CPSC 365 / ECON 365:

Algorithms

Lecture 2: Review on Asymptotics and Graphs

Andre Wibisono

Yale University

January 27, 2022

Last time

- Logistics, overview
- Review: Logic, proofs

Today:

- Review: Asymptotics
- Review: Graphs
- Stable Matching Problem

Plan

Review: Asymptotics

Review: Graphs

Stable Matching Problem

Asymptotic Notation

Let $f(n)$, $g(n)$ be functions of $n \in \mathbb{N}$
(for example, running times of two algorithms on inputs of size n)

- We say $f = O(g)$ if there exist $C > 0$, $N \in \mathbb{N}$ such that for all $n \geq N$, we have $f(n) \leq C \cdot g(n)$.

Asymptotic Notation

Let $f(n)$, $g(n)$ be functions of $n \in \mathbb{N}$
(for example, running times of two algorithms on inputs of size n)

- We say $f = O(g)$ if there exist $C > 0$, $N \in \mathbb{N}$ such that for all $n \geq N$, we have $f(n) \leq C \cdot g(n)$.

  Equivalently, $\frac{f(n)}{g(n)}$ is bounded above by a constant over $n \in \mathbb{N}$.

- We say $f = \Omega(g)$ if $g = O(f)$.
- We say $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

$f = O(g) \wedge g = O(f) \iff f = \Theta(g)$

Analogy

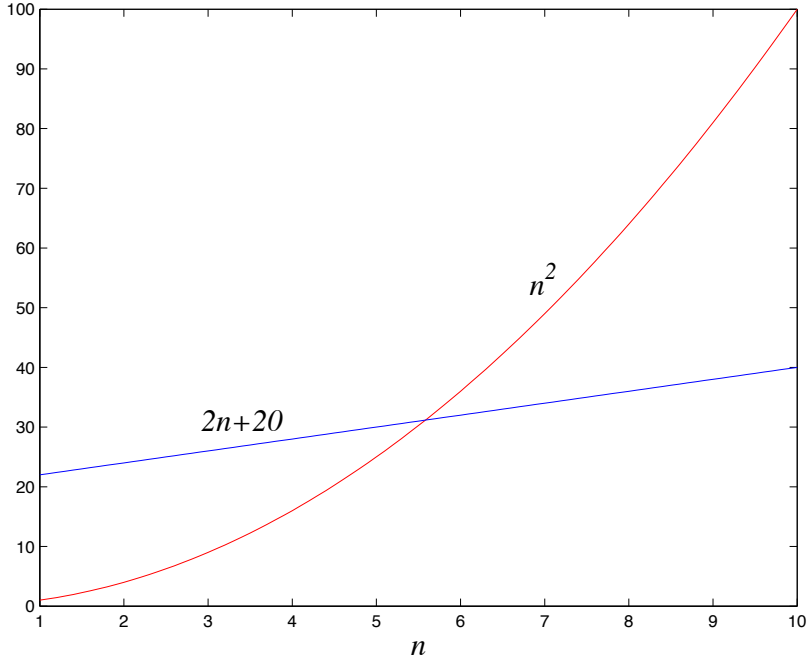
$f = O(g)$	$f \lesssim g$
$f = \Omega(g)$	$f \gtrsim g$
$f = \Theta(g)$	$f \approx g$

Quiz 1

Which running time is better?

(a) $2n + 20$

(b) n^2

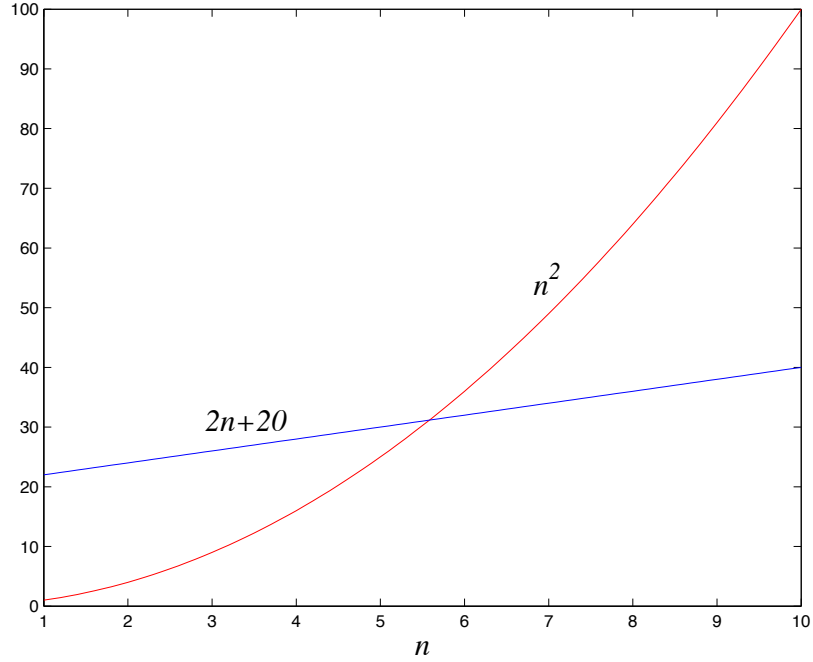


Quiz 1

Which running time is better?

(a) $2n + 20$

(b) n^2



Answer: (a) $2n + 20$ is better (for large n)

Examples

- $f(n) = 2n + 20, \quad g(n) = n^2$

  $f = O(g)$ because $2n + 20 \leq n^2$ for all $n \geq 5$. ($N = 5$, $C = 1$)

Examples

- $f(n) = 2n + 20, \quad g(n) = n^2$

  $f = O(g)$ because $2n + 20 \leq n^2$ for all $n \geq 5$. ($N = 5$, $C = 1$)

- Let $f(n) = 2n^2 + 5, \quad g(n) = n^2 + 2n + 25$

  $f = \Theta(g)$ because $f(n) \leq 2g(n)$ for all $n \geq 1$,
  and $f(n) \geq g(n)$ for all $n \geq 5$.

Addition

Carry:	1			1	1	1		
		1	1	0	1	0	1	(53)
		1	0	0	0	1	1	(35)
	1	0	1	1	0	0	0	(88)

- What is running time of this algorithm to add two n-bit numbers?

$$\Theta(n)$$

Quiz 2: Multiplication

					1	1	0	1
				$\times$	1	0	1	1
					1	1	0	1
				1	1	0	1	
			0	0	0	0		
$+$		1	1	0	1			
1	0	0	0	1	1	1	1	

(1101 times 1)
(1101 times 1, shifted once)
(1101 times 0, shifted twice)
(1101 times 1, shifted thrice)
(binary 143)

What is running time of this algorithm to multiply two n-bit numbers?

(a) $\Theta(n)$
(b) $\Theta(n \log n)$
(c) $\Theta(n^2)$
(d) $\Theta(n^4)$

Quiz 2: Multiplication

$$
\begin{array}{cccccccc}
 & & & & 1 & 1 & 0 & 1 \\
 & & & \times & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 1 & 0 & 1 \\
 & & & 1 & 1 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 1 & 1 & 0 & 1 & & & \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

(1101 times 1)
(1101 times 1, shifted once)
(1101 times 0, shifted twice)
(1101 times 1, shifted thrice)
(binary 143)

What is running time of this algorithm to multiply two n-bit numbers?

(a) $\Theta(n)$
(b) $\Theta(n \log n)$
(c) $\Theta(n^2)$
(d) $\Theta(n^4)$

Answer: (c) $\Theta(n^2)$

Arithmetic Algorithms: Multiplication

					1	1	0	1
				×	1	0	1	1
					1	1	0	1
				1	1	0	1	
			0	0	0	0		
+		1	1	0	1			
1	0	0	0	1	1	1	1	

(1101 times 1)
(1101 times 1, shifted once)
(1101 times 0, shifted twice)
(1101 times 1, shifted thrice)
(binary 143)

- $\Theta(n^2)$ to multiply two n-bit numbers with this algorithm
- Can we do better? Yes! See Lecture 12

Orders of Growth

- **Polynomial:** $n,\ n^2,\ n^3,\ n^k\ (k > 0)$

  If input doubles, output is multiplied by a constant:

  $$(2n)^k = 2^k \cdot n^k$$

- **Exponential:** $2^n,\ 3^n,\ e^n,\ b^n\ (b > 0)$

  If input doubles, output is squared:

  $$b^{2n} = (b^n)^2$$

- **Logarithm:** $\log_2 n,\ \log n,\ \log_b n\ (b > 0)$

  If input doubles, output is added by a constant:

  $$\log(2n) = \log n + \log 2$$

Exponential

Euler's number:

$$e = \lim_{x \to \infty} \left(1 + \frac{1}{x}\right)^x = 2.71828...$$

For all $b > 0$, $m, n > 0$:

- $b^{m+n} = b^m \cdot b^n$
- $(b^m)^n = b^{mn}$
- $b^n = e^{n \log b}$

Logarithm

Logarithm is inverse function of exponential

$$\log_b n = x \quad \Leftrightarrow \quad b^x = n$$

- $b > 0$ is *base*
- By default, $\log = \log_e$. Will often use $\log_2$.
- For any $a, b > 0$: $\log_b n = \dfrac{\log_a n}{\log_a b}$. So $\log_b n = O(\log n)$.
- $\log(mn) = \log m + \log n$

Asymptotic Notation: Rules

Rules to help simplify calculations.

- Multiplicative constants can be omitted: $365n^2$ becomes n^2
- n^a dominates n^b if $a > b$: for example, n^2 dominates n
- Any exponential dominates any polynomial: 2^n dominates n^{10}
- Polynomial dominates logarithm: $n^{0.1}$ dominates $(\log n)^{10}$

$n = O(n^2)$
$n^2 = \Omega(n)$

Exercise: Prove the above.

