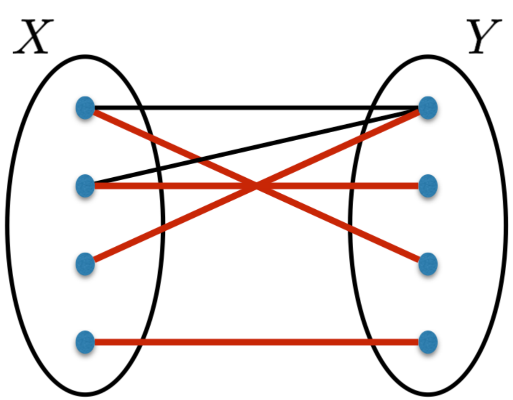
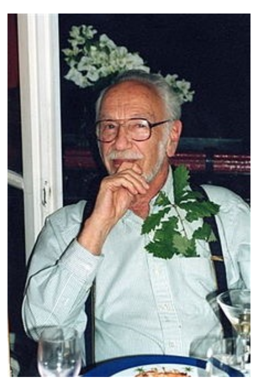
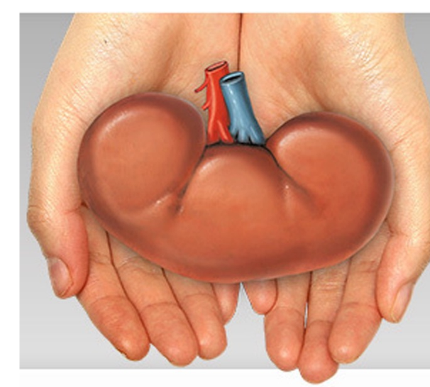
15-251
Great Theoretical Ideas in Computer Science

Lecture 11:
Graphs III: Maximum and Stable Matchings







October 8th, 2015

Today's Goal:

Save lives.

Today's Goal:

Maximum matching problem
(in bipartite graphs)

Stable matching problem

Maximum matching problem
(in bipartite graphs)

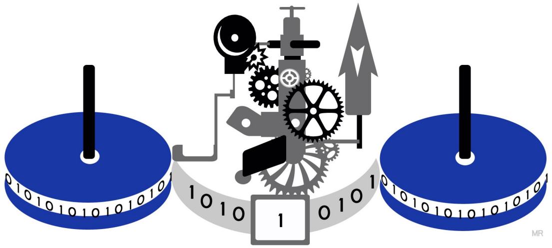
Some motivating real-world examples

matching machines and jobs

Job 1

Job 2

⋮ ⋮



Job n

Some motivating real-world examples

matching professors and courses

⋮

15-110

15-112

15-122

15-150

15-251

⋮

Some motivating real-world examples

matching students and internships

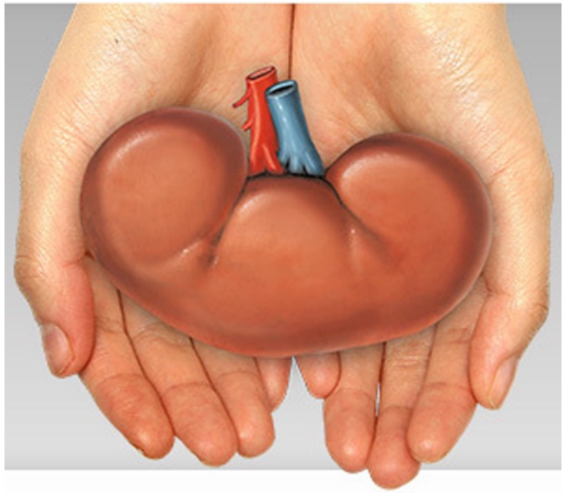
Macrosoft

Moogle
Google

UMBRELLA
CORPORATION
"Our business is life itself..."

Some motivating real-world examples

matching kidney donors and patients



How do you solve a problem like this?

1. Formulate the problem

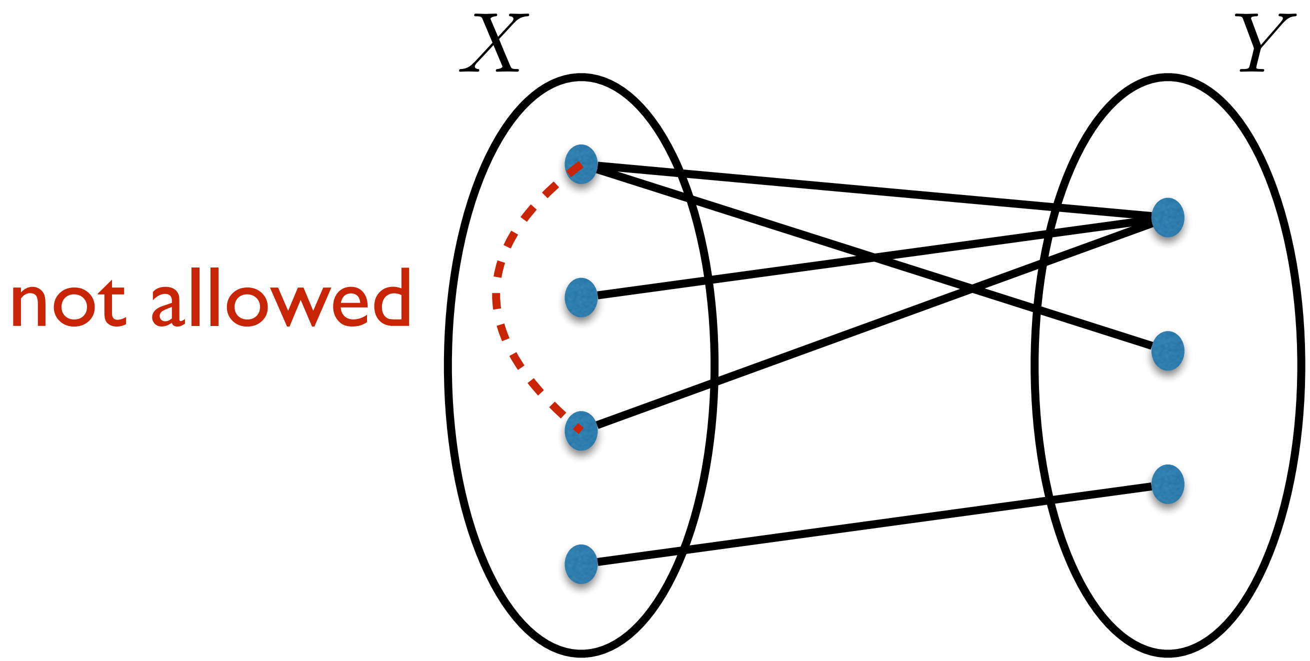
2. **Ask**: Is there a trivial algorithm?

3. **Ask**: Is there a better algorithm?

4. Find and analyze

If your problem has a graph, great. If not, try to make it have a graph!

Bipartite Graphs



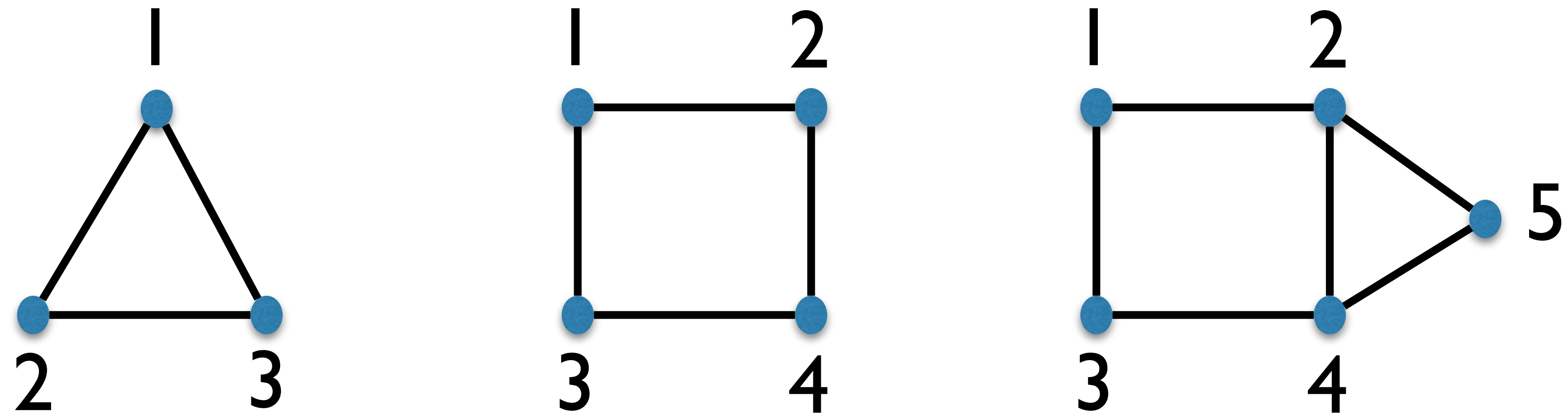
$G = (V, E)$ is **bipartite** if:

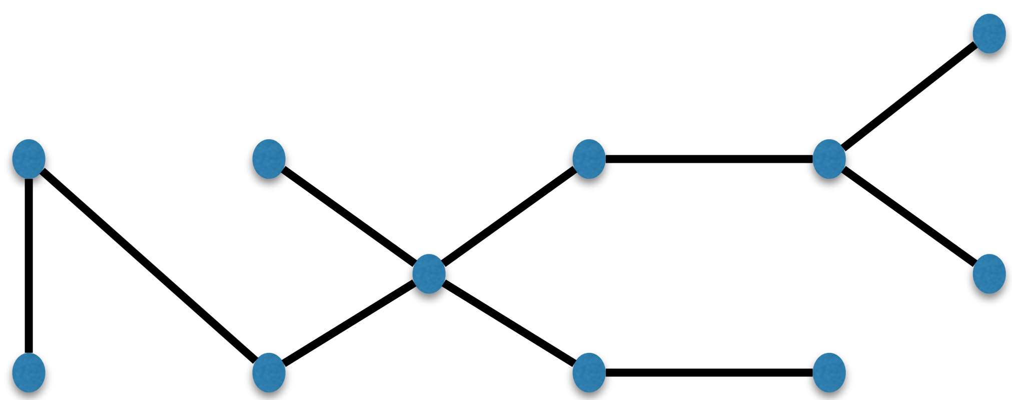
- there exists a bipartition X and Y of V
- each edge connects a vertex in X to a vertex in Y

Given a graph $G = (V, E)$, we could ask, is it bipartite?

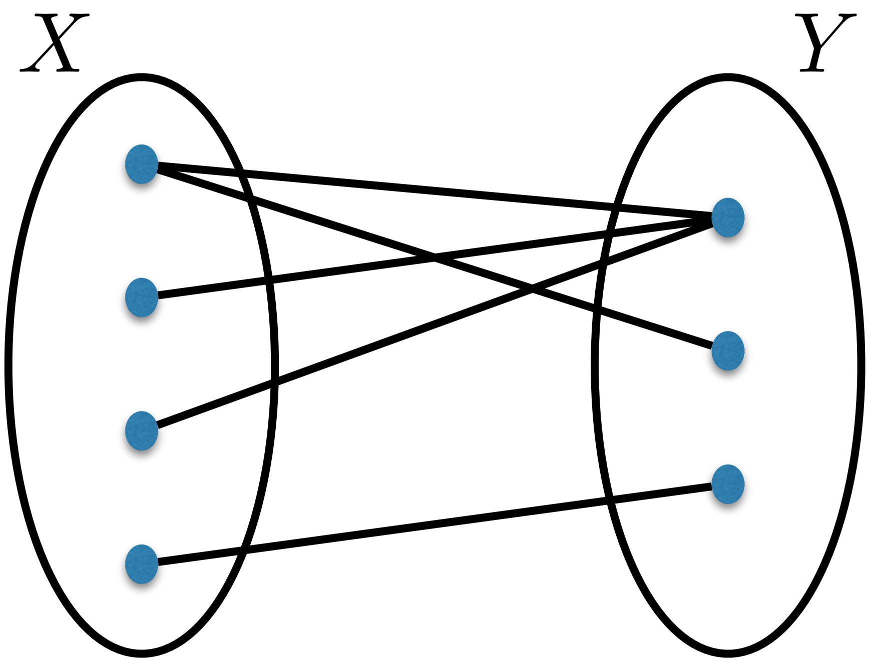
Bipartite Graphs

Given a graph $G = (V, E)$, we could ask, is it bipartite?





Bipartite Graphs



Sometimes we write the bipartition explicitly:

$$G = (X, Y, E)$$

Bipartite Graphs

Great for modeling relations between two classes of objects.

Examples:

X = machines, Y = jobs

An edge $\{x, y\}$ means x is capable of doing y.

X = professors, Y = courses

An edge $\{x, y\}$ means x can teach y.

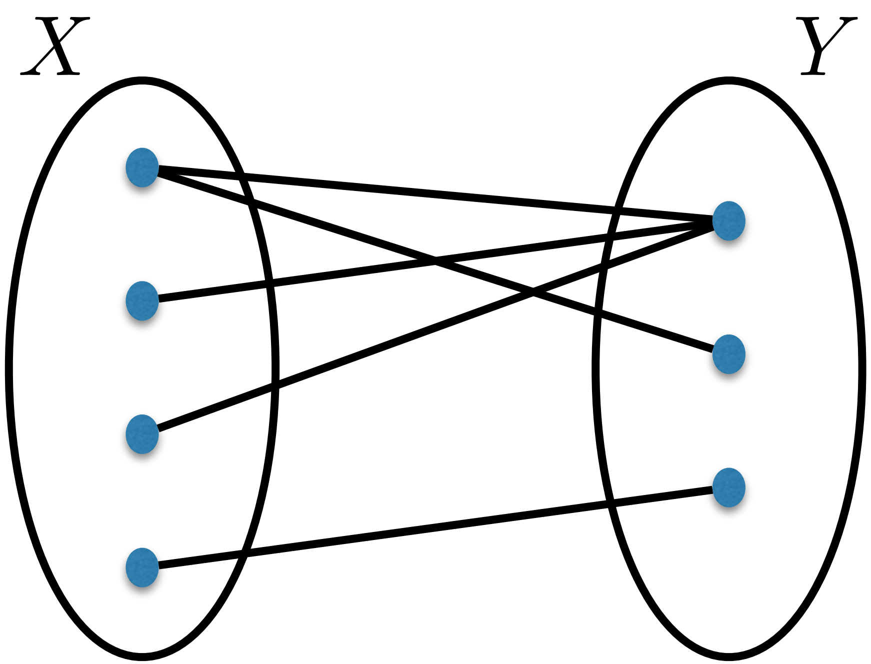
X = students, Y = internship jobs

An edge $\{x, y\}$ means x and y are interested in each other.

...

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

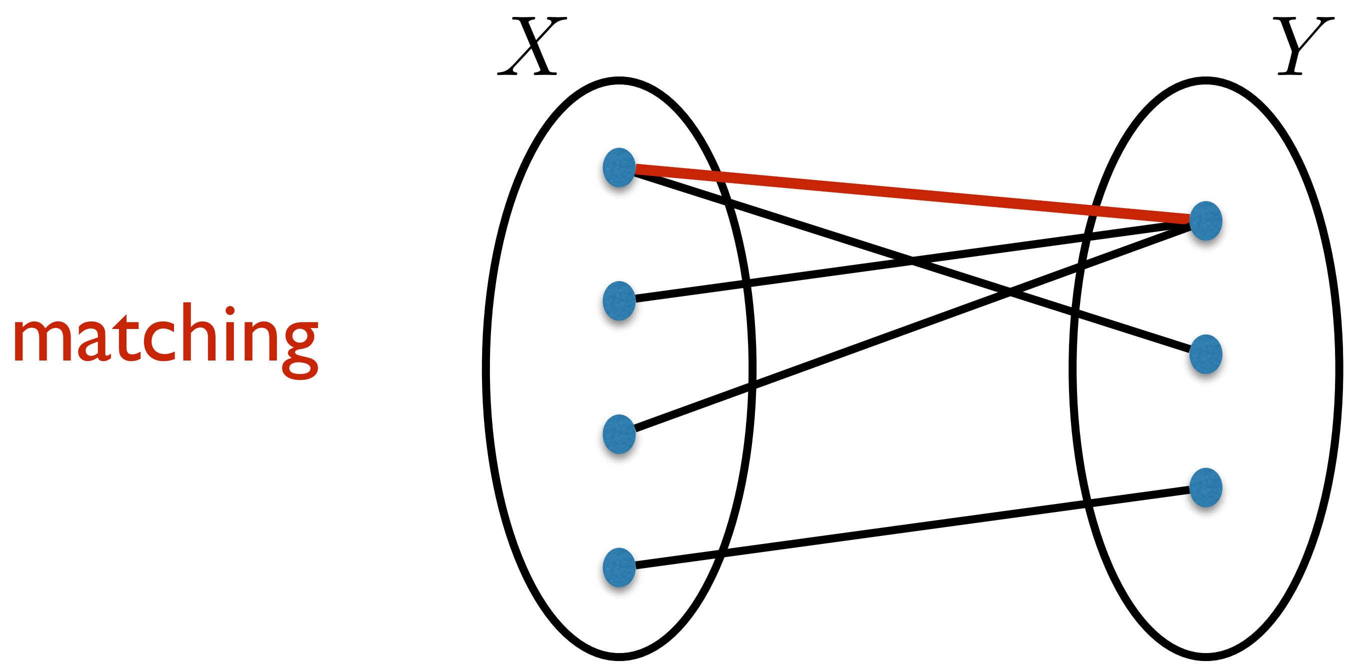


A **matching** :

A subset of the edges that do not share an endpoint.

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph



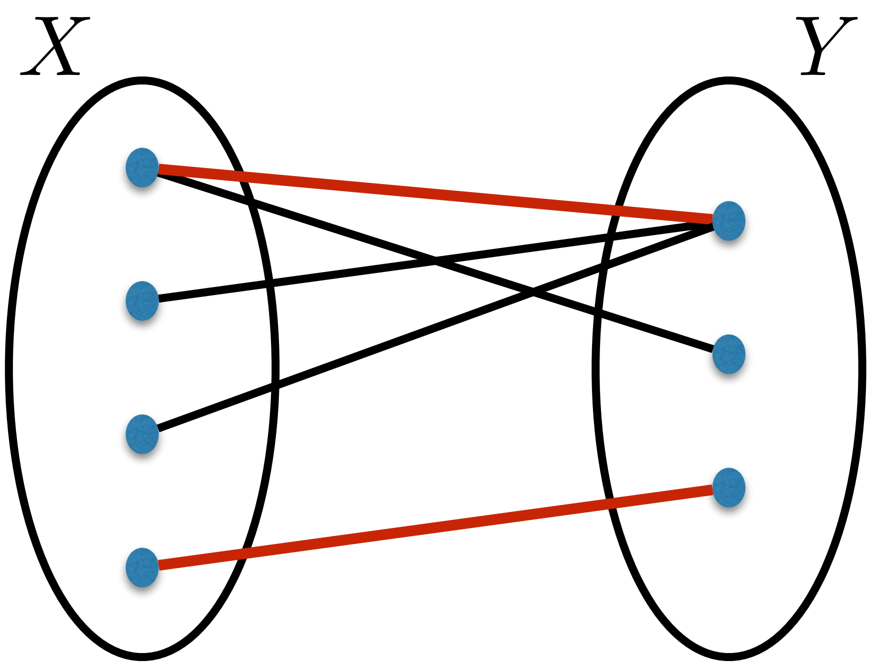
A **matching** :

A subset of the edges that do not share an endpoint.

