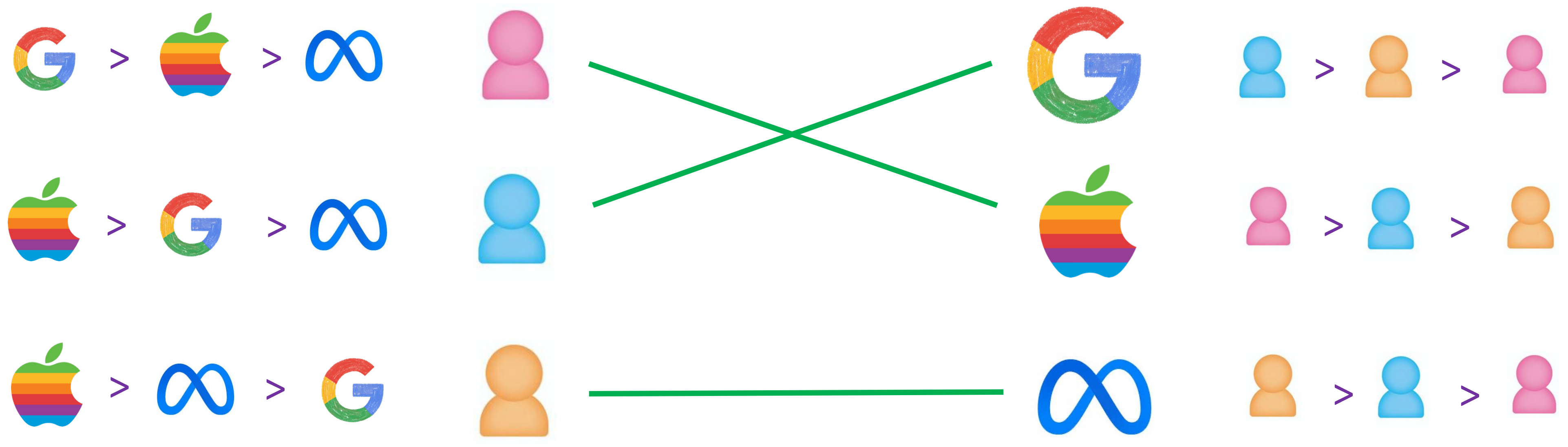
Lecture 4: Stable Matching



Instructor OH

- This week.
  - Today : 3:30 – 4 (lecture hall).

    We have TA meeting starting from 4.

  - Tuesday / Wednesday:

    3:30 – 4 (lecture hall)

    4 – 5 (SODA 729).

  - No OH Thursday / Friday.

Recap of Lecture 2 & 3

- The art of writing mathematical proofs.
    - Direct Proof.
    - Proof by contraposition.
    - Proof by contradiction.
    - Proof by cases.
    - Proof by induction.

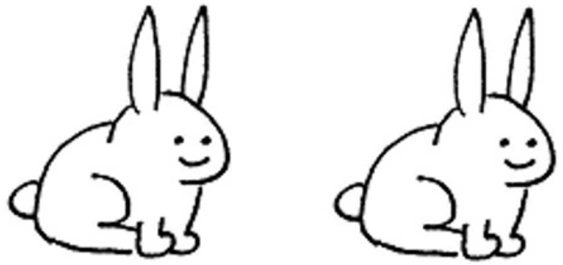
Today's Plan

- Stable Matching
  - A playground for your proof techniques.
  - Prepare you for Algorithm Analysis in CS 170.

  - A theoretical solution for the allocations system.
  - Won Nobel Prize (Free Parking!) in Economics.
  - Use in NYC/Boston Public School Match & National Resident Matching Program
  - Saved numerous lives when applied to kidney exchange.

Suppose you have one rabbit.

Now suppose someone gives you one more rabbit.



Now, if you count your rabbits, you have two rabbits. So one rabbit plus one rabbit equals two rabbits. So one plus one equals two.

$$1 + 1 = 2$$

And that is how arithmetic is done.

Now that you understand the basic idea behind arithmetic, let's take a look at a simple easy-to-understand example that puts into practice what we just learned.

Try It Out
Example 1.7

$$\log \Pi(N) = \left(N + \frac{1}{2}\right) \log N - N + A - \int_N^\infty \frac{\overline{B}_1(x) \mathrm{d}x}{x}, \quad A = 1 + \int_1^\infty \frac{\overline{B}_1(x) \mathrm{d}x}{x}$$

$$\log \Pi(s) = \left(s + \frac{1}{2}\right) \log s - s + A - \int_0^\infty \frac{\overline{B}_1(t) \mathrm{d}t}{t + s}$$

$$\begin{aligned}
\log \Pi(s) &= \lim_{n \to \infty} \left[s \log(N+1) + \sum_{n=1}^{N} \log n - \sum_{n=1}^{N} \log(s+n) \right] \\
&= \lim_{n \to \infty} \Bigg[s \log(N+1) + \int_1^N \log x \, \mathrm{d}x - \frac{1}{2} \log N + \int_1^N \frac{\overline{B}_1(x) \mathrm{d}x}{x} \\
&\quad - \int_1^N \log(s+x) \, \mathrm{d}x - \frac{1}{2}[\log(s+1) + \log(s+N)] \\
&\quad - \int_1^N \frac{\overline{B}_1(x) \mathrm{d}x}{s+x} \\
&= \lim_{n \to \infty} \Bigg[s \log(N+1) + N \log N - N + 1 + \frac{1}{2} \log N + \int_1^N \frac{\overline{B}_1(x) \mathrm{d}x}{x} \\
&\quad - (s+N) \log(s+N) + (s+N) + (s+1) \log(s+1) \\
&\quad - (s+1) - \frac{1}{2} \log(s+1) - \frac{1}{2} \log(s+N) - \int_1^N \frac{\overline{B}_1(x) \mathrm{d}x}{s+x} \Bigg]
\end{aligned}$$

$$
\begin{aligned}
&= \lim_{n\to\infty}\Bigg[s\log(N+1) + N\log N - N + 1 + \frac{1}{2}\log N + \int_1^N \frac{\overline{B}_1(x)\mathrm{d}x}{x} \\
&\quad - (s+N)\log(s+N) + (s+N) + (s+1)\log(s+1) \\
&\quad - (s+1) - \frac{1}{2}\log(s+1) - \frac{1}{2}\log(s+N) - \int_1^N \frac{\overline{B}_1(x)\mathrm{d}x}{s+x}\Bigg] \\
&= \left(s+\frac{1}{2}\right)\log(s+1) + \int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{x} - \int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{s+x} \\
&\quad + \lim_{n\to\infty}\Bigg[s\log(N+1) + \left(N+\frac{1}{2}\right)\log N \\
&\quad - \left(s+N+\frac{1}{2}\right)\log(s+N)\Bigg] \\
&= \left(s+\frac{1}{2}\right)\log(s+1) + (A-1) - \int_1^\infty \frac{\overline{B}_1(x)\mathrm{d}x}{s+x} \\
&\quad + \lim \Bigg[s\log\frac{N+1}{} \quad \left(N + \frac{1}{}\right)\log\left(1 + \frac{s}{}\right)\Bigg]
\end{aligned}
$$

If the authors of computer programming books wrote arithmetic textbooks...

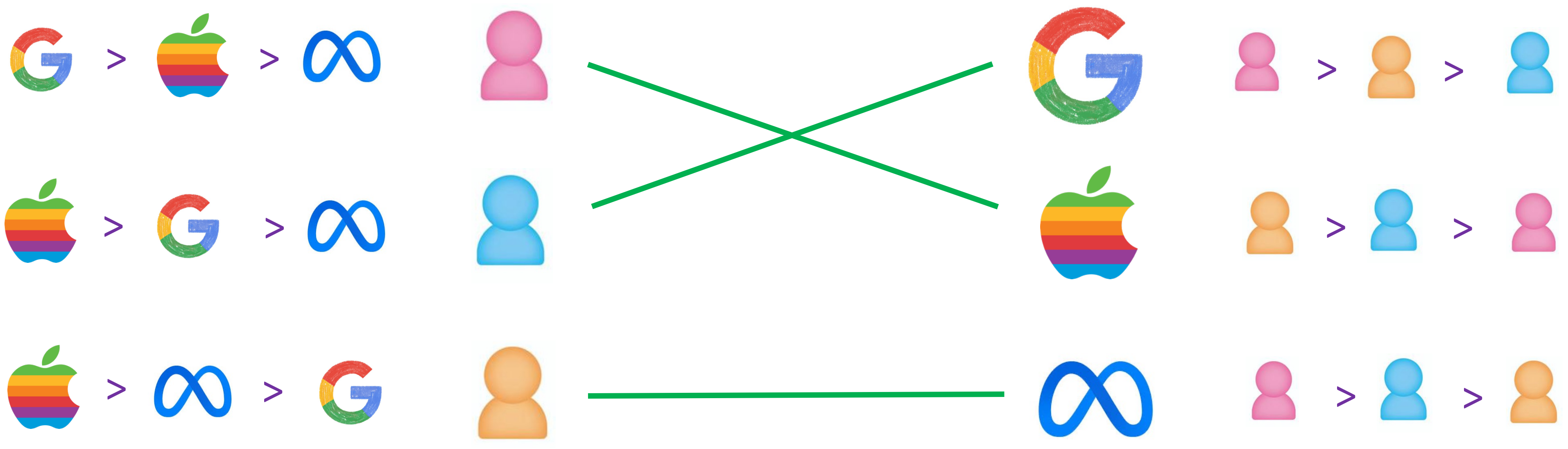
Today's Plan

- Matching Theory
  - What is a reasonable matching?
  - Stability.
- Gale-Sharpley Algorithm
  - Algorithm
  - Improvement Lemma
  - Job-optimal

Matching Theory

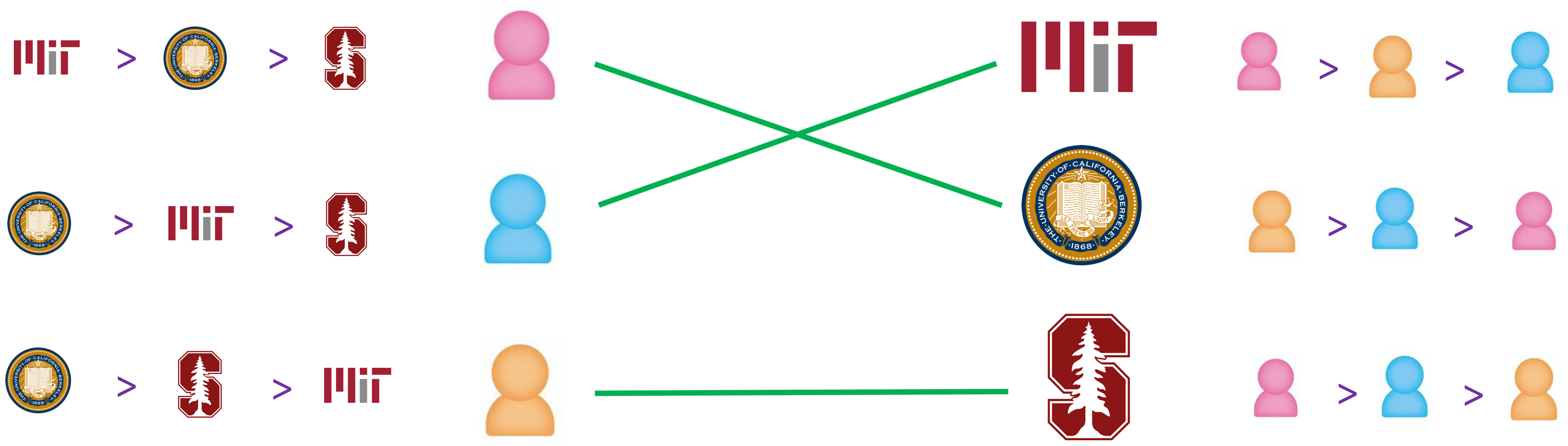
- Two sided-matching
  - On one side, we have n candidates.
  - On the other side, we have n jobs
  - Let's say each job wants to take one candidates.