Basic Recursion

Let $f(n)$ be a function of $n \in \mathbb{N}$. Verify that:

1. If $f(n) = f(n-1) + 1$, then $f(n) = \Theta(n)$
2. If $f(n) = f(n-1) + n$, then $f(n) = \Theta(n^2)$
3. If $f(n) = 2f(n-1) + 1$, then $f(n) = \Theta(2^n)$
4. If $f(n) = f(\frac{n}{2}) + 1$, then $f(n) = \Theta(\log n)$

Basic Counting

Let S be a set with n elements.

- Factorial: $n! = n \cdot (n-1) \cdots 2 \cdot 1$
  This is the number of permutations of the elements of S
- 2^n = the number of subsets (of any size) of S
- n^2 = the number of ordered pairs of elements of S $= \{(i,j) : i,j \in S\}$
- $\binom{n}{2} = \frac{n(n-1)}{2}$ = the number of unordered pairs of distinct elements of S $= \{\{i,j\} : i,j \in S,\ i \neq j\}$
- $\lceil \log_2 n \rceil$ = the number of times to halve n to reach 1

Quiz: Sorting

How long does it take to sort a list of n elements?

(a) $\Theta(n^2)$

(b) $\Theta(n \log n)$

(c) $\Theta(n)$

(d) $\Theta(\log n)$

Quiz: Sorting

How long does it take to sort a list of n elements?

(a) $\Theta(n^2)$

(b) $\Theta(n \log n)$

(c) $\Theta(n)$

(d) $\Theta(\log n)$

Answer: $\Theta(n \log n)$.

Plan

Review: Asymptotics

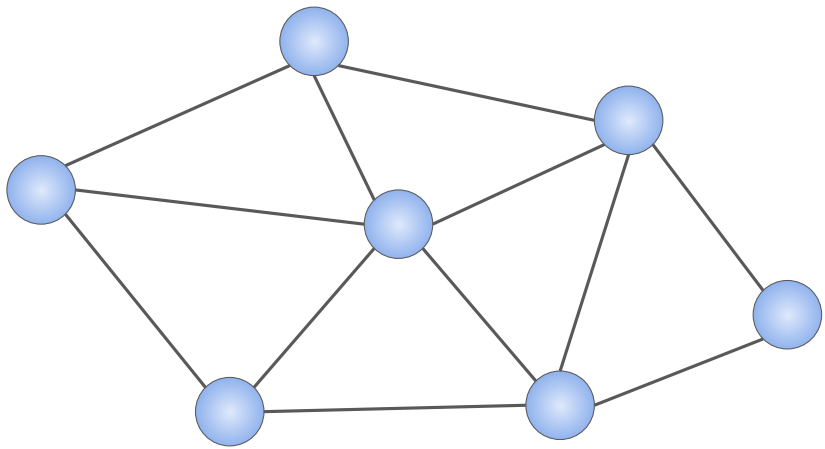
Review: Graphs

Stable Matching Problem

Graph

A **graph** $G = (V, E)$ is composed of:

- A collection of vertices $V = \{1, \ldots, n\}$
- A collection of edges $E \subseteq V \times V$ between some vertices



Graphs

Let $G = (V, E)$ be a graph.

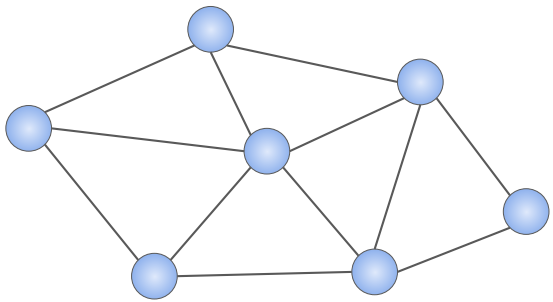
By default (unless indicated otherwise), we assume:

- Edges are *undirected*: If $(i, j) \in E$, then $(j, i) \in E$.

  Also write $\{i, j\} \in E$.

- No self-loops: $(i, i) \notin E$.

- Edges are *unweighted*: All $(i, j) \in E$ have weight 1.

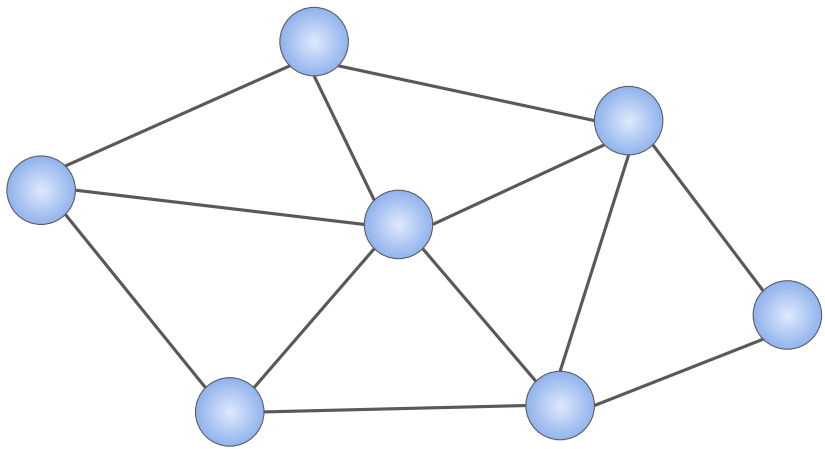


Graph

Let $G = (V, E)$ be a graph on n vertices

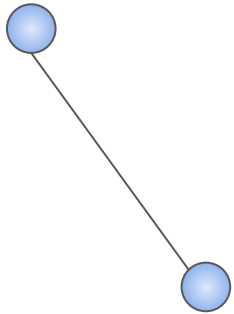
Maximum number of edges is

$$\binom{n}{2} = \frac{n!}{(n-2)!2!} = \frac{n(n-1)}{2}$$

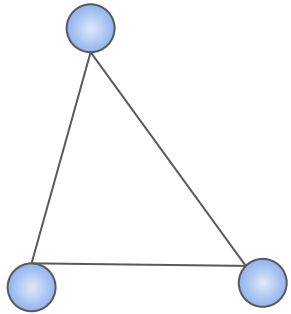


Complete Graph

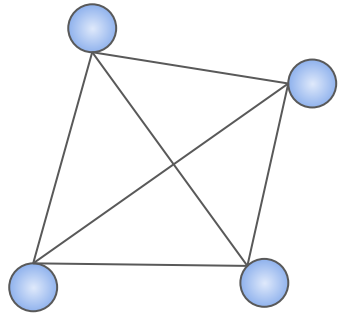
Let K_n denote the **complete graph** on n vertices



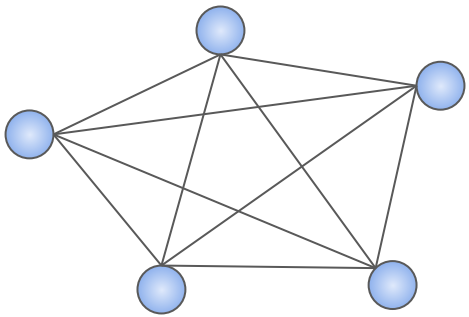
K_2



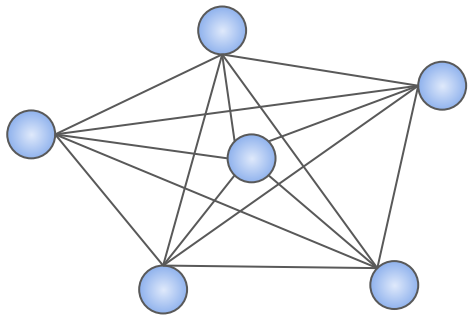
K_3



K_4



K_5



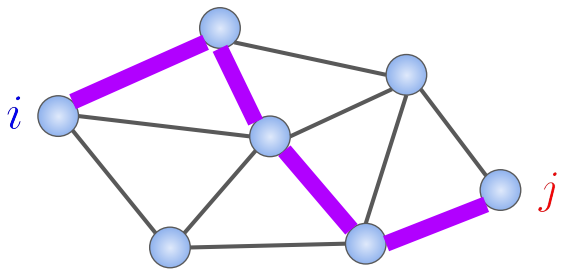
K_6

Connected Graph

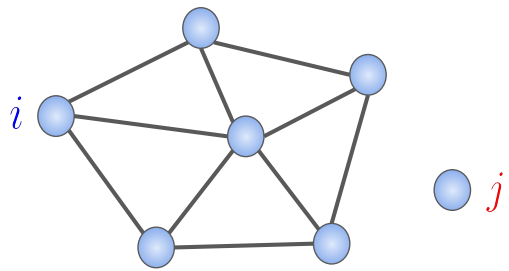
A graph $G = (V, E)$ is **connected** if any two nodes $i, j \in V$ are connected via a sequence of connecting edges:

$$(i_0, i_1),\ (i_1, i_2),\ \ldots,\ (i_{n-1}, i_n) \in E$$

with $i_0 = i$ and $i_n = j$



connected



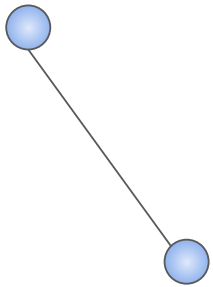
not connected

Cycle Graph

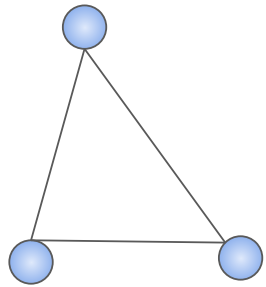
$C_n = (V, E)$, $V = \{1, \ldots, n\}$

$E = \{\{i, i+1\} : 1 \le i \le n-1\} \cup \{\{1, n\}\}$

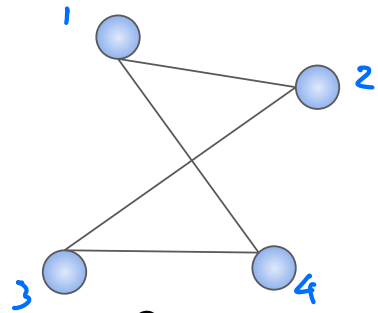
Let C_n denote the cycle graph on n vertices



