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

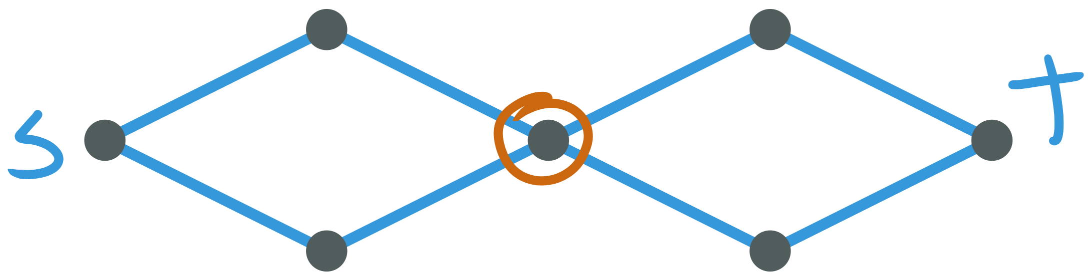
2. Start solving them one by one

"only two algorithms"*

① identify subproblems

② change the input

Vertex Disjoint Paths



2 edge disjoint paths

1 vertex disjoint path

vertex cut = 1

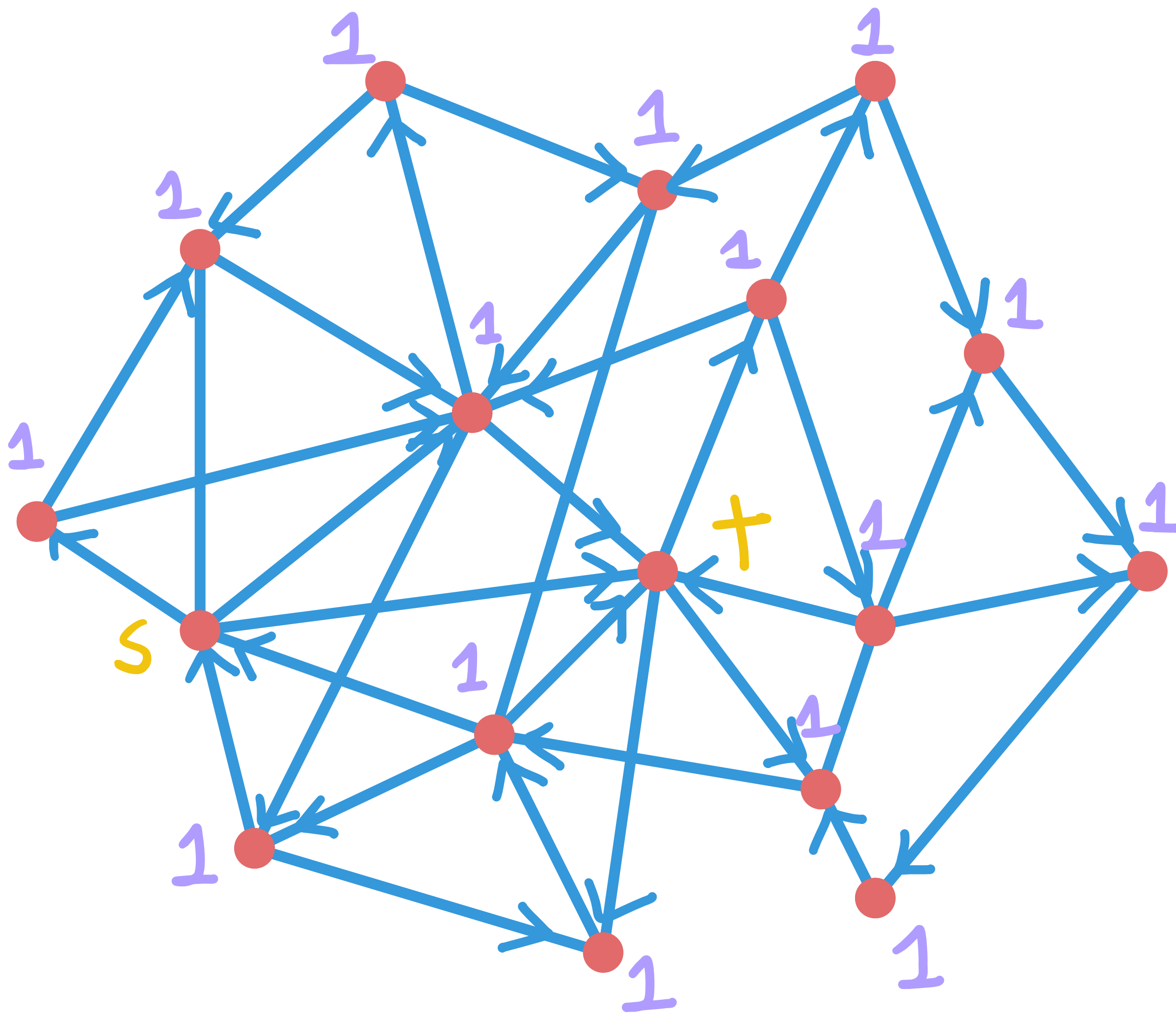
"vertex capacities" $c : V \to \mathbb{R}_{>0}$

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

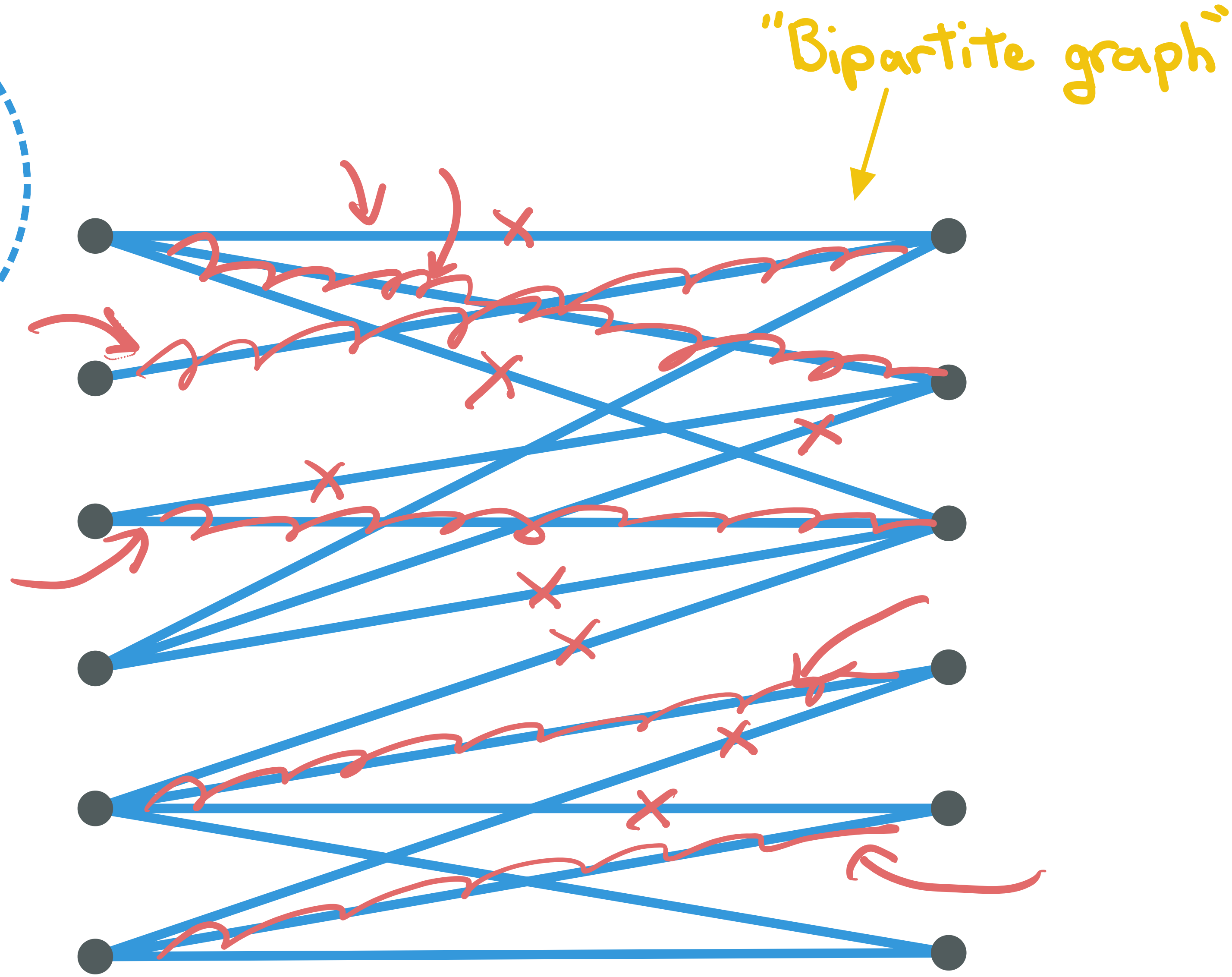
e.g. $c(v) = 1$

$\forall v \notin \{s,t\}$

$\Rightarrow$ <u>vertex</u> disjoint paths



Bipartite Matching

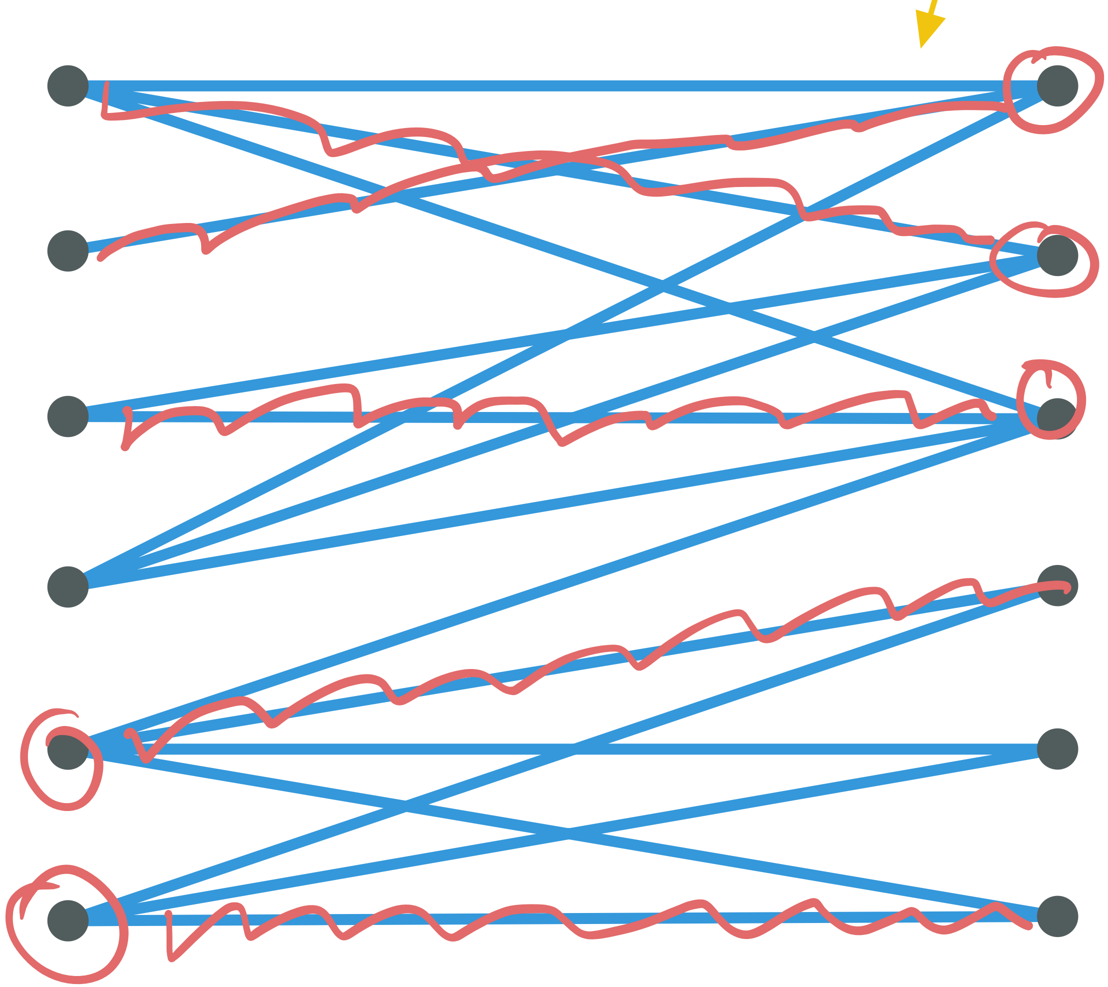


Goal: max set of edges w/ distinct endpoints

"matching"

Bipartite Vertex Cover

"Bipartite graph"

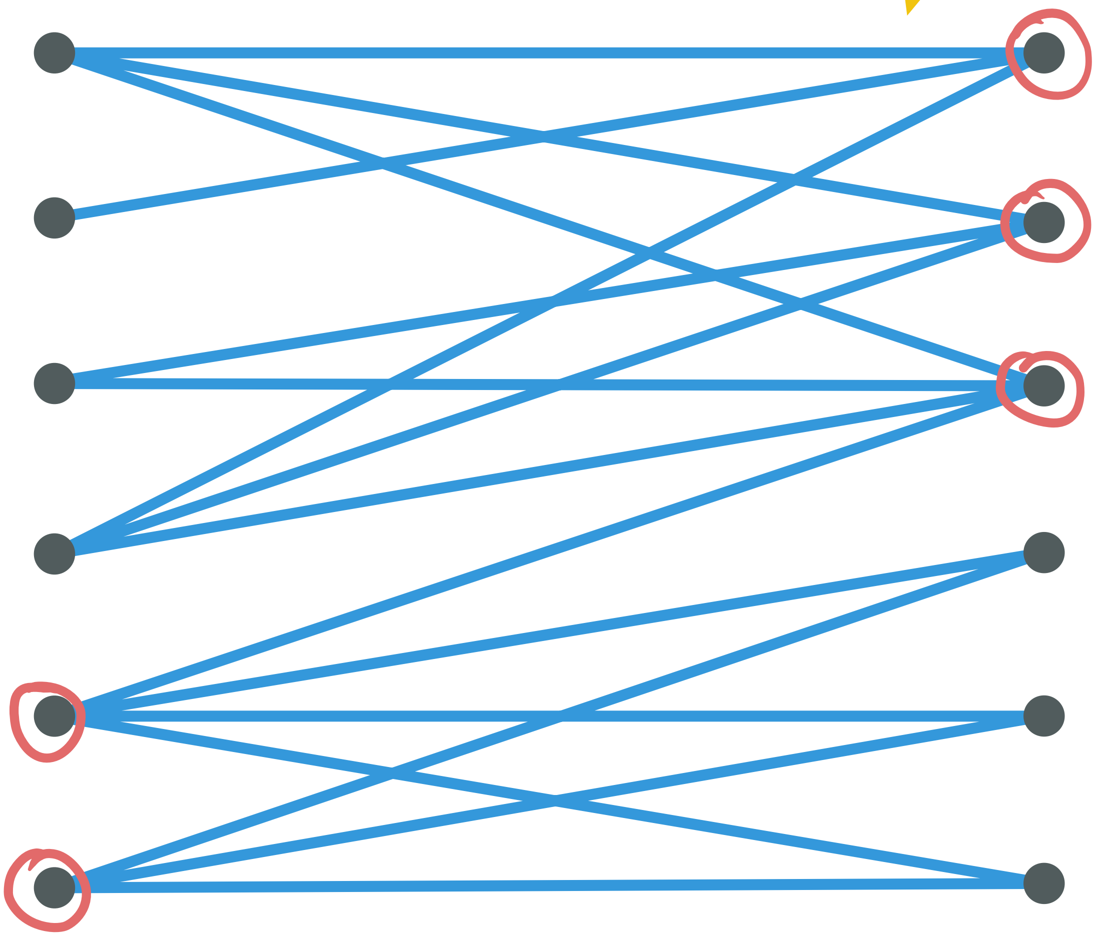


Goal: min set of vertices incident to all edges

"vertex cover"

