

Applications of Flows + Cuts

Applications of Flows + Cuts

Bipartite
Matching

Vertex
Disjoint
Paths

Path
Covers

Project
Selection

Assignment
Problems

Edge
Orientation

Densest
Subgraph

Playoff
Elimination

scheduling

Bipartite
Vertex
Cover

(high-level comment)

"only two algorithms"*

① identify subproblems

② change the input

Applications of Flows + Cuts

Gameplan

1. Survey a bunch of problems
2. Start solving them one by one

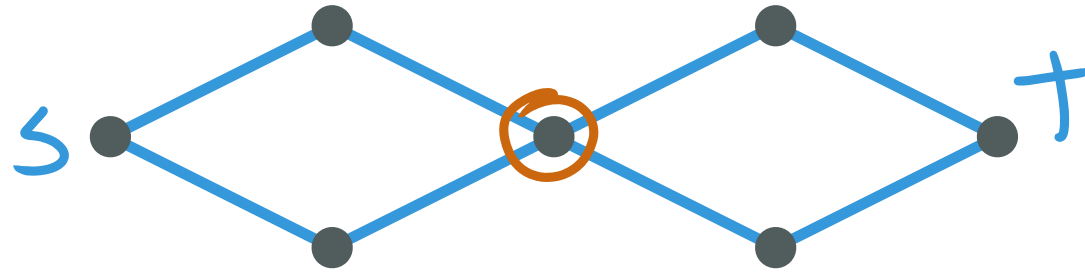


"only two algorithms"

① identify subproblems

② change the input

Vertex Disjoint Paths



2 edge disjoint paths

1 vertex disjoint path

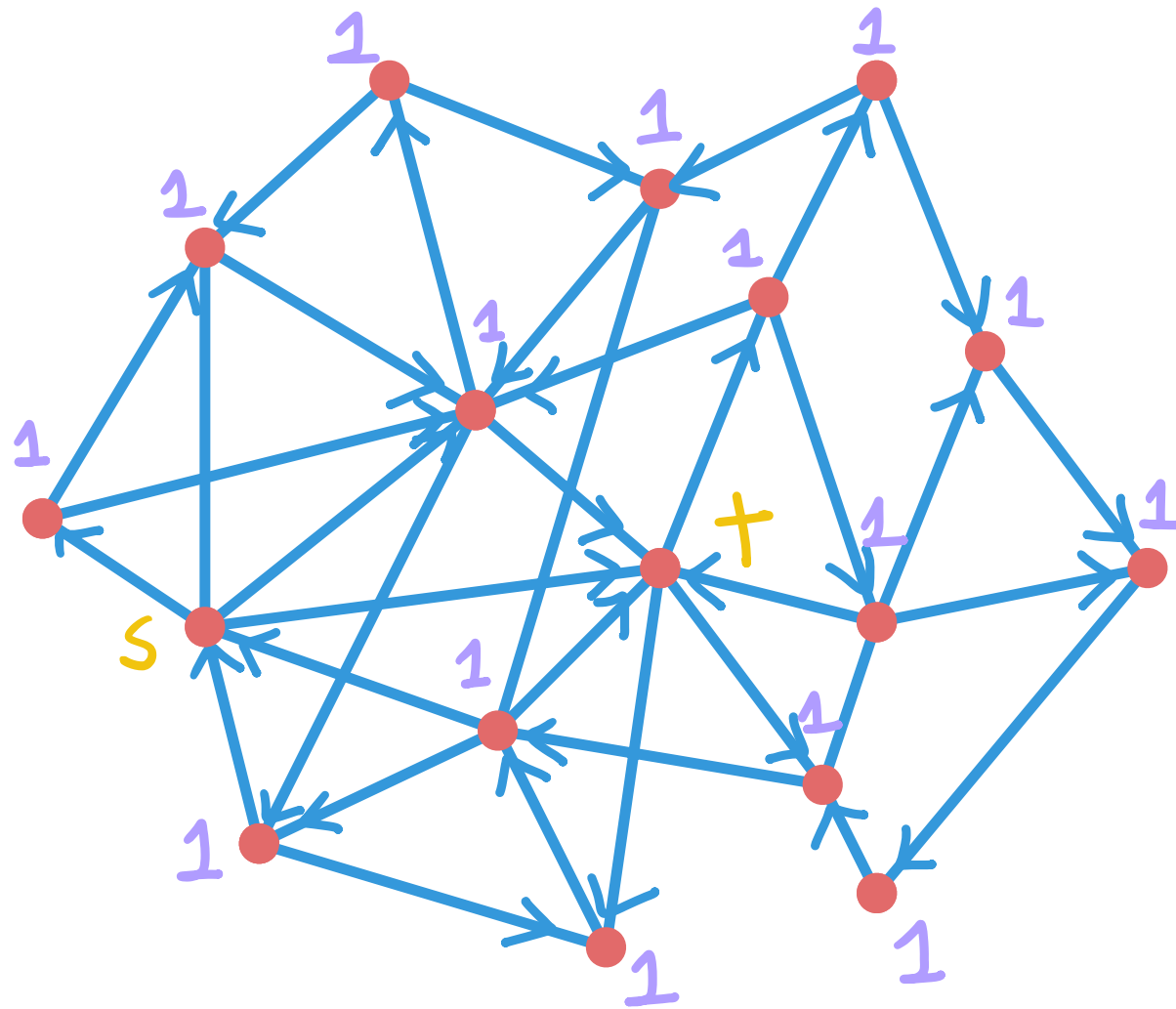
vertex cut = 1

"vertex capacities" $c: V \rightarrow \mathbb{R}_{>0}$

$c(v)$ = limit of flow through v

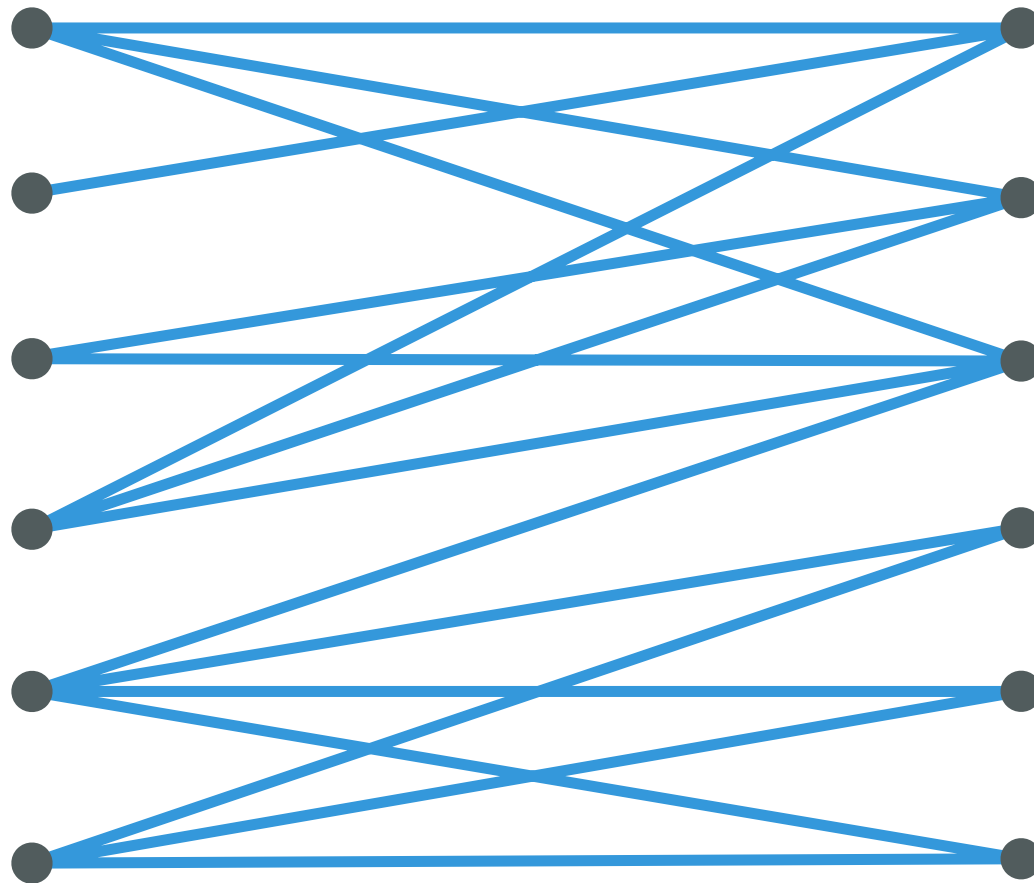
e.g. $c(v) = 1$
 $\forall v \notin \{s, t\}$

$\Rightarrow$ vertex
disjoint paths



Bipartite Matching

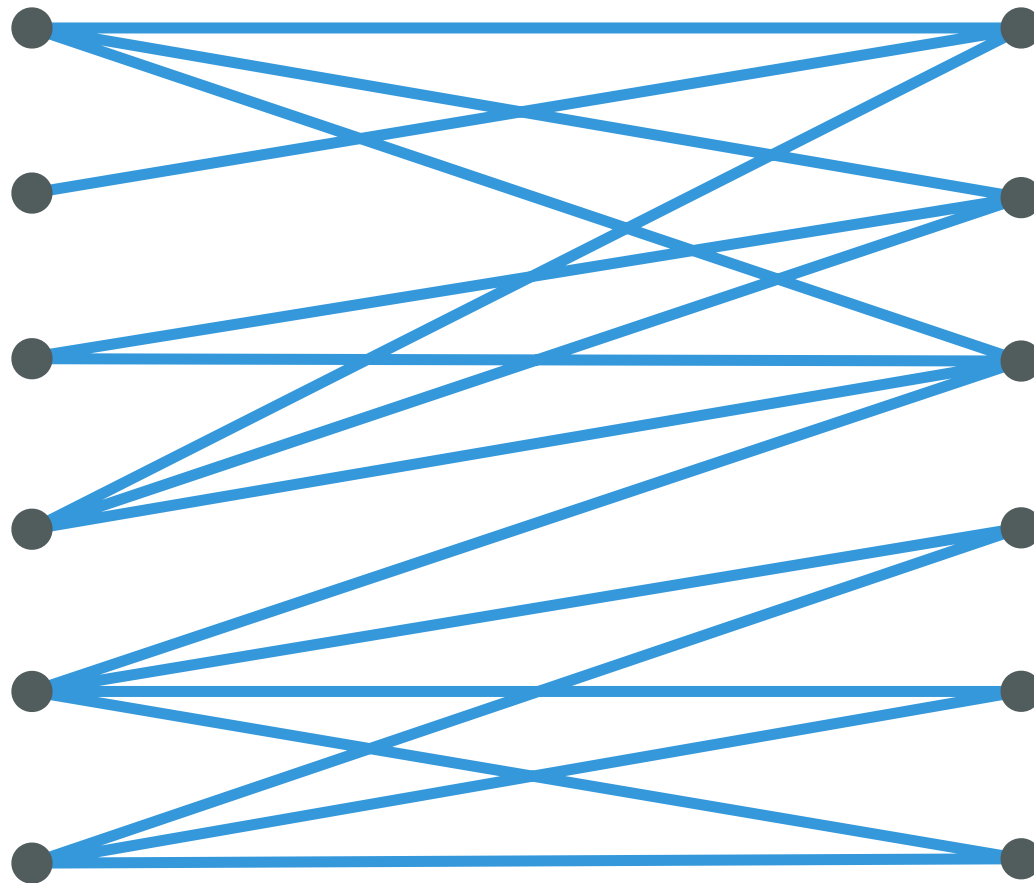
"Bipartite graph"



Goal: max set of edges w/ distinct endpoints
"matching"

Bipartite
Vertex
Cover

"Bipartite graph"

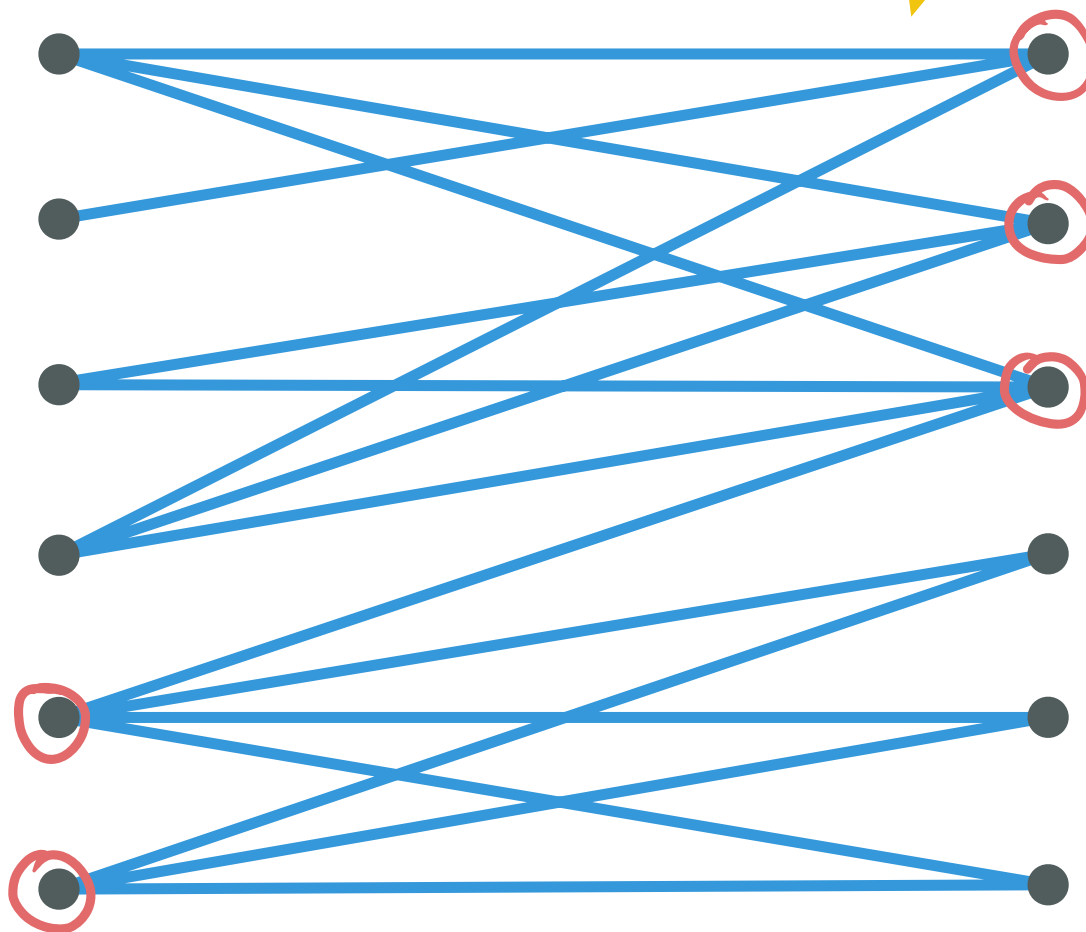


Goal: min set of vertices incident to all edges

"vertex cover"