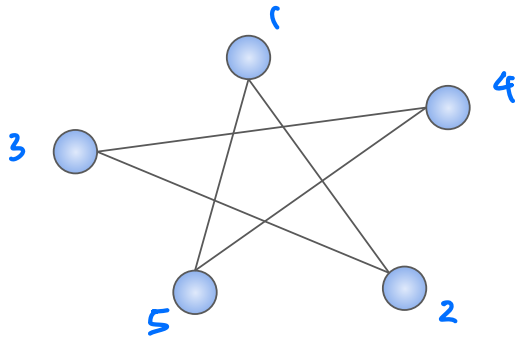
C_2



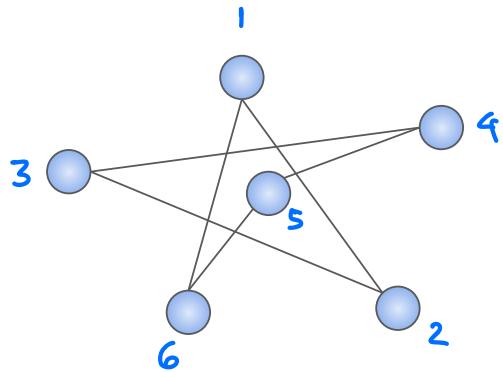
C_3



C_4



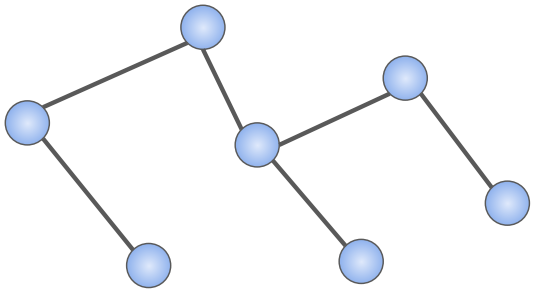
C_5



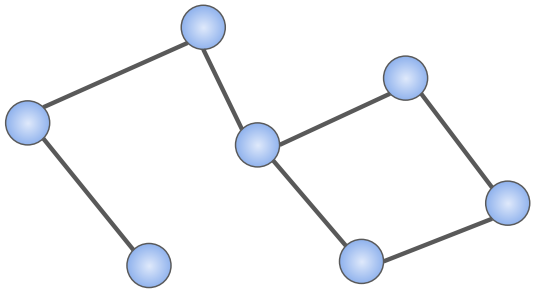
C_6

Tree

A **tree** $G = (V, E)$ is a connected graph with no cycles



tree



not a tree

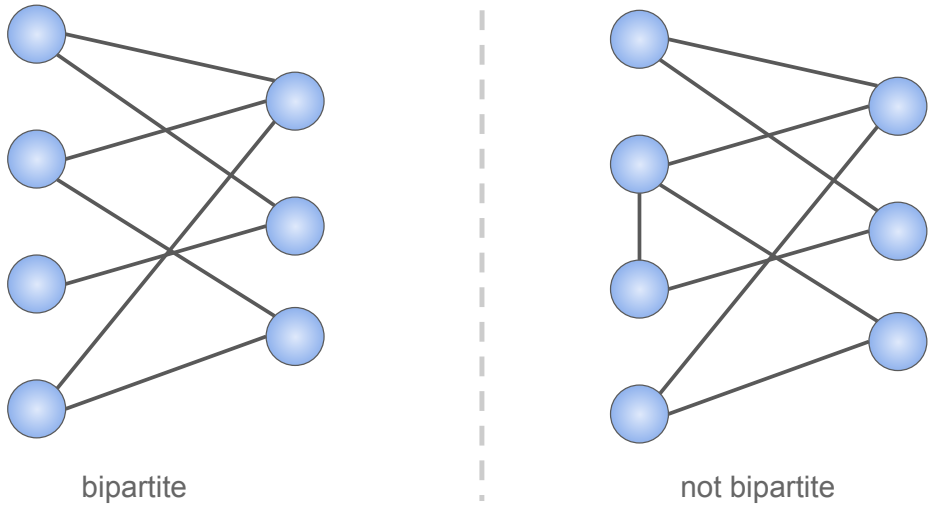
- Also known as spanning tree
- A tree on n vertices has $n - 1$ edges (PS1)
- What if G has no cycles, but is not connected? Forest

Bipartite Graph

A graph $G = (V, E)$ is **bipartite** if the vertices can be partitioned into disjoint subsets:

$$V = A \cup B$$

such that all edges connect between A and B
(no edges within A, no edges within B)



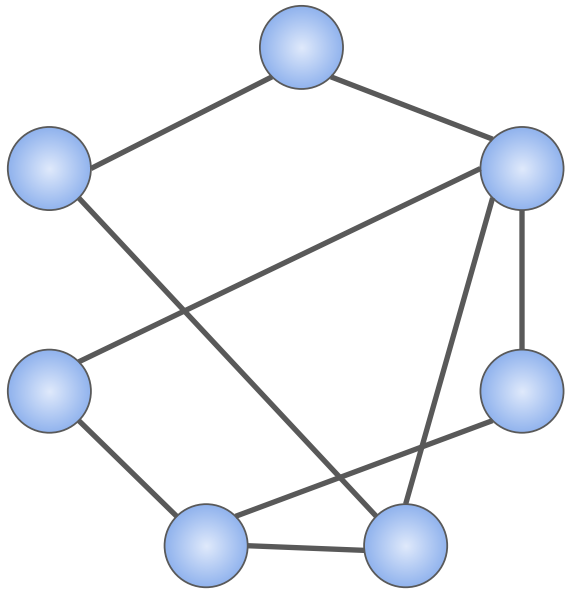
- G is bipartite if and only if it has no cycles of odd length (PS1)

Quiz: Bipartite

Is the following graph bipartite?

(a) Yes

(b) No



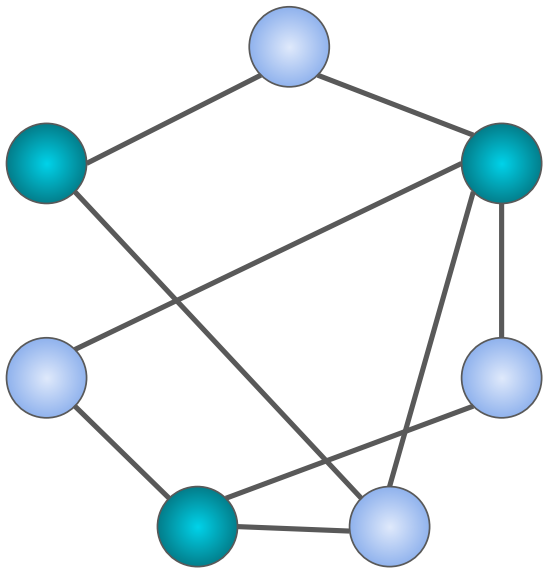
Quiz: Bipartite

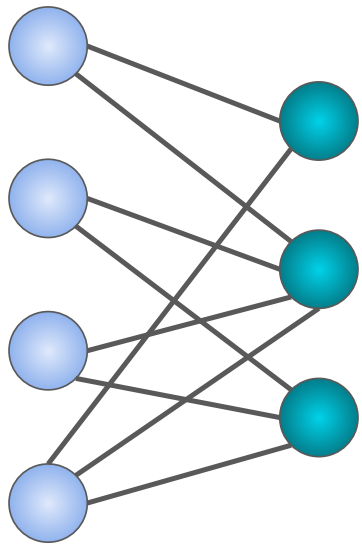
Is the following graph bipartite?

(a) Yes

(b) No

Answer: (a) Yes



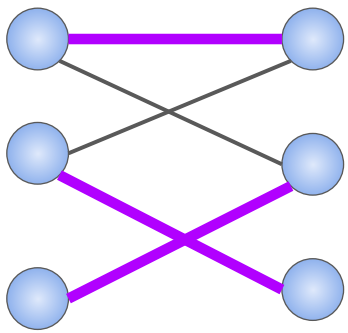


Matching in Bipartite Graph

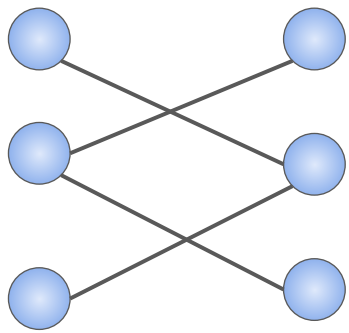
Let $G = (V, E)$ be a bipartite graph on $2n$ vertices, with n vertices on each side: $V = A \cup B$ with $|A| = |B| = n$

A **perfect matching** on G is a choice of n edges from E, such that no two edges in the matching share a common vertex

(each vertex in V is included in exactly one edge in the matching)



perfect matching

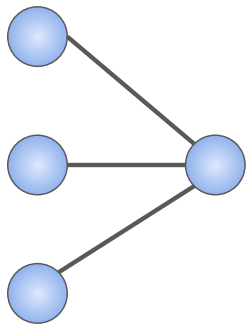


no perfect matching

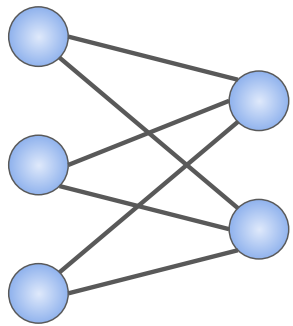
- When does a perfect matching exist? Hall's Marriage Theorem

Complete Bipartite Graph

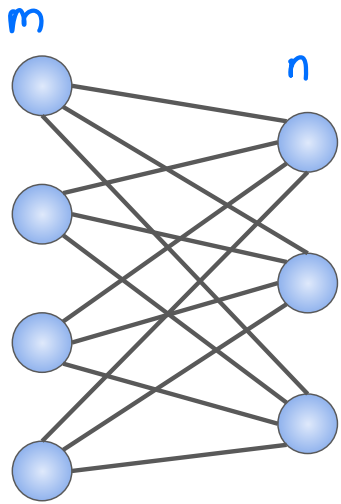
Let $K_{m,n}$ denote the complete bipartite graph on $m + n$ vertices, with m, n vertices on each side



$K_{3,1}$



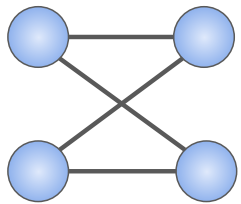
$K_{3,2}$



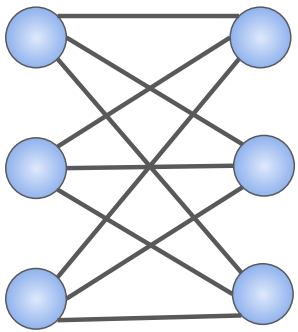
$K_{4,3}$

Matching in Complete Bipartite Graph

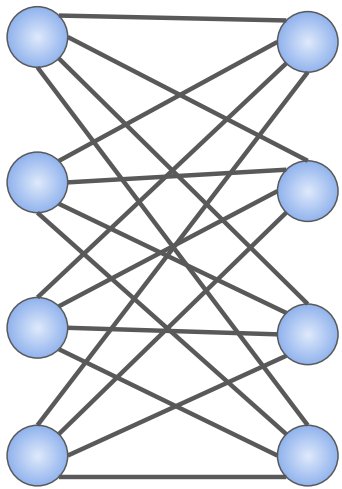
Let $K_{n,n}$ denote the complete bipartite graph on $2n$ vertices
Then a perfect matching exists. In fact, many of them.