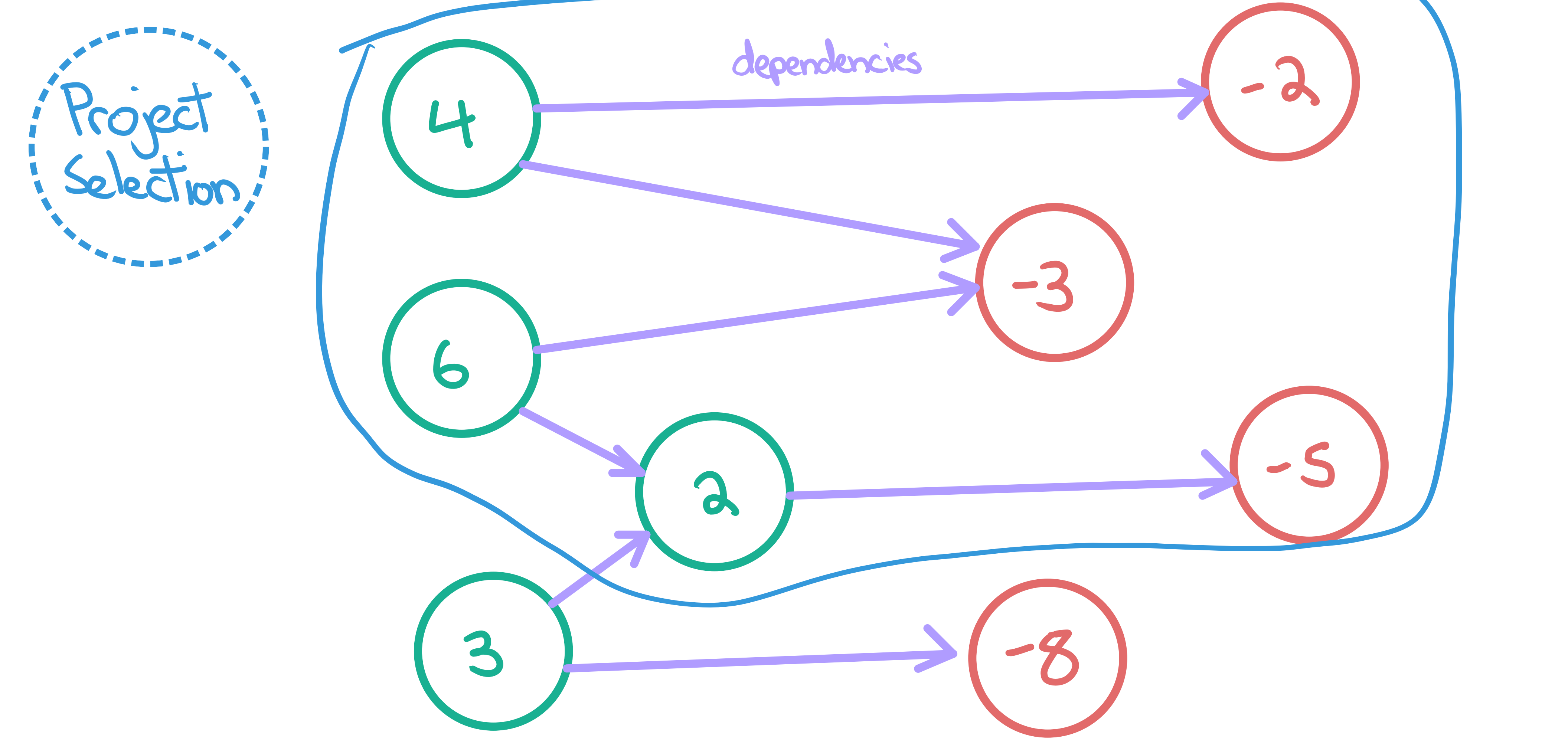
Bipartite Vertex Cover

"Bipartite graph"



Goal: min set of vertices incident to all edges

"vertex cover"



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

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

Playoffs Elimination

Playoffs Elimination

can Boston catch the Yankees?

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

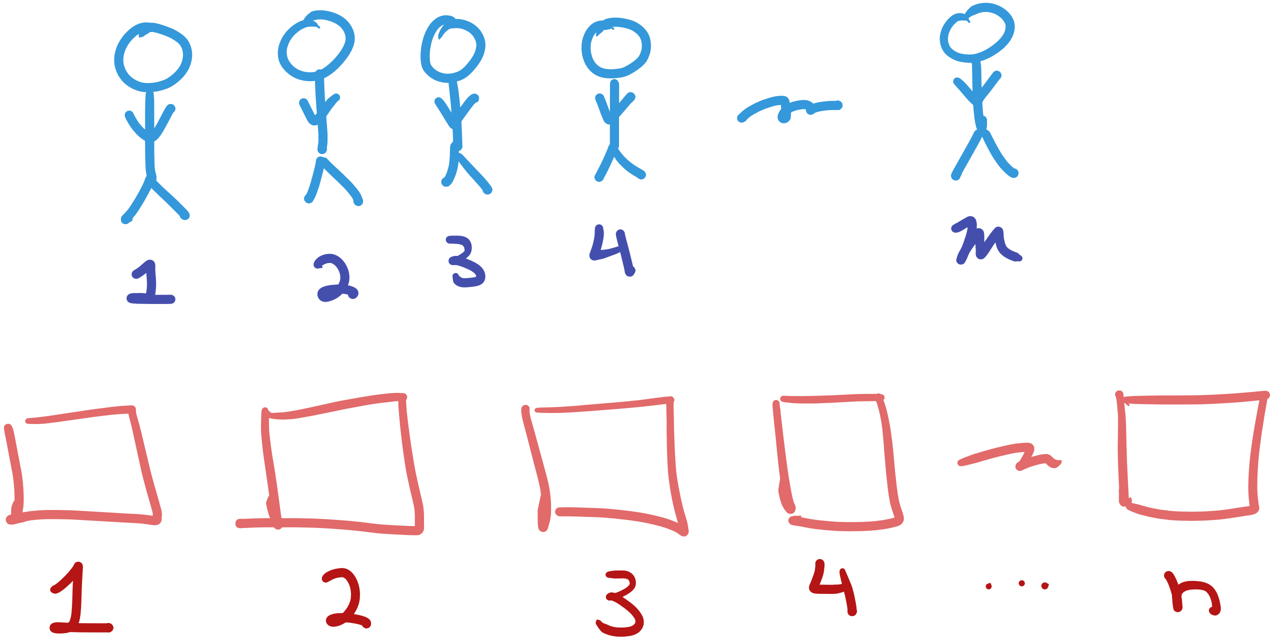
Remaining schedule

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

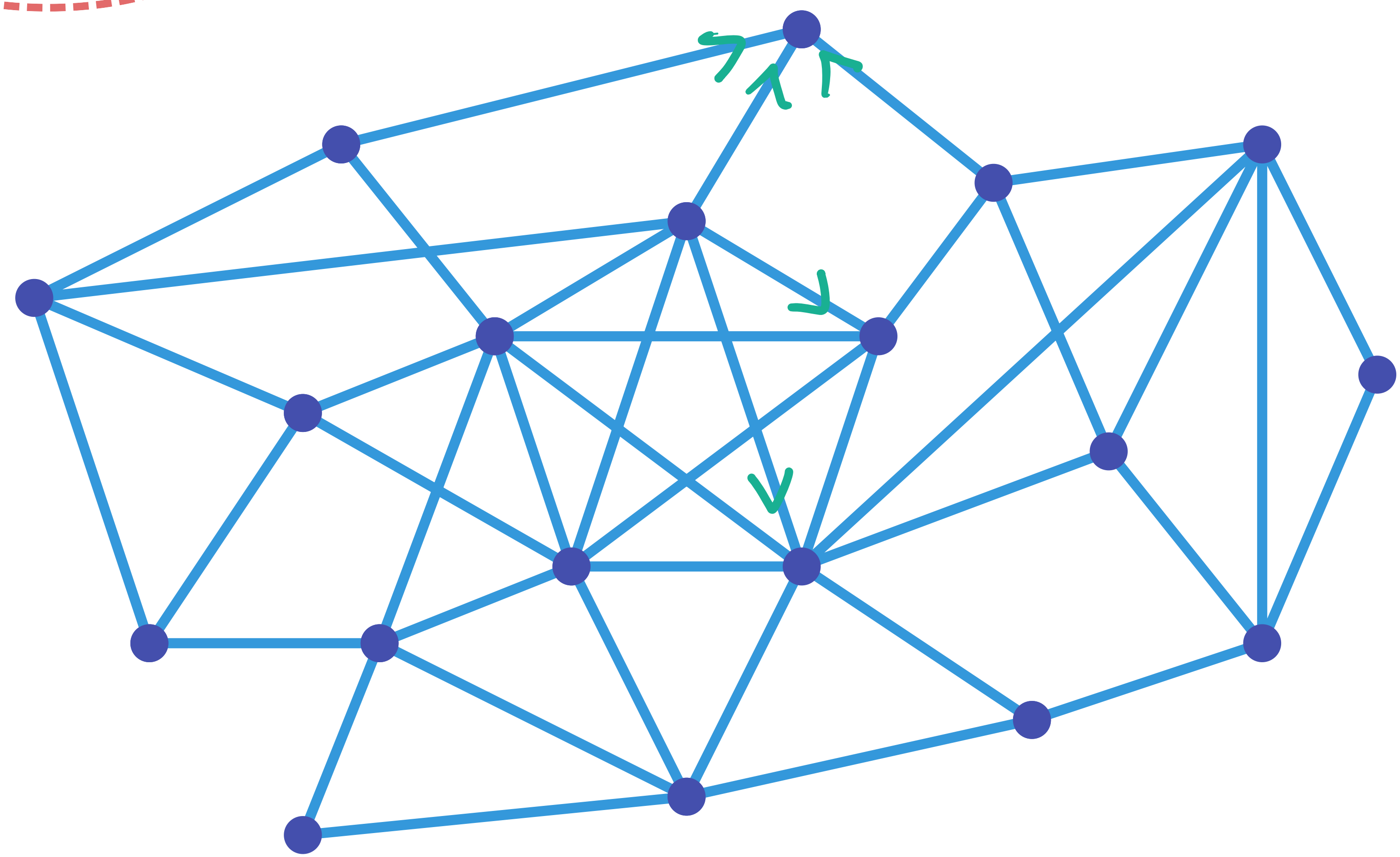
"Qualification matrix"