Bipartite
Vertex
Cover

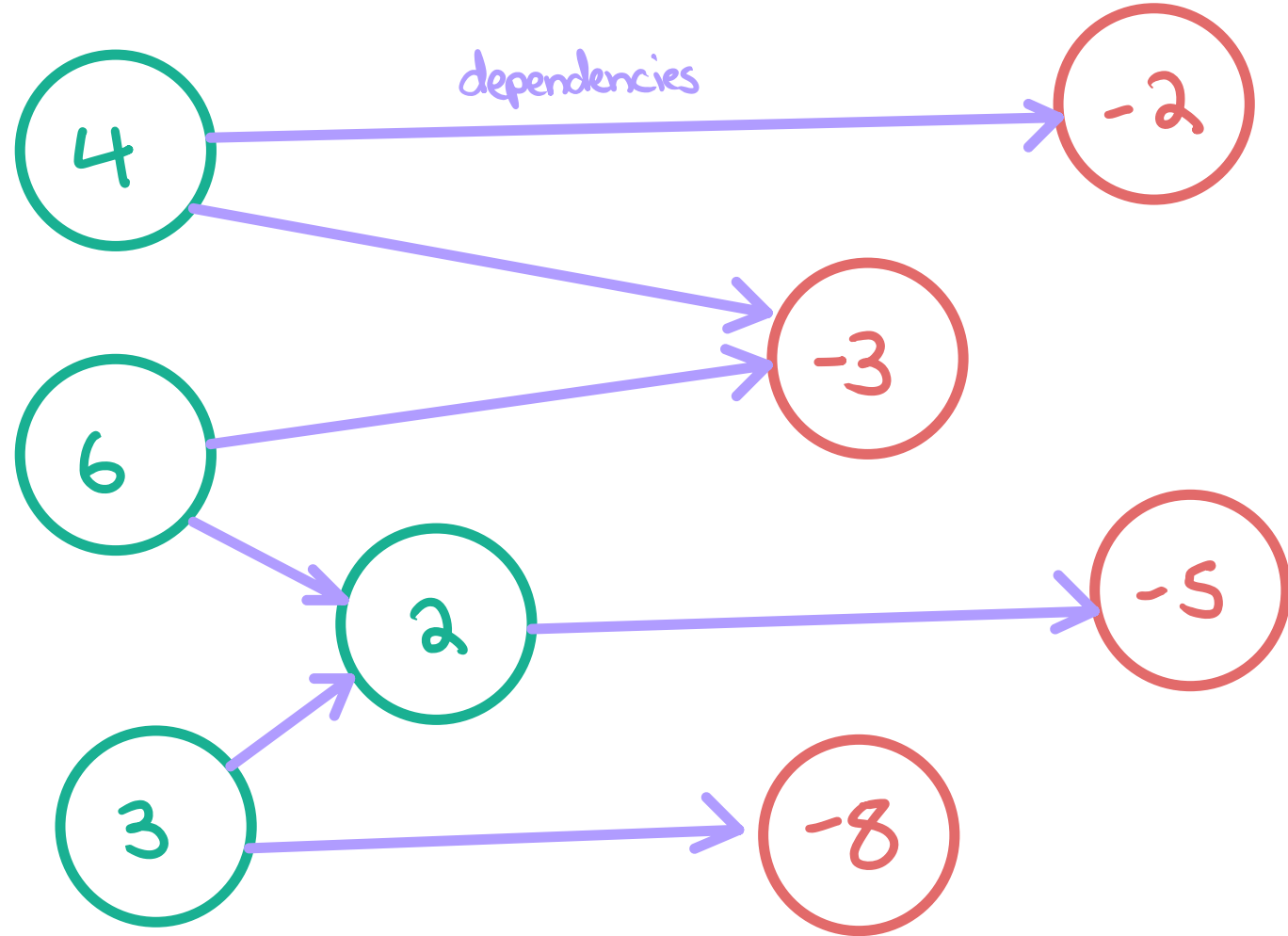
"Bipartite graph"



Goal: min set of vertices incident to all edges

"vertex cover"

Project Selection



Projects have net profits (can be < 0 for costs) and dependencies

Goal: select projects (including dependencies) w/ max total profit

Playoff
Elimination



Playoff Elimination



can Boston catch the Yankees?

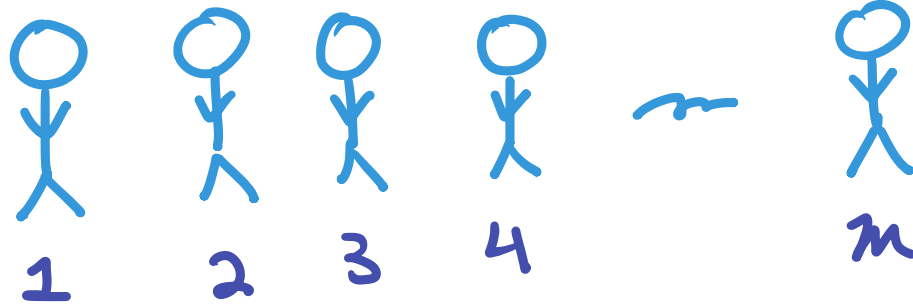
Team	Wins	Games left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	90	2

Remaining schedule

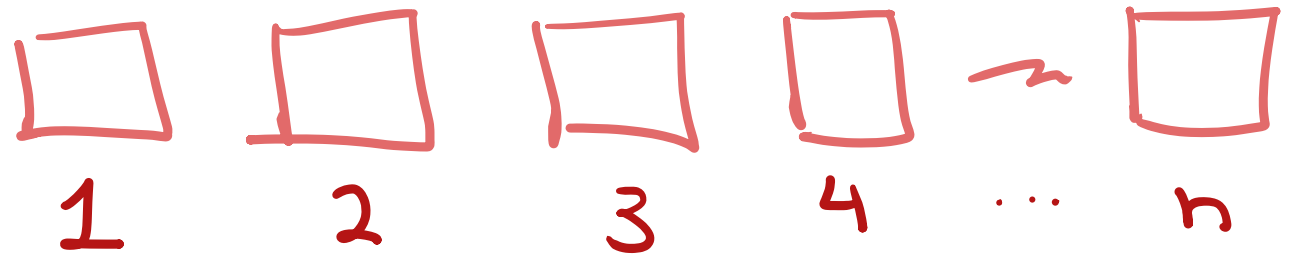
(wins)	(92)	(91)	(91)	(90)
	New York	Baltimore	Toronto	Boston
New York	X	1	1	0
Baltimore	1	X	1	1
Toronto	1	1	X	1
Boston	0	1	1	X

Assignment Problems

m individuals



n jobs



goal: fill as many jobs as possible w/ individuals

the catch: not everyone is qualified for all jobs

Assignment Problems

"Qualification matrix"

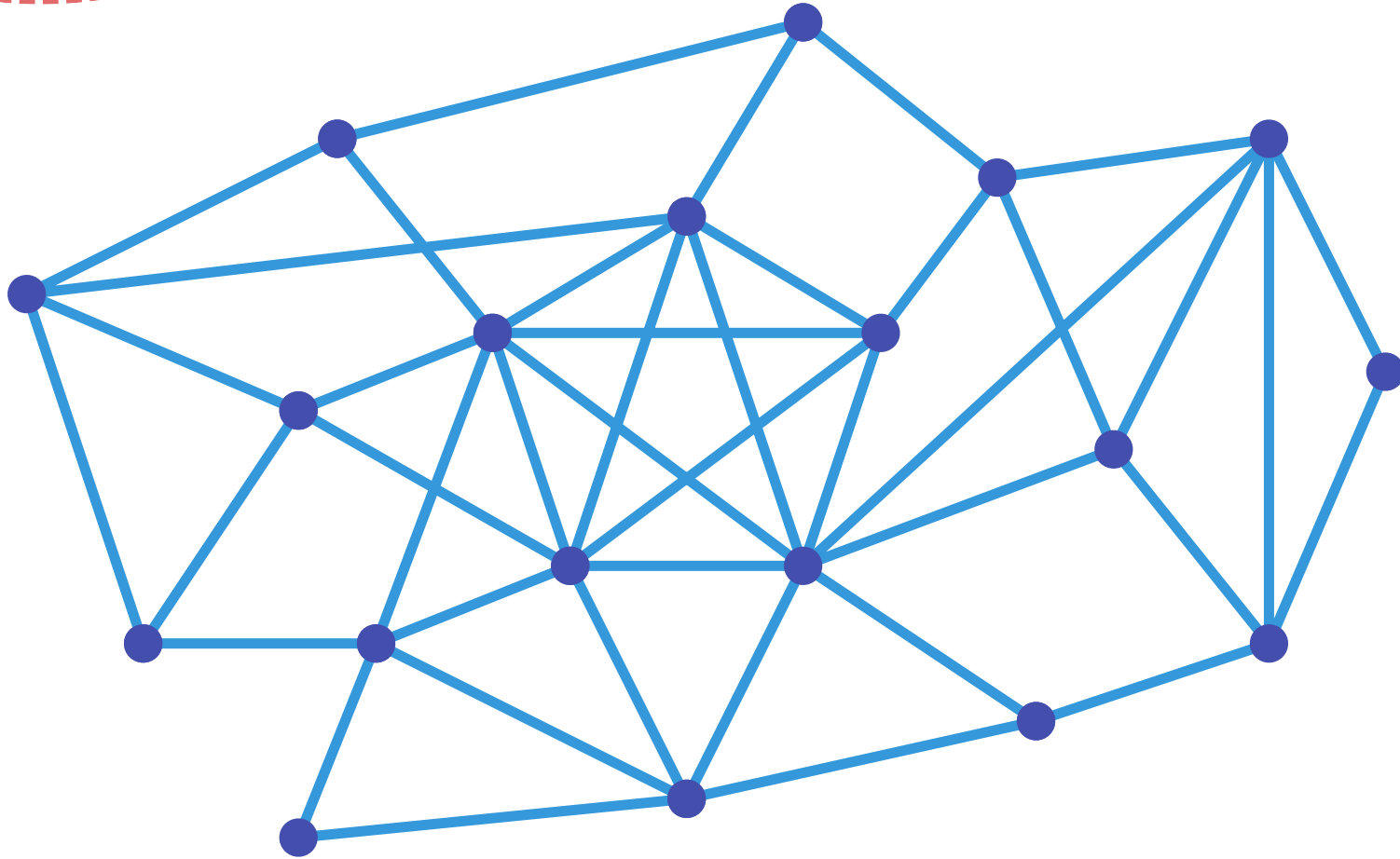
	☐	☐	☐	☐
	1	1	1	0
	0	0	1	0
	0	0	0	1
	0	0	0	1

