COMP 251

Algorithms & Data Structures (Winter 2021)

Graphs – Bipartite Graphs

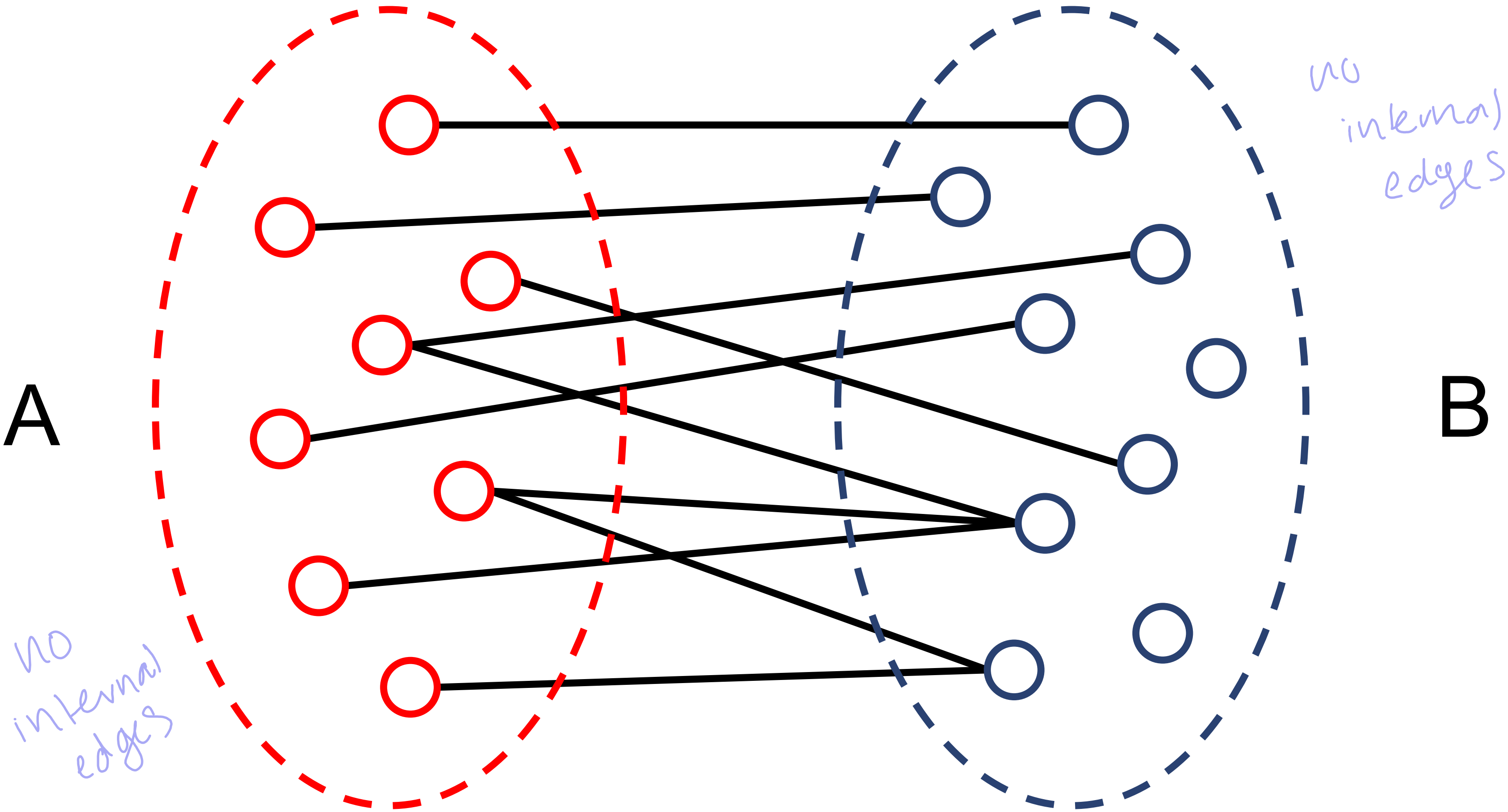
School of Computer Science
McGill University

Slides of (Comp321 ,2021), Langer (2014), Kleinberg & Tardos, 2005 & Cormen et al., 2009, Jaehyun Park' slides CS 97SI, Top-coder tutorials, T-414-AFLV Course, Programming Challenges books, slides from D. Plaisted (UNC) and Comp251-Fall McGill, P. Beame (UofW) & K. Wayne (Princeton)

Outline

- Graphs.
  - Introduction.
  - Topological Sort. / Strong Connected Components
  - Network Flow 1.
    - Introduction
    - Ford-Fulkerson
  - Network Flow 2.
    - Min-cuts
  - Shortest Path.
  - Minimum Spanning Trees.
  - Bipartite Graphs.

Bipartite Graphs - Problem



Vertices are partitioned into 2 sets.

All edges cross the sets.

Bipartite Graphs - Examples

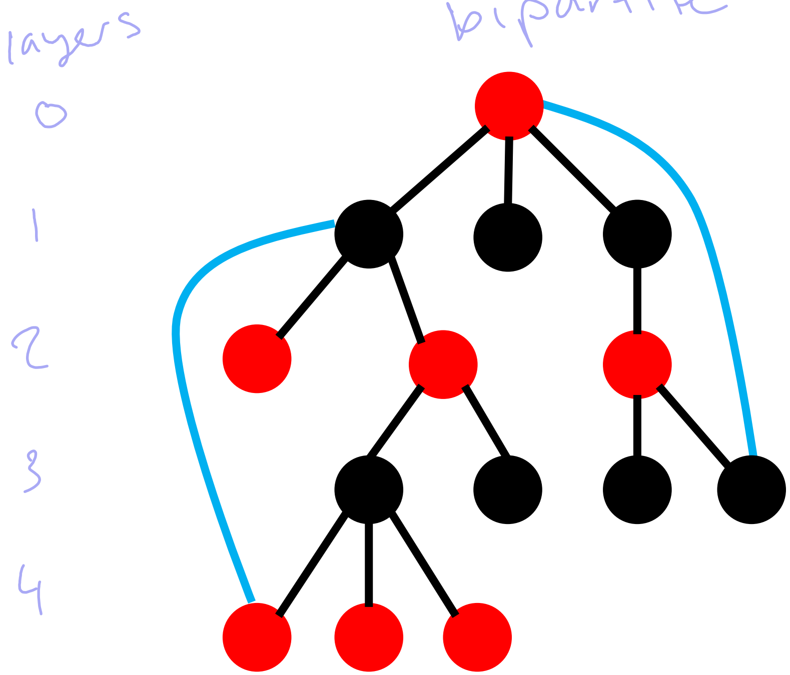
A		B
Courses	registration	Students
Candidates	employment	Companies
People	Have read/seen	Books/Movies

Bipartite Graphs – Is it a bipartite graph?