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

matching

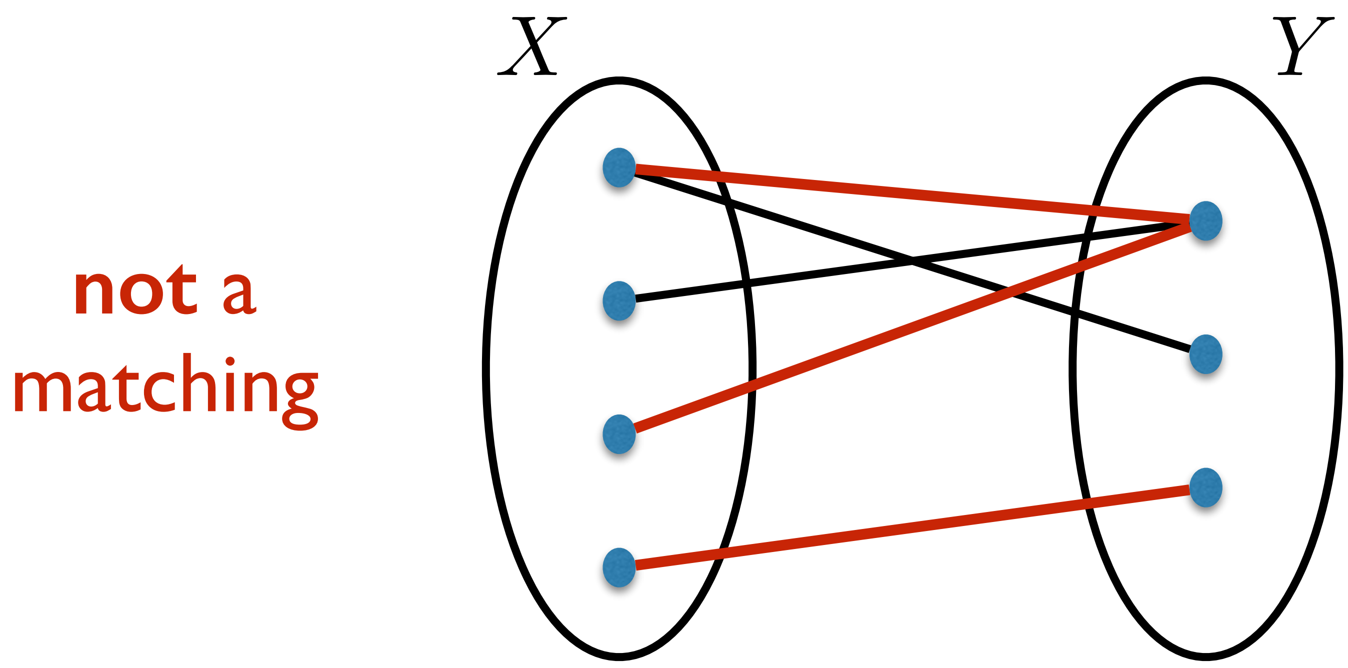


A **matching** :

A subset of the edges that do not share an endpoint.

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph



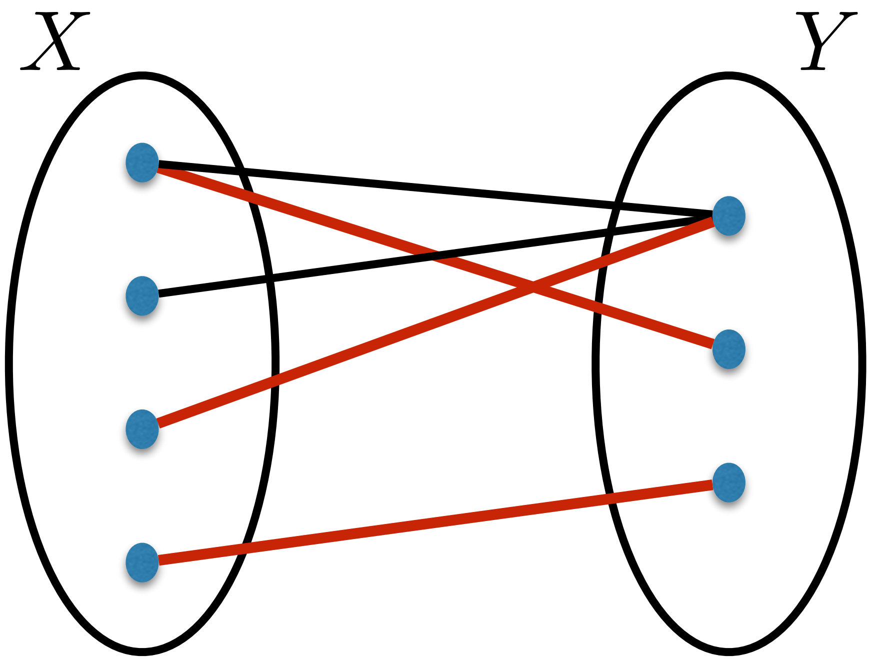
A **matching** :

A subset of the edges that do not share an endpoint.

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

maximum matching

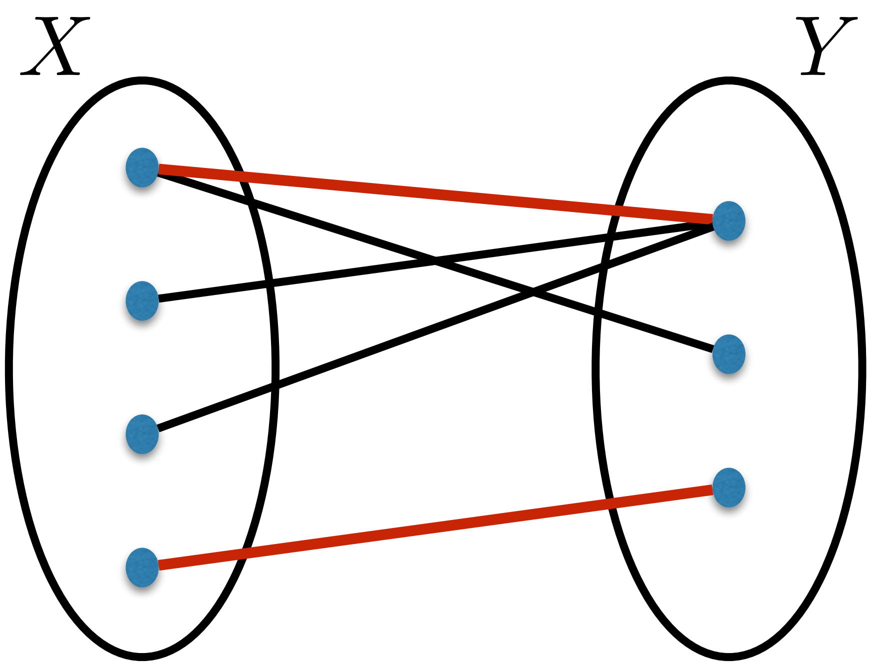


Maximum matching: a matching with largest number of edges (among all possible matchings).

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

maximal matching



Cannot add more edges.

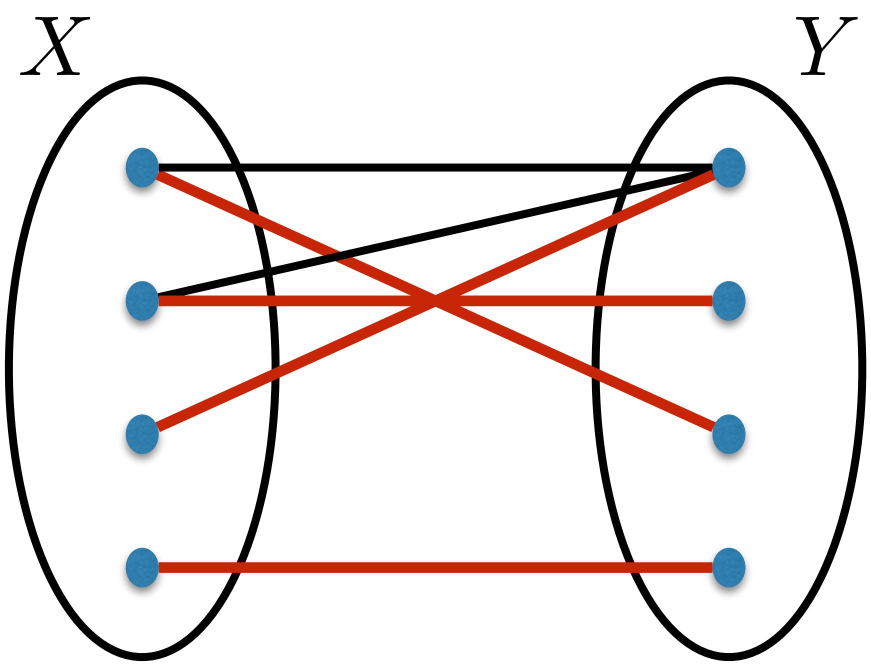
"Local optimum"

Maximal matching: a matching which cannot contain any more edges.

Matchings in bipartite graphs

Often, we are interested in finding a **matching** in a bipartite graph

perfect matching



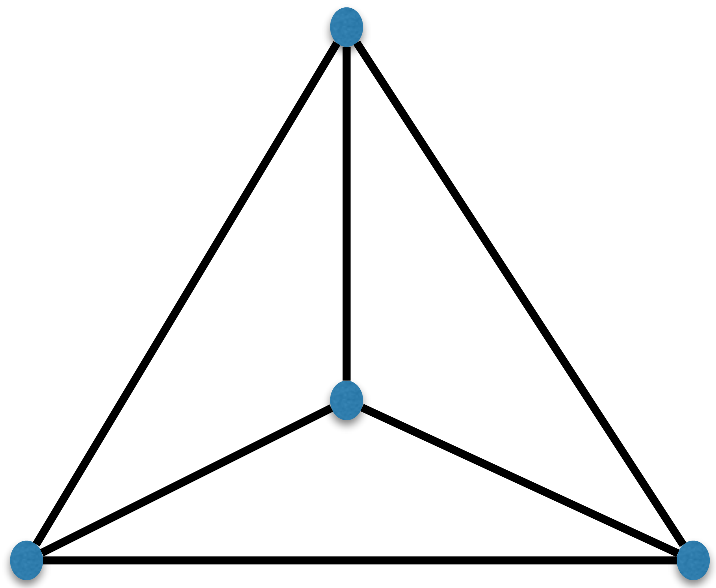
a necessary condition for perfect matching:

$|X| = |Y|$

Perfect matching: a matching that covers all vertices.

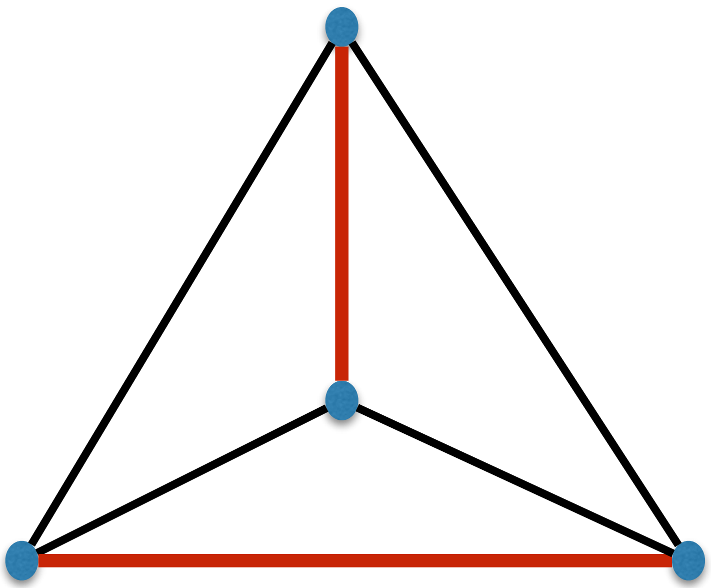
Important Note

We can define matchings for non-bipartite graphs as well.



Important Note

We can define matchings for non-bipartite graphs as well.



Maximum matching problem

The problem we want to solve is:

Maximum matching problem

Input: A graph $G = (V, E)$.

Output: A maximum matching in G.

The restriction where G is *bipartite* is already interesting!

Bipartite maximum matching problem

The problem we want to solve is:

Bipartite maximum matching problem

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

Output: A maximum matching in G.

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

Bipartite maximum matching problem

Bipartite maximum matching problem

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

Output: A maximum matching in G.

Is there a (trivial) algorithm to solve this problem?

Try all possible subsets of the edges.
Check if it is a matching.
Keep track of the maximum one found.

Running time: $\Omega(2^m)$

How do you solve a problem like this?

1. Formulate the problem

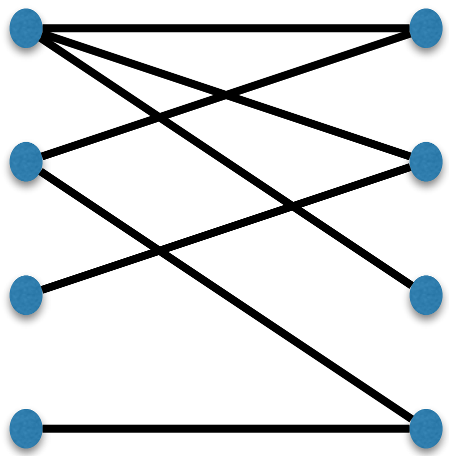
2. **Ask**: Is there a trivial algorithm?

3. **Ask**: Is there a better algorithm?

4. Find and analyze

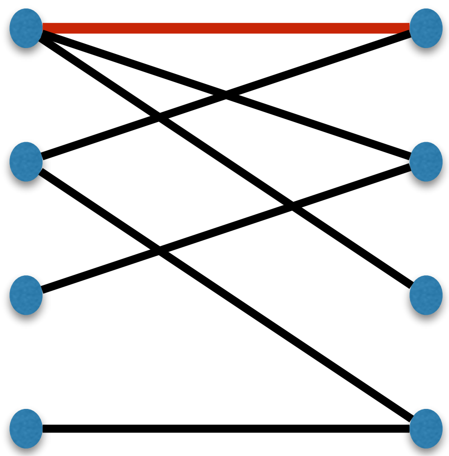
Bipartite maximum matching problem

What if we picked edges greedily?



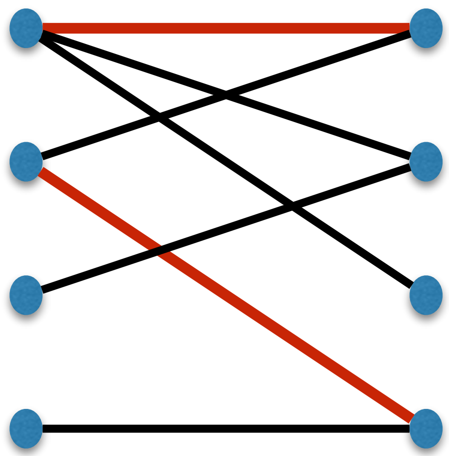
Bipartite maximum matching problem

What if we picked edges greedily?



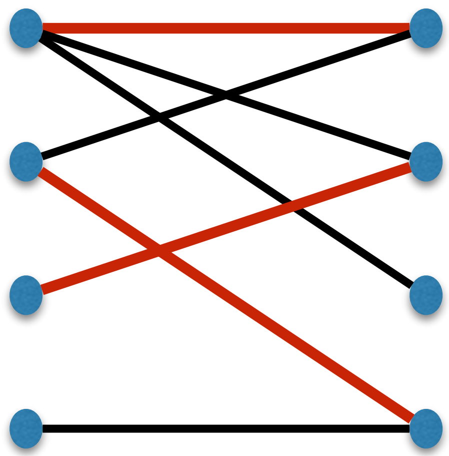
Bipartite maximum matching problem

What if we picked edges greedily?



Bipartite maximum matching problem

What if we picked edges greedily?



maximal matching

but not maximum

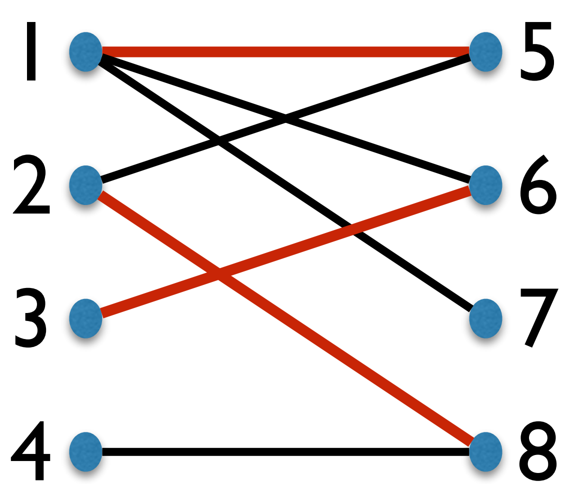
Is there a way to get out of this local optimum?

Augmenting paths

Let **M** be some matching.

