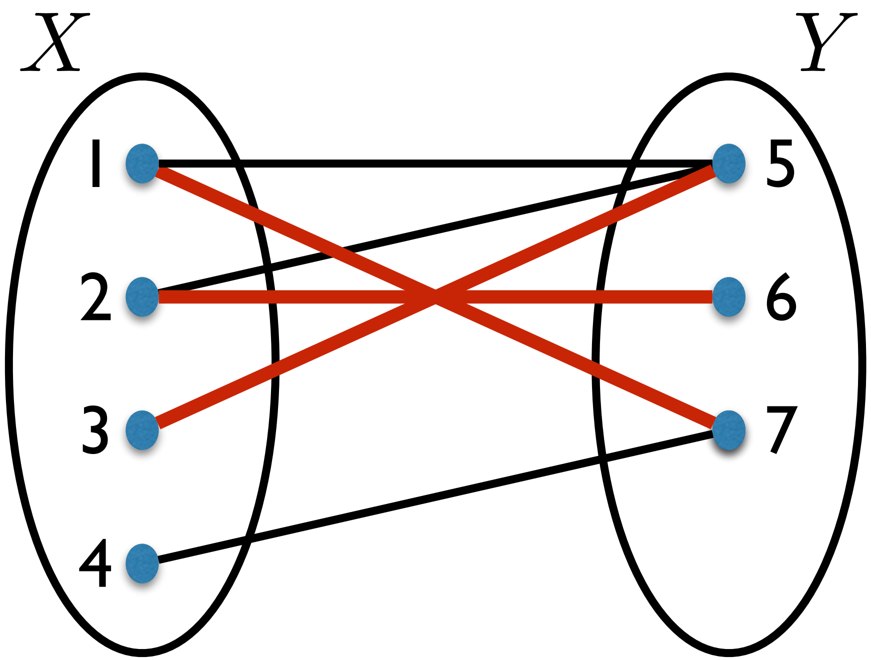
If $|X| > |N(X)|$, we cannot "cover" all the nodes in X.

Characterization for perfect matchings

Often we are interested in perfect matchings.

$S = \{1, 3, 4\}$

$N(S) = \{5, 7\}$



<u>An obstruction:</u>

For $S \subseteq X$:

if $|S| > |N(S)|$, we cannot "cover" all the nodes in S.

Characterization for perfect matchings

Is this the only type of obstruction?

Theorem [Hall's Theorem]:

Let $G = (X, Y, E)$ be a bipartite graph.

There is a matching covering all vertices in X **iff**

$$\forall S \subseteq X : \quad |S| \leq |N(S)| \, .$$

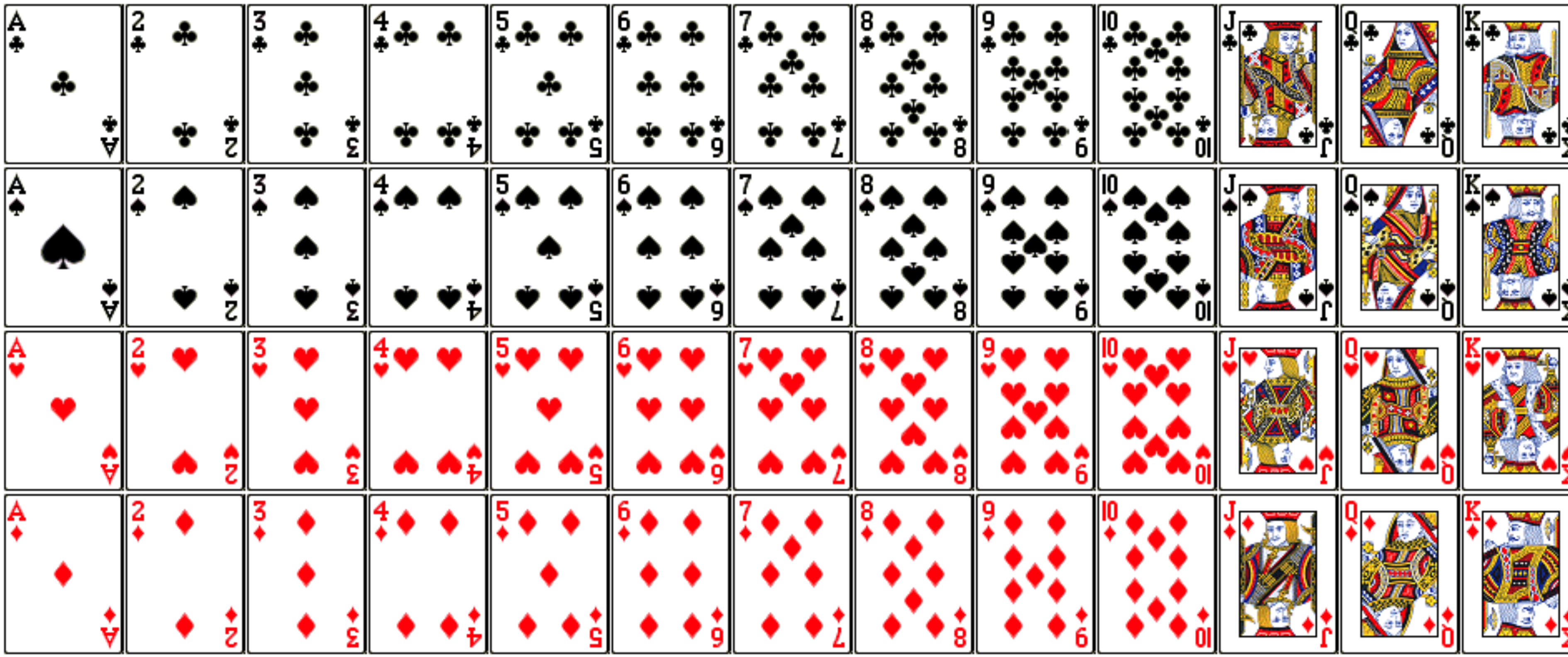
Corollary:

$G = (X, Y, E)$ has a perfect matching **iff**

$|X| = |Y|$ and $\forall S \subseteq X , \quad |S| \leq |N(S)| \, .$

An application of Hall’s Theorem

Rank: 1 2 3 4 5 6 7 8 9 10 J Q K

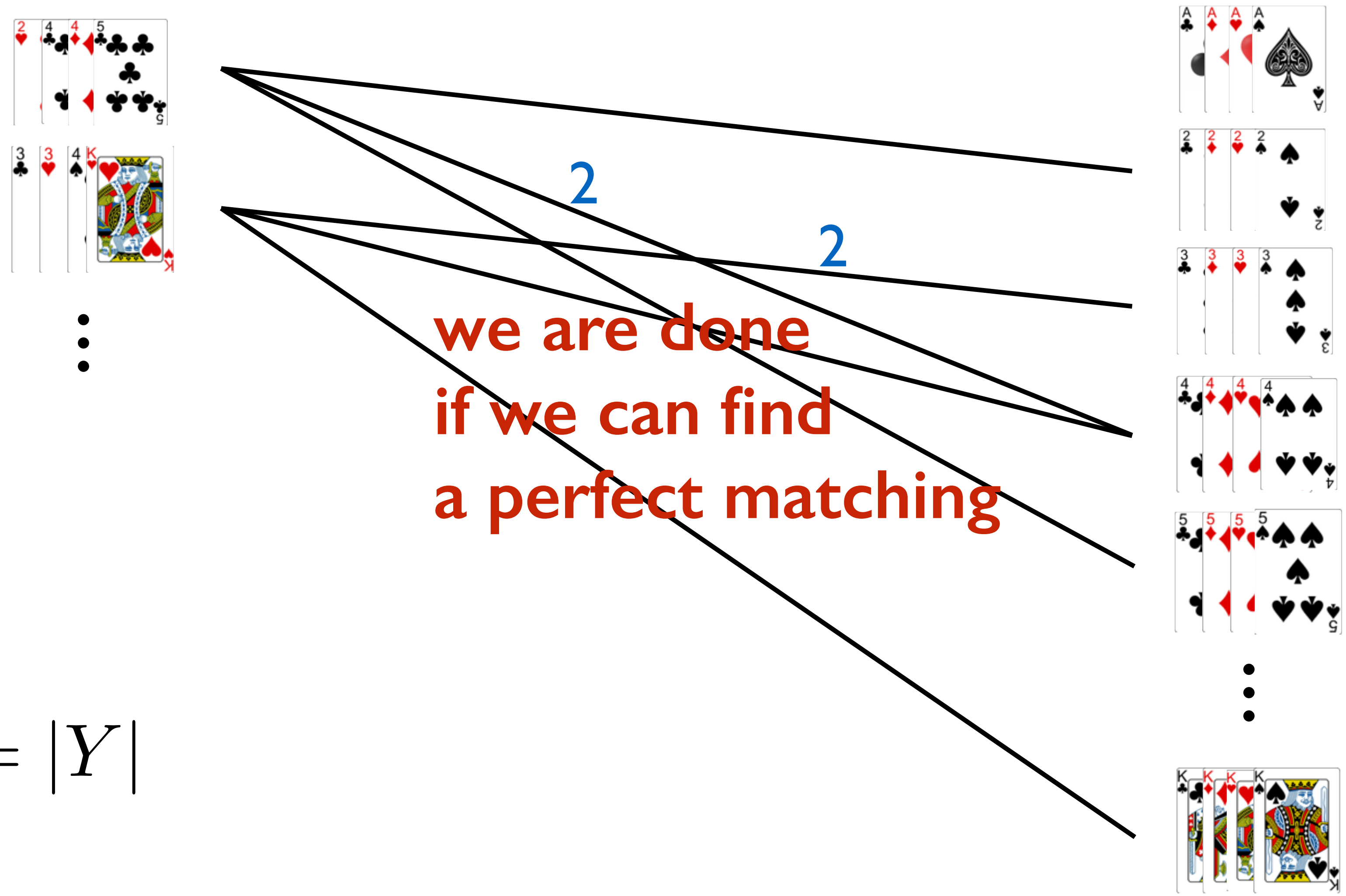


Suppose a deck of cards is dealt into 13 piles of 4 cards each.

Claim: there is a way to select one card from each pile so that you have one card from each rank.

An application of Hall's Theorem

X



Y

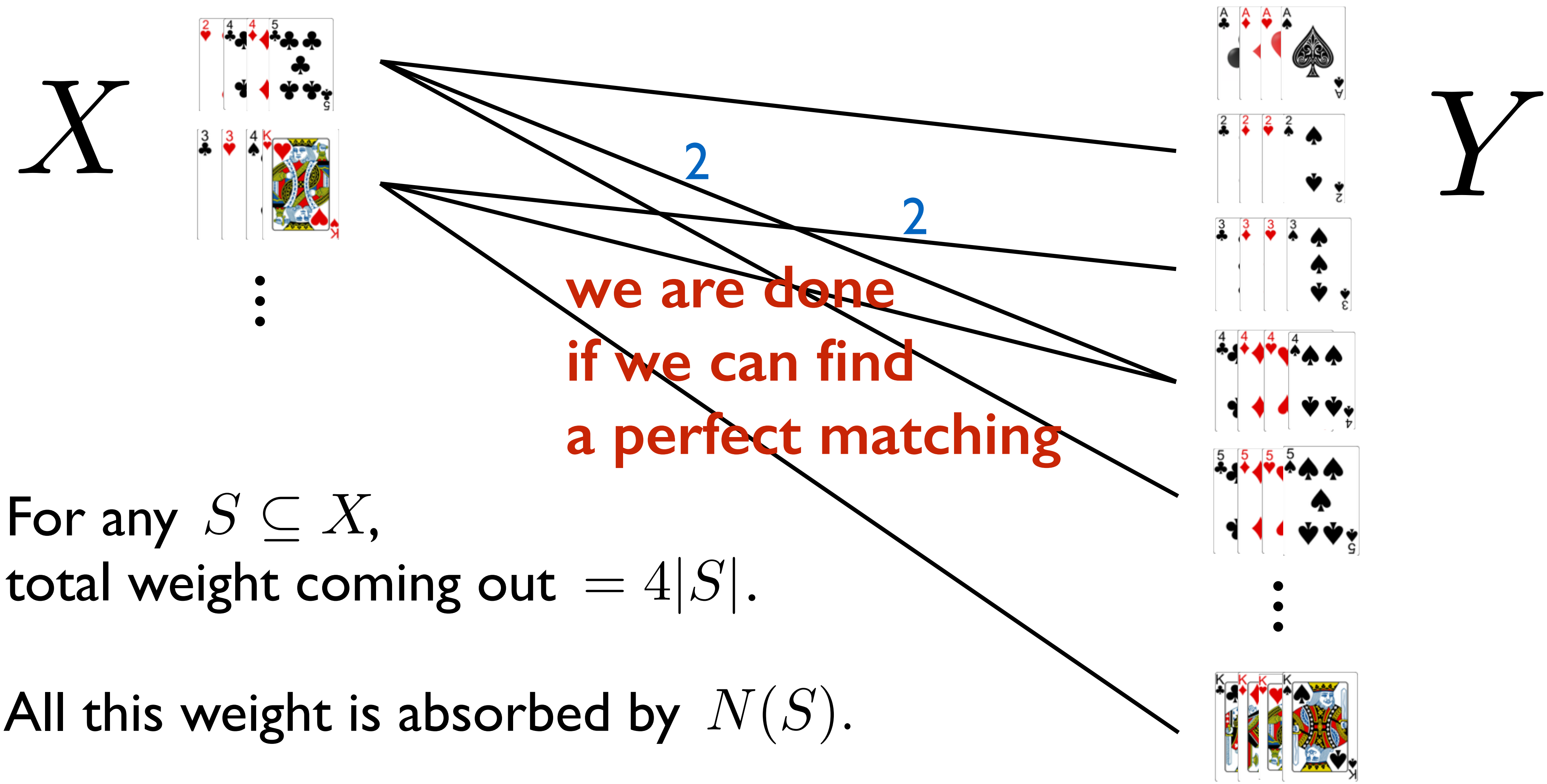
we are done
if we can find
a perfect matching

$$|X| = |Y|$$

<u>**Want to show:**</u>

For any $S \subseteq X$, $|S| \leq |N(S)|$.

An application of Hall's Theorem



For any $S \subseteq X$,
total weight coming out $= 4|S|$.

All this weight is absorbed by $N(S)$.

Each $y \in N(S)$ absorbs ≤ 4 units of this weight.

$\implies N(S)$ absorbs ≤ $4|N(S)|$ units. $\implies \cancel{4}|S| \leq \cancel{4}|N(S)|$

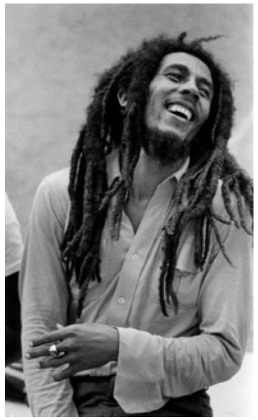
Stable matching problem

2-Sided Markets

A market with 2 distinct groups of participants each with their own preferences.

2-Sided Markets

1. Moogle Google
2. Macrosoft
3. KLG
4. UMBRELLA CORPORATION "Our business is life itself..."



Macrosoft

Moogle Google

UMBRELLA CORPORATION

"Our business is life itself..."