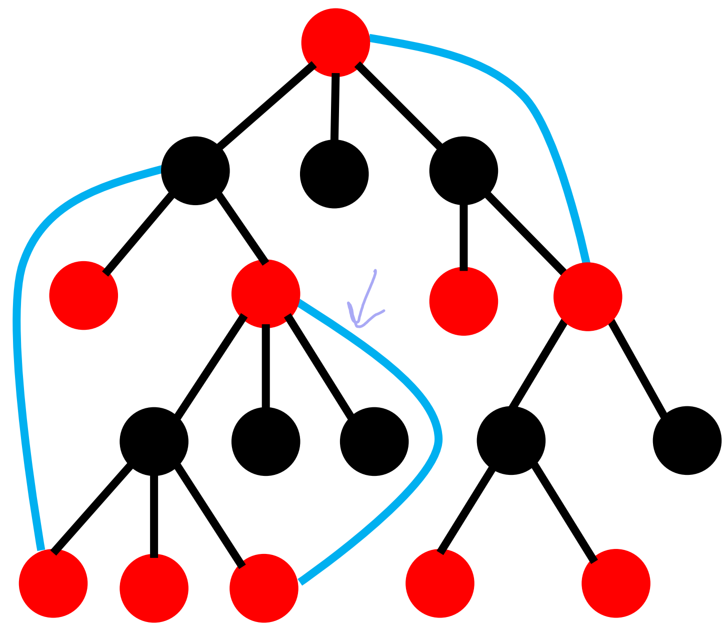
Assuming G=(V,E) is an undirected connected graph.

1. Run DFS and use it to build a DFS tree.
2. Color vertices by layers (e.g. red & black) with a bit 1/0
3. If all non-tree edges join vertices of different color, then the graph is bipartite.

bipartite

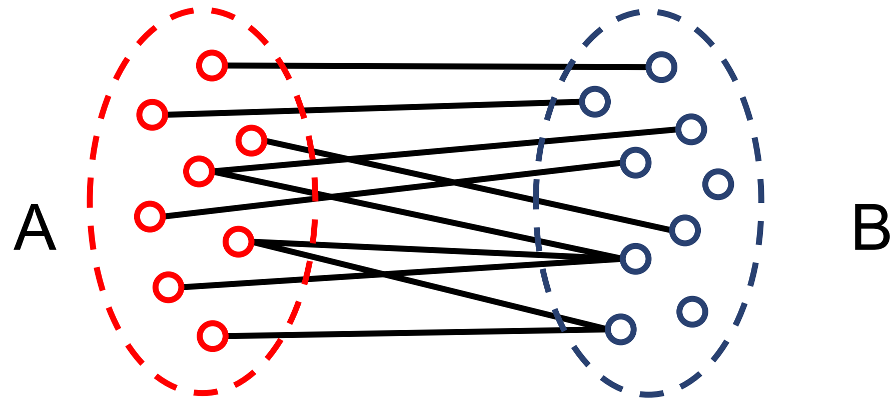


not bipartite



Bipartite matching

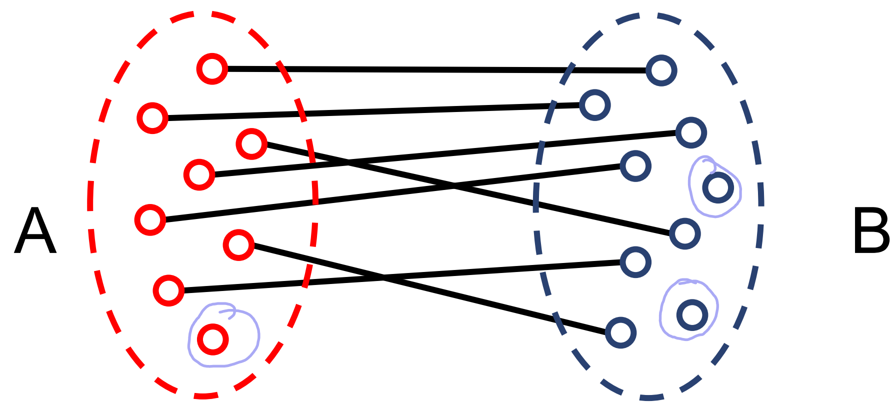
Consider an undirected bipartite graph.



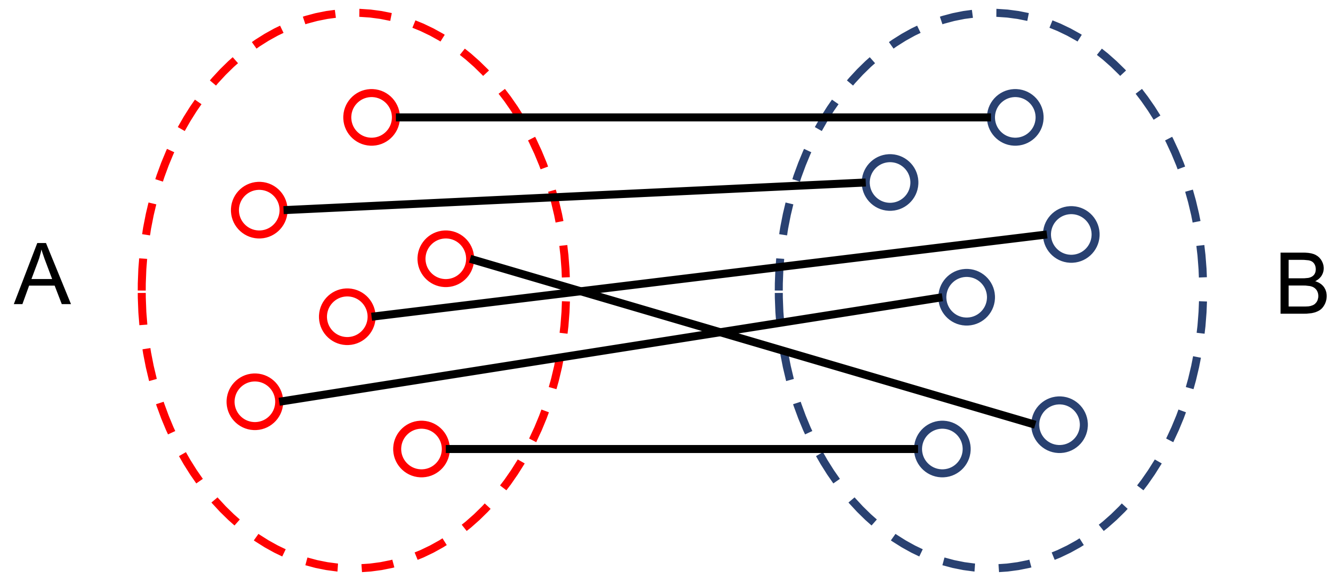
no internal edges!

A matching is a subset of the edges { (α, β) } such that no two edges share a vertex.