An *augmenting path* with respect to **M** is a path in **G** such that:

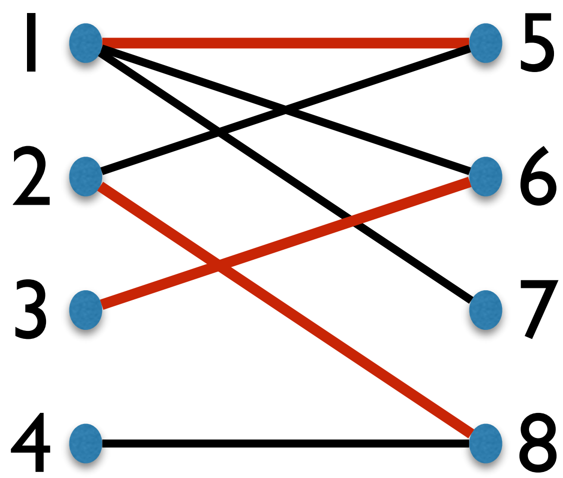
- the edges in the path alternate between being in **M** and not being in **M**
- the first and last vertices are not matched by **M**



Augmenting path:

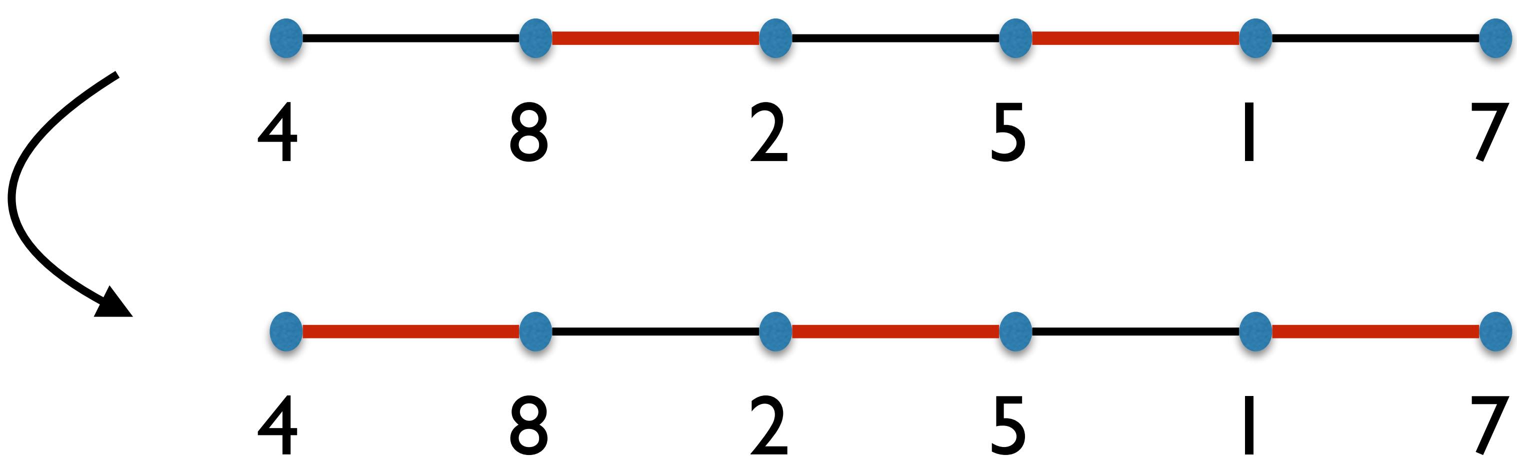
4-8-2-5-1-7

Augmenting paths



Augmenting path:

4-8-2-5-1-7



augmenting path $\Longrightarrow$ can obtain a bigger matching.

Augmenting paths and maximum matchings

augmenting path $\implies$ can obtain a bigger matching.

In fact, it turns out:

no augmenting path $\implies$ maximum matching.

Theorem:

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

Augmenting paths and maximum matchings

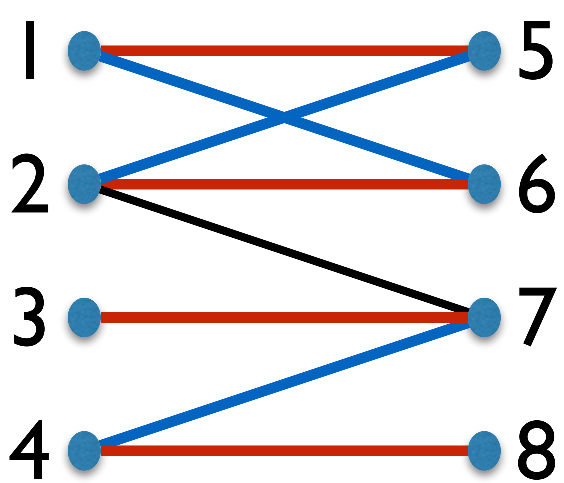
Proof:

If there is an augmenting path with respect to **M**, we saw that **M** is not maximum.

Want to show:

If **M** is not maximum, then there is an augmenting path.

Let **M*** be a maximum matching. $|M^*| > |M|$.

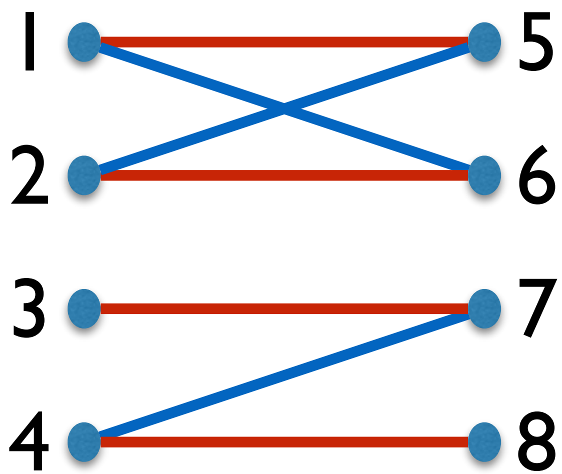


Let **S** be the set of edges contained in **M*** or **M** but not both.

$S = (M^* \cup M) - (M \cap M^*)$

Augmenting paths and maximum matchings

Proof:



Let **S** be the set of edges contained in **M*** or **M** but not both.

S = (**M*** ∪ **M**) - (**M** ∩ **M***)

(will find an augmenting path in **S**)

What does **S** look like?

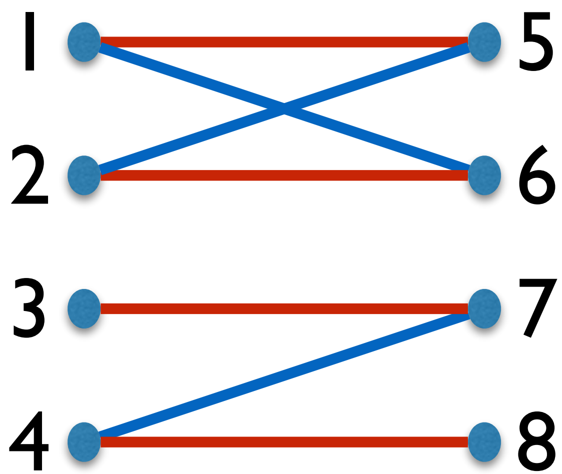
Each vertex has degree at most 2. (why?)

So **S** is a collection of cycles and paths. (exercise)

The edges alternate **red** and **blue**.

Augmenting paths and maximum matchings

Proof:



Let **S** be the set of edges contained in **M*** or **M** but not both.

$$\mathbf{S} = (\mathbf{M}^* \cup \mathbf{M}) - (\mathbf{M} \cap \mathbf{M}^*)$$

So **S** is a collection of cycles and paths. (exercise)

The edges alternate **red** and **blue**.

\# red > # blue in **S**

\# red = # blue in cycles

So $\exists$ a path with # red > # blue.

This is an *augmenting path* with respect to **M**. $\square$

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?

Exercise (homework?)

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.

$O(m \cdot n)$ time algorithm in bipartite graphs.

Today's Goal:

Maximum matching problem
(in bipartite graphs)

Stable matching problem

Stable matching problem

Finding internship

Macrosoft

Moogle
Google

UMBRELLA
CORPORATION
"Our business is life itself..."

Finding internship

1. Moogle Google
2. Macrosoft
3. 

4. UMBRELLA CORPORATION "Our business is life itself..."

Macrosoft

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

Moogle Google

.
.
.

UMBRELLA CORPORATION

"Our business is life itself..."

Other examples:

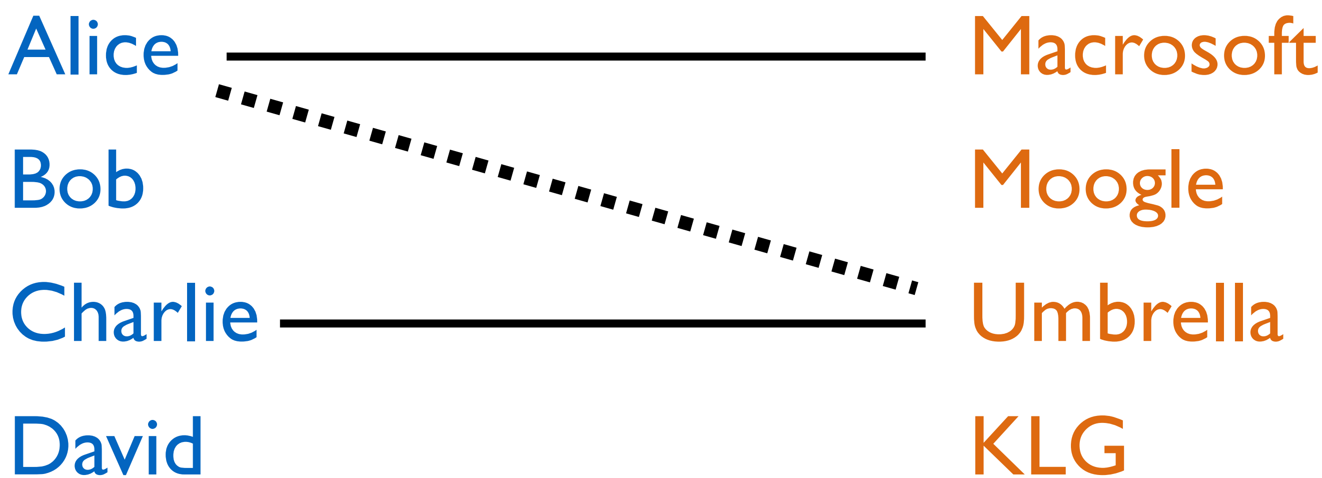
medical residents - hospitals

students - colleges

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

Finding internship

What can go wrong?

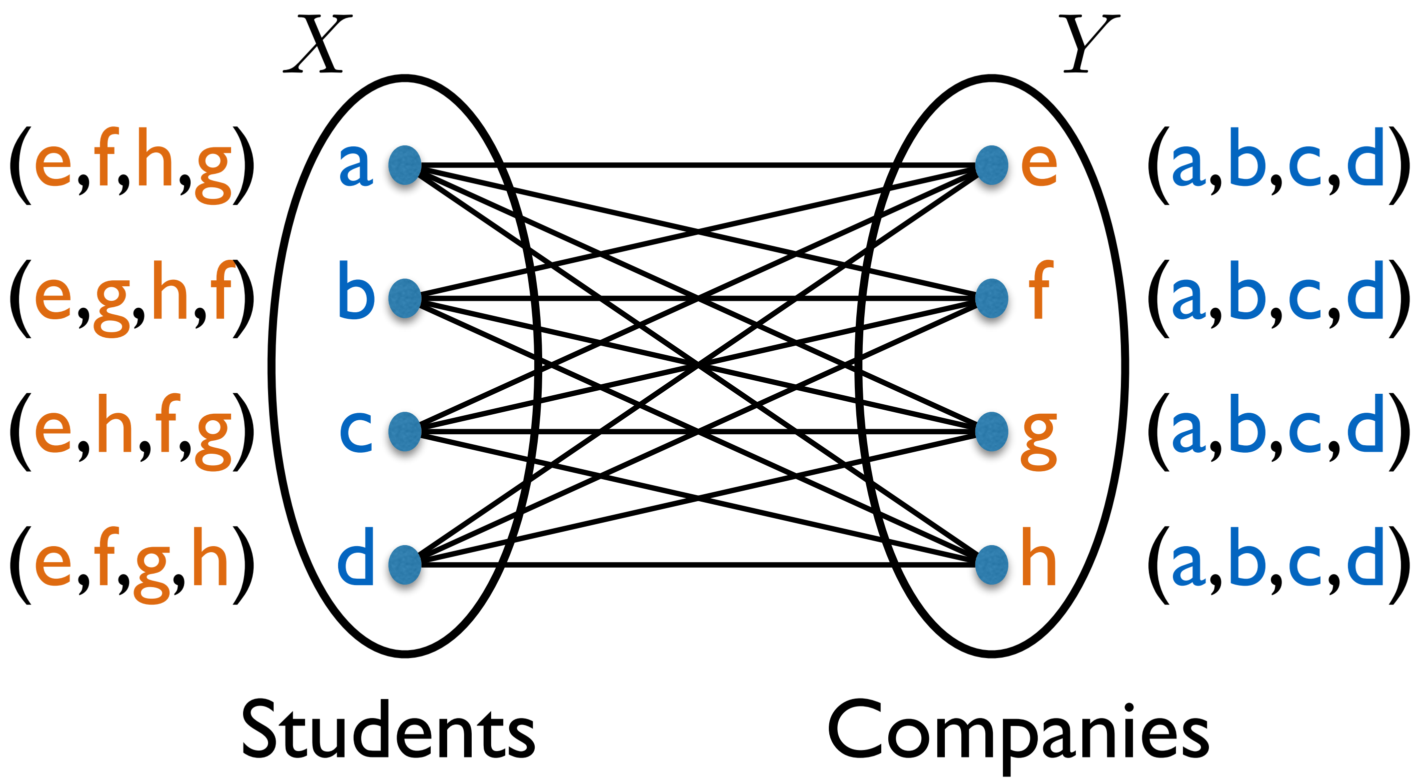


Suppose Alice gets "matched" with Macrosoft.
Charlie gets "matched" with Umbrella.

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

Finding internship: Formalizing the problem

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

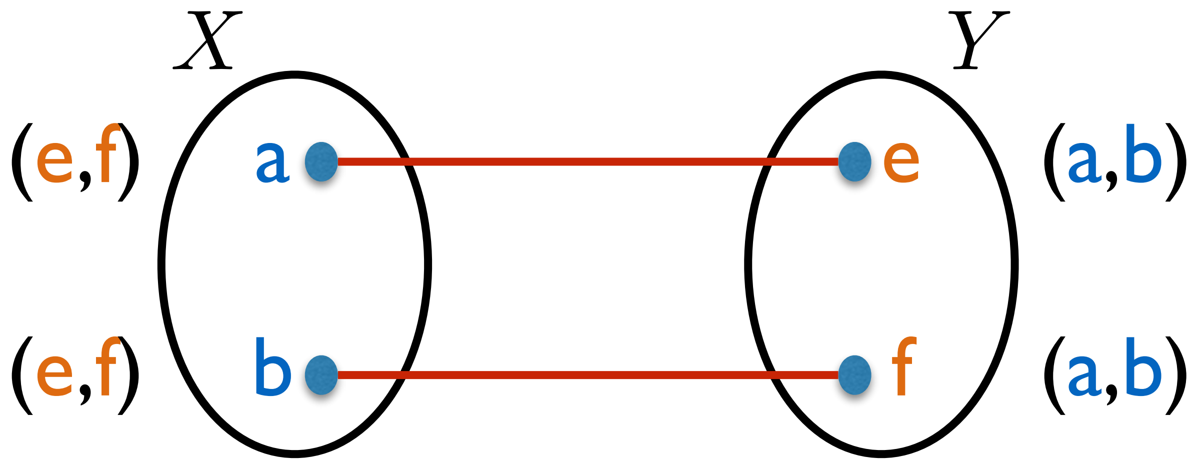


$$|X| = |Y| = n$$

Goal: Find a stable matching.

Finding internship: Formalizing the problem

What is a stable matching?

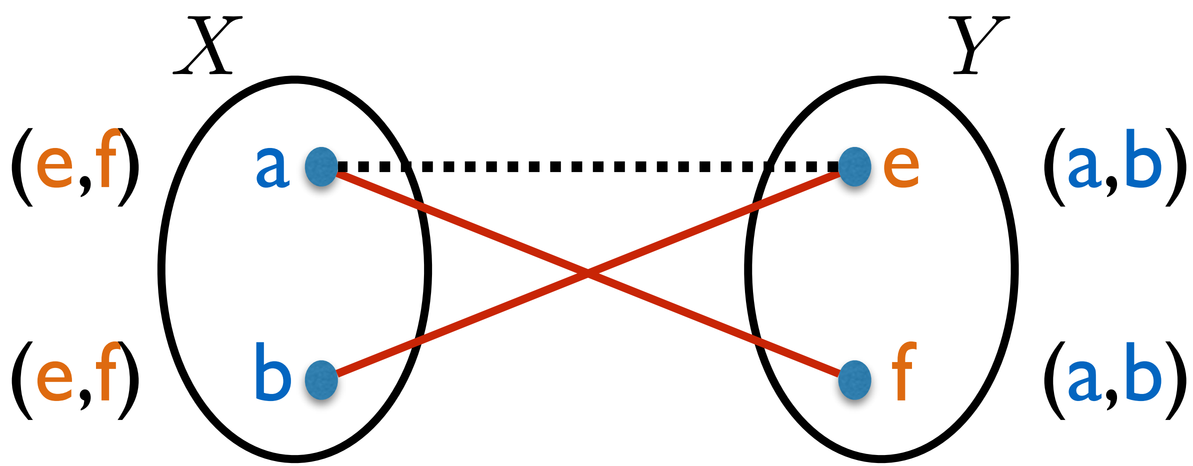


1. It has to be a perfect matching.
2. Cannot contain an unstable pair:

   A pair (x, y) not matched
   but they prefer each other over their current partners.

Finding internship: Formalizing the problem

What is a stable matching?