$K_{2,2}$



$K_{3,3}$



$K_{4,4}$

Quiz: Matching in Bipartite Graph

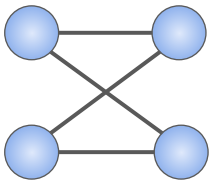
How many perfect matchings are there in $K_{n,n}$?

(a) n^n

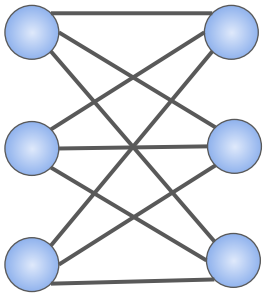
(b) 2^n

(c) $n!$

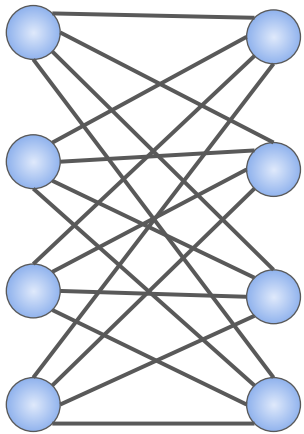
(d) n^2



$K_{2,2}$



$K_{3,3}$



$K_{4,4}$

Quiz: Matching in Bipartite Graph

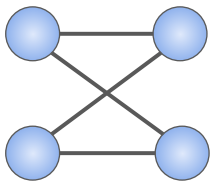
How many perfect matchings are there in $K_{n,n}$?

(a) n^n

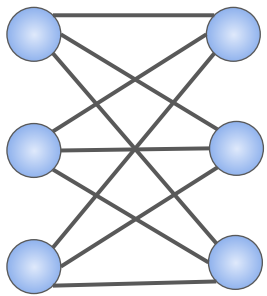
(b) 2^n

(c) $n!$

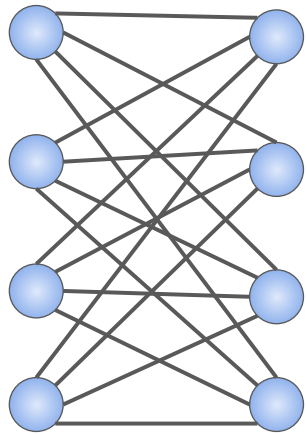
(d) n^2



$K_{2,2}$



$K_{3,3}$



$K_{4,4}$

Answer: (c) $n!$

Plan

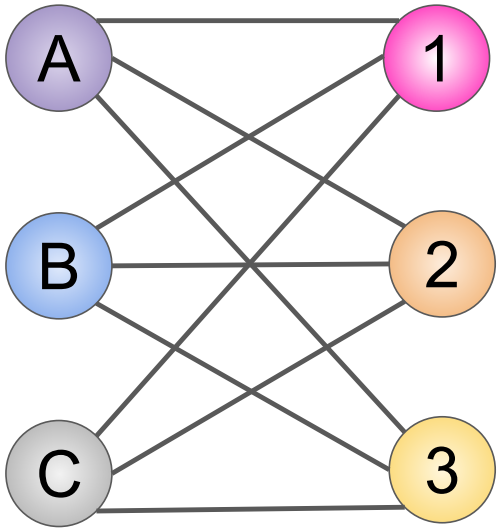
Review: Asymptotics

Review: Graphs

Stable Matching Problem

Matching with Preferences

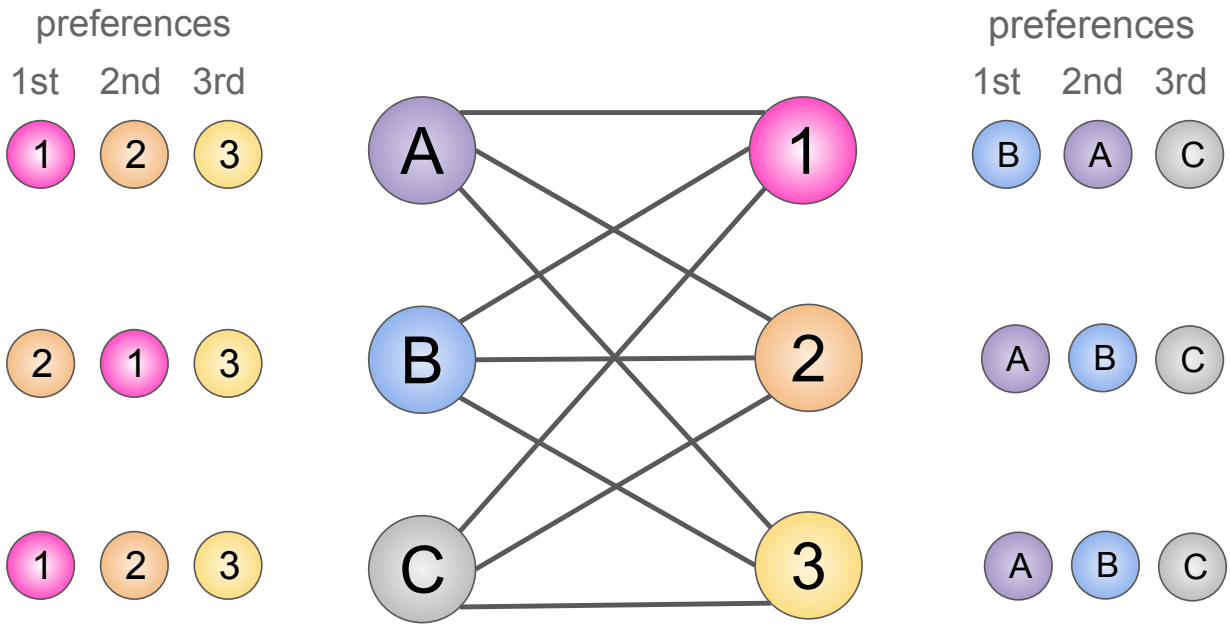
Suppose we have two groups of n nodes each, connected via $K_{n,n}$



We want to find a perfect matching between them

Matching with Preferences

Suppose each node has a preference over the n nodes in the other group (who they'd like to be matched with)



Question: Can we find a matching that makes "everyone happy"?

Motivation and history

- College admissions: How to allocate admissions to yield a certain number of accepted students?
  - Gale and Shapley, "College Admissions and the Stability of Marriage", *American Mathematical Monthly*, **69** (1962), `https://www.jstor.org/stable/2312726`
  - Described an algorithm in terms of marriage proposals between n men and n women
  - For simplicity and to maintain consistency with textbook, we also use men and women in describing the algorithm
  - Paper also considered single-group variant, but technically different
- Used since 1952 for National Resident Matching Program to match hospitals and medical residents

Nobel Prize

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012

© The Nobel Foundation. Photo: U. Montan

Alvin E. Roth

Prize share: 1/2

© The Nobel Foundation. Photo: U. Montan

Lloyd S. Shapley

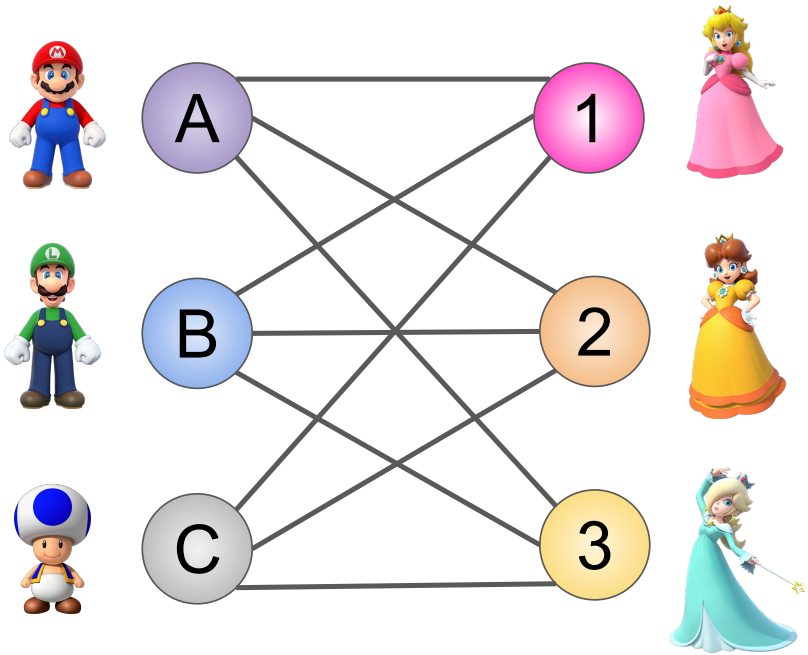
Prize share: 1/2

The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2012 was awarded jointly to Alvin E. Roth and Lloyd S. Shapley "for the theory of stable allocations and the practice of market design."