Matching Theory



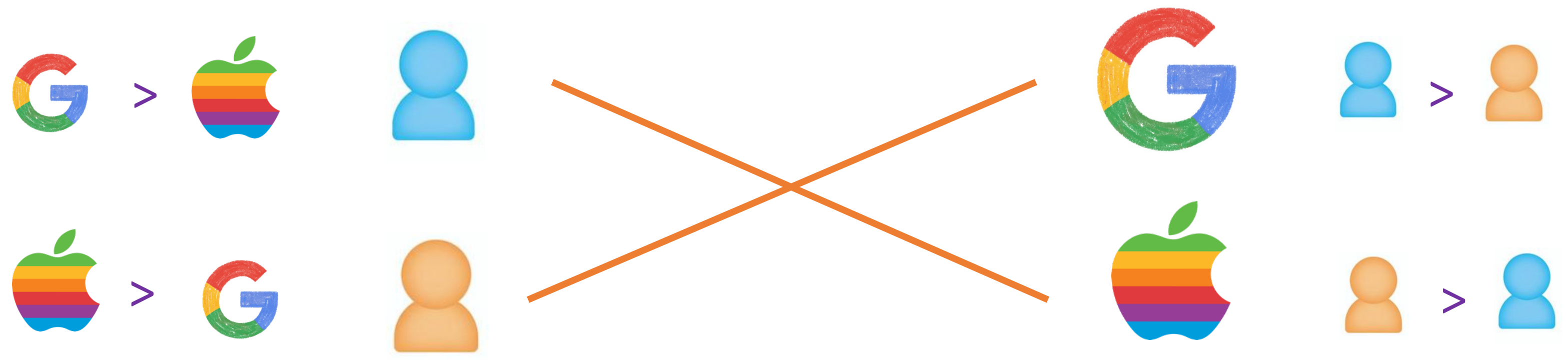
- Two sided-matching
  - Jobs have preference over candidates.
  - Candidates also have preference over jobs.
  - We want to match them in a reasonable way.

Matching Theory



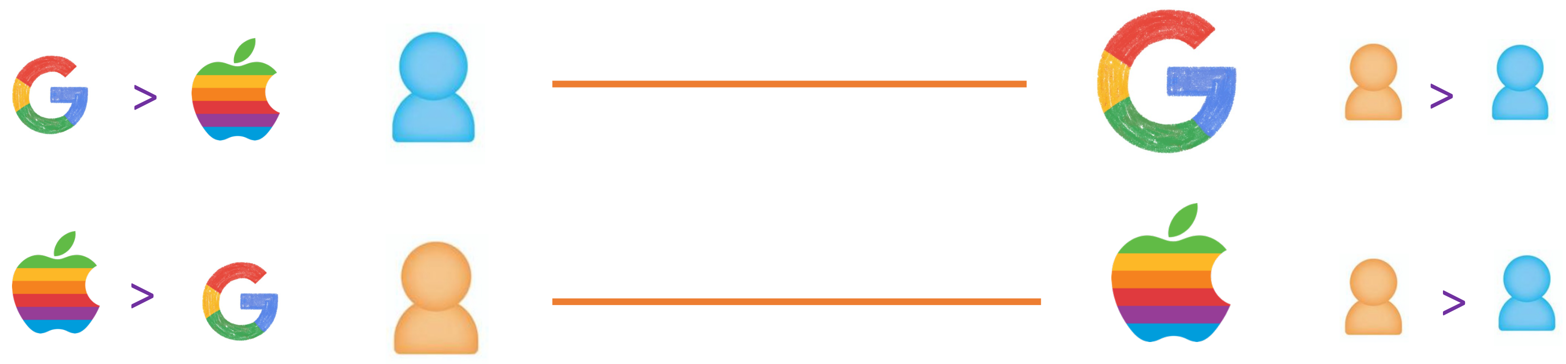
- Numberous applications
  - Students & Schools: The Boston Public School Match (Abdulkadiroglu, Pathak, Roth, and Sonmez, 2005)
  - Doctors & Hospitals: National Resident Matching Program
  - Patient & Organ Donors (a different setting): How a matchmaking algorithm saved lives (Medium)

Which matchings are reasonable?



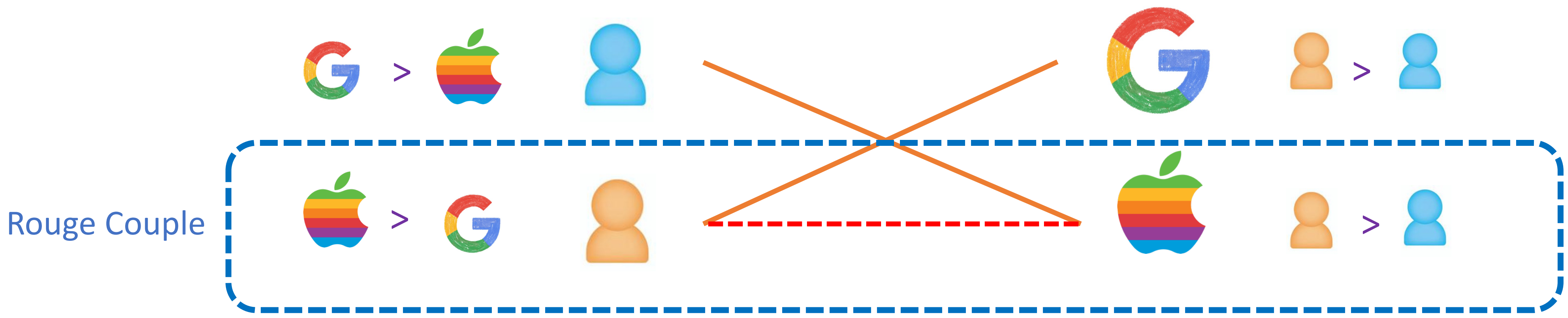
- What is wrong with this matching?
    - When swapped, everyone is happier.

Which matchings are reasonable?



- Is there anything wrong with this matching?
  - No. It is also reasonable.

Which matchings are reasonable?



- Is there anything wrong with this matching?
  - Apple could fire Blue and persuade Orange to quit and join them.

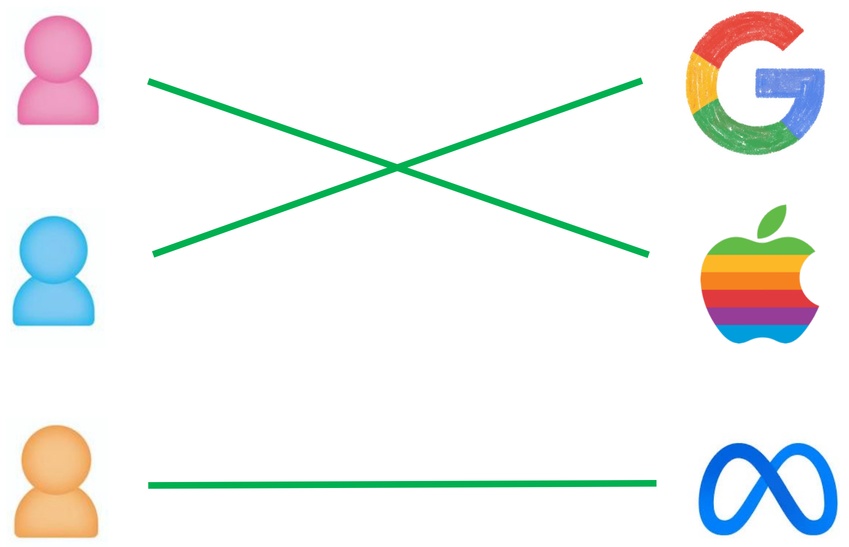
Which matchings are reasonable?

Definition (Matching)

Let jobs be $j_1, j_2, \ldots, j_n$ and candidates be $c_1, c_2, \ldots, c_n$.

A matching is a disjoint set of n candidate-job pairs.

E.g. $\{(c_2, j_1), (c_1, j_2), (c_3, j_3)\}$.

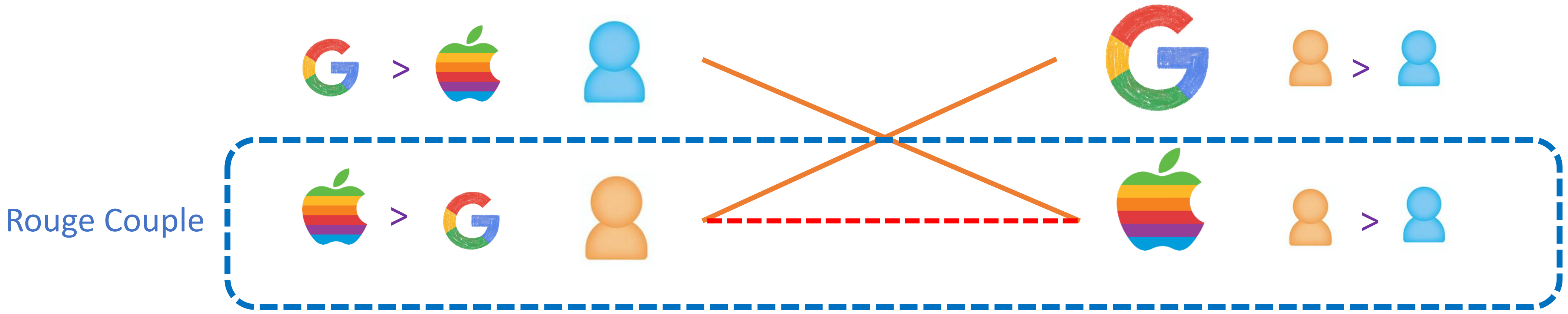


Which matchings are reasonable?

