Announcements

Some OH are cancelled this week

- all R/F OH
- Robbie's tomorrow
- Allie's Wednesday

Matchings and Covers (Via Flow)

CSE 421 22Au
Lecture 23

Today

One last day of max-flow.

Two new problems we can solve with max-flow/min-cut.

Today's proofs are short but subtle; I'm intentionally taking us through slowly. Please ask questions.

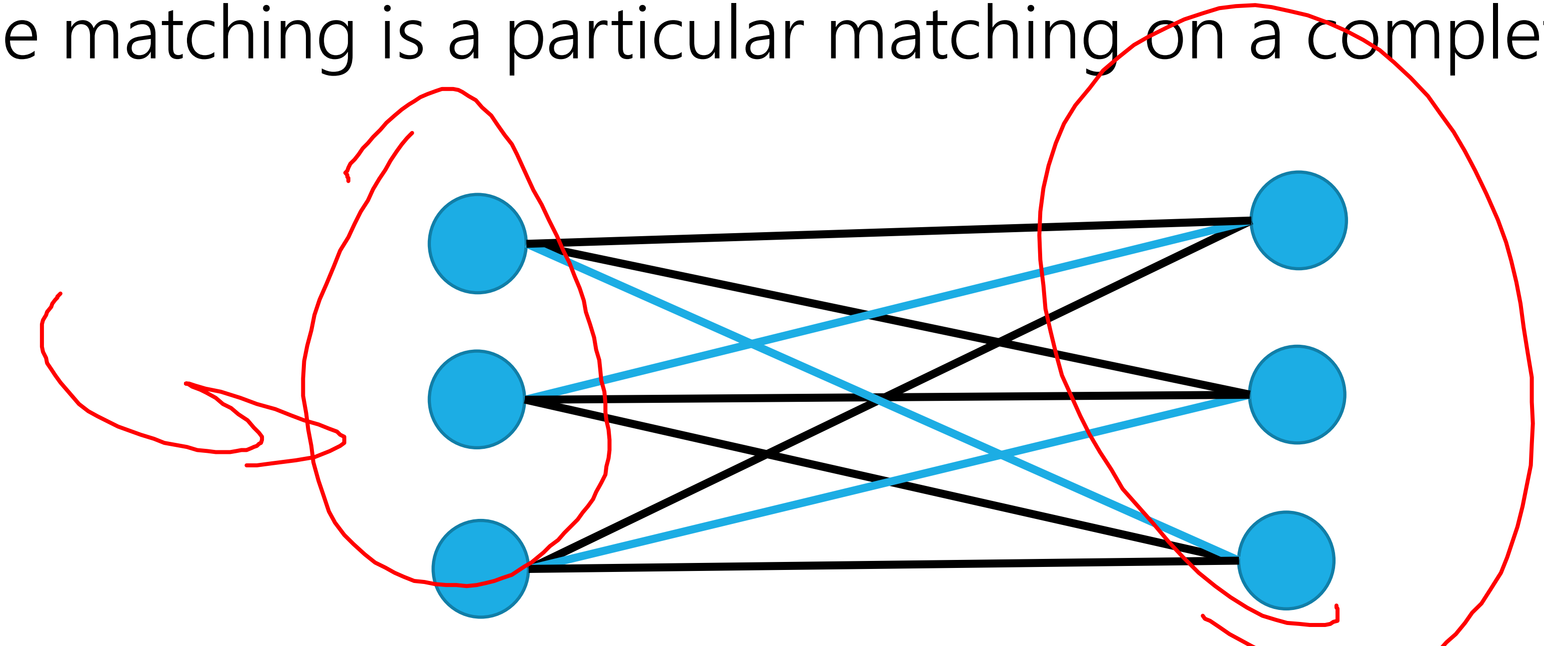
A (possibly) simple problem

Given a bipartite graph, find a maximum matching.

A **matching** is a set of edges that do not share an endpoint.

Think of it as *matching* the endpoints of the edges to each other.

A stable matching is a particular matching on a complete bipartite graph.

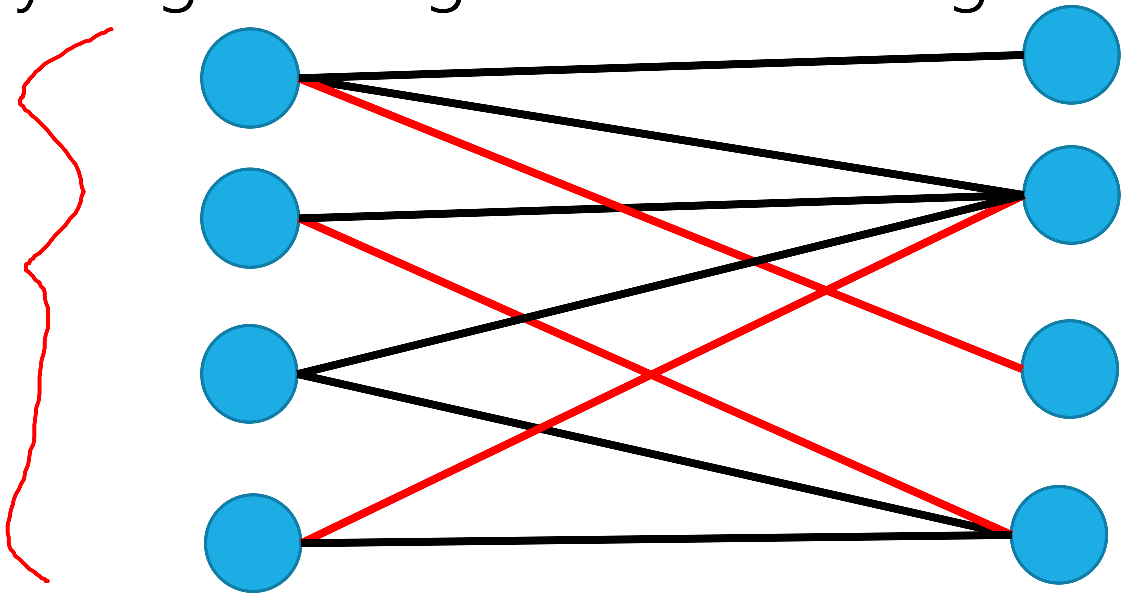


A (possibly) simple problem

Given a bipartite graph, find a maximum matching.

A **matching** is a set of edges that do not share an endpoint.