`https://www.nobelprize.org/prizes/economic-sciences/2012/summary/`

Matching with Preferences

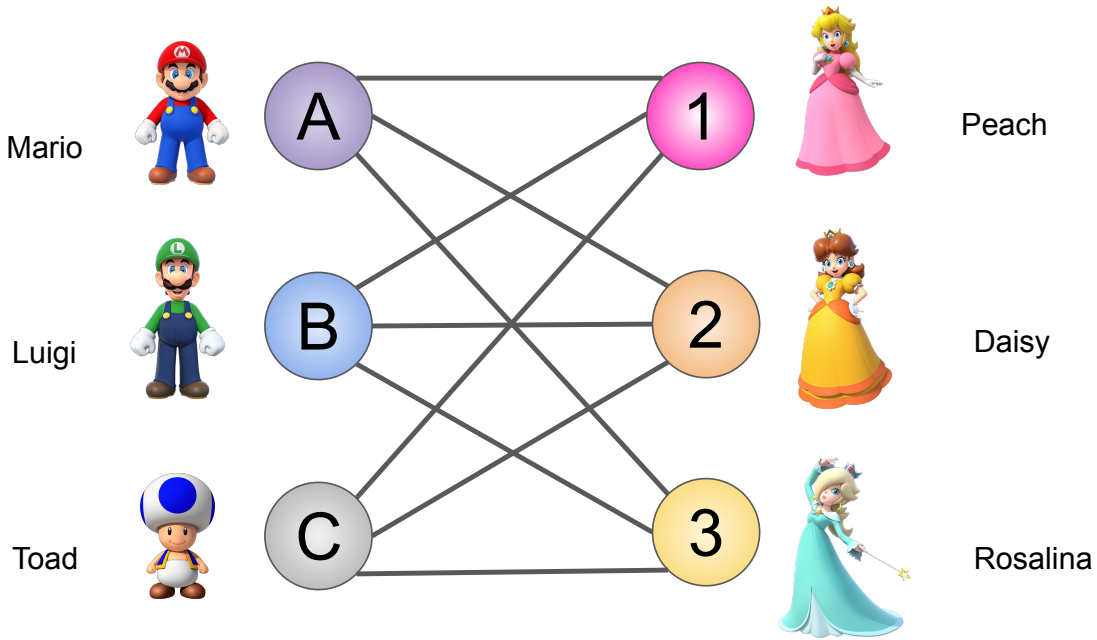
We have n men and n women, and we want to match them.



https://mario.nintendo.com/characters/

Matching with Preferences

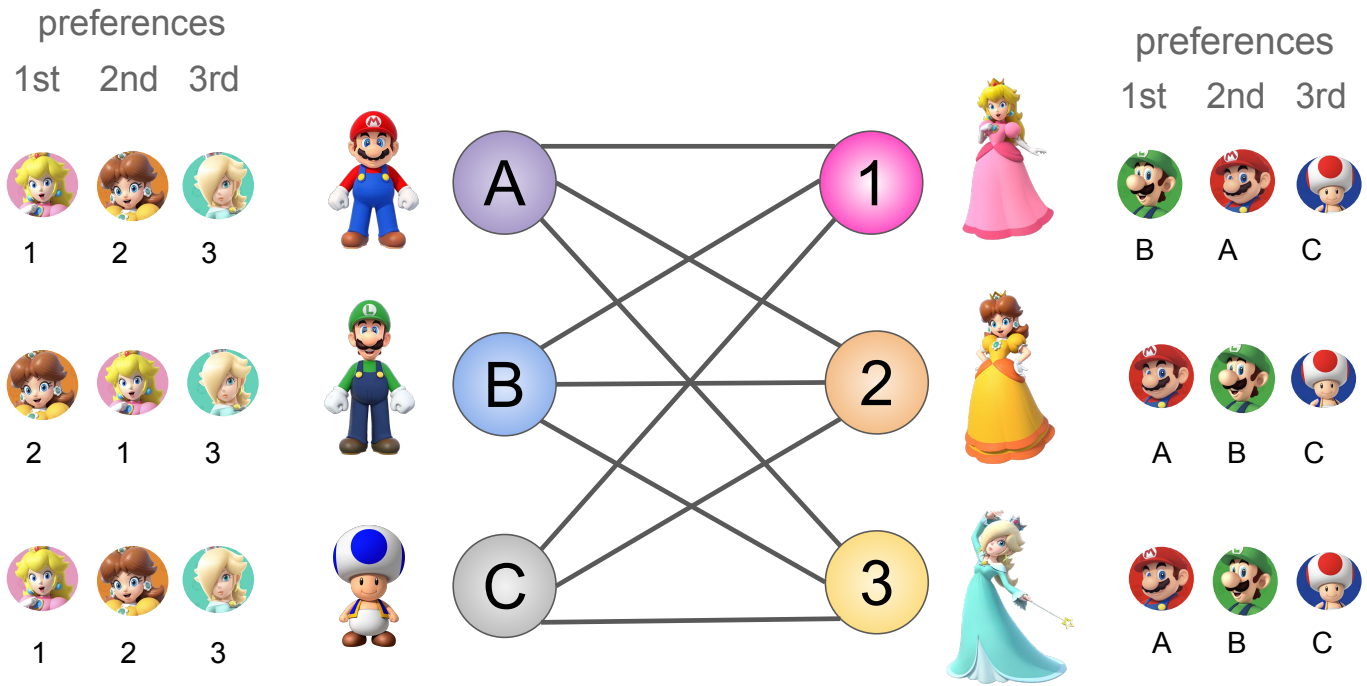
We have n men and n women, and we want to match them.



`https://mario.nintendo.com/characters/`

Matching with Preferences

Each person has a preference over everyone in the opposite group.



Q: Can we find the best matching that makes everyone happy?

Matching with Preferences

What makes a good matching? Many possible answers:

- Maximize number of first-ranked choices
- Minimize number of last-ranked choices
- Minimize sum of rank of choices (max average happiness)
- We will use: Stability

Stable Matching

- A matching is **unstable** if there is a man and a woman who prefer each other to their current partners.

  We call such a pair a *rogue couple*.

- A matching is **stable** if it has no rogue couples.

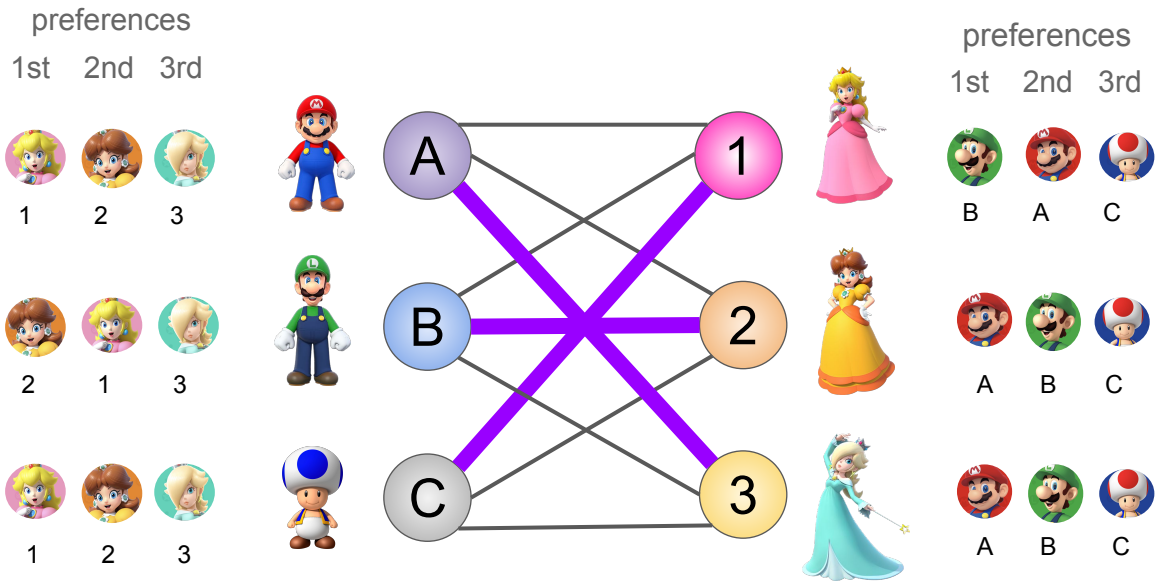
Stable Matching

- A matching is **unstable** if there is a man and a woman who prefer each other to their current partners.

  We call such a pair a *rogue couple*.

- A matching is **stable** if it has no rogue couples.

Is the following matching stable?



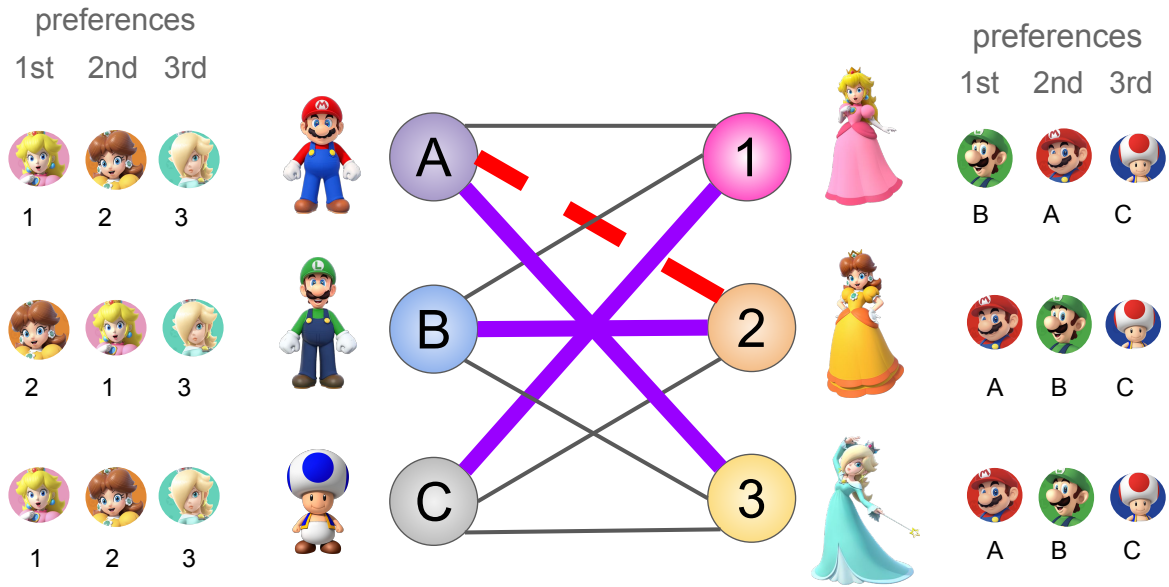
Stable Matching

- A matching is **unstable** if there is a man and a woman who prefer each other to their current partners.