KLG

1. Alice
2. Bob
3. Charlie
4. David

•
•
•

1. Bob
2. David
3. Alice
4. Charlie

Other examples:

medical residents - hospitals

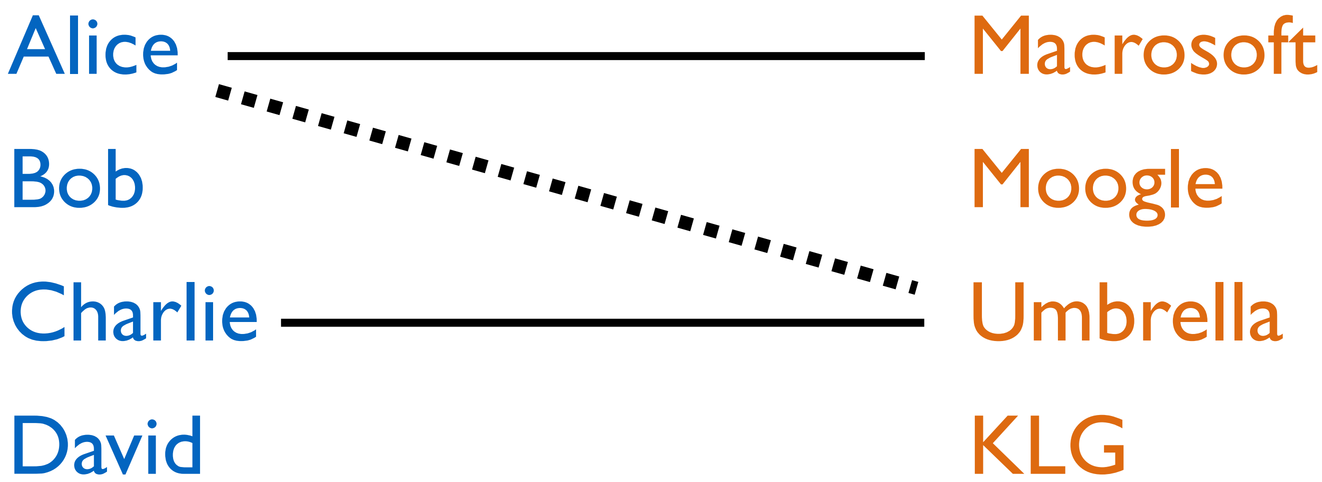
students - colleges

professors - colleges

⋮

Aspiration: A Good Centeralized System

What can go wrong?

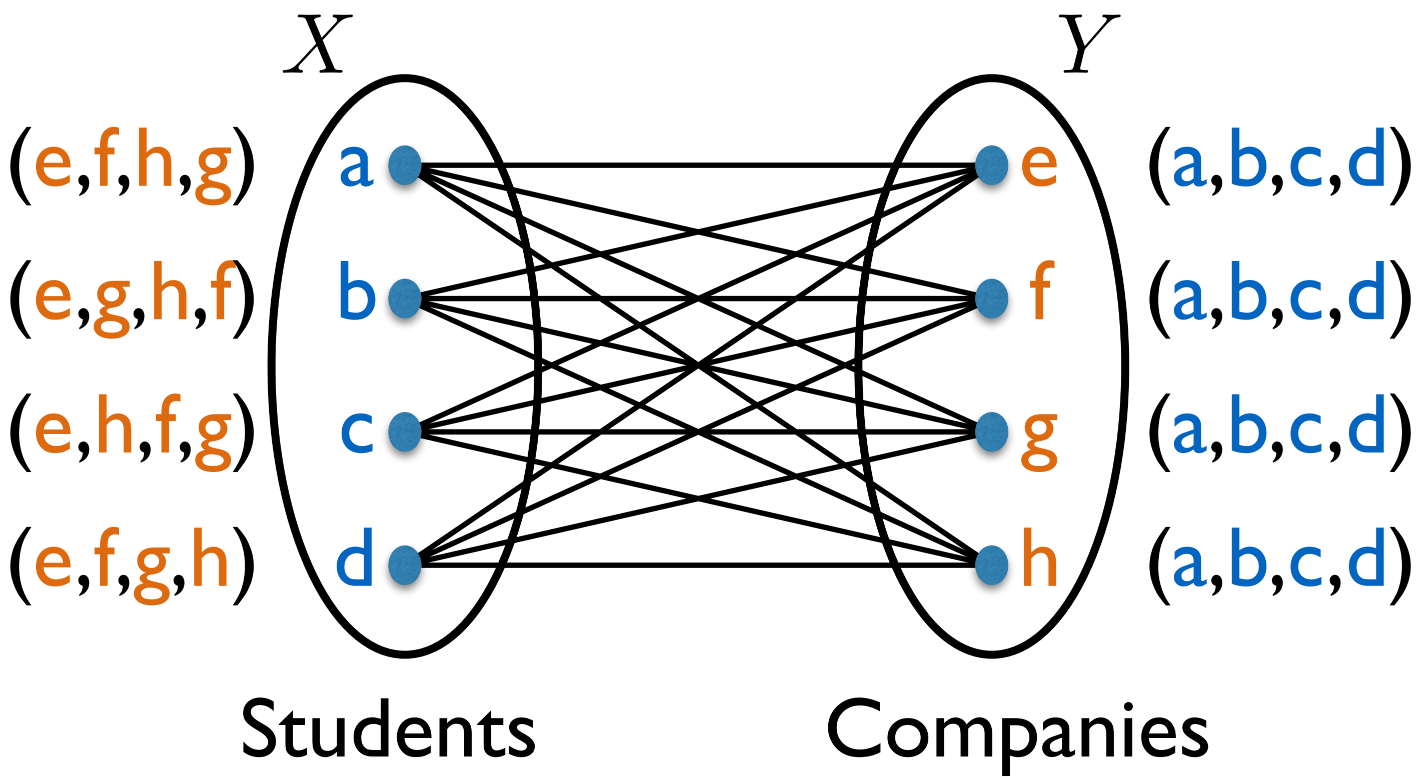


Suppose Alice gets "matched" with Macrosoft.
Charlie gets "matched" with Umbrella.

But, say, Alice prefers Umbrella over Macrosoft and Umbrella prefers Alice over Charlie.

Formalizing the problem

An instance of the problem can be represented as a *complete bipartite graph* + *preference list of each node.*

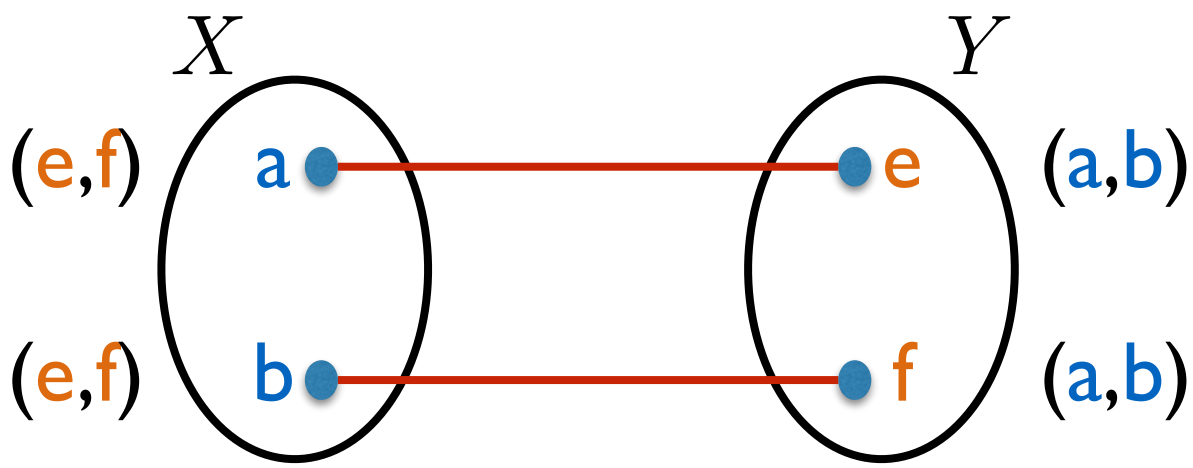


$$|X| = |Y| = n$$

Goal: Find a stable matching.

Formalizing the problem

What is a stable matching?



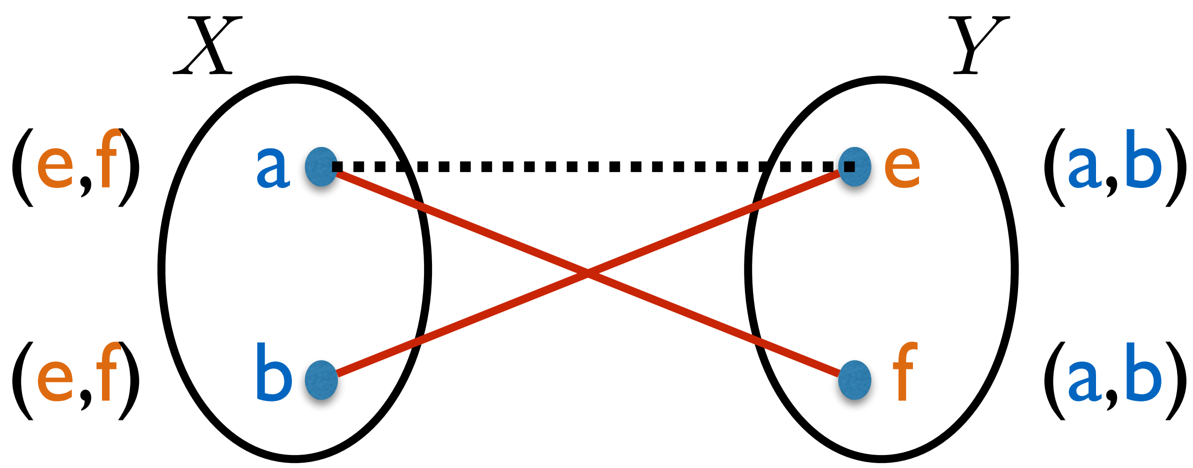
1. It has to be a perfect matching.

2. Cannot contain an unstable pair:

   A pair (x, y) <u>unmatched</u>
   but they prefer each other over their current partners.

Formalizing the problem

What is a stable matching?



(a, e) is an unstable pair.

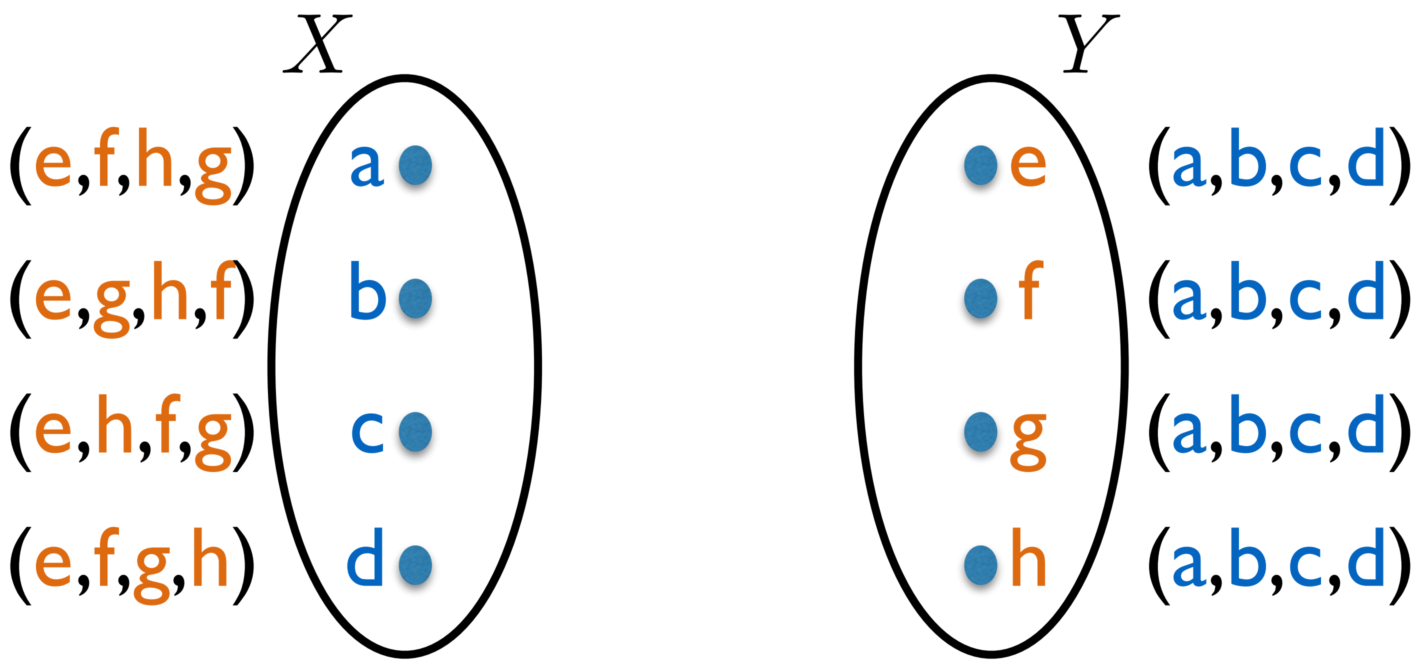
1. It has to be a perfect matching.

2. Cannot contain an unstable pair:

A pair (x, y) unmatched
but they prefer each other over their current partners.

Formalizing the problem

An instance of the problem can be represented as a *complete bipartite graph* + *preference list of each node.*



$$|X| = |Y| = n$$

Goal: Find a stable matching.

(Is it guaranteed to always exist?)

A variant: Roommate problem

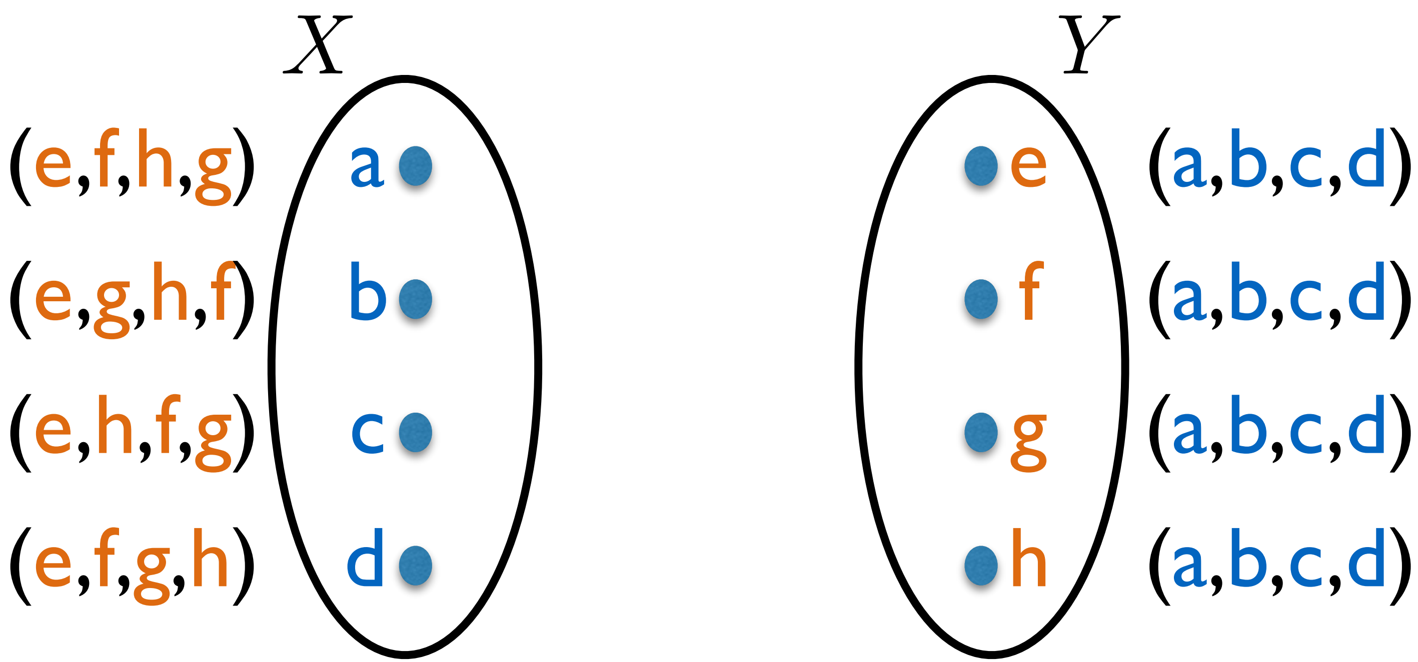
A non-bipartite version

(c,b,d) a • • c (b,a,d)

(a,c,d) b • • d (a,c,b)

Does this have a stable matching?

Stable matching: Is there a trivial algorithm?

