CS251

Great Ideas in *Theoretical* Computer Science

Stable Matchings

2-Sided Markets

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

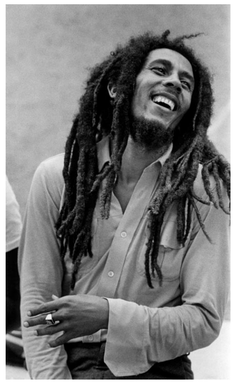
2-Sided Markets

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

-
-
-

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

Bob



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

Other examples:

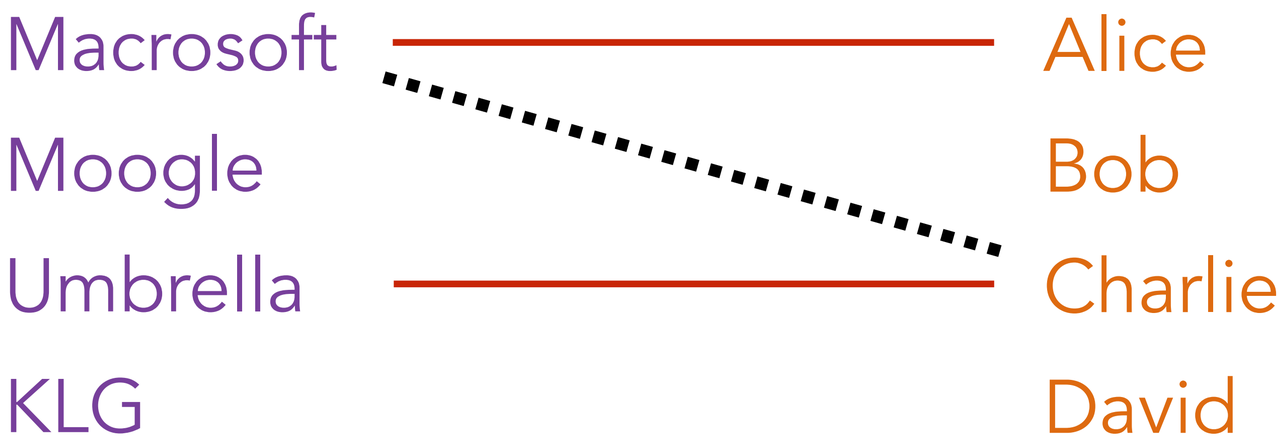
medical residents - hospitals

students - colleges

professors - colleges

Goal: "Good" Centralized Matching System

What can go wrong?



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

But: Macrosoft prefers Charlie over Alice.
Charlie prefers Macrosoft over Umbrella.

How do you solve a problem like this?

1. Formulate the problem

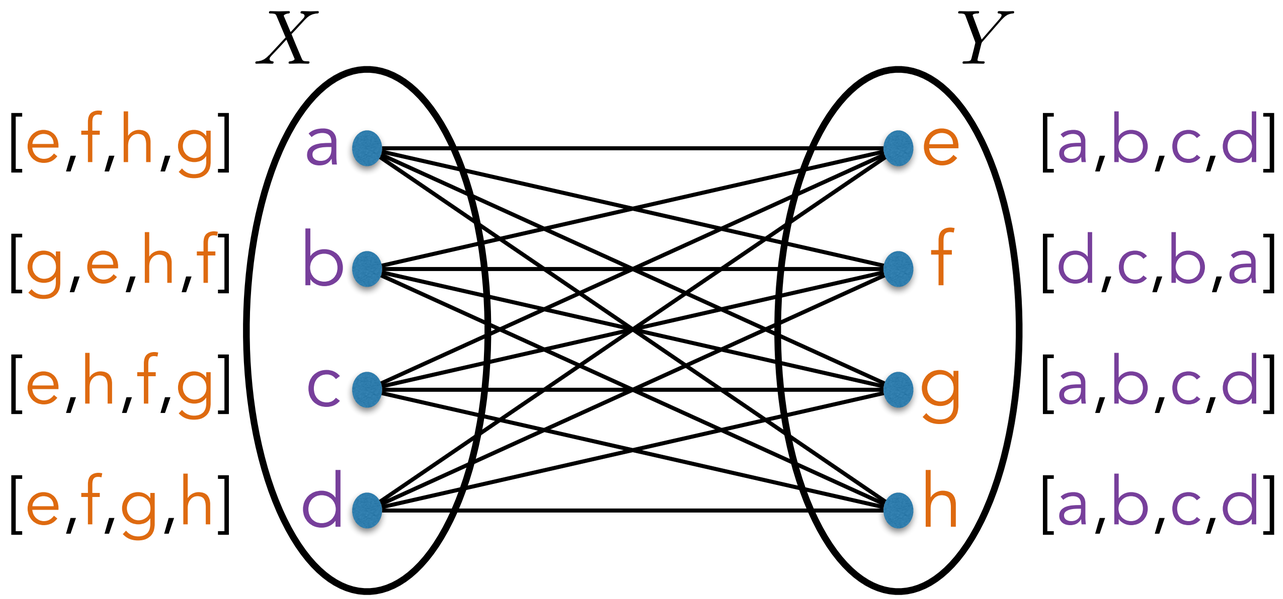
2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

Formalizing the problem

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

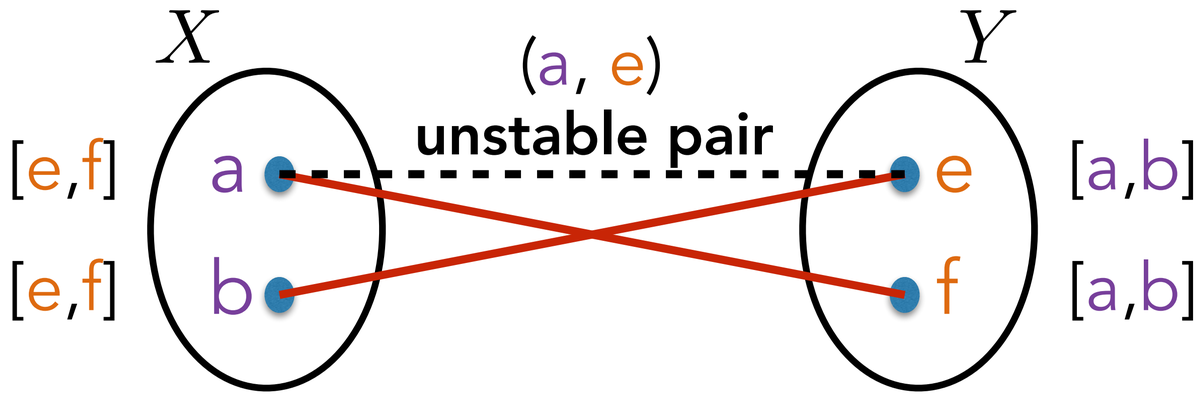


$$|X| = |Y| = n$$

Goal: Find a **stable matching**.

Formalizing the problem

What is a **stable matching**?



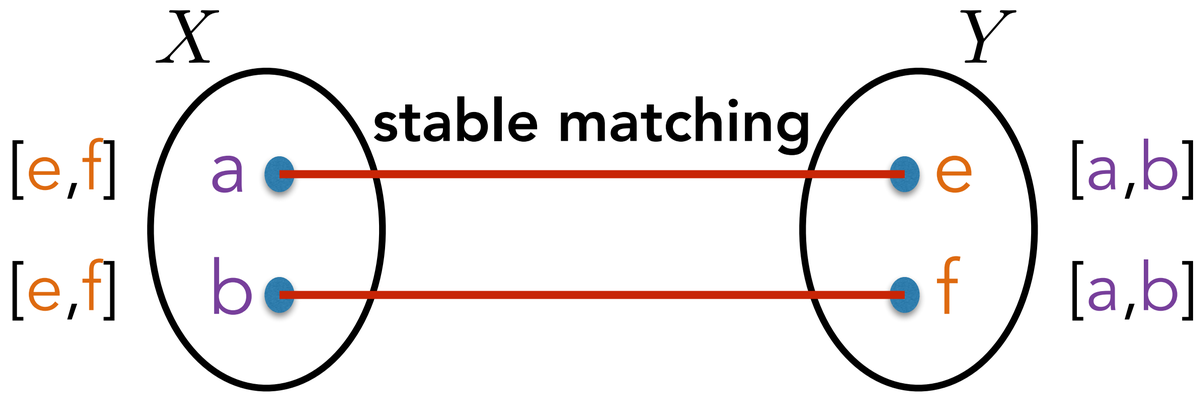
1. It is a perfect matching.

2. Does not contain an **unstable pair**.

(an unmatched pair (x, y)
who <u>both</u> prefer each other over their current partners.)

Formalizing the problem

What is a **stable matching**?