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

"1" = qualified "0" = not qualified

Edge Orientation

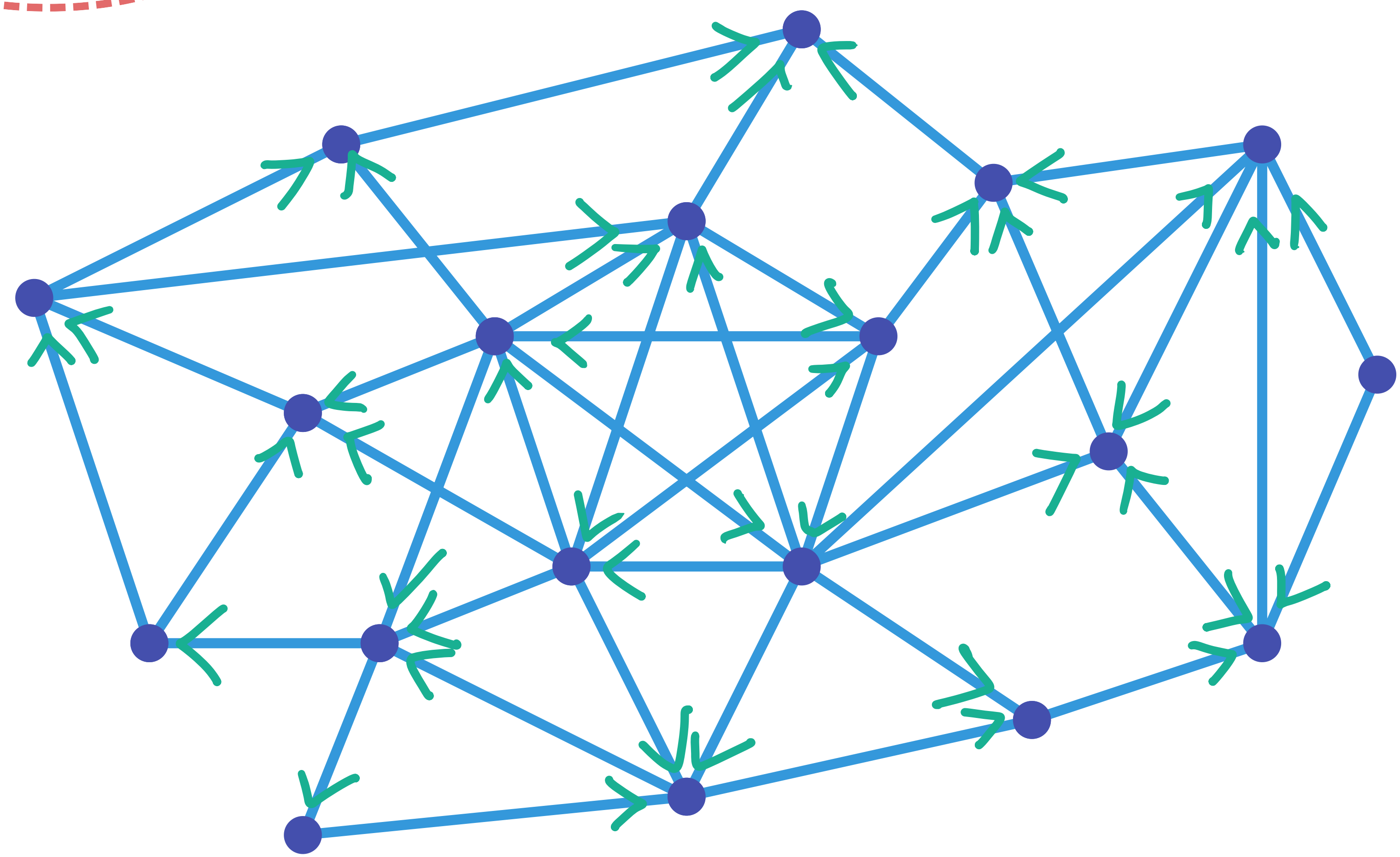
input: undirected graph



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

Edge Orientation

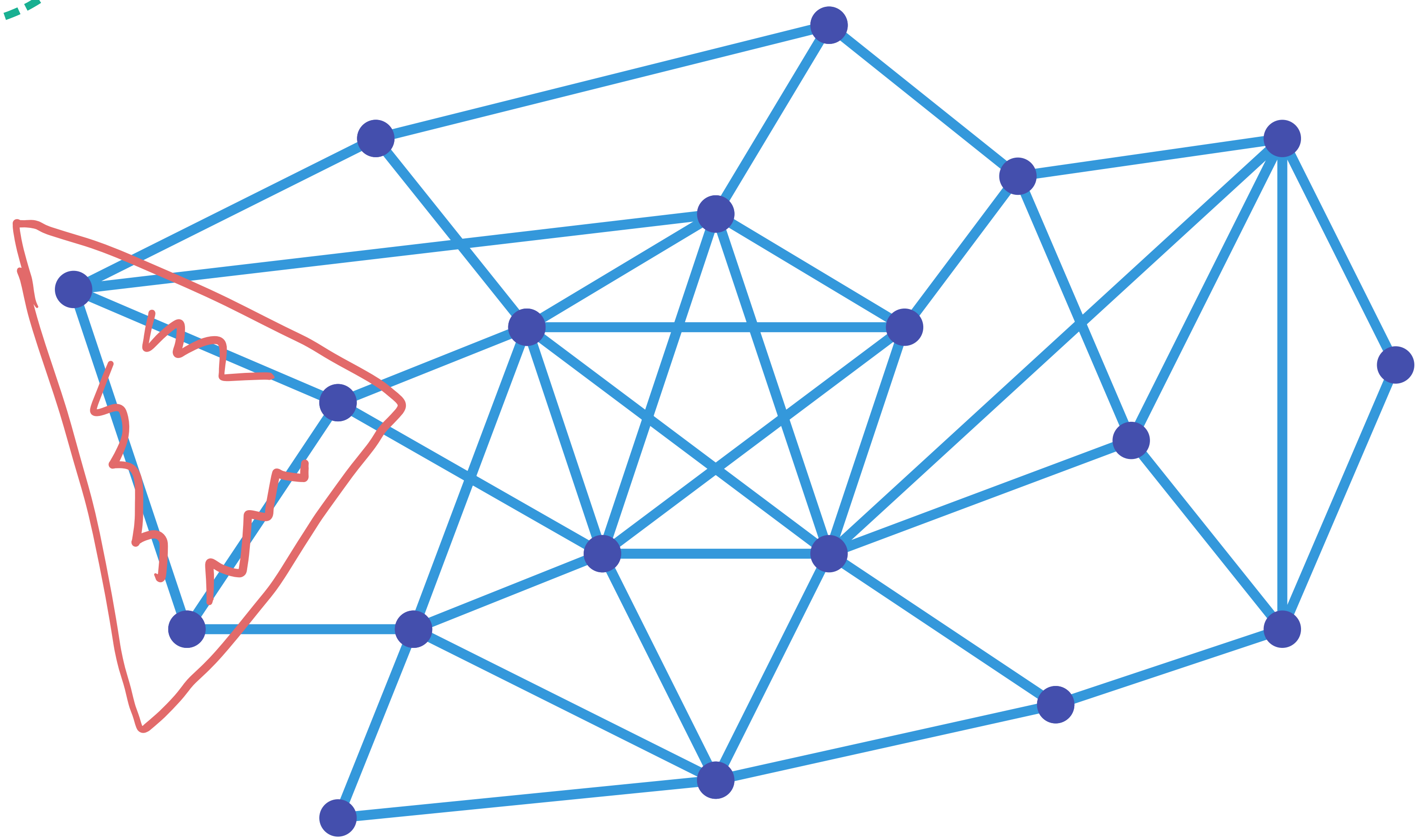
input: undirected graph



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

Densest Subgraph

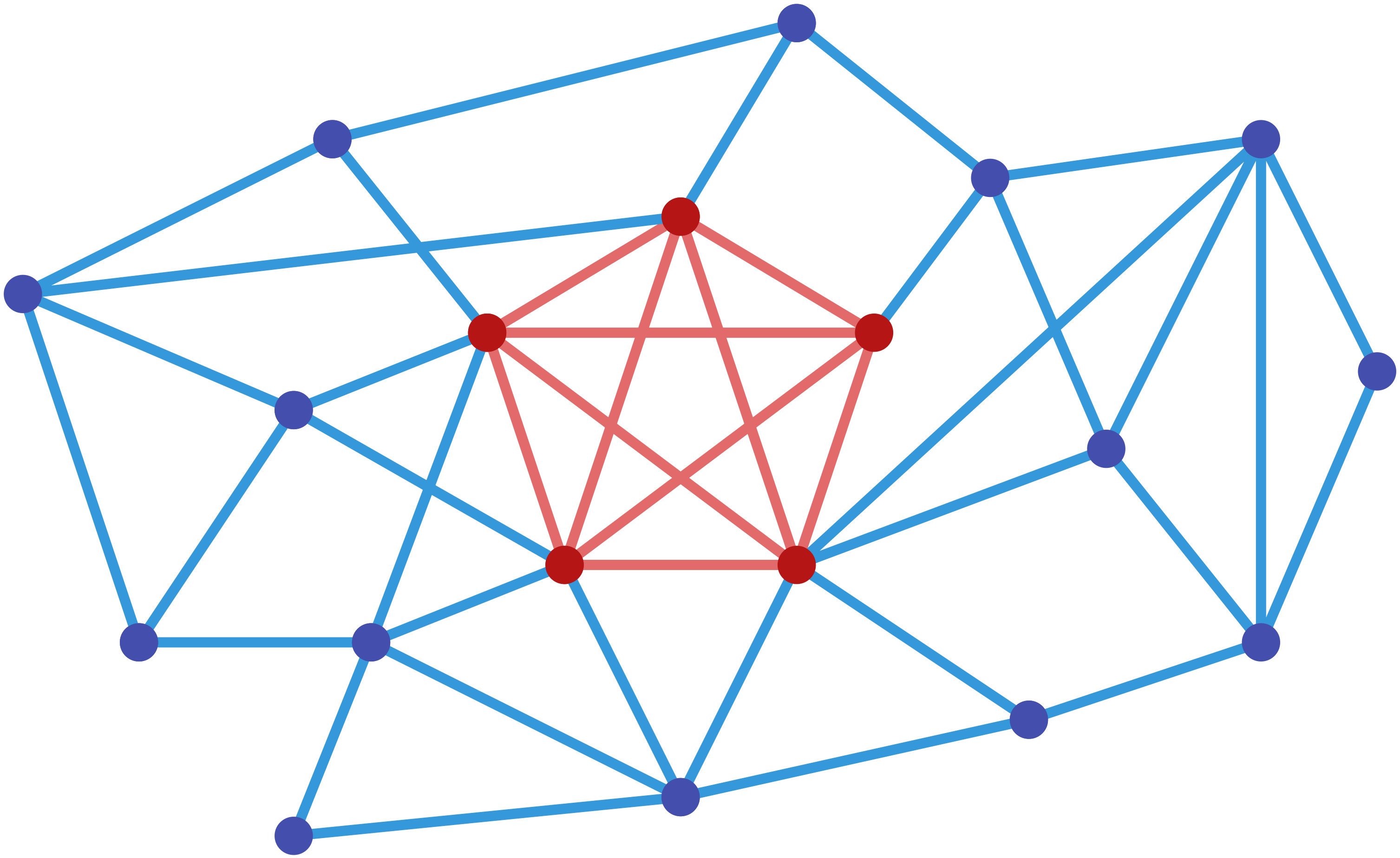
"density" = $\frac{\text{\# edges}}{\text{\# vertices}}$



Goal: compute densest subgraph

Densest Subgraph

"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

Bipartite Matching

Vertex Disjoint Paths

Project Selection

Path Covers

Edge Orientation

Assignment Problems

Densest Subgraph

Playoffs Elimination

Bipartite Vertex Cover

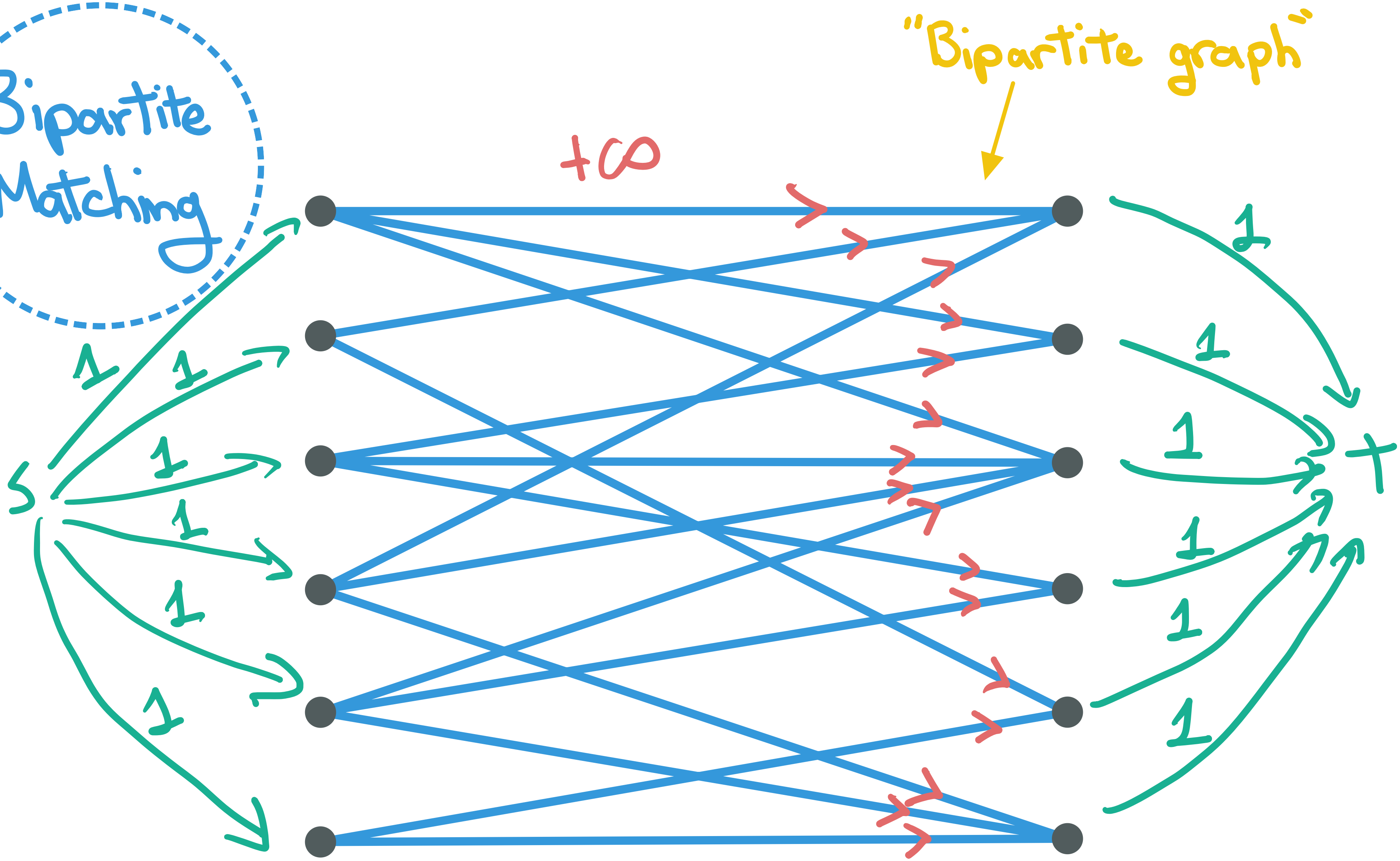
scheduling

"only two algorithms"*

① identify subproblems

② change the input

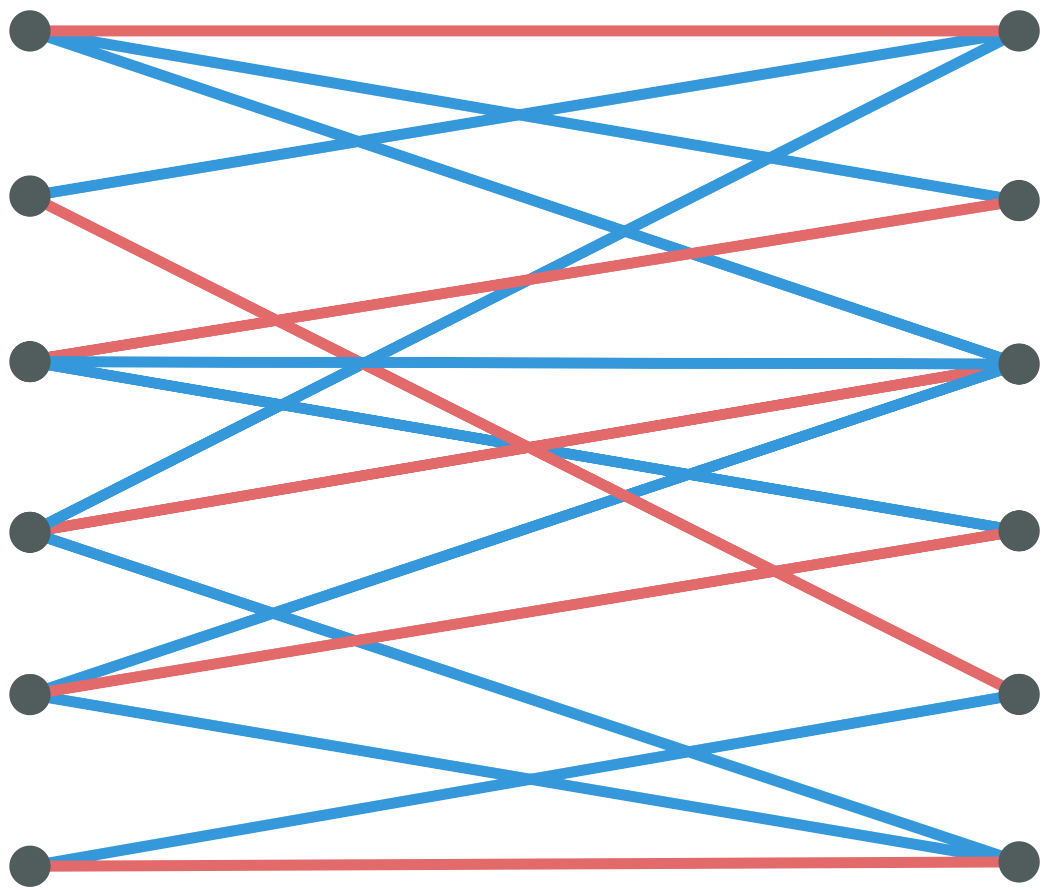
Bipartite Matching



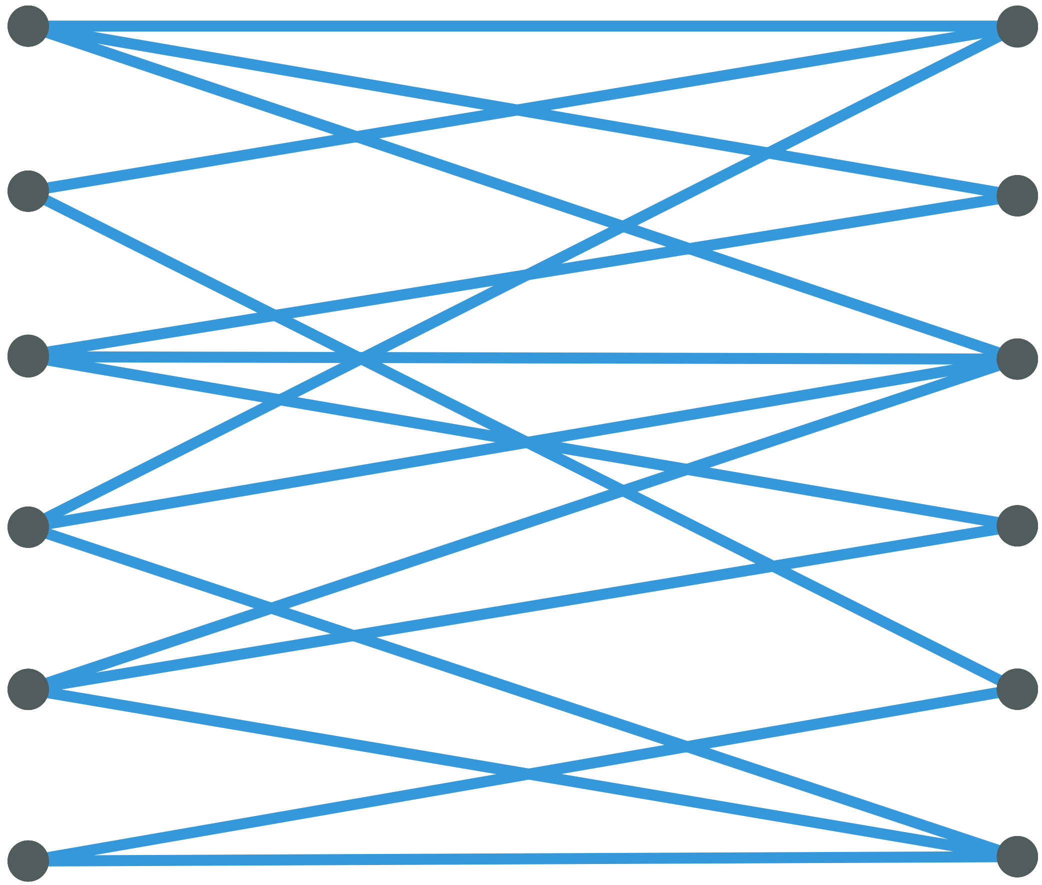
Goal: max $\underbrace{\text{set of edges w/ distinct endpoints}}_{\text{"matching"}}$