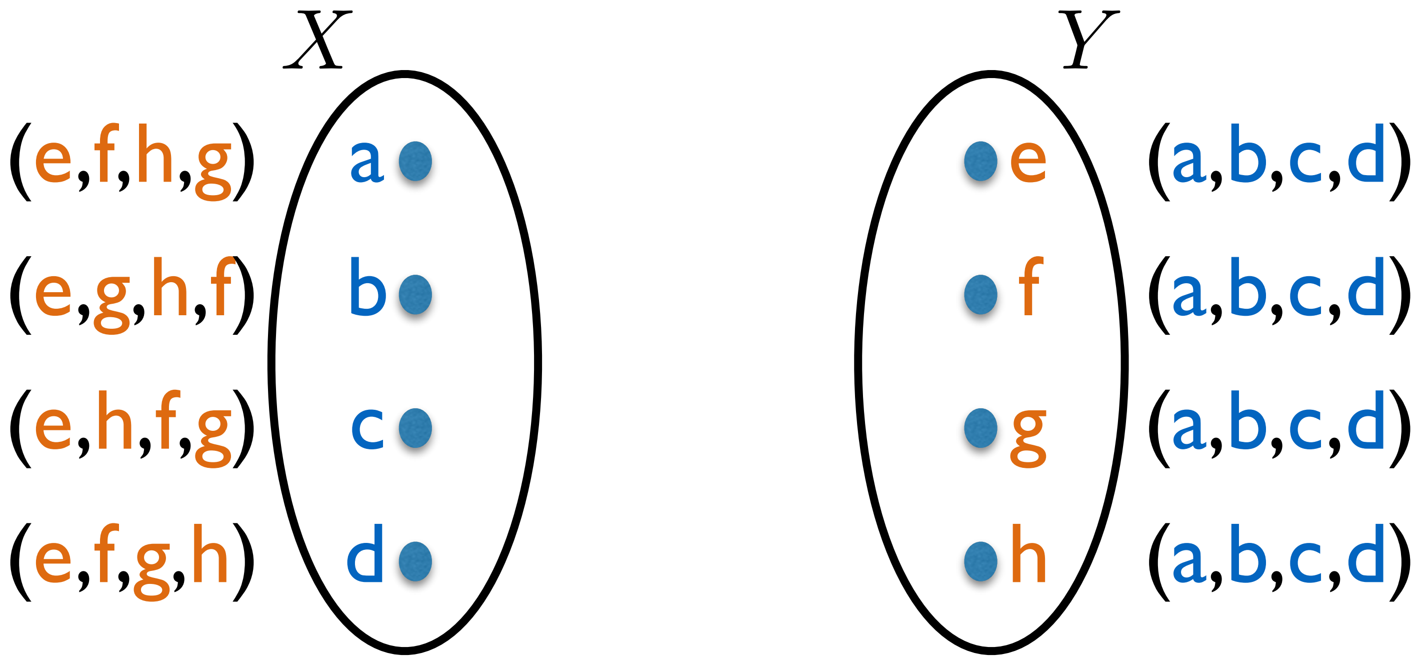


Trivial algorithm:

Try all possible perfect matchings,
and check if it is stable.

perfect matchings in terms $n = |X|$:

Stable matching: Is there a trivial algorithm?

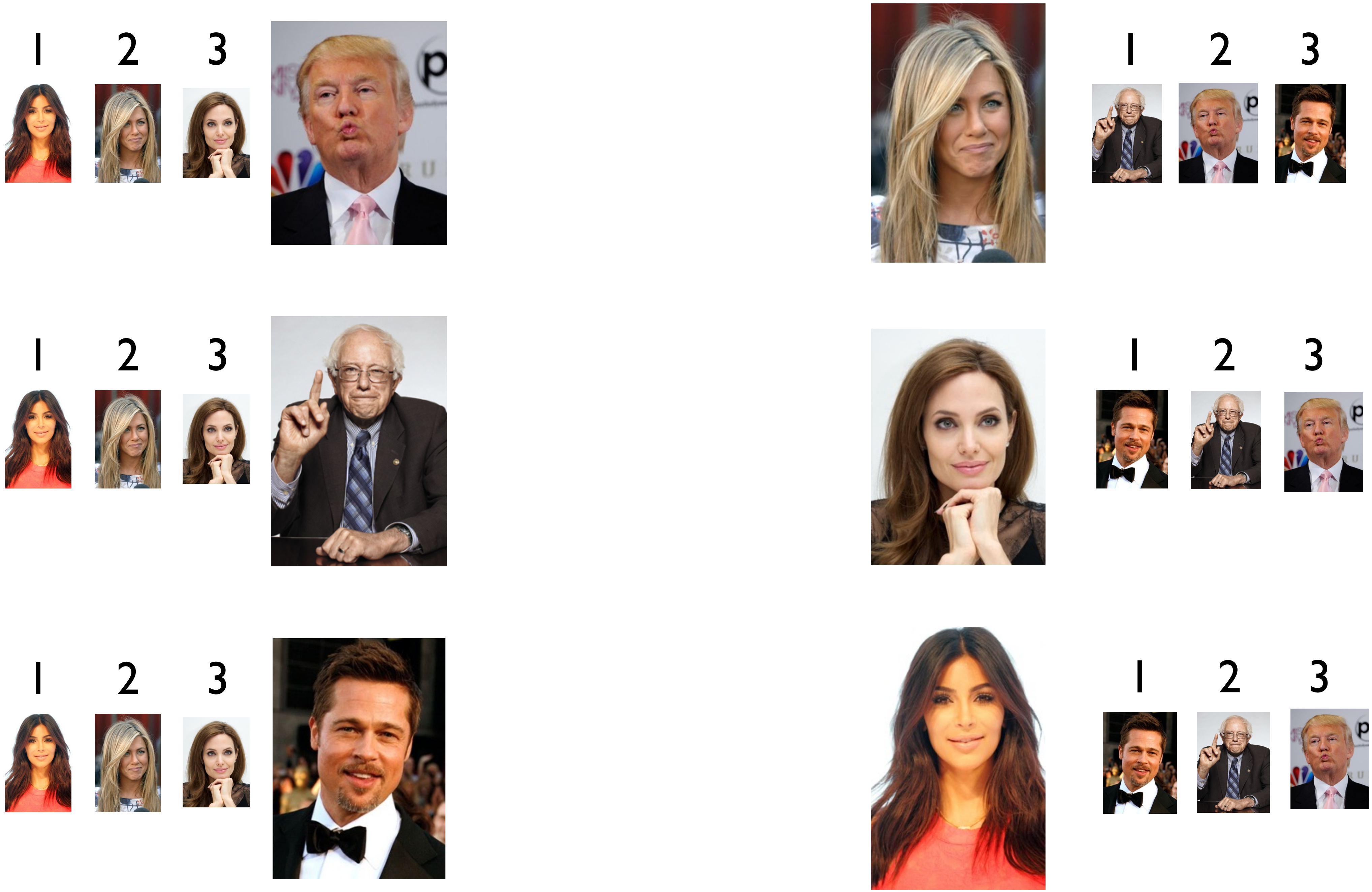


Trivial algorithm:

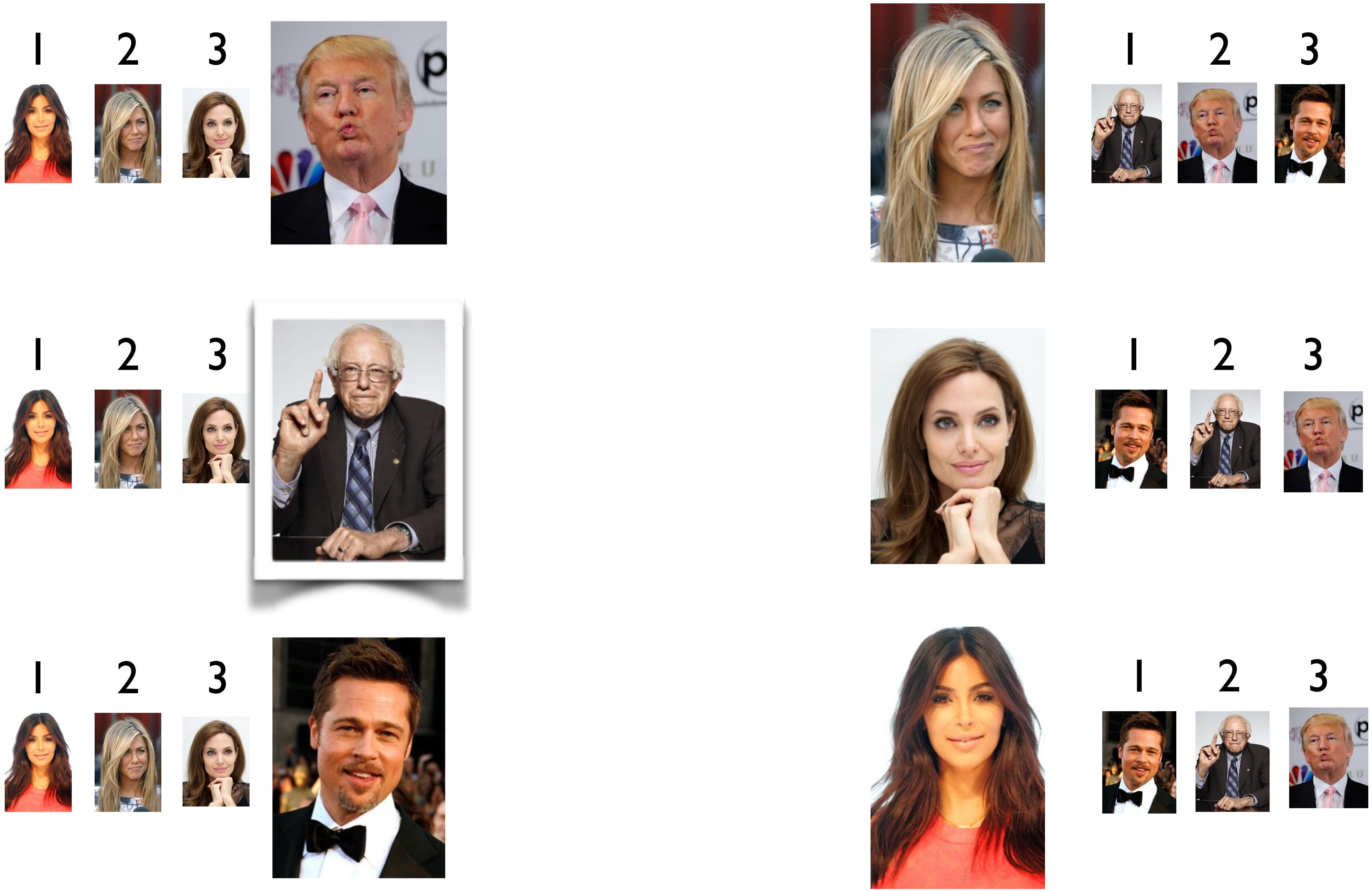
Try all possible perfect matchings,
and check if it is stable.

\# perfect matchings in terms $n = |X|$: $n!$

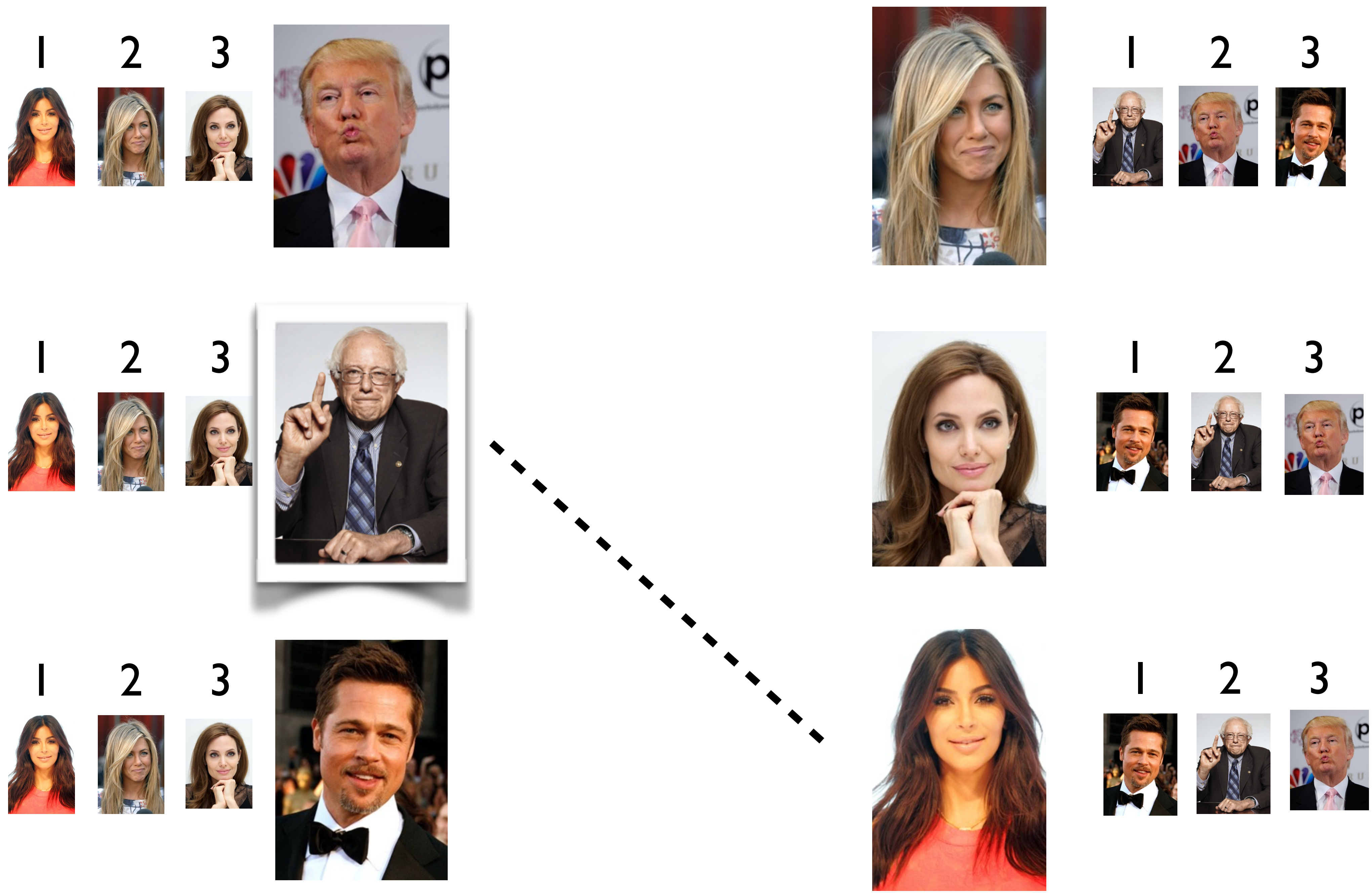
The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm

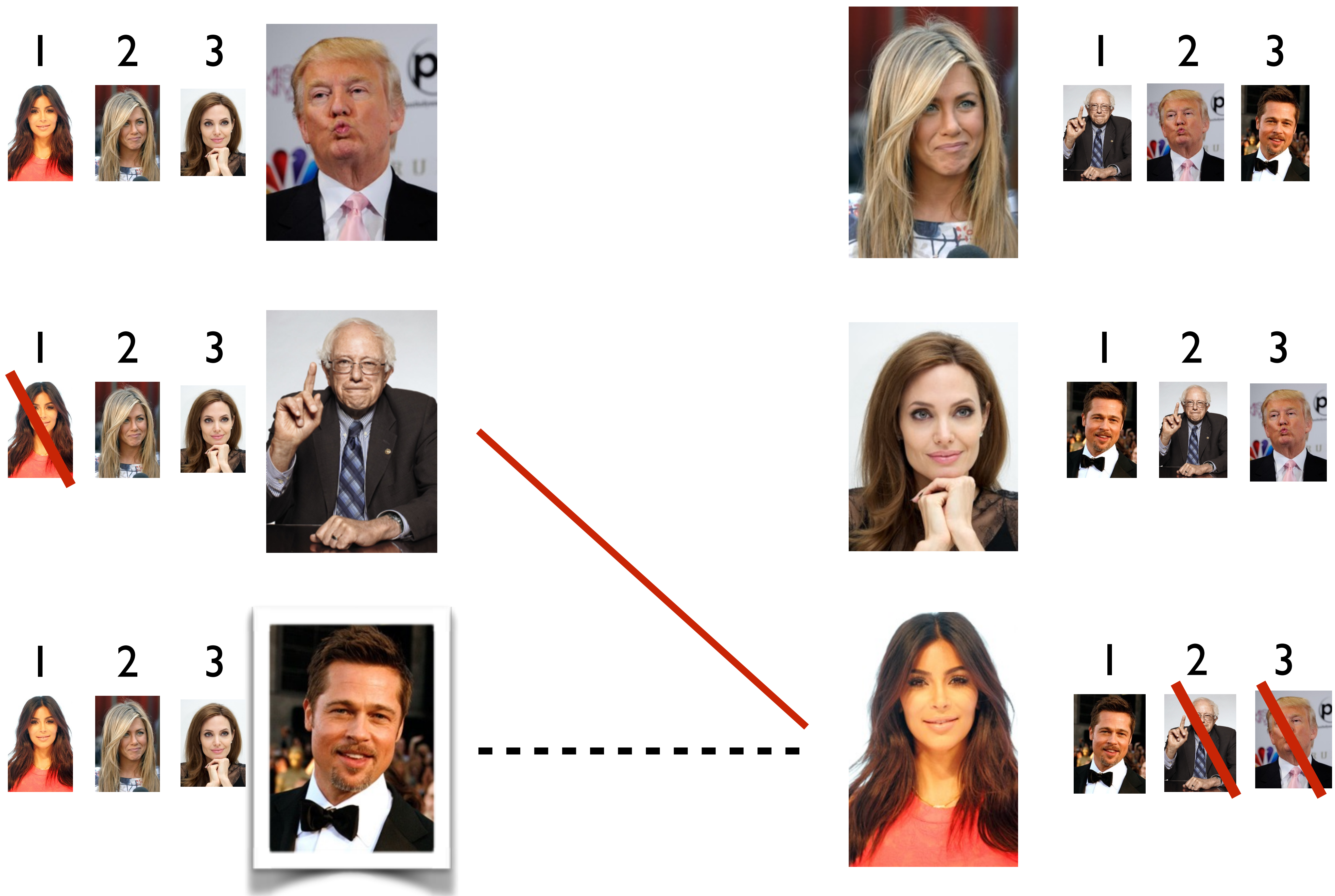


The Gale-Shapley proposal algorithm

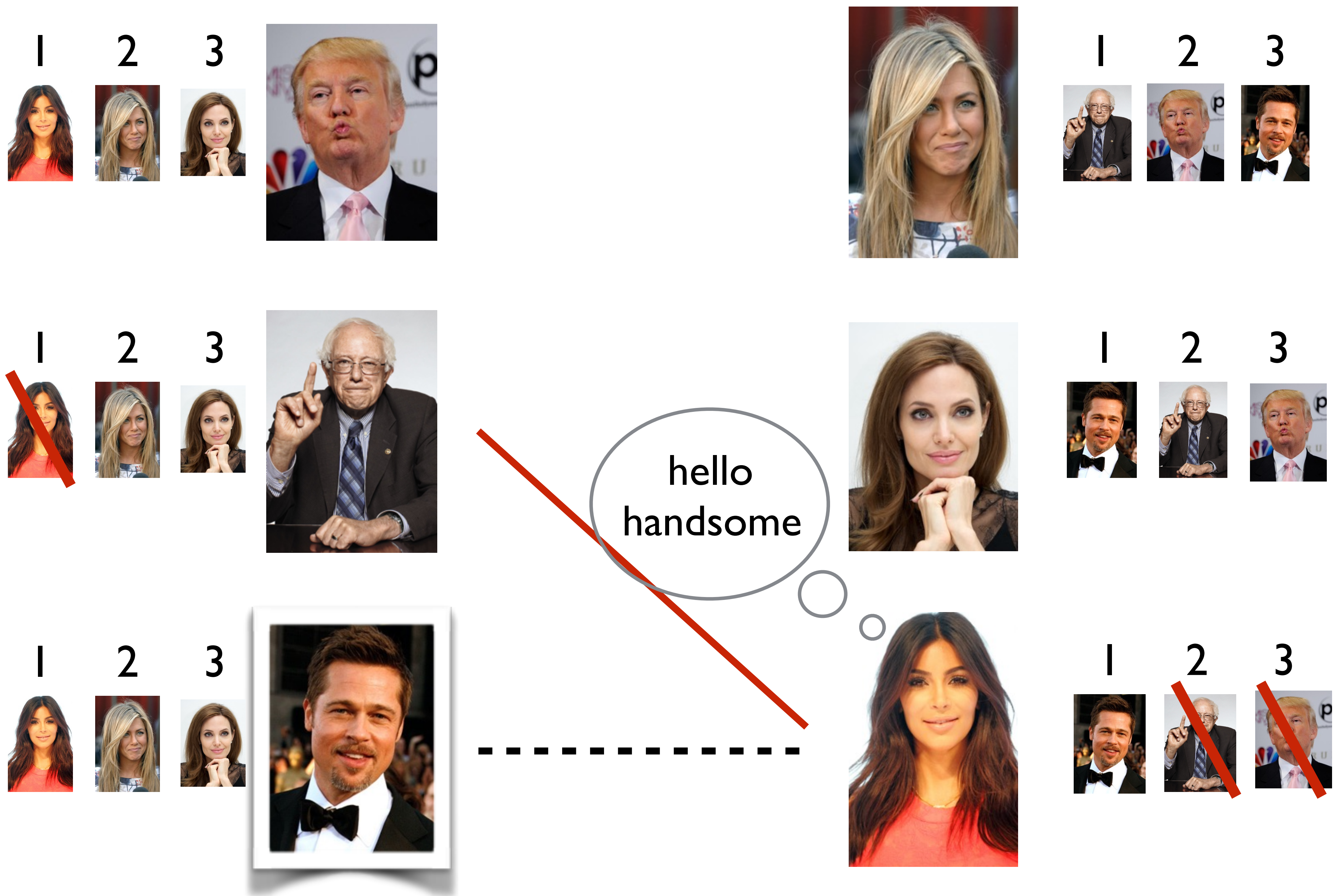
The Gale-Shapley proposal algorithm

The Gale-Shapley proposal algorithm

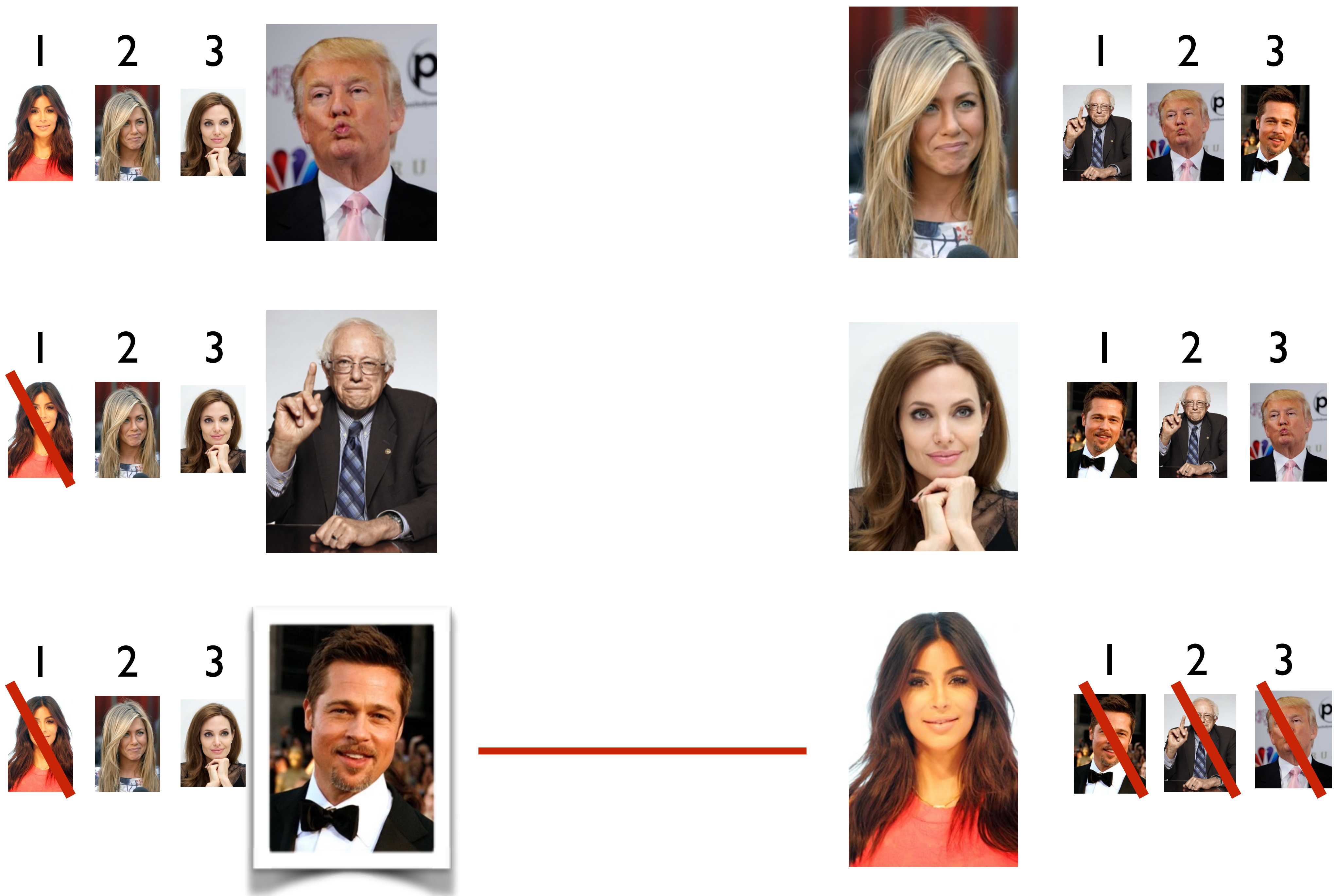
The Gale-Shapley proposal algorithm



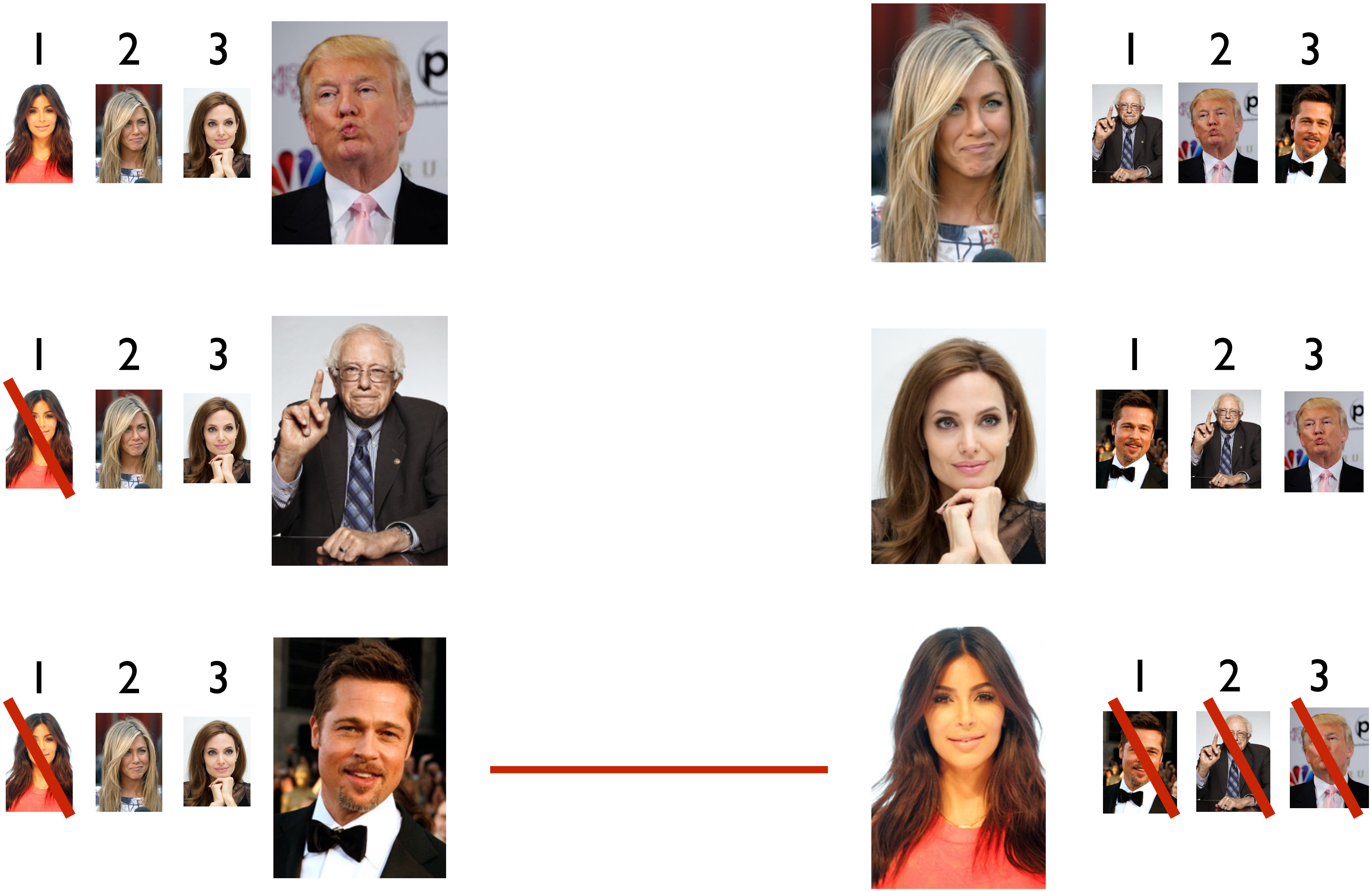
The Gale-Shapley proposal algorithm



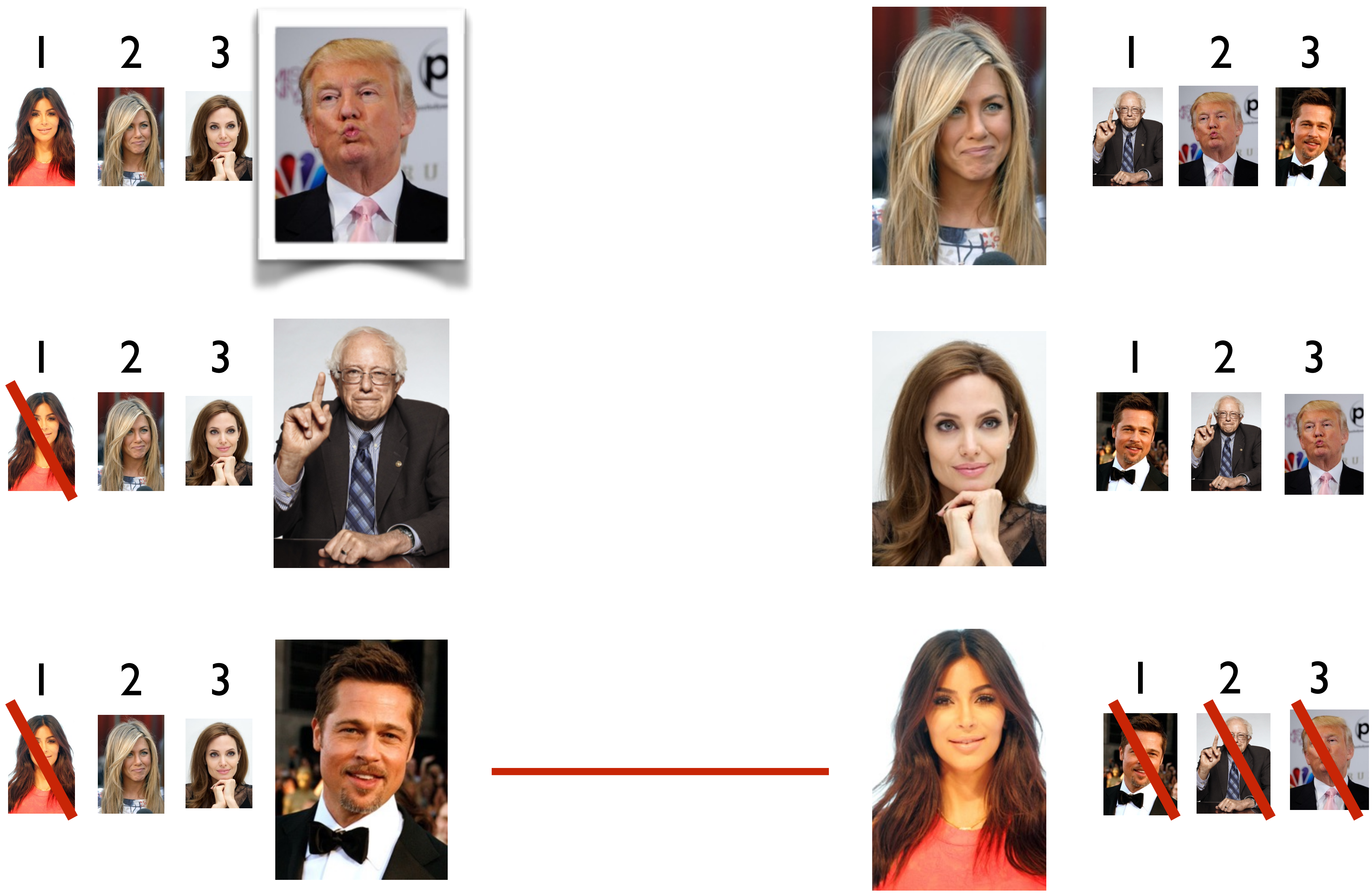
The Gale-Shapley proposal algorithm