"1" = qualified

"0" = not qualified

Edge Orientation

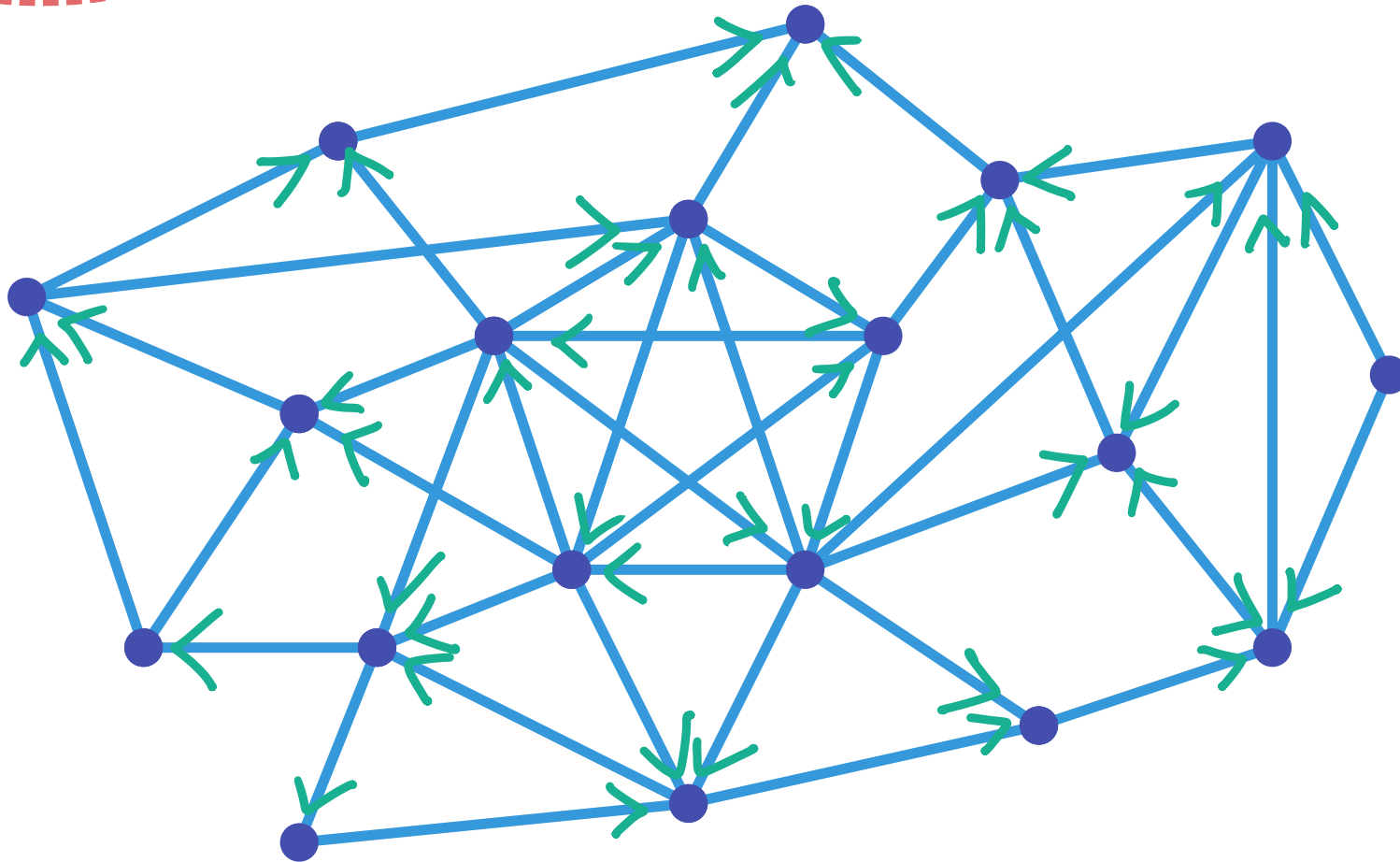
input: undirected graph



Goal: direct the edges to minimize the max in-degree

Edge
Orientation

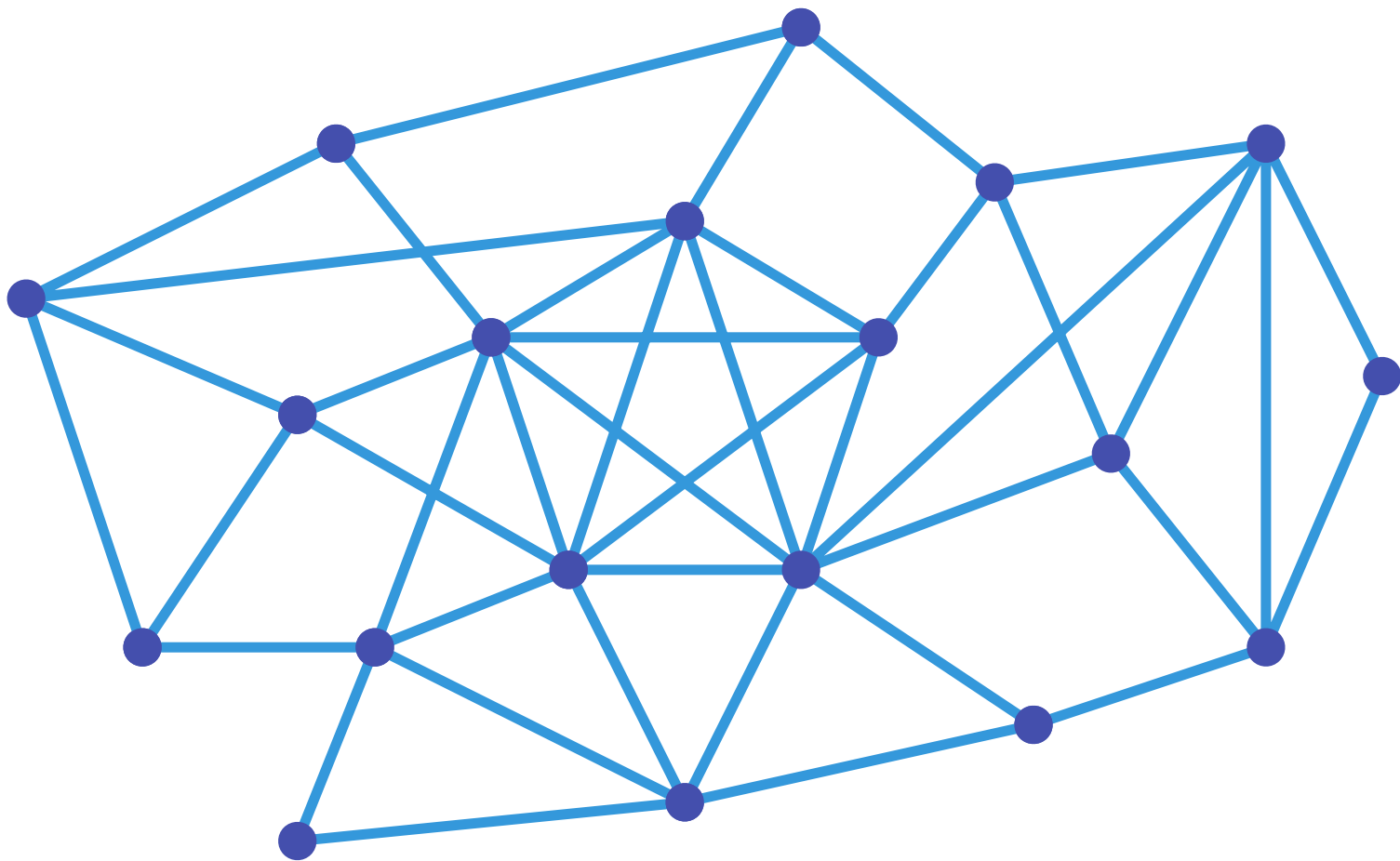
input: undirected graph



Goal: direct the edges to minimize the max in-degree

Densest
Subgraph

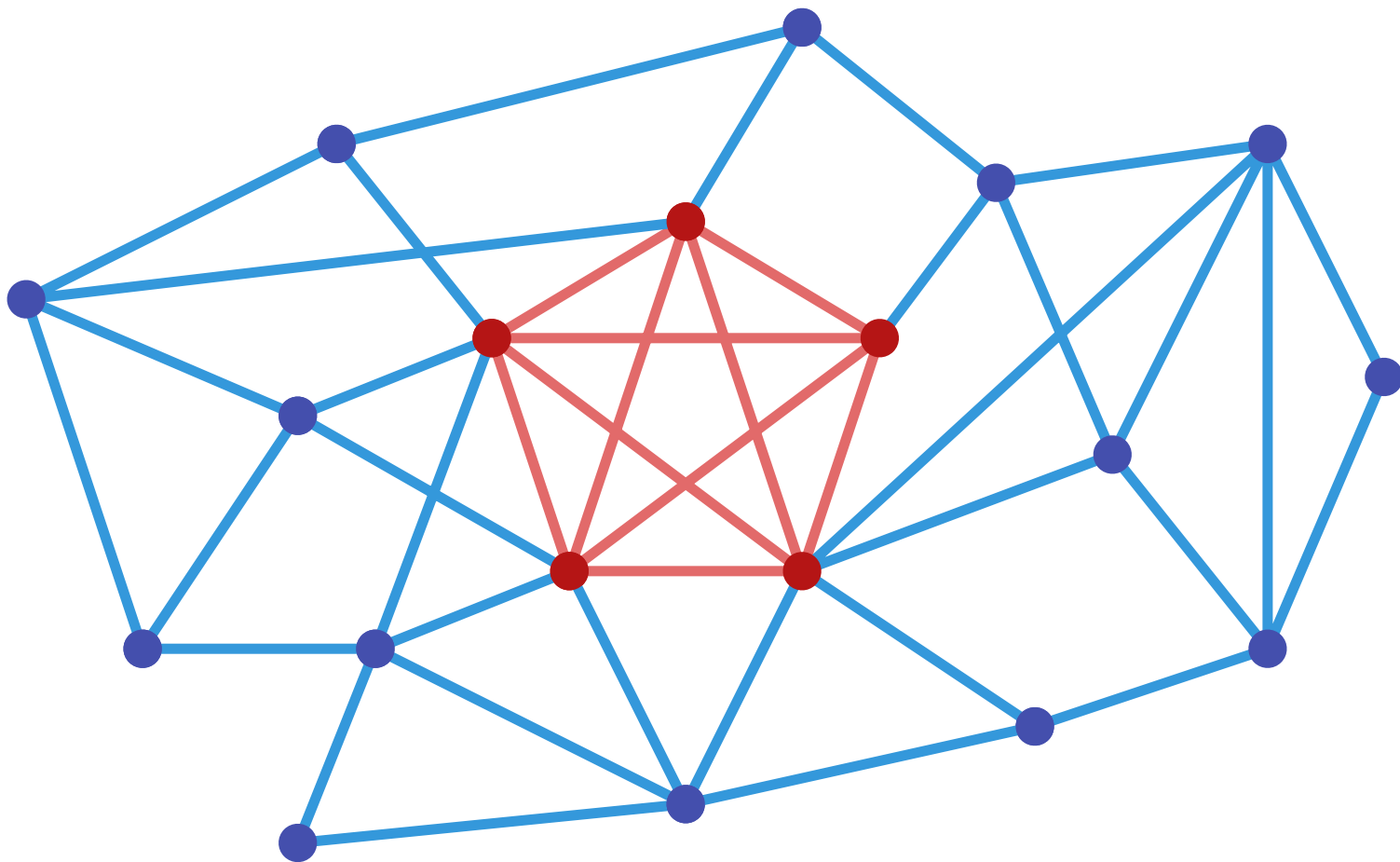
$$\text{"density"} = \frac{\# \text{ edges}}{\# \text{ vertices}}$$



Goal: compute densest subgraph

Densest
Subgraph

$$\text{"density"} = \frac{\# \text{ edges}}{\# \text{ vertices}}$$



Goal: compute densest subgraph

Applications of Flows + Cuts

Gameplan

1. Survey a bunch of problems

2. Start solving them one by one



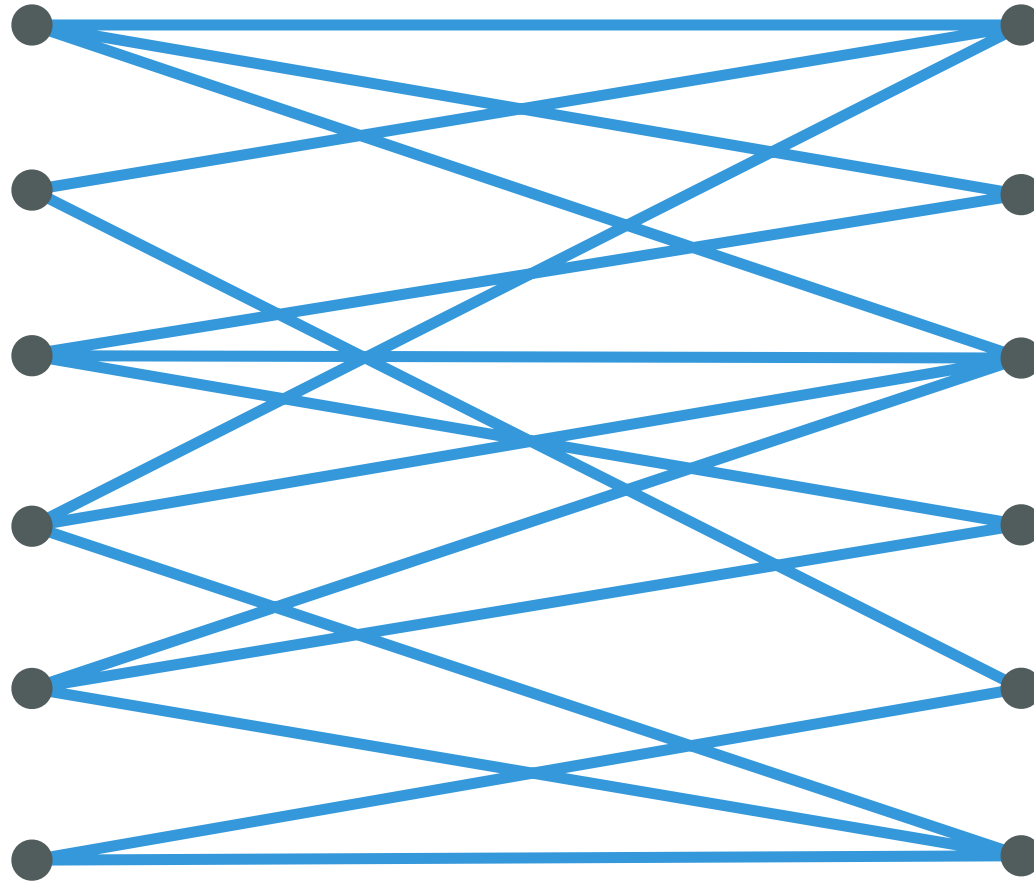
"only two algorithms"

① identify subproblems

② change the input

Bipartite Matching

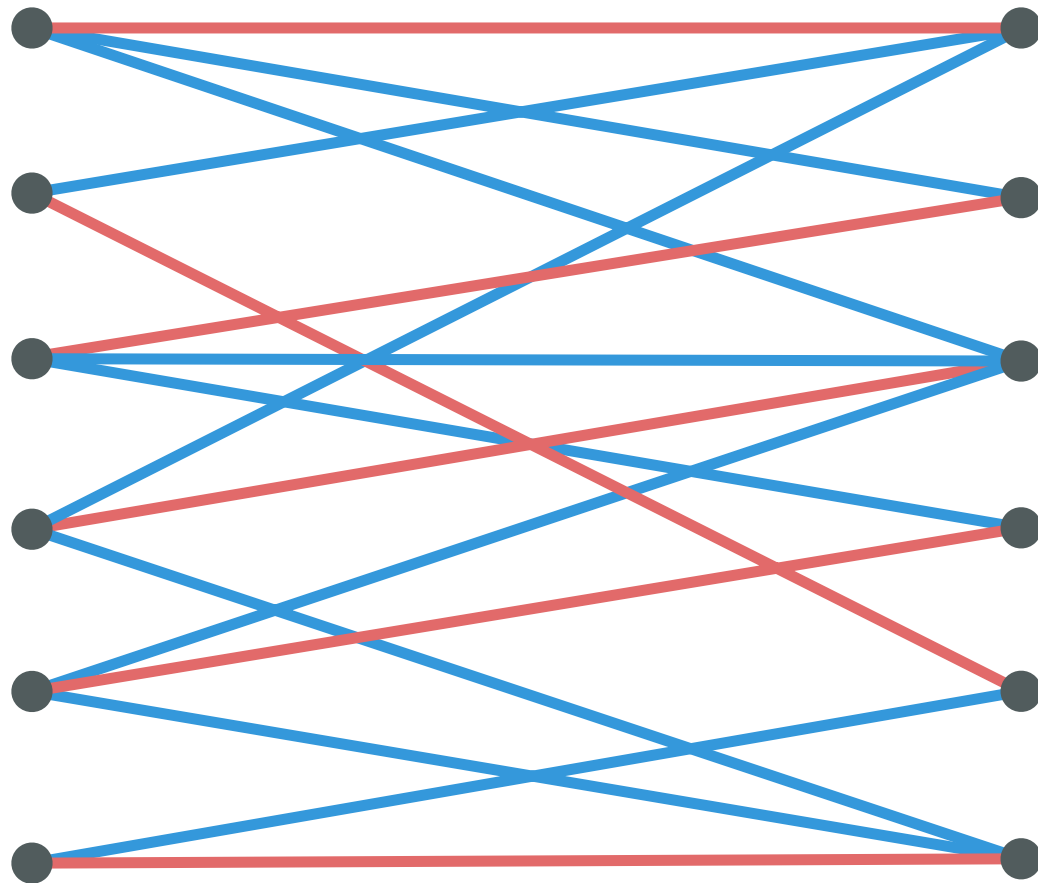
"Bipartite graph"



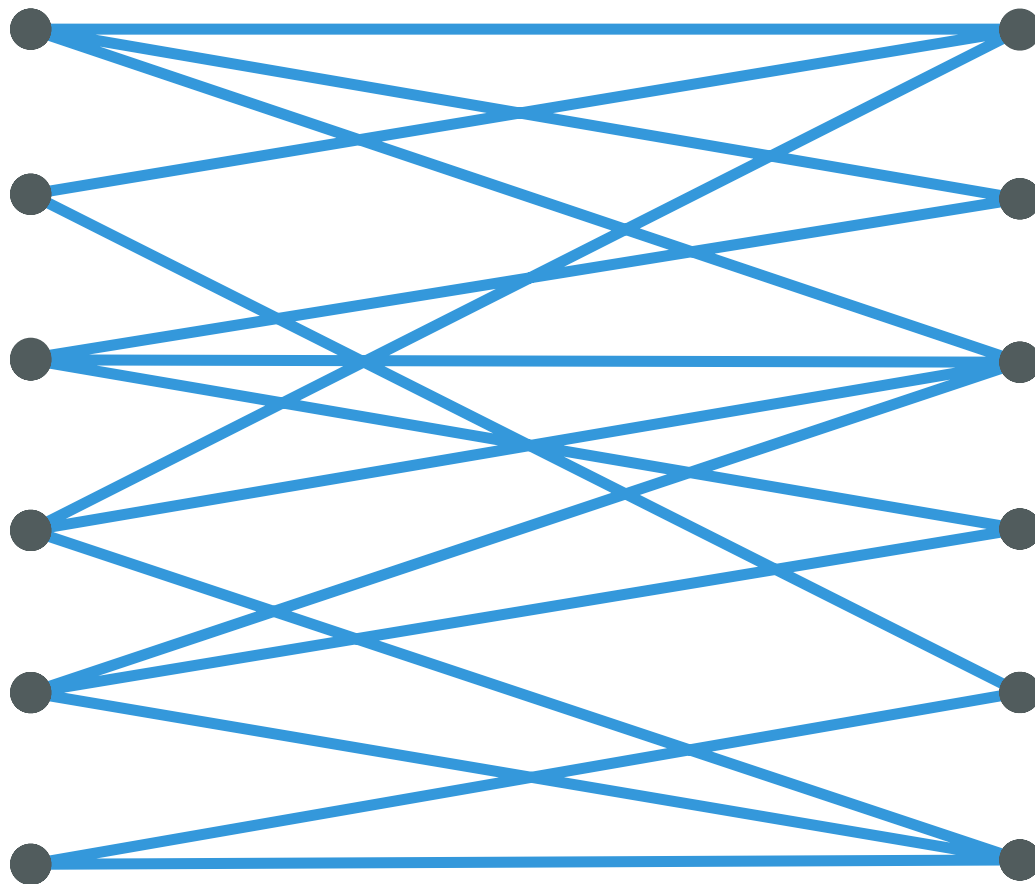
Goal: max set of edges w/ distinct endpoints
"matching"