1-to-1 relationship



Perfect matching

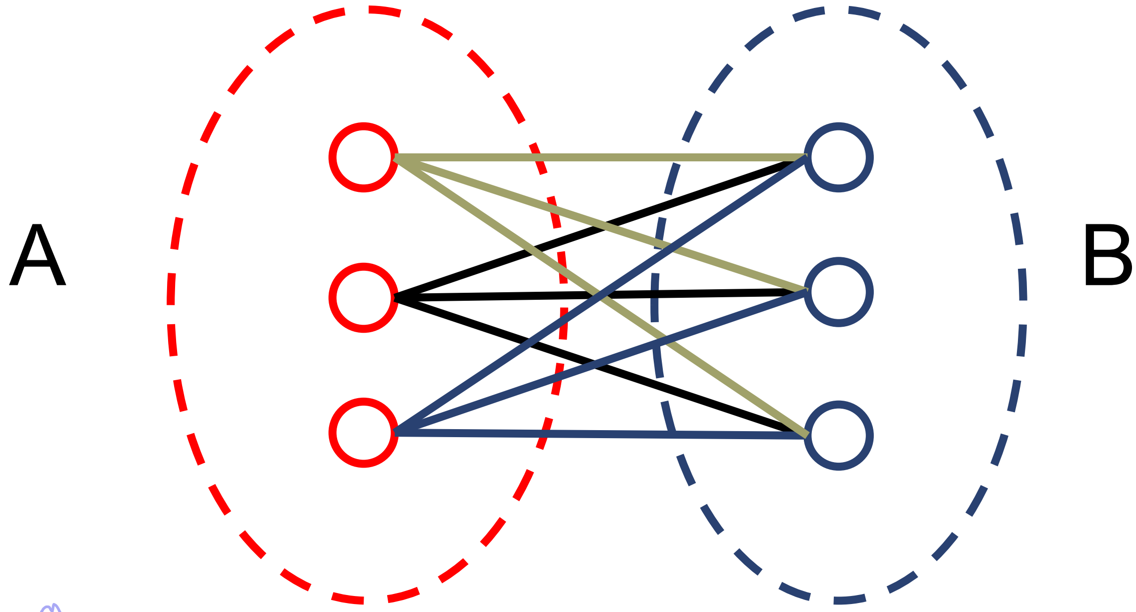


every element in A has a match in B

Suppose we have a bipartite graph with *n* vertices in each A and B.
A **perfect matching** is a matching that has *n* edges.

Note: It is not always possible to find a perfect matching.

Complete bipartite graph



everything connected with everything

A complete bipartite graph is a bipartite graph that has an edge for every pair of vertices (α, β) such that α∈A, β∈B.

The algorithm of happiness

Resident matching program

- **Goal:** Given a set of preferences among hospitals and medical school students, design a self-reinforcing admissions process. - doesn't allow rematching
- **Unstable pair:** applicant **x** and hospital **y** are unstable if:
  - **x** prefers **y** to their assigned hospital.
  - **y** prefers **x** to one of its admitted students.
- **Stable assignment:** Assignment with no unstable pairs.
  - Natural and desirable condition.
  - Individual self-interest will prevent any applicant/hospital deal from being made behind the scenes.

Stable matching problem

Goal: Given **n** elements of **A** and **n** elements of **B**, find a "suitable" matching. Participants rate members of opposite set:

- Each element of A lists elements of B in order of preference from best to worst.
- Each element of B lists elements of A in order of preference from best to worst.

A's preferences

	1st	2nd	3rd
Xavier	Alphabet	Baidu	Campbell
Yulia	Baidu	Alphabet	Campbell
Zoran	Alphabet	Baidu	Campbell

B's preferences

	1st	2nd	3rd
Alphabet	Yulia	Xavier	Zoran
Baidu	Xavier	Yulia	Zoran
Campbell	Xavier	Yulia	Zoran

Stable matching problem

- **Context:** Candidates apply to companies.
- **Perfect matching:** everyone is matched with a single company.
  - Each candidate gets exactly one company.
  - Each company gets exactly one candidate.
- **Stability:** no incentive for some pair of participants to undermine assignment by joint action.
  - In matching **M**, an unmatched pair **α-β** is unstable if candidate **α** and company **β** prefer each other to current match.
  - Unstable pair **α-β** could each improve by "escaping".
- **Stable matching:** perfect matching with no unstable pairs.
- **Stable matching problem:** Given the preference lists of **n** candidates and **n** companies, find a stable matching (if one exists).

Stable matching problem – toy example 1

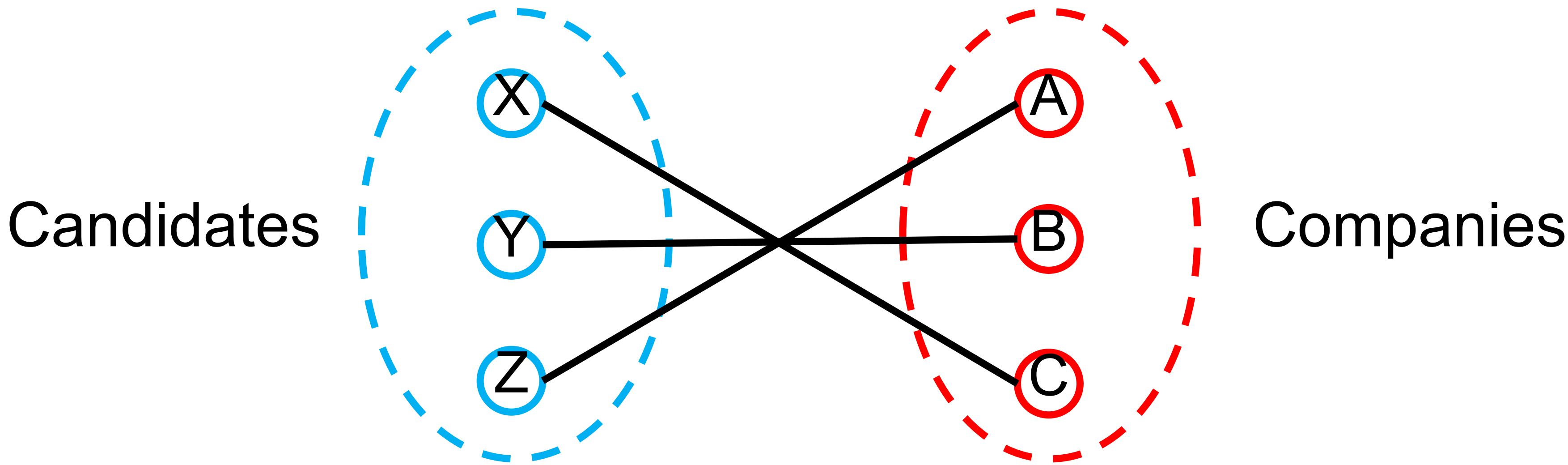
- Suppose we have a set of two applicants $\{a_1, a_2\}$, and a set of two companies $\{c_1, c_2\}$. The preference list is as follows.
  - a_1 prefers c_1 to c_2
  - a_2 prefers c_1 to c_2
  - c_1 prefers a_1 to a_2
  - c_2 prefers a_1 to a_2
- The list represents complete agreement: the applicants agree on the order of the companies, and the companies agree on the order of the applicants.
  - There is a unique stable matching, consisting of the pairs (a_1, c_1) and (a_2, c_2).
  - The other perfect matching (a_2, c_1) and (a_1, c_2), would not be a stable matching, because the pair (a_1, c_1) would form an instability with respect to this matching.
    - Both a_1 and c_1 would want to leave their respective partners and pair up.