  We call such a pair a *rogue couple*.

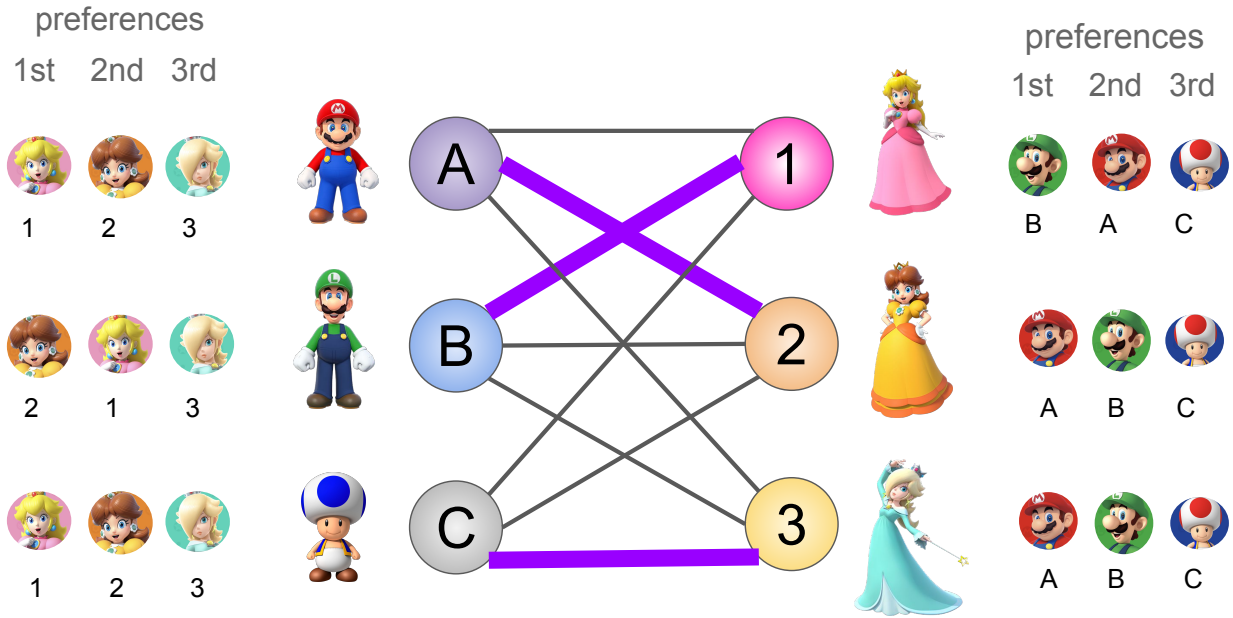
- A matching is **stable** if it has no rogue couples.

Is the following matching stable? No. (A, 2) is a rogue couple.



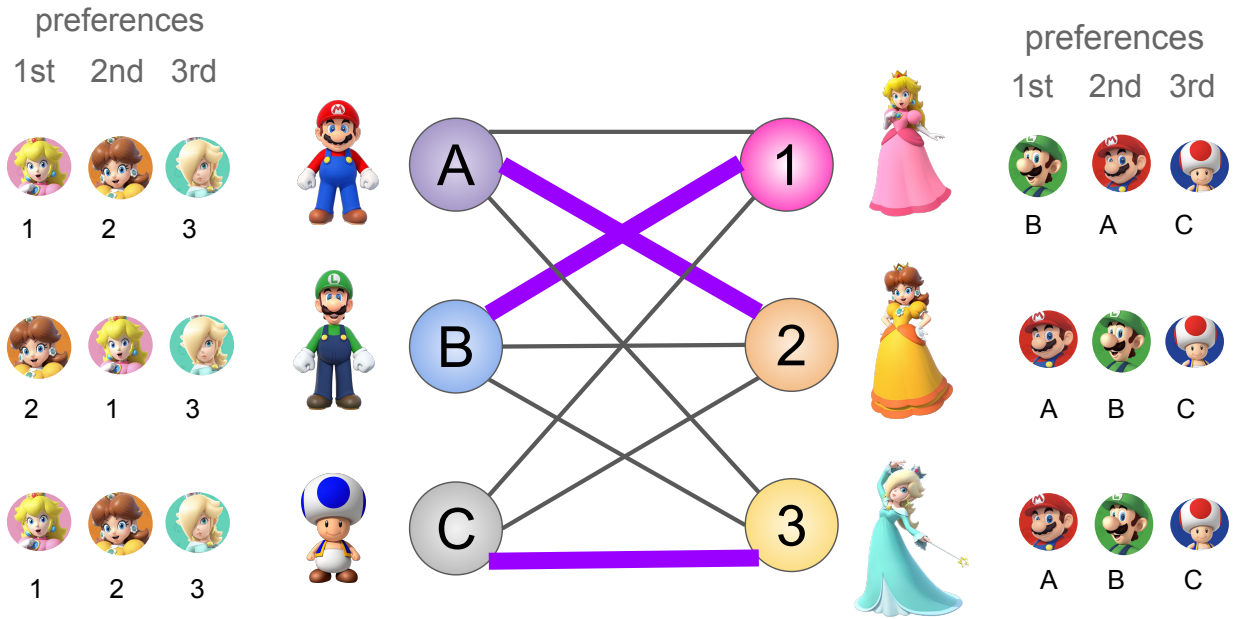
Stable Matching

Is the following matching stable?



Stable Matching

Is the following matching stable? Yes.

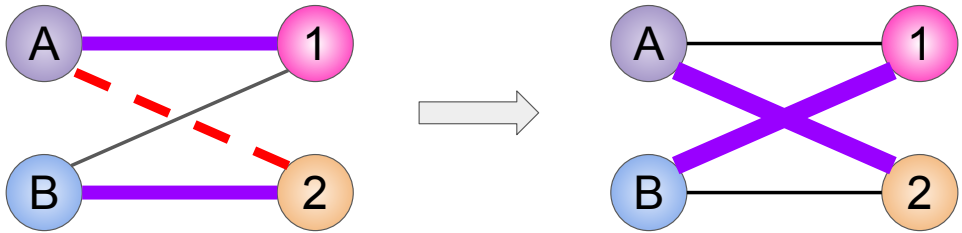


Existence of Stable Matching

Question: Does a stable matching always exist?

Intuitively yes, since we can start with any matching and make it more and more stable as follows:

- If there is a rogue couple, modify the current pairing so that they are together. Repeat until there is no more rogue pairing.



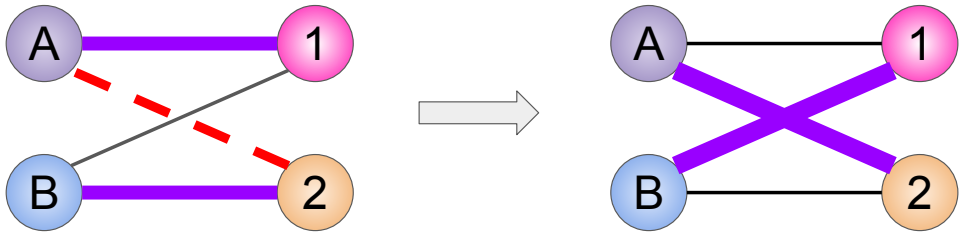
Then the procedure above will result in a stable pairing.

Existence of Stable Matching

Question: Does a stable matching always exist?

Intuitively yes, since we can start with any matching and make it more and more stable as follows:

- If there is a rogue couple, modify the current pairing so that they are together. Repeat until there is no more rogue pairing.



Then the procedure above will result in a stable pairing.

Quiz: Is this reasoning correct?

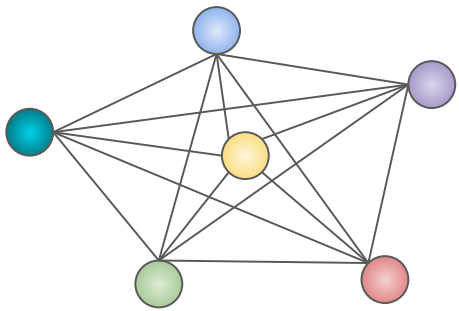
(a) Yes

(b) No

Stable Roommates Problem

The single-group version of the Stable Matching Problem.

- We have $2n$ people, want to pair them into n roommates.
  Each person can be paired with anyone else.
- Each person has a preference over all other $2n - 1$ people.



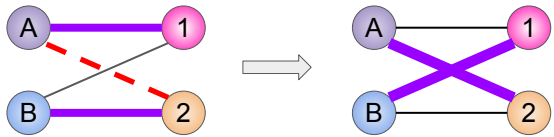
Question: Is there always a stable matching?

Stable Roommates Problem

Question: Is there always a stable matching of roommates?

Not obvious. But:

- Suppose our approach of iteratively matching up rogue couples indeed *did* work.

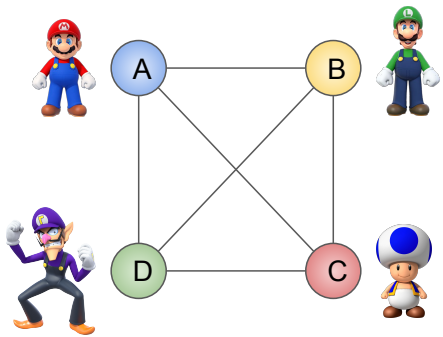


- Since this approach does not exploit the existence of different genders, we can apply it to the roommates problem.
- Thus, we conclude there must always exist a stable pairing for roommates!