Bipartite Matching

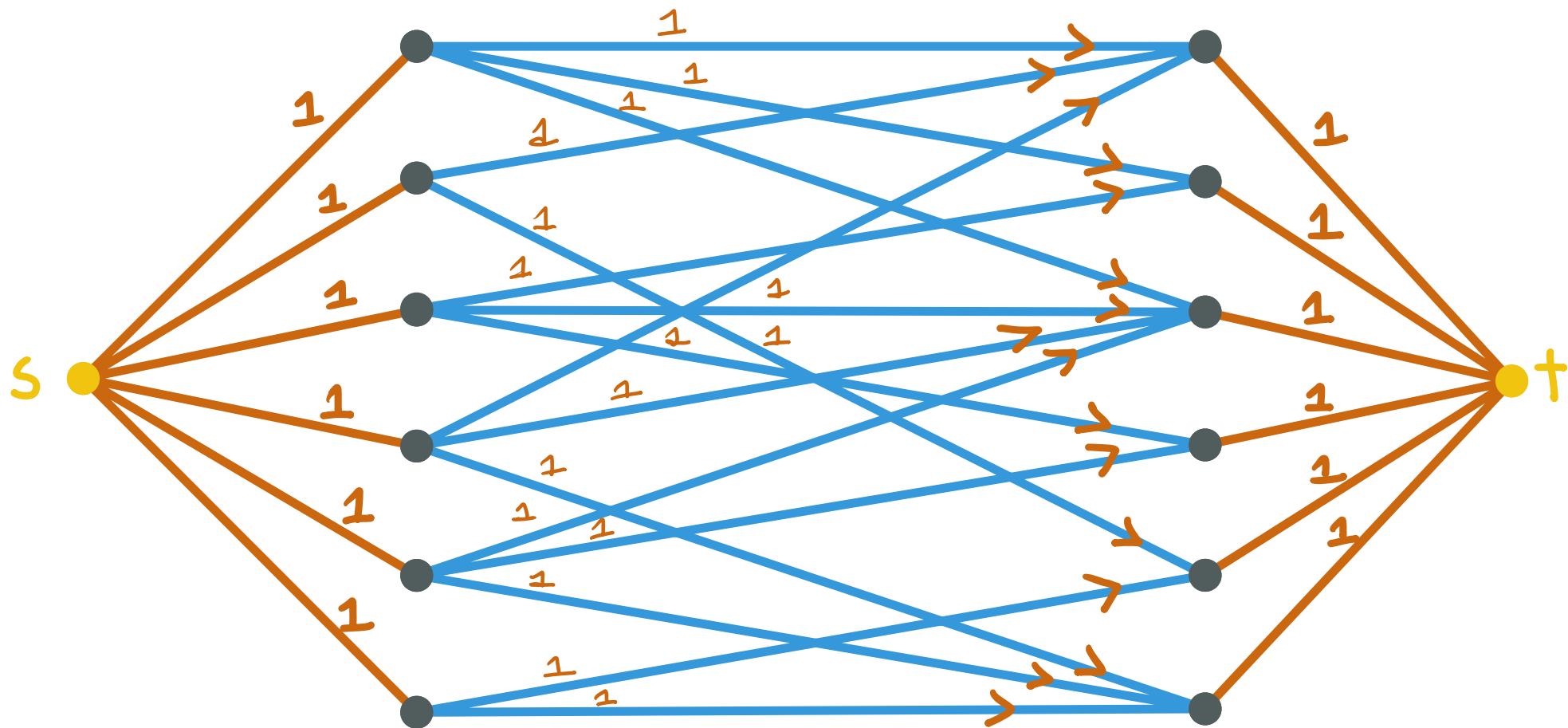
(bipartite graph)



Goal: max set of edges w/ distinct endpoints
"matching"



(see lecture notes for detailed proof)



Matching
w/ k edges

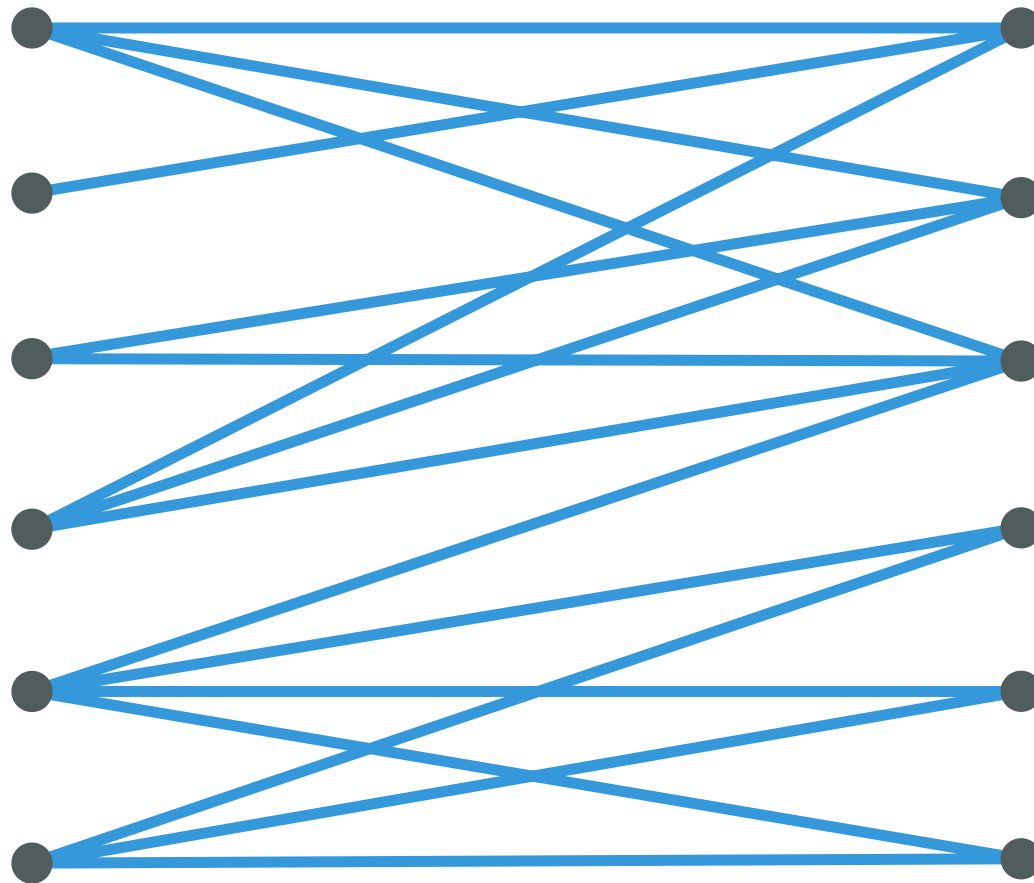


k edge-disjoint
 (s, t) -paths

(see lecture notes for detailed proof)

Bipartite
Vertex
Cover

"Bipartite graph"

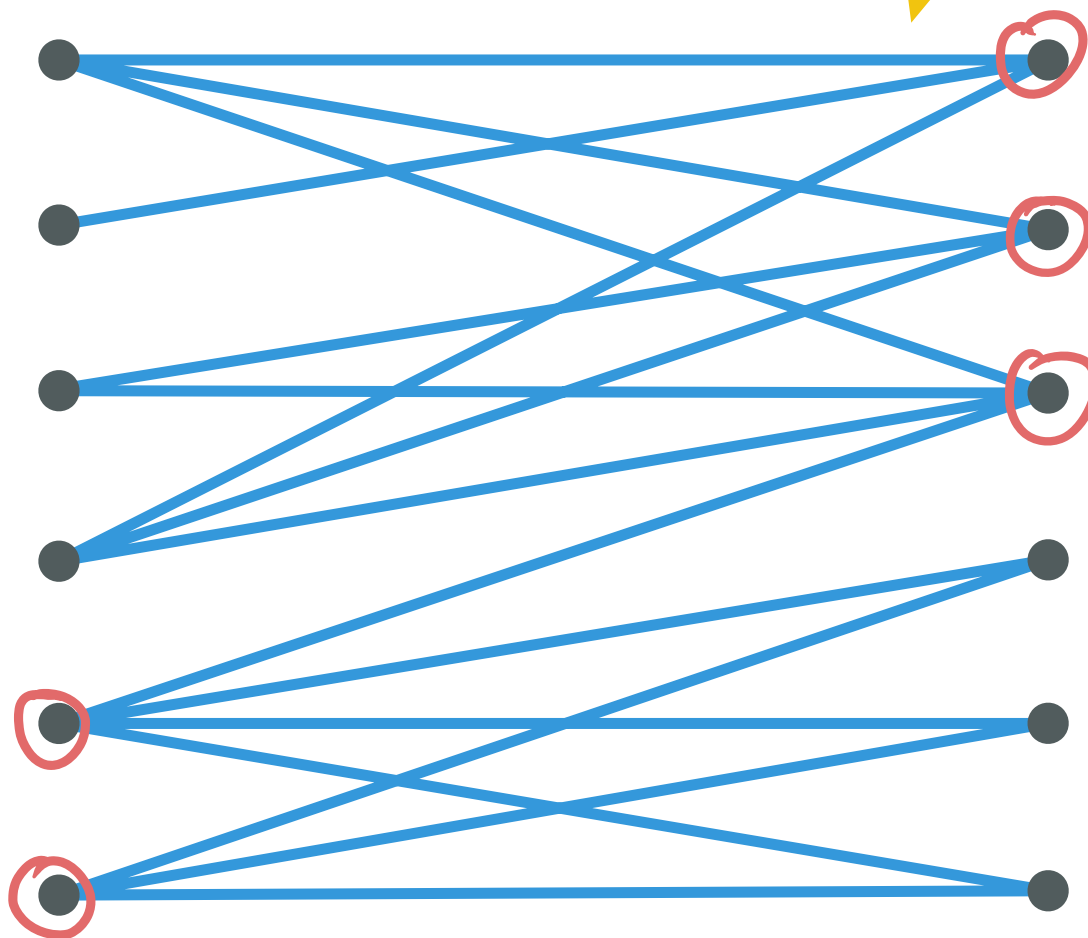


Goal: min set of vertices incident to all edges

"vertex cover"

Bipartite
Vertex
Cover

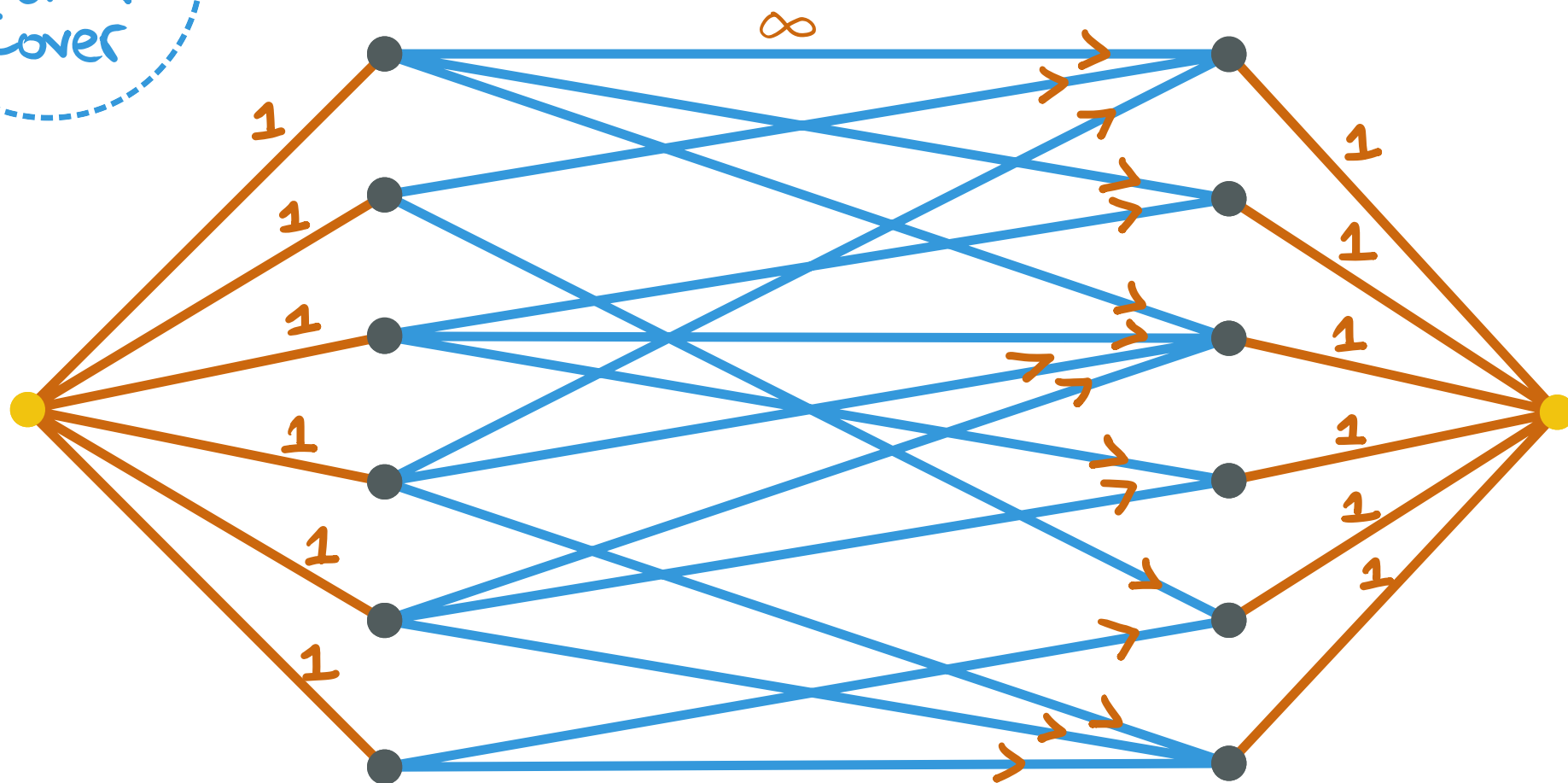
"Bipartite graph"