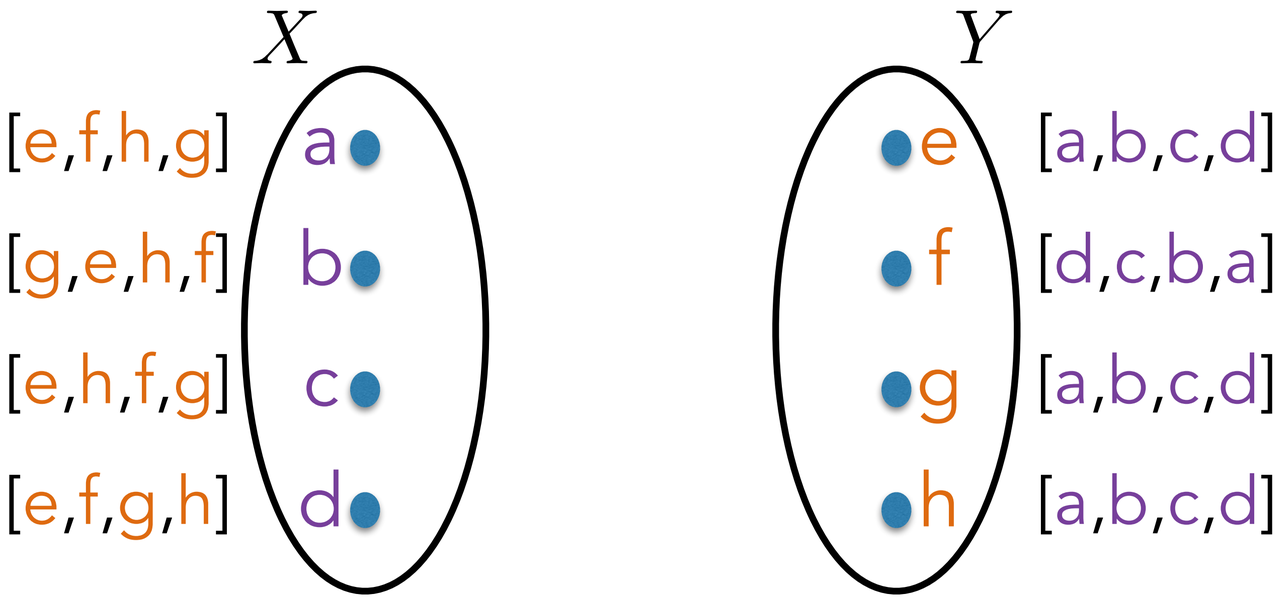
1. It is a perfect matching.

2. Does not contain an **unstable pair**.

(an unmatched pair (x, y)
who <u>both</u> 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**.

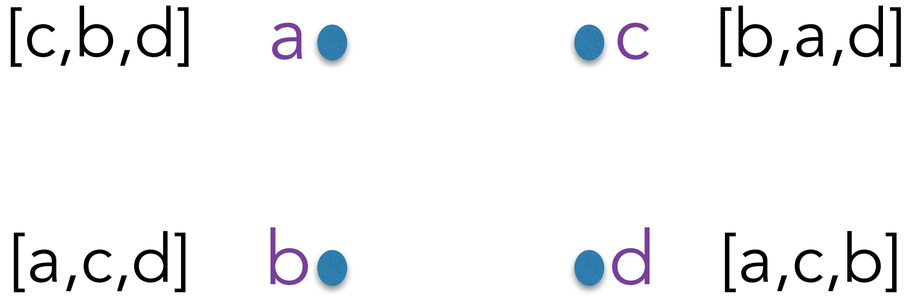


Goal: Find a **stable matching**.

? Is it always guaranteed to exist?

A variant

Roommate Problem (non-bipartite version)



Is there a **stable matching**?

(perfect matching with no unstable pair)

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

How do you solve a problem like this?

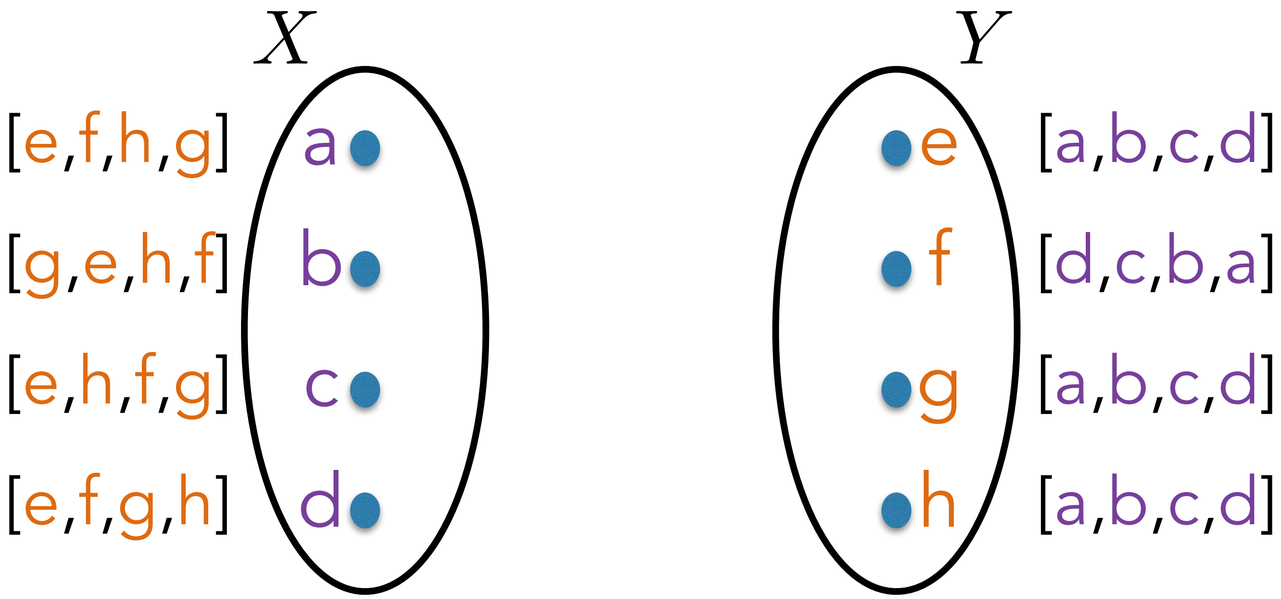
1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

Stable matching: Is there a trivial algorithm?



Trivial algorithm:

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

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

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

Gale-Shapley Proposal Algorithm

COLLEGE ADMISSIONS AND THE STABILITY OF MARRIAGE

D. GALE* AND L. S. SHAPLEY, Brown University and the RAND Corporation

1. Introduction. The problem with which we shall be concerned relates to the following typical situation: A college is considering a set of n applicants of which it can admit a quota of only q. Having evaluated their qualifications, the admissions office must decide which ones to admit. The procedure of offering admission only to the q best-qualified applicants will not generally be satisfactory, for it cannot be assumed that all who are offered admission will accept. Accordingly, in order for a college to receive q acceptances, it will generally have to offer to admit more than q applicants. The problem of determining how many and which ones to admit requires some rather involved guesswork. It may not be known (a) whether a given applicant has also applied elsewhere; if this is known it may not be known (b) how he ranks the colleges to which he has applied; even if this is known it will not be known (c) which of the other colleges will offer to admit him. A result of all this uncertainty is that colleges can expect only that the entering class will come reasonably close in numbers to the desired quota, and be reasonably close to the attainable optimum in quality.

The usual admissions procedure presents problems for the applicants as well as the colleges. An applicant who is asked to list in his application all other colleges applied for in order of preference may feel, perhaps not without reason, that by telling a college it is, say, his third choice he will be hurting his chances of being admitted.