Bipartite Matching

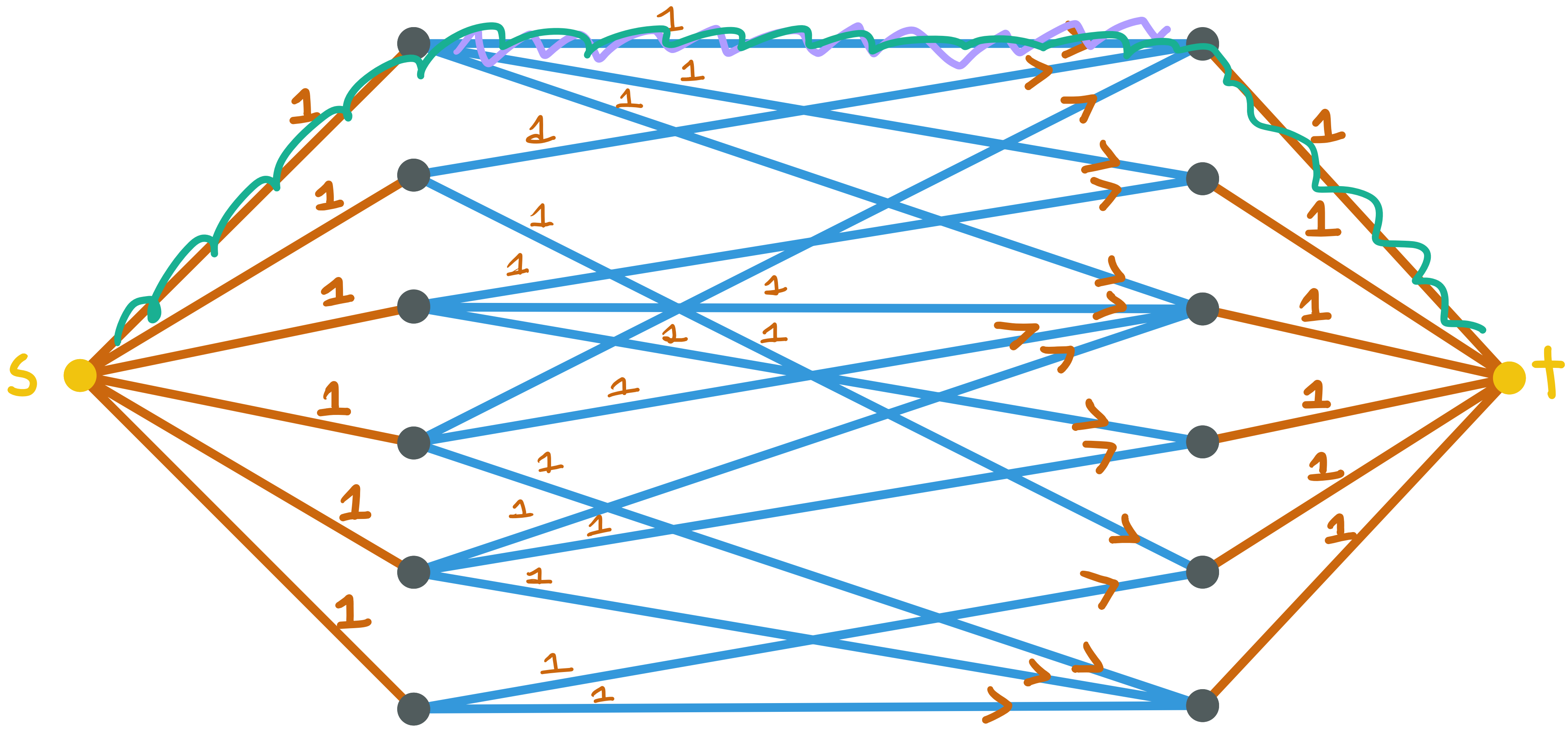
(bipartite graph)



Goal: max $\underbrace{\text{set of edges w/ distinct endpoints}}_{\text{"matching"}}$



(see lecture notes for detailed proof)

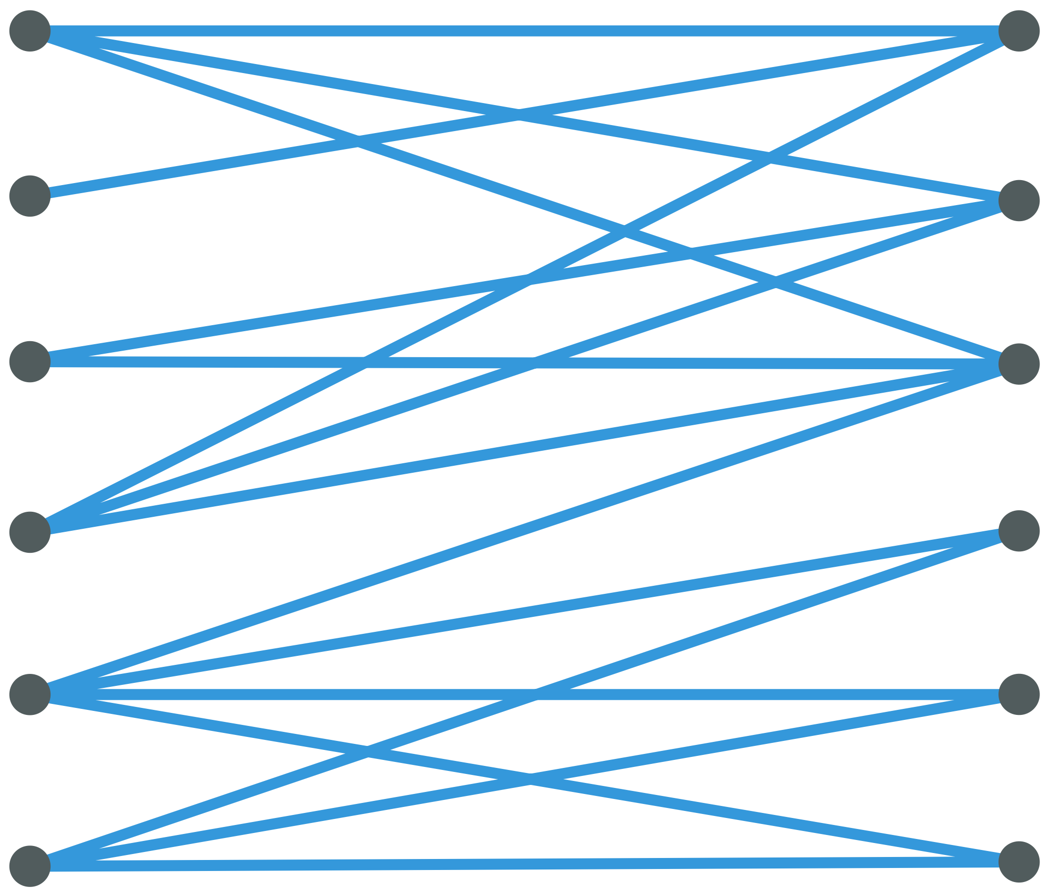


Matching w/ k edges ⟷ k edge-disjoint (s,t)-paths

(see lecture notes for detailed proof)

Bipartite Vertex Cover

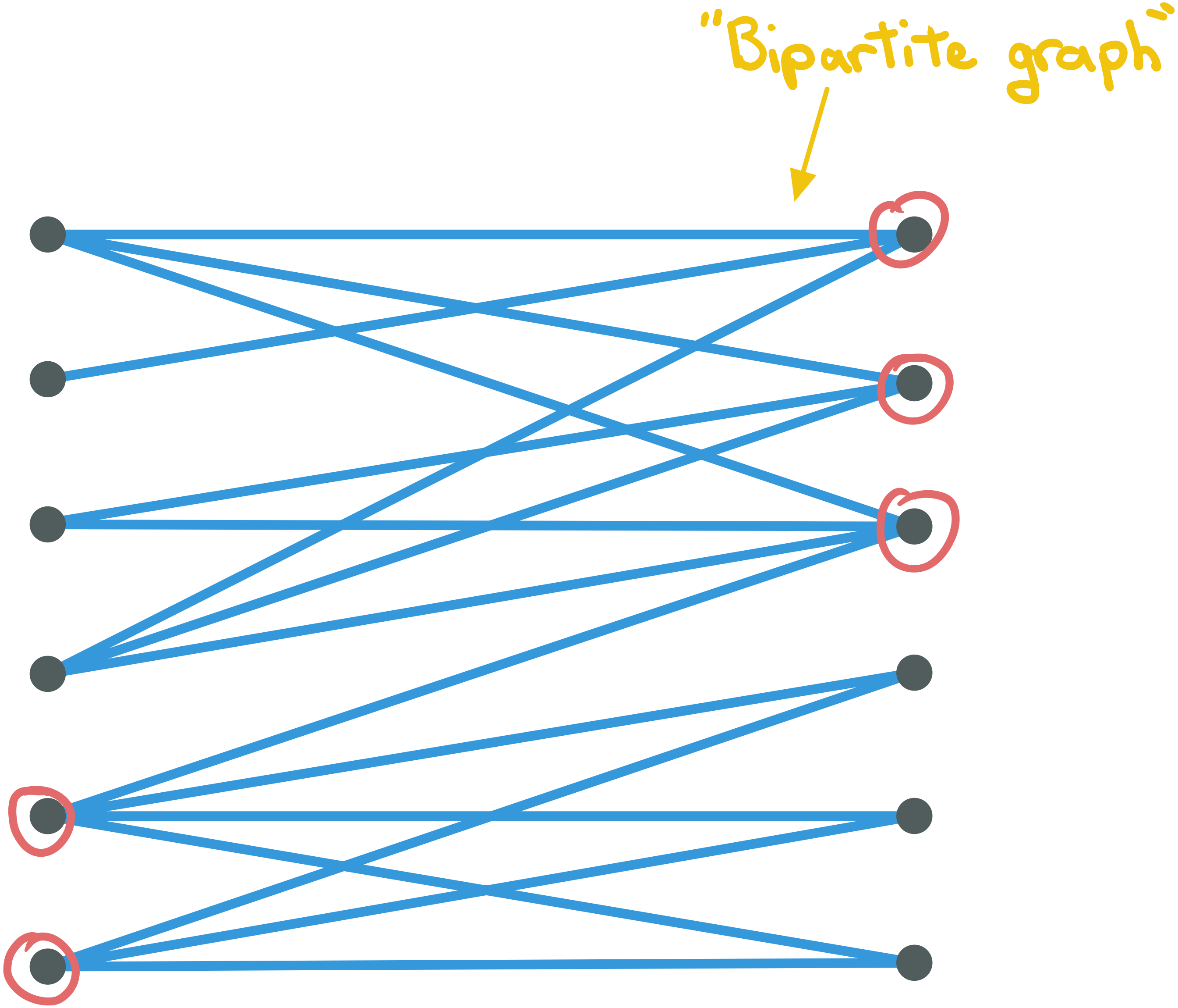
"Bipartite graph"



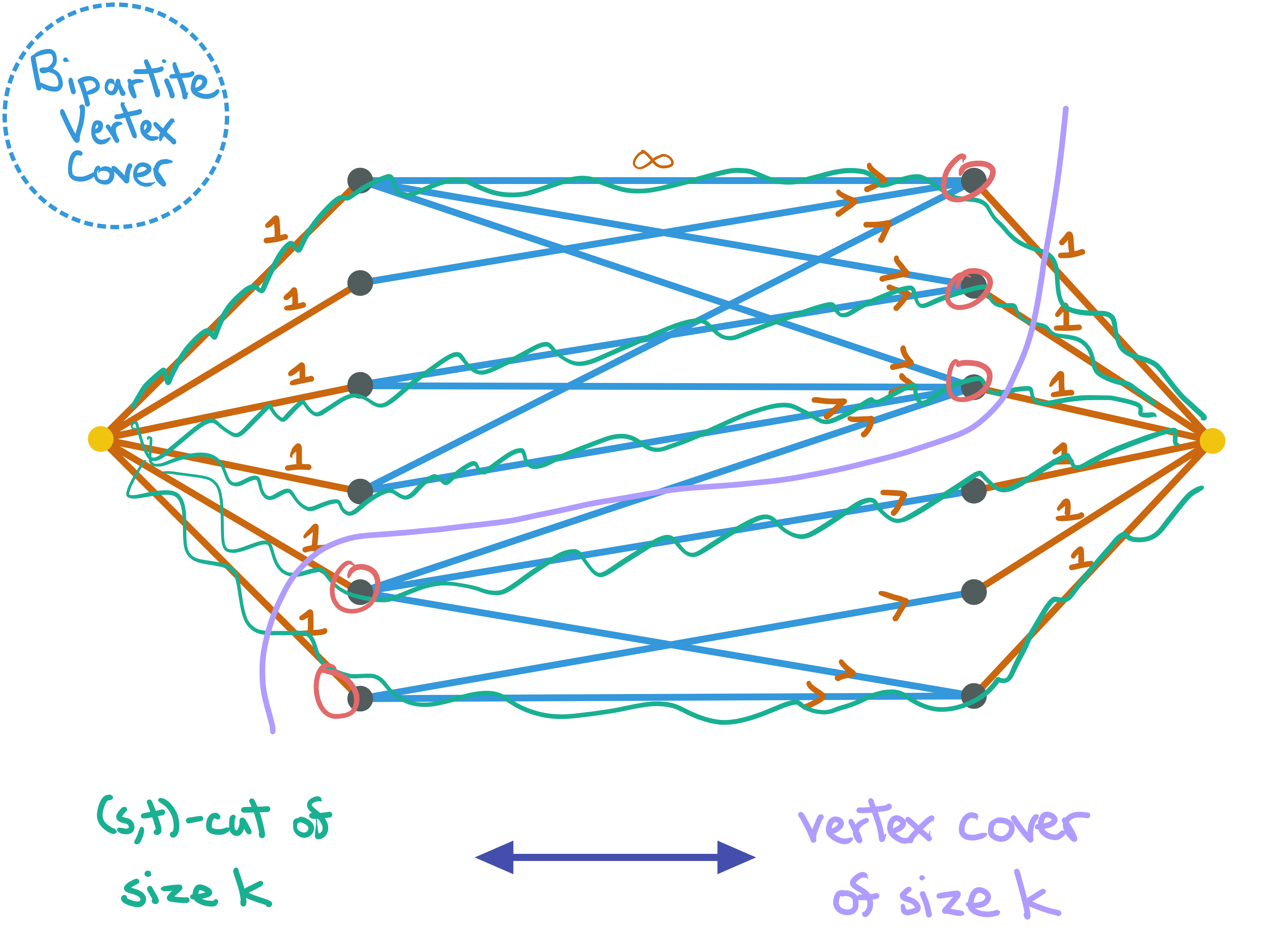
Goal: min set of vertices incident to all edges