Goal: min set of vertices incident to all edges

"vertex cover"

Bipartite
Vertex
Cover

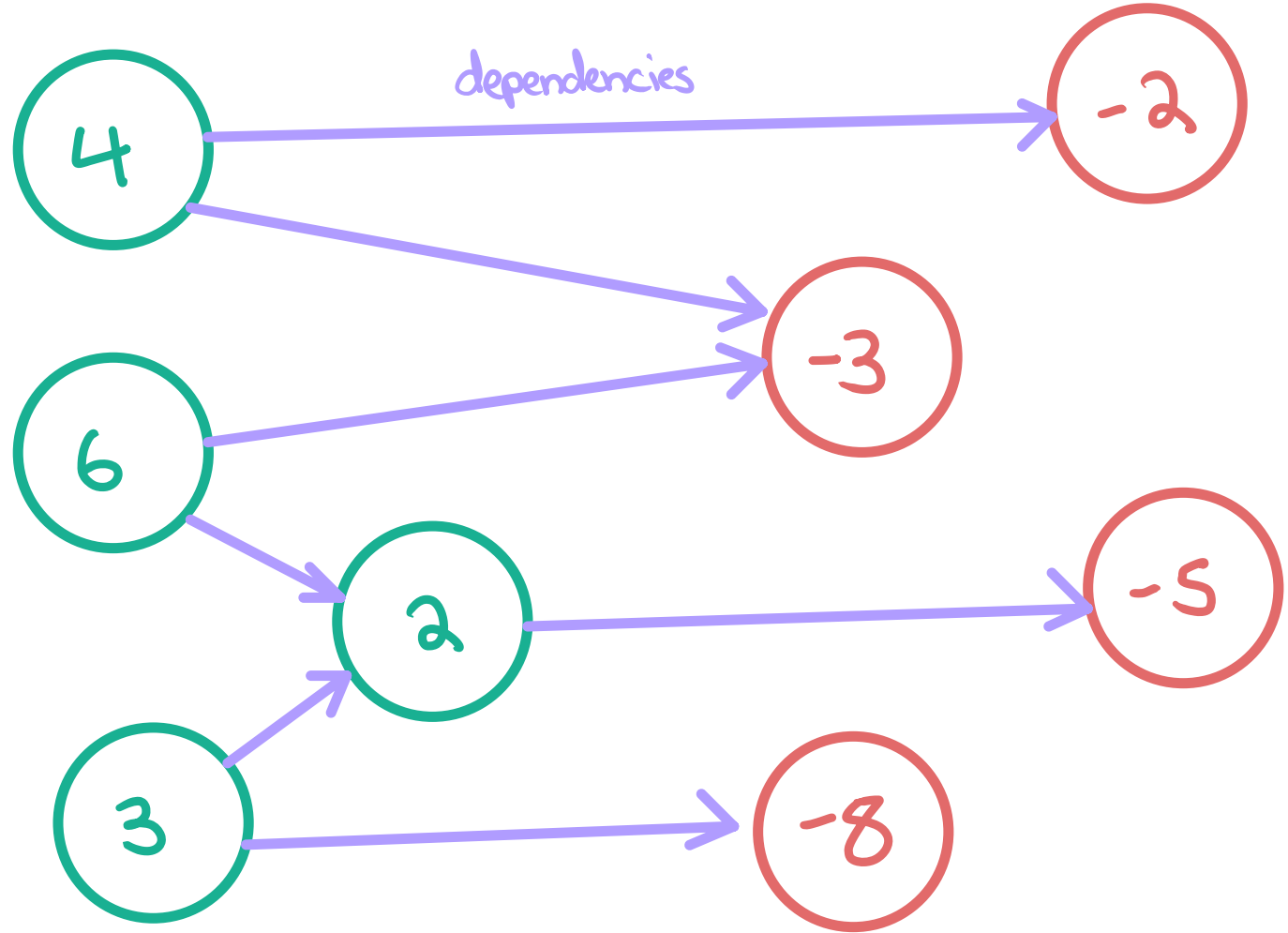


(s,t)-cut of
size k



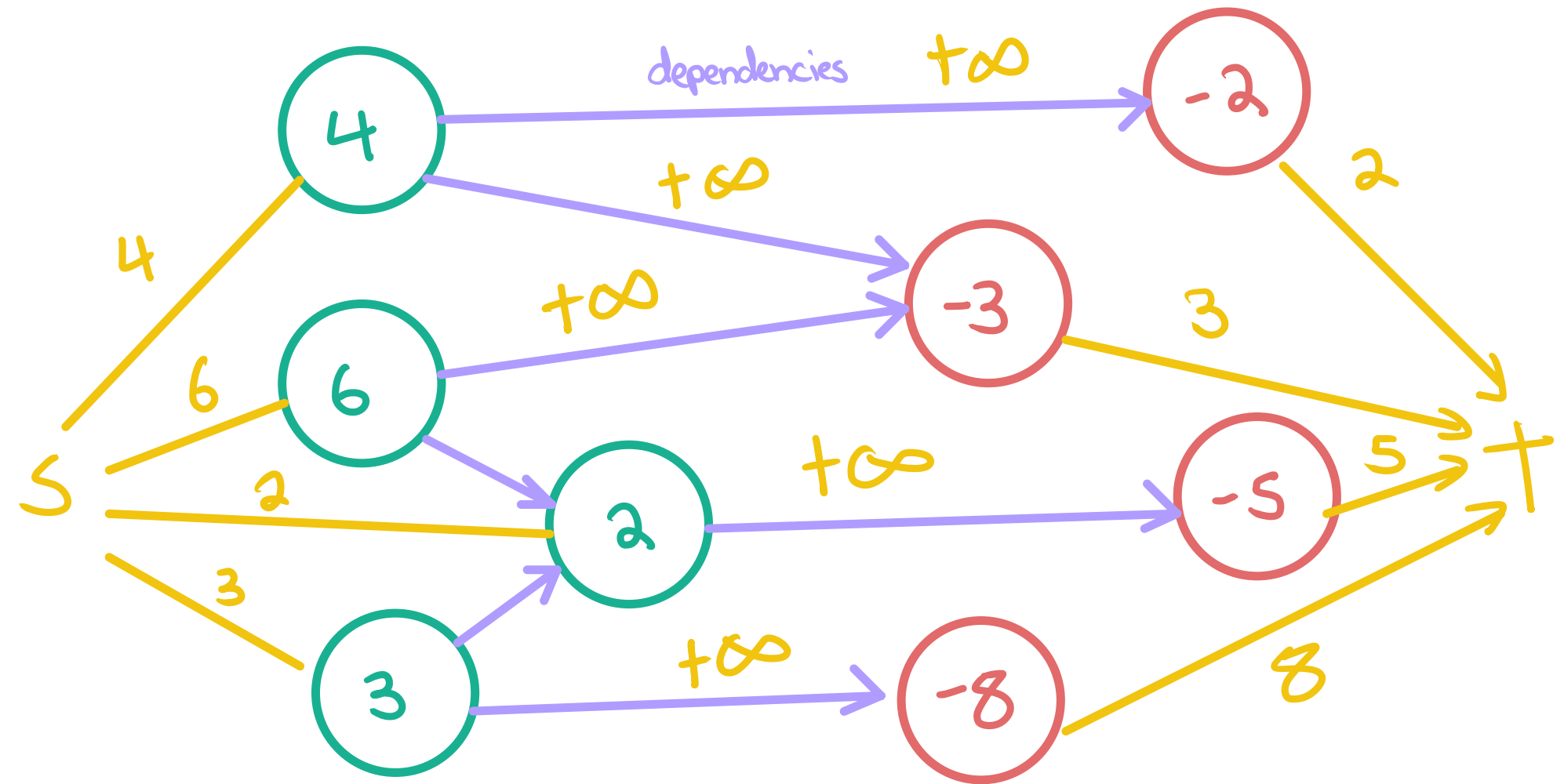
vertex cover
of size k

Project Selection

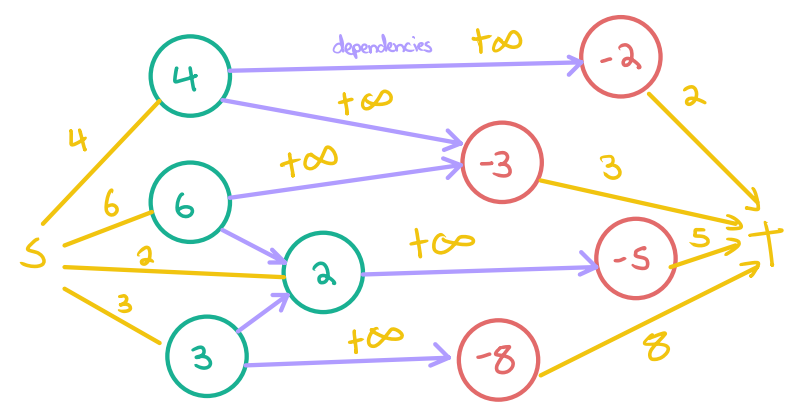


Projects have net profits (can be < 0 for costs) and dependencies

Goal: select projects (including dependencies) w/ max total profit



claim: min-cut $\Rightarrow$ best project selection



Playoffs
Elimination



Playoff Elimination



can Boston catch the Yankees?

Team	Wins	Games left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	90	2

Remaining schedule

(wins)	(92)	(91)	(91)	(90)
	New York	Baltimore	Toronto	Boston
New York	X	1	1	0
Baltimore	1	X	1	1
Toronto	1	1	X	1
Boston	0	1	1	X

(wins)	(92) New York	(91) Baltimore	(91) Toronto	(90) Boston
New York	X	1	1	0
Baltimore	1	X	1	1
Toronto	1	1	X	1
Boston	0	1	1	X

New York vs Baltimore

New York vs Toronto

Baltimore vs Toronto

Baltimore vs Boston

Toronto vs Boston

New York

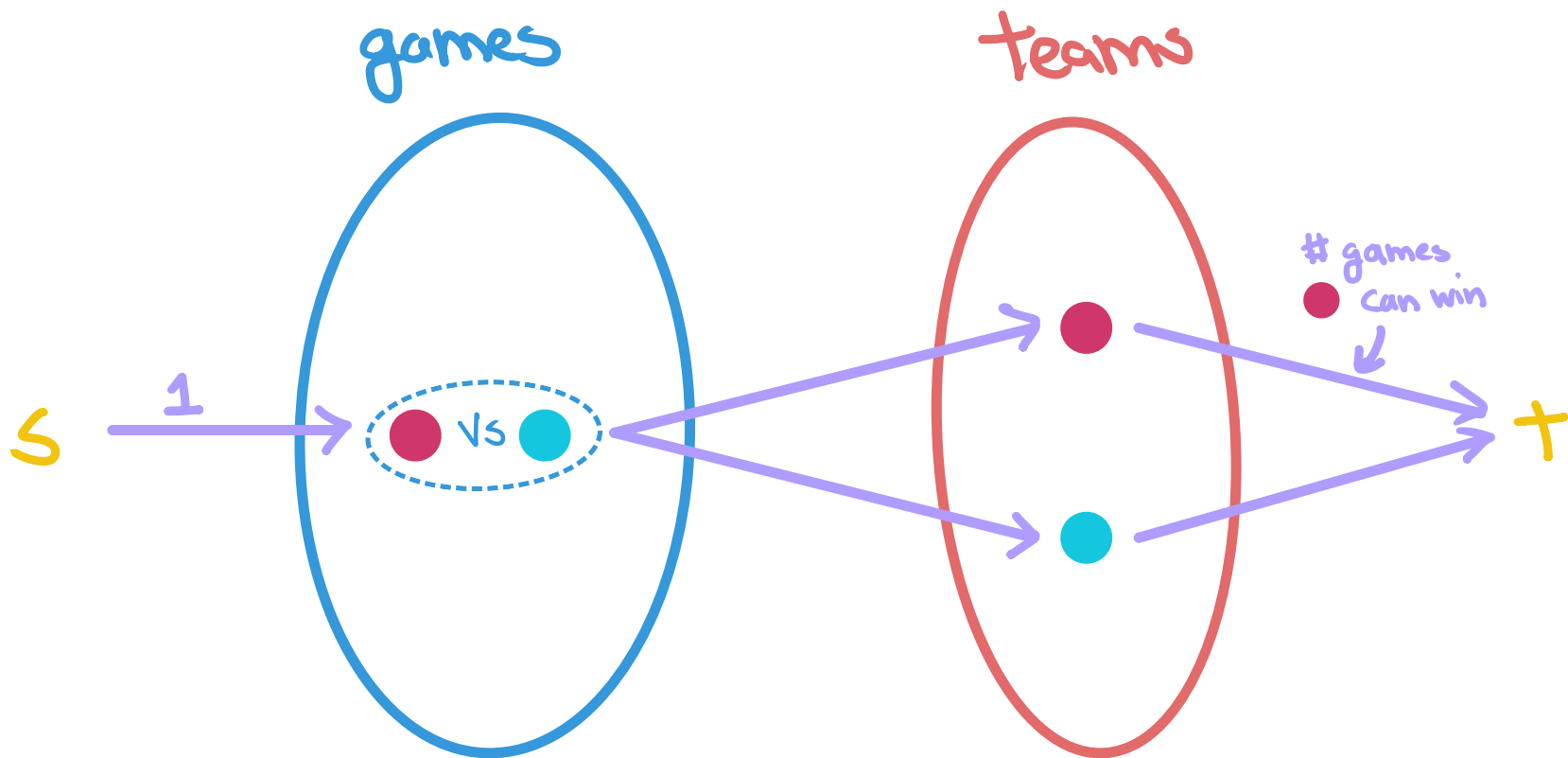
Baltimore

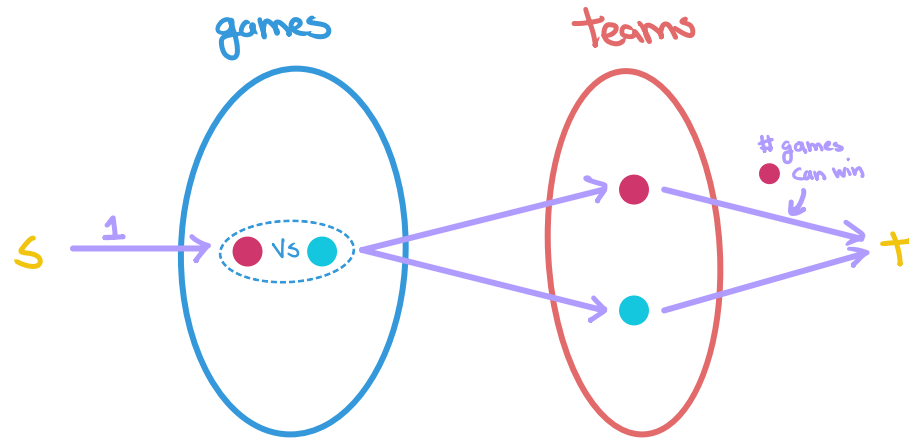
Toronto

Boston

(wins)	(92) New York	(91) Baltimore	(91) Toronto	(90) Boston
New York	×	1	1	0
Baltimore	1	×	1	1
Toronto	1	1	×	1
Boston	0	1	1	×