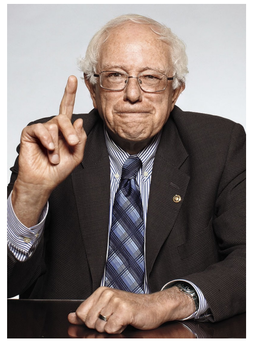
X

1 2 3




1 2 3



1 2 3




Y

1 2 3



1 2 3



1 2 3



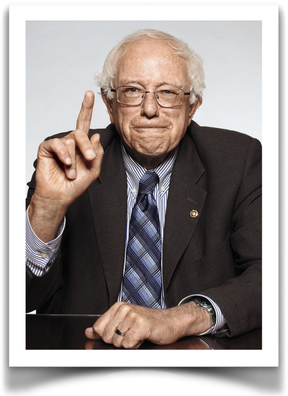
X

1 2 3




1 2 3



1 2 3




Y

1 2 3



1 2 3



1 2 3



X

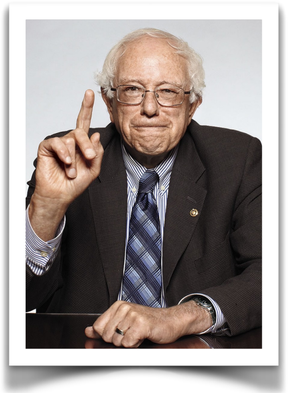
Y

1 2 3



1 2 3





1 2 3



1 2 3



1 2 3



1 2 3



X

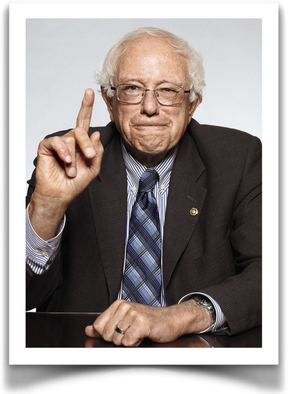
Y

1 2 3




1 2 3



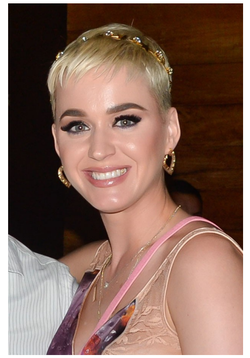



1 2 3




1 2 3





1 2 3



1 2 3



X

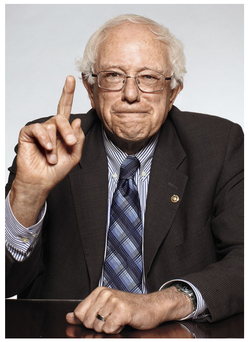
Y

1 2 3




1 2 3



1 2 3




1 2 3



1 2 3



1 2 3



X

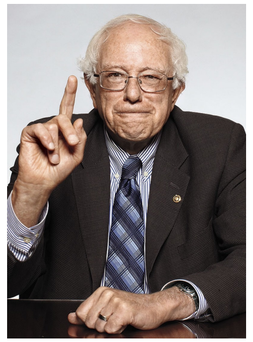
Y

1 2 3




1 2 3



1 2 3




1 2 3

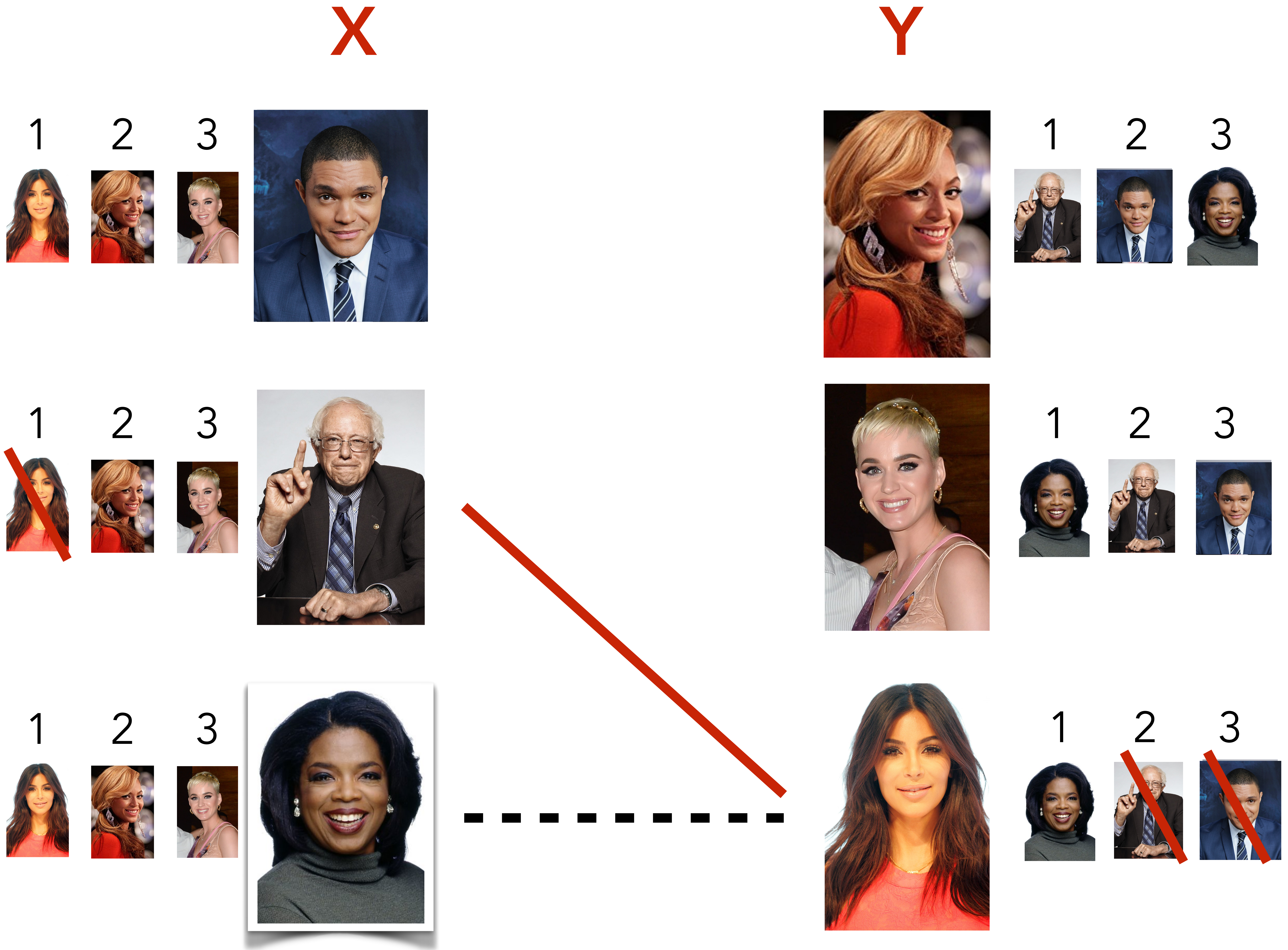


1 2 3



1 2 3

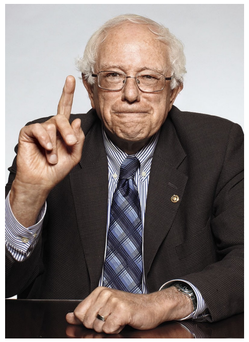



X
Y
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3