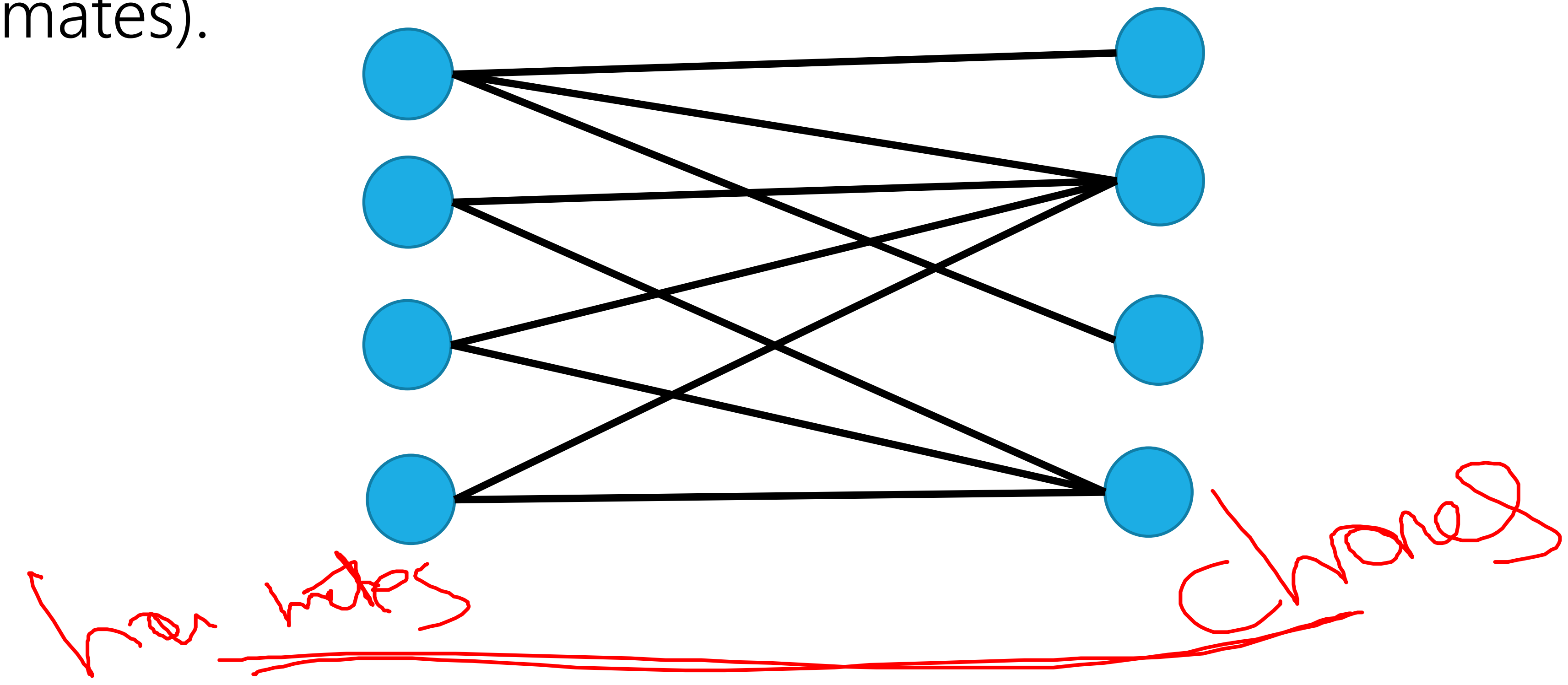
For example, on this graph the red edges are a maximum matching.
There is no way to get 4 edges in a matching.



A (possibly) simple problem

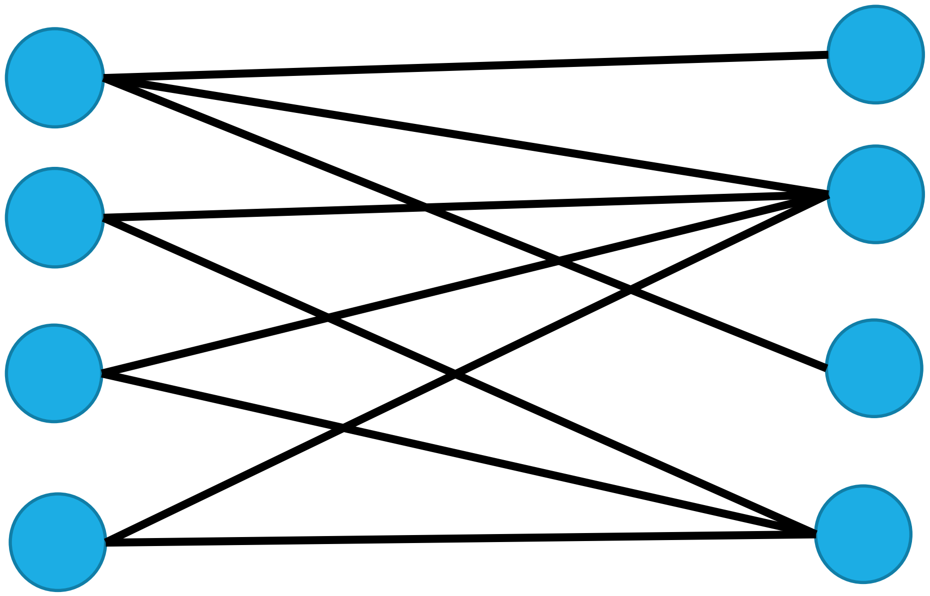
Design an algorithm to find a maximum matching on a bipartite graph.

(*hint:* what if the vertices on one side are chores and the other are housemates).



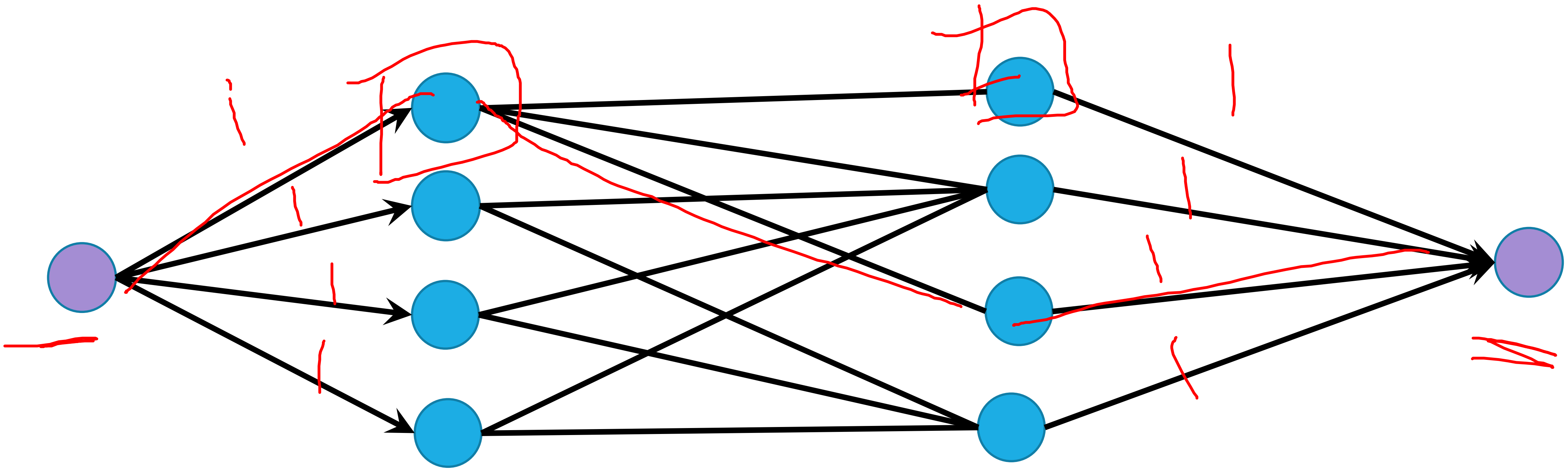
Algorithm for Bipartite Maximum Matching

Use the “tuple selection/assignment” we did last week!