Definition (Rouge Couple)

(c^*, j^*) is a rouge couple, if they both prefer each other than their current match.

In other words, they will elope.



Stability

Definition (Stable matching)

A matching is stable if and only if there is no rouge couple.

- Stability is in some sense the minimum requirement of a reasonable matching.
  - If a matching is not stable, it cannot persist for long.
- How to find a stable matching? Do they always exist?

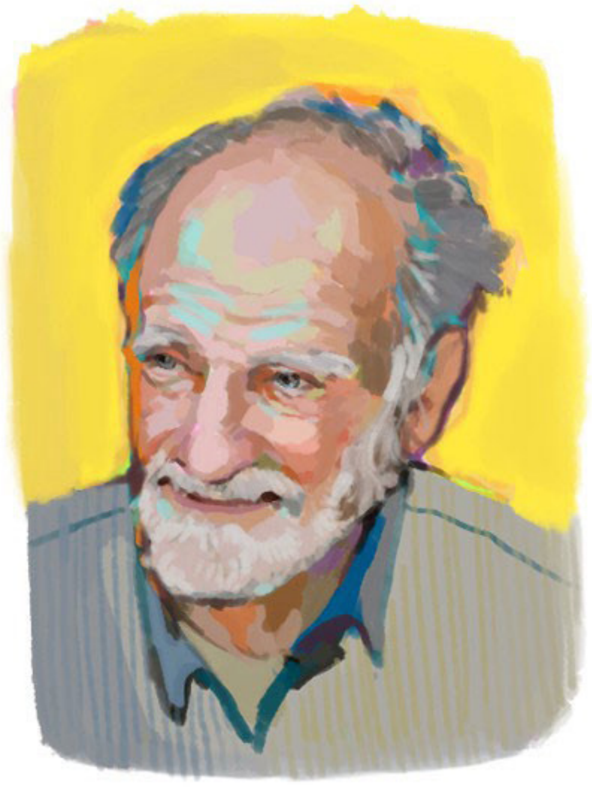
Today’s Plan

- Matching Theory
    - What is a reasonable matching?
    - Stability.
- Gale-Sharpley Algorithm
    - Algorithm
    - Improvement Lemma
    - Job-optimal

Gale-Sharpley Algorithm (propose-and-reject).