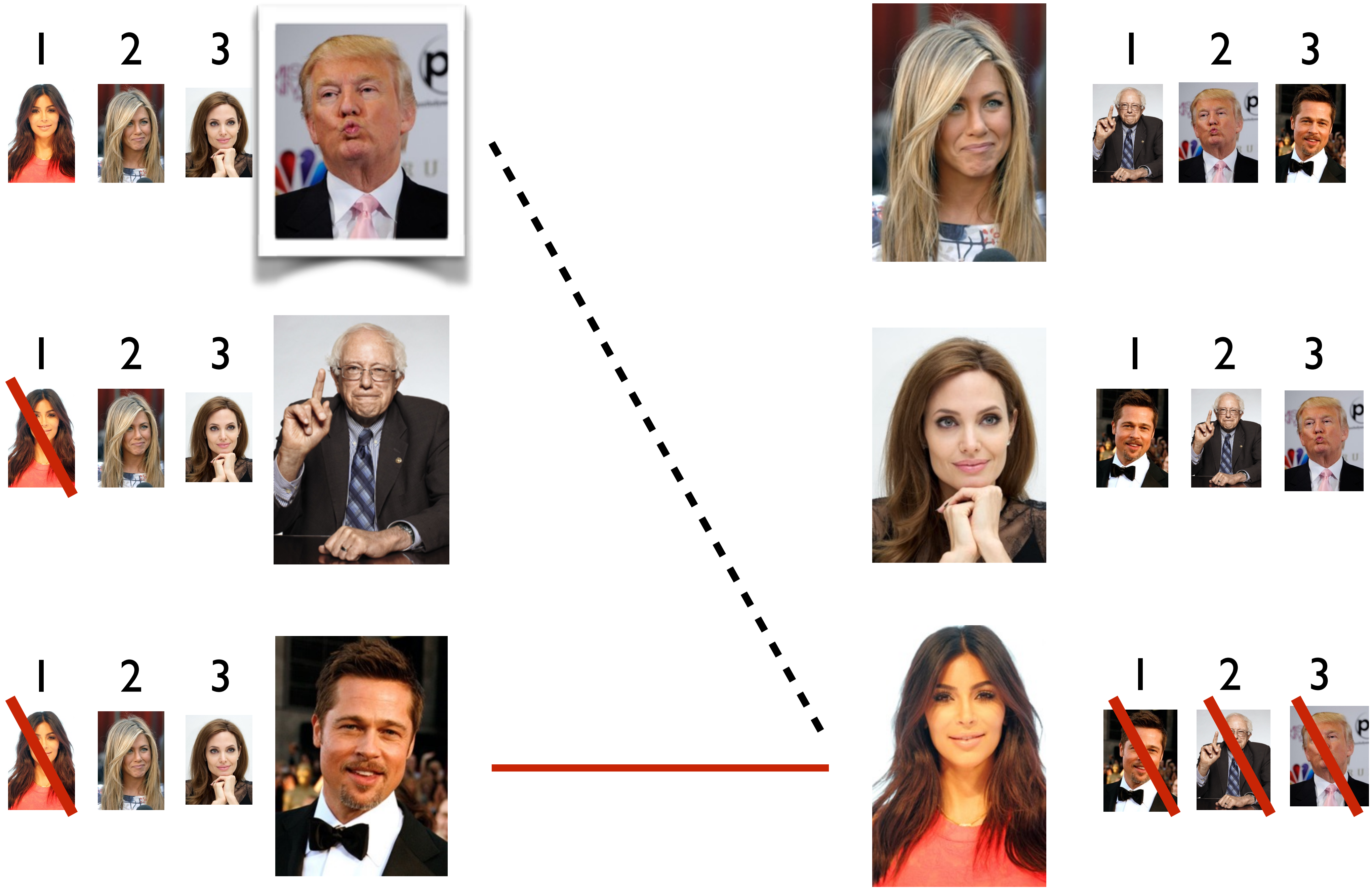
The Gale-Shapley proposal algorithm



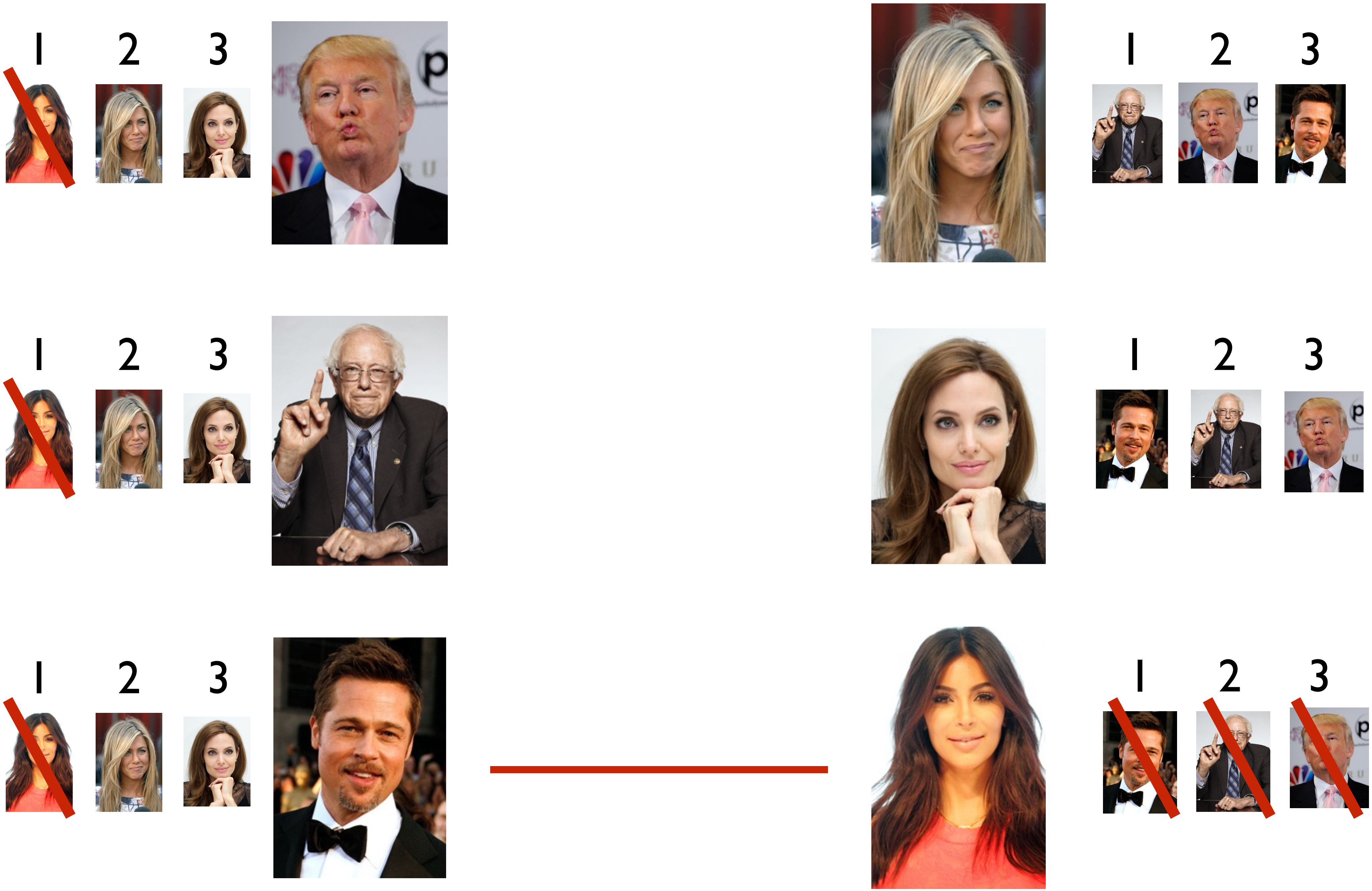
The Gale-Shapley proposal algorithm



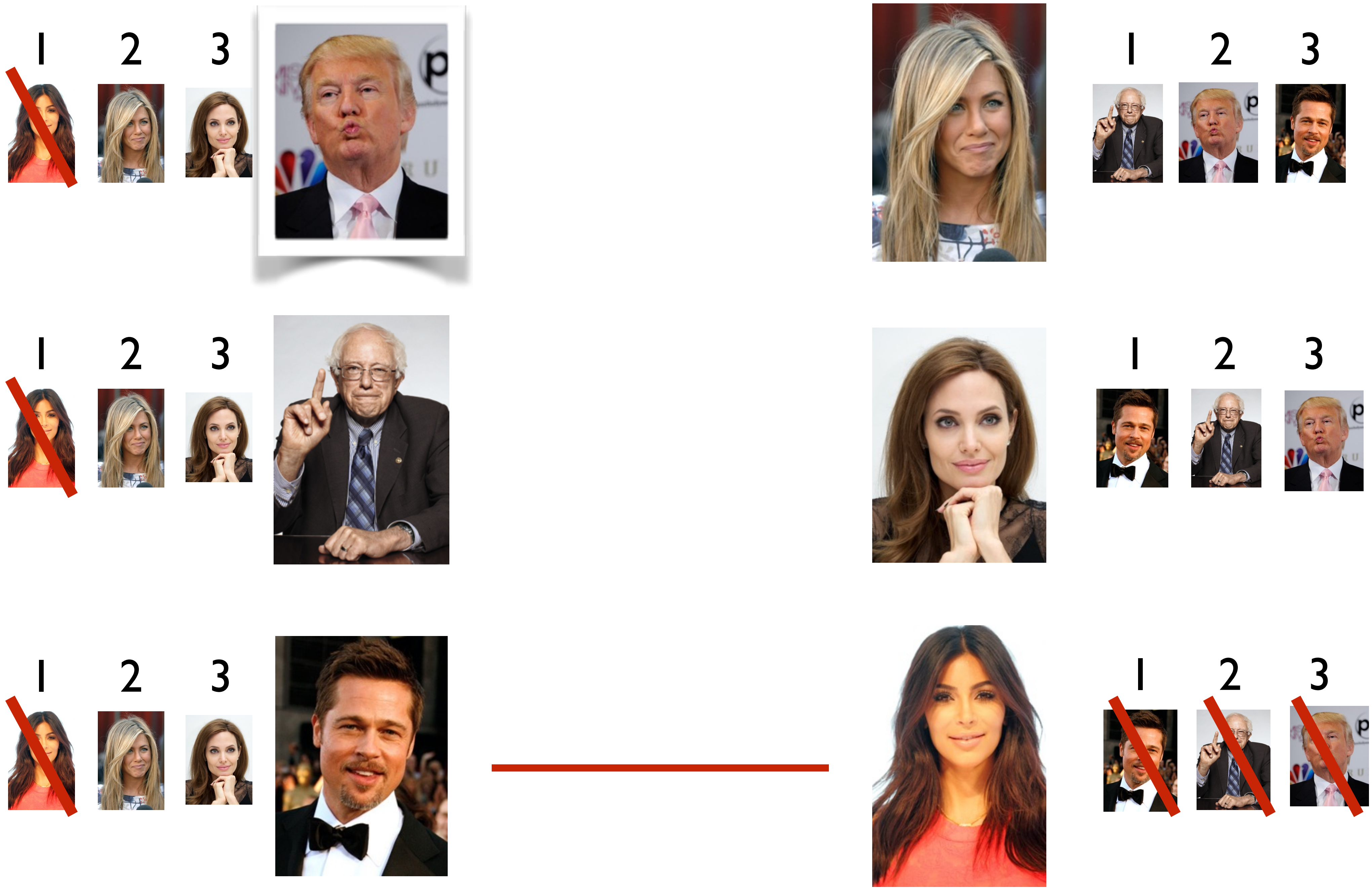
The Gale-Shapley proposal algorithm



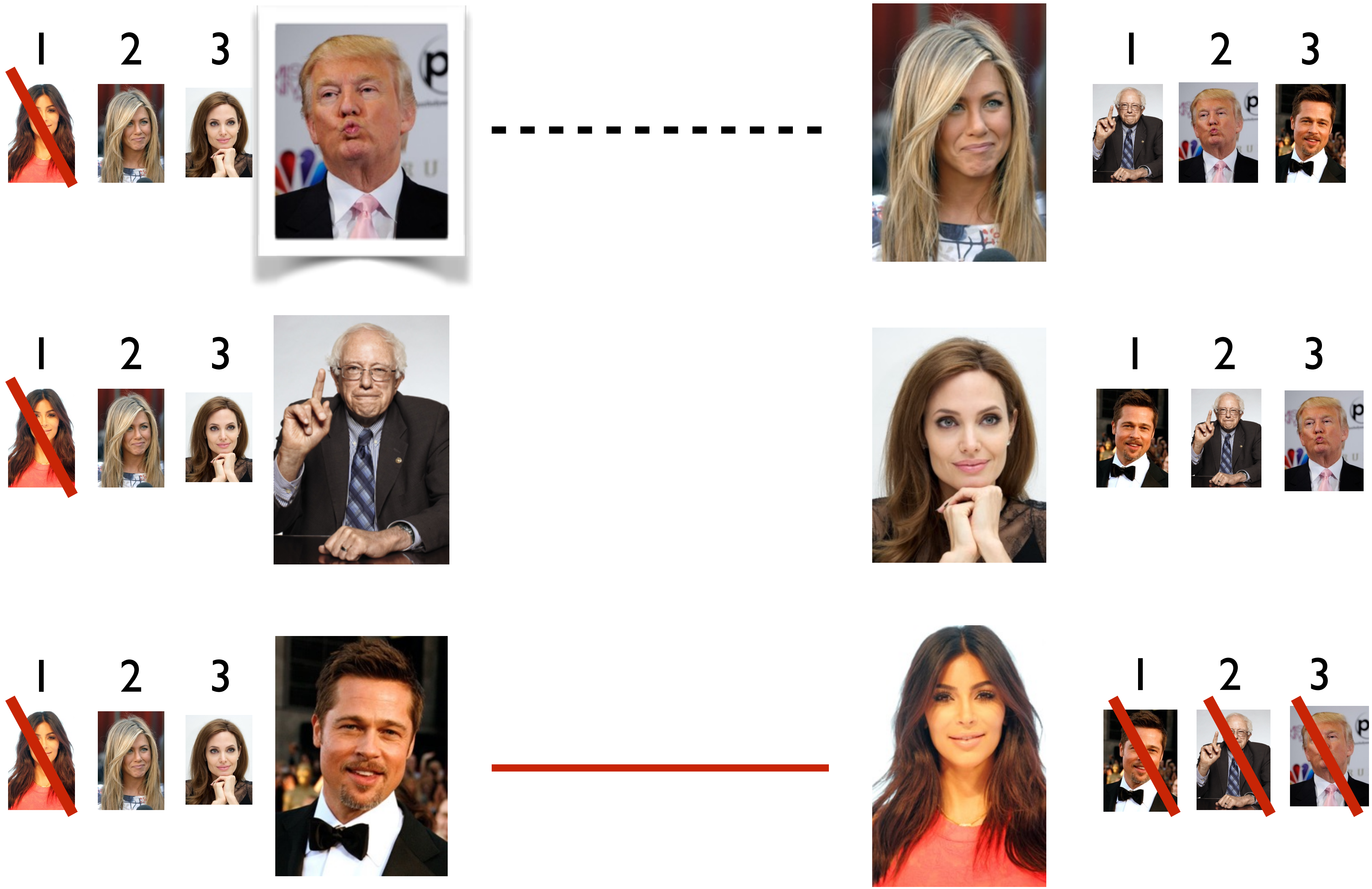
The Gale-Shapley proposal algorithm



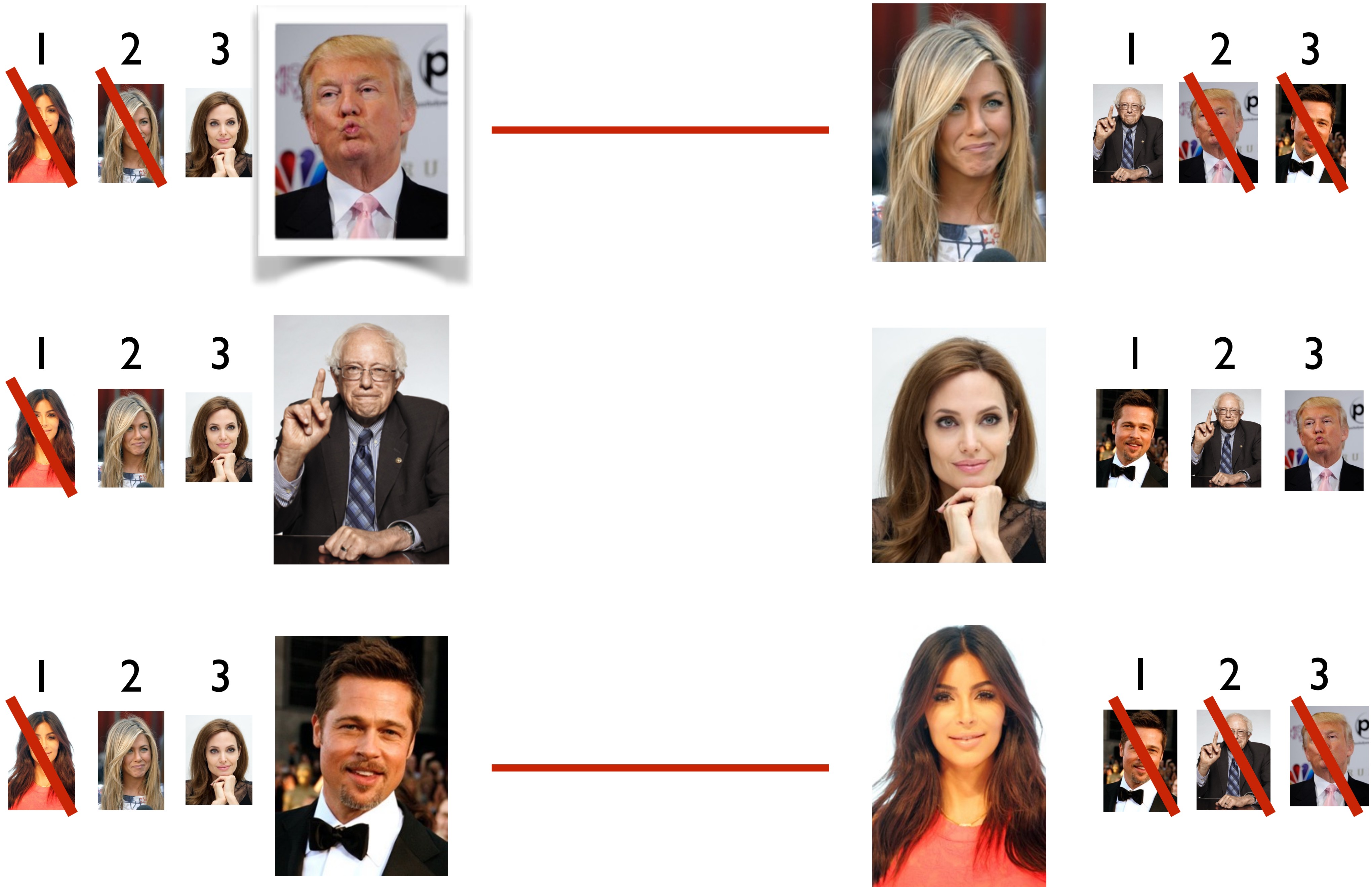
The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm

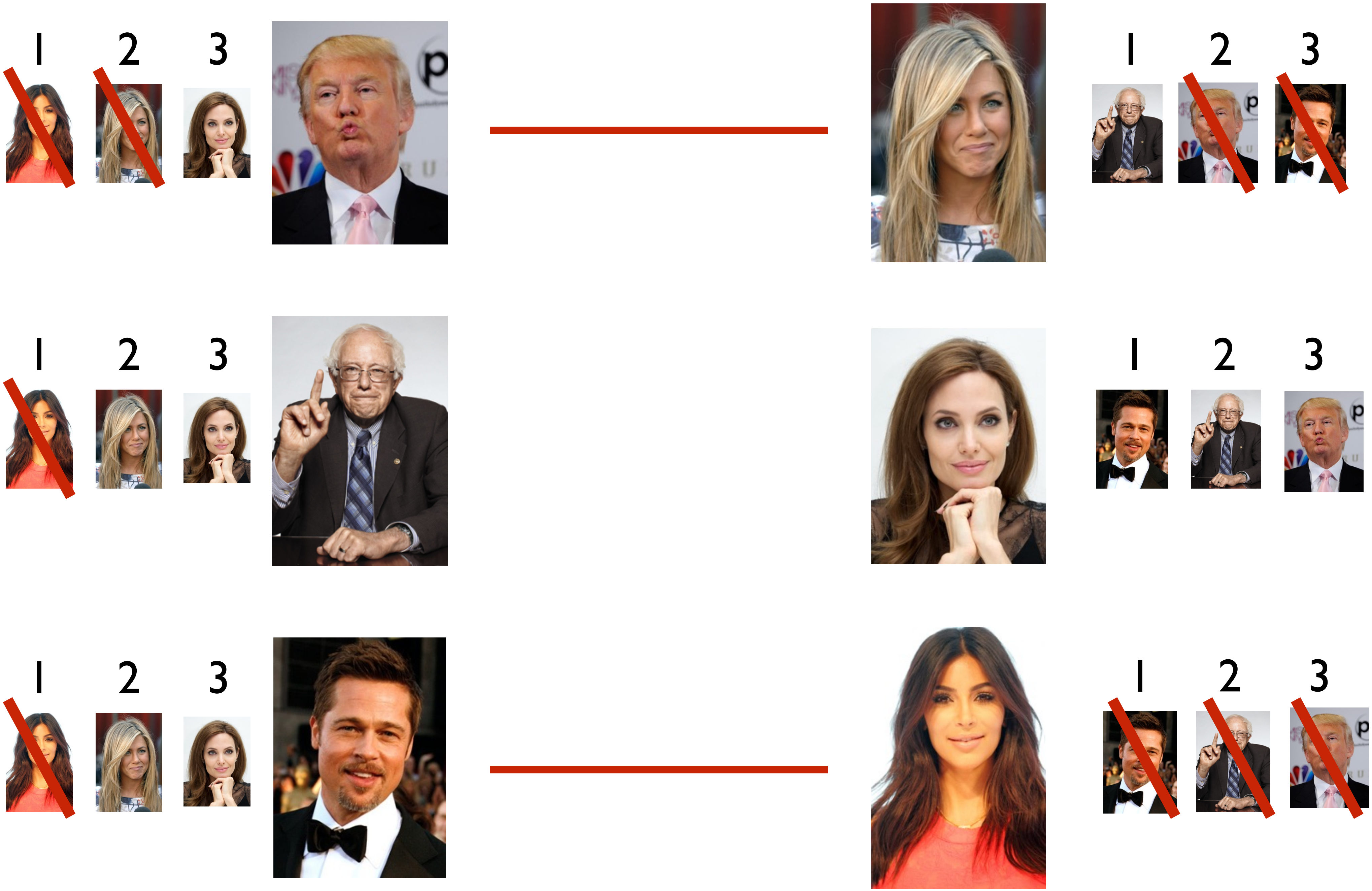


The Gale-Shapley proposal algorithm

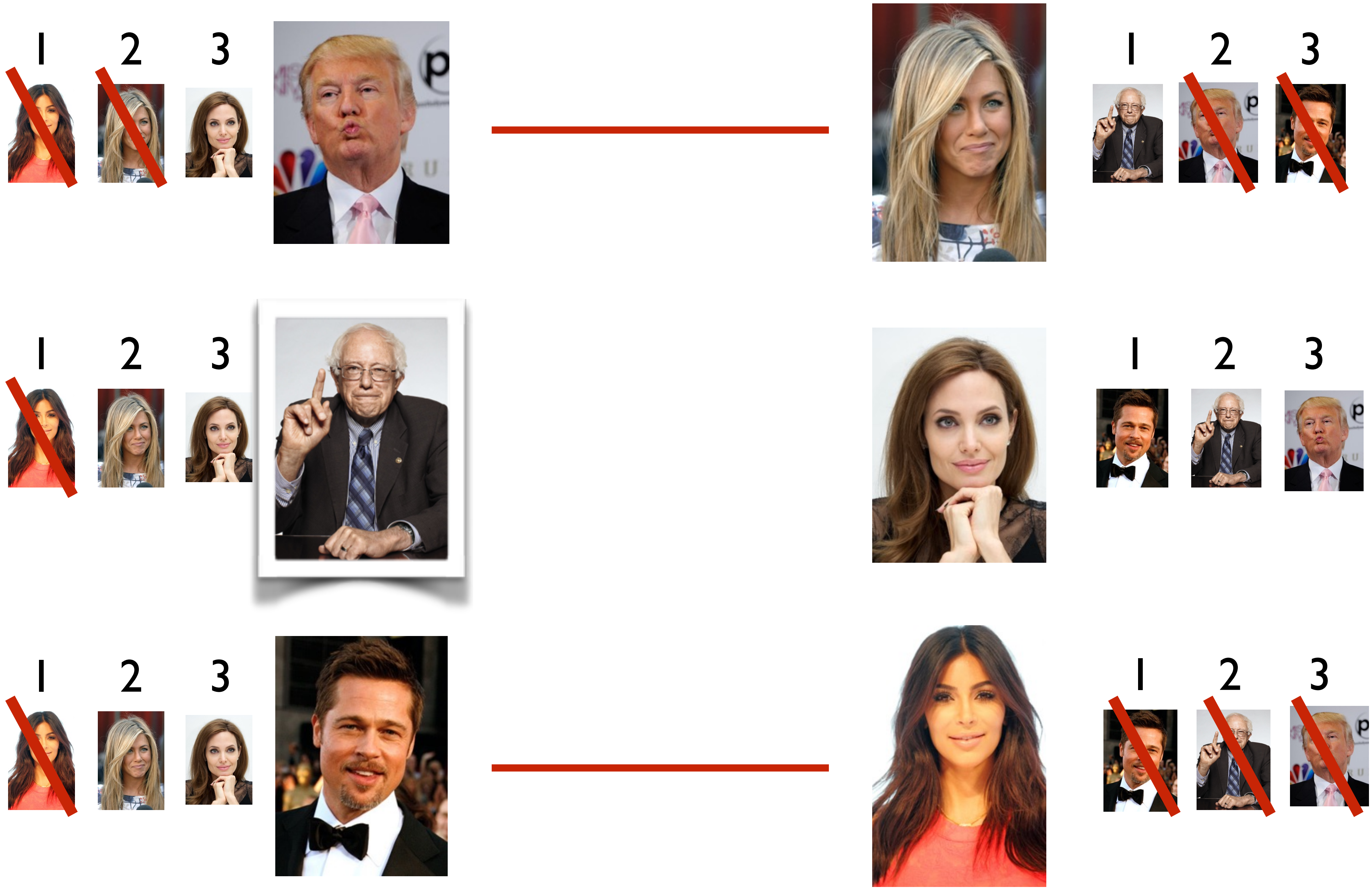


The Gale-Shapley proposal algorithm

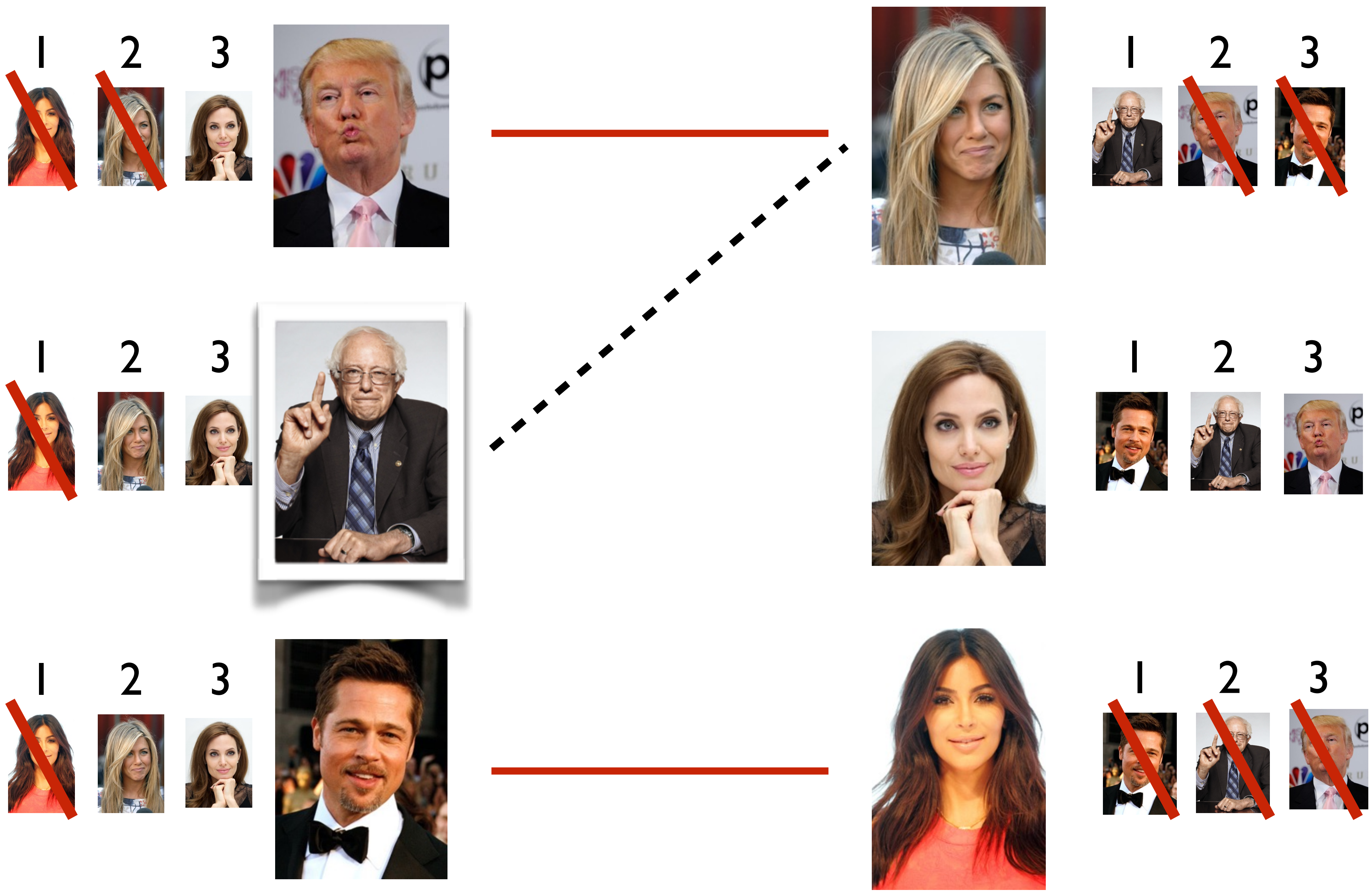
The Gale-Shapley proposal algorithm



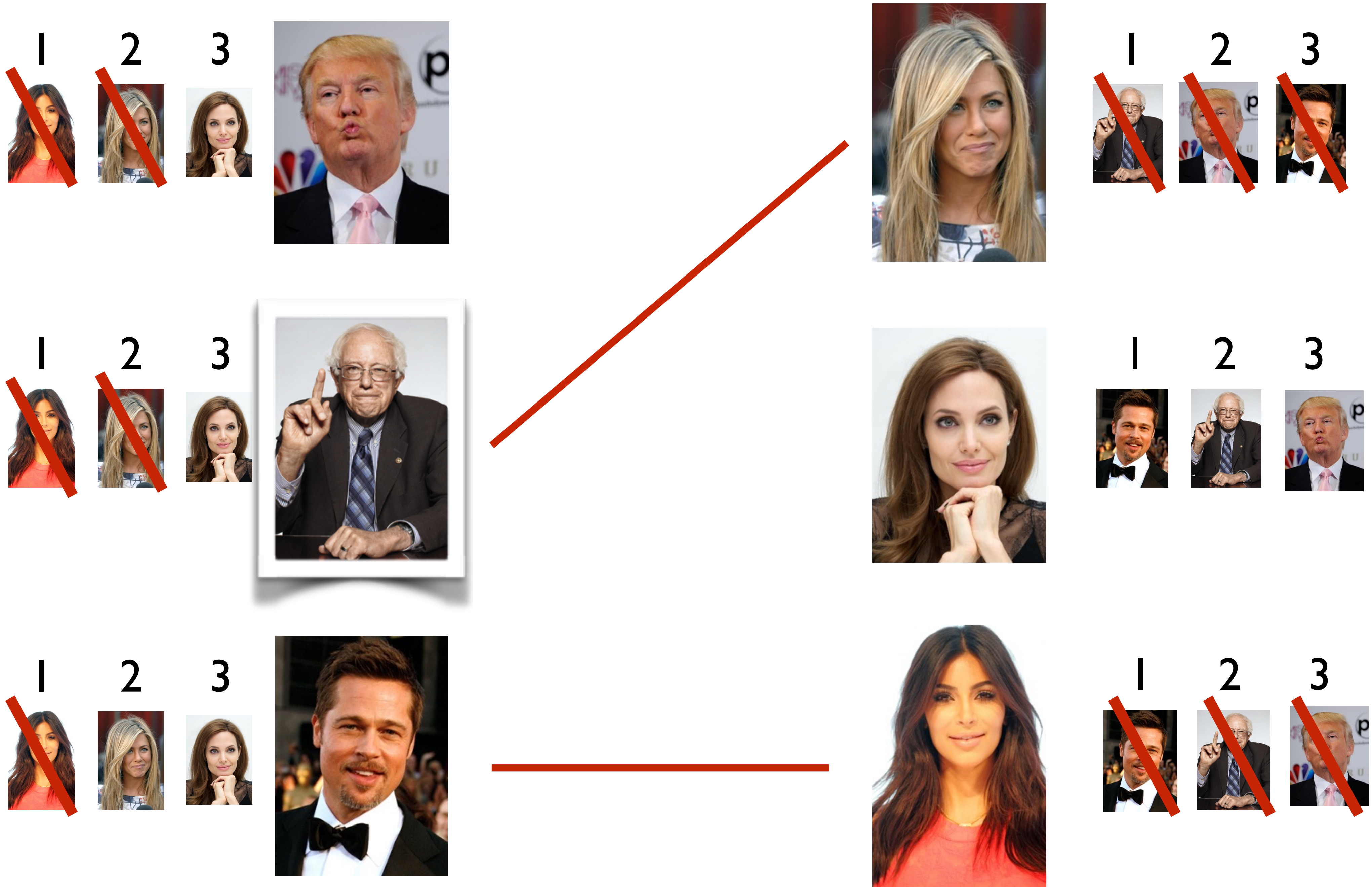
The Gale-Shapley proposal algorithm



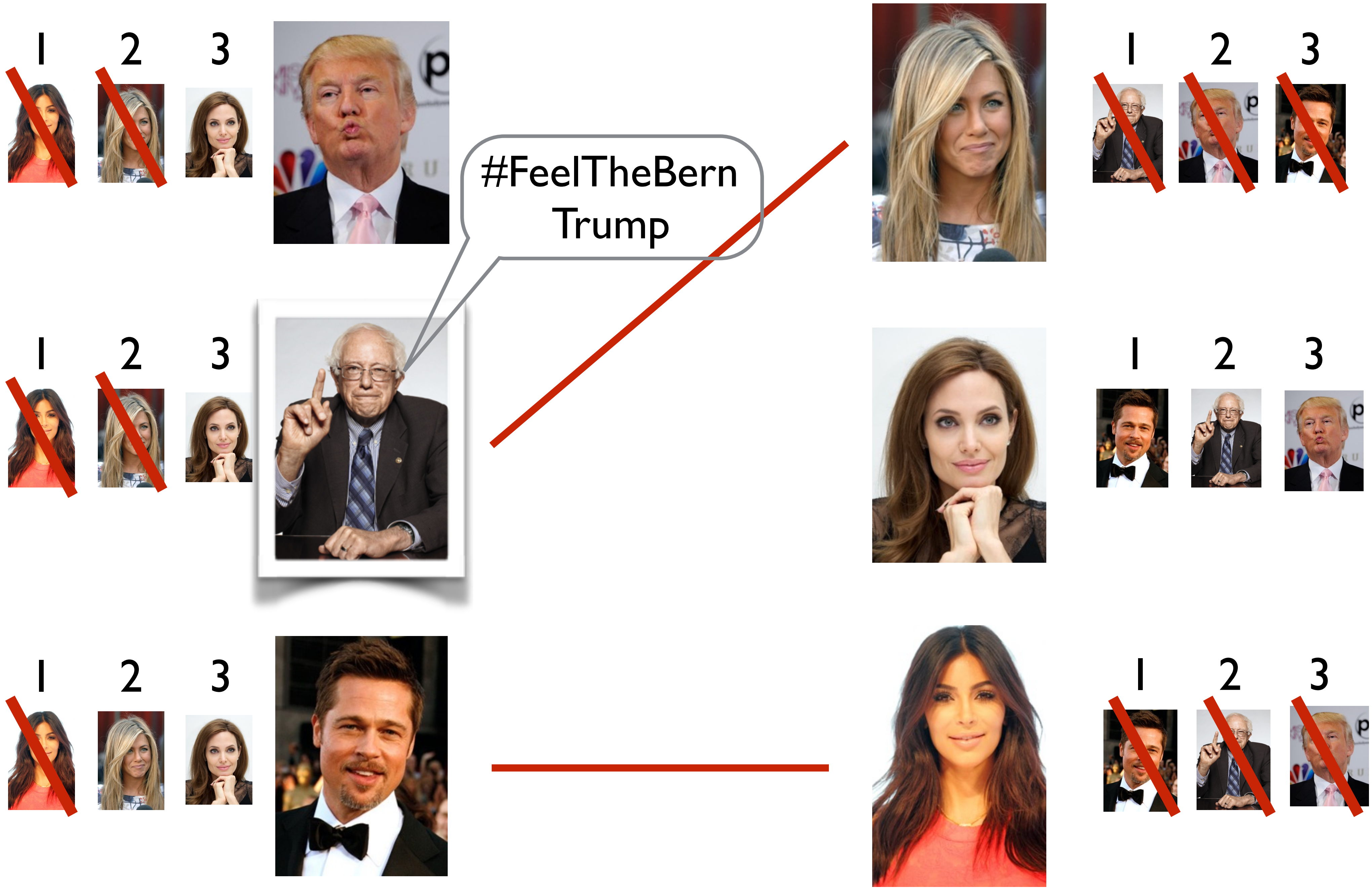
The Gale-Shapley proposal algorithm



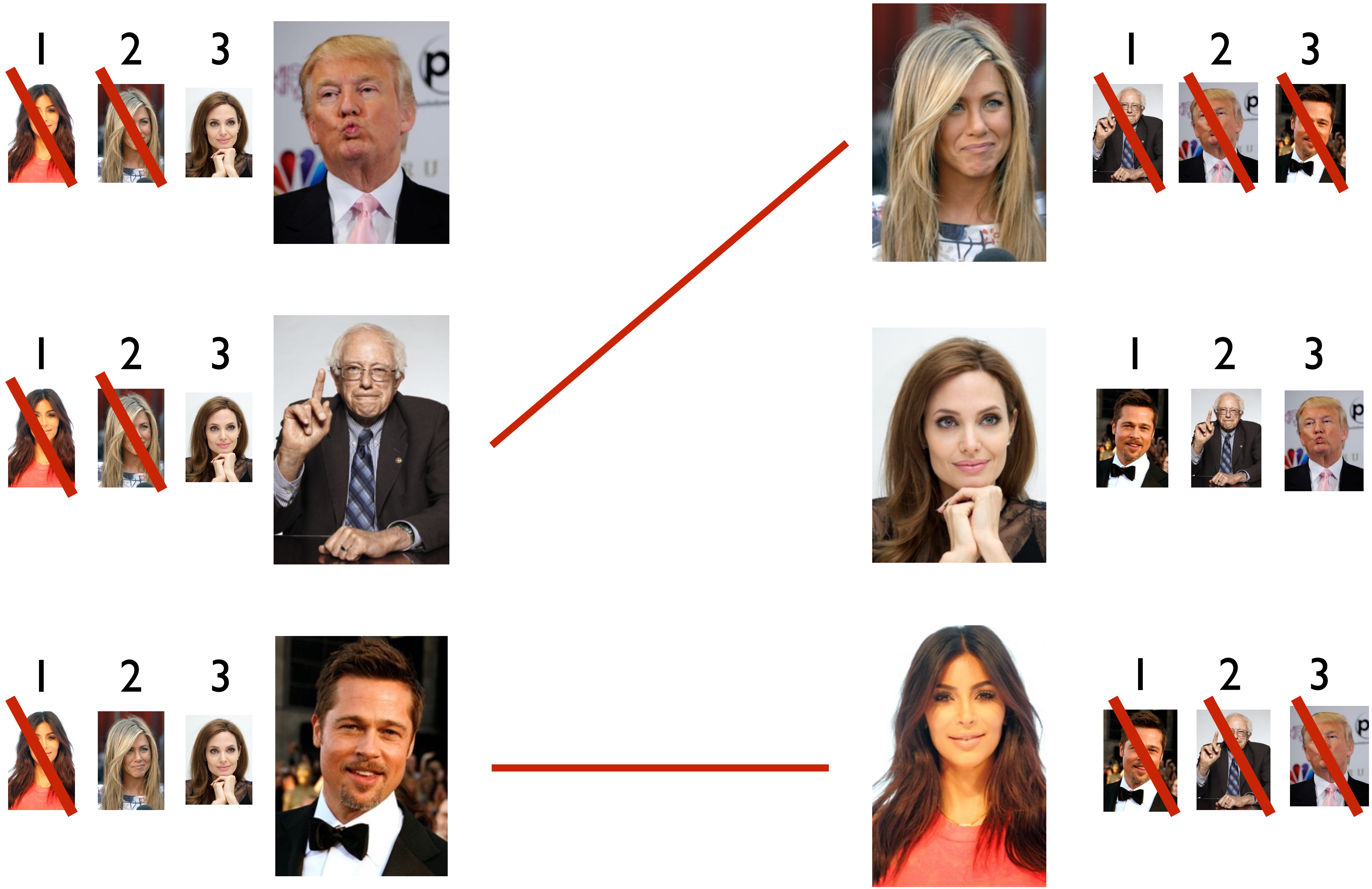
The Gale-Shapley proposal algorithm



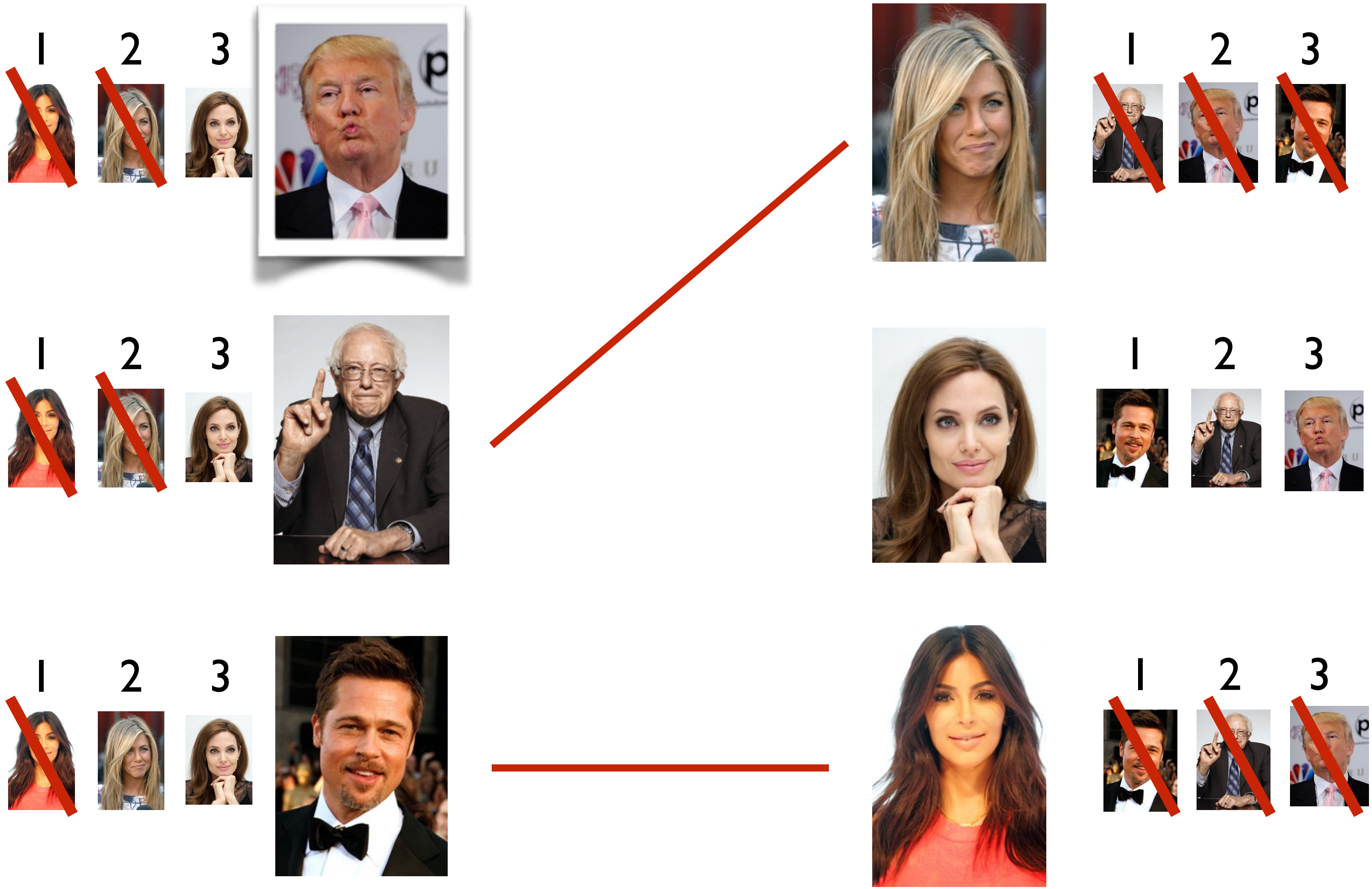
The Gale-Shapley proposal algorithm



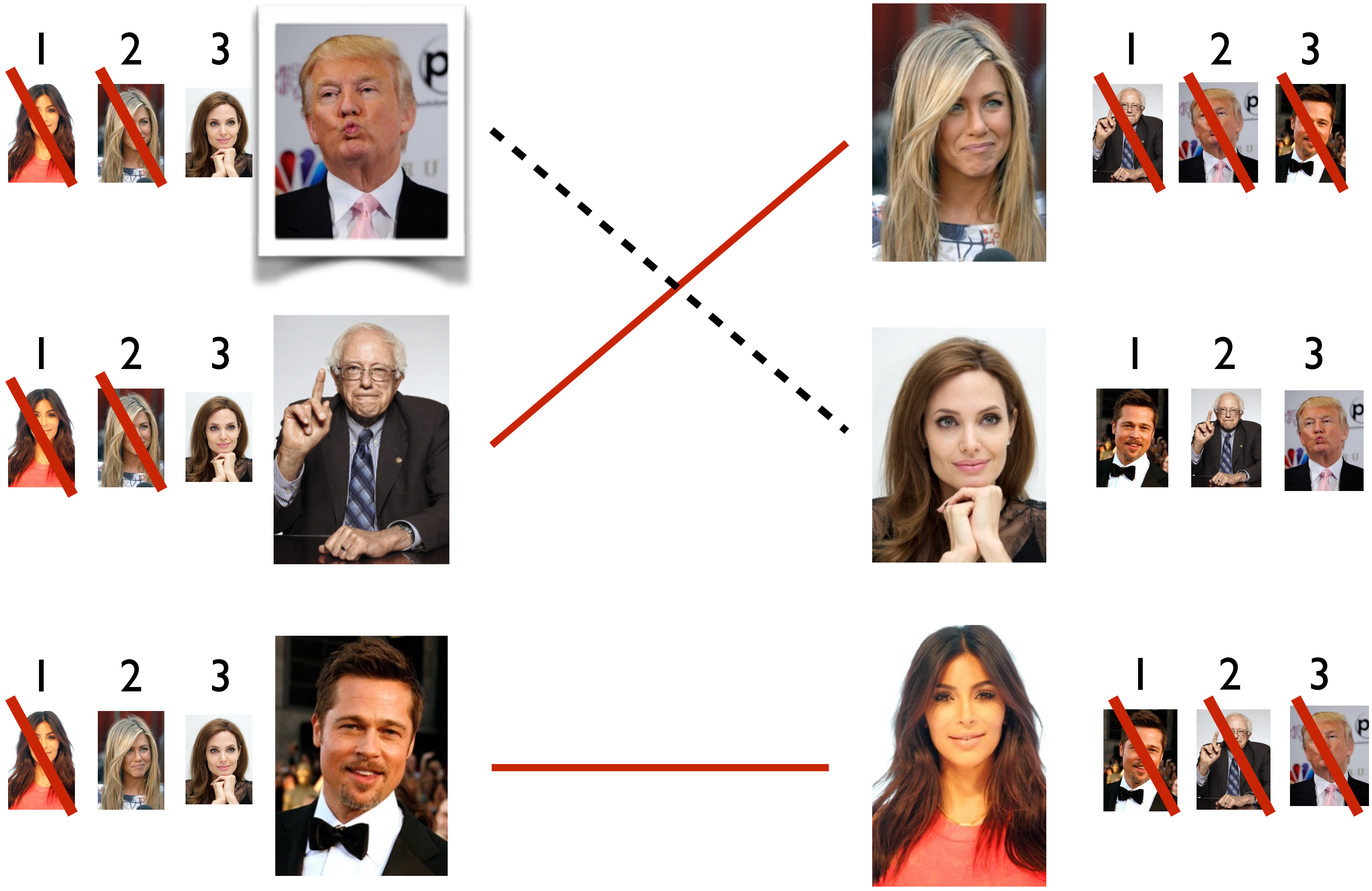
The Gale-Shapley proposal algorithm



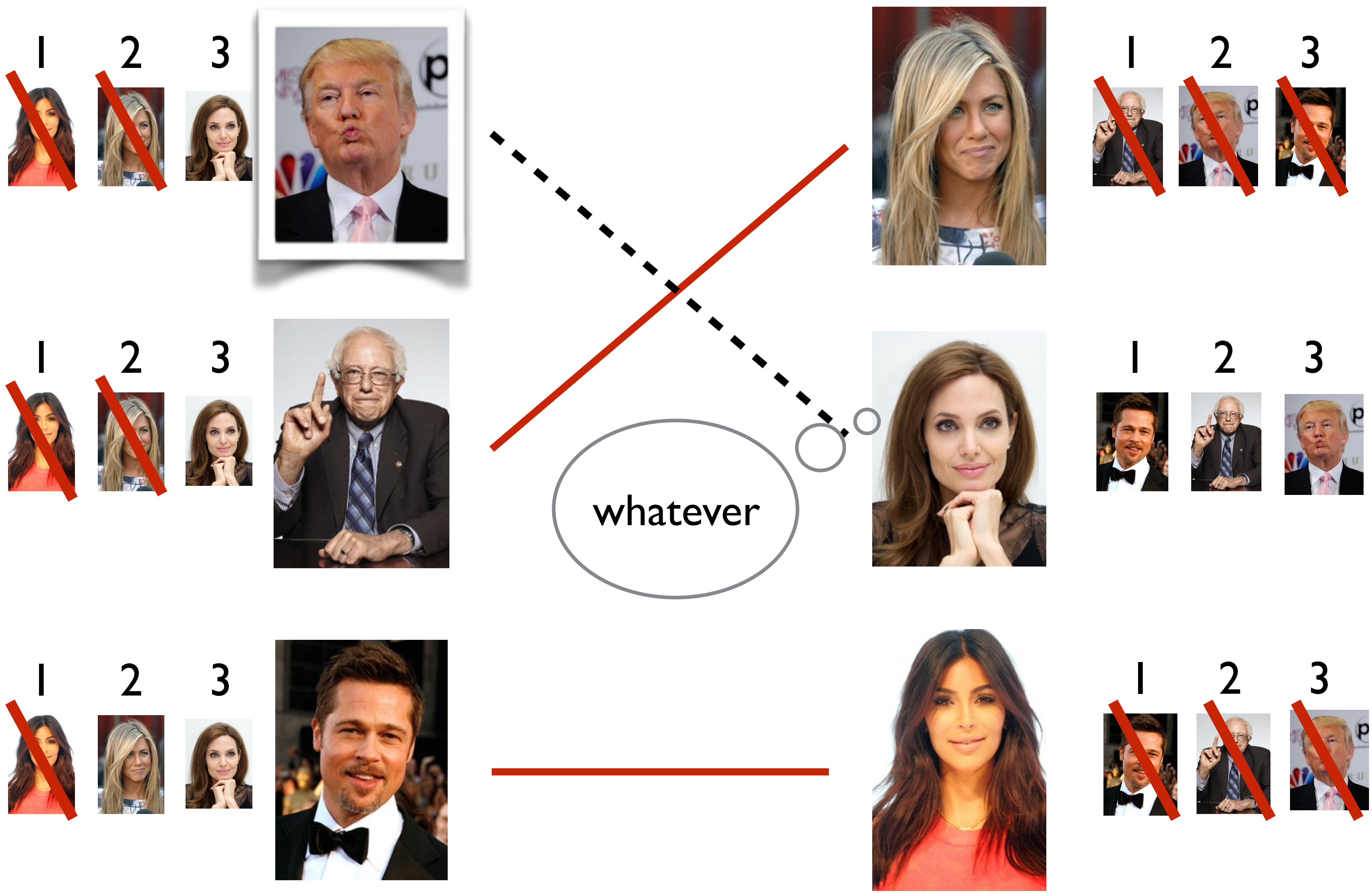
The Gale-Shapley proposal algorithm



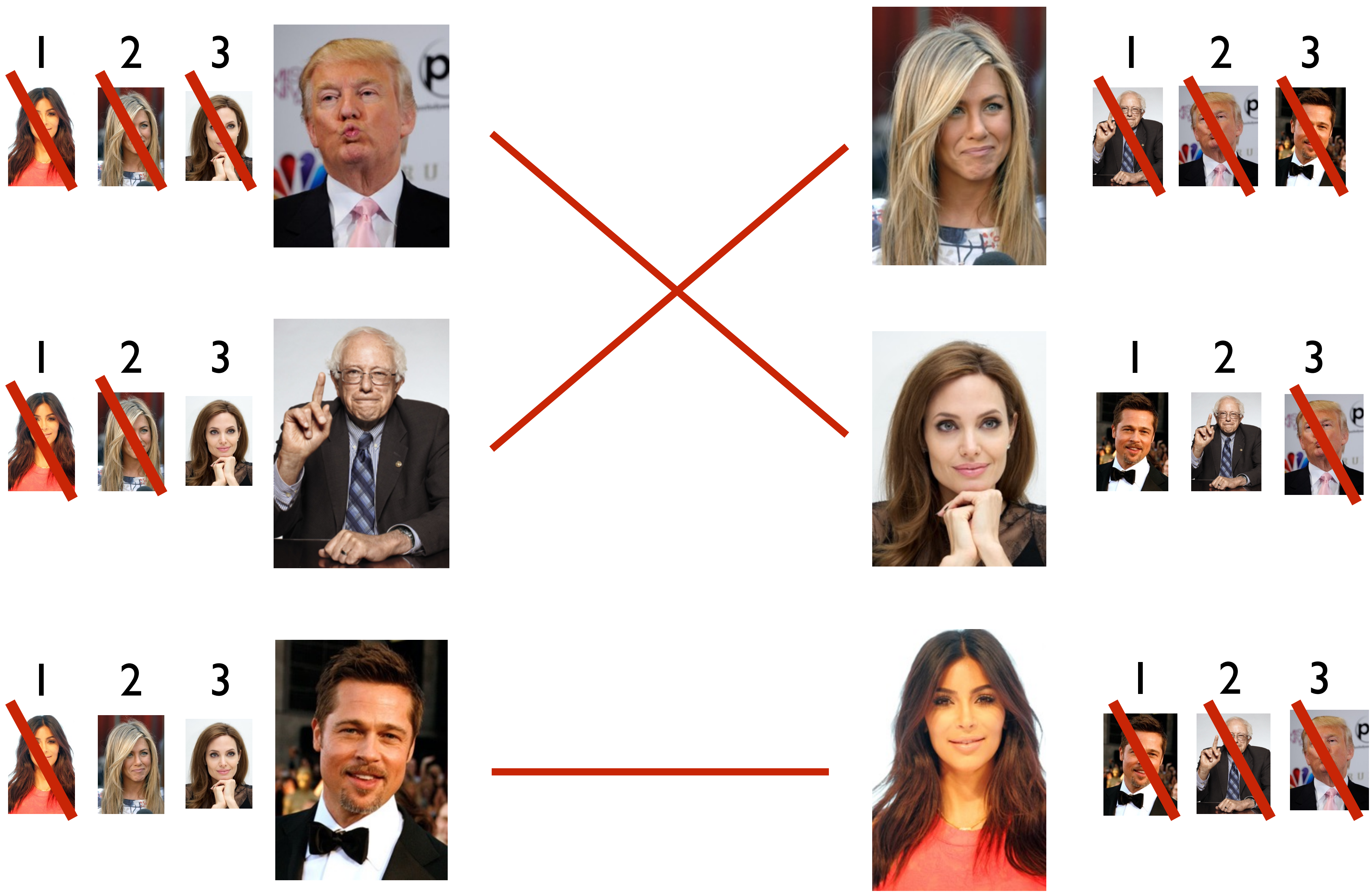
The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm

While there is a man m who is not matched:

- Let w be the highest ranked woman in m's list to whom m has not proposed yet.
- If w is unmatched, or w prefers m over her current match:
  - Match m and w.
    (The previous match of w is now unmatched.)

Cool, but does it work correctly?

- Does it always terminate?
- Does it always find a stable matching?
  (Does a stable matching always exist?)

Gale-Shapley algorithm analysis

Theorem:

The *Gale-Shapley proposal algorithm* always terminates with a stable matching after at most n^2 iterations.

A *constructive* proof that a stable matching always exists.

3 things to show:

1. Number of iterations is at most n^2.