Stable matching problem – toy example 2

- Suppose we have a set of two applicants $\{a_1, a_2\}$, and a set of two companies $\{c_1, c_2\}$. The preference list is as follows.
  - a_1 prefers c_1 to c_2
  - a_2 prefers c_2 to c_1
  - c_1 prefers a_2 to a_1
  - c_2 prefers a_1 to a_2
- The two applicant's preferences mesh perfectly with each other (they rank different companies first), and the two companies' preferences likewise mesh perfectly with each other. But the applicant's preferences clash completely with the companies' preferences.
  - There are two different stable matchings. 1) The matching consisting of the pairs (a_1, c_1) and (a_2, c_2) and 2) the matching consisting of the pairs (a_2, c_1) and (a_1, c_2).
    - In 1) both applicants are as happy as possible, so neither would leave their matched company. In 2) both companies are as happy as possible.

Stable matching problem – toy example 3

Q: Is X-C, Y-B, Z-A a good assignment?



Candidates' preferences

	1st	2nd	3rd
Xavier	Alphabet	Baidu	Campbell
Yulia	Baidu	Alphabet	Campbell
Zoran	Alphabet	Baidu	Campbell

Companies' preferences

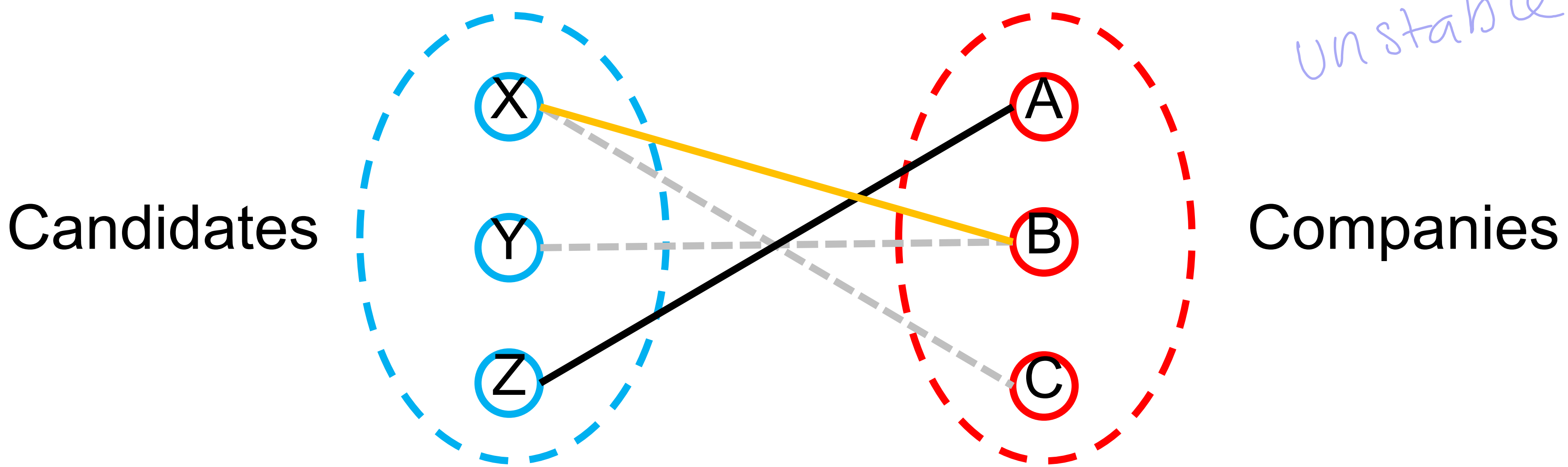
	1st	2nd	3rd
Alphabet	Yulia	Xavier	Zoran
Baidu	Xavier	Yulia	Zoran
Campbell	Xavier	Yulia	Zoran

Stable matching problem – toy example 3

Q: Is X-C, Y-B, Z-A a good assignment?

A: No! Xavier and Baidu will hook up…

unstable



Candidates' preferences

	1st	2nd	3rd
Xavier	Alphabet	Baidu	Campbell
Yulia	Baidu	Alphabet	Campbell
Zoran	Alphabet	Baidu	Campbell

Companies' preferences

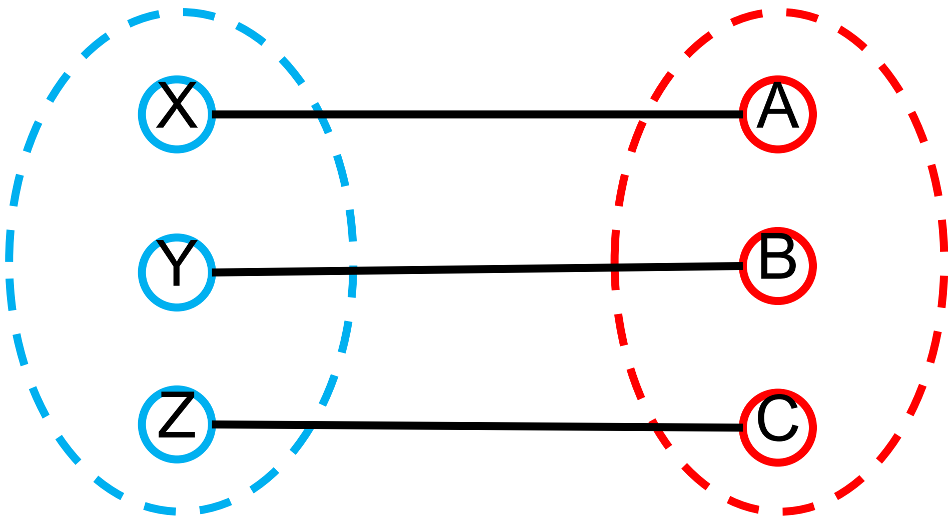
	1st	2nd	3rd
Alphabet	Yulia	Xavier	Zoran
Baidu	Xavier	Yulia	Zoran
Campbell	Xavier	Yulia	Zoran

Stable matching problem – toy example 3

Q: Is X-A, Y-B, Z-C a good assignment?

A: ?