David Gale
PROFESSOR, UC BERKELEY

1921 - 2008



Lloyd Shapley
PROFESSOR EMERITUS, UCLA

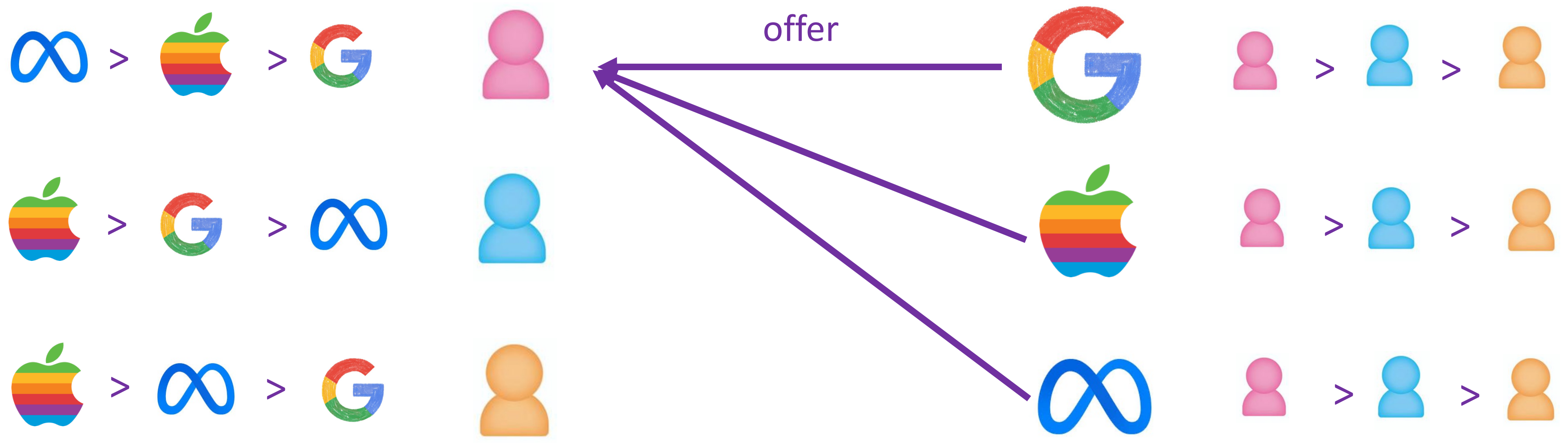
1923 - 2016

Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 1 Morning

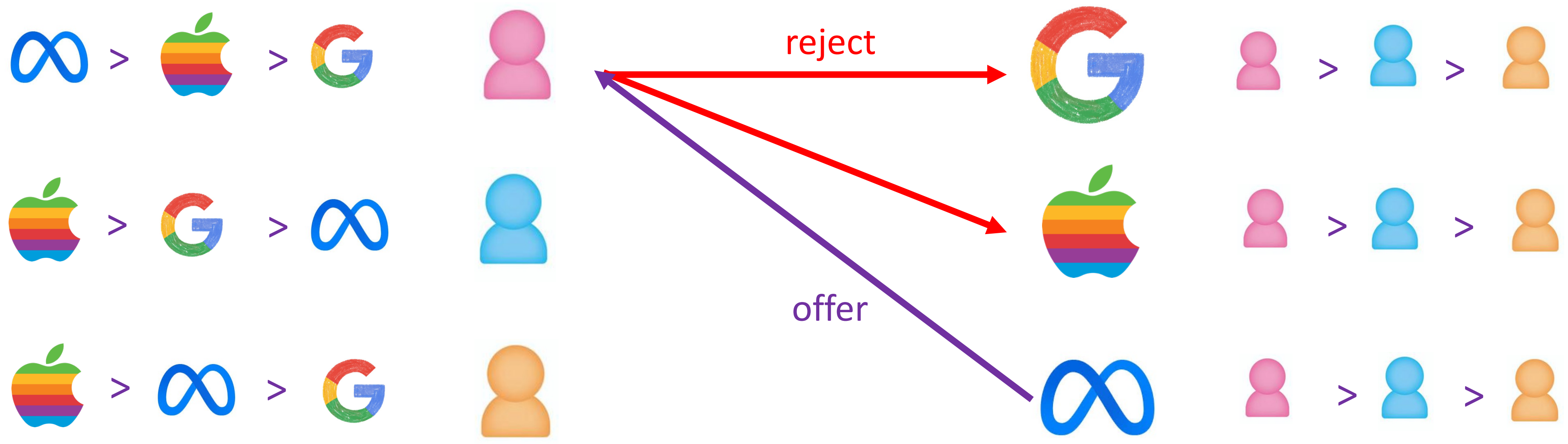


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 1 Afternoon

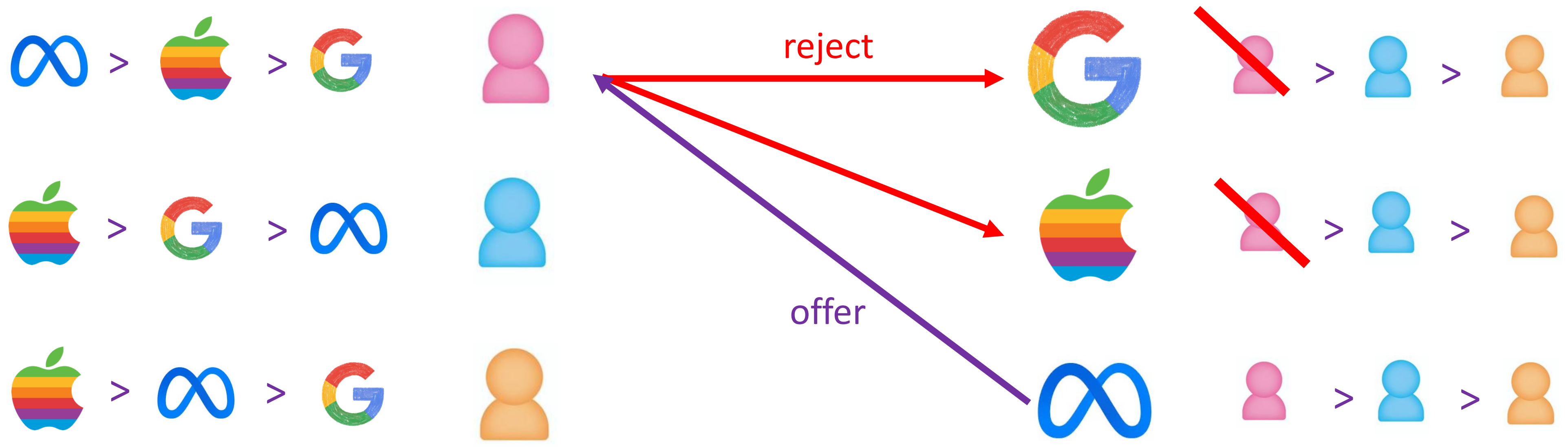


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

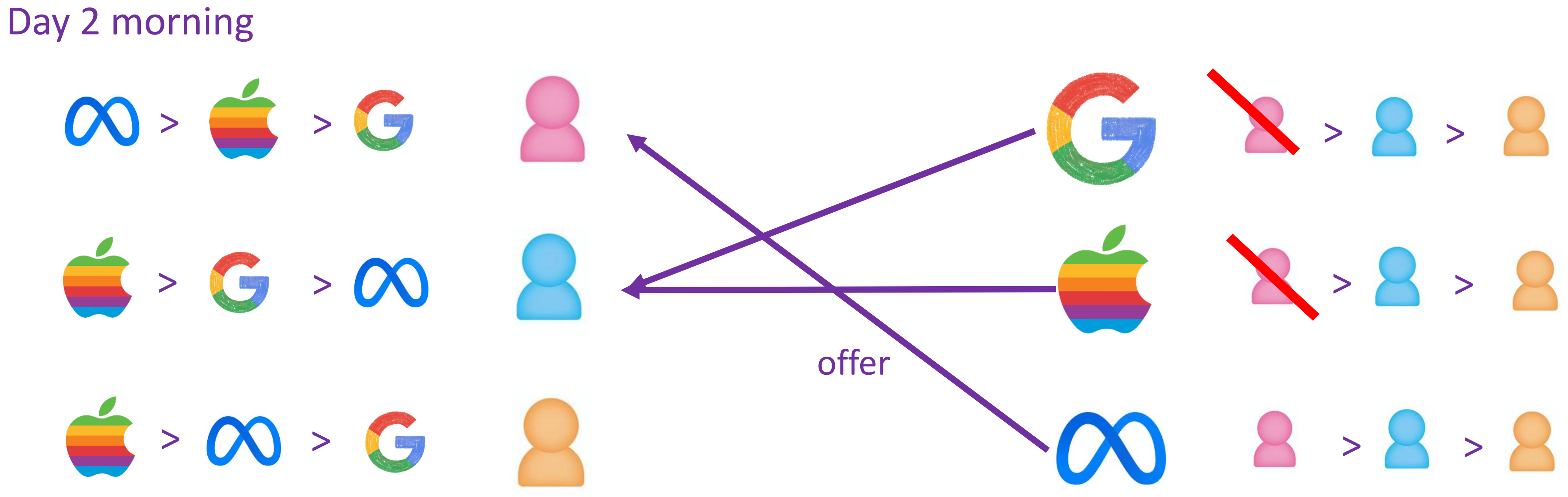
Day 1 night



Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

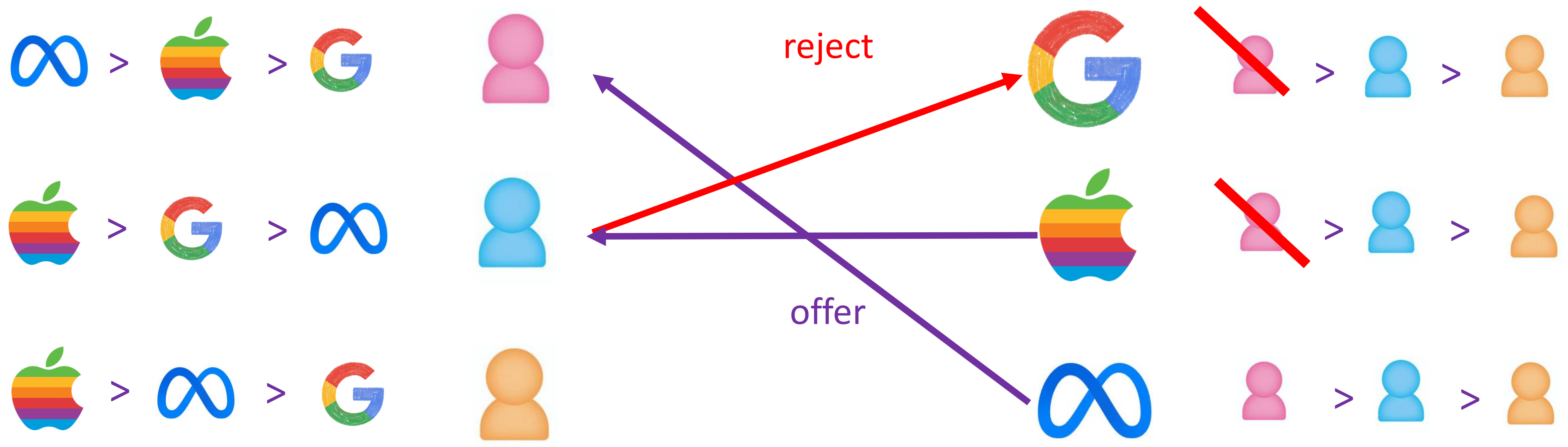


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 2 afternoon

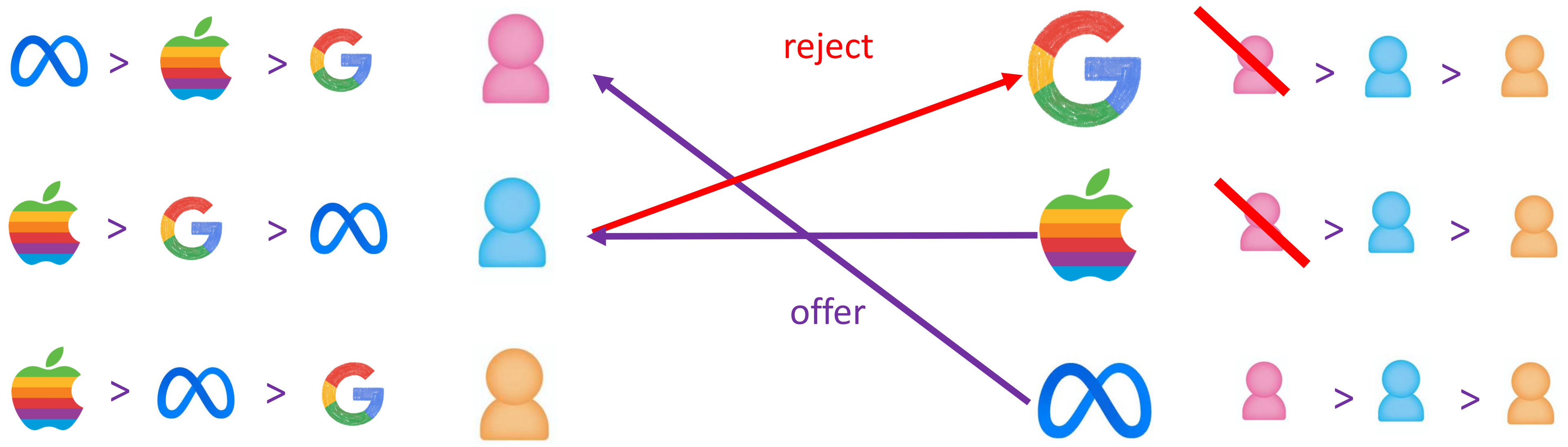


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

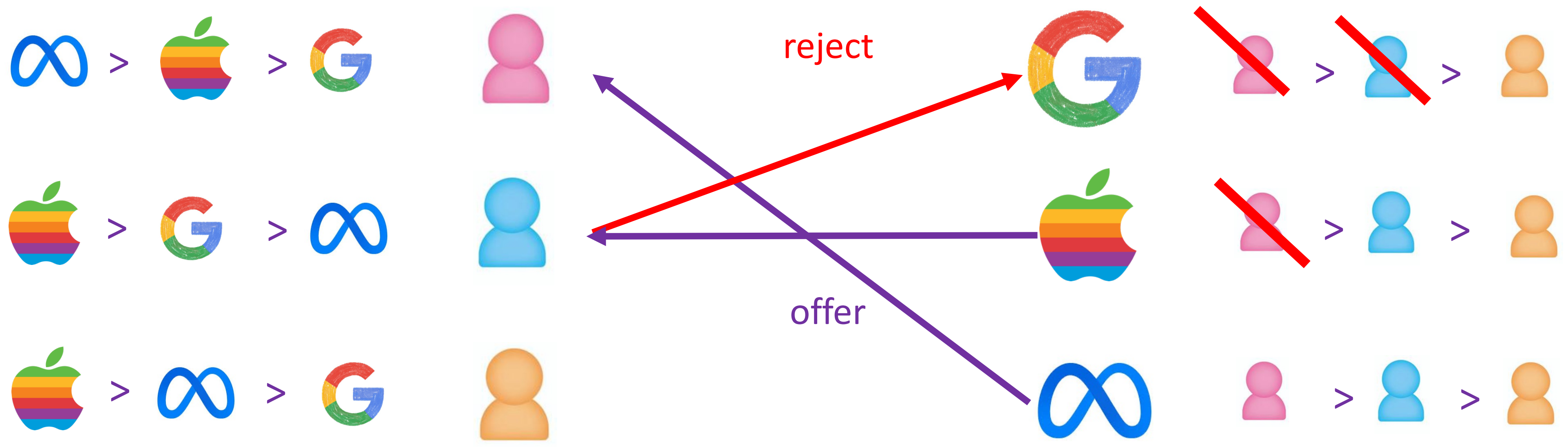
Day 2 afternoon



Propose-and-reject Algorithm.