Stable Roommates Problem

But a stable matching of roommates may not exist!

Counterexample: (where $*$ can be anything)



	Preference		
A	B	C	D
B	C	A	D
C	A	B	D
D	*	*	*

Exercise: Verify that there is no stable matching.

Stable Roommates Problem

We showed stable matching of roommates may *not* exist. Therefore, from this argument:

- Suppose our approach of iteratively matching up rogue couples indeed *did* work.

- Since this approach does not exploit the existence of different genders, we can apply it to the roommates problem.
- Thus, we conclude there must always exist a stable pairing for roommates!

We conclude that: Our approach (of matching up rogue couples) does *not* work to produce a stable matching.

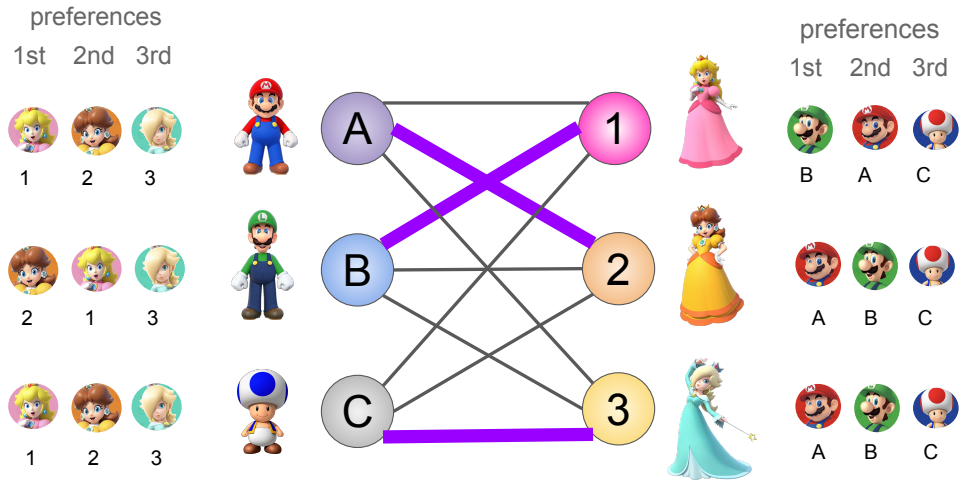
Lesson: Any proof that a stable matching always exists in the Stable Matching Problem (with two groups) *must* use the fact that there are two groups in a crucial way.

Back to Stable Matching Problem

Suppose we have n men and n women with preferences.

Theorem: There always exists a stable matching.

Proof: We will describe an algorithm to construct a stable matching, and prove it is correct. □



Stable Matching Algorithm

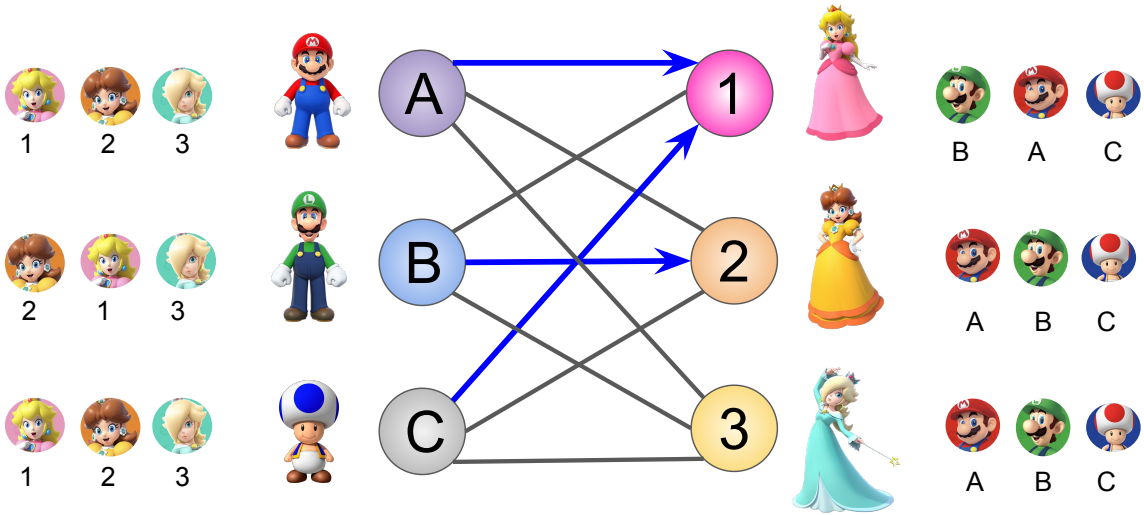
The algorithm proceeds in stages (days). In each day:

1. **Every Morning:** Each man proposes to the most preferred woman on his list who has not yet rejected him
2. **Every Afternoon:** Each woman collects all the proposals she received in the morning. She responds:
   - To the man she likes best: "Maybe, come back tomorrow." (She now has him on a *string*)
   - To the others, she rejects: "No, never."
3. **Every Evening:** Each rejected man crosses off the woman who rejected him from his list.

The above is repeated until each woman has a man on a string.
On this day, each woman marries the man she has on a string.