Candidates



Companies

Candidates' preferences

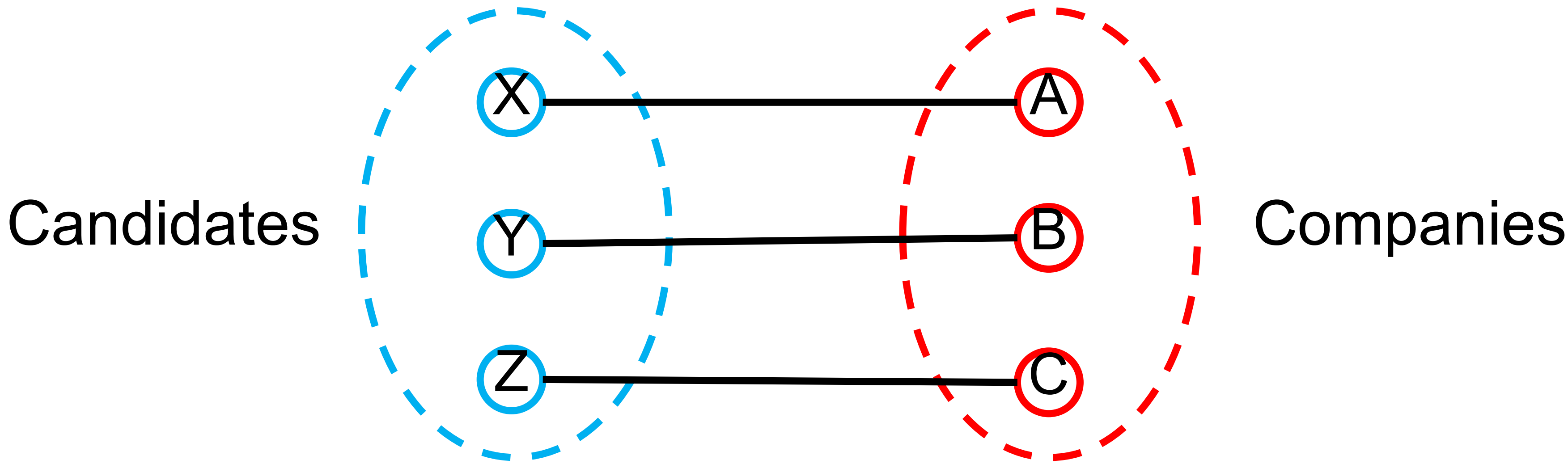
	1st	2nd	3rd
Xavier	Alphabet	Baidu	Campbell
Yulia	Baidu	Alphabet	Campbell
Zoran	Alphabet	Baidu	Campbell

Companies' preferences

	1st	2nd	3rd
Alphabet	Yulia	Xavier	Zoran
Baidu	Xavier	Yulia	Zoran
Campbell	Xavier	Yulia	Zoran

Stable matching problem – toy example 3

Q: Is X-A, Y-B, Z-C a good assignment?
A: Yes!



Candidates' preferences

	1st	2nd	3rd
Xavier	Alphabet	Baidu	Campbell
Yulia	Baidu	Alphabet	Campbell
Zoran	Alphabet	Baidu	Campbell

Companies' preferences

	1st	2nd	3rd
Alphabet	Yulia	Xavier	Zoran
Baidu	Xavier	Yulia	Zoran
Campbell	Xavier	Yulia	Zoran

Stable matching problem

Consider a complete bipartite graph such that |A|=|B|=n.

- Each member of A has a preference ordering of members of B.
- Each member of B has a preference ordering of members of A.

Algorithm for finding a matching:

- Each A member offer to a B, in preference order.
- Each B member accepts the first offer from an A, but then rejects that offer if/when it receives an offer from a A that it prefers more.

In our example: Candidates applies to companies. Companies accept the first offer they receive, but companies will drop their applicant when/if a preferred candidate applies after.

Note the asymmetry between A and B.

Gale-Shapley algorithm

For each $\alpha \in A$, let pref[α] be the ordering of its preferences in B

For each $\beta \in B$, let pref[β] be the ordering of its preferences in A

Let matching be a set of crossing edges between A and B

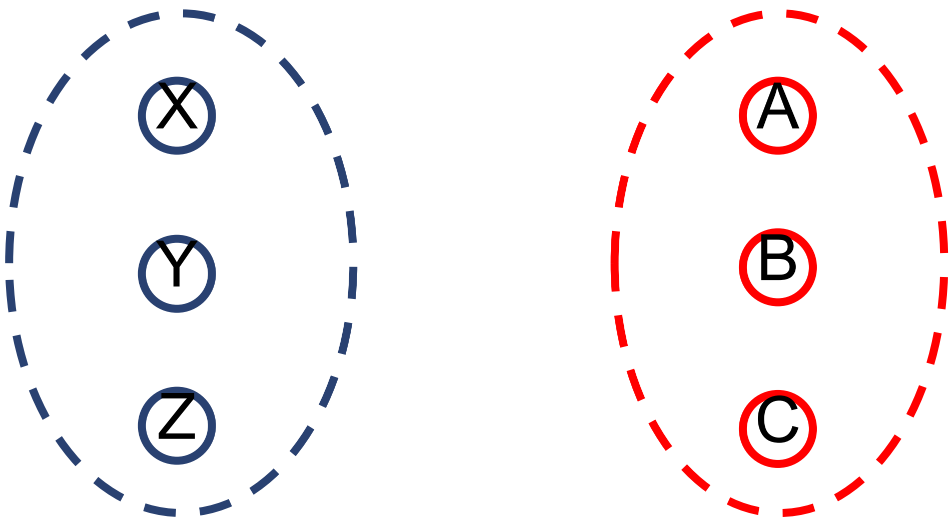
```
matching←∅
while there is α∈A not yet matched do
        β←pref[α].removeFirst()
        if β not yet matched then
                matching←matching∪{(α,β)}
        else
                γ←β's current match
                if β prefers α over γ then
                        matching←matching-{(γ,β)}∪{(α,β)}
return matching
```

first applicant ?
⇒ accept

prefers new student ?

Gale-Shapley algorithm - Example

Candidates



Companies

Candidates' preferences

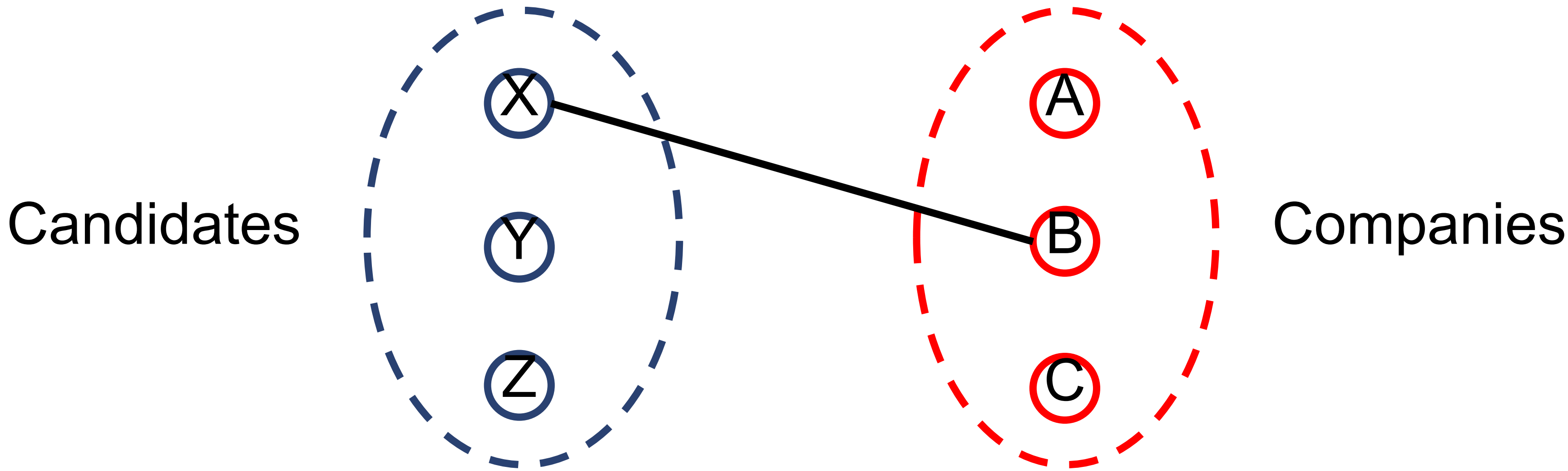
	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Note: In practice, we inverse the roles. Companies makes offers…