Each day:
1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 2 night

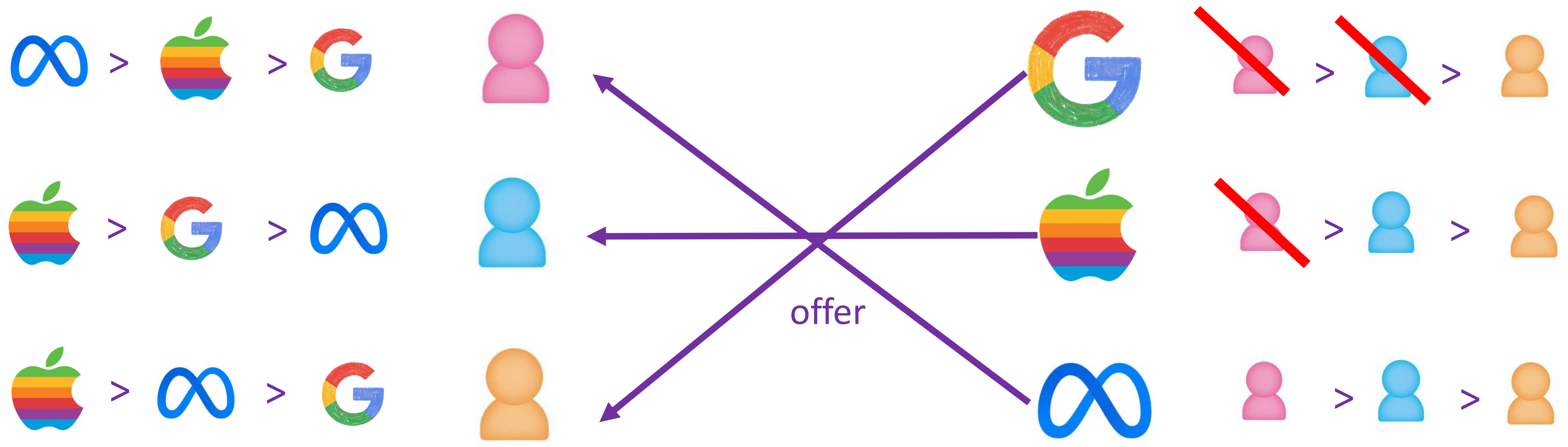


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 3 morning

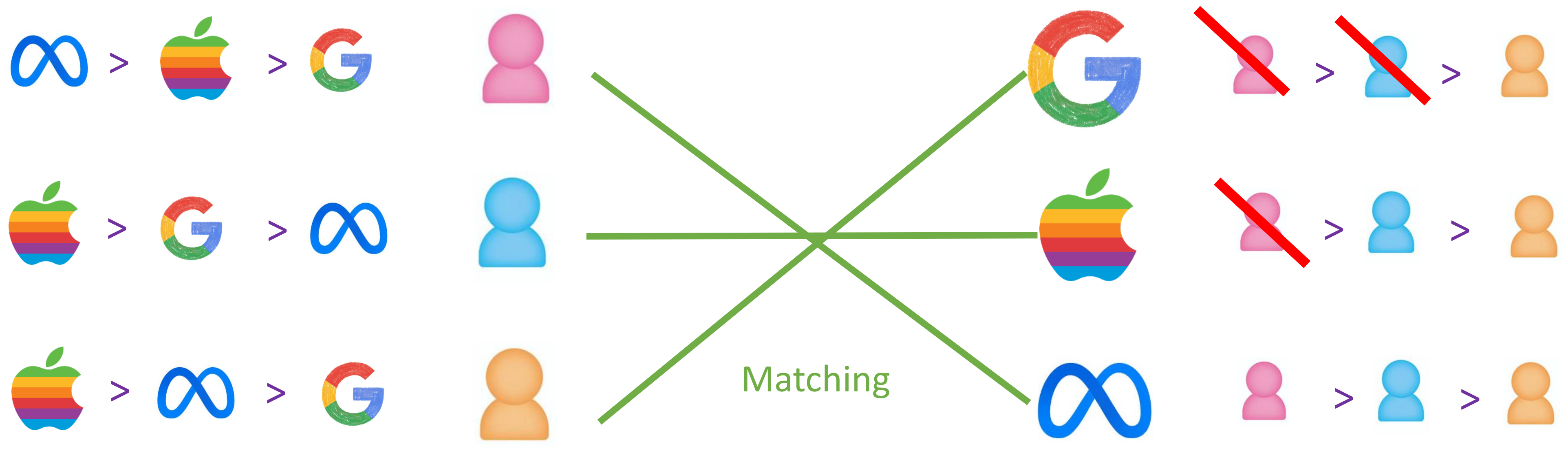


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

We finalize the matching when no one rejects offer anymore.