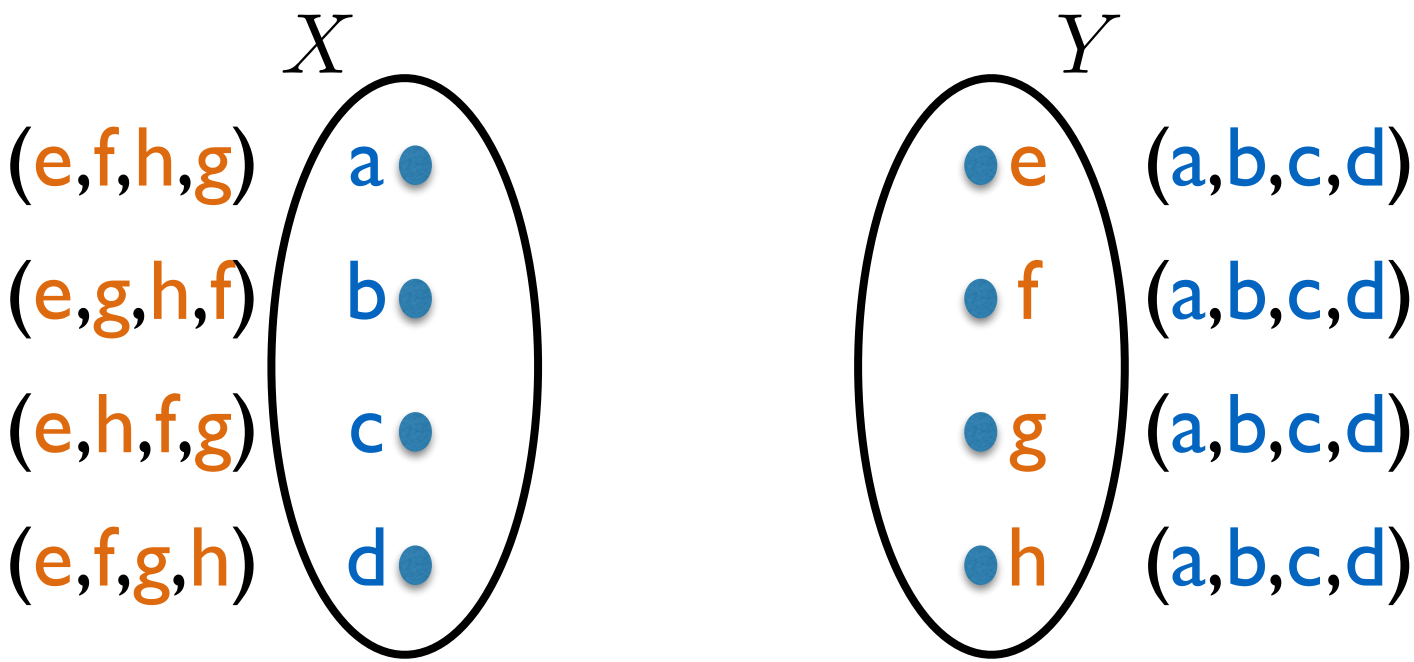
(a, e) is an unstable pair.

1. It has to be a perfect matching.
2. Cannot contain an unstable pair:

   A pair (x, y) not matched
   but they prefer each other over their current partners.

Finding internship: Formalizing the problem

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

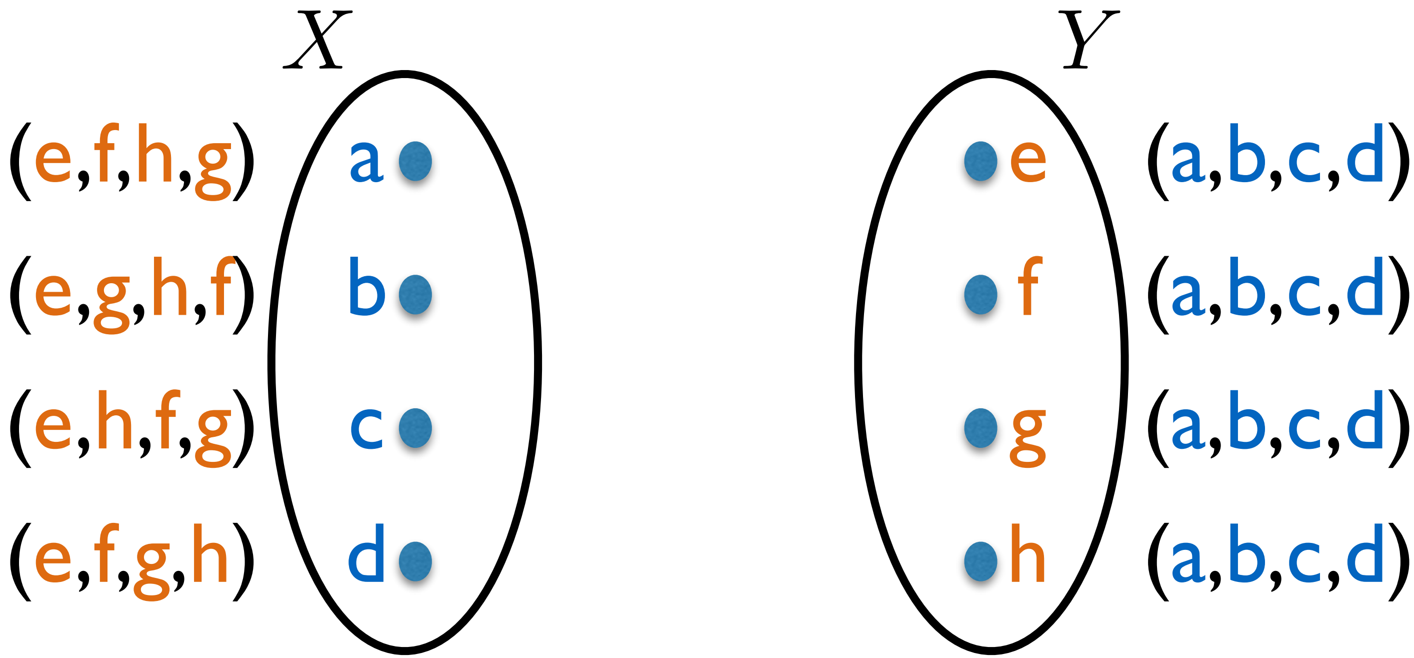


$$|X| = |Y| = n$$

Goal: Find a stable matching.

(Is it guaranteed to always exist?)

Stable matching: Is there a trivial algorithm?



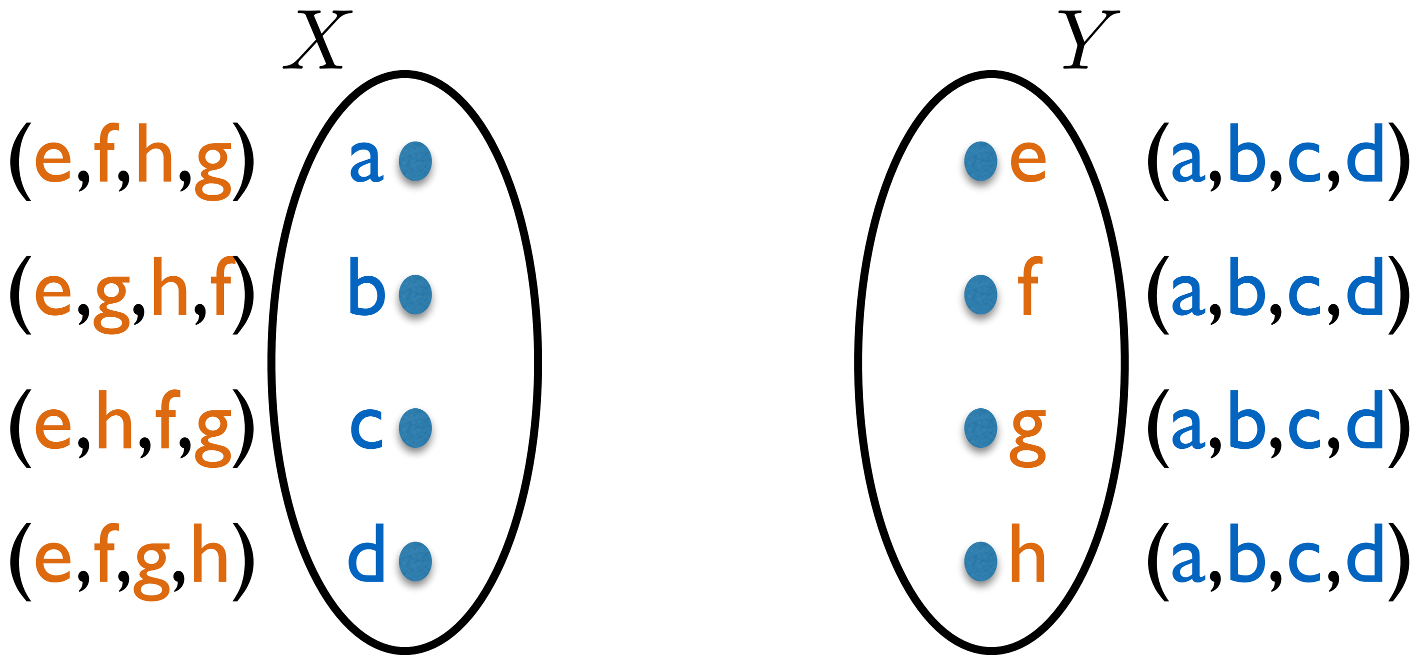
Trivial algorithm:

Try all possible perfect matchings,

and check if it is stable.

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

Stable matching: Is there a trivial algorithm?



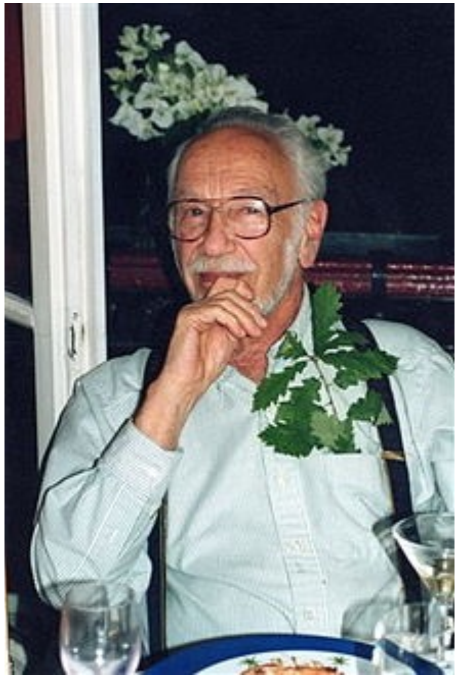
Trivial algorithm:

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

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

Stable matching: Can we do better?

The Gale-Shapley Proposal Algorithm (1962)