Algorithm for Bipartite Maximum Matching

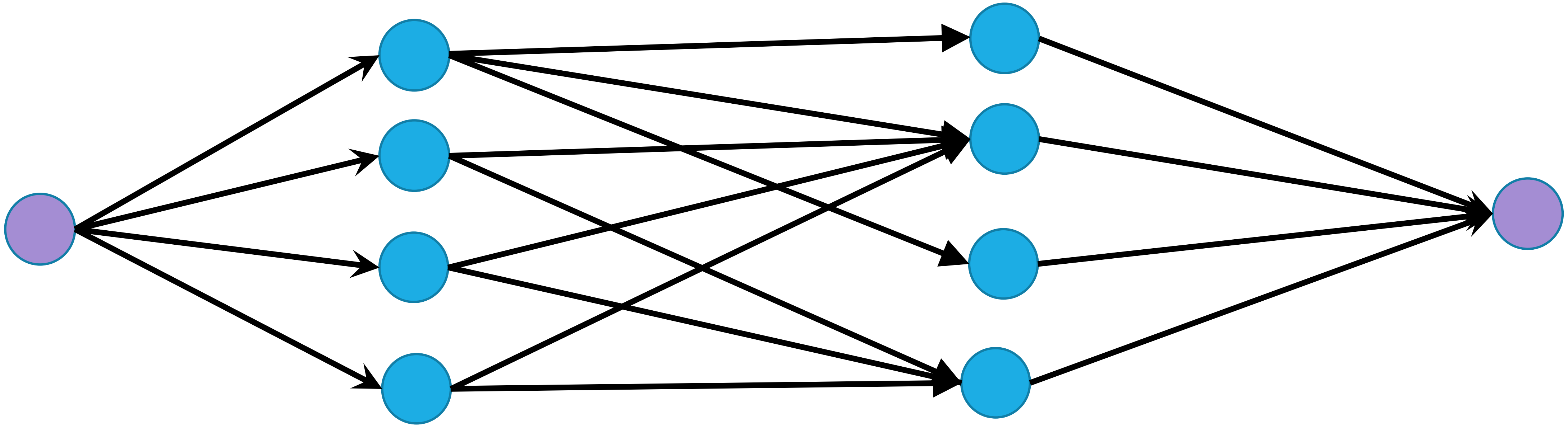
Add a dummy source and sink. Attach each to one side.



Algorithm for Bipartite Maximum Matching

Add a dummy source and sink. Attach each to one side.

Direct (previously undirected) edges toward sink

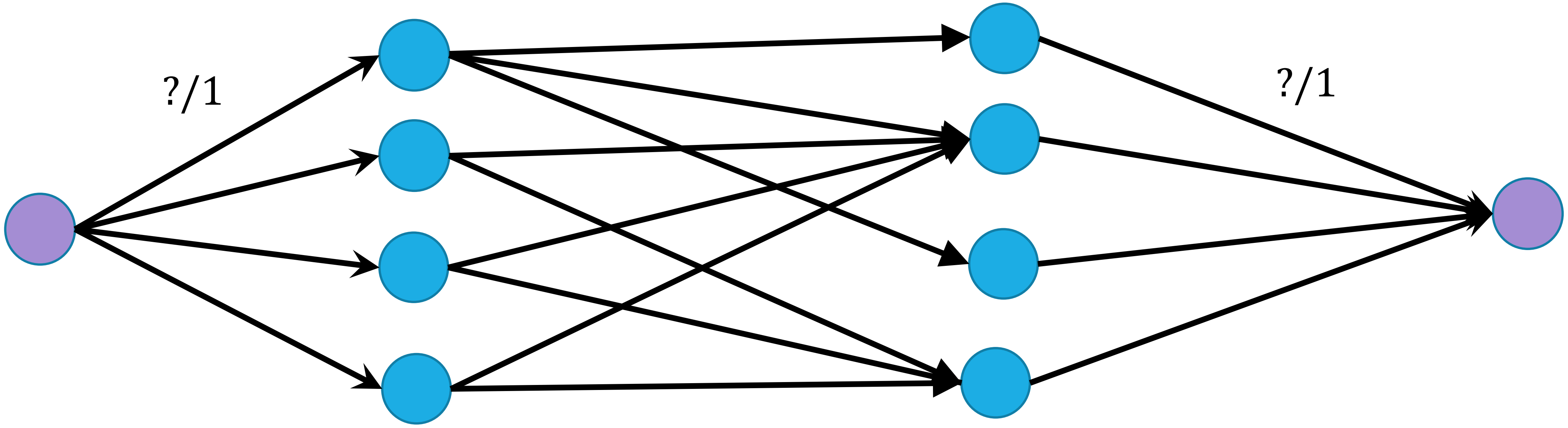


Algorithm for Bipartite Maximum Matching

Add a dummy source and sink. Attach each to one side.

Direct (previously undirected) edges toward sink.

Capacity 1 entering and leaving each (real) vertex ensures matching



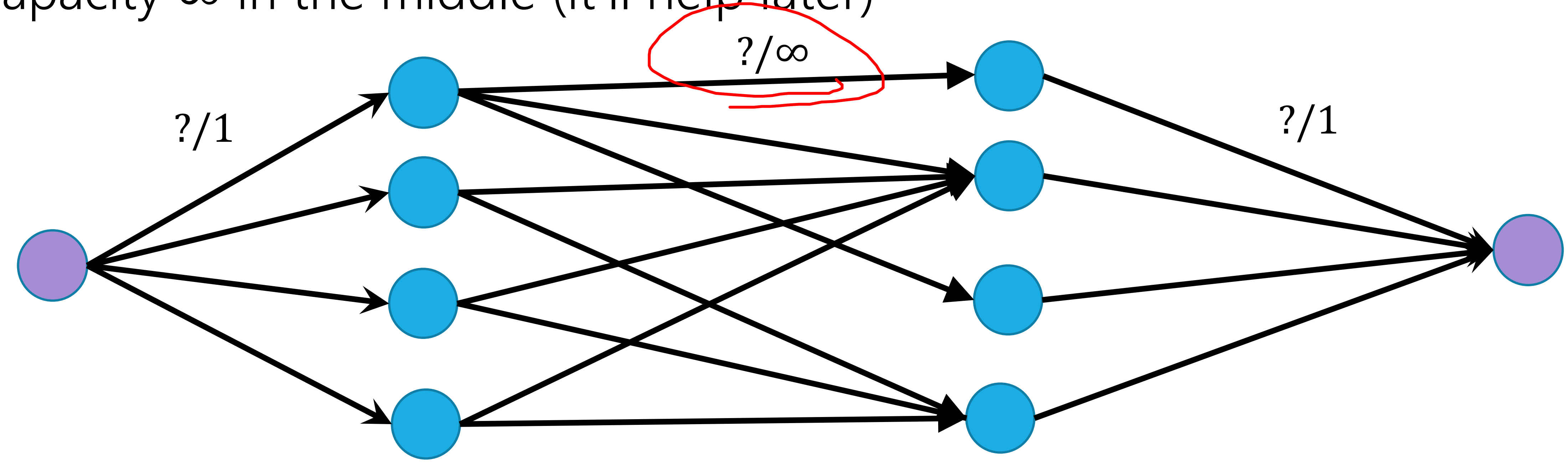
Algorithm for Bipartite Maximum Matching

Add a dummy source and sink. Attach each to one side.

Direct (previously undirected) edges toward sink.