Gale-Shapley algorithm - Example



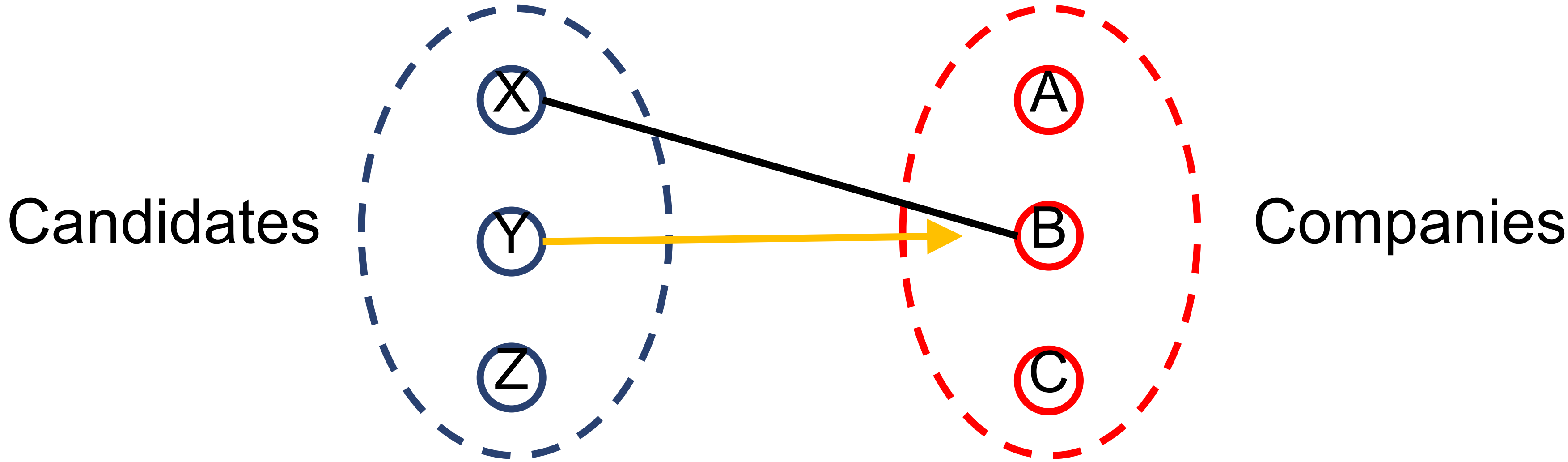
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



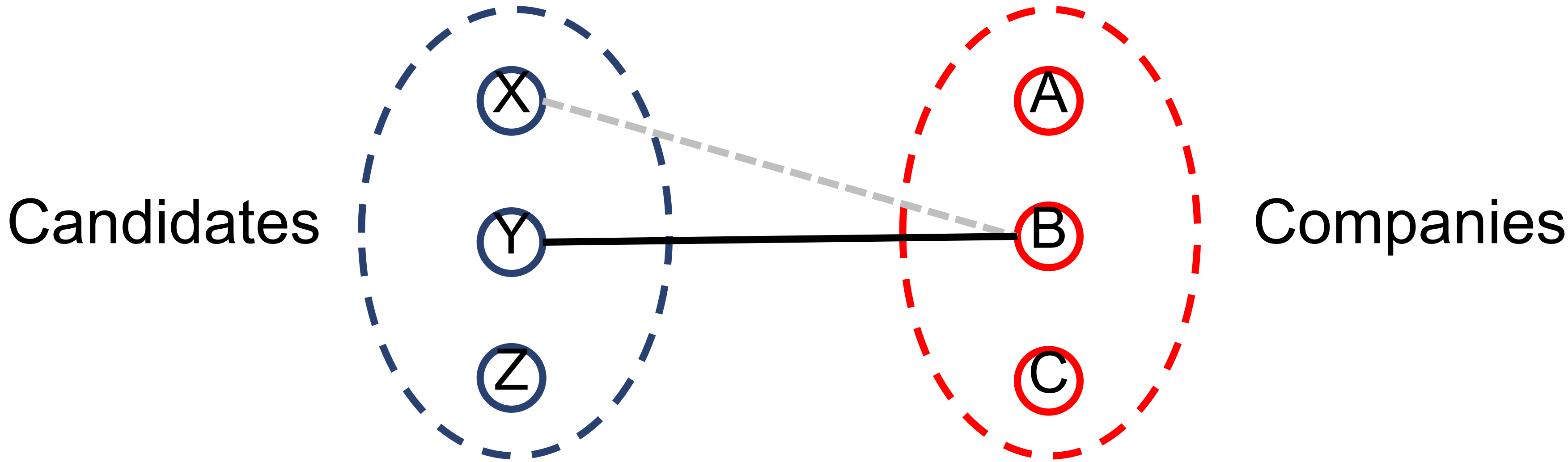
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