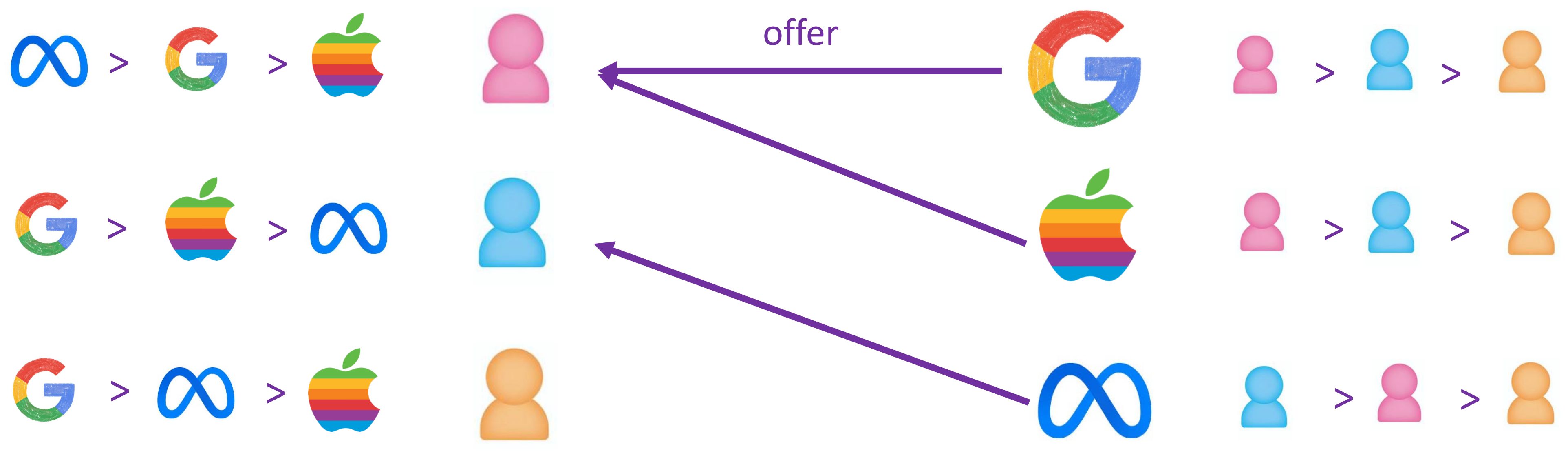


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 1 Morning

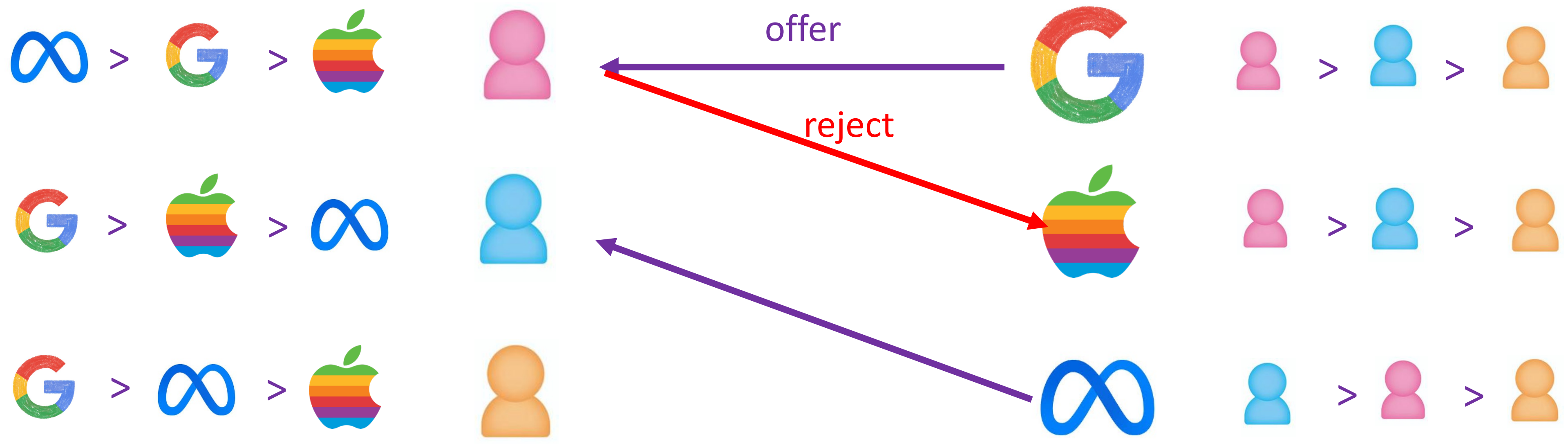


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 1 afternoon

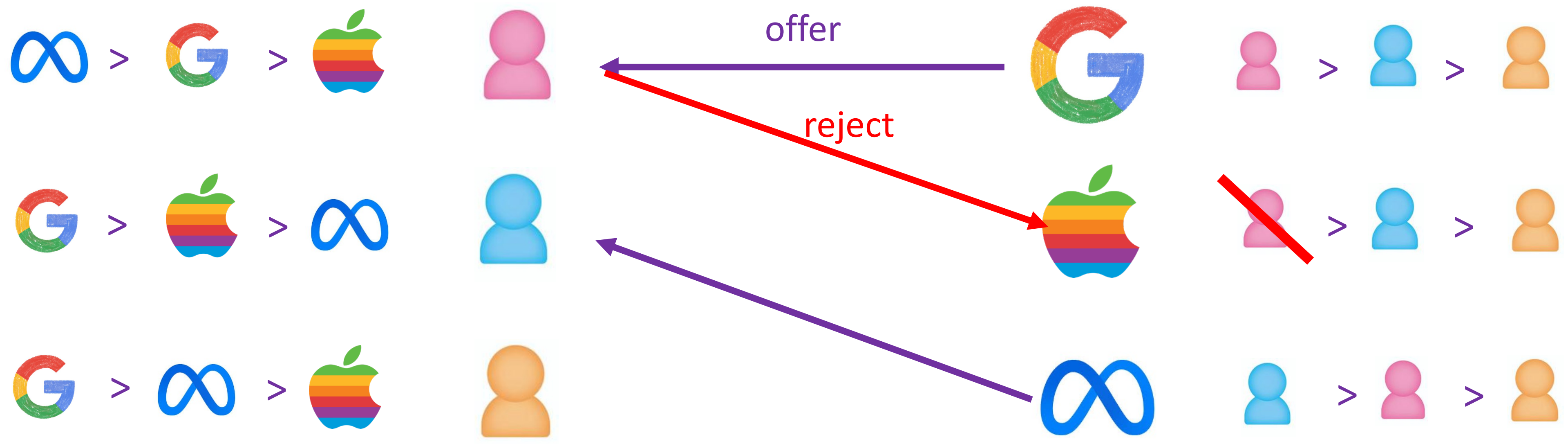


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 1 night

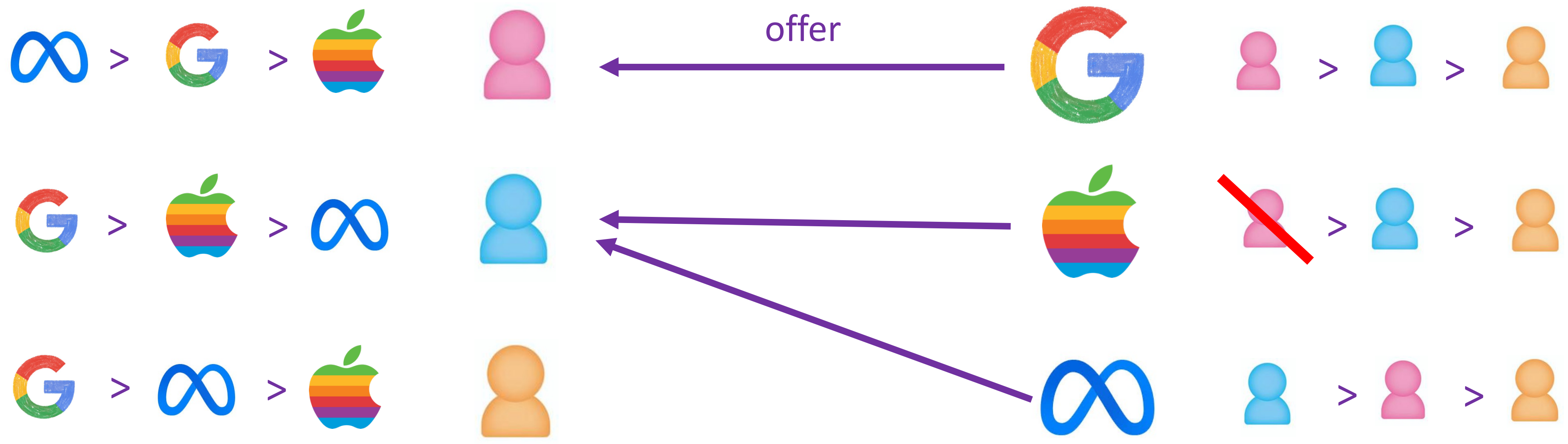


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 2 morning

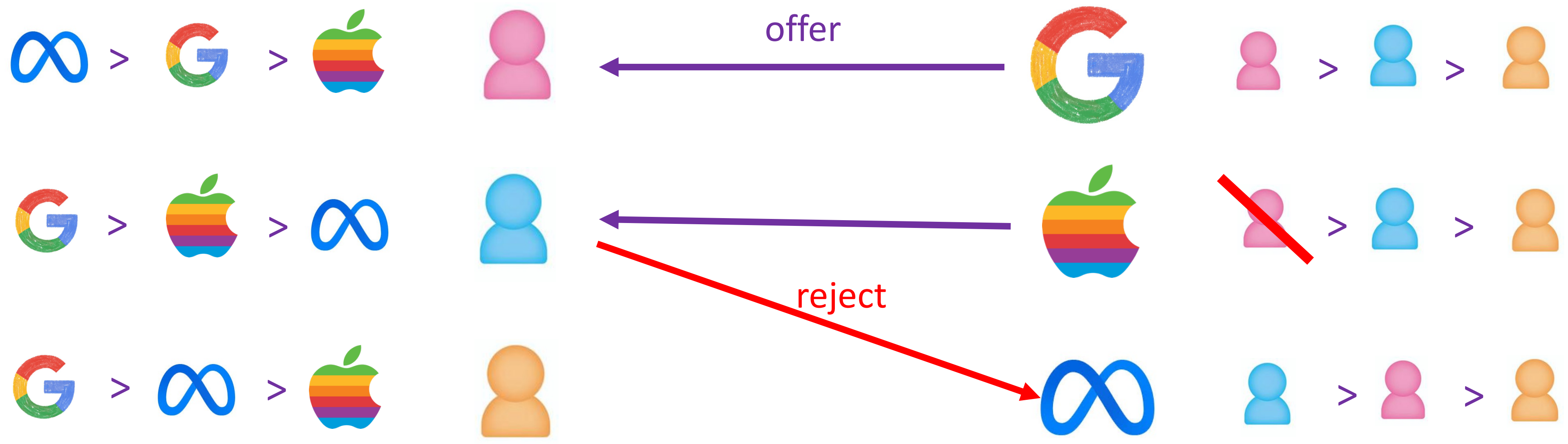


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 2 afternoon

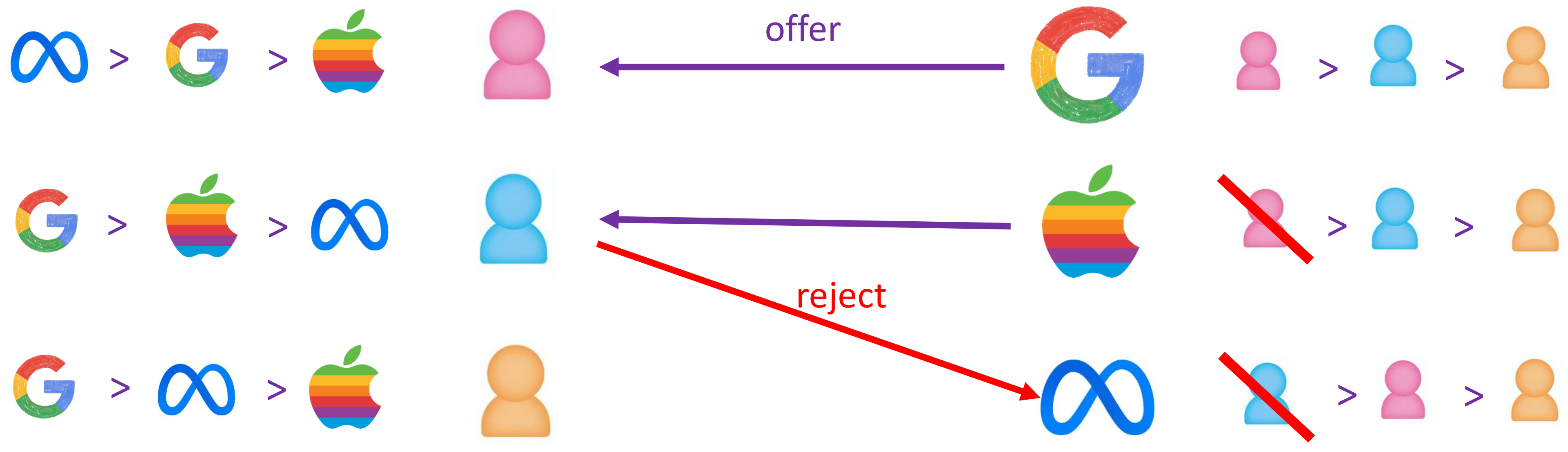


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 2 night

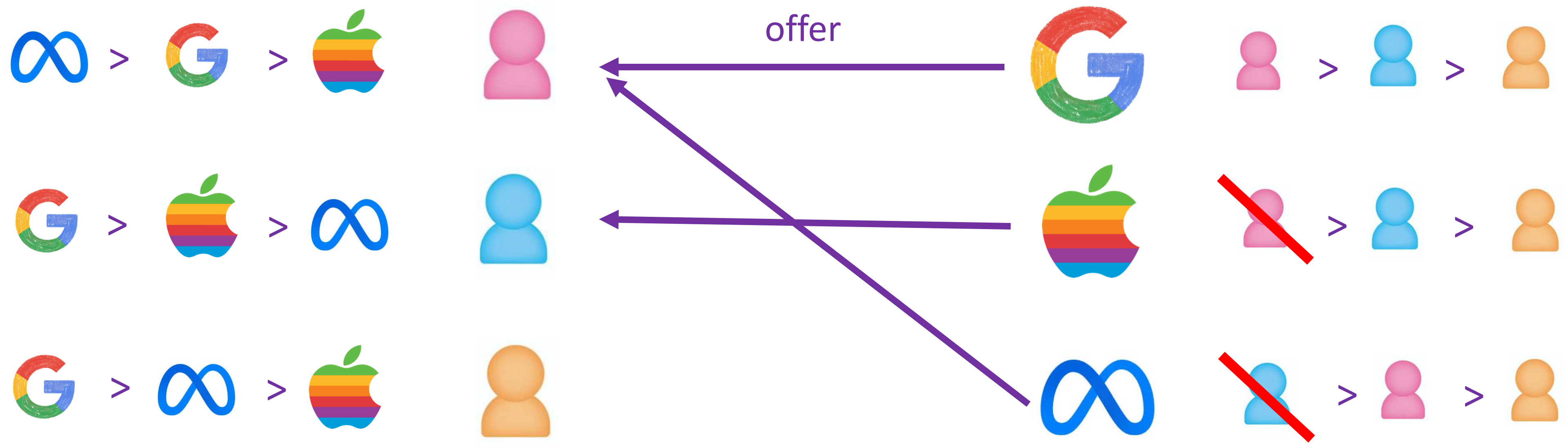