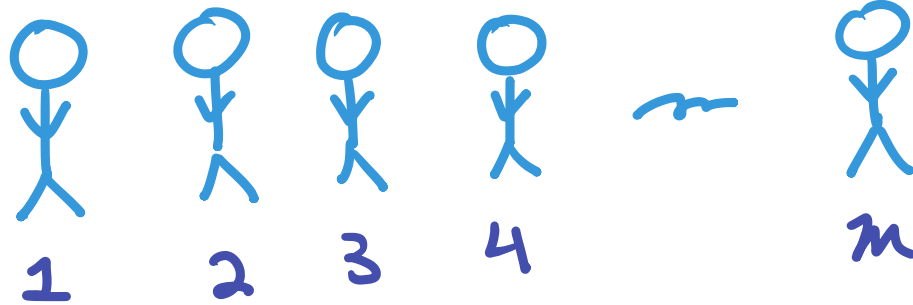




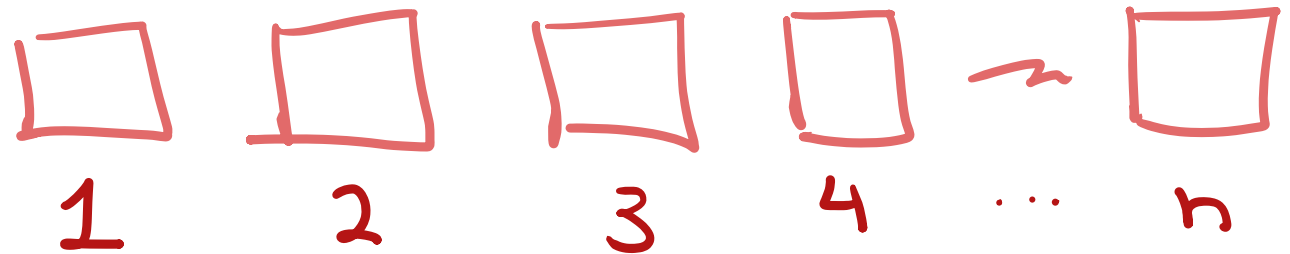


Assignment Problems

m individuals



n jobs



goal: fill as many jobs as possible w/ individuals

the catch: not everyone is qualified for all jobs

Assignment Problems

"Qualification matrix"