1
2
3

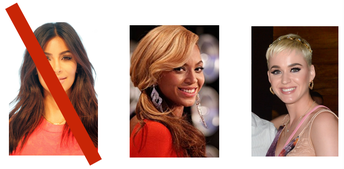
X

1 2 3

1 2 3

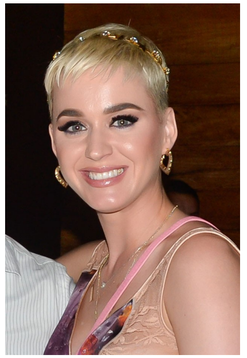


1 2 3



Y

1 2 3

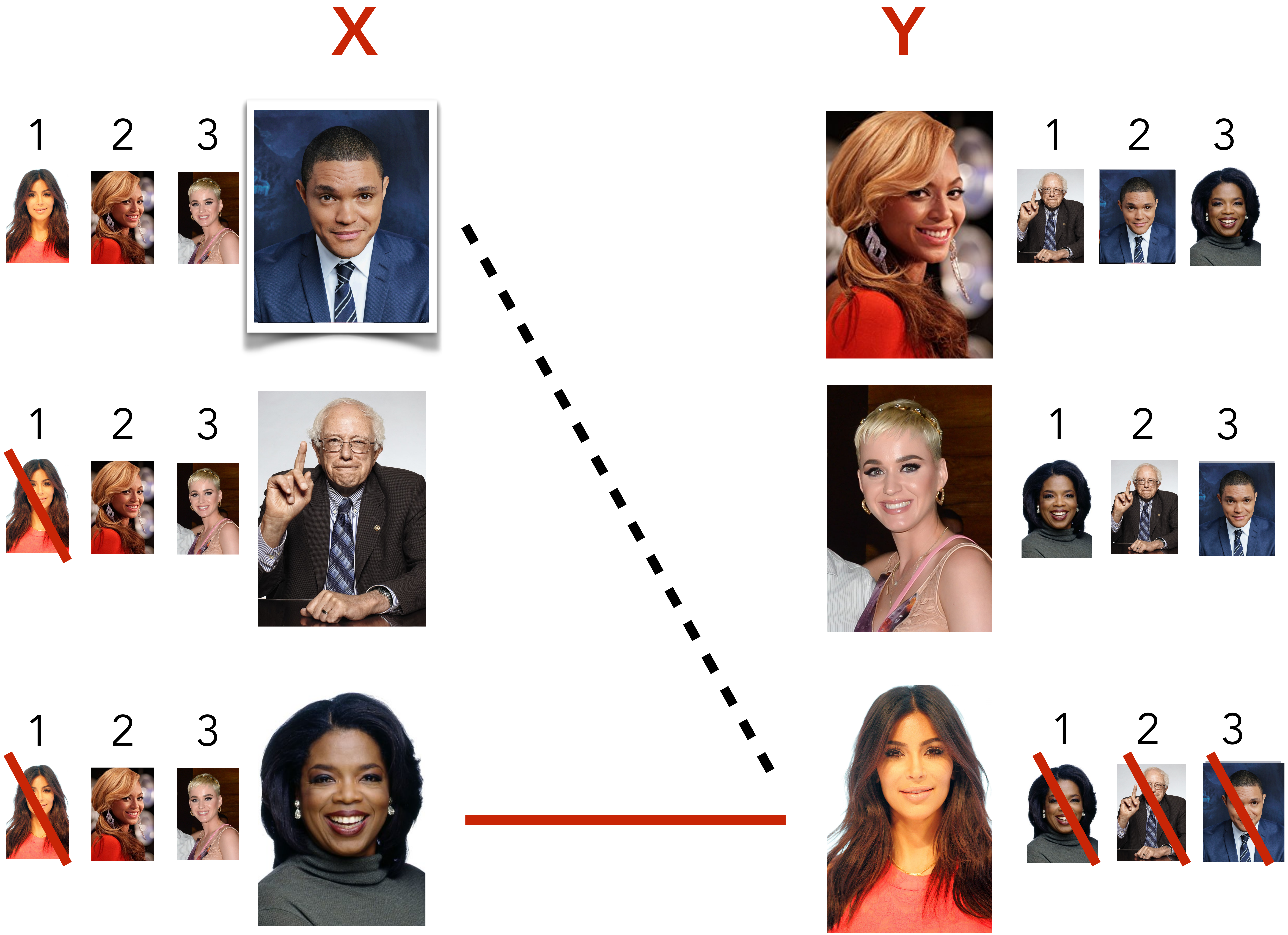


1 2 3

1 2 3

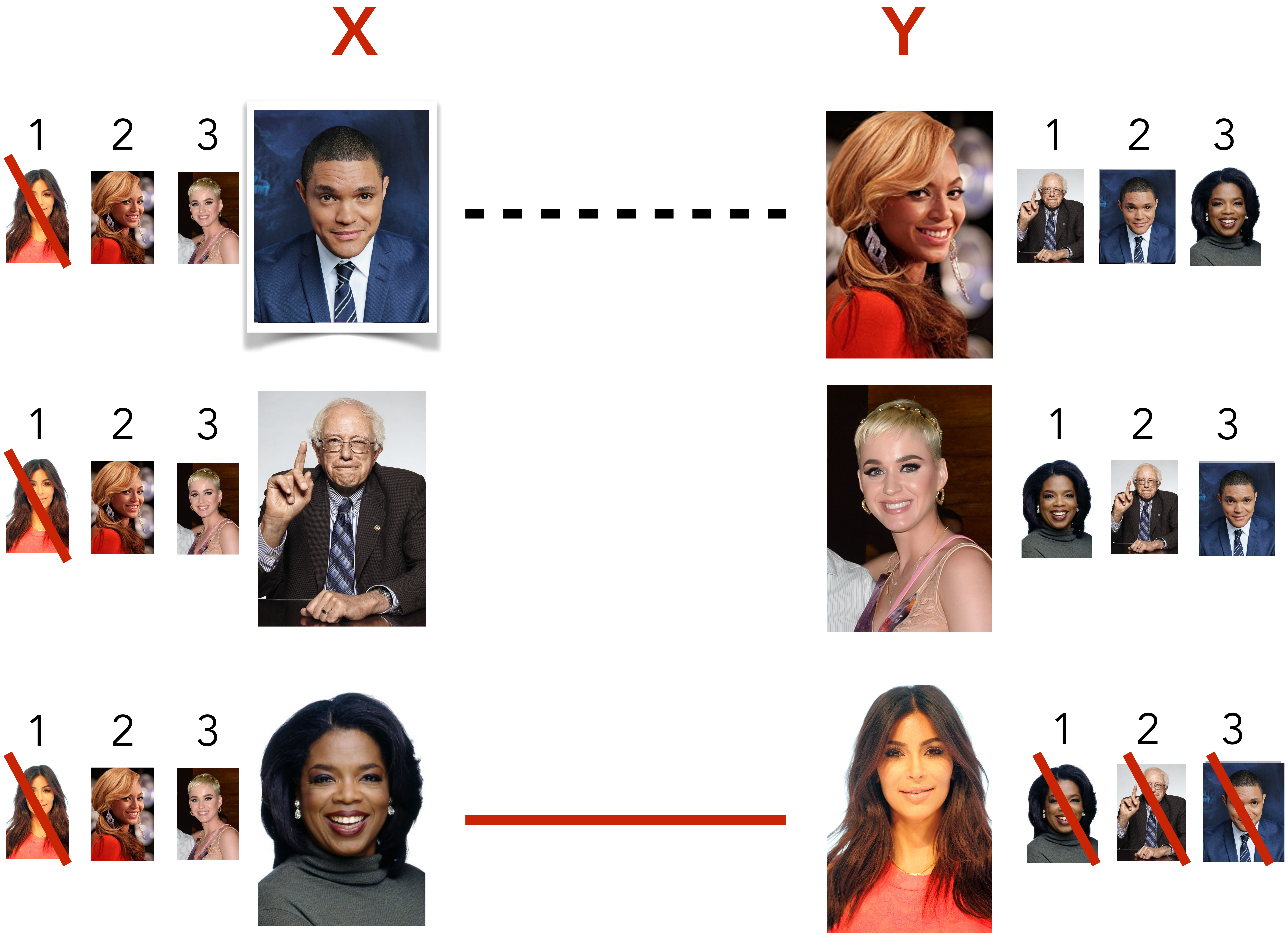

X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

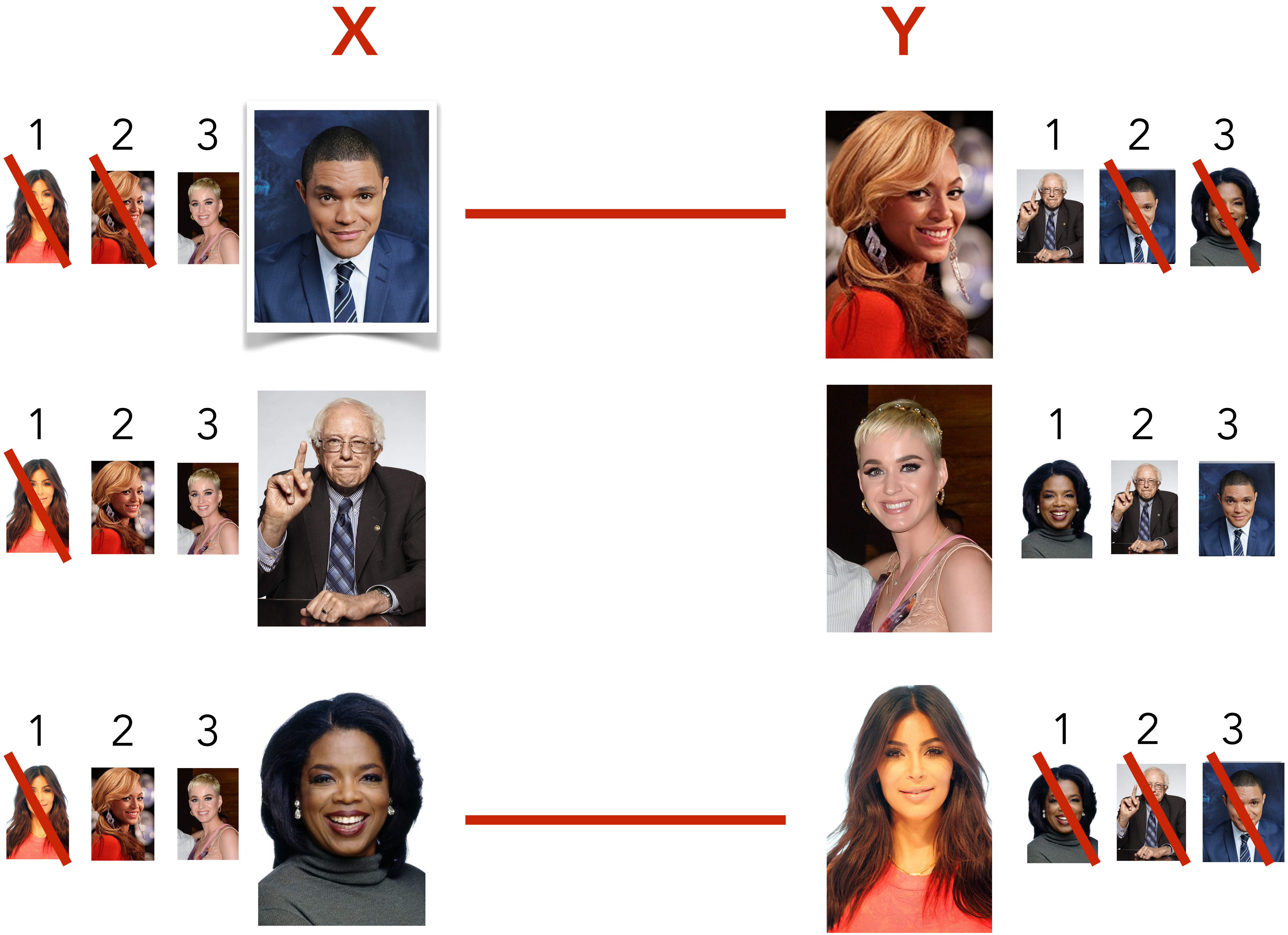

X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

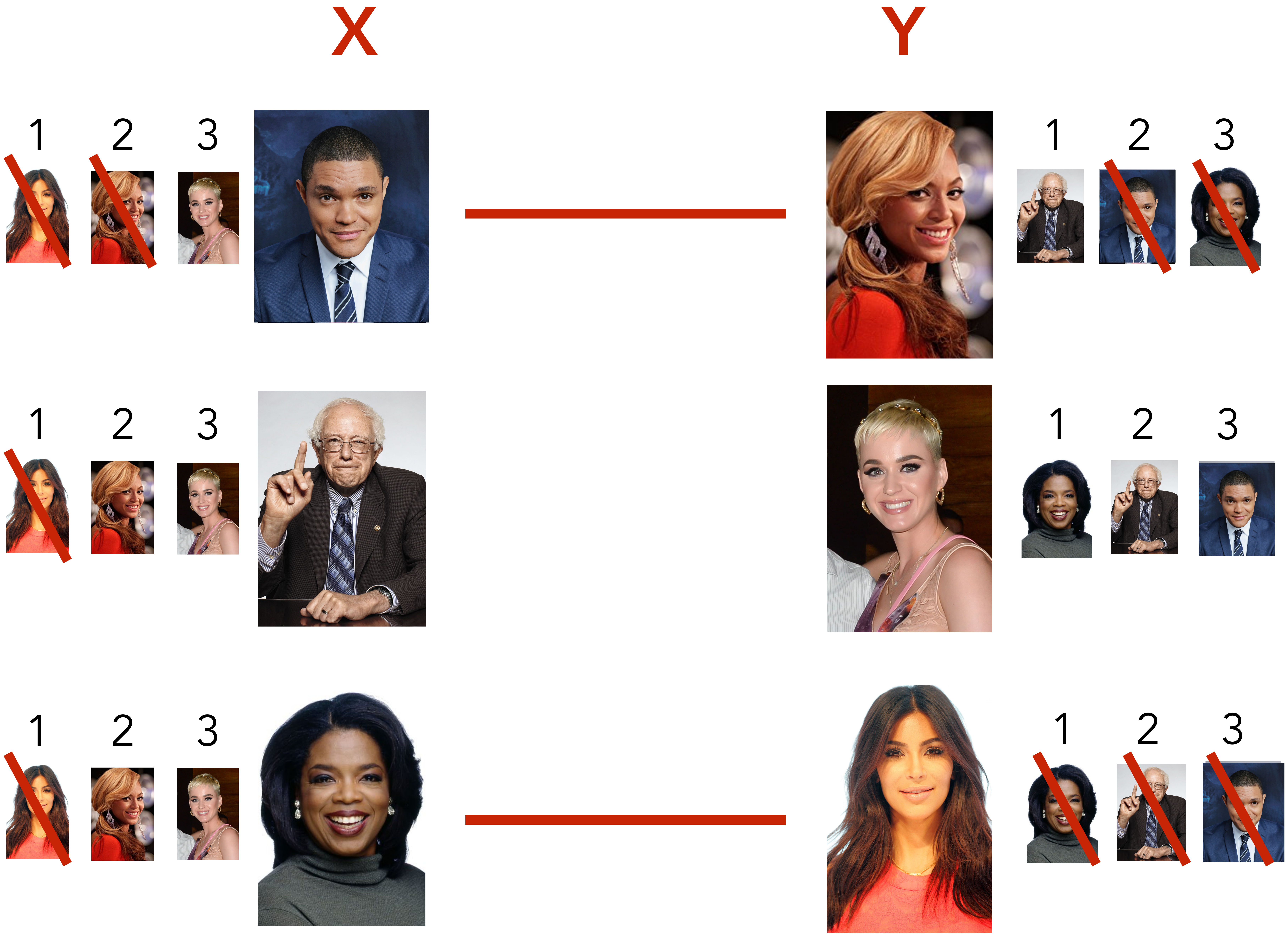

X
Y
1
2
3

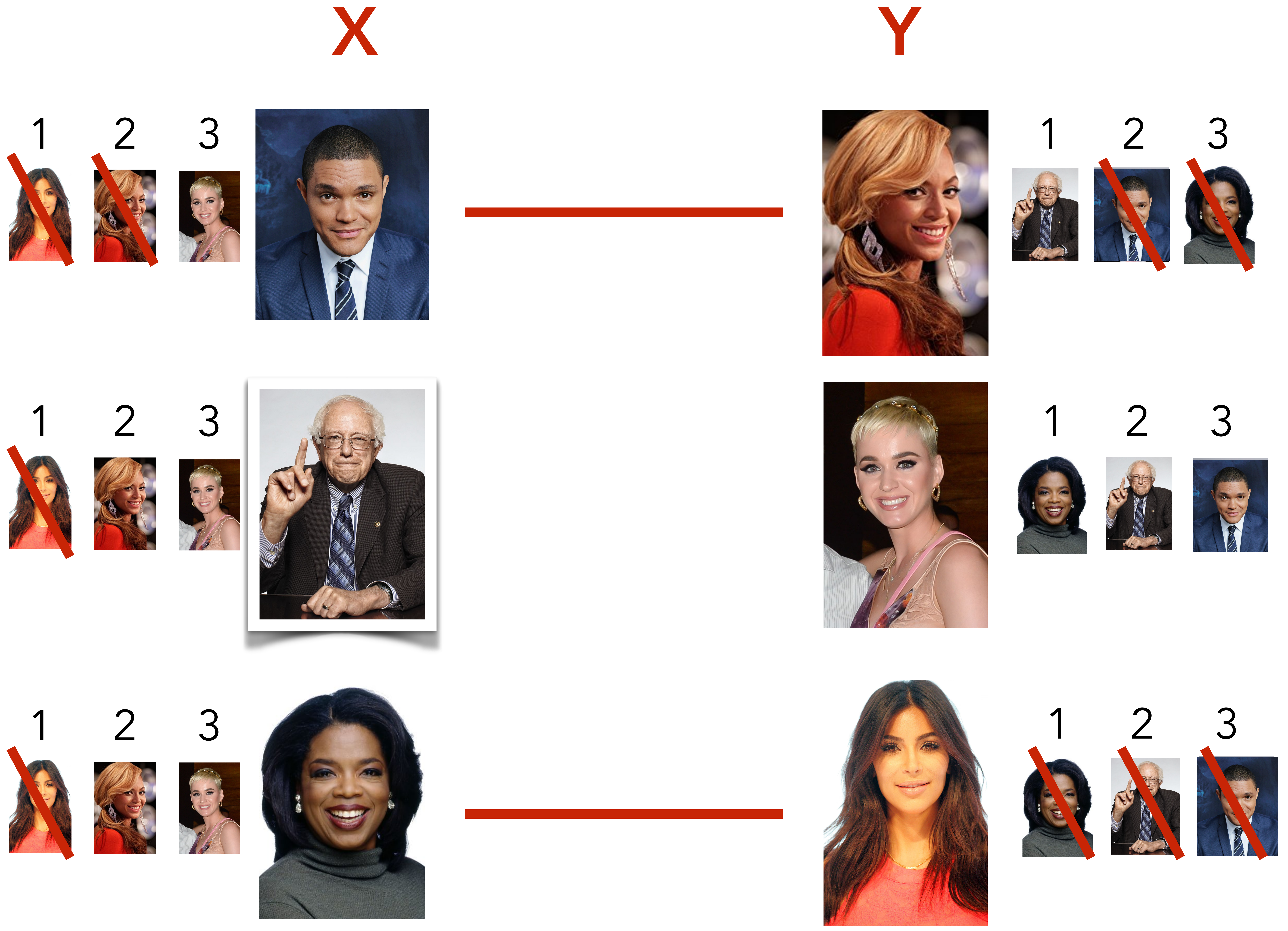

X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3


X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3


X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3


X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3


X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