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 3 morning

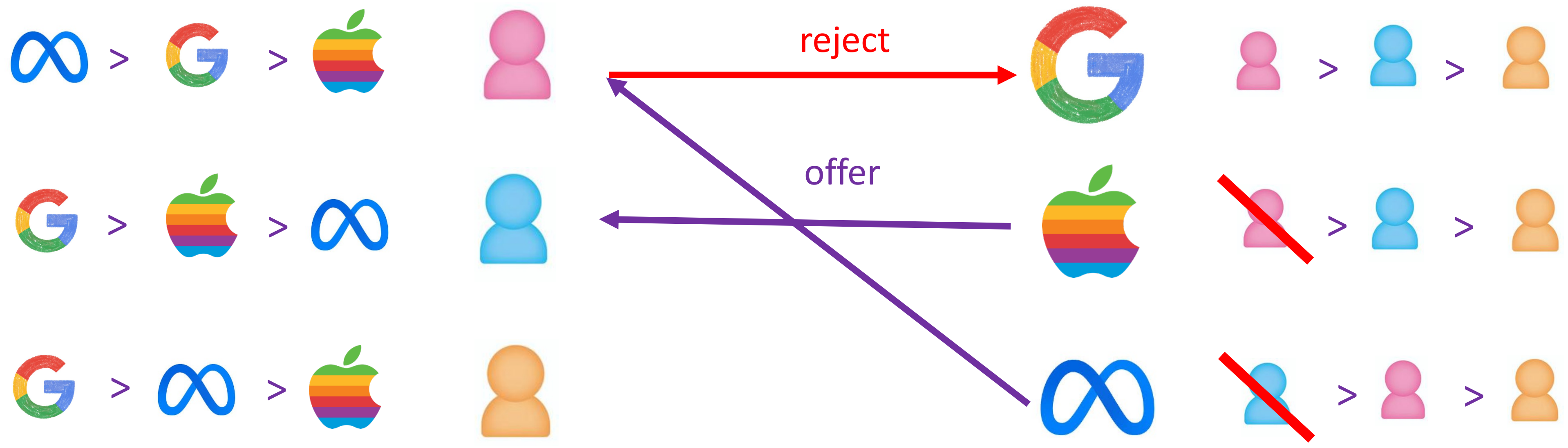


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 3 afternoon

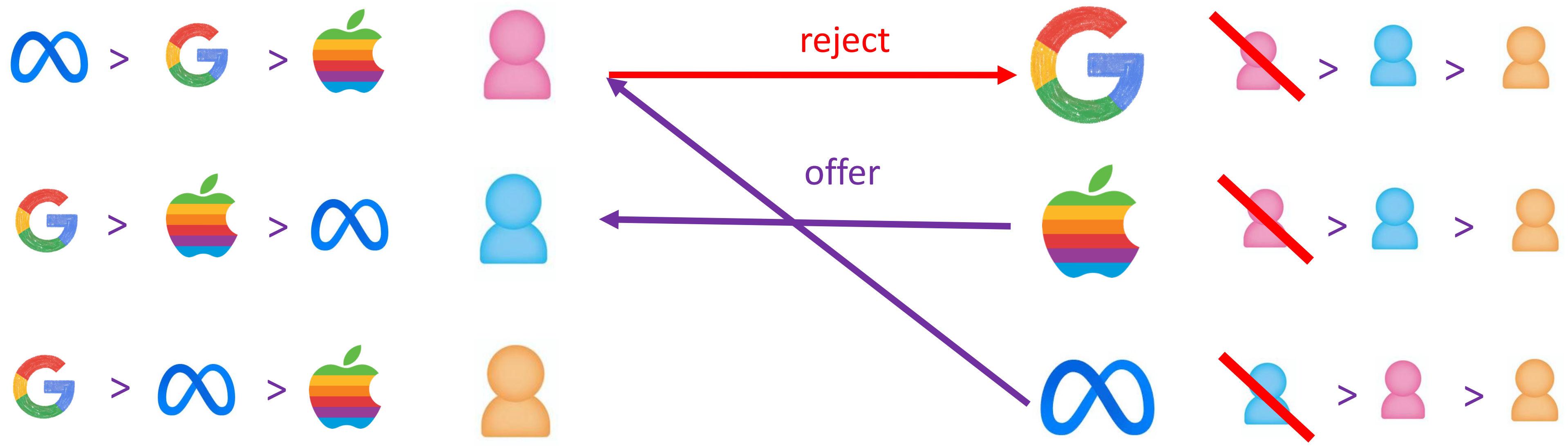


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 3 night

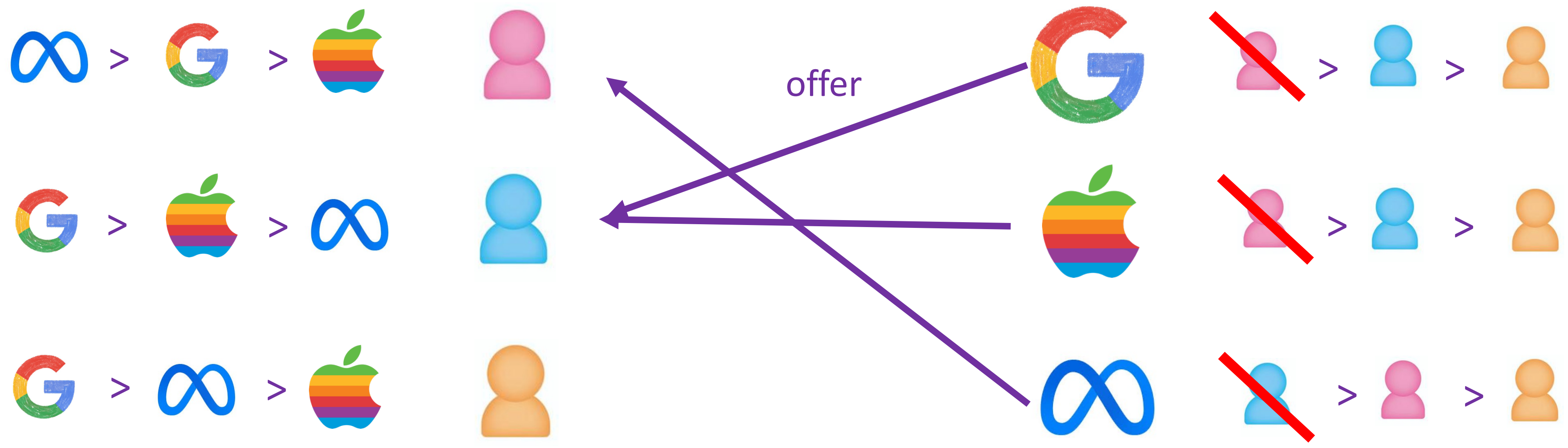


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 4 morning

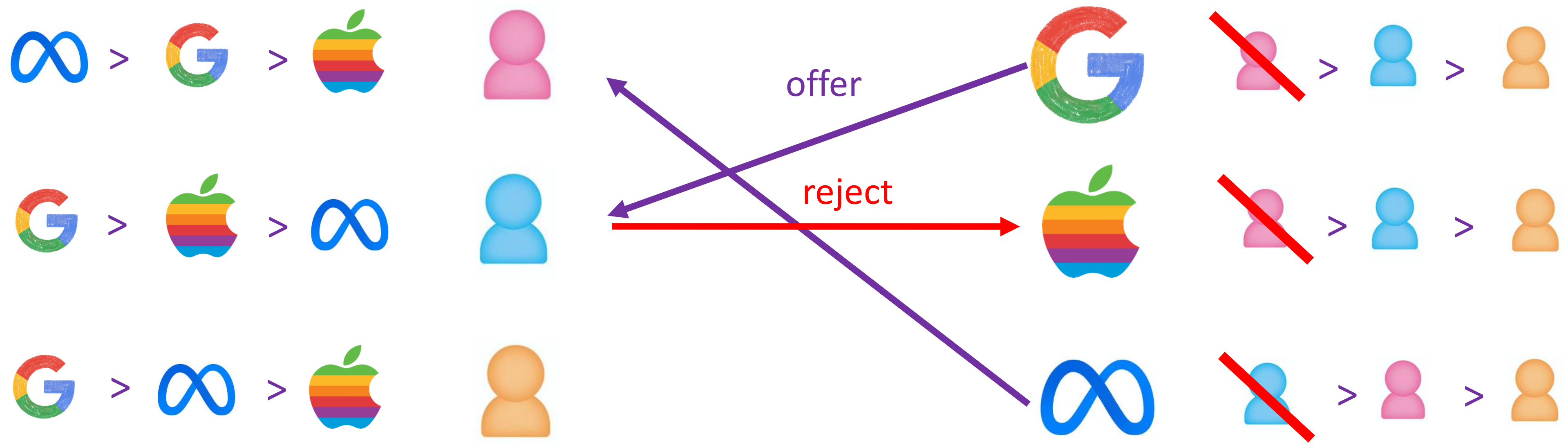


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 4 afternoon

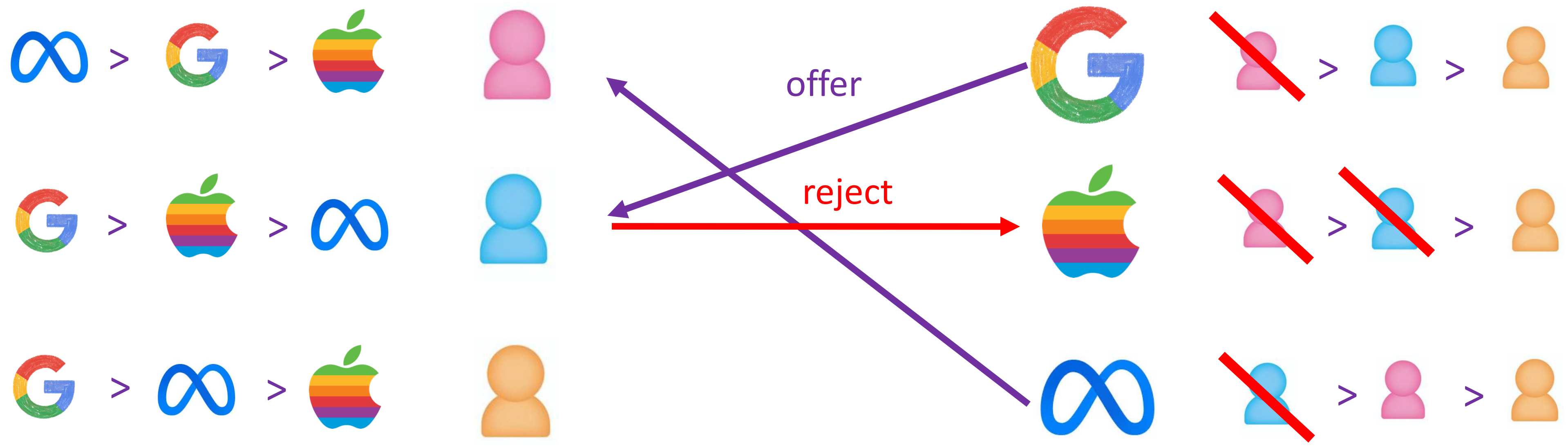


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

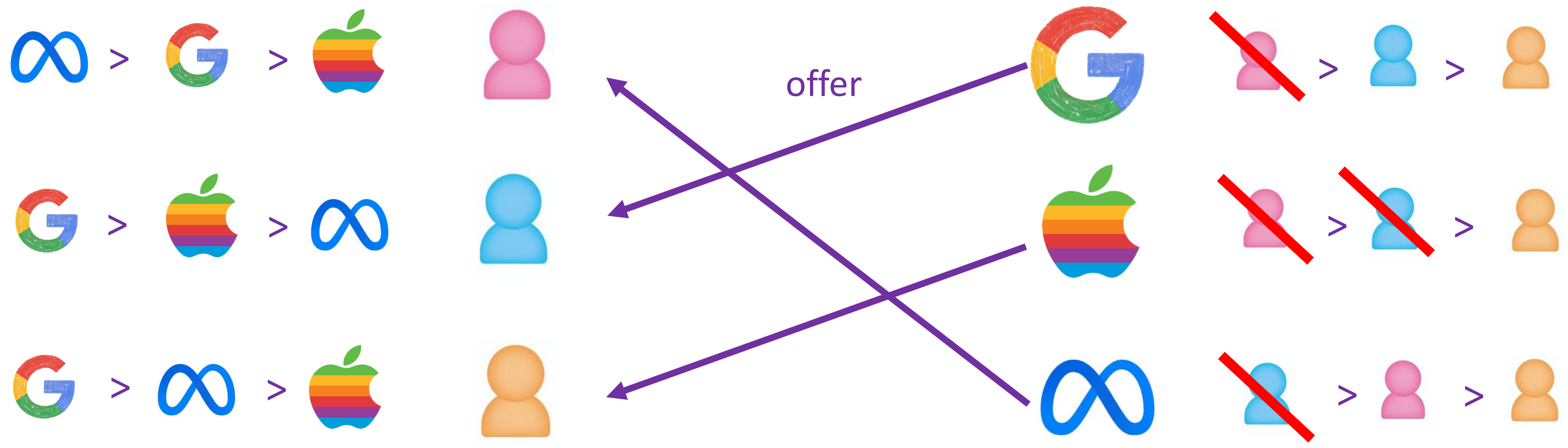
Day 4 night



Propose-and-reject Algorithm.