Capacity 1 entering and leaving each (real) vertex ensures matching

Capacity ∞ in the middle (it'll help later)



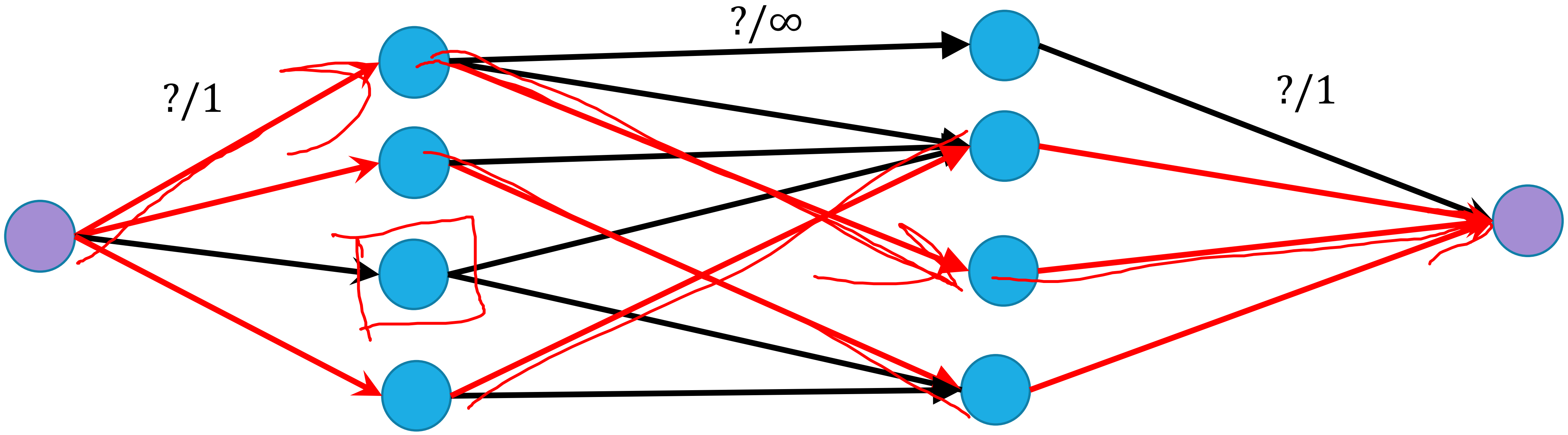
Algorithm for Bipartite Maximum Matching

Add a dummy source and sink. Attach each to one side.

Direct (previously undirected) edges toward sink.

Capacity 1 entering and leaving each (real) vertex ensures matching

Capacity ∞ in the middle (it'll help later)



Algorithm for Bipartite Maximum Matching

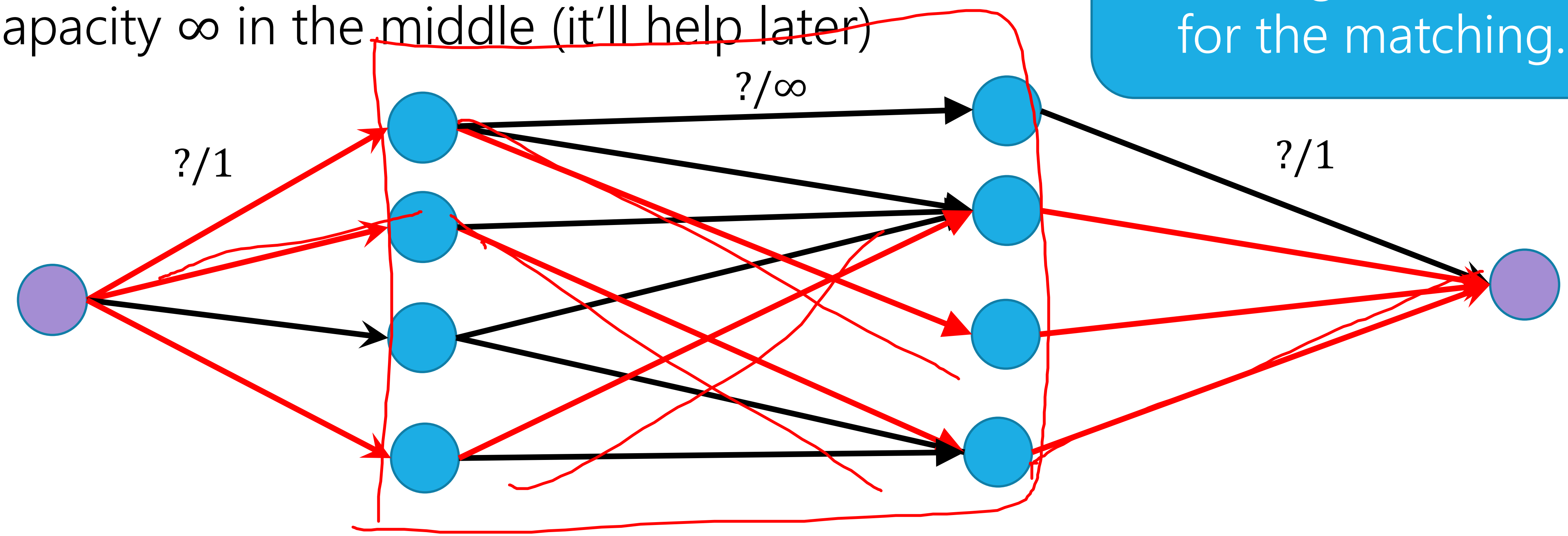
Add a dummy source and sink. Attach each to one side.

Direct (previously undirected) edges toward sink.

Capacity 1 entering and leaving each (real) vertex

Capacity ∞ in the middle (it'll help later)

Red edges have flow.
Take edges with flow
for the matching.



Algorithm for Bipartite Matching

Modify the (undirected) graph G into the network flow graph.

Find a maximum flow, taking all edges of G which have flow.

Is it correct?

The set of edges found should be a matching.

There should be no larger matching.

Algorithm for Bipartite Matching

Modify the (undirected) graph G into the network flow graph.

Find a maximum flow, taking all edges of G which have flow.

Is it correct?

The set of edges found should be a matching.

Capacities ensure no edge has more than one unit of flow through it. By integrality of flow, we have only one edge.

There should be no larger matching.

Suppose, for contradiction, that M is a larger matching, then put a unit of flow on M, entering each vertex with an endpoint in M on the left and leaving on the right. This is a valid flow! But then its value is $|M|$, which would be larger than the max-flow, a contradiction.

Solving a new problem (with matchings)

Let G be an undirected graph.

A minimum vertex cover is the smallest set of vertices such that every edge has at least one of its endpoints in the set.