"vertex cover"

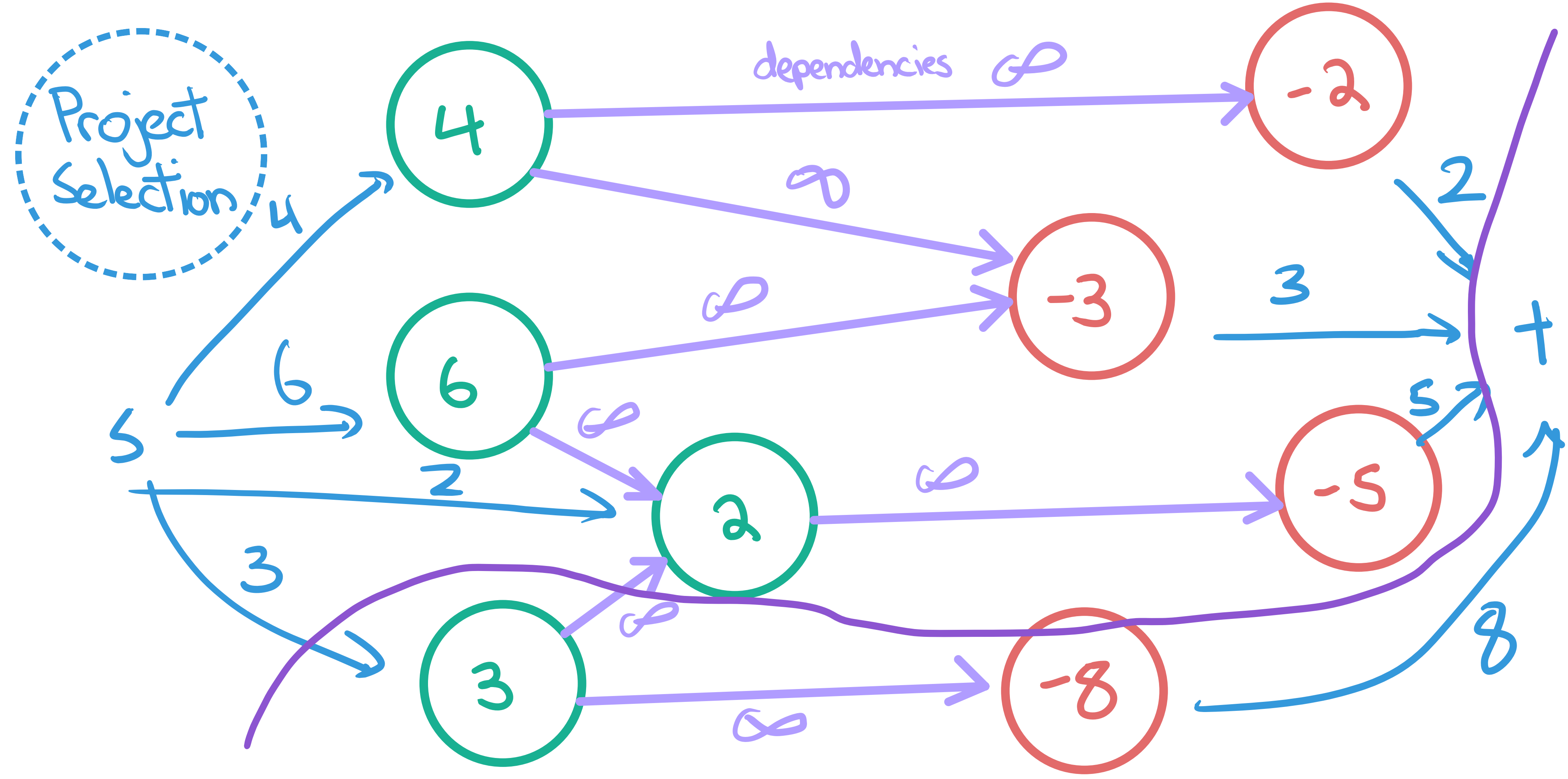
Bipartite Vertex Cover



Goal: min $\underbrace{\text{set of vertices incident to all edges}}_{\text{"vertex cover"}}$

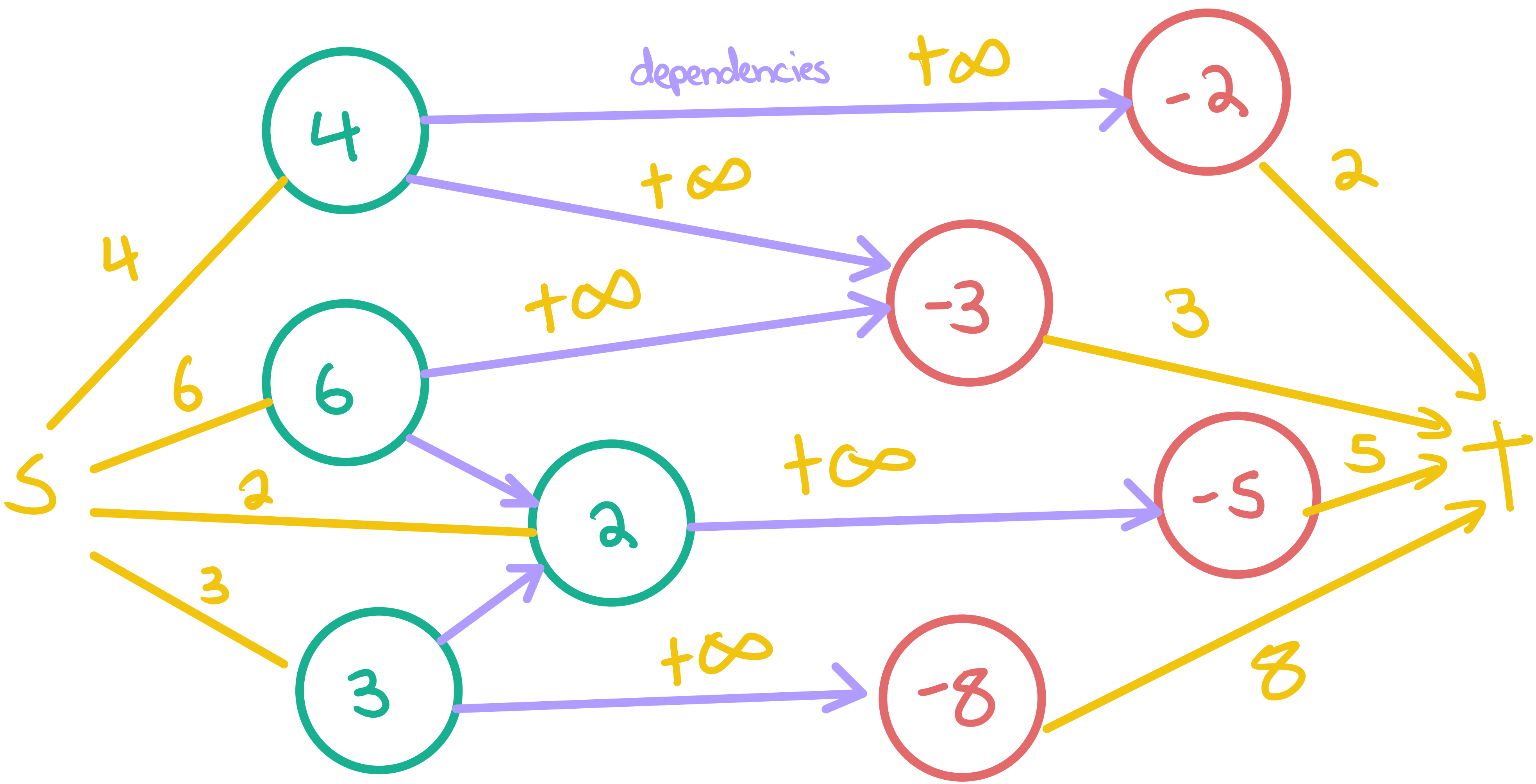


Bipartite Vertex Cover
∞
1
(s,t)-cut of size k
vertex cover of size k

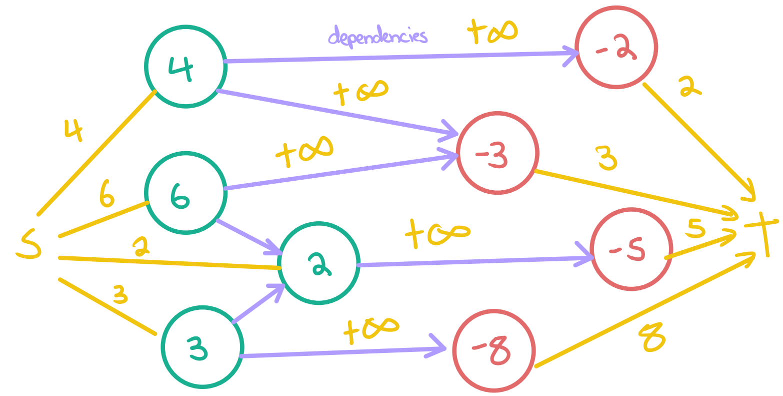


Projects have net profits (can be <0 for costs) and dependencies

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



claim: min-cut ⇒ best project selection



dependencies
s
t
4
6
2
3
-2
-3
-5
-8
+∞
+∞
+∞
+∞
+∞
4
6
2
3
2
3
5
8

Playoffs Elimination

Playoffs 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) 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	

(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	×

New York vs Baltimore

New York vs Toronto

Baltimore vs Toronto

Baltimore vs Boston

Toronto vs Boston

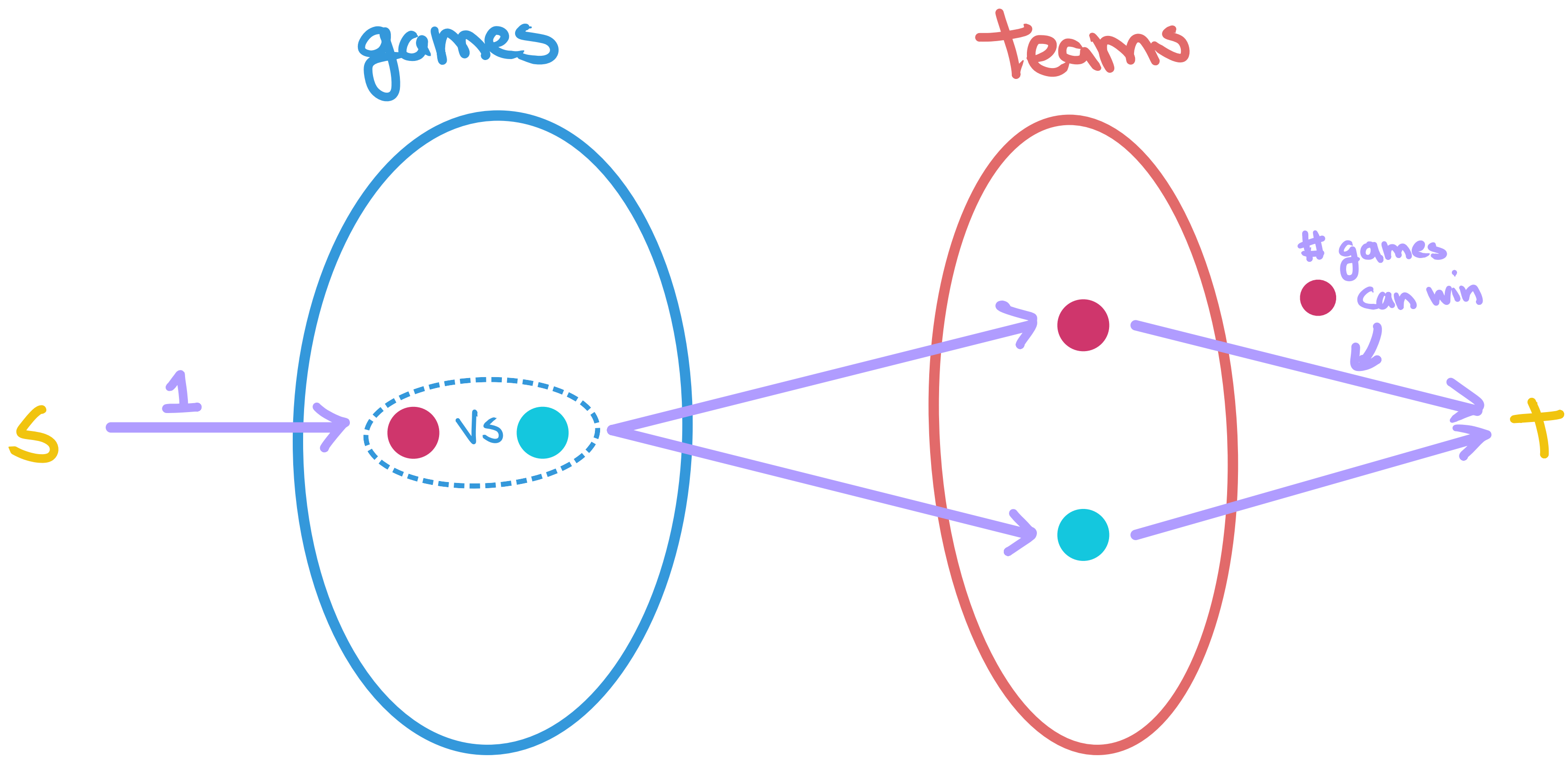
New York

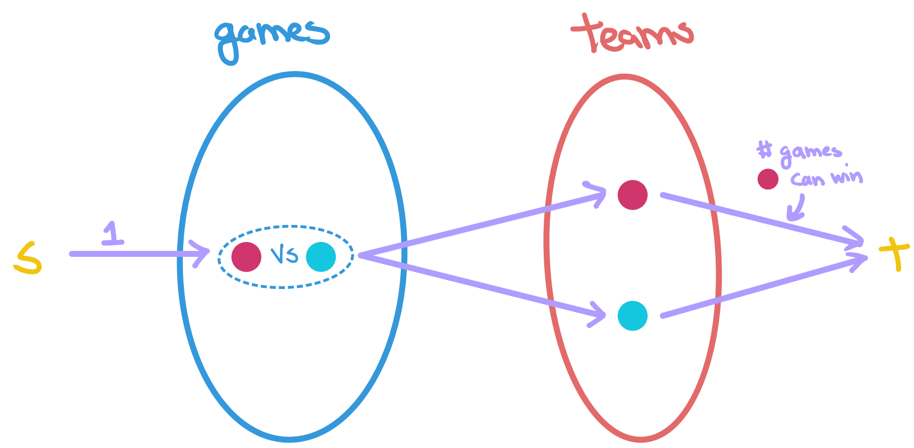
Baltimore

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	×

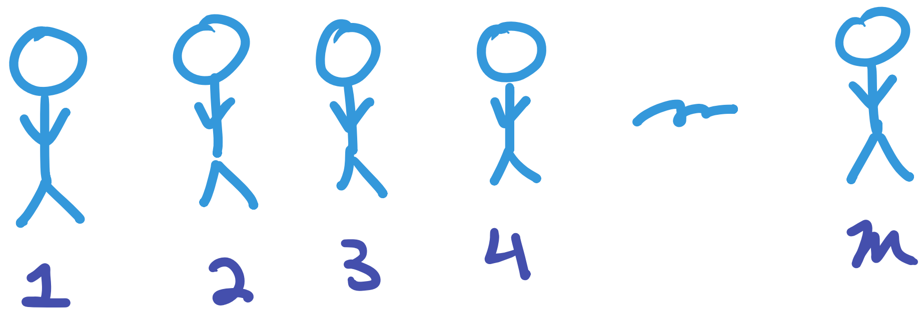

games
teams
s
1
vs
games
can win
t


games
teams
s
1
vs
games can win
t

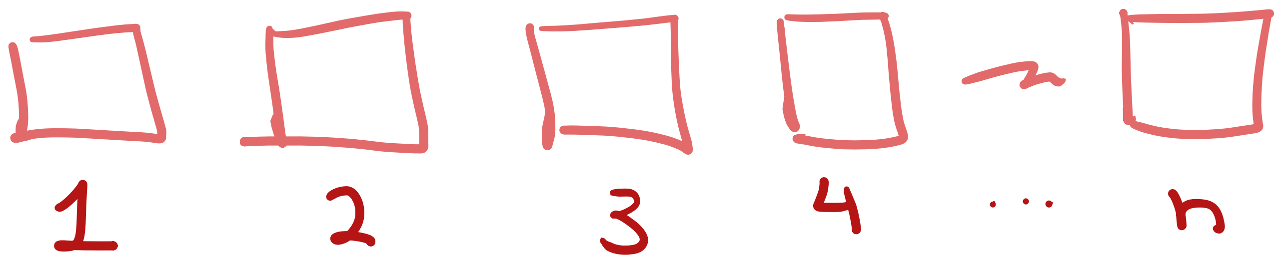

s
1
1
1
1
1
New York vs Baltimore
New York vs Toronto
Baltimore vs Toronto
Baltimore vs Boston
Toronto vs Boston
New York
Baltimore
Toronto
Boston
0
1
1
∞
t