	☐	☐	☐	☐
	1	1	1	0
	0	0	1	0
	0	0	0	1
	0	0	0	1

"1" = qualified

"0" = not qualified

Assignment Problems

individuals



jobs



	1	1	1	0
	0	0	1	0
	0	0	0	1
	0	0	0	1

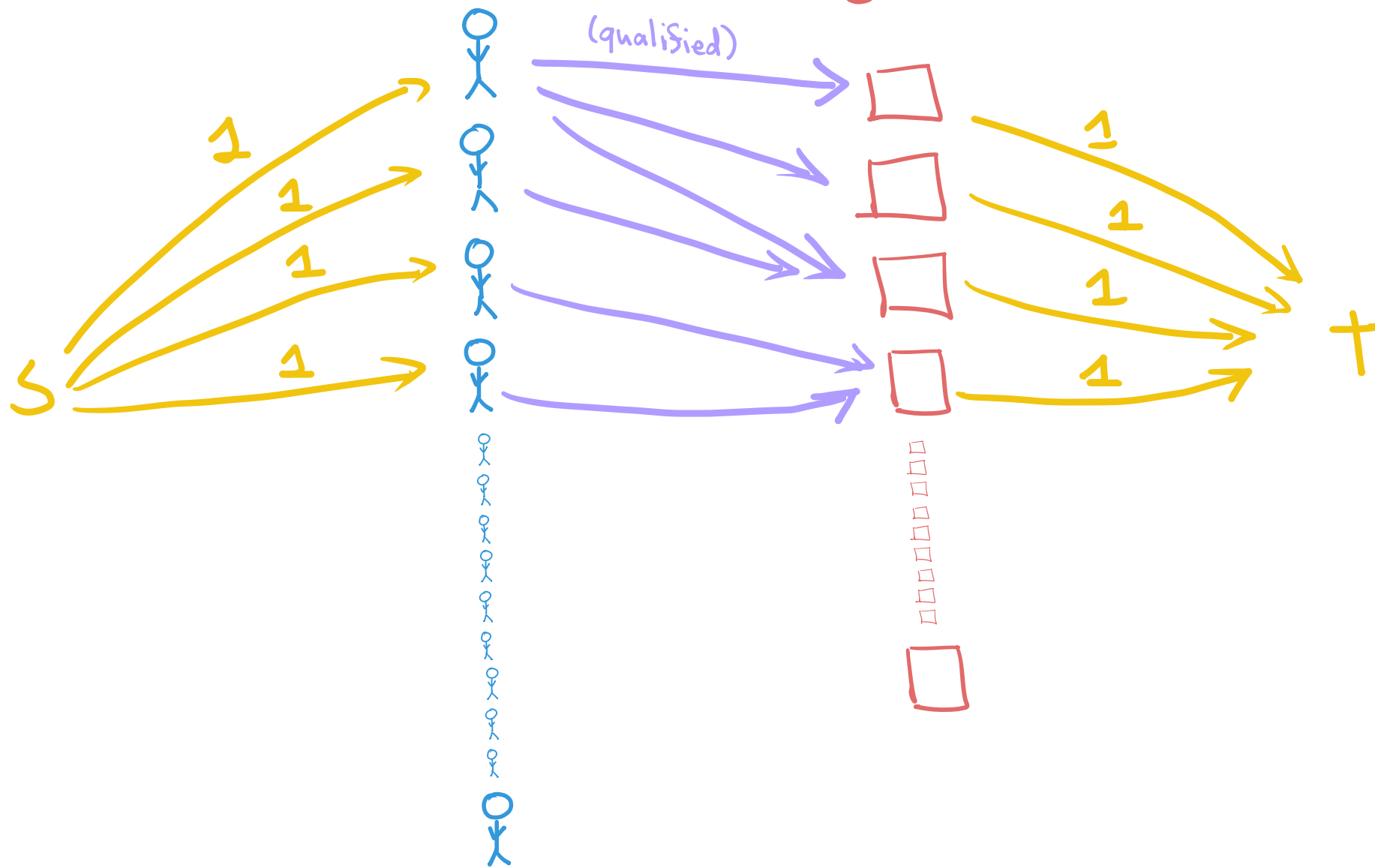
3

Assignment Problems

	☐	☐	☐	☐
☐	1	1	1	0
☐	0	0	1	0
☐	0	0	0	1
☐	0	0	0	1

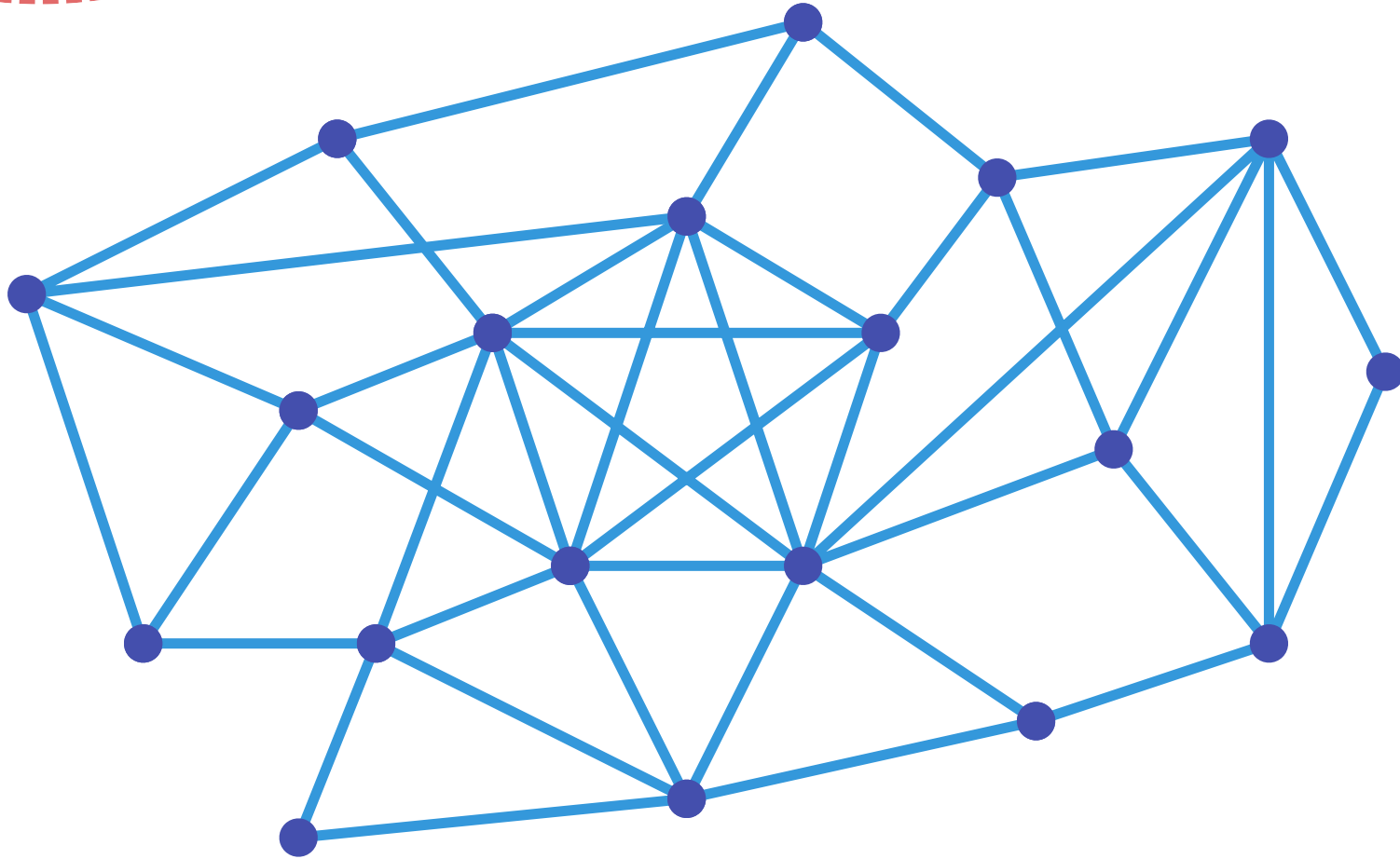
individuals

jobs



Edge Orientation

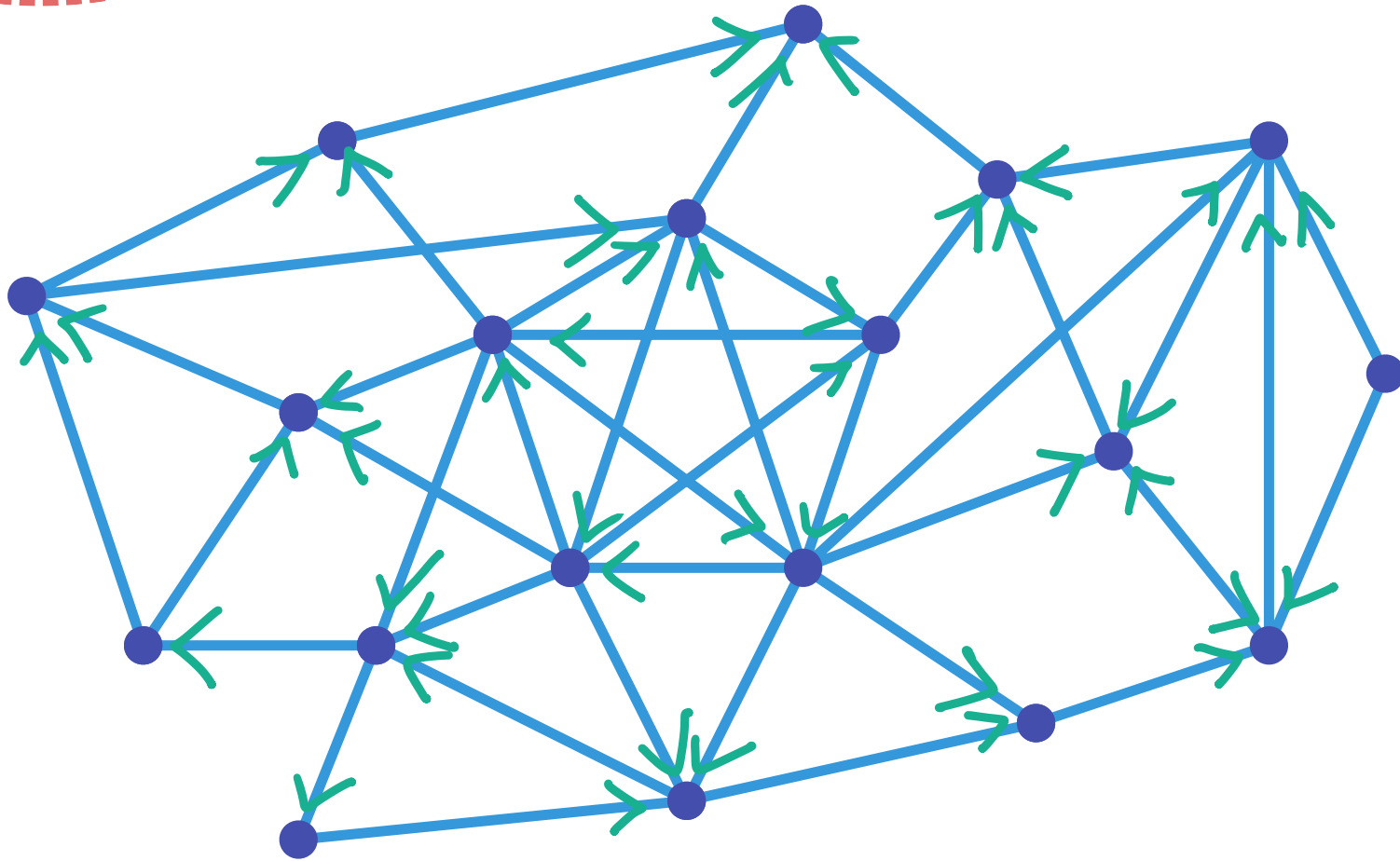
input: undirected graph



Goal: direct the edges to minimize the max density

Edge
Orientation

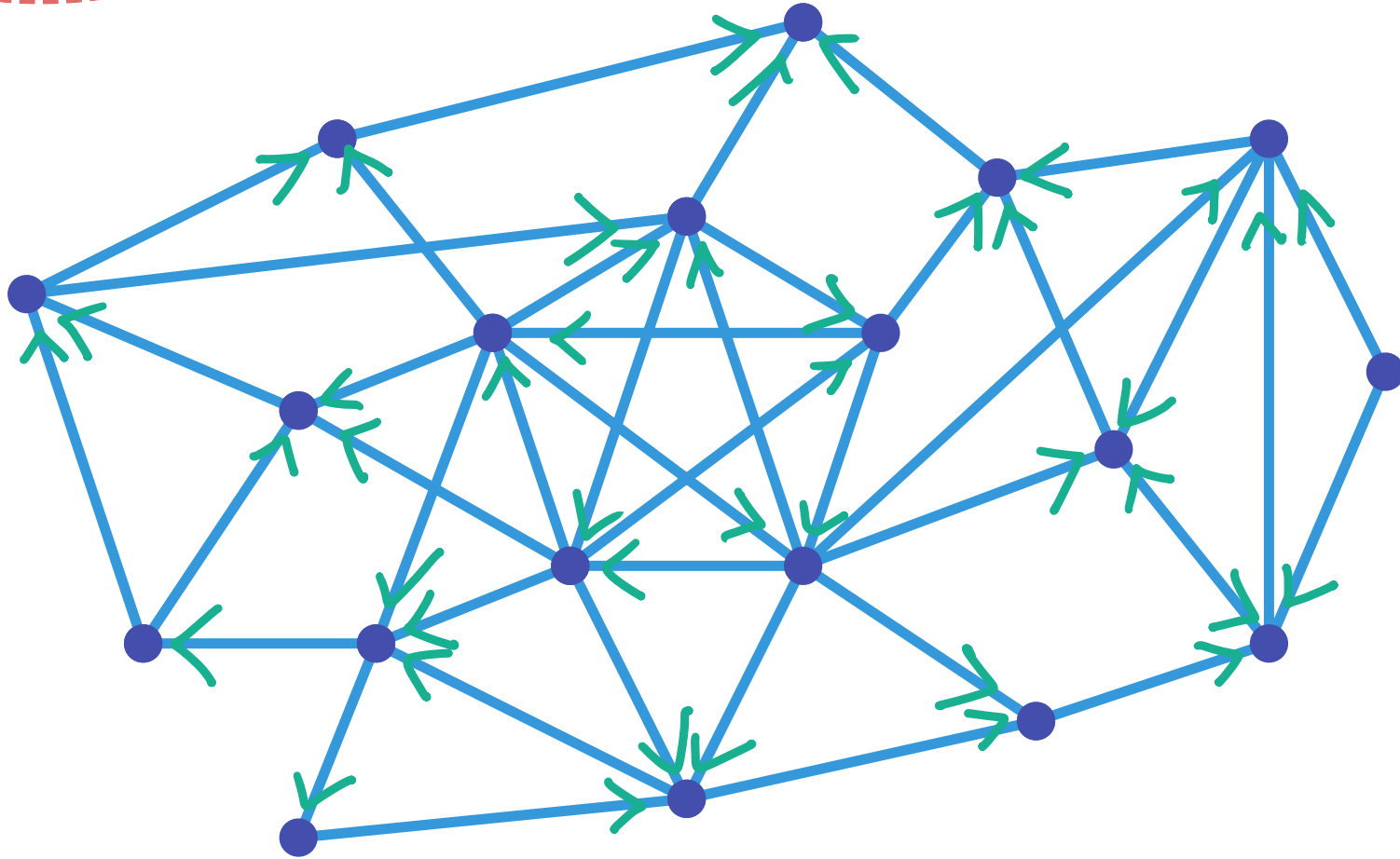
input: undirected graph



Goal: direct the edges to minimize the max in-degree

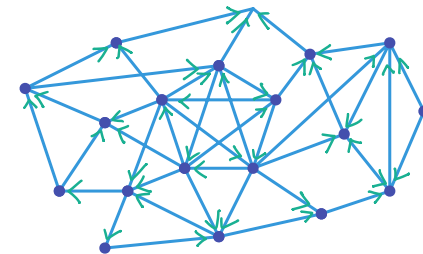
Edge
Orientation

input: undirected graph



Decision version: given $h > 0$, is there an orientation w/
 $\max \text{ in-degree} \leq h$?

Edge Orientation

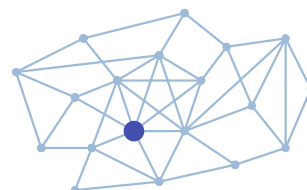
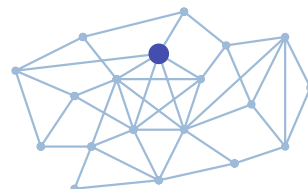
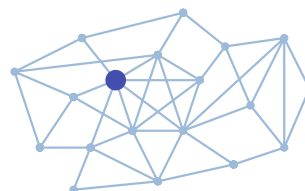
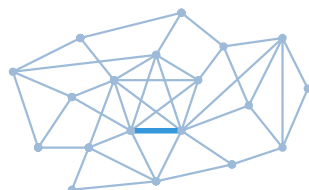


edges

vertices

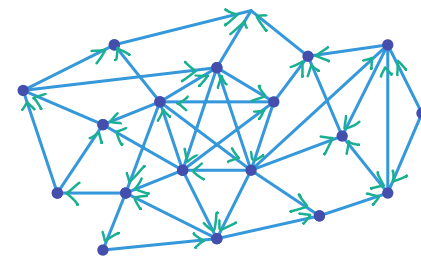
$h = \text{target in-degree}$

Σ



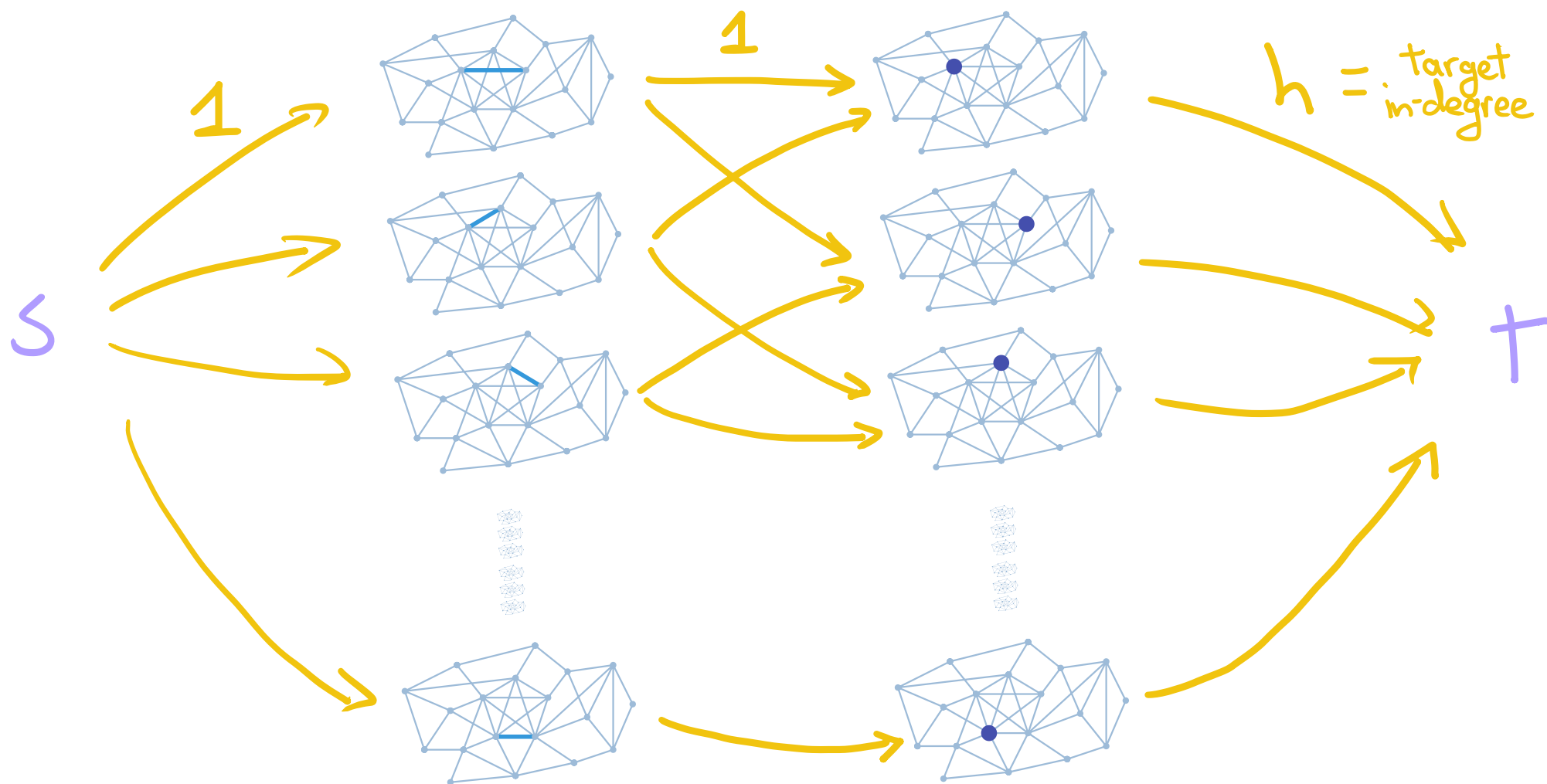
$+$

Edge Orientation



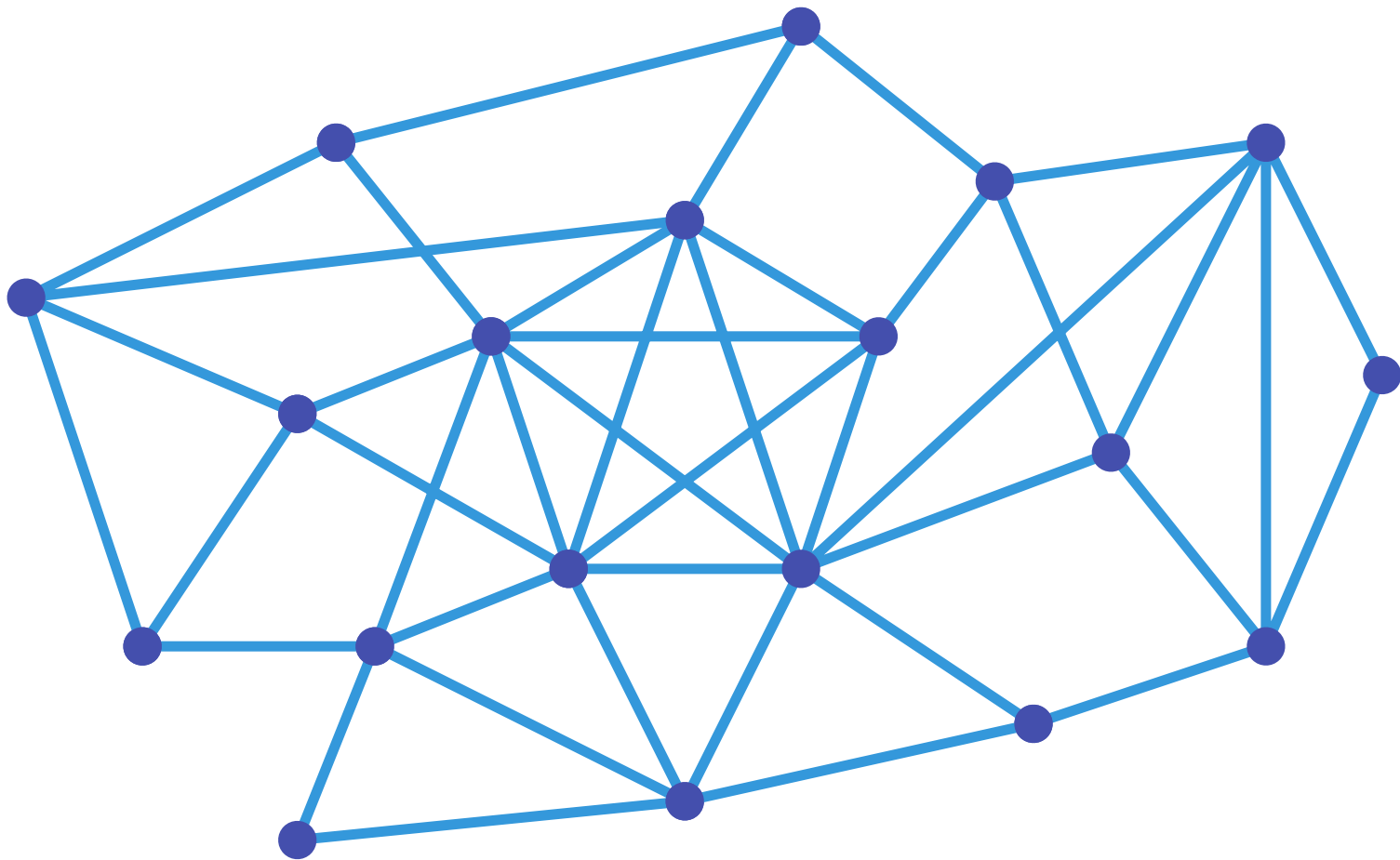
edges

vertices



Densest
Subgraph

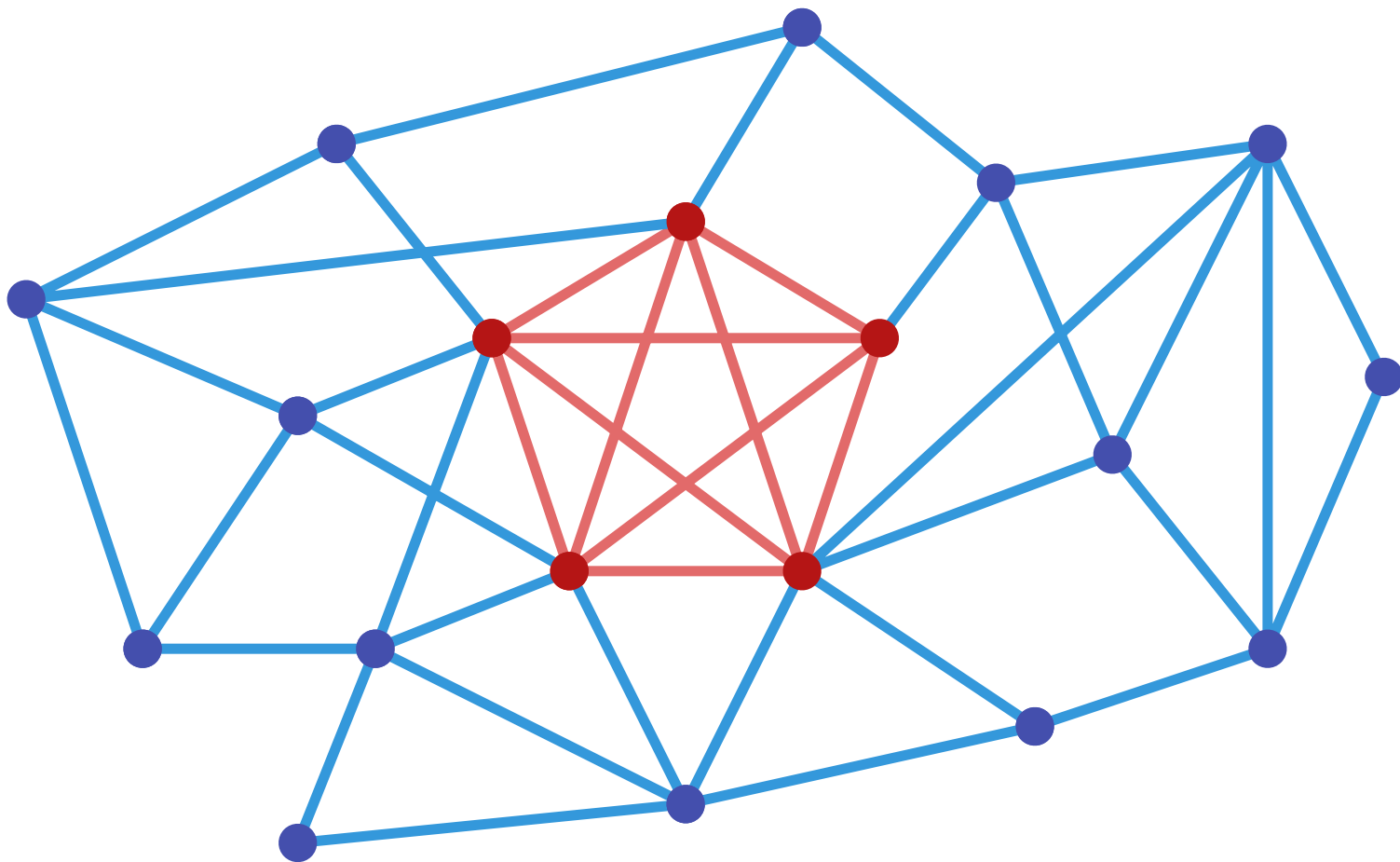
$$\text{"density"} = \frac{\# \text{ edges}}{\# \text{ vertices}}$$