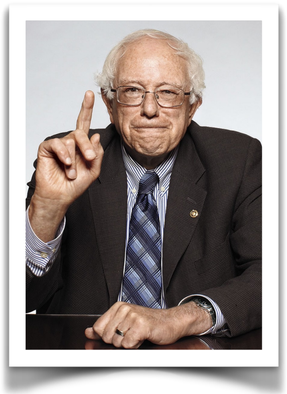

X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

X

Y

1 2 3

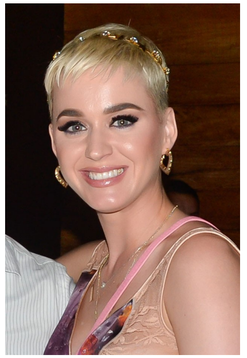
1 2 3



1 2 3

1 2 3

1 2 3

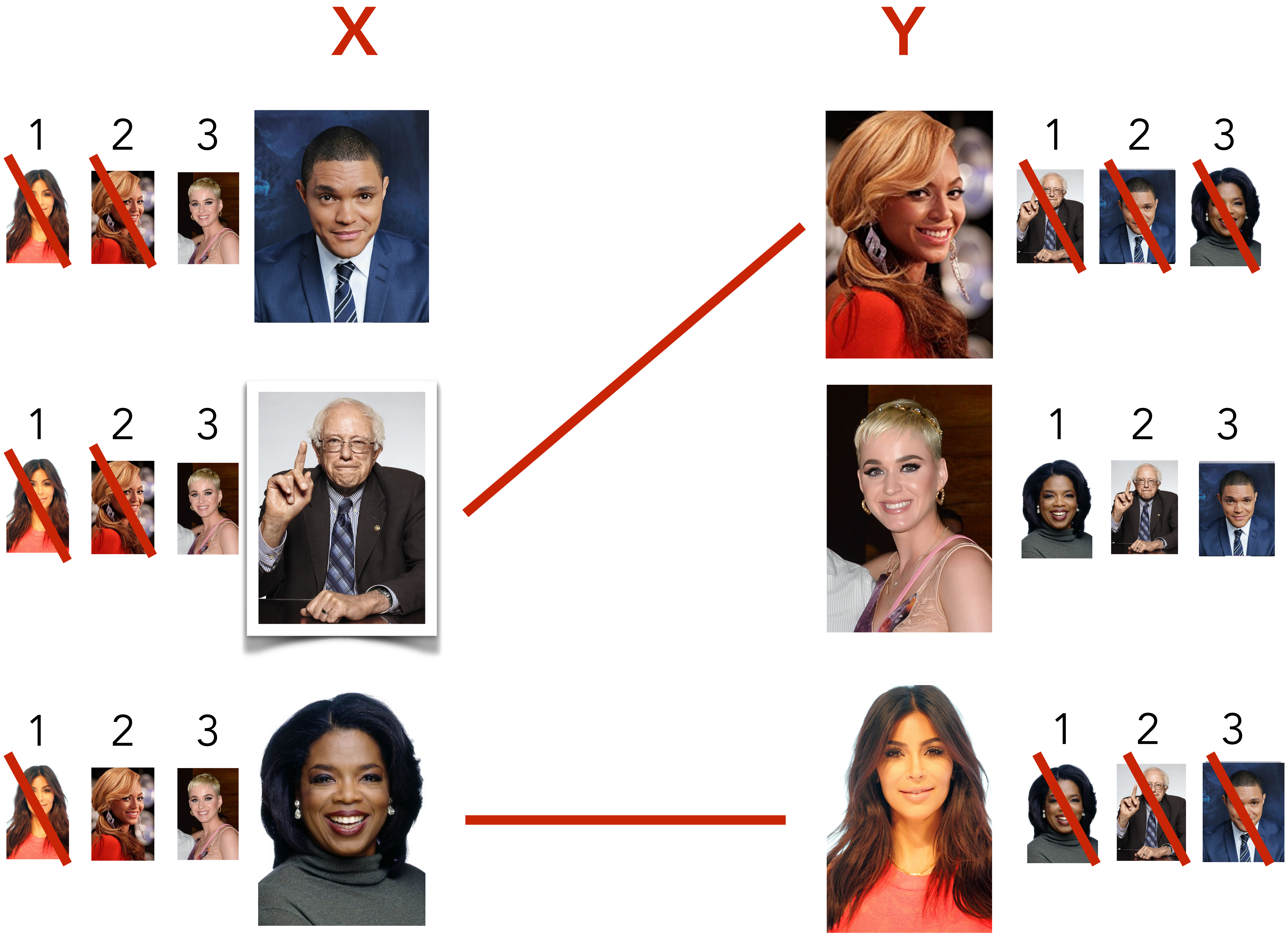


1 2 3

1 2 3

1 2 3

1 2 3


X
Y
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3
1 2 3

X

Y

1 2 3



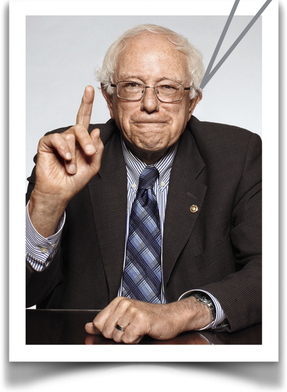
#FeelTheBern
Trevor

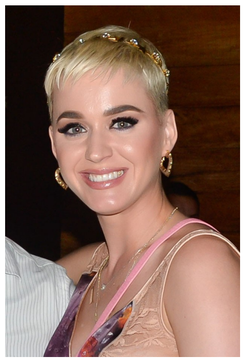
1 2 3



1 2 3







1 2 3



1 2 3



1 2 3



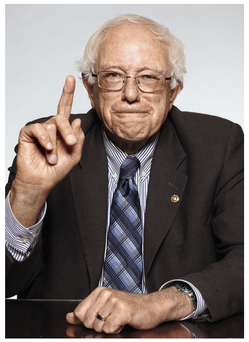
X

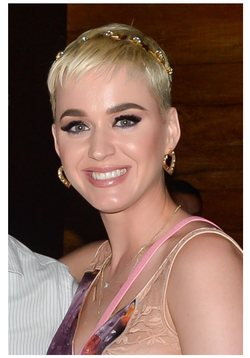
Y

1 2 3

1 2 3

1 2 3





1 2 3

1 2 3

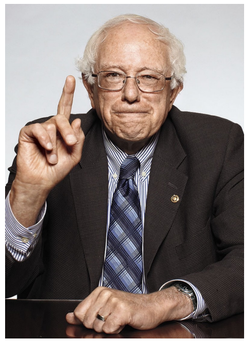
1 2 3

X

Y

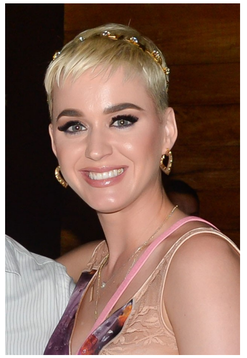
1 2 3

1 2 3



1 2 3

1 2 3

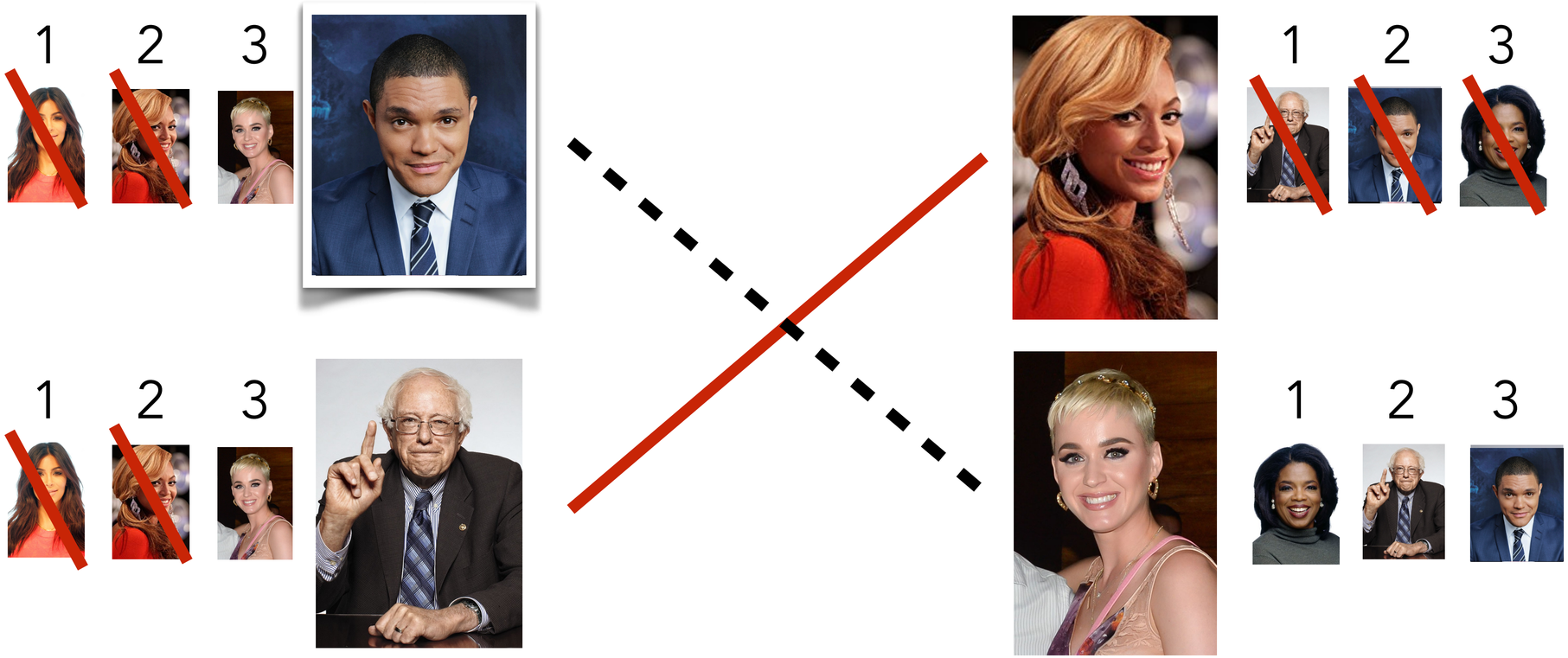