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"

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

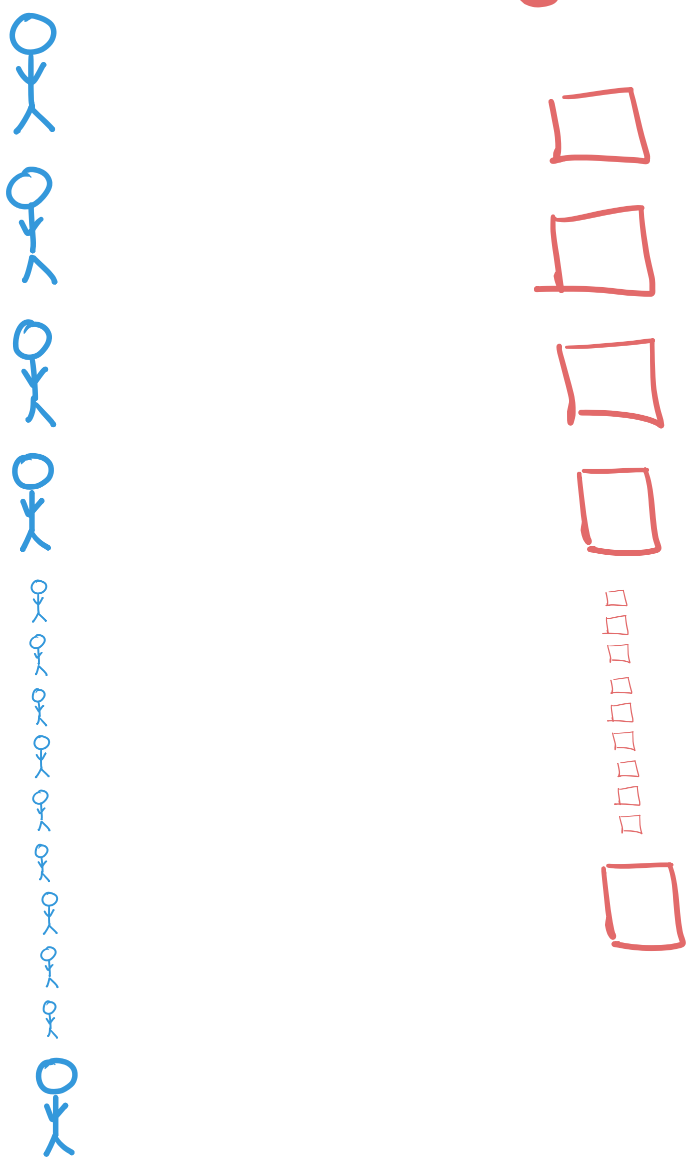
"1" = qualified

"0" = not qualified

Assignment Problems

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

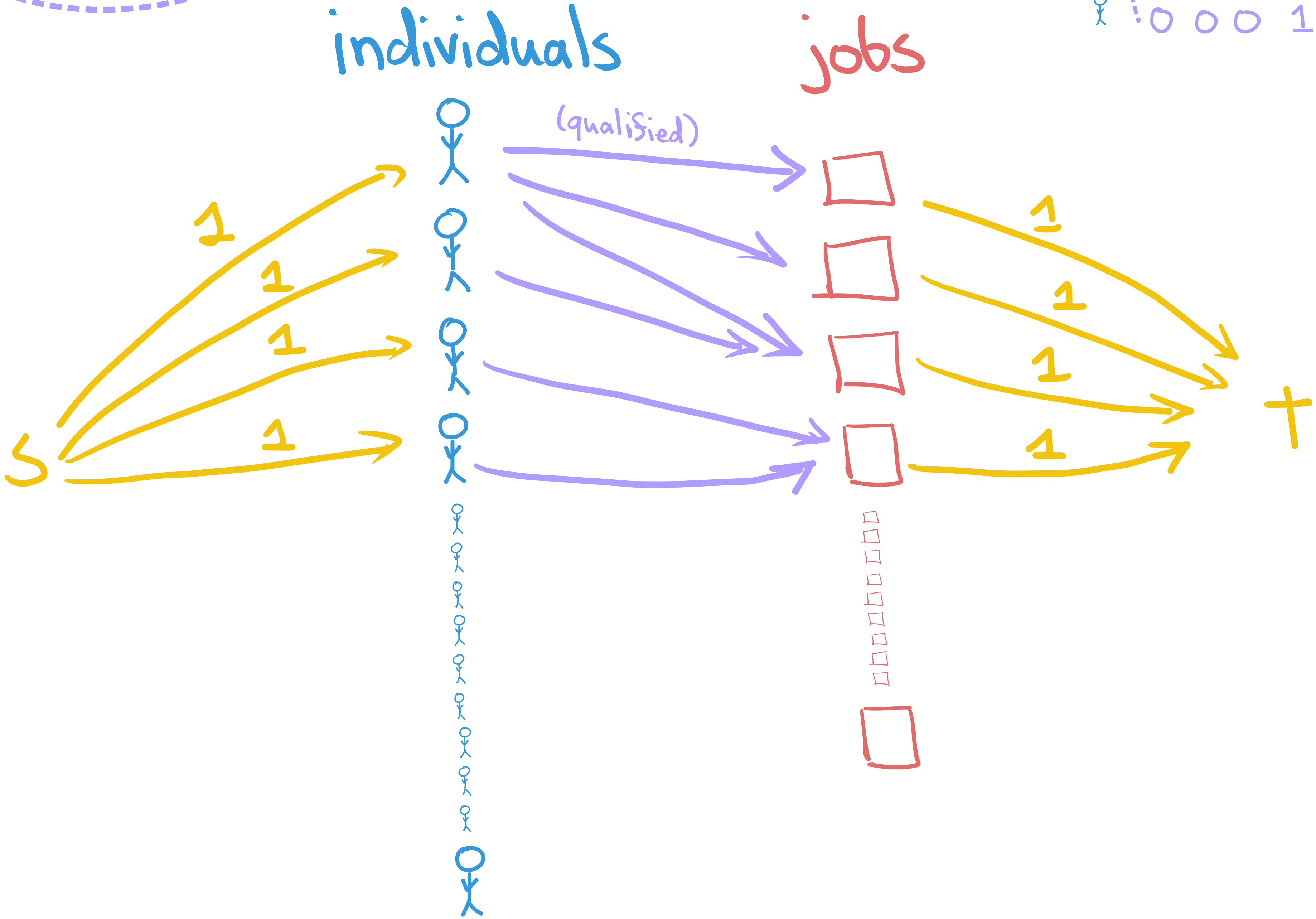
individuals jobs



S

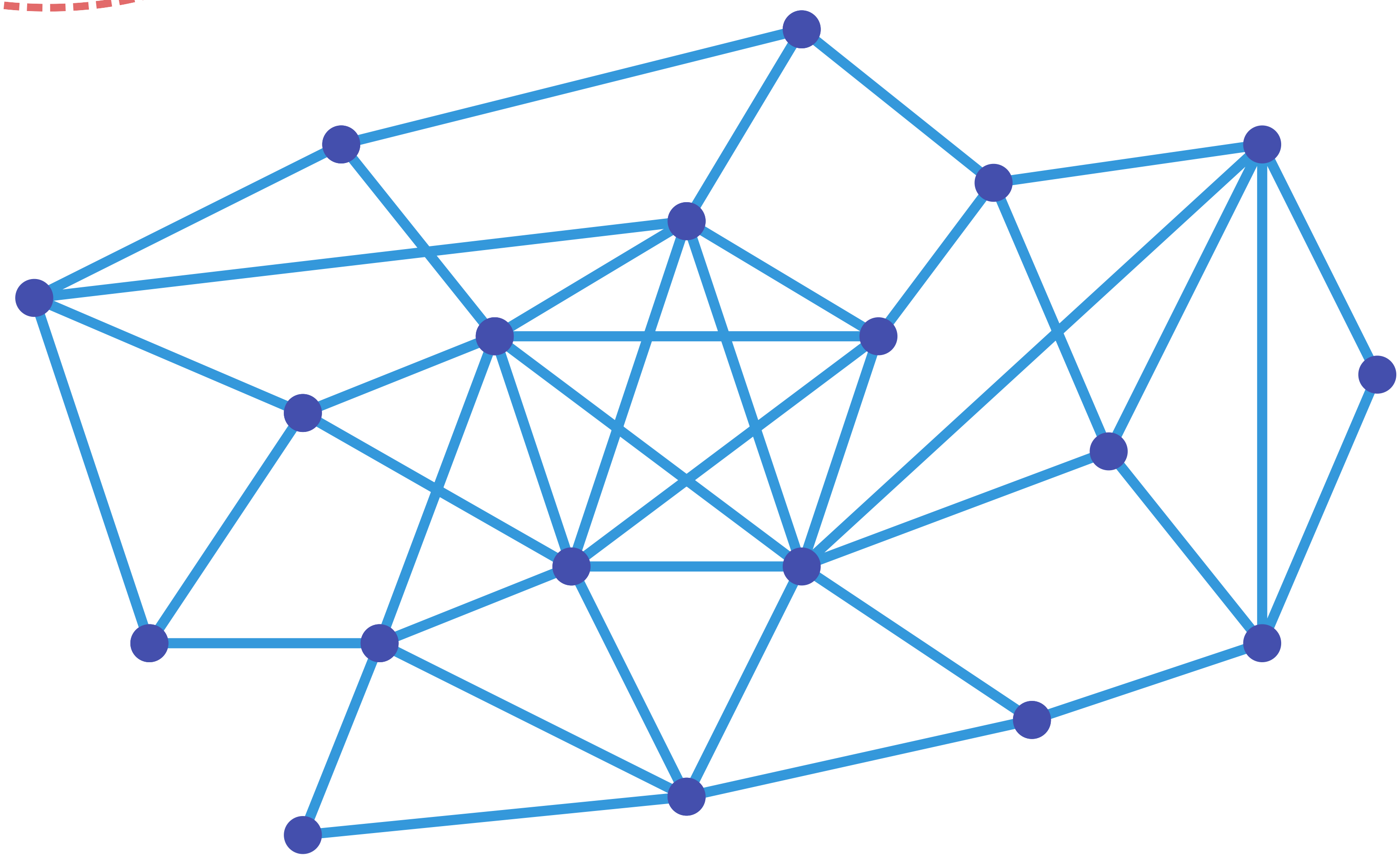
Assignment Problems

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Edge Orientation

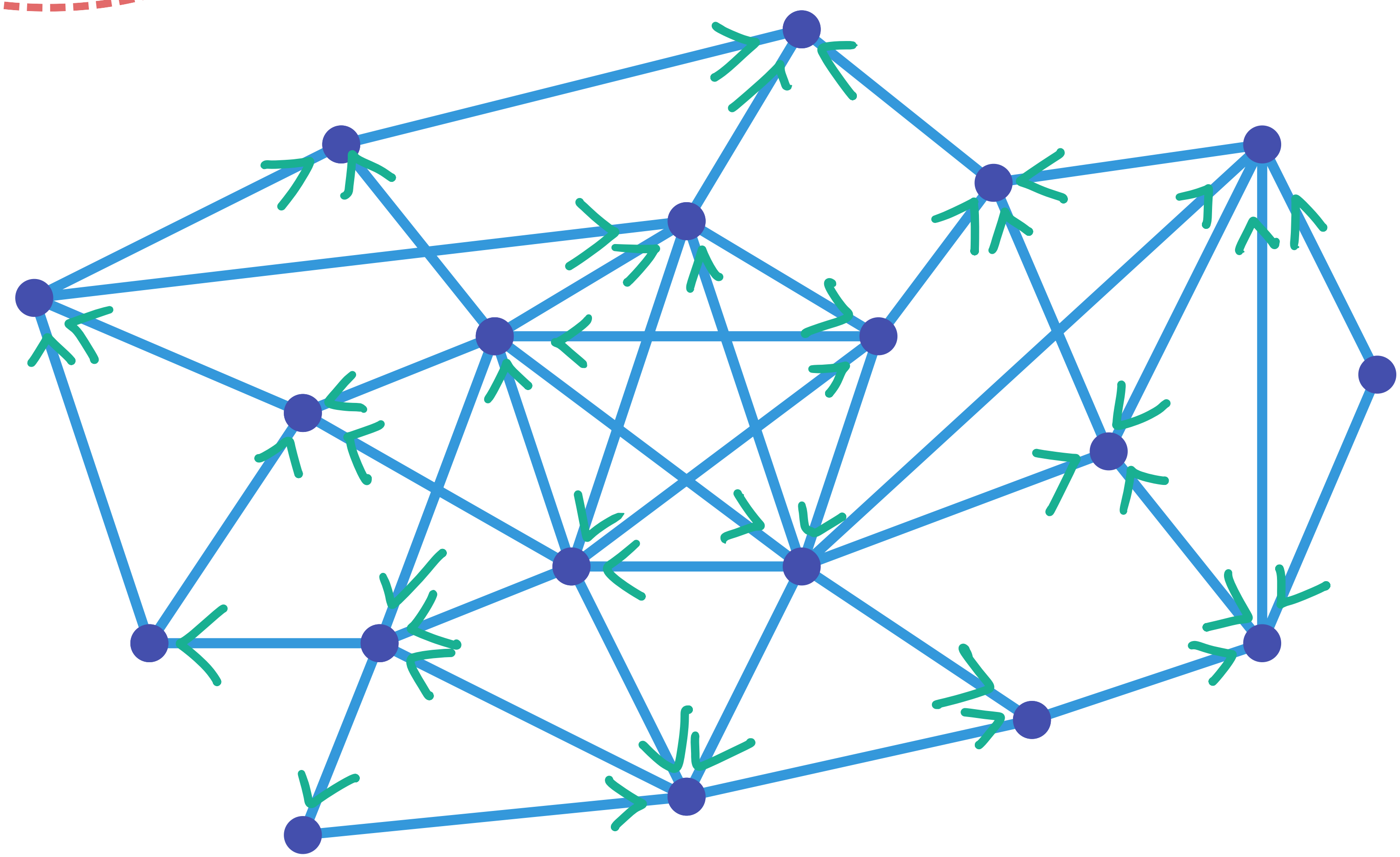
input: undirected graph



Goal: direct the edges to minimize the max density

Edge Orientation

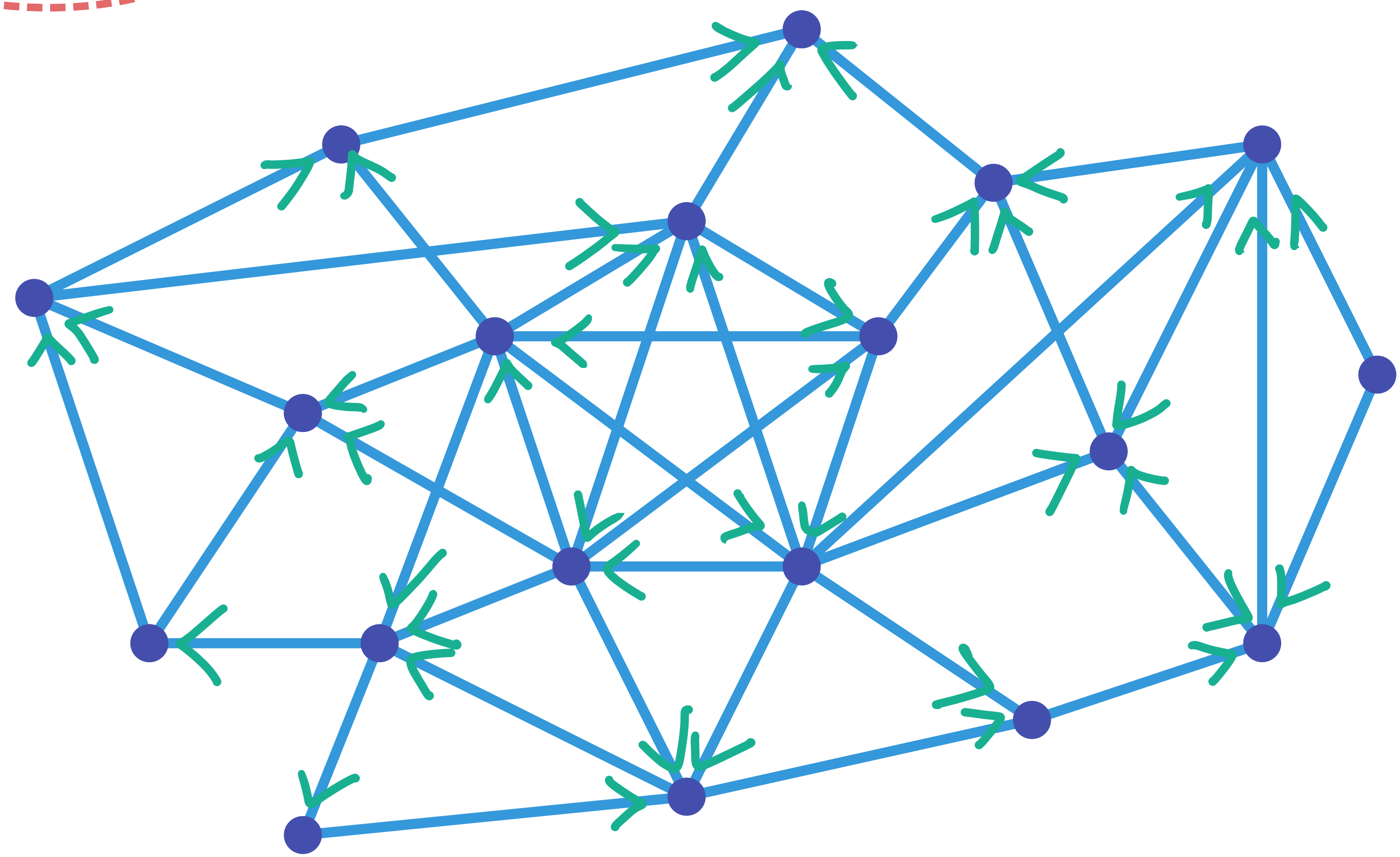
input: undirected graph



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