Goal: compute densest subgraph

Densest
Subgraph

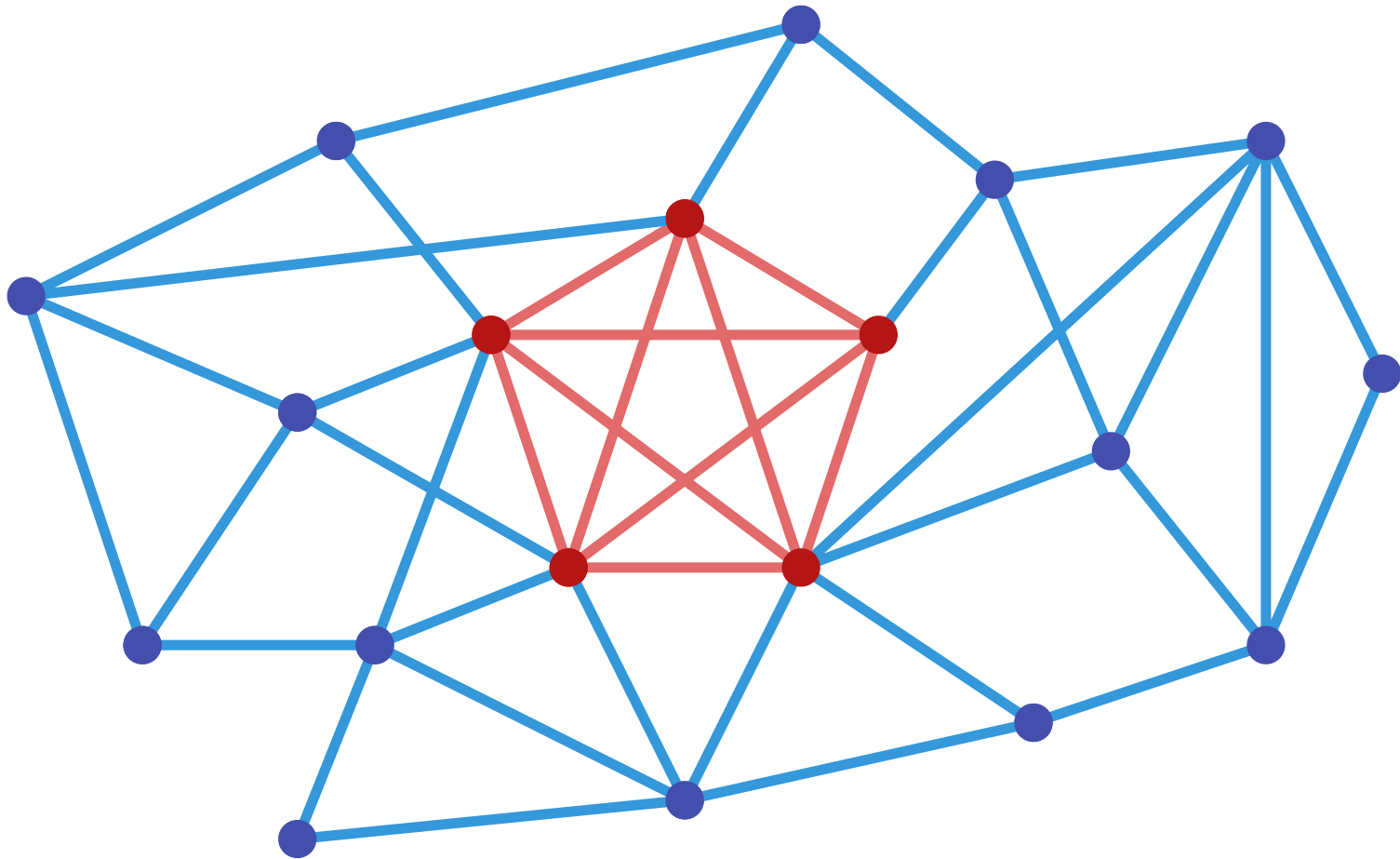
$$\text{"density"} = \frac{\# \text{ edges}}{\# \text{ vertices}}$$



Goal: compute densest subgraph

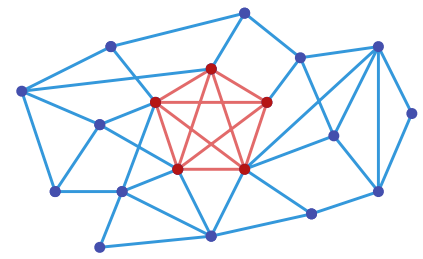
Densest
Subgraph

$$\text{"density"} = \frac{\# \text{ edges}}{\# \text{ vertices}}$$



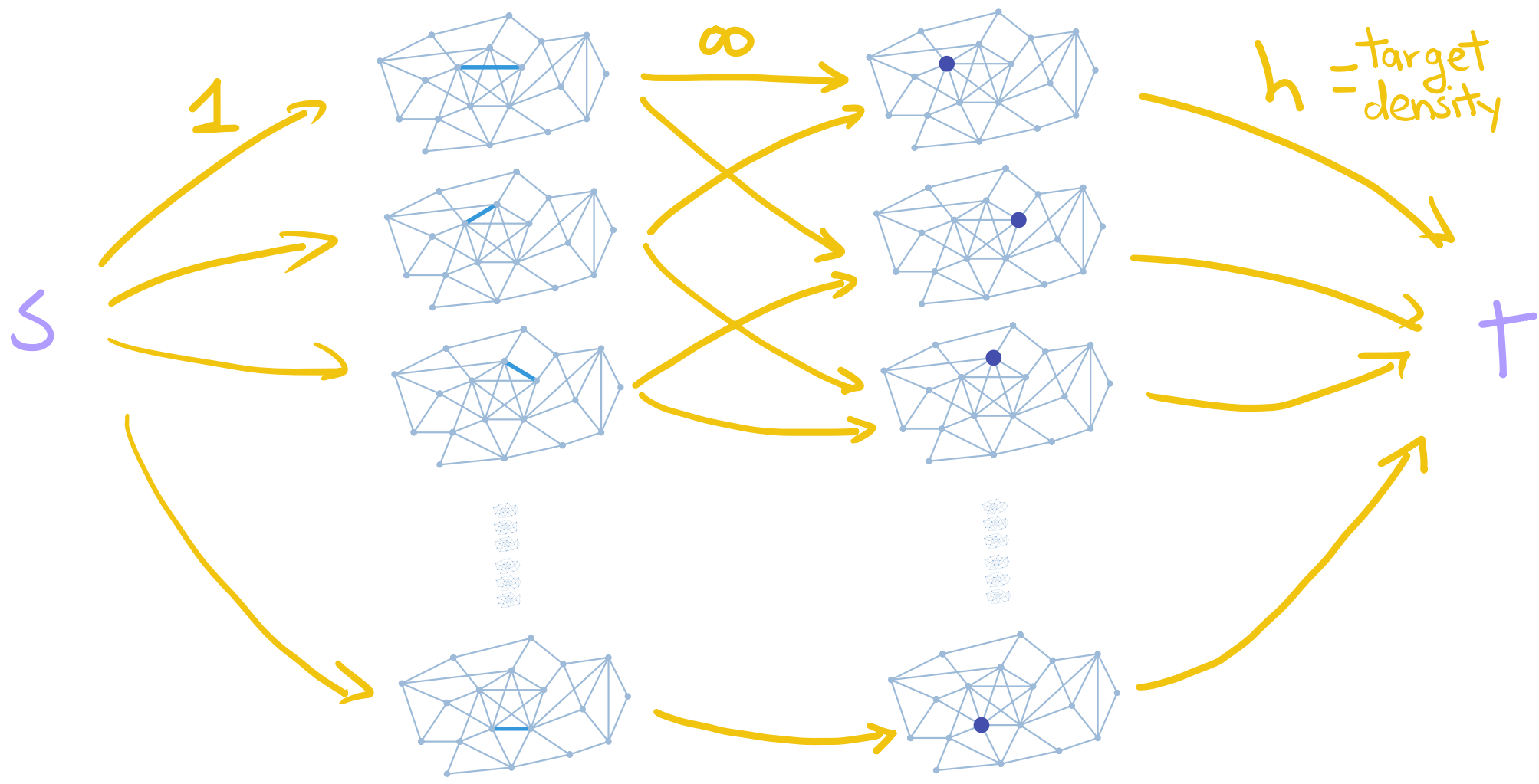
Decision version: Given $h > 0$, is there a subgraph w/
density $> h$?

Densest Subgraph

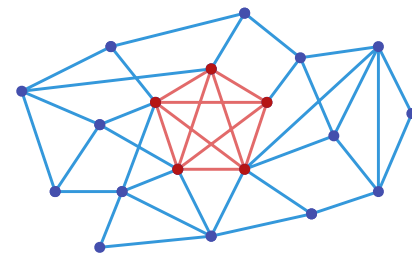
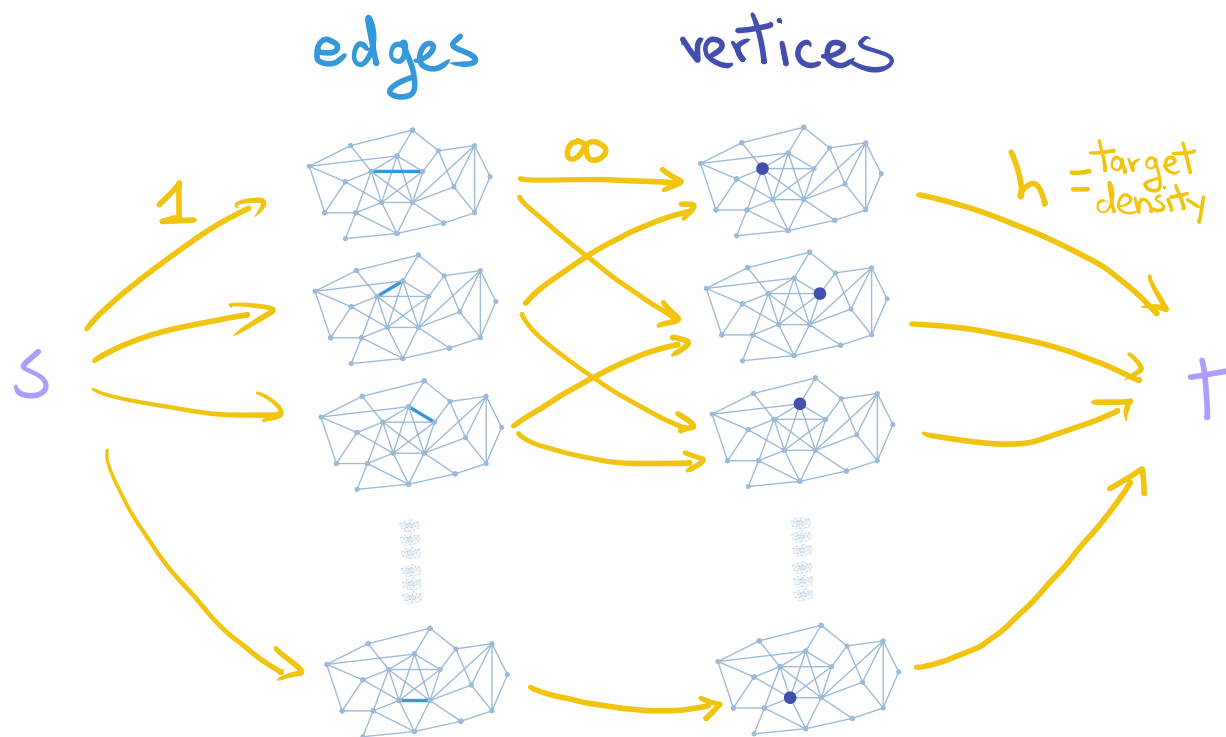


edges

vertices



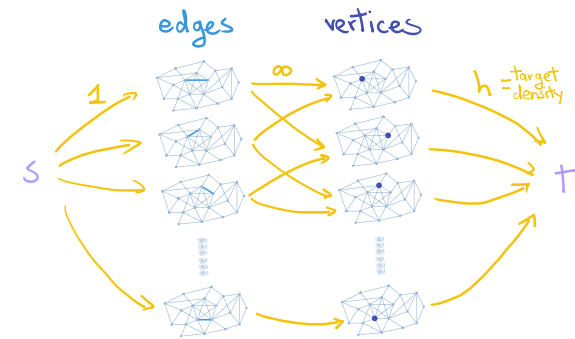
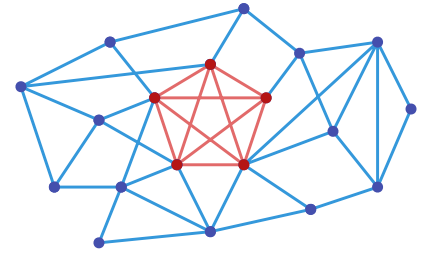
Densest Subgraph



claim: if $\text{max-flow} = m$, then all subgraphs have density $\leq h$

A subgraph of density $> h$ induces (s,t) -cut of size $< m$

claim: if $\text{max-flow} = m$, then all subgraphs have density $\leq h$



A subgraph of density $> \lambda$ induces (s,t) -cut of size $< m$