1 2 3

1 2 3

X

Y

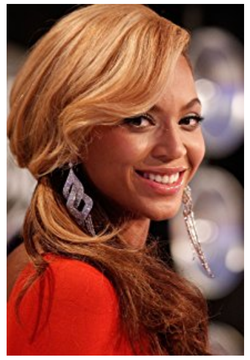


X

Y

1 2 3



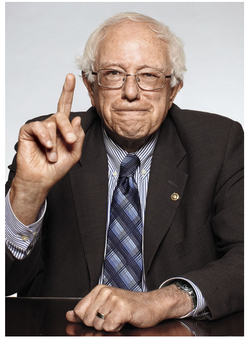



1 2 3

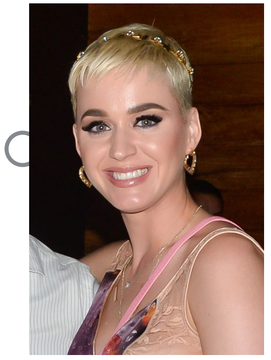


1 2 3



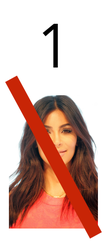
whatever



1 2 3



1 2 3




1 2 3



X

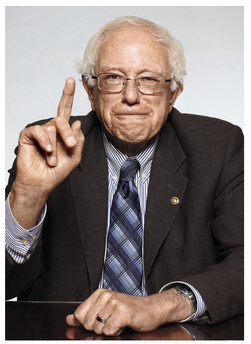
Y

1 2 3




1 2 3



1 2 3




1 2 3



1 2 3



1 2 3



The Gale-Shapley proposal algorithm

While there is an unmatched company **x**:

- Let **y** be the highest ranked student in **x**'s list to whom **x** has not proposed yet.
- If **y** is unmatched, or **y** prefers **x** over her current match:
  - Match **x** and **y**.
    (The previous match of **y** 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 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.