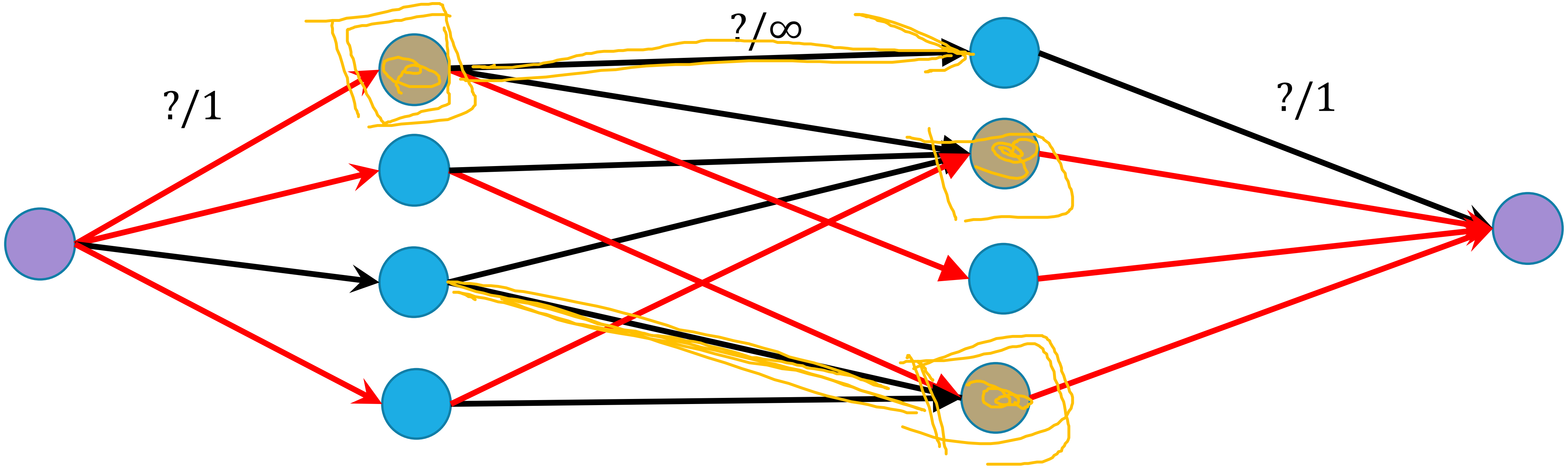
Here's an[other] algorithm to find a minimum vertex cover in a bipartite graph…

Vertex Cover

How to find a vertex cover?

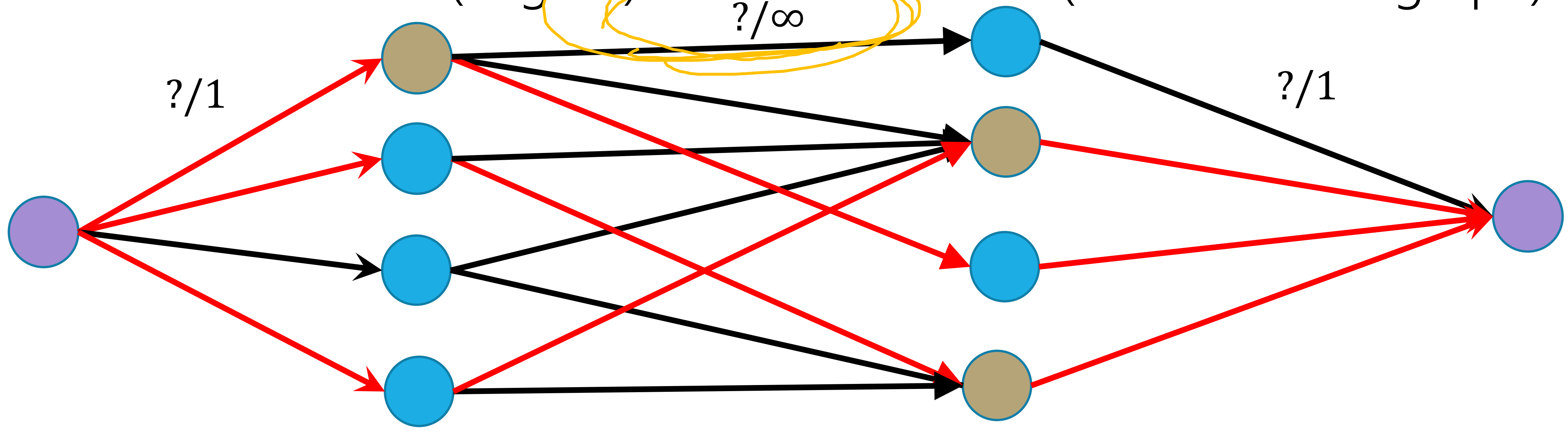
Well a max-flow says “you can’t use this edge; (at least) one of its endpoints is already used by the matching.

Here’s a vertex cover (in gold). Notice something about it?



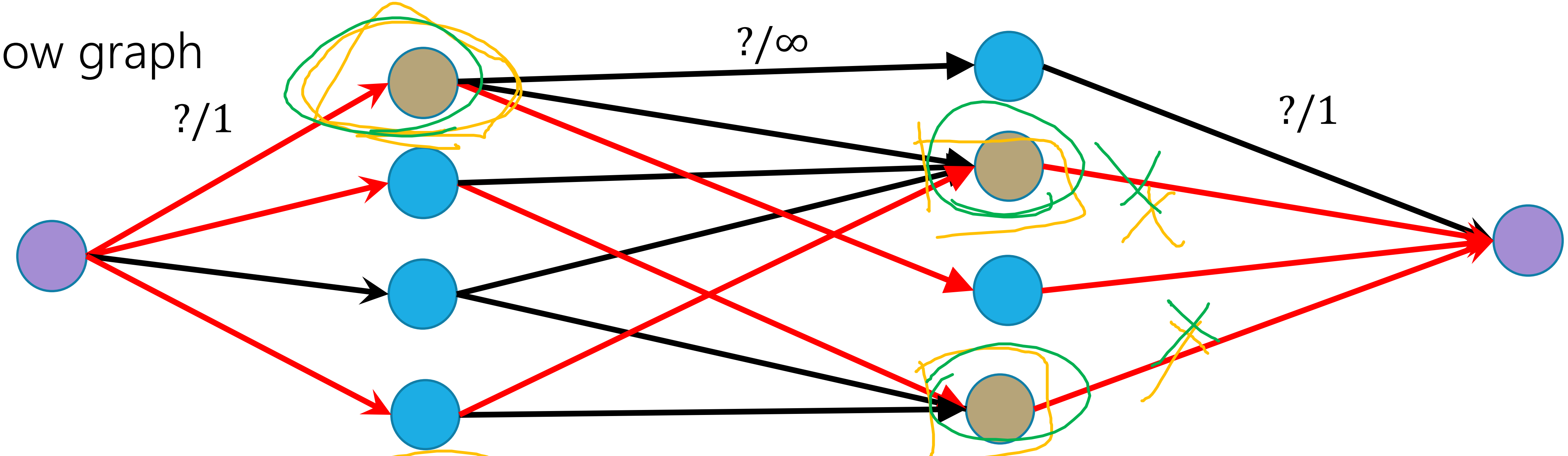
Vertex Cover

Here's a vertex cover (in gold). Find the min-cut (write residual graph)