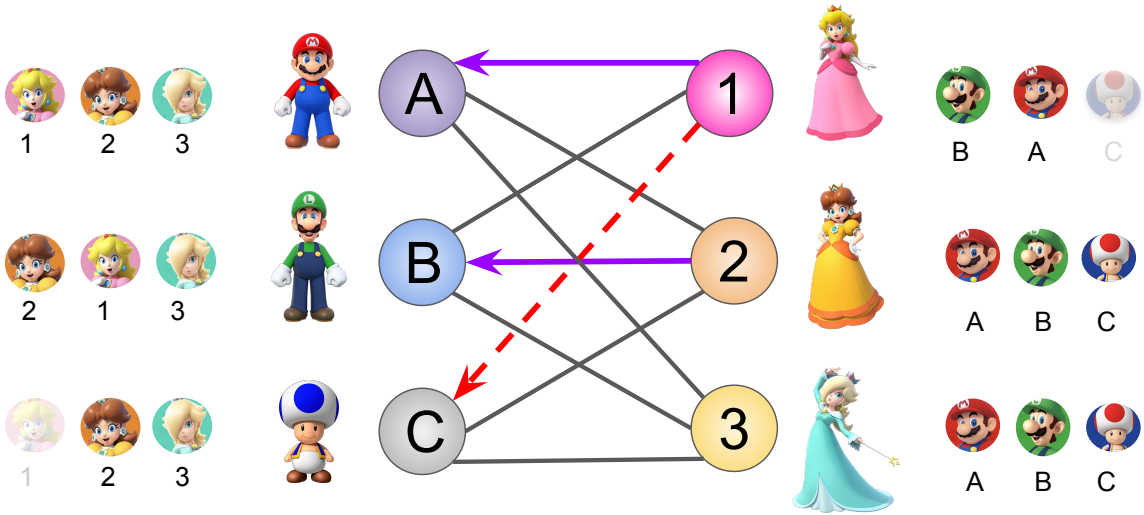
Running the Algorithm: Day 1 Morning

	Day 1					
	am					
A	**1**					
B	**2**					
C	**1**					



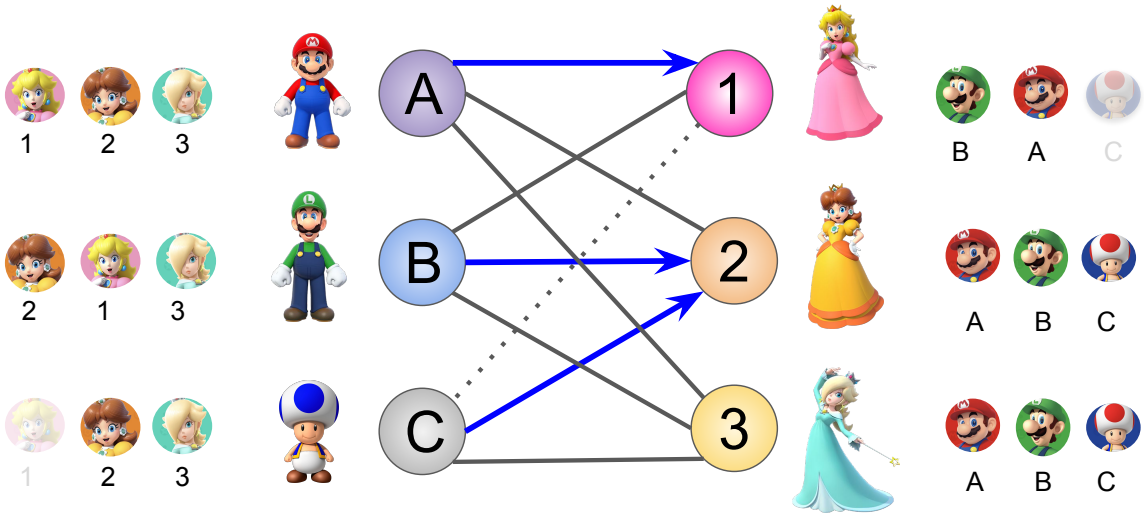
Running the Algorithm: Day 1 Evening

	Day 1					
	am	pm				
A	**1**	**1**				
B	**2**	**2**				
C	**1**	**–**				



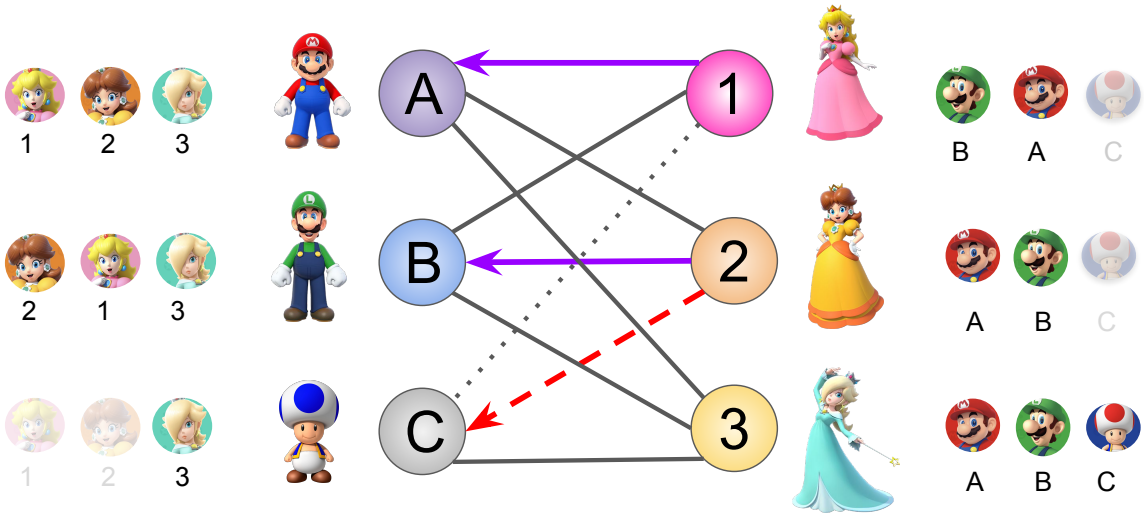
Running the Algorithm: Day 2 Morning

	Day 1		Day 2			
	am	pm	am			
A	**1**	**1**	**1**			
B	**2**	**2**	**2**			
C	**1**	**–**	**2**			



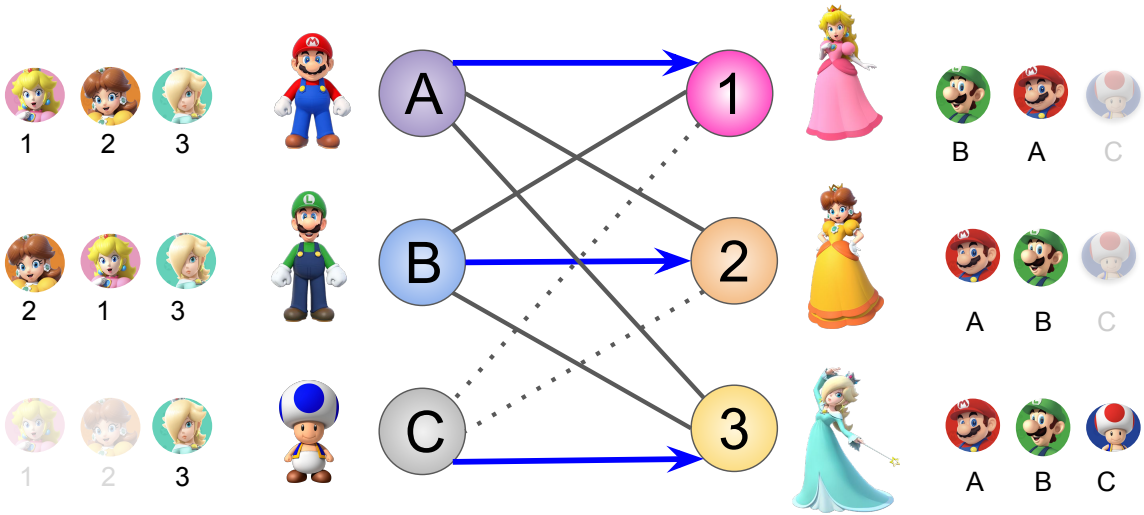
Running the Algorithm: Day 2 Evening

	Day 1		Day 2			
	am	pm	am	pm		
A	**1**	**1**	**1**	**1**		
B	**2**	**2**	**2**	**2**		
C	**1**	**–**	**2**	**–**		



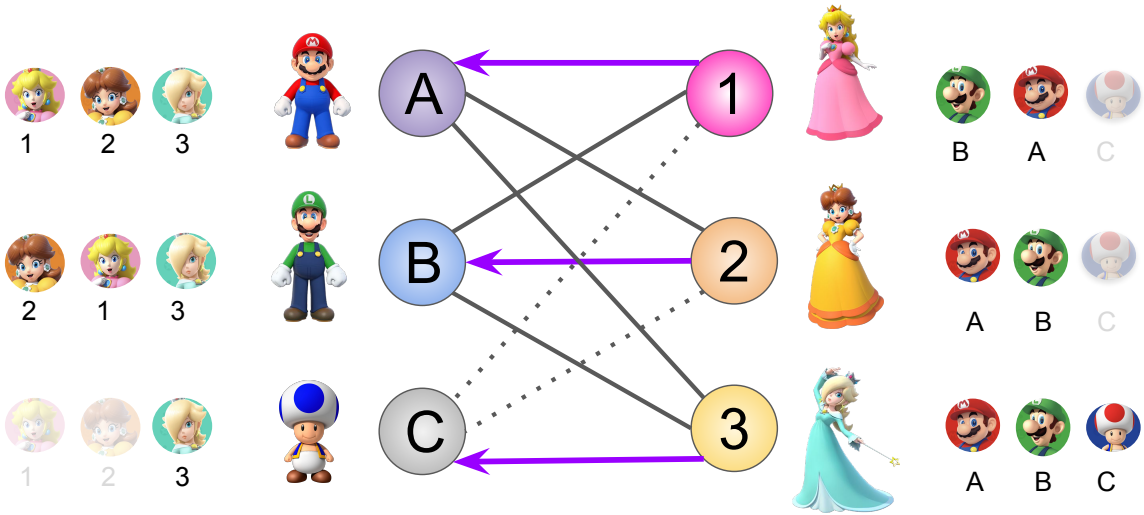
Running the Algorithm: Day 3 Morning

	Day 1		Day 2		Day 3	
	am	pm	am	pm	am	
A	**1**	**1**	**1**	**1**	**1**	
B	**2**	**2**	**2**	**2**	**2**	
C	**1**	**–**	**2**	**–**	**3**	



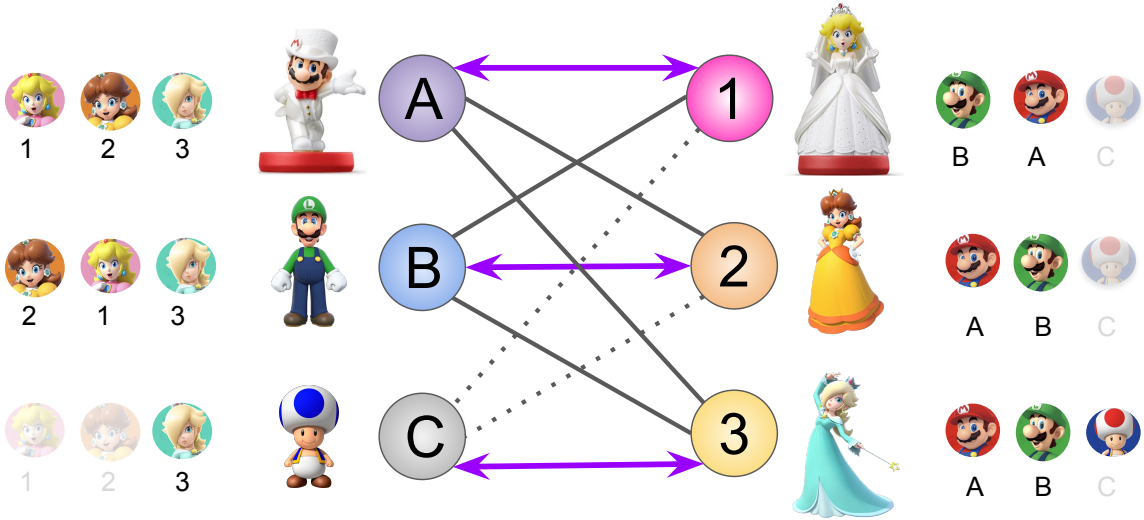
Running the Algorithm: Day 3 Evening

	Day 1		Day 2		Day 3	
	am	pm	am	pm	am	pm
A	**1**	**1**	**1**	**1**	**1**	**1**
B	**2**	**2**	**2**	**2**	**2**	**2**
C	**1**	**–**	**2**	**–**	**3**	**3**



Running the Algorithm: Success

	Day 1		Day 2		Day 3	
	am	pm	am	pm	am	pm
A	**1**	**1**	**1**	**1**	**1**	**1**
B	**2**	**2**	**2**	**2**	**2**	**2**
C	**1**	**–**	**2**	**–**	**3**	**3**



Stable Matching Algorithm

Consider the Stable Matching Problem with n men and n women.

Lemma: The Stable Matching Algorithm halts.
Furthermore, it uses at most n^2 (unique) proposals.

Stable Matching Algorithm

Consider the Stable Matching Problem with n men and n women.

Lemma: The Stable Matching Algorithm halts.
Furthermore, it uses at most n^2 (unique) proposals.

Proof: On each day that the algorithm runs, at least one man must eliminate a woman from his list (otherwise the halting condition for the algorithm would be invoked). Since there are n men, and each list has n elements, there are at most n^2 possible (unique) proposals. In particular, the algorithm must terminate in at most n^2 days. □

Stable Matching Algorithm

Consider the Stable Matching Problem with n men and n women.

Lemma: The Stable Matching Algorithm halts.
Furthermore, it uses at most n^2 (unique) proposals.

Proof: On each day that the algorithm runs, at least one man must eliminate a woman from his list (otherwise the halting condition for the algorithm would be invoked). Since there are n men, and each list has n elements, there are at most n^2 possible (unique) proposals. In particular, the algorithm must terminate in at most n^2 days. □

- Note: In fact, it uses at most $n^2 - n + 1$ proposals (PS1).
- Question: Is the output a stable matching? Is it even a matching?

Next Lecture

- Analysis of Stable Matching Problem
- Graph problems
- Reading: 3.2 – 2.6
- PS 1 due on Tuesday, Feb 1, at 2:30 pm

Please fill survey: `https://forms.gle/NS3LzK7MHzSbU7JR6`