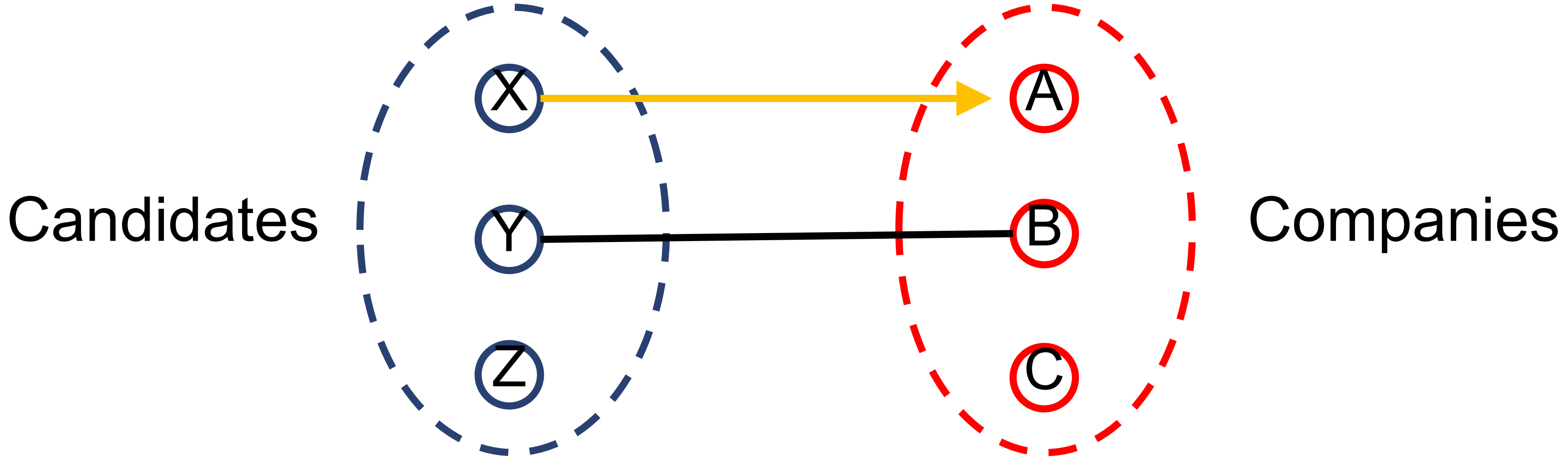
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



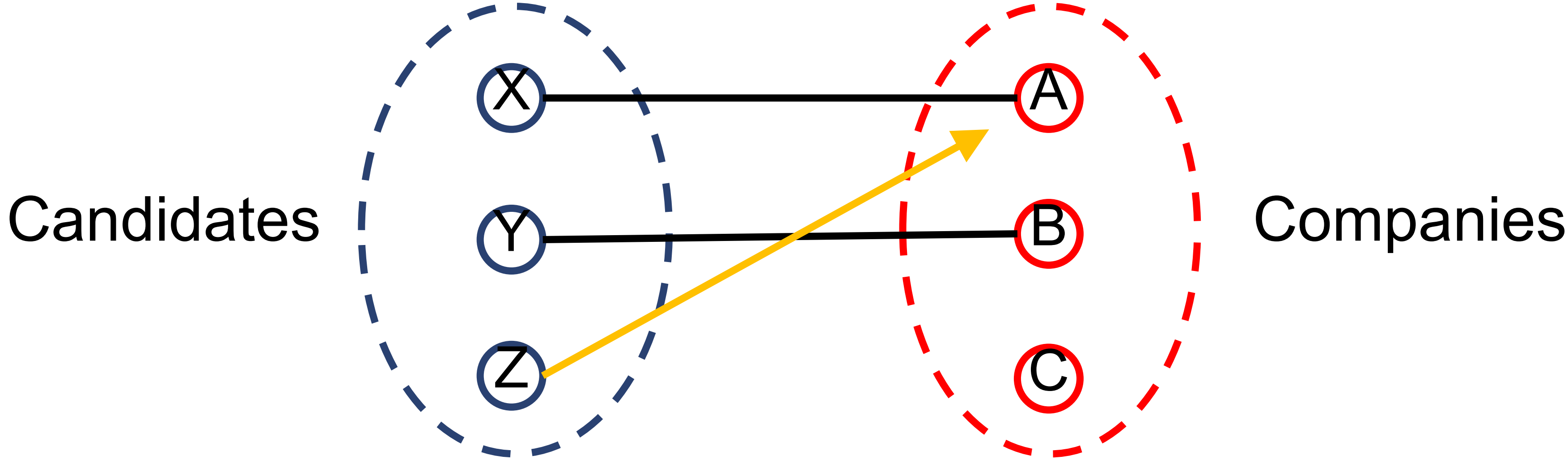
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



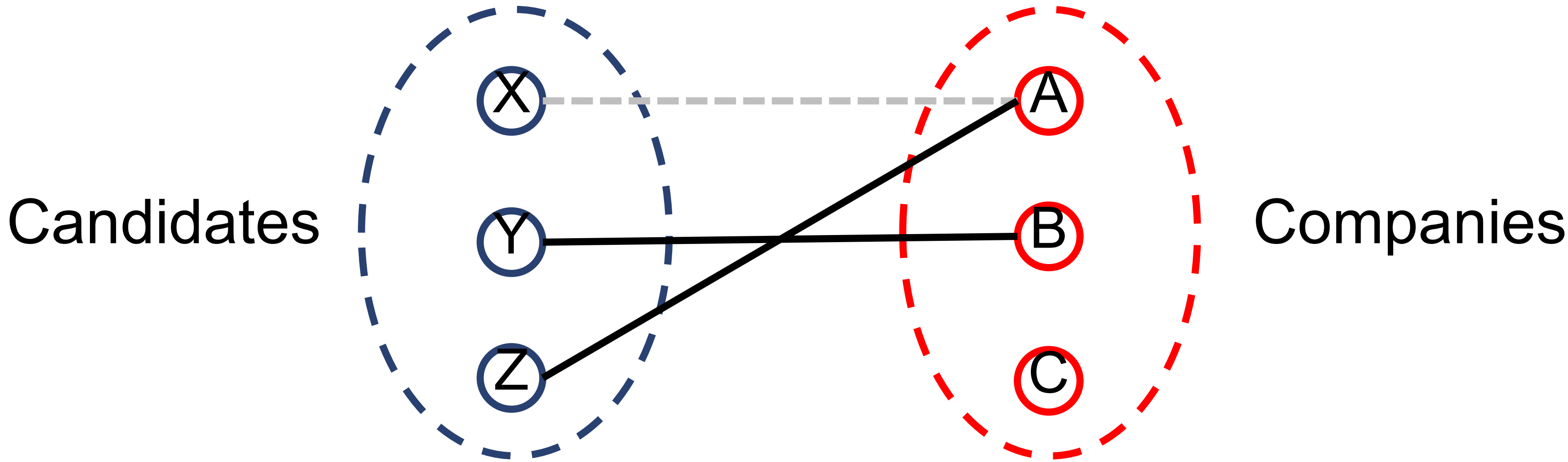
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