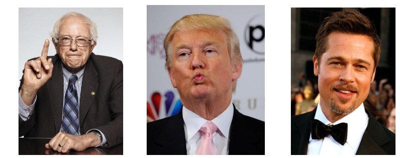
Nobel Prize in Economics
2012

The Gale-Shapley proposal algorithm

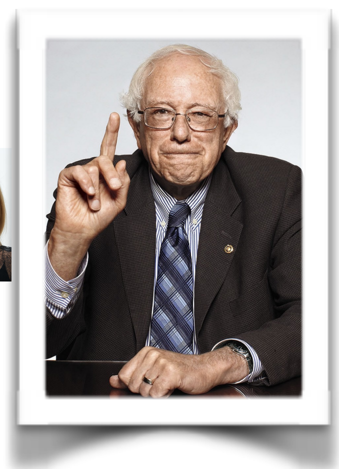
The Gale-Shapley proposal algorithm

1 2 3

1 2 3



1 2 3

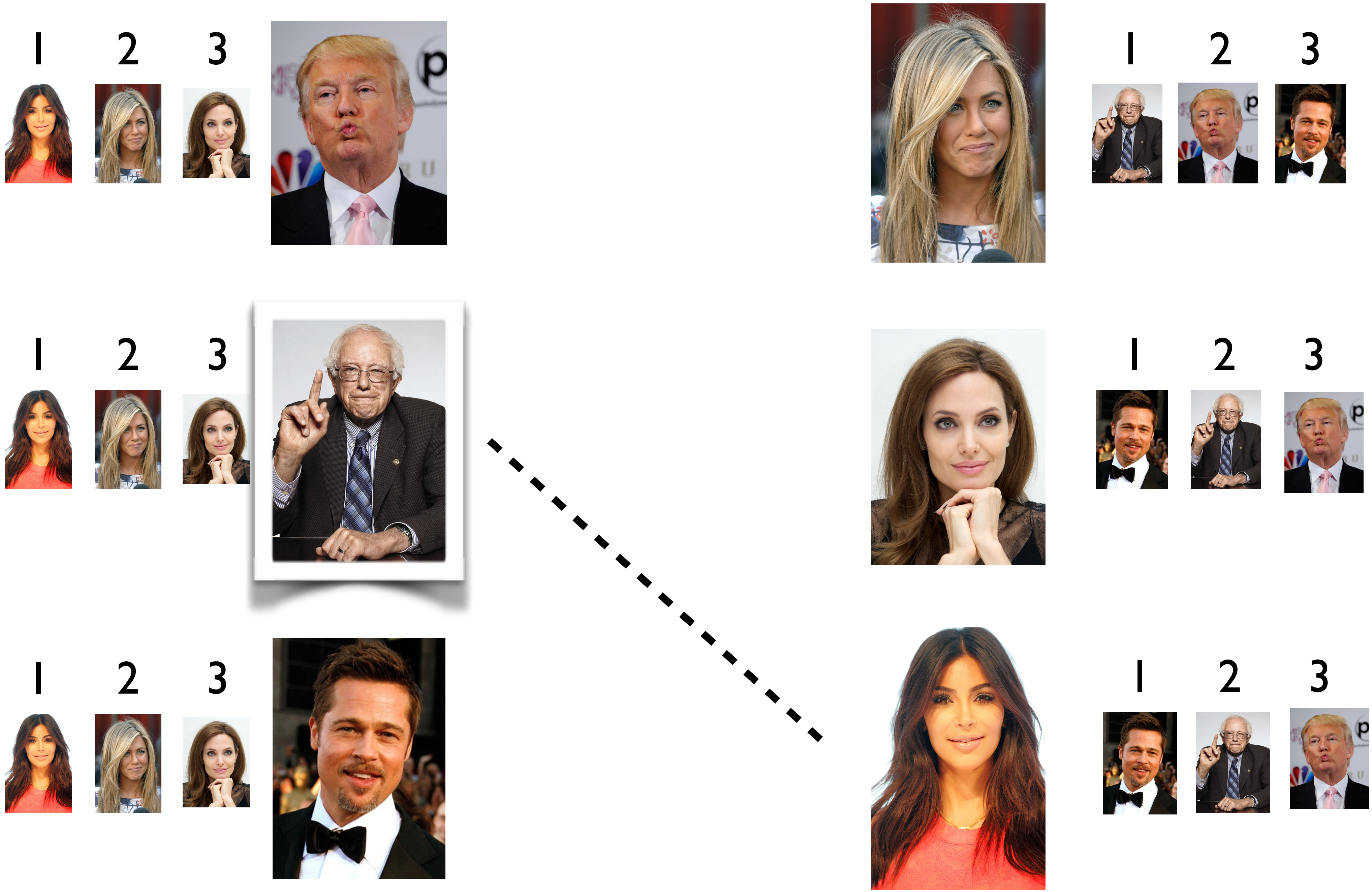


1 2 3

1 2 3

1 2 3

The Gale-Shapley proposal algorithm

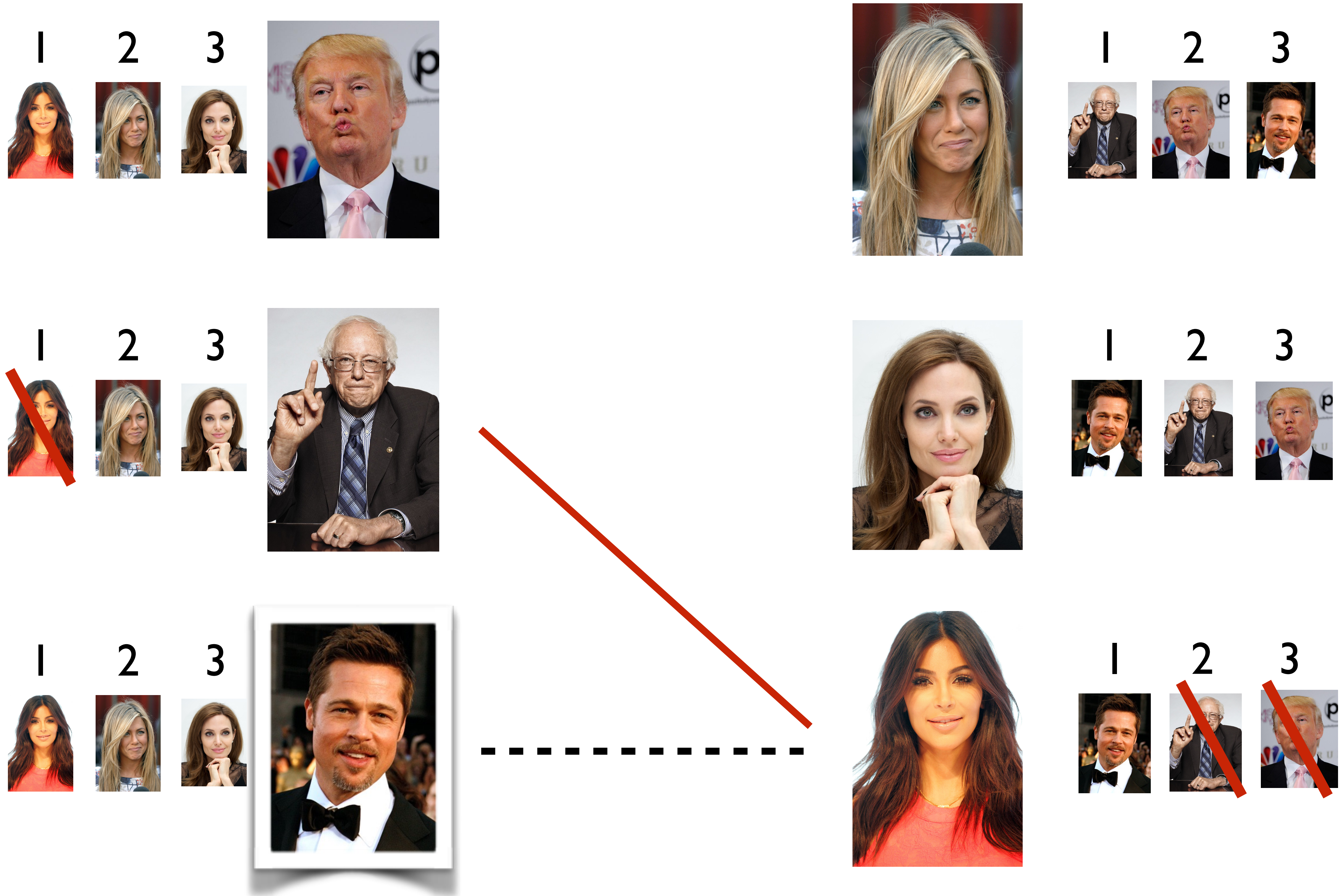


The Gale-Shapley proposal algorithm

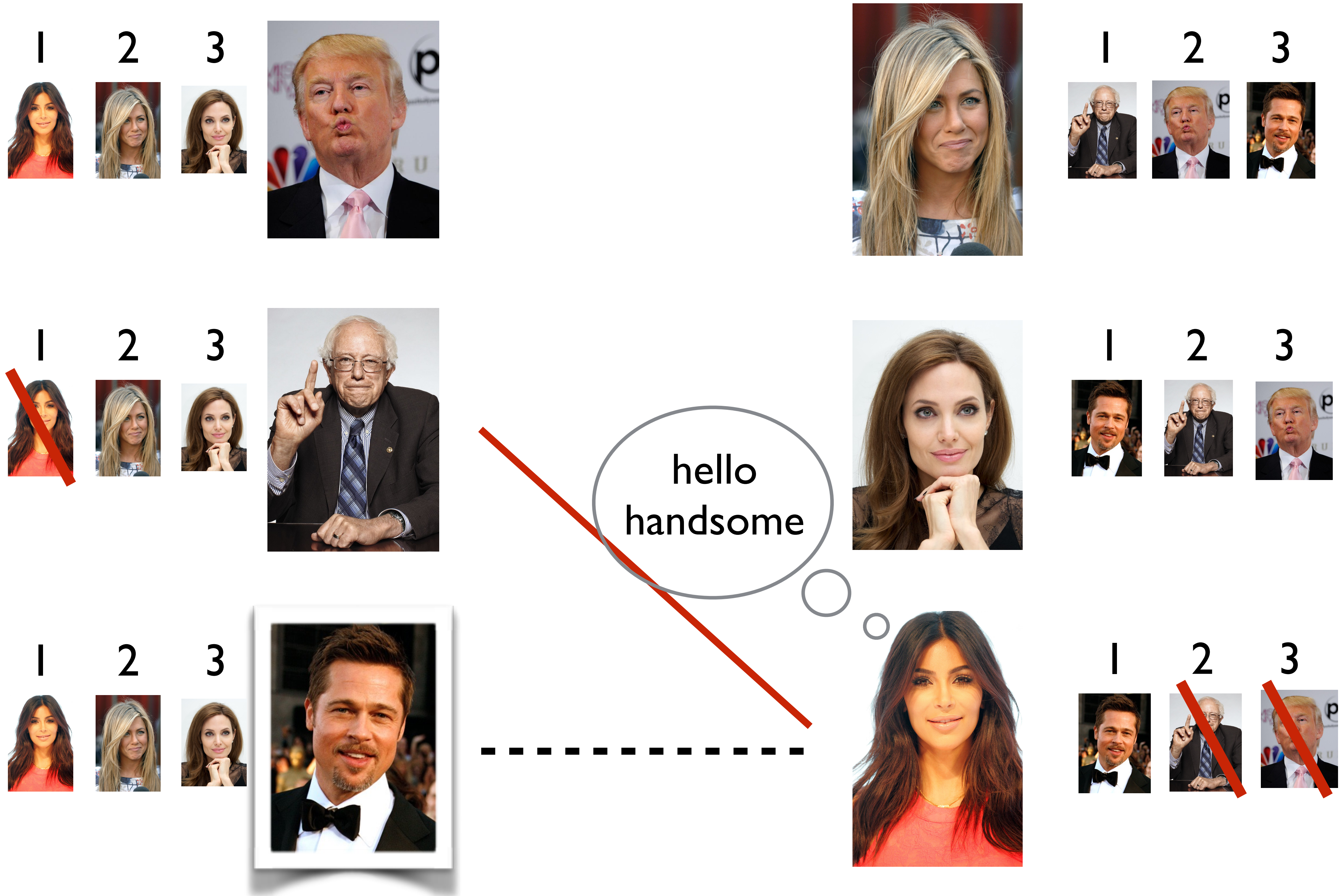
The Gale-Shapley proposal algorithm

The Gale-Shapley proposal algorithm

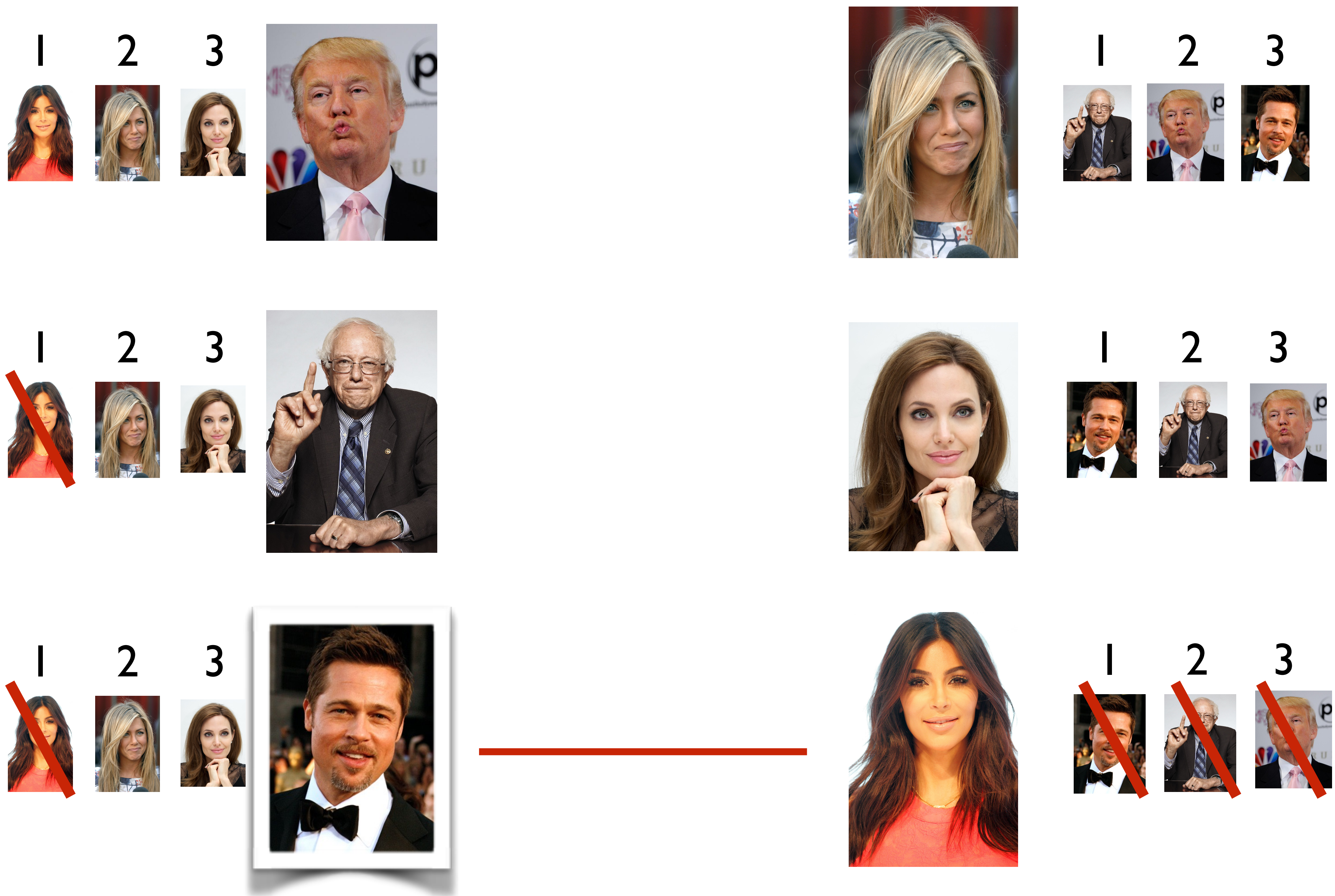
The Gale-Shapley proposal algorithm



The Gale-Shapley proposal algorithm

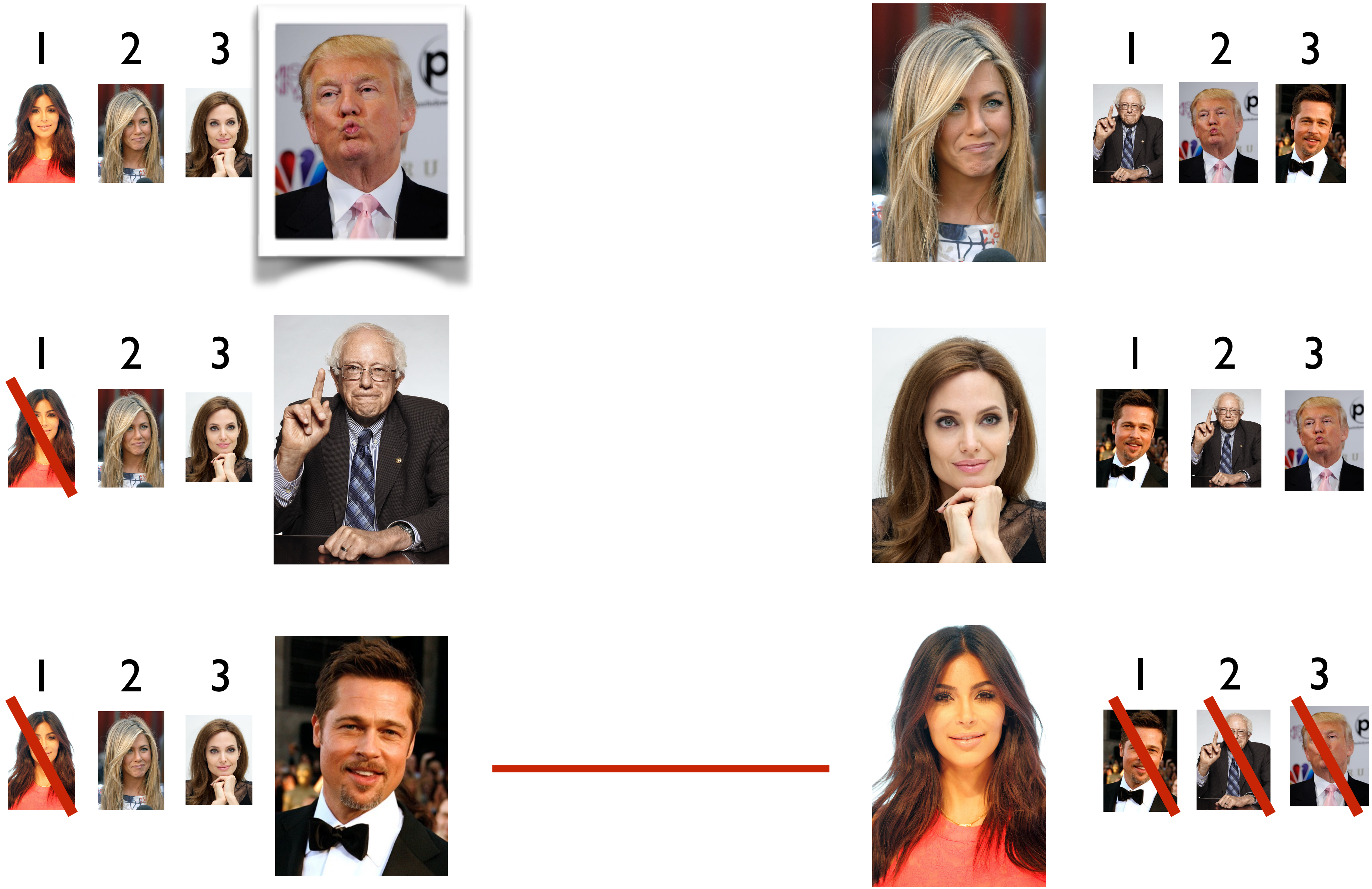


The Gale-Shapley proposal algorithm

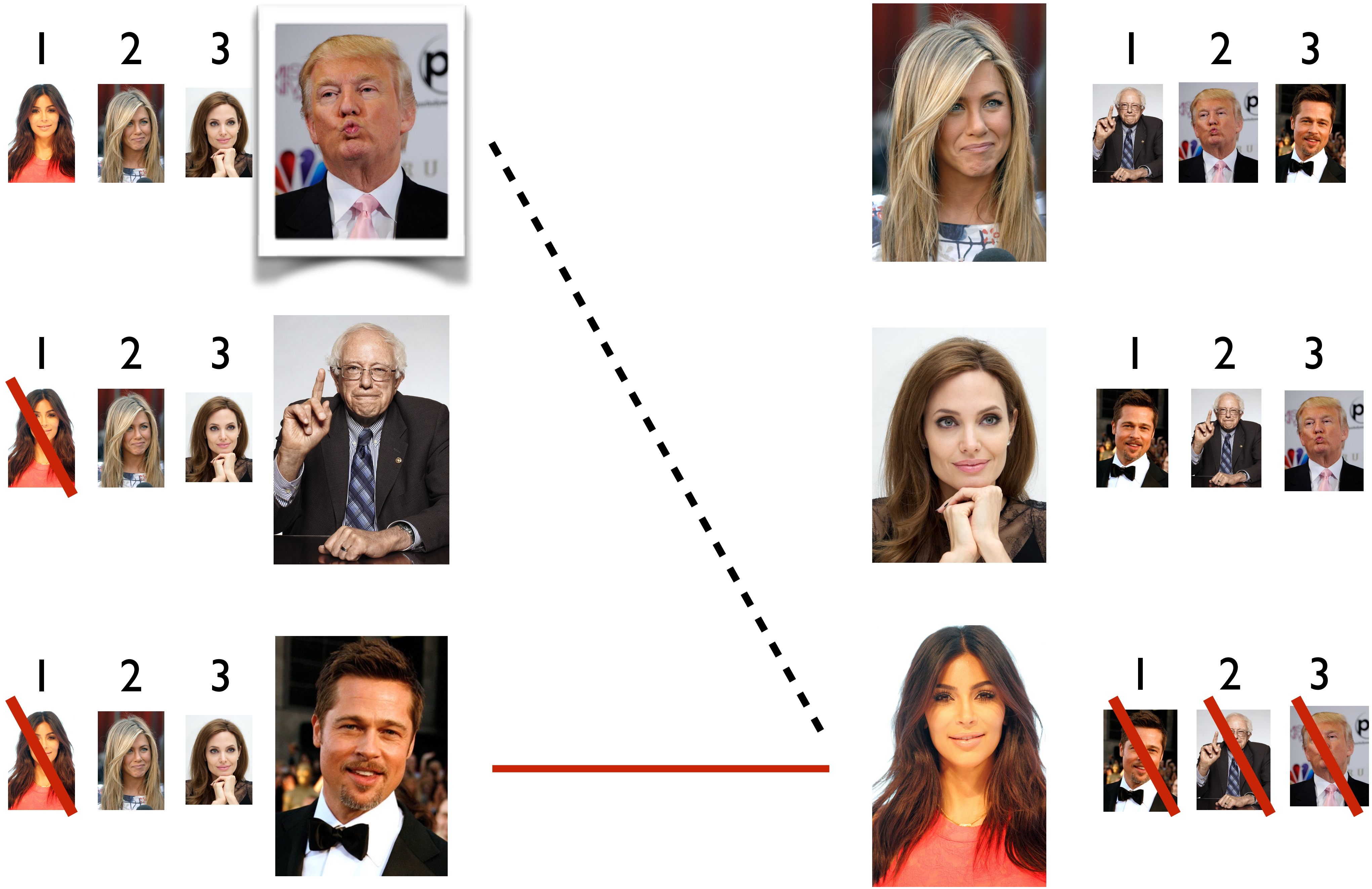


The Gale-Shapley proposal algorithm

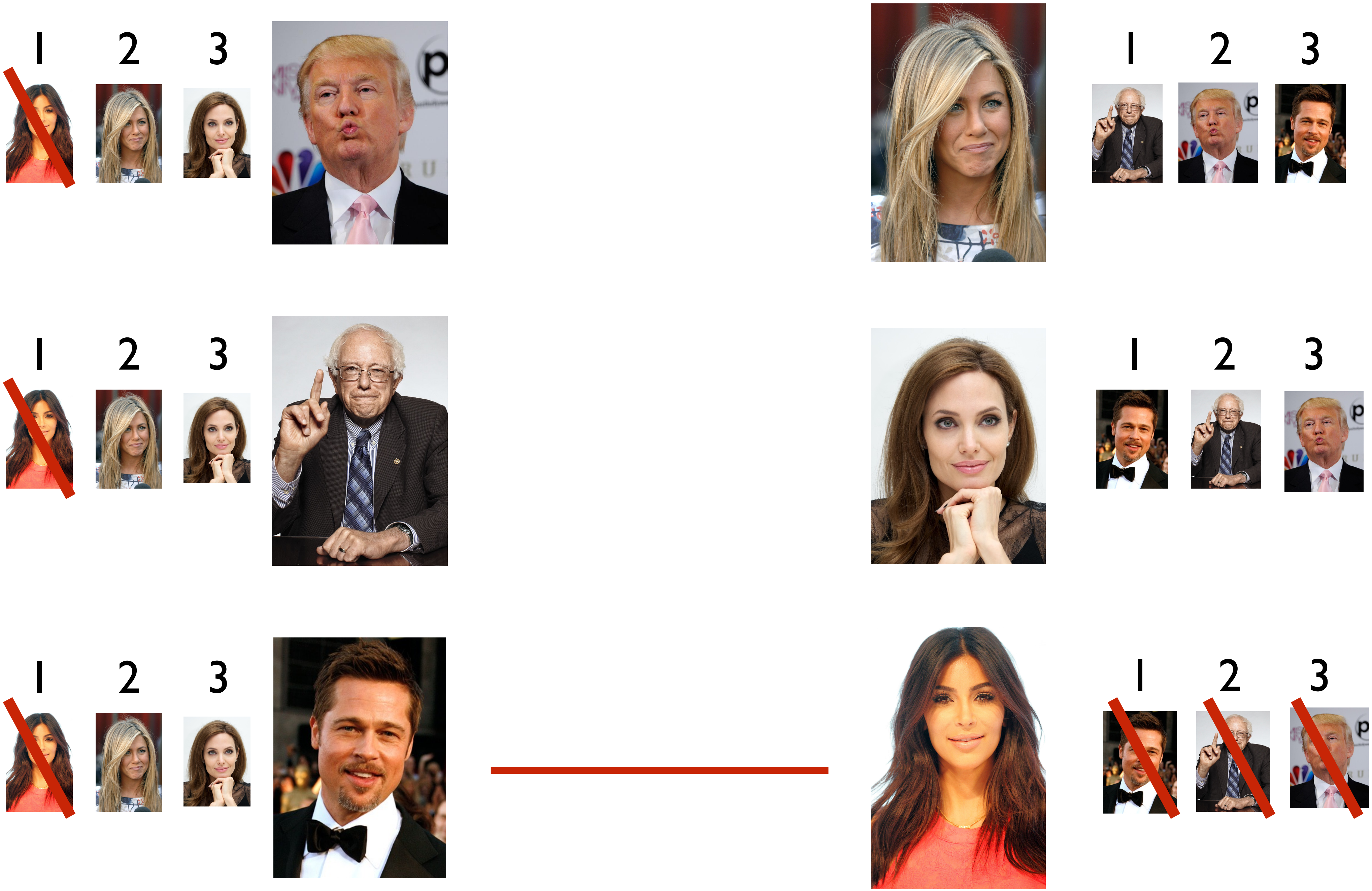
The Gale-Shapley proposal algorithm