2. The algorithm terminates with a perfect matching.
3. The matching has no unstable pairs.

Gale-Shapley algorithm analysis

1. Number of iterations is at most n^2.

iterations = # proposals

No man proposes to a woman more than once.

So each man makes at most n proposals.

There are n men in total.

$\implies$ # proposals $\leq n^2$.

$\implies$ # iterations $\leq n^2$.

Gale-Shapley algorithm analysis

2. The algorithm terminates with a perfect matching.

If we don't have a perfect matching:

A man is not matched

$\Longrightarrow$ All women must be matched

$\Longrightarrow$ All men must be matched.

Contradiction

Second implication:

There are an equal number of men and women.

Gale-Shapley algorithm analysis

2. The algorithm terminates with a perfect matching.

If we don't have a perfect matching:

A man is not matched

$\Longrightarrow$ All women must be matched

$\Longrightarrow$ All men must be matched.

Contradiction

<u>First implication:</u>

Observe: once a woman is matched, she stays matched.

A man got rejected by every woman:

case1: she was already matched, or

case2: she got a better offer

Either way, she was matched at some point.

Gale-Shapley algorithm analysis

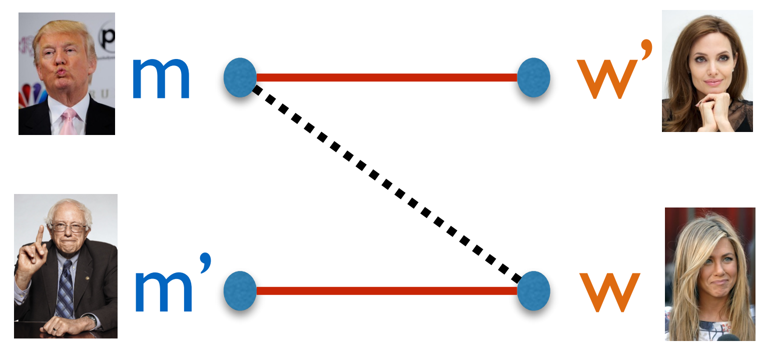
3. The matching has <u>no</u> unstable pairs.

> "<u>Improvement" Lemma</u>:
> **(i)** A man can only go down in his preference list.
> **(ii)** A woman can only go up in her preference list.

Unstable pair:

(m,w) <u>unmatched</u>

but they prefer each other.



Consider any <u>unmatched</u> (m,w). **WTS**: it cannot be unstable.

Case 1: m never proposed to w

by (i), m prefers w' over w

Case 2: m proposed to w

w rejected m $\Longrightarrow$ by (ii), w prefers m' over m $\square$

Further questions

Theorem:

The *Gale-Shapley proposal algorithm* always terminates with a stable matching after at most n^2 iterations.

Does the order of how we pick men matter?

Would it lead to different matchings?

Is the algorithm "fair"?

Does this algorithm favor men or women or neither?

Further questions

m and w are *valid partners* if there is a stable matching in which they are matched.

best(m) = highest ranked valid partner of m

Theorem:

Gale-Shapley algorithm returns $\{(m, \text{best}(m)) : m \in X\}$.

Not at all obvious this would be a matching, let alone a stable matching!

Further questions

worst(w) = lowest ranked valid partner of w

Theorem:

Gale-Shapley algorithm returns $\{(\text{worst}(w), w) : w \in Y\}$.

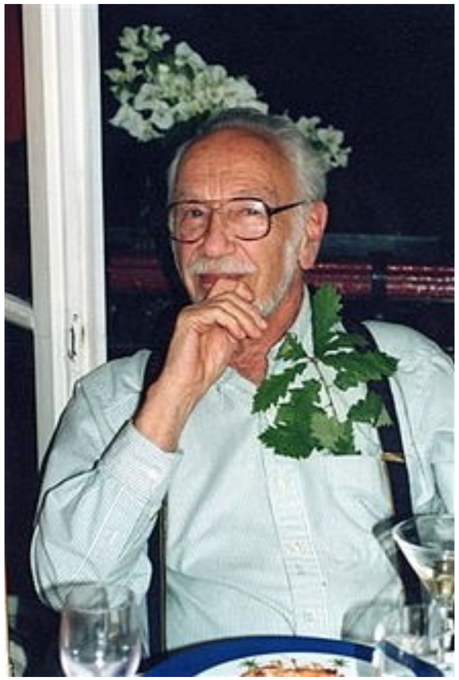
Real-world applications

Variants of the Gale-Shapley algorithm
is used for:

- matching medical students and hospitals
- matching students to high schools (e.g. in New York)
- matching students to universities (e.g. in Hungary)
- matching users to servers

⋮

The Gale-Shapley Proposal Algorithm (1962)



Nobel Prize in Economics 2012

"for the theory of stable allocations and the practice of market design."