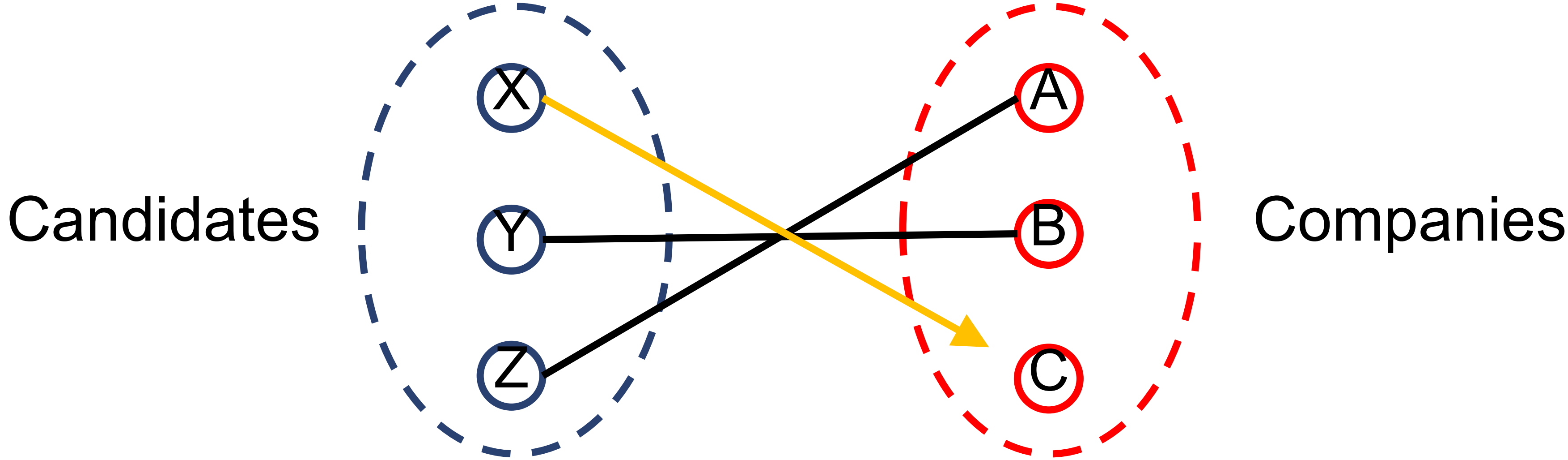
Men's preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



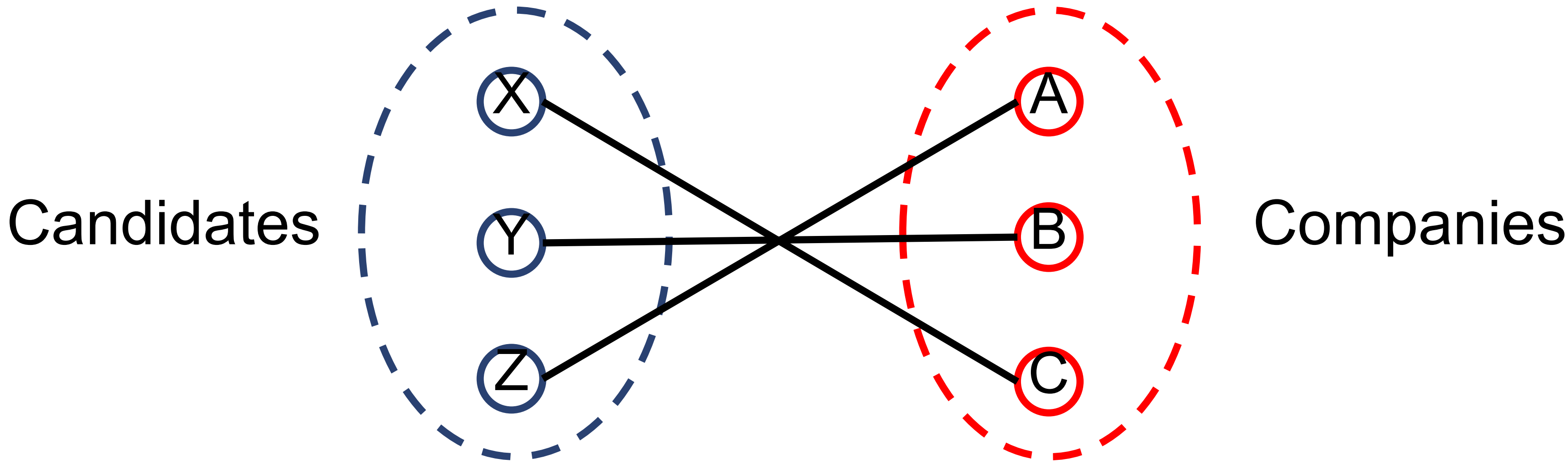
Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

Gale-Shapley algorithm - Example



Candidates' preferences

	1st	2nd	3rd
Xavier	Baidu	Alphabet	Campbell
Yulia	Baidu	Campbell	Alphabet
Zoran	Alphabet	Campbell	Baidu

Companies' preferences

	1st	2nd	3rd
Alphabet	Zoran	Xavier	Yulia
Baidu	Yulia	Zoran	Xavier
Campbell	Xavier	Yulia	Zoran

G-S algorithm – Correctness - Termination

Observations:

1. Candidates apply to companies in decreasing order of preference.
2. Once a company is matched, it never becomes unmatched; it only "trades up."

Claim: Algorithm terminates after at most n^2 iterations of while loop (i.e. $O(n^2)$ running time).

Proof: Each time through the while loop a candidate applies to a new company. There are only n^2 possible matches. ■

G-S algorithm – Correctness - Perfection

Claim: All candidates and companies get matched.

Proof: (by contradiction)

- Suppose, for sake of contradiction, that Zoran is not matched upon termination of algorithm.
- Then some company, say Alphabet, is not matched upon termination.
- By Observation 2 (only trading up, never becoming unmatched), Alphabet never received any application.
- But, Zoran applies everywhere. Contradiction. ■

G-S algorithm – Correctness - Stability

Claim: No unstable pairs.

Proof: (by contradiction)

- Suppose **Z-A** is an unstable pair: they prefer each other to the association made in Gale-Shapley `matching`.
- Case 1: **Z** never applied to **A**.
  - ⇒ **Z** prefers his GS match to **A**.
  - ⇒ **Z-A** is stable.
- Case 2: **Z** applied to **A**.
  - ⇒ **A** rejected **Z** (right away or later)
  - ⇒ **A** prefers its GS match to **Z**.
  - ⇒ **Z-A** is stable.
- In either case **Z-A** is stable. Contradiction. ■

G-S algorithm – Correctness - Optimal

Definition: Candidate **α** is a valid partner of company **β** if there exists some stable matching in which they are matched.

Applicant-optimal assignment: Each candidate receives **best** valid match (according to his preferences).

same algorithm but switched

Claim: All executions of GS yield an (the same) **applicant-optimal** assignment, which is a stable matching!

Note: the notation "Applicant-optimal" refers to α-optimality

Claim: Each element of (β) receive the worst valid partner. GS finds a **company-worst** stable matching!

G-S algorithm – Correctness - Optimal

	1st	2nd	3rd
X	B	A	C
Y	A	B	C
Z	A	B	C

	1st	2nd	3rd
A	X	Y	Z
B	Y	X	Z
C	X	Y	Z

Two stable matchings: and S = { Y-A, X-B, Z-C } and S' = { X-A, Y-B, Z-C }

Then:

- Both X and Y are valid partners for A.
- Both X and Y are valid partners for B.
- Z is the only valid partner for C.
- In S, X Y Z match their best valid partner.

Outline

- Graphs.
  - Introduction.
  - Topological Sort. / Strong Connected Components
  - Network Flow 1.
    - Introduction
    - Ford-Fulkerson
  - Network Flow 2.
    - Min-cuts
  - Shortest Path.
  - Minimum Spanning Trees.
  - Bipartite Graphs.


time for questions