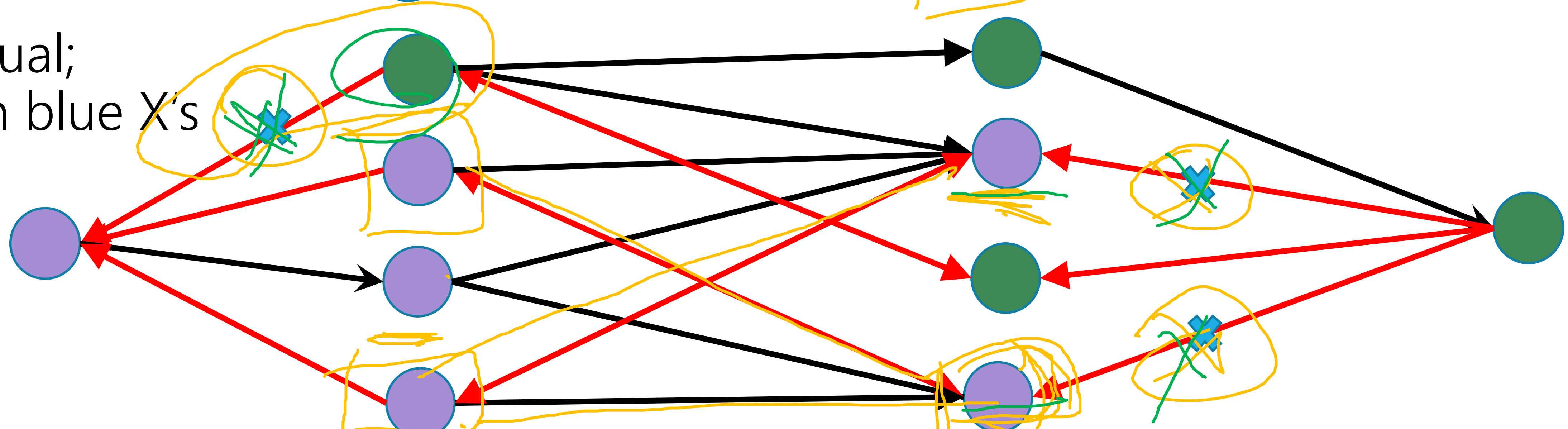


Vertex Cover

Flow graph



Residual;
Cut in blue X's

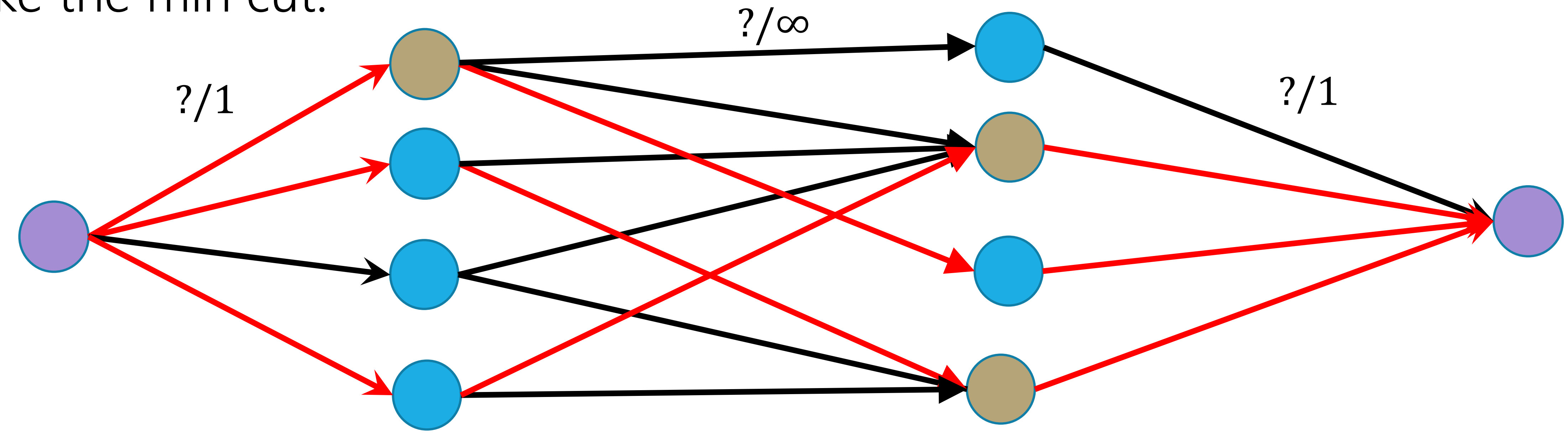


Vertex Cover

How to find a vertex cover?

Well a max-flow says "you can't use this unsaturated edge; (at least) one of its endpoints is already used by the matching.

Here's a vertex cover (in gold). Notice something about it? It looks a lot like the min cut.

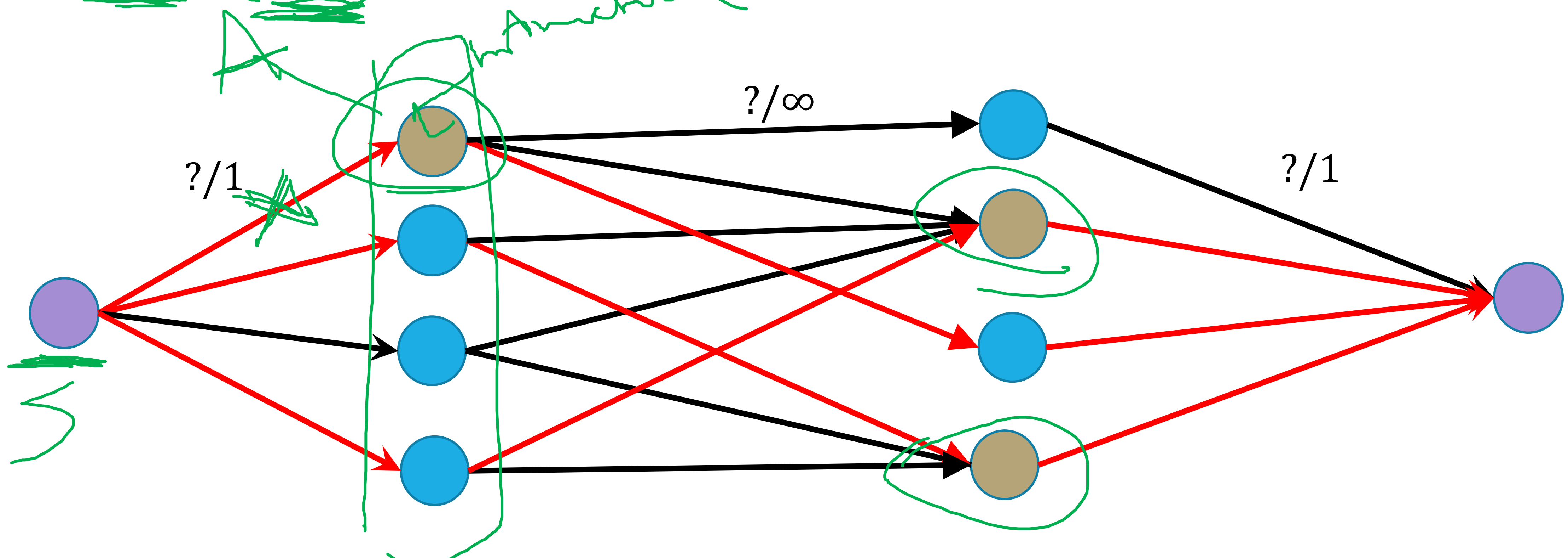


Vertex Cover

Let A be the left bipartition, B the right side. Let (S, T) be the min-cut.

$A_S = A \cap S$; A_T, B_S, B_T defined correspondingly.

Here, A_T and B_S form a vertex cover



Vertex Cover

Is $A_T \cup B_S$ **always** a vertex cover? If so, how big is it?