Each day:
1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

Day 5 morning

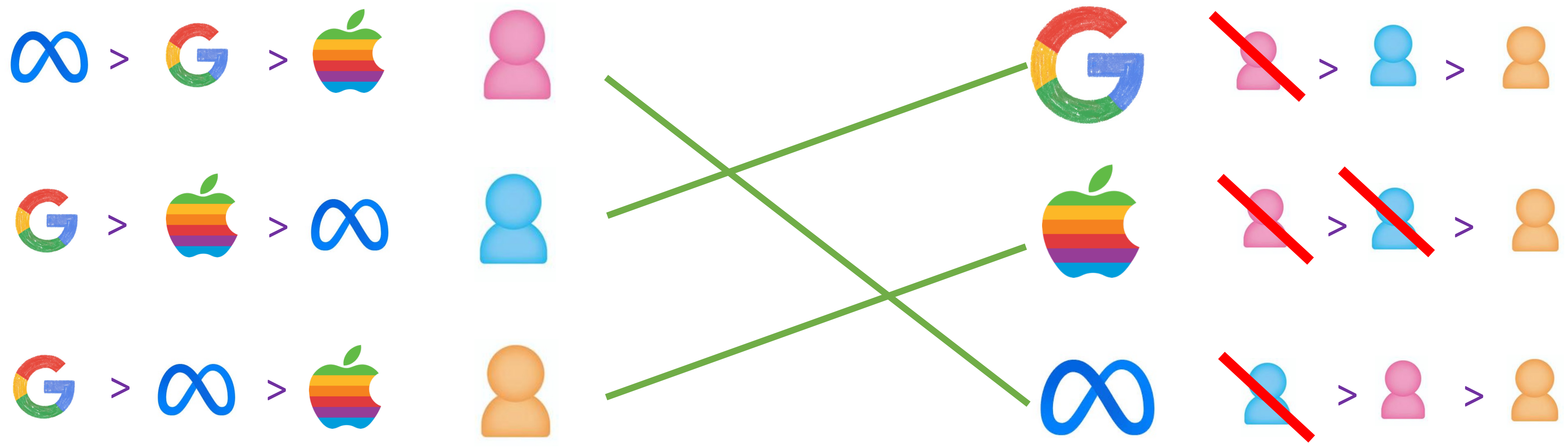


Propose-and-reject Algorithm.

Each day:

1. Jobs send offer to the most preferred candidate who has not rejected the job.
2. Each candidate rejects all but the favorite offer received today.
3. Each job crosses off the rejecting candidates from its list.

We finalize the matching



Termination

- Every non-terminated day a job crossed >=1 candidate off the list.
  - We have n^2 total list size.
  - Must terminate in n^2 days (finite).

Why is it stable?

- To prove this, we need Improvement Lemma.

Lemma (Improvement Lemma: it only gets better for candidates)
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Proof (by induction)

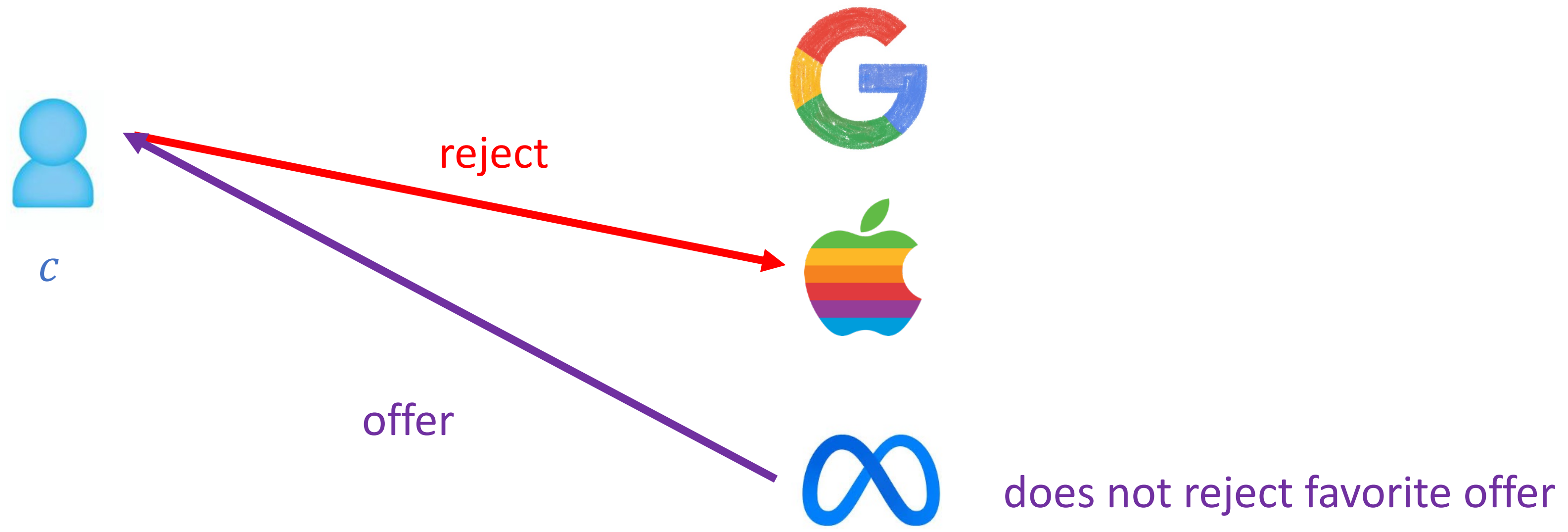
Base Case: For t = 1, there is no previous day. This is vacuously true.
Induction Hypothesis: Suppose this is true for day t – 1.
Inductive Step: By induction hypothesis, c's favorite offer on day t – 1, is better than that of day 1, 2, …., t – 1.
But as c does not reject favorite offer on day t – 1. This offer is still there on day t.
The favorite offer of c on day t must be same or even better.

Why is it stable?

- To prove this, we need Improvement Lemma.

Lemma (Improvement Lemma: it only gets better for candidates)
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Day t - 1

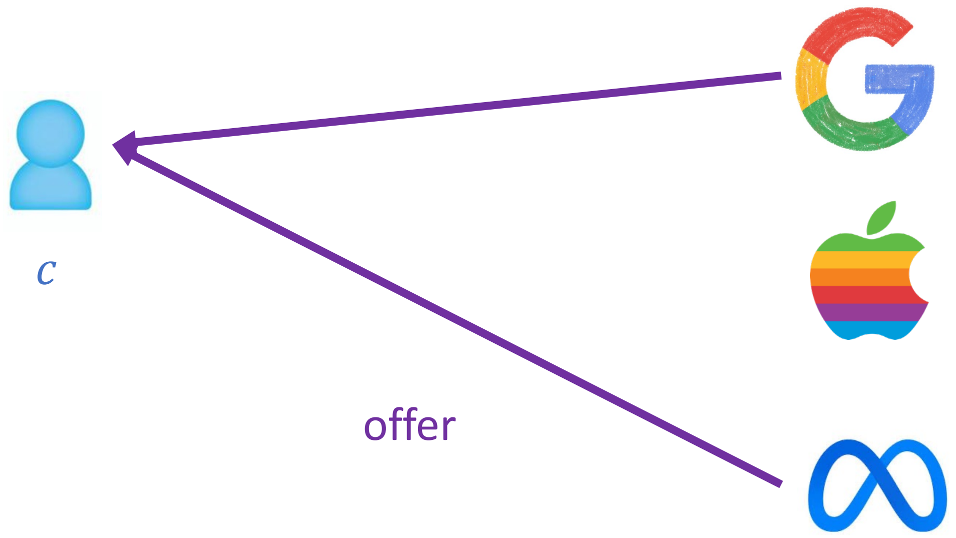


Why is it stable?

- To prove this, we need Improvement Lemma.

Lemma (Improvement Lemma: it only gets better for candidates)
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Day t



This offer still available, and may have Offer from other company

Why is it stable?

Lemma (Improvement Lemma: it only gets better for candidates)

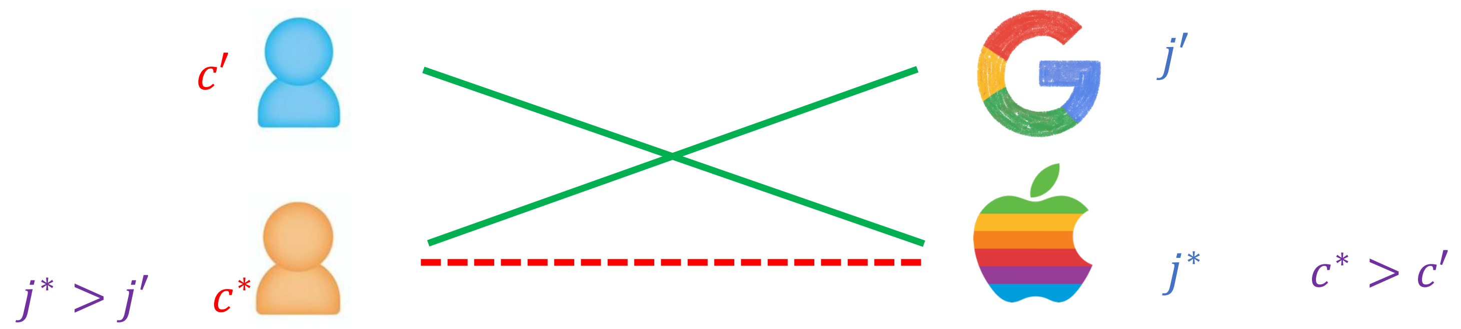
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Theorem (Stability)

There is no rouge couple in the matching found by this algorithm.

Proof (by contradiction).

Suppose there is a rouge couple (c^*, j^*) with their match c', j'.
Since j^* likes c^* more, it must have given offer to c^* before it gives offer to c' and got rejected.



Why is it stable?

Lemma (Improvement Lemma: it only gets better for candidates)
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Theorem (Stability)
There is no rouge couple in the matching found by this algorithm.

Proof (by contradiction).
Suppose there is a rouge couple (c^*, j^*) with their match c', j'.
Since j^* likes c^* more, it must have given offer to c^* before it gives offer to c' and got rejected.

Thus c^* must have a better offer then. But now c^* 's favorite offer is j' which is worse. Contradiction with Improvement Lemma.

Every job is matched at end.

Lemma (Improvement Lemma: it only gets better for candidates)
Suppose, on day t, candidate c's favorite offer is from job j. It must be more preferred or equal to c's favorite offer on any previous day t' < t.

Lemma
At the end, each job is matched.

Proof (by contradiction).
If not, this job has given offer to every candidate and got rejected.
By Improvement Lemma, each candidate must have at least one offer in the end.
n jobs and n candidates.
Thus each job should be one candidate's favorite offer. Contradiction.

Today’s Plan

- Matching Theory
  - What is a reasonable matching?
  - Stability.
- Gale-Sharpley Algorithm
  - Algorithm
  - Improvement Lemma
  - Job-optimal

Job-Optimal

Theorem
With job proposing,
  Every job gets its best match among all stable matchings.
  Every candidate gets its worst match among all stable matchings.

Claim: The optimal partner for a job must be first in its preference list.
True / False?

False!

Proof. See discussion 2A.