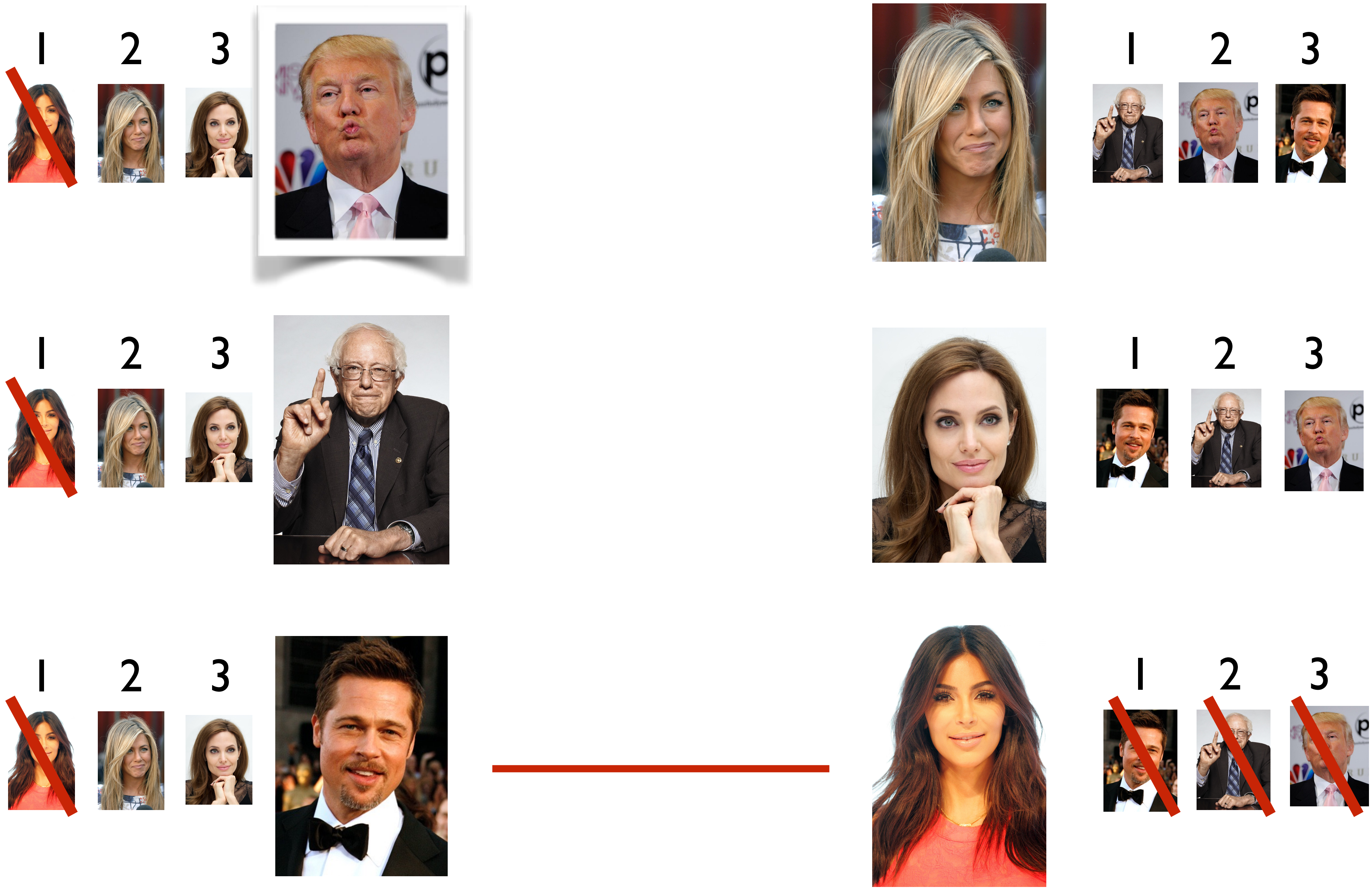
The Gale-Shapley proposal algorithm



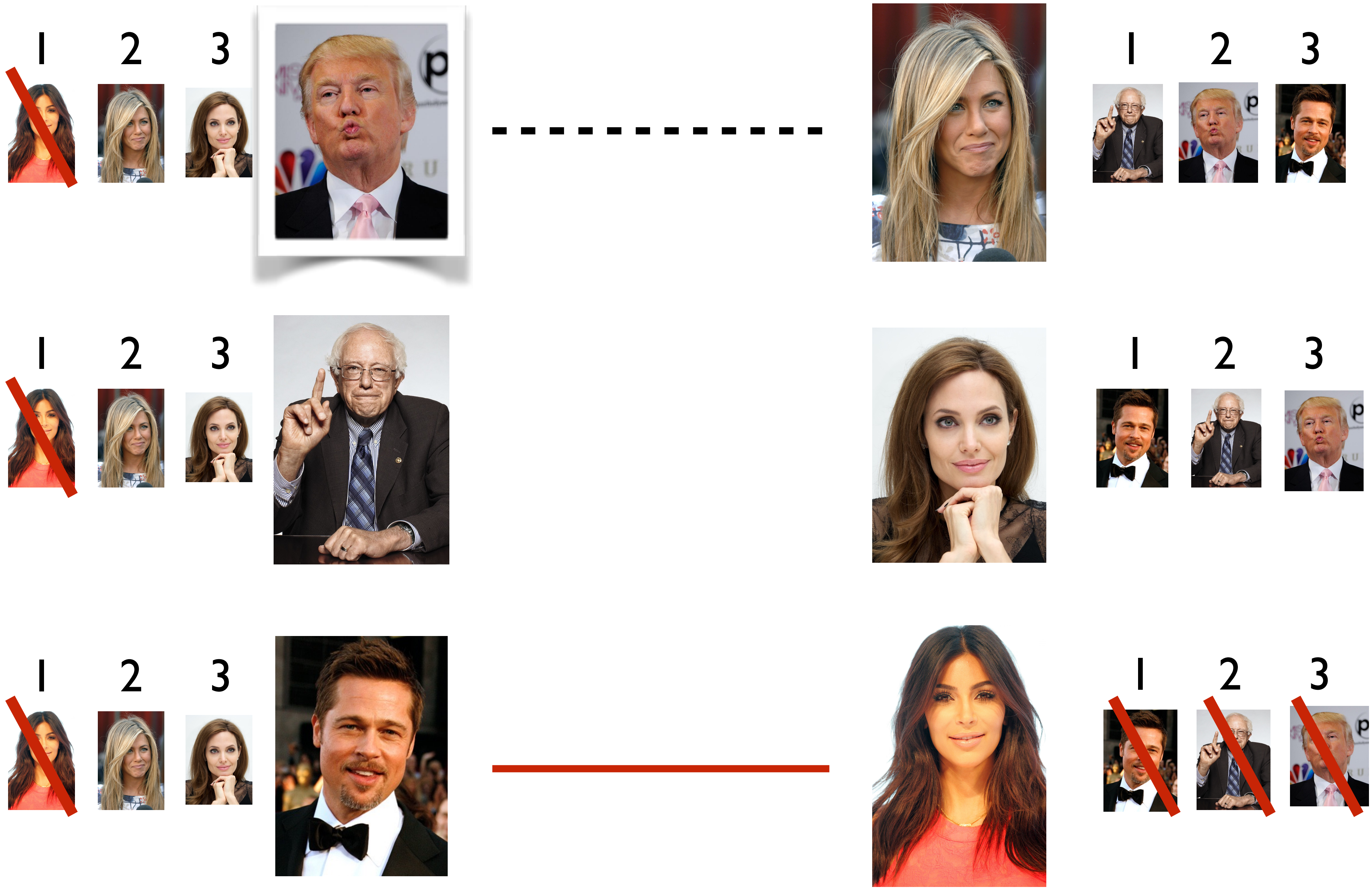
The Gale-Shapley proposal algorithm



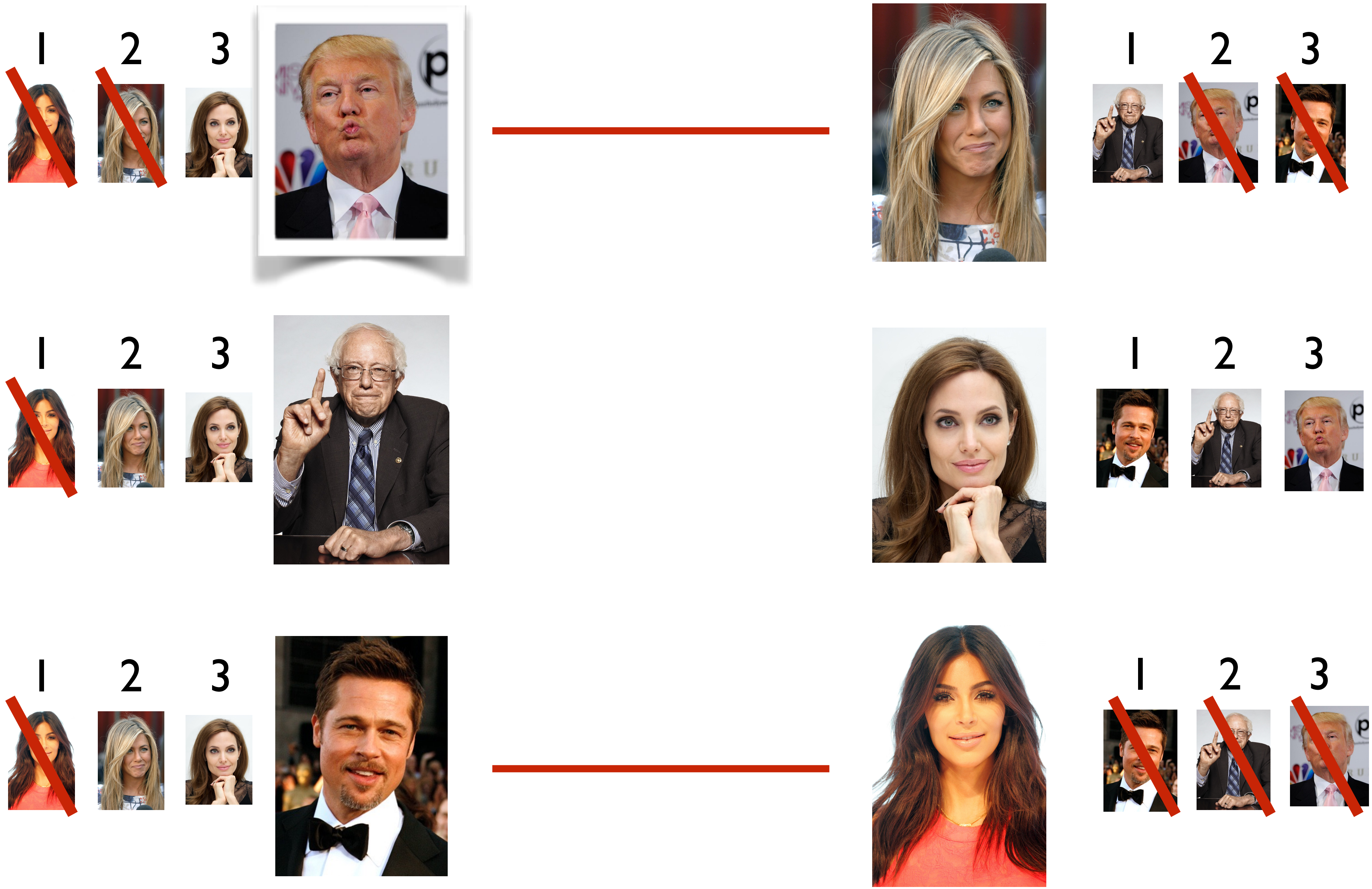
The Gale-Shapley proposal algorithm



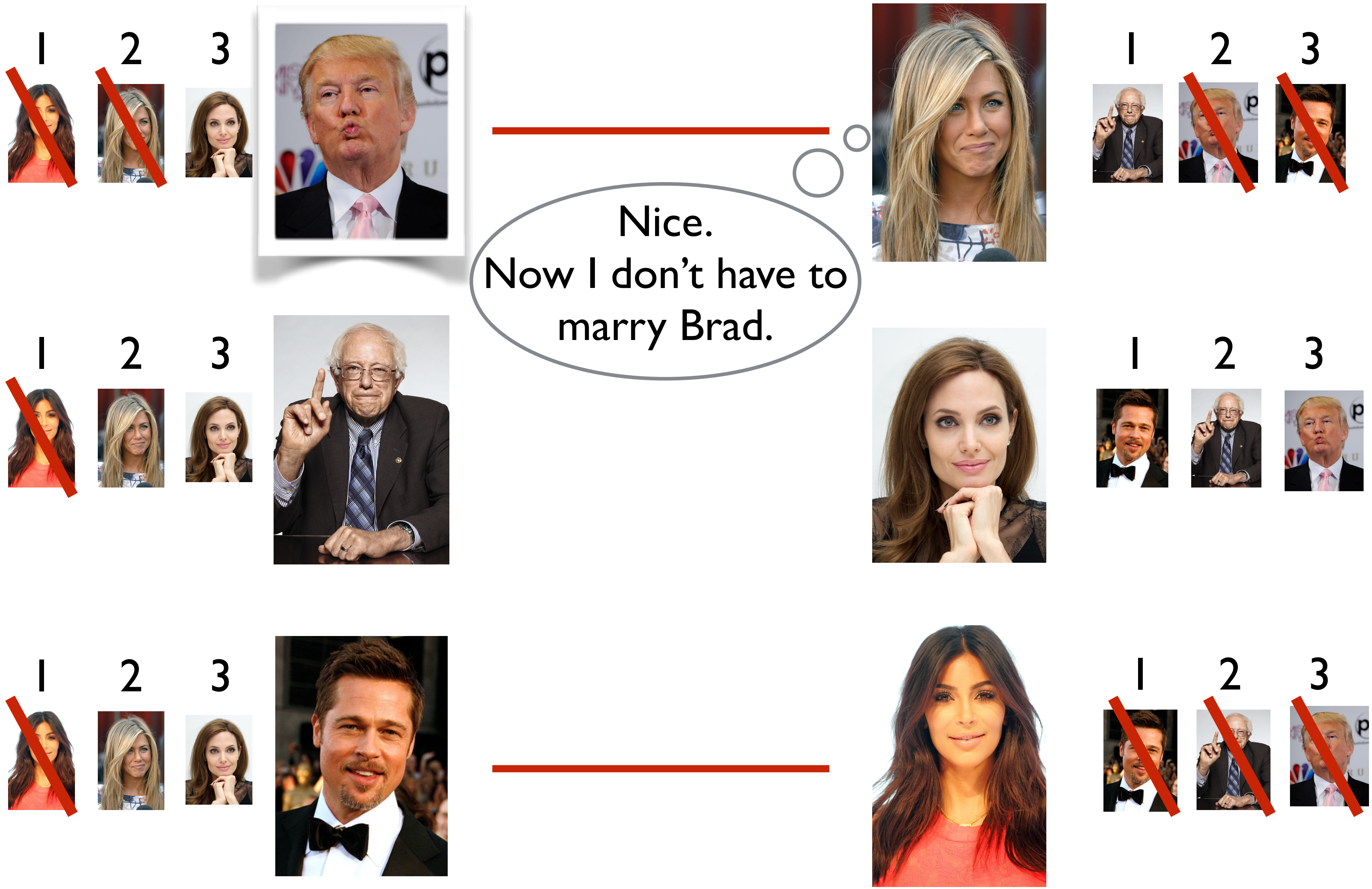
The Gale-Shapley proposal algorithm



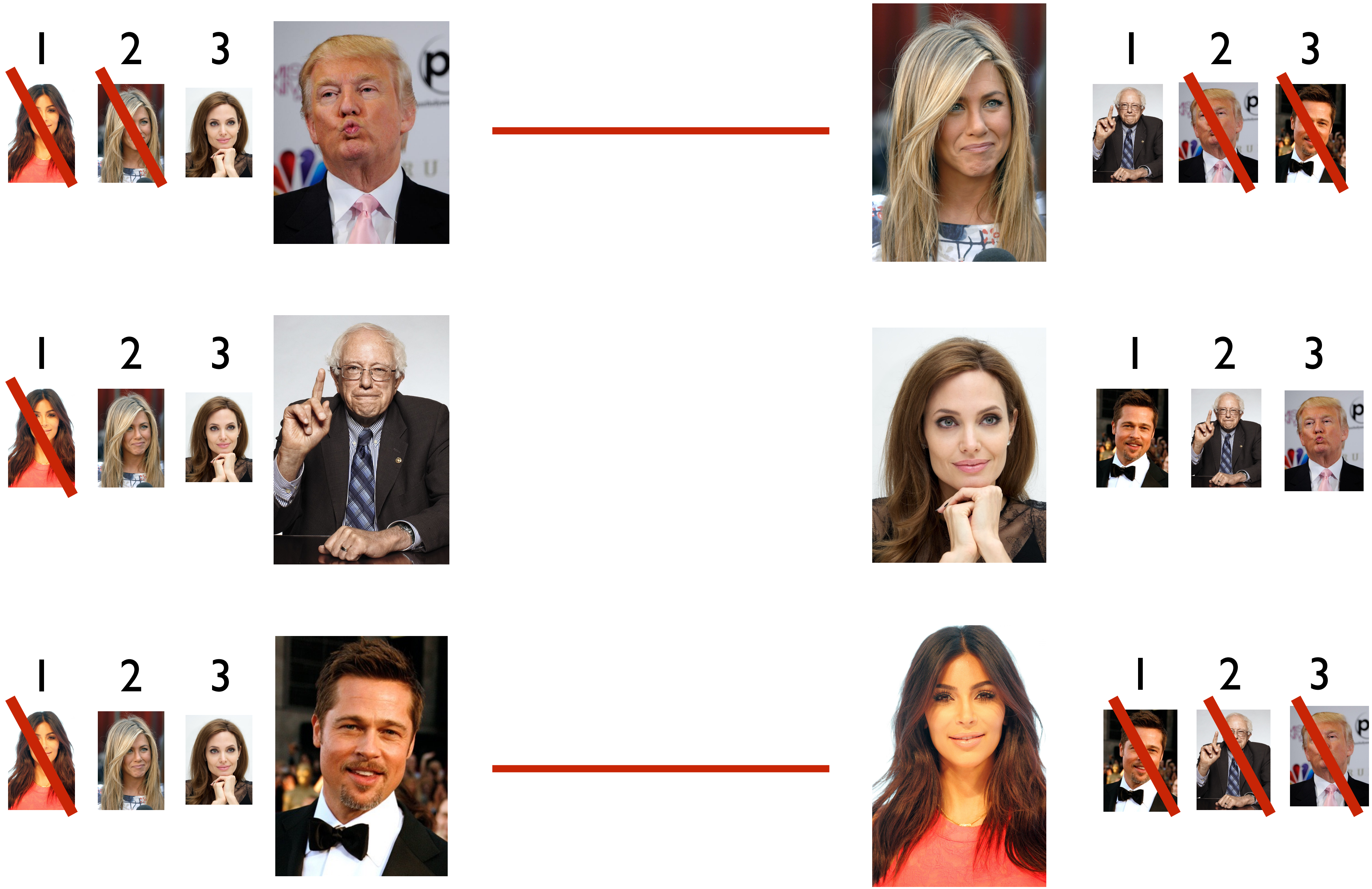
The Gale-Shapley proposal algorithm



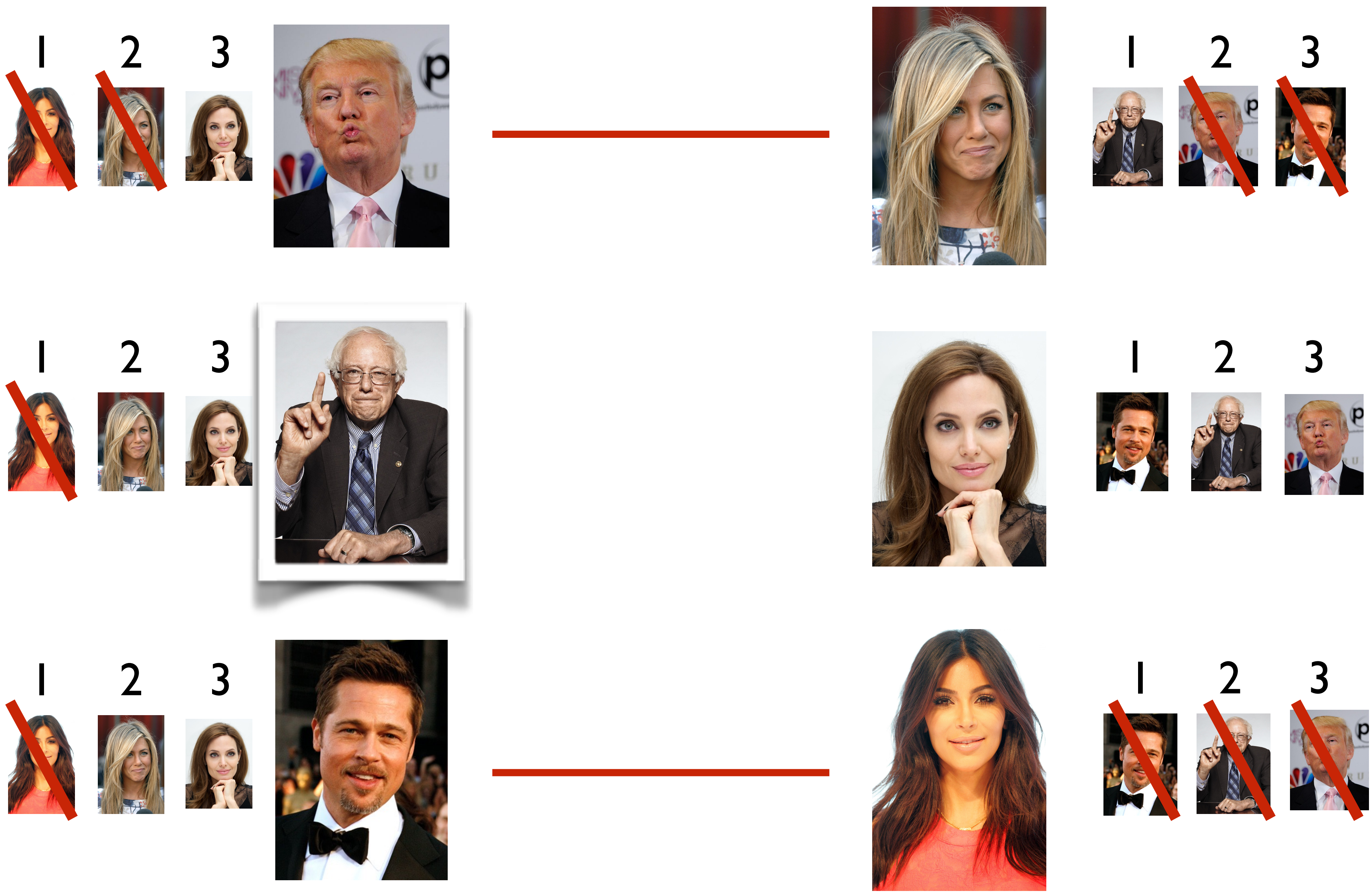
The Gale-Shapley proposal algorithm



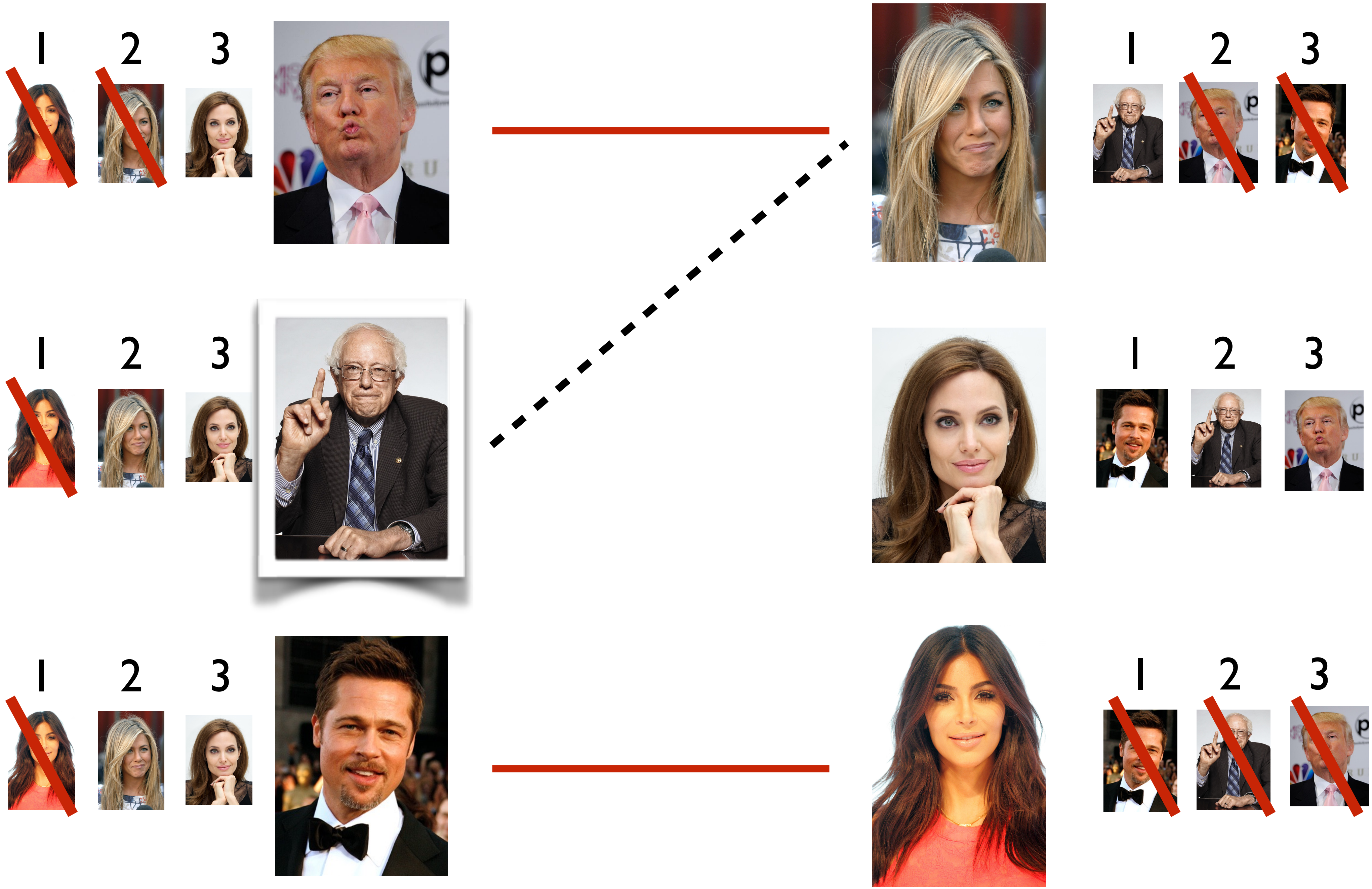
The Gale-Shapley proposal algorithm



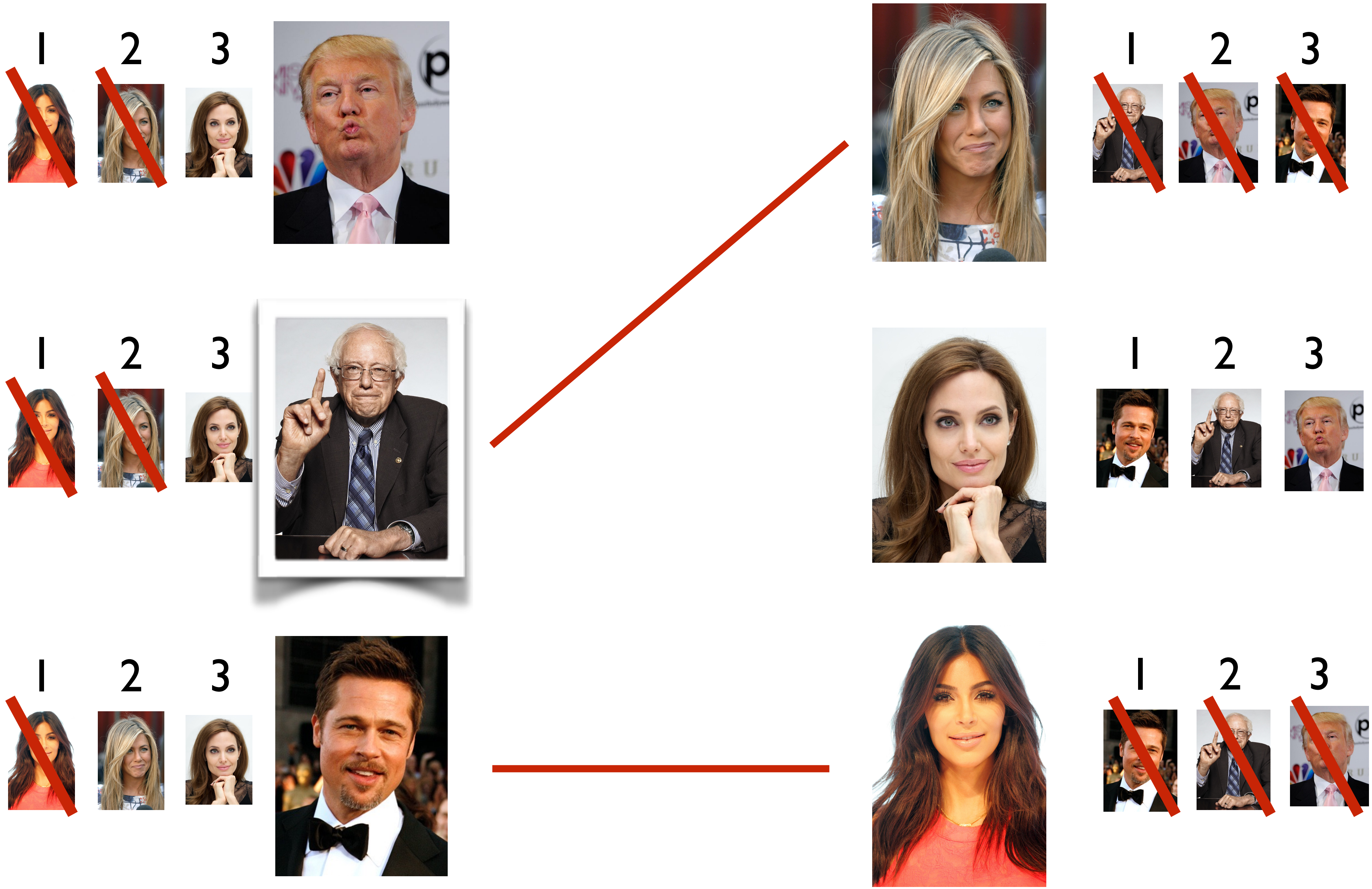
The Gale-Shapley proposal algorithm



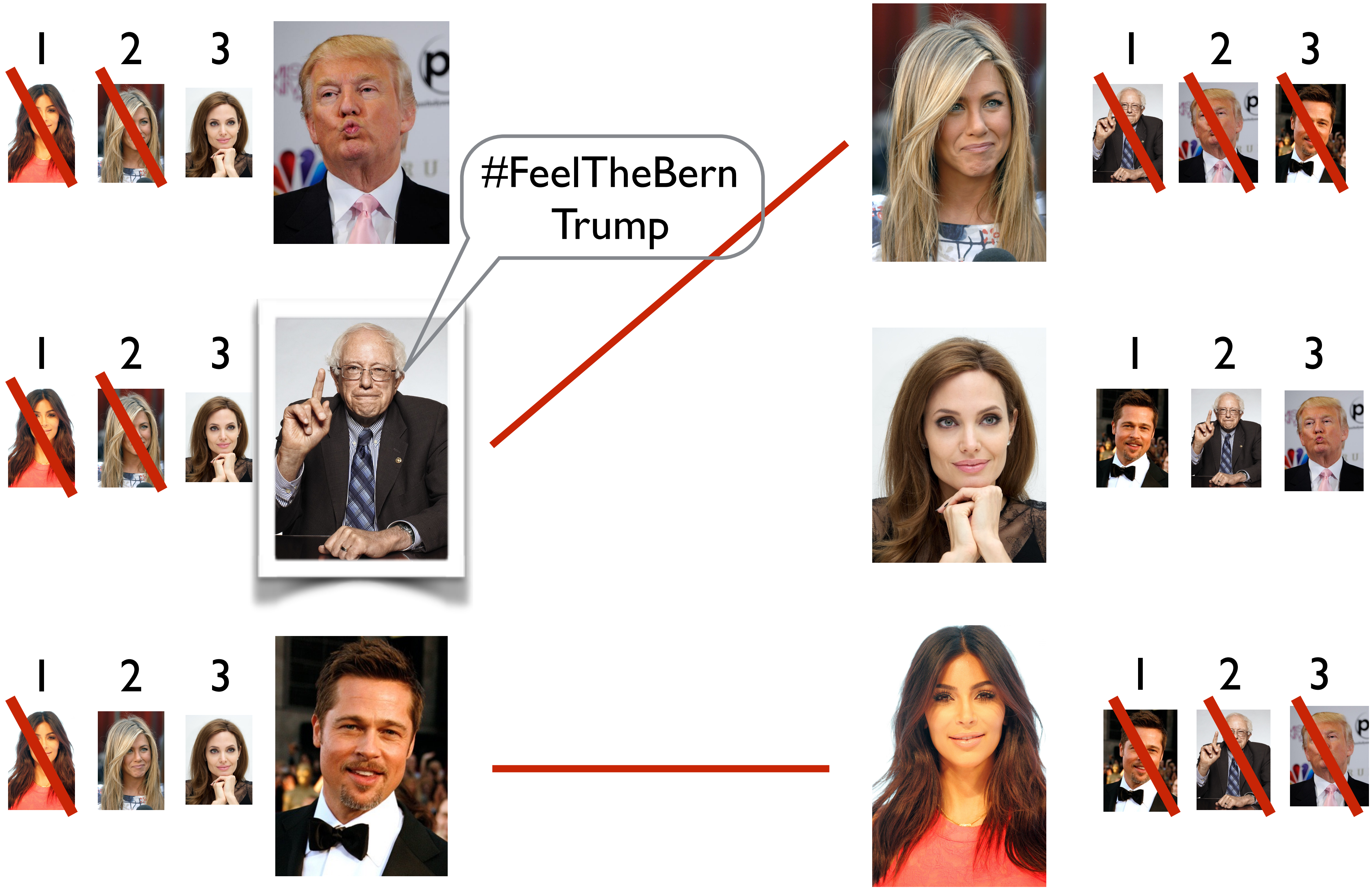
The Gale-Shapley proposal algorithm



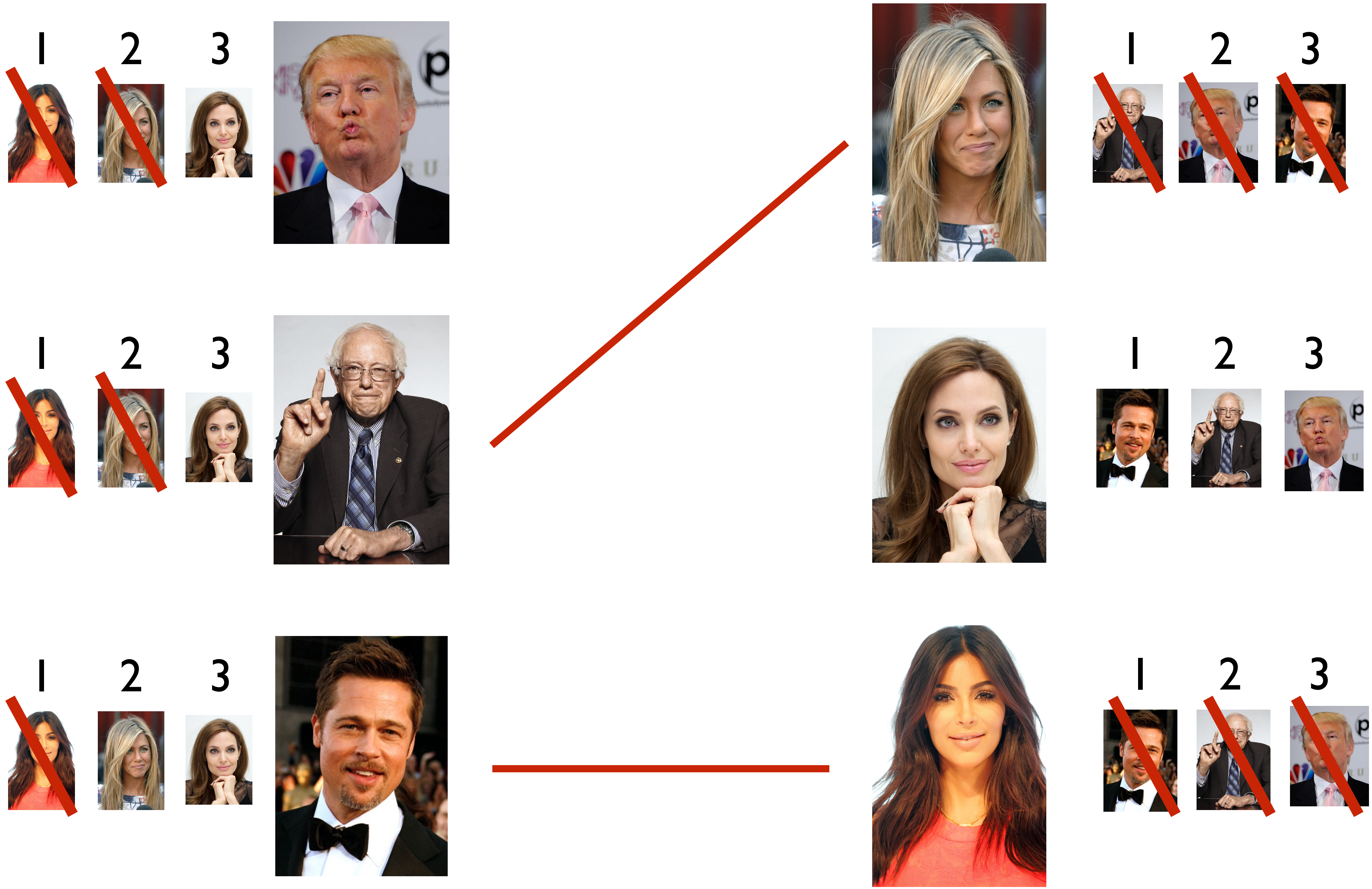
The Gale-Shapley proposal algorithm



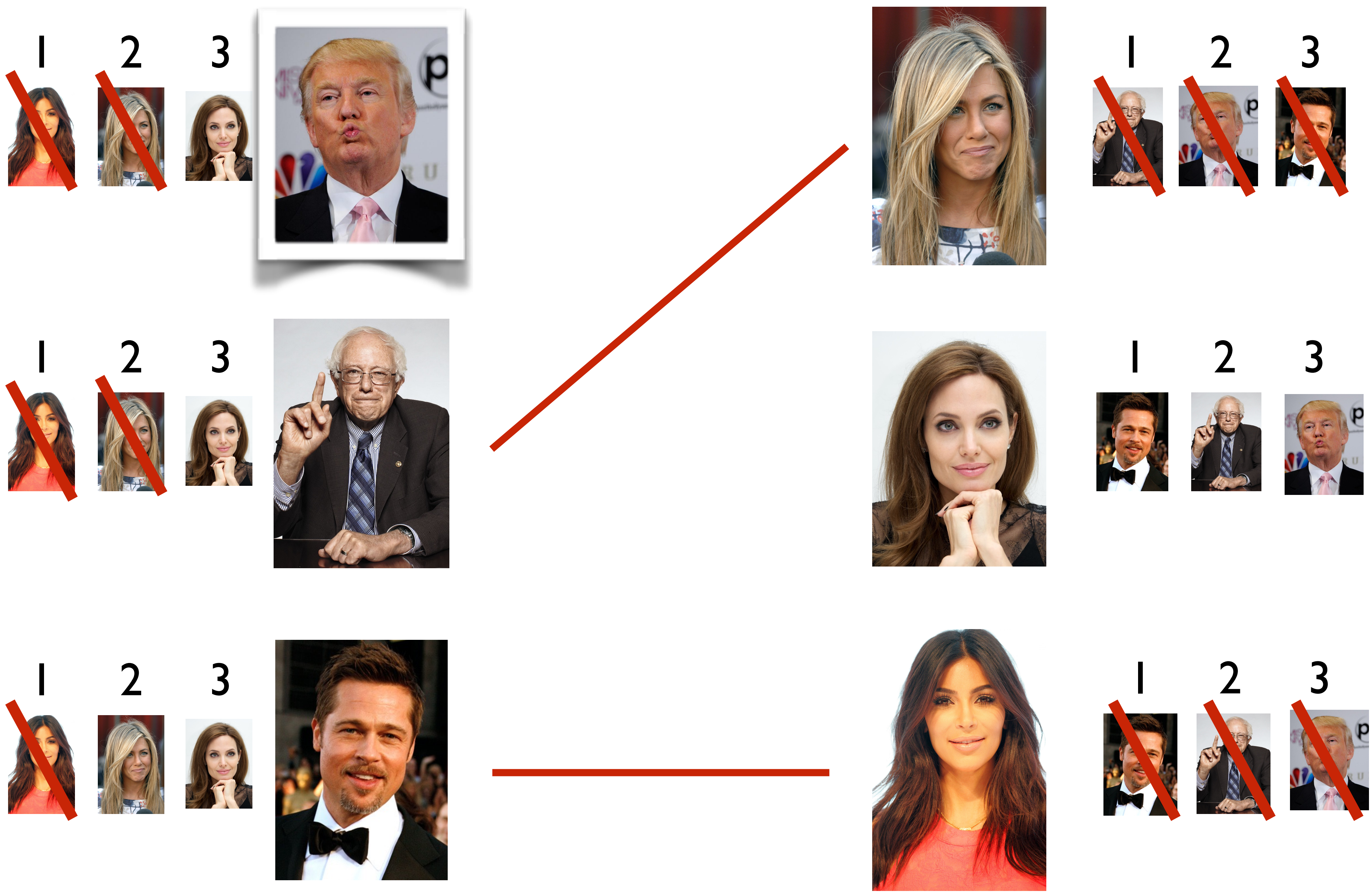
The Gale-Shapley proposal algorithm