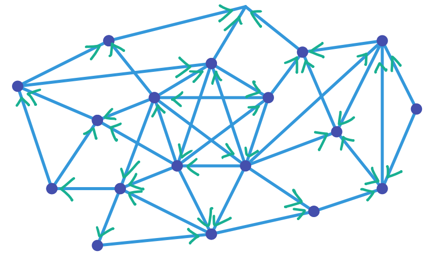
Edge Orientation

input: undirected graph



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

Edge Orientation

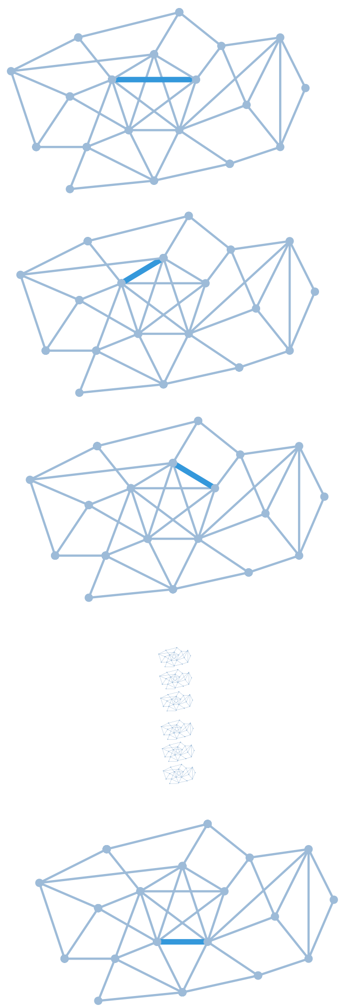


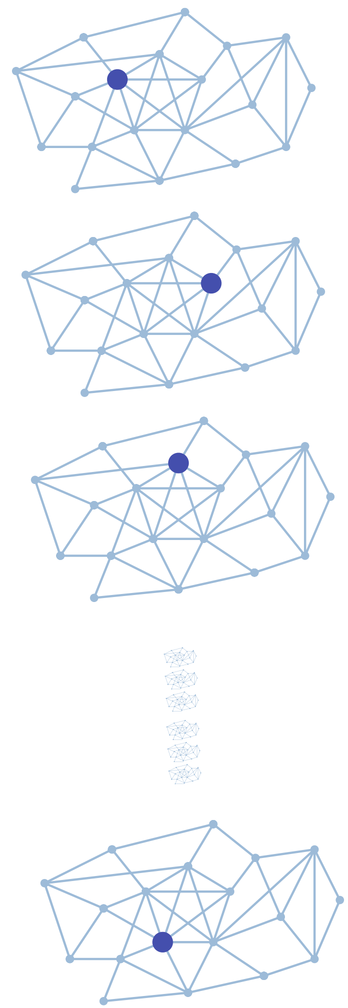
edges

vertices

h = target in-degree

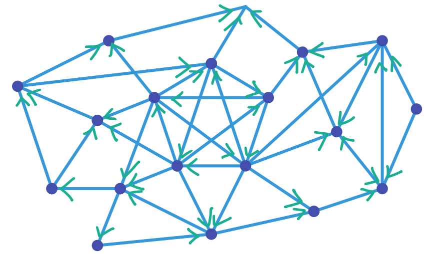
s

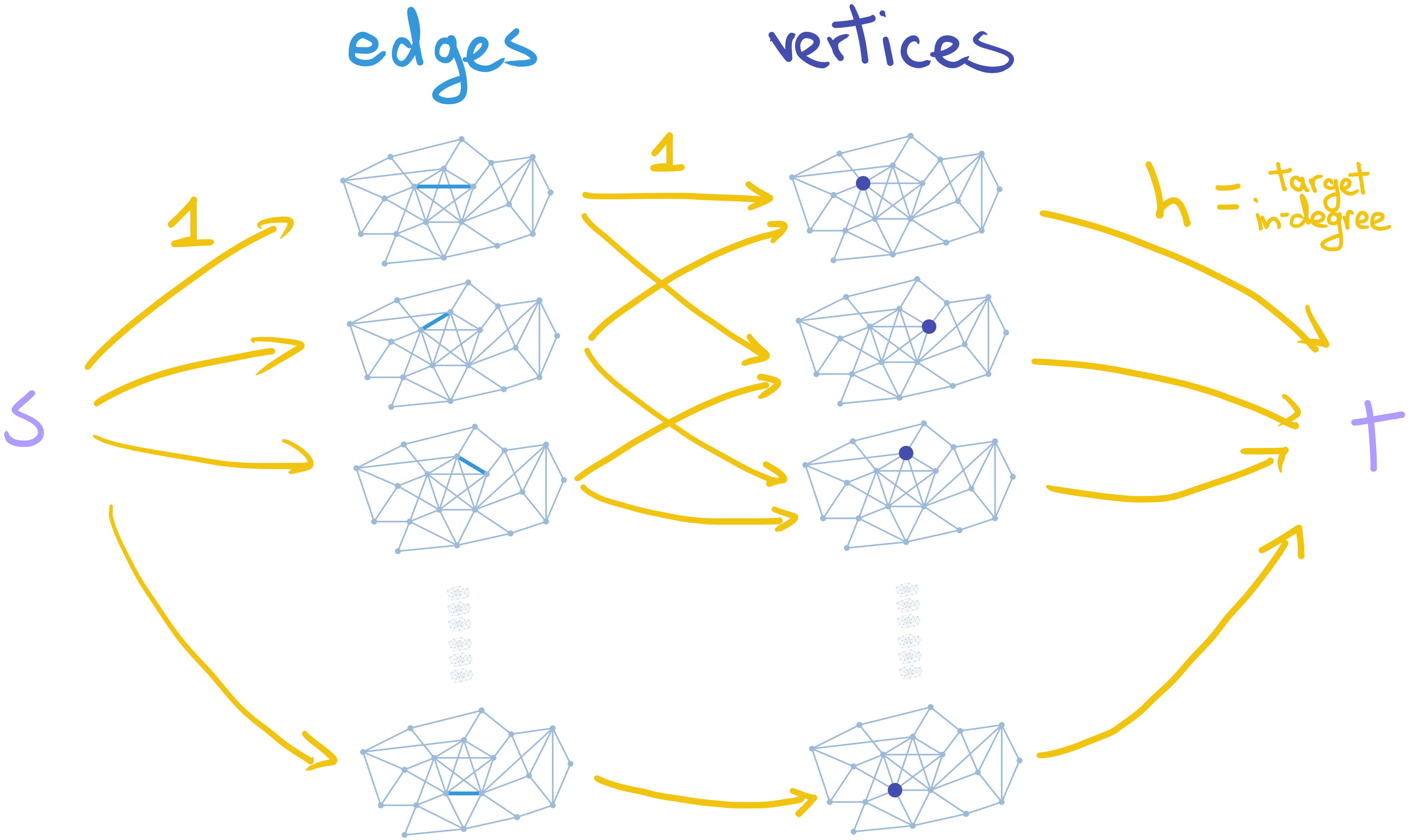




t

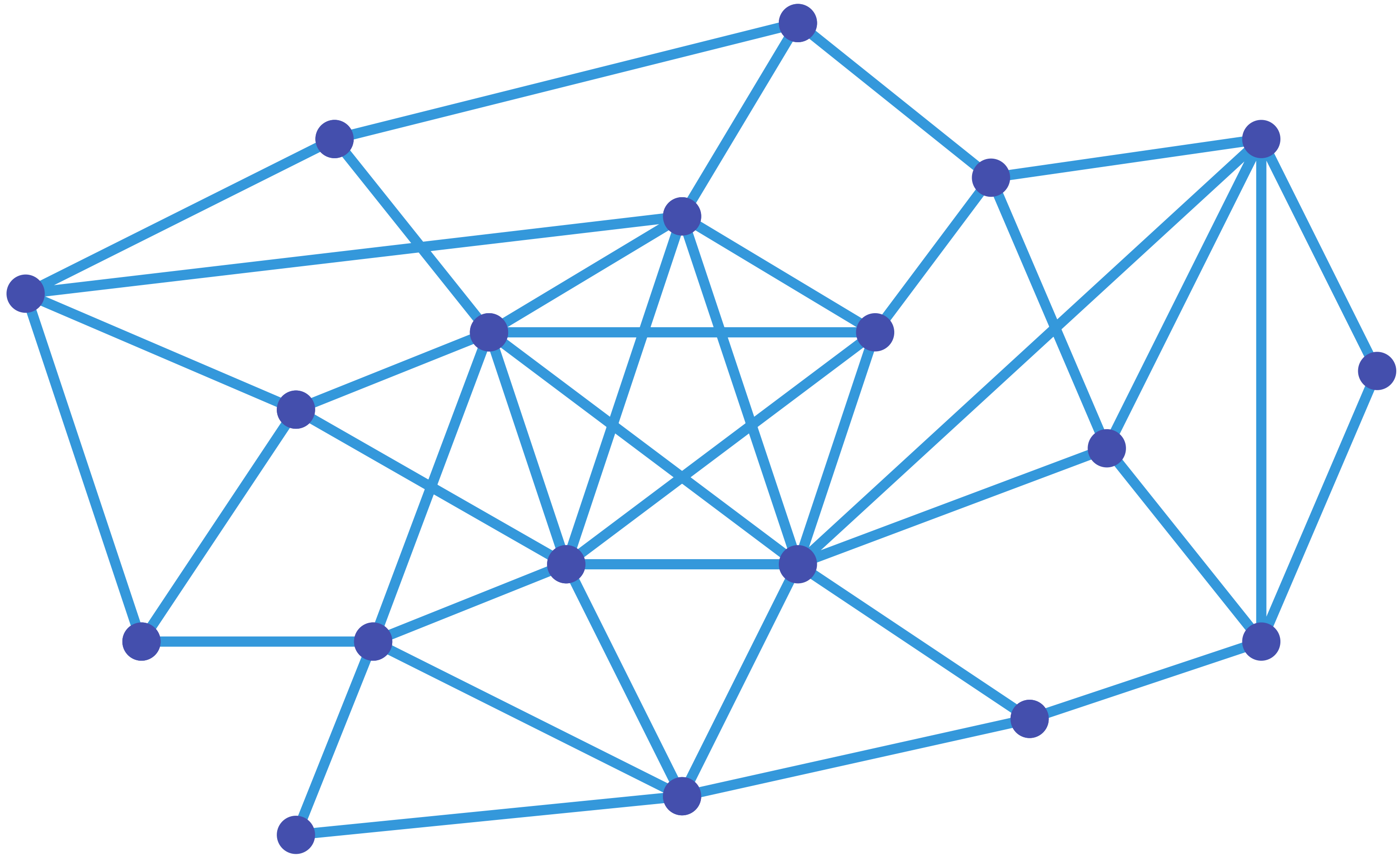
Edge Orientation





Densest Subgraph

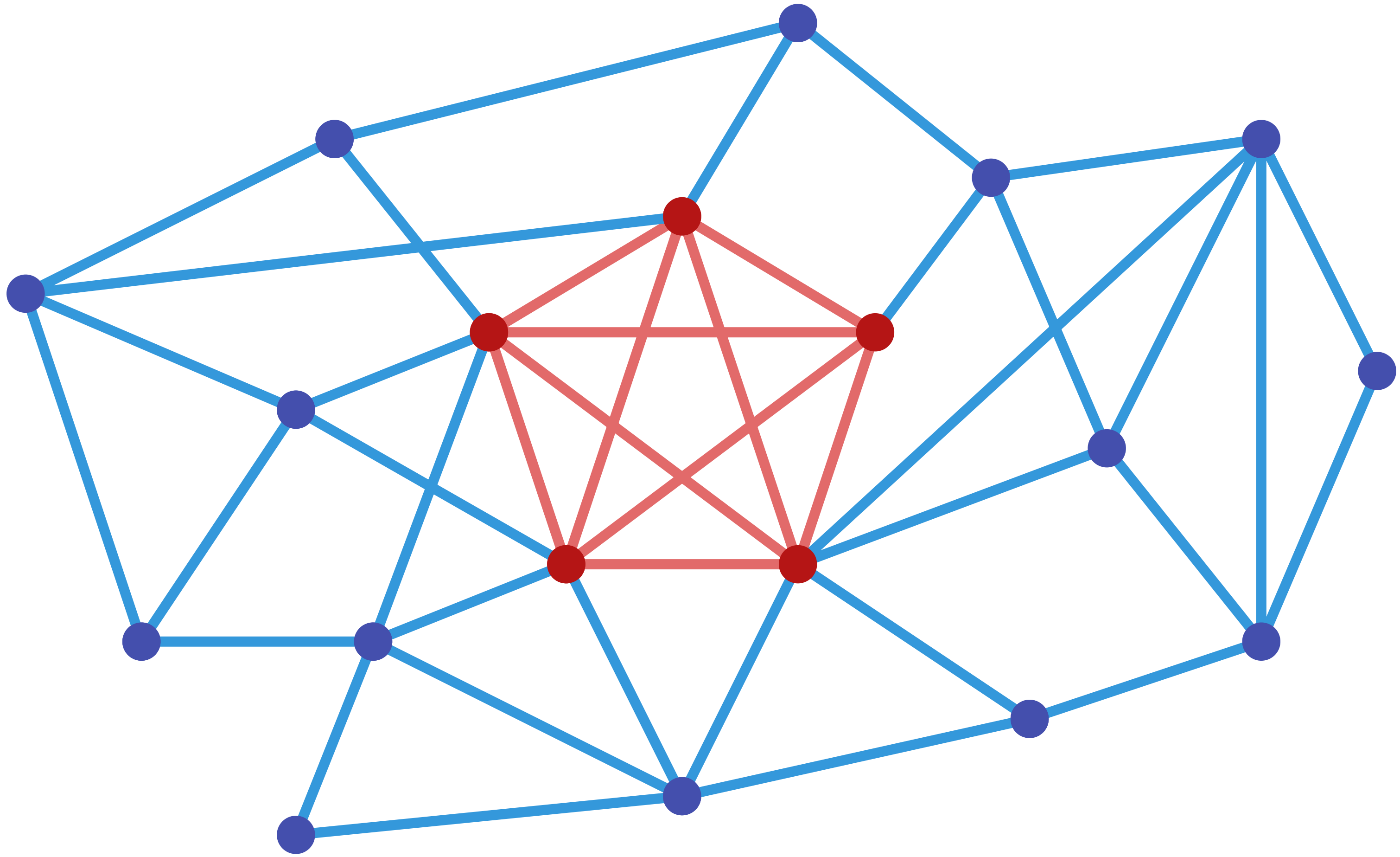
"density" = $\dfrac{\text{\# edges}}{\text{\# vertices}}$



Goal: compute densest subgraph

Densest Subgraph

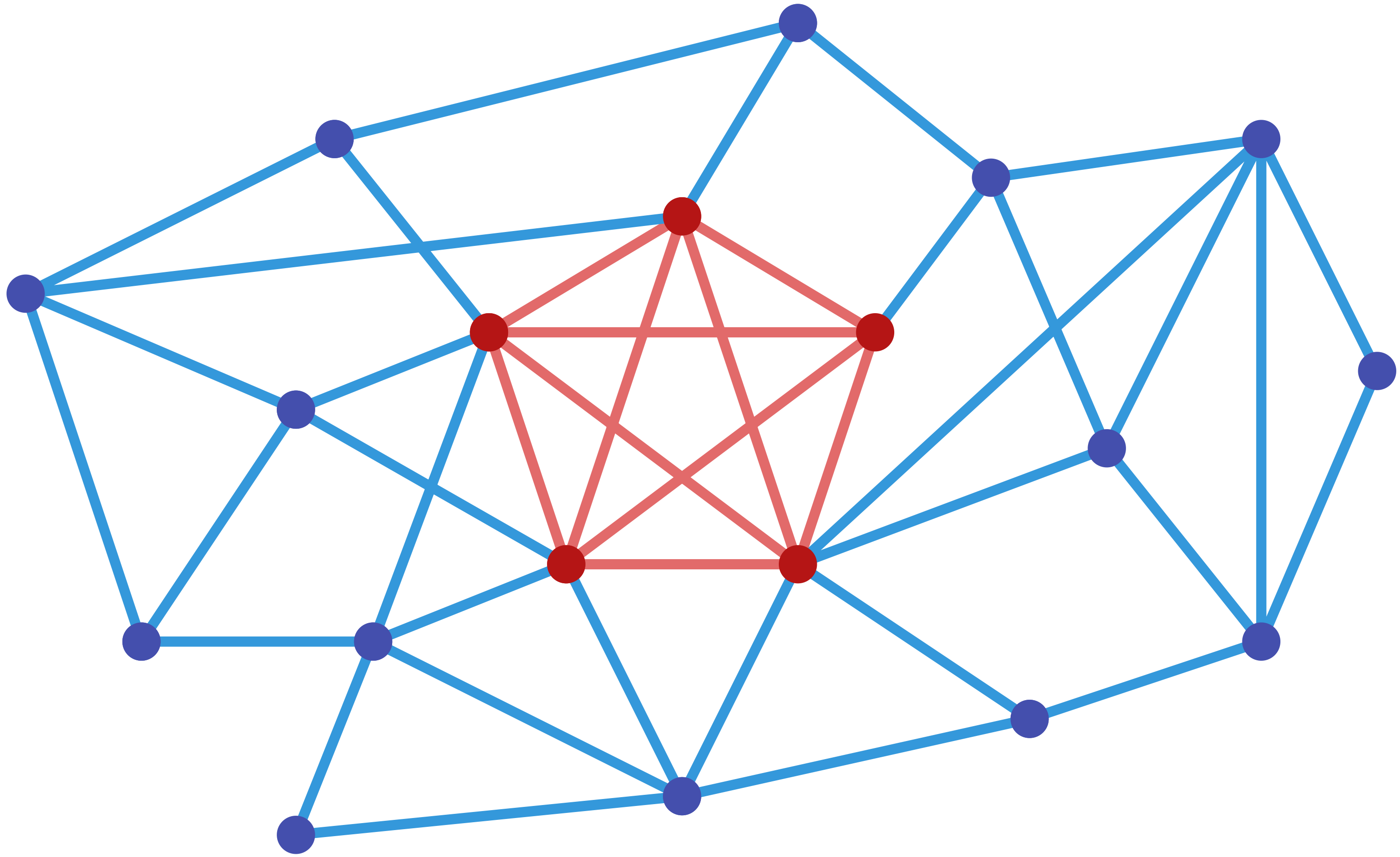
"density" $= \dfrac{\#\ \text{edges}}{\#\ \text{vertices}}$