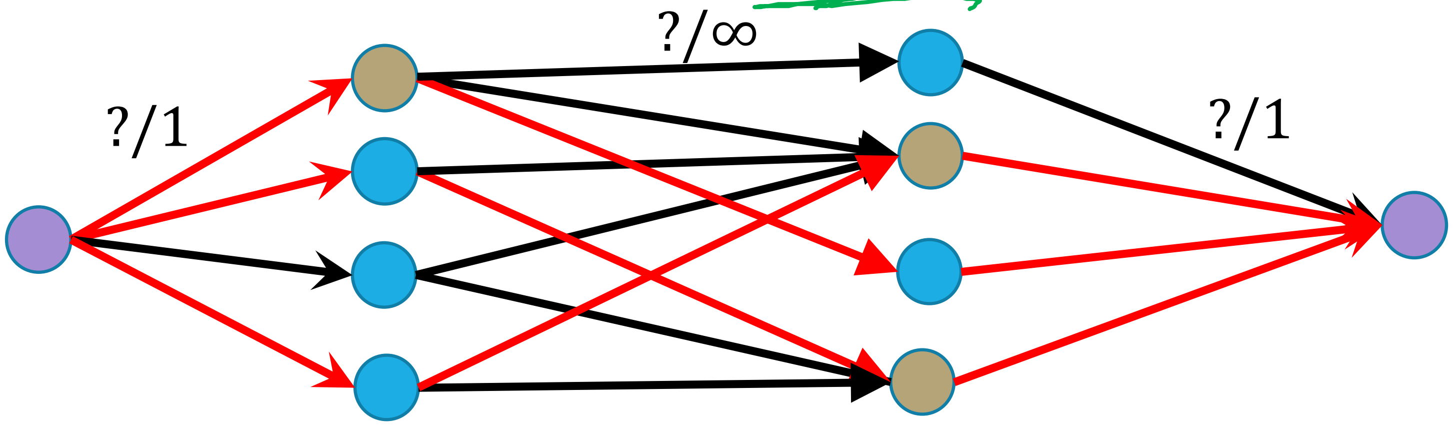
There are 4 potential kinds of edges. Which kind is a problem for the vertex cover? Can they all exist?

A_S to B_S

A_S to B_T

A_T to B_S

A_T to B_T



Vertex Cover

Is $A_T \cup B_S$ **always** a vertex cover? If so, how big is it?

There are 4 potential kinds of edges. Which kind is a problem for the vertex cover? Can they all exist?

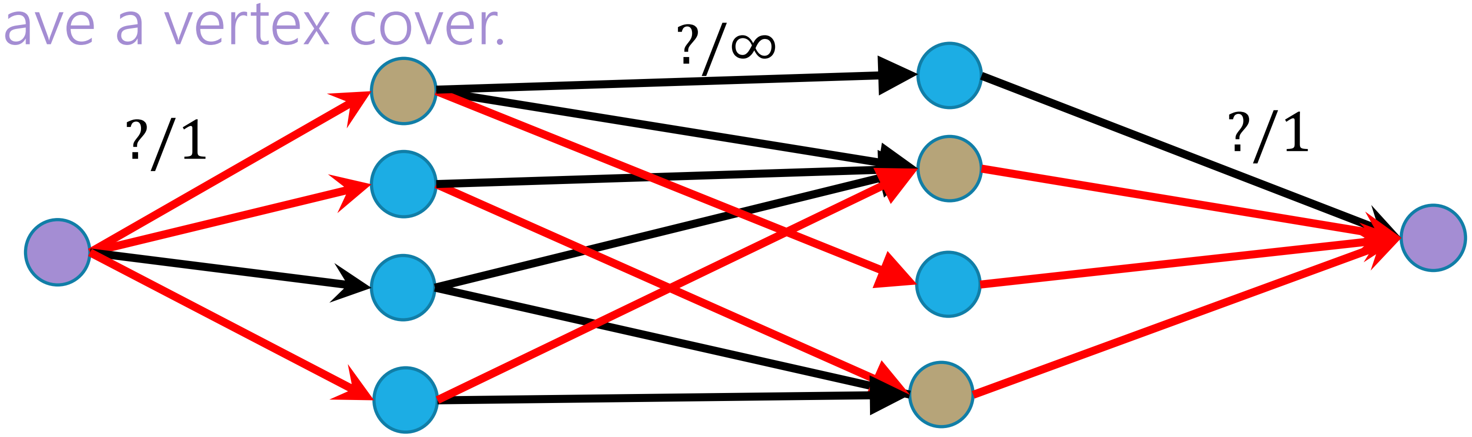
A_S to B_S

A_S to B_T Only kind we have to worry about!!

A_T to B_S

A_T to B_T

But you can't have an edge from S to T! We have a vertex cover.

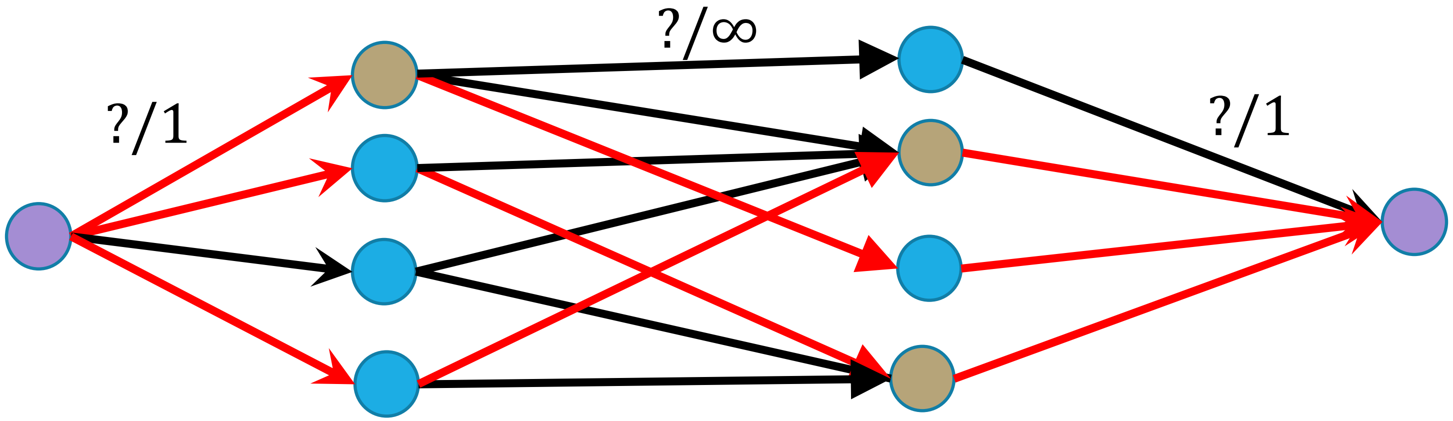


Vertex Cover

How big is $A_T \cup B_S$?

Well, it looks a lot like the min-cut. (s, A_T) and (B_S, t) are cut.

Each of those edges is capacity 1. Each has an endpoint we put in the vertex cover! $|A_T \cup B_S| = \text{cap}(S,T) = |f|$.



But is it a minimum VC?