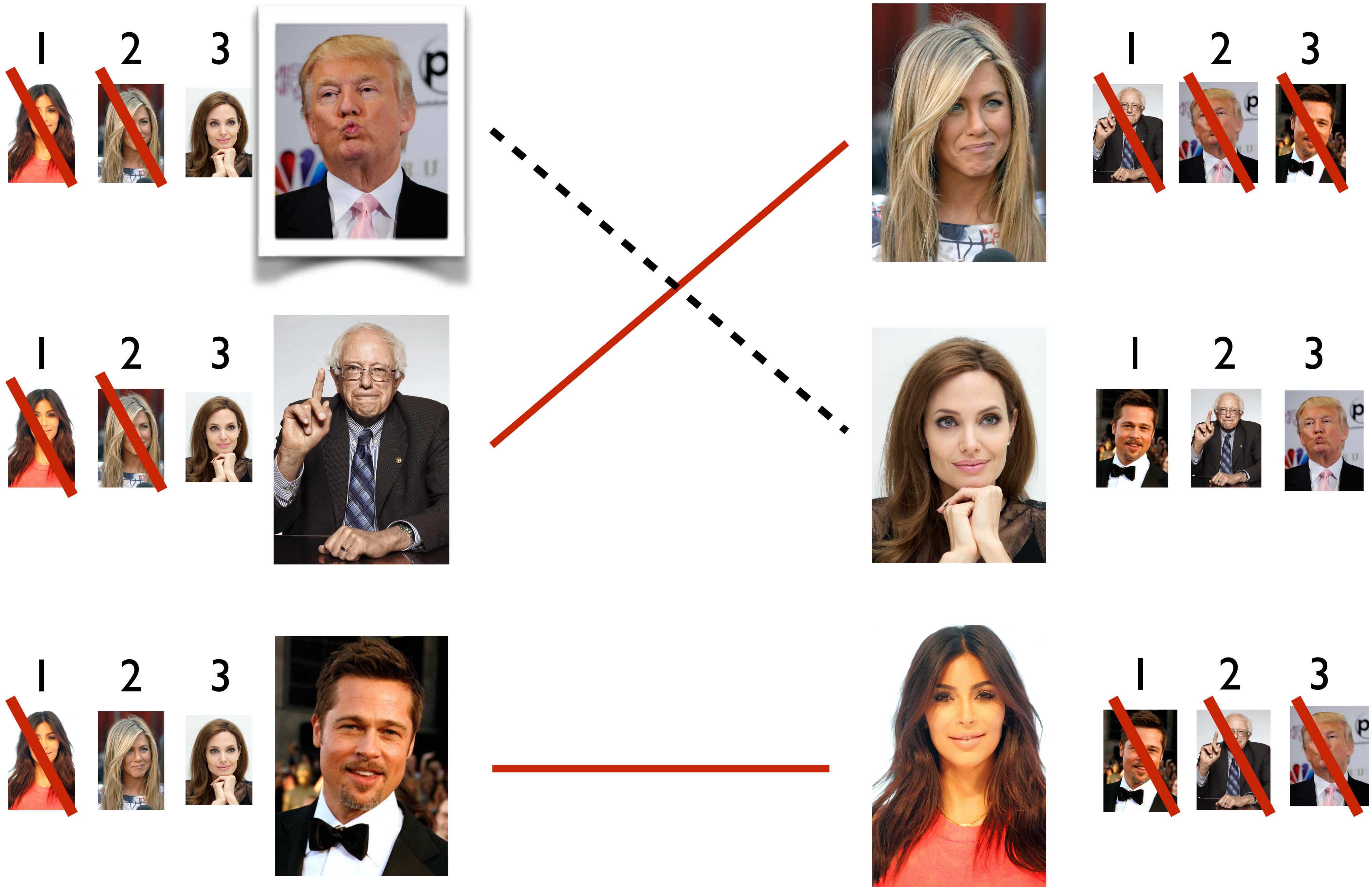
The Gale-Shapley proposal algorithm



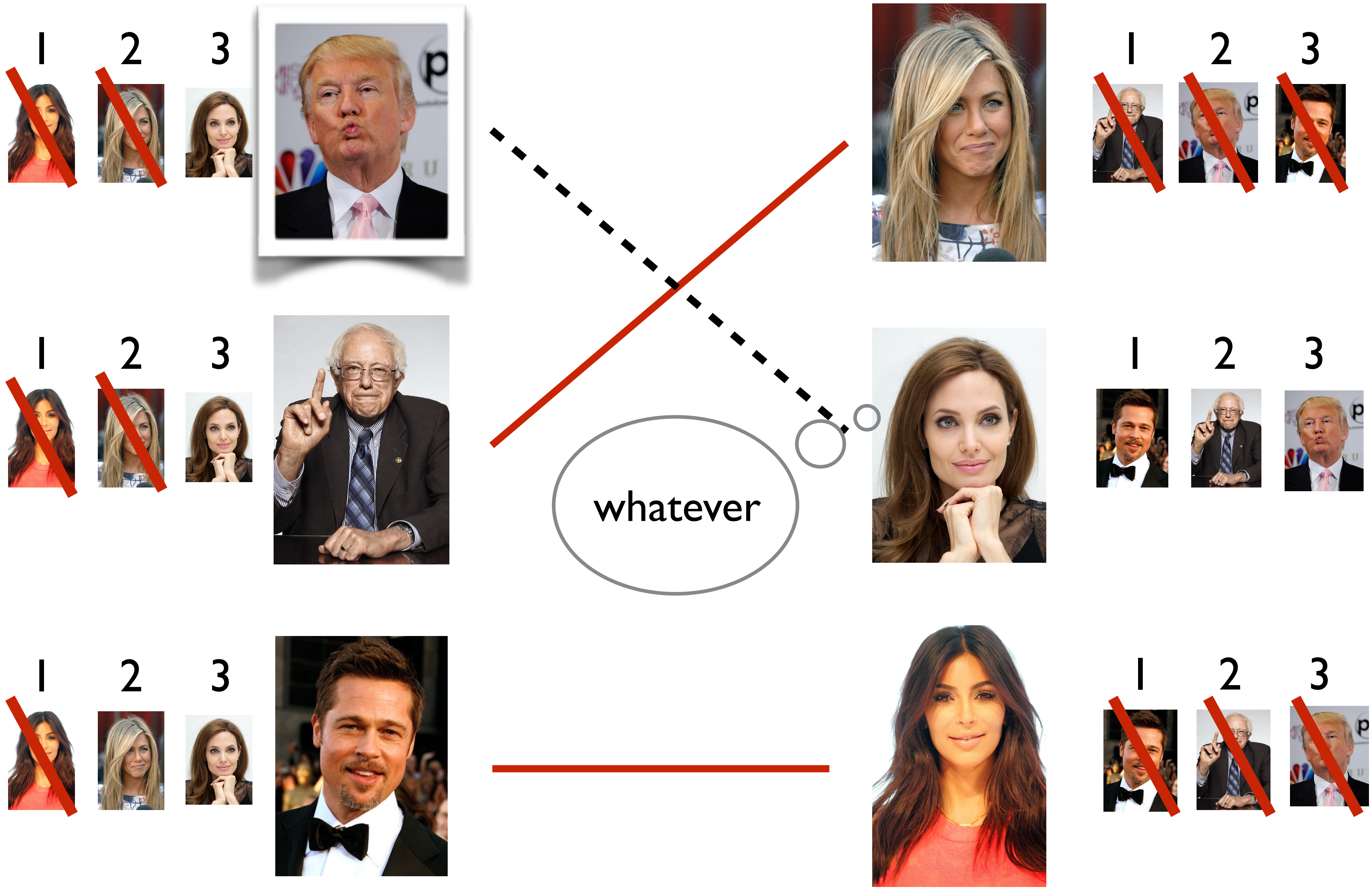
The Gale-Shapley proposal algorithm



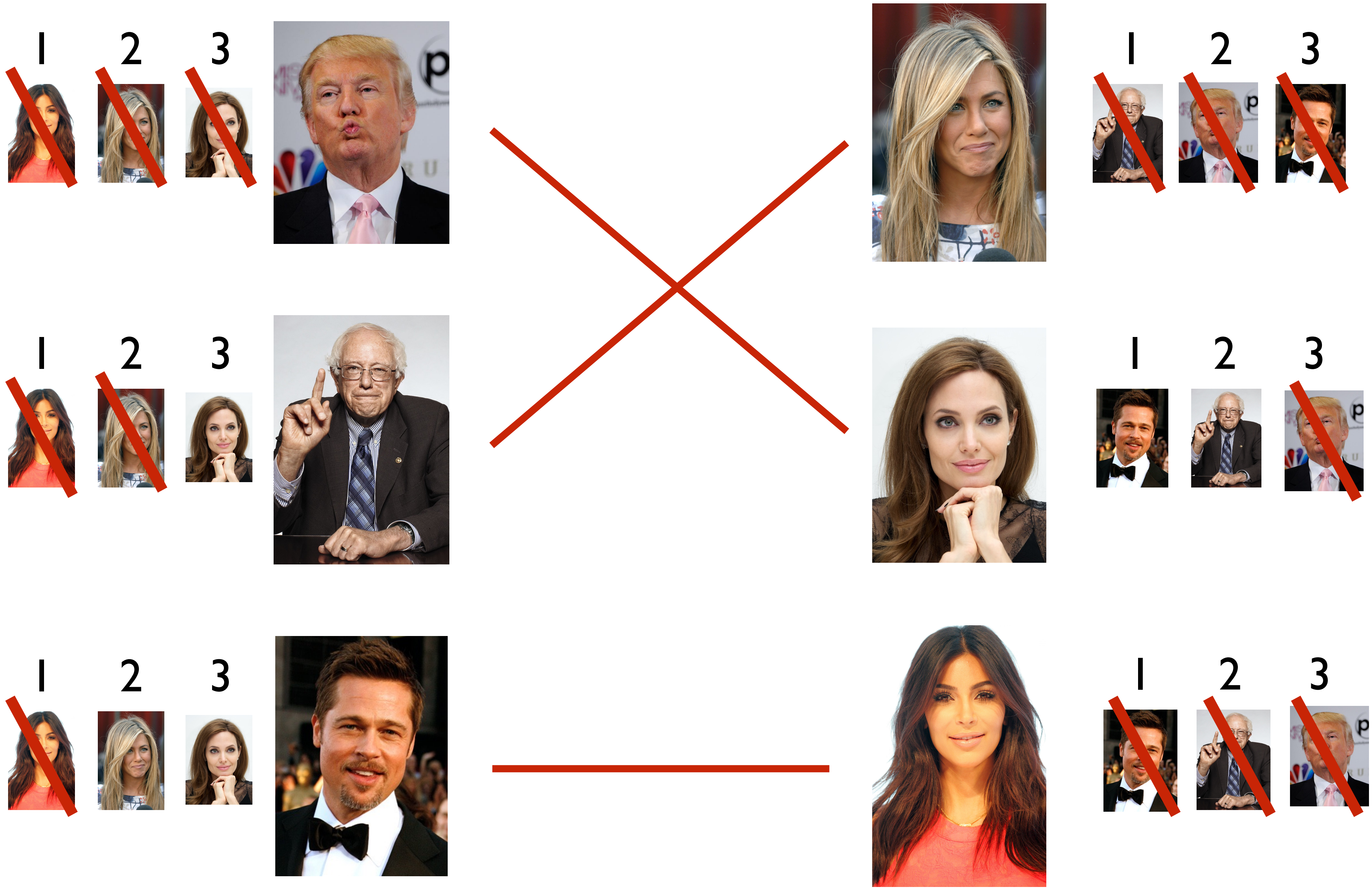
The Gale-Shapley proposal algorithm



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.

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.

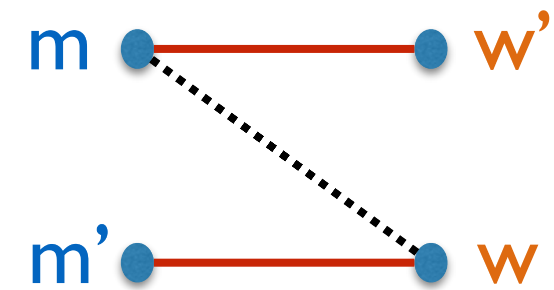
Gale-Shapley algorithm analysis

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

Unstable pair:

(m, w) not matched

but they prefer each other.



<u>Observations</u>:

> A man can only go down in his preference list.

> A woman can only go up in her preference list.

Consider any unmatched (m,w).

Case 1: m never proposed to w

w' must be higher in the preference list of m than w

Case 2: m proposed to w

w rejected m $\implies$ w prefers her current partner $\square$

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

Theorem:

The *Gale-Shapley proposal algorithm* always matches m with its best valid partner.

Theorem:

The *Gale-Shapley proposal algorithm* always matches w with its worst valid partner.

Real-world applications

Alvin Roth

Variants of the Gale-Shapley algorithm is used for:

- matching doctors and hospitals
- matching students to high schools (e.g. in New York)
- matching kidney donors to patients
  > revolutionized the way kidney transplants were handled in the US

  > in 2003, 3436 patients on the waitlist died.

"Throughout the United States nearly 2,000 patients have received kidneys under the system developed on Roth and Shapley's models that would otherwise not have received them."

- Ruthanne Hanto,
Program Manager, Kidney Paired Donation Program,
Organ Procurement and Transplantation Network (OPTN)

Today's Goal:

Save lives.