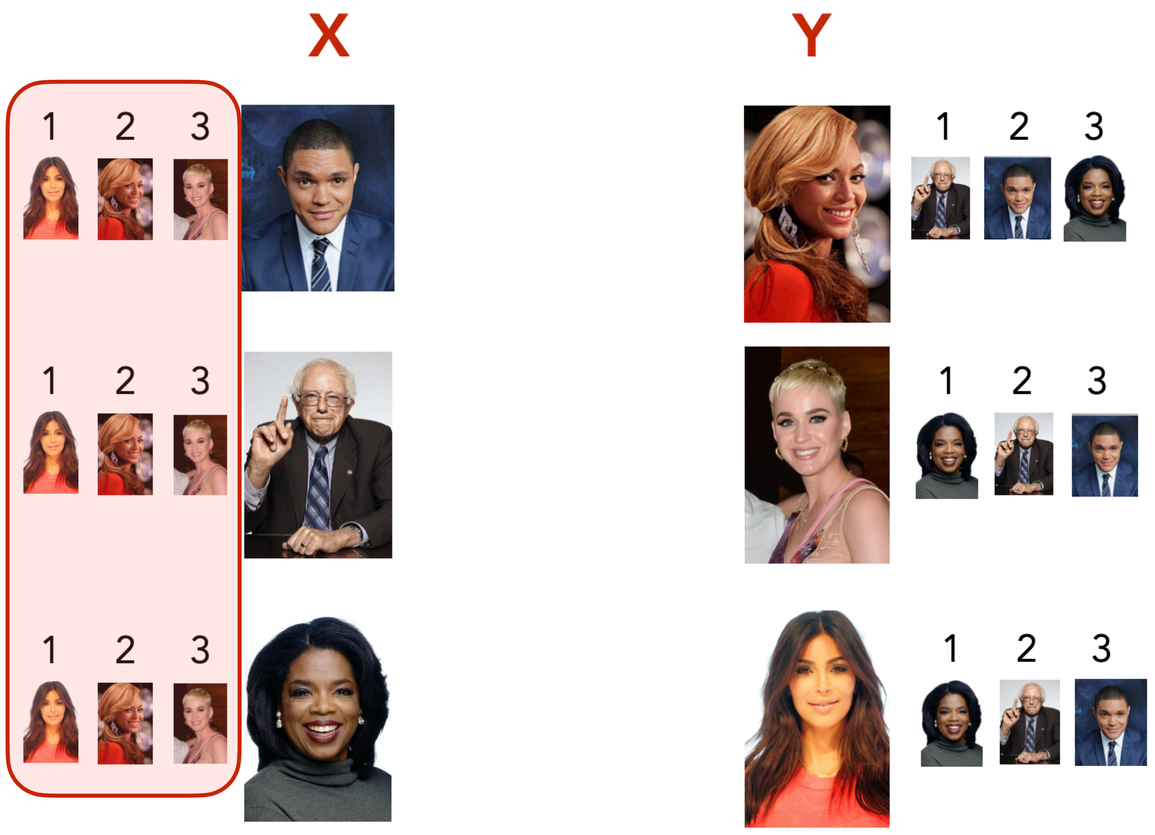
Gale-Shapley algorithm analysis

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

iterations = # proposals

No company proposes to a student more than once.

proposals ≤ n^2



Gale-Shapley algorithm analysis

2. The algorithm terminates with a perfect matching.

AFSOC we don't have a perfect matching:

A company **x*** is not matched

$\Longrightarrow$ All students must be matched

$\Longrightarrow$ All companies must be matched.

Contradiction

2nd implication:

True since # companies **=** # students.

Gale-Shapley algorithm analysis

2. The algorithm terminates with a perfect matching.

AFSOC we don't have a perfect matching:

A company **x*** is not matched

$\Longrightarrow$ All students must be matched

$\Longrightarrow$ All companies must be matched.

Contradiction

1st implication:

Observe: Once a student is matched, she stays matched.

A company **x*** got rejected by every student:

case1: student was already matched, or

case2: student got a better offer and upgraded

Either way, student was matched at some point.

Gale-Shapley algorithm analysis

3. The matching has no unstable pairs.

"Improvement" Principle:

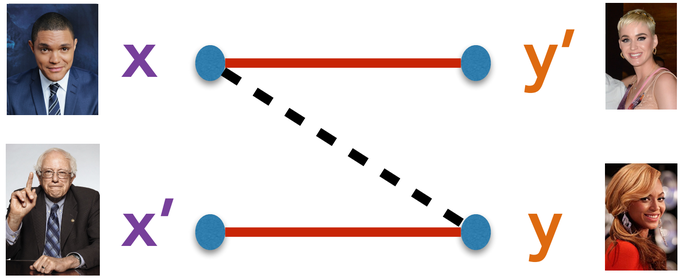
(i) A company can only ***go down*** in its preference list.

(ii) A student can only ***go up*** in her preference list.

Unstable pair:

(x, y) unmatched

but they **both** prefer each other.



Consider any unmatched (x, y). WTS: it cannot be unstable.

Case 1: x never proposed to y

x proposed to y′ $\Longrightarrow$ by **(i)**, x prefers y′ over y

Case 2: x proposed to y

y rejected x $\Longrightarrow$ by **(ii)**, y prefers x′ over x

■

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

How do you solve a problem like this?

1. Formulate the problem

2. **Ask**: Is there a trivial algorithm? Find and analyze.

3. **Ask**: Is there a better algorithm? Find and analyze.

4. **Reflect**: Upshots? Downsides? Improvements?

Further questions

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

Does the order of how we pick companies matter?

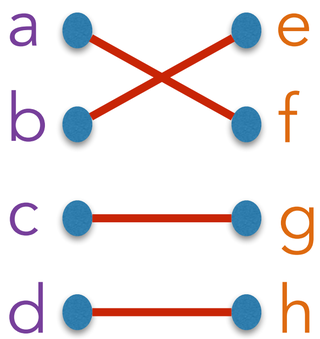
Would it lead to different matchings?

Is the algorithm "fair"?

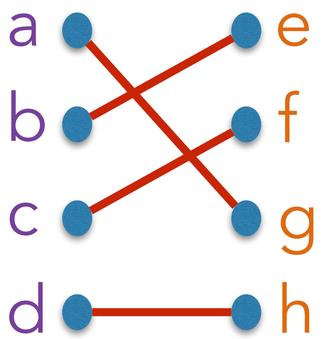
Does it favor companies or students or neither?

Further questions

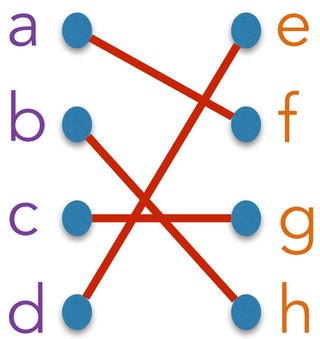
Company **x** and student **y** are **valid partners**
if there is a stable matching in which they are matched.



stable matching 1



stable matching 2



stable matching 3

best(**x**) = highest ranked valid partner of **x**

Further questions

Theorem: The Gale-Shapley algorithm returns $\{(x, \textbf{best}(x)) : x \in X\}$.

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

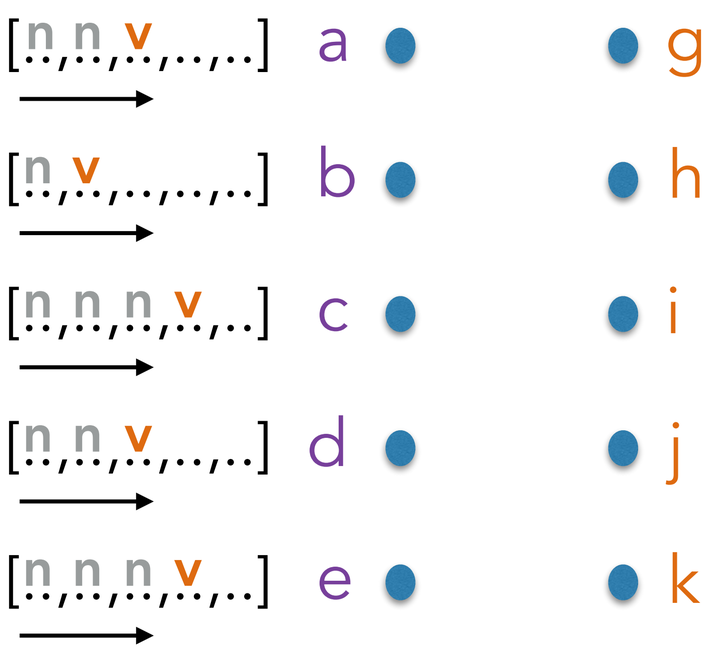
The order in which companies propose doesn't matter.

Proof

AFSOC, Gale-Shapley does not match **x** with **best**(**x**).

Gale-Shapley Algorithm

n = non-valid partner v = valid partner



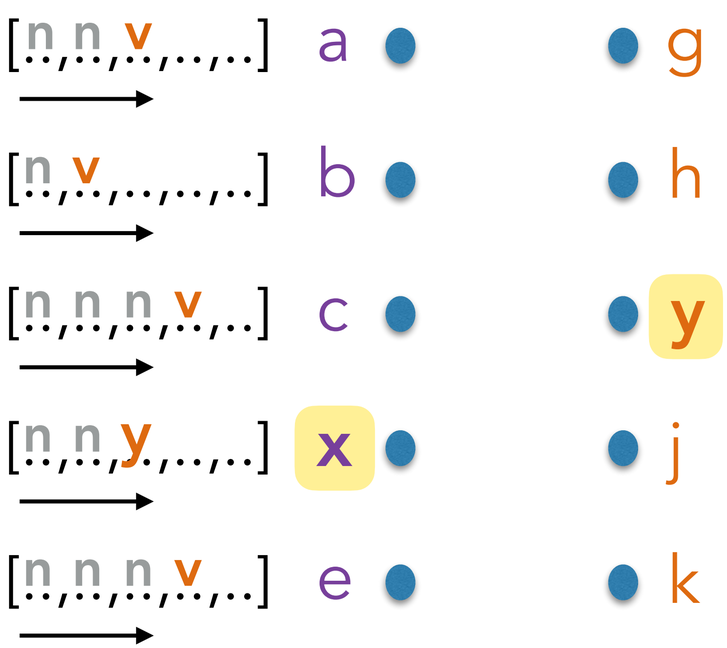
Consider the **first time** an **x** gets rejected by a valid partner **y**.