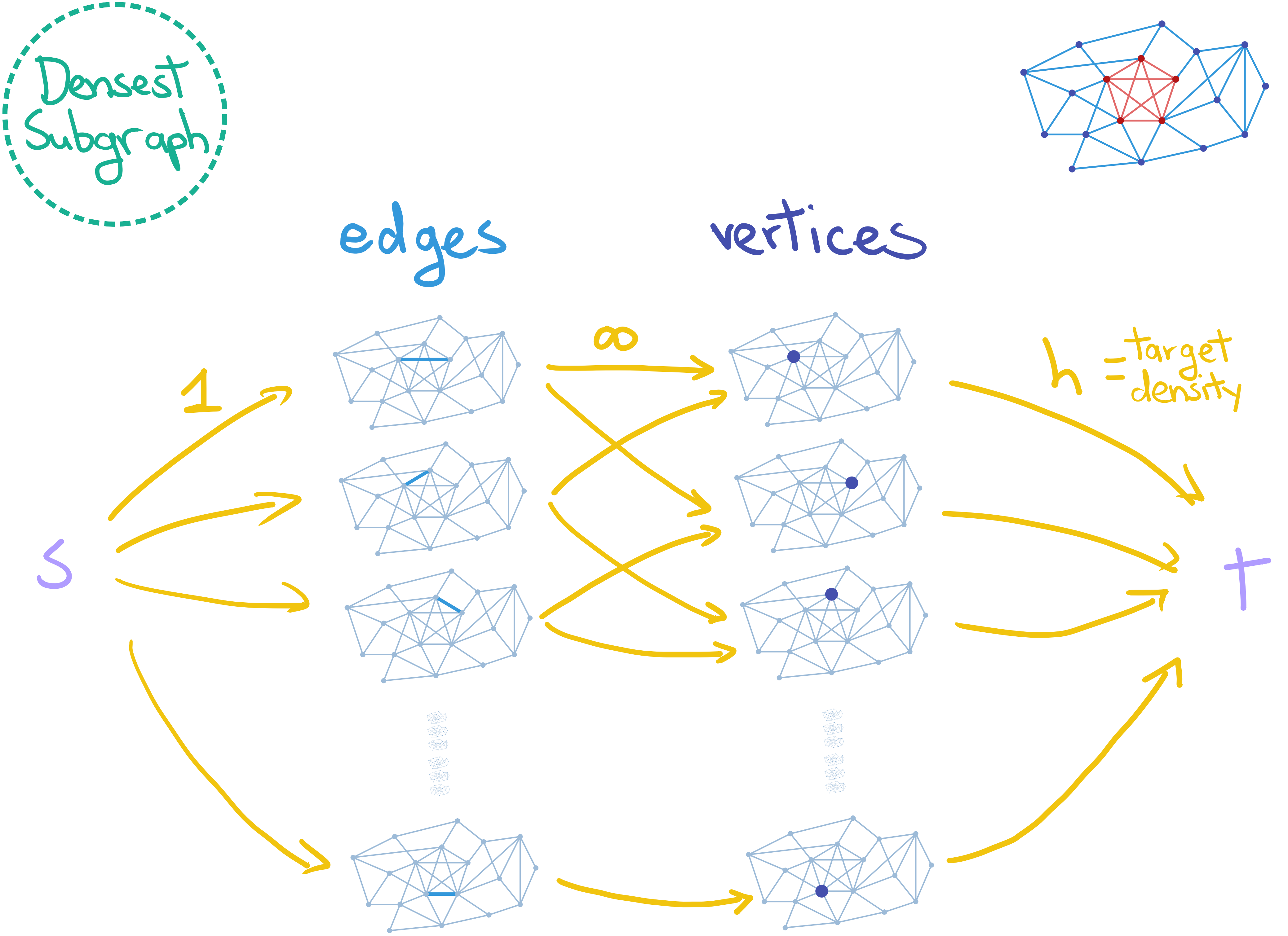
Goal: compute densest subgraph

Densest Subgraph

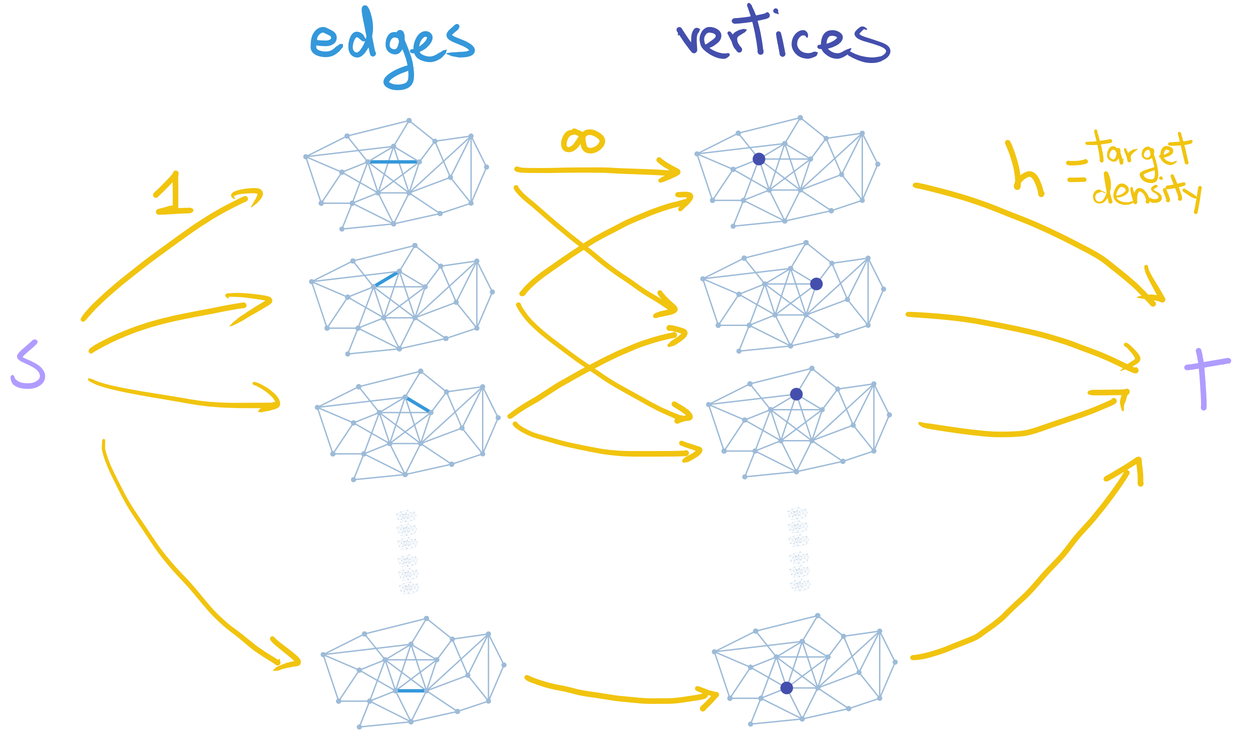
"density" = $\frac{\#\text{ edges}}{\#\text{ vertices}}$

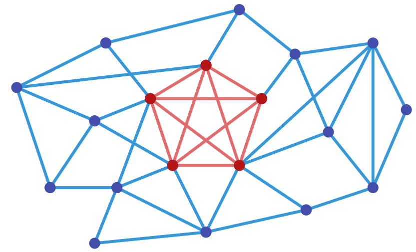


Decision version: Given $h>0$, is there a subgraph w/ density $>h$?


Densest Subgraph
edges
vertices
∞
1
h = target density
s
t

Densest Subgraph

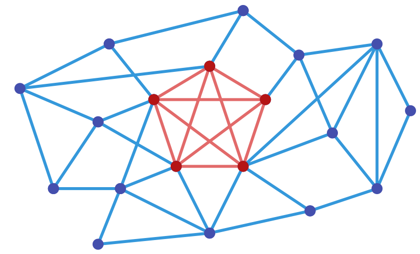


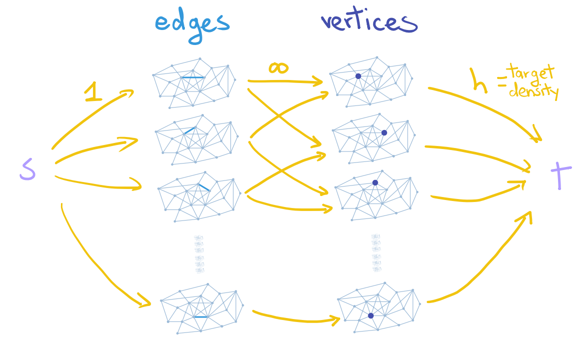


claim: if max-flow $= m$, then all subgraphs have density $\leq h$

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

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





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