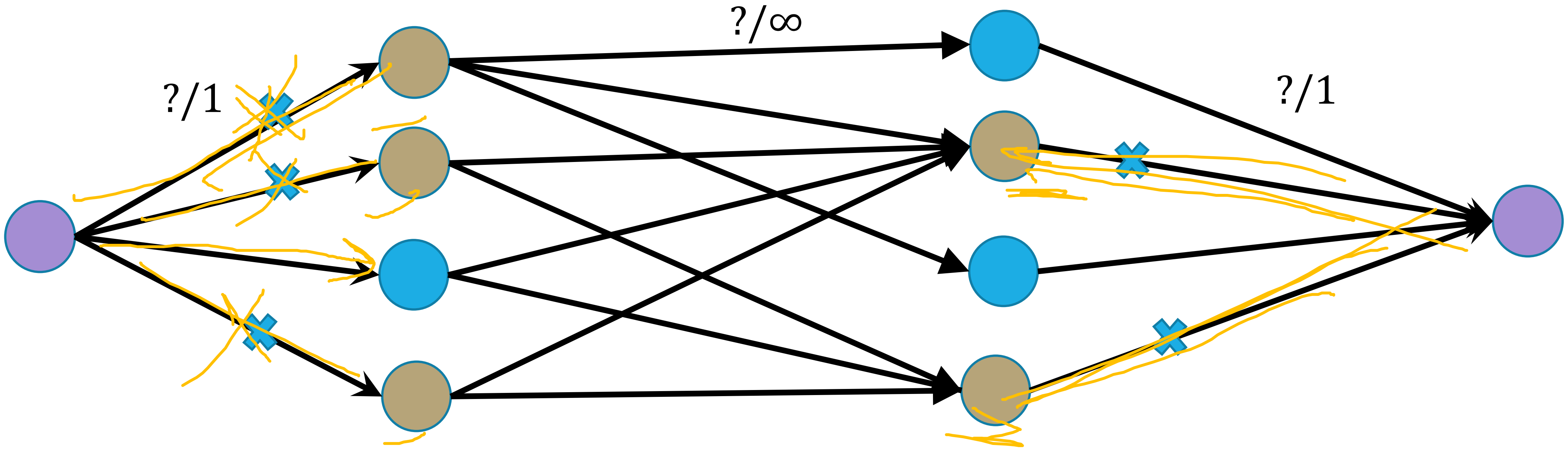
So it's a vertex cover, of size equal to the min-cut…what if there's a smaller vertex cover?

Let X be any vertex cover, define $A_X, \mathrm{B_X}$ as before.

No smaller VC

Let X be any vertex cover, define A_X, B_X as before.

Claim: $(\{s\} \cup B_X \cup (A \setminus A_X), \{t\} \cup (B \setminus B_X) \cup A_X)$ is a cut with cut edges are $(s, A_X), (B_X, t)$. [this is the corresponding cut to before]



But is it a minimum VC?

So it's a vertex cover, of size equal to the min-cut…what if there's a smaller vertex cover?

Let X be any vertex cover, define $A_X, \mathrm{B_X}$ as before.

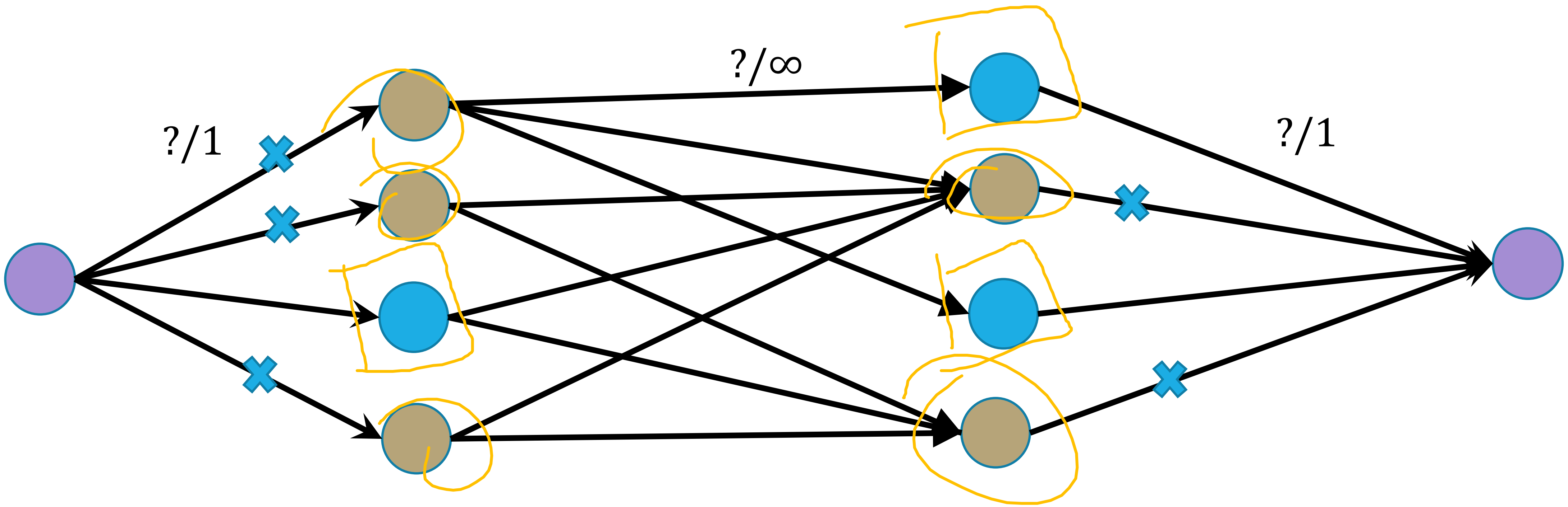
Claim: $(\{s\} \cup B_X \cup (A \setminus A_X)$, $\{t\} \cup (B \setminus B_X) \cup A_X)$ is a cut with cut edges are $(s, A_X), (B_X, t)$.

No $(A \setminus A_X)$, $(B \setminus B_X)$ edges because we're a cover!

No smaller VC

Every vertex cover lets you discover a cut of the same size.

Our algorithm found a VC of size equal to the minimum cut! A smaller VC would give a smaller cut! But there isn't one. So we have found the min VC.



Wrapping it up

You can find the following of a bipartite graph (using only flow)

1. Maximum matching
2. Minimum Vertex cover

And their value is equal to the size of the max-flow (and the min-cut) in the modified graph.

"Kőnig's Theorem" (aka Kőnig-Egevary Theorem)

In a bipartite graph, the size of the maximum matching is equal to the size of the minimum vertex cover.

Wrapping It Up

We've found maximum matchings and minimum vertex covers in bipartite graphs using flow.

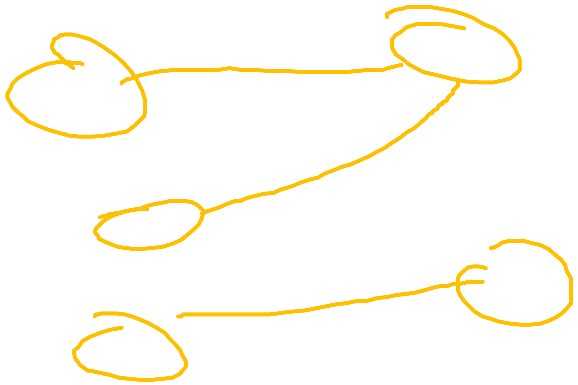
If your graph isn't bipartite:

Efficiently finding a maximum matching is still possible.

but more complicated. Look up "Edmond's Blossom Algorithm"

Efficiently finding a minimum vertex cover, is a taller task.

It's an NP-complete problem. We'll more clearly define what that means and



But Wait, There's More Applications!

Finding Edge disjoint paths in a directed graph.

Maximum number of paths from u to v that don't repeat an edge.

I want to send a packet from u to v but edges are unreliable. Want completely independent copies.

Finding internally-vertex-disjoint paths in a directed graph

Send packet, but I don't trust the vertices.

Image Segmentation

A surprisingly useful tool for matching and separating things.

Also for proving graph theory results (Konig-Egevary, Hall's Theorem)