Proof

AFSOC, Gale-Shapley does not match **x** with **best**(**x**).

Gale-Shapley Algorithm

n = non-valid partner v = valid partner



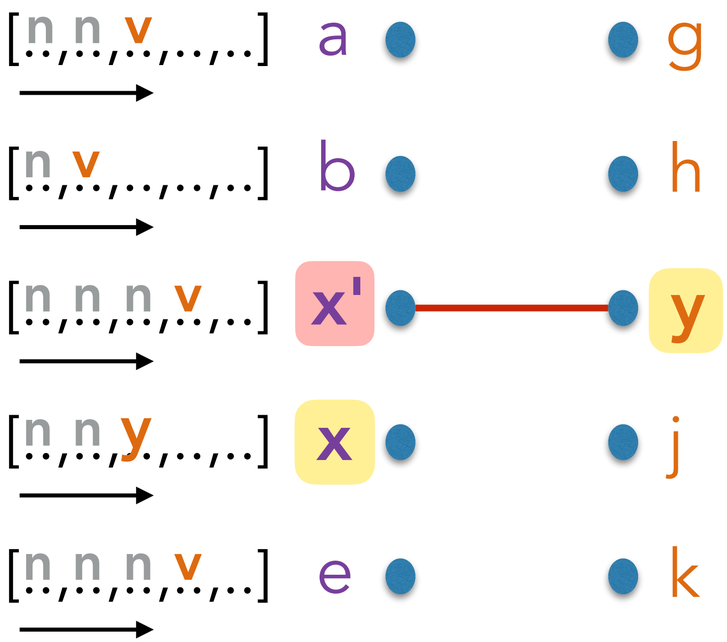
Consider the **first time** an **x** gets rejected by a valid partner **y**.

Proof

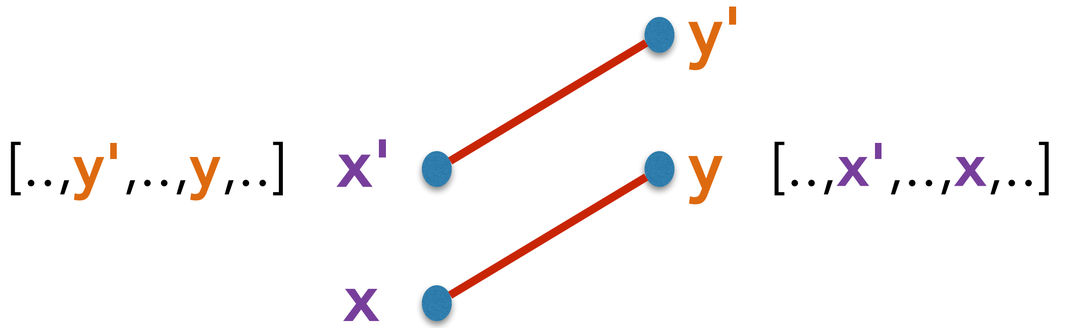
AFSOC, Gale-Shapley does not match **x** with **best(x)**.

Gale-Shapley Algorithm

n = non-valid partner v = valid partner



Another Stable Matching



Consider the **first time** an **x** gets rejected by a valid partner **y**.

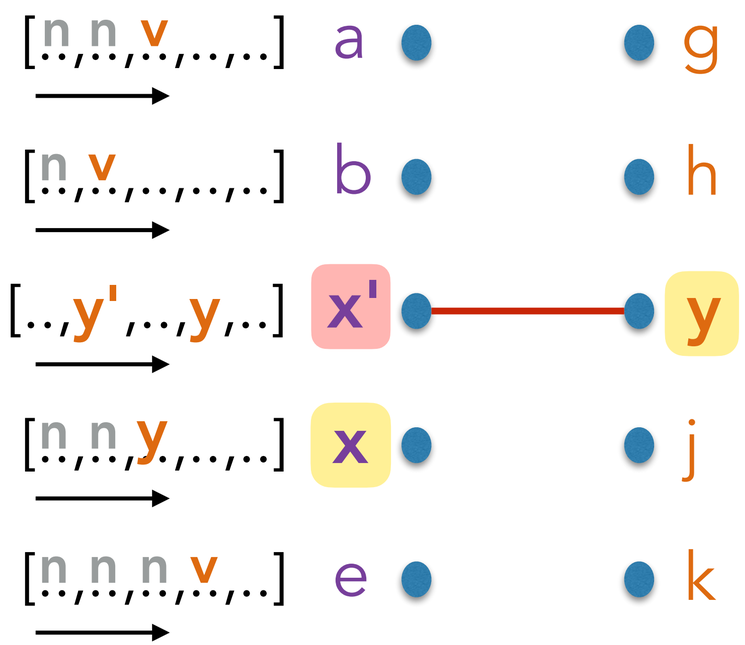
Suppose **x'** is the reason for the rejection.

Proof

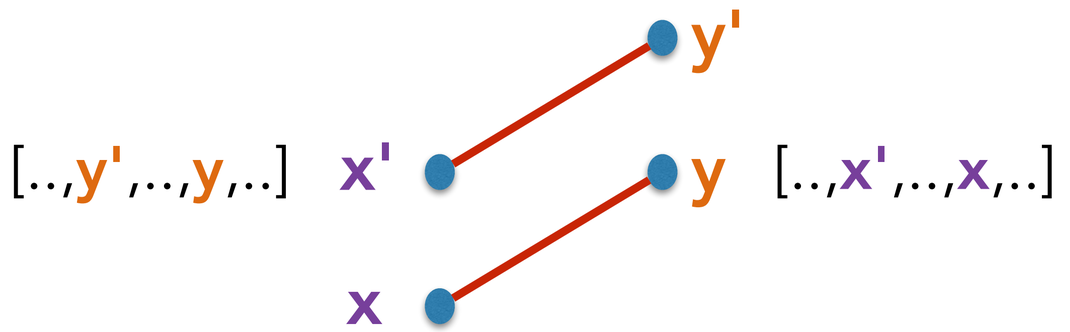
AFSOC, Gale-Shapley does not match **x** with **best**(**x**).

Gale-Shapley Algorithm

n = non-valid partner v = valid partner



Another Stable Matching



Consider the **first time** an **x** gets rejected by a valid partner **y**.

Suppose **x'** is the reason for the rejection.

Then **x'** got rejected by a valid partner **y'**, **before** **x** got rejected by **y**. ■

Further questions

worst(y) = lowest ranked valid partner of y

Theorem: The Gale-Shapley algorithm returns $\{(\textbf{worst}(y), y) : y \in Y\}$.

Exercise

Further questions

Can players lie to improve their matches?

"Incentive Compatibility"

Roth's Theorem:

No matter what algorithm you use,
there is always going to be incentive to lie.

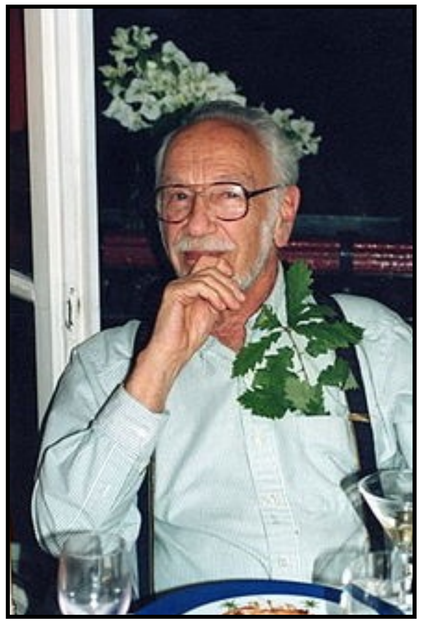
Real-world applications

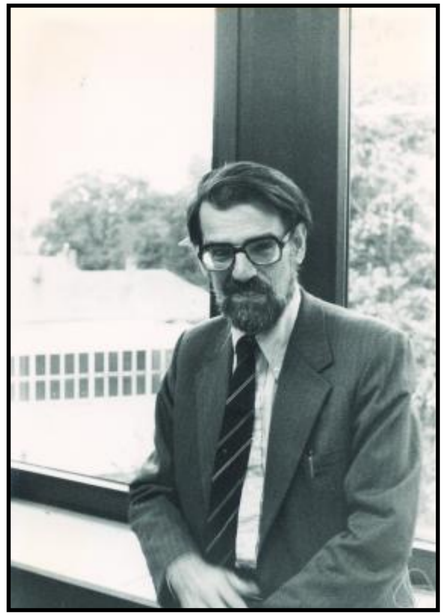
Variants of the Gale-Shapley algorithm are used for:

- 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."