Applications of Flows + Cuts

Applications of Flows + Cuts

(high-level comment)

"only two algorithms"*

① identify subproblems

② change the input

Applications of Flows + Cuts

Gameplan

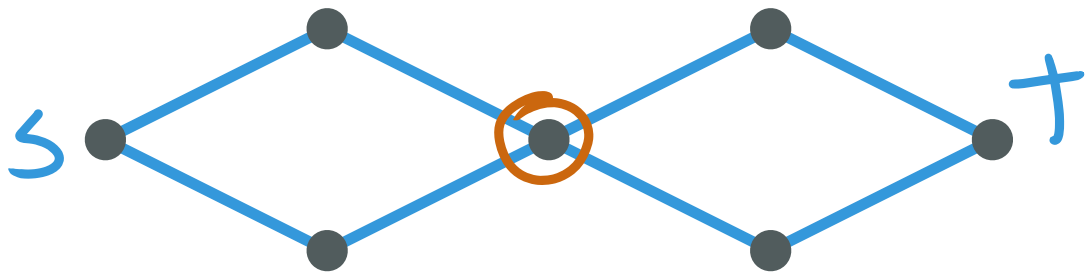
1. Survey a bunch of problems
2. Start solving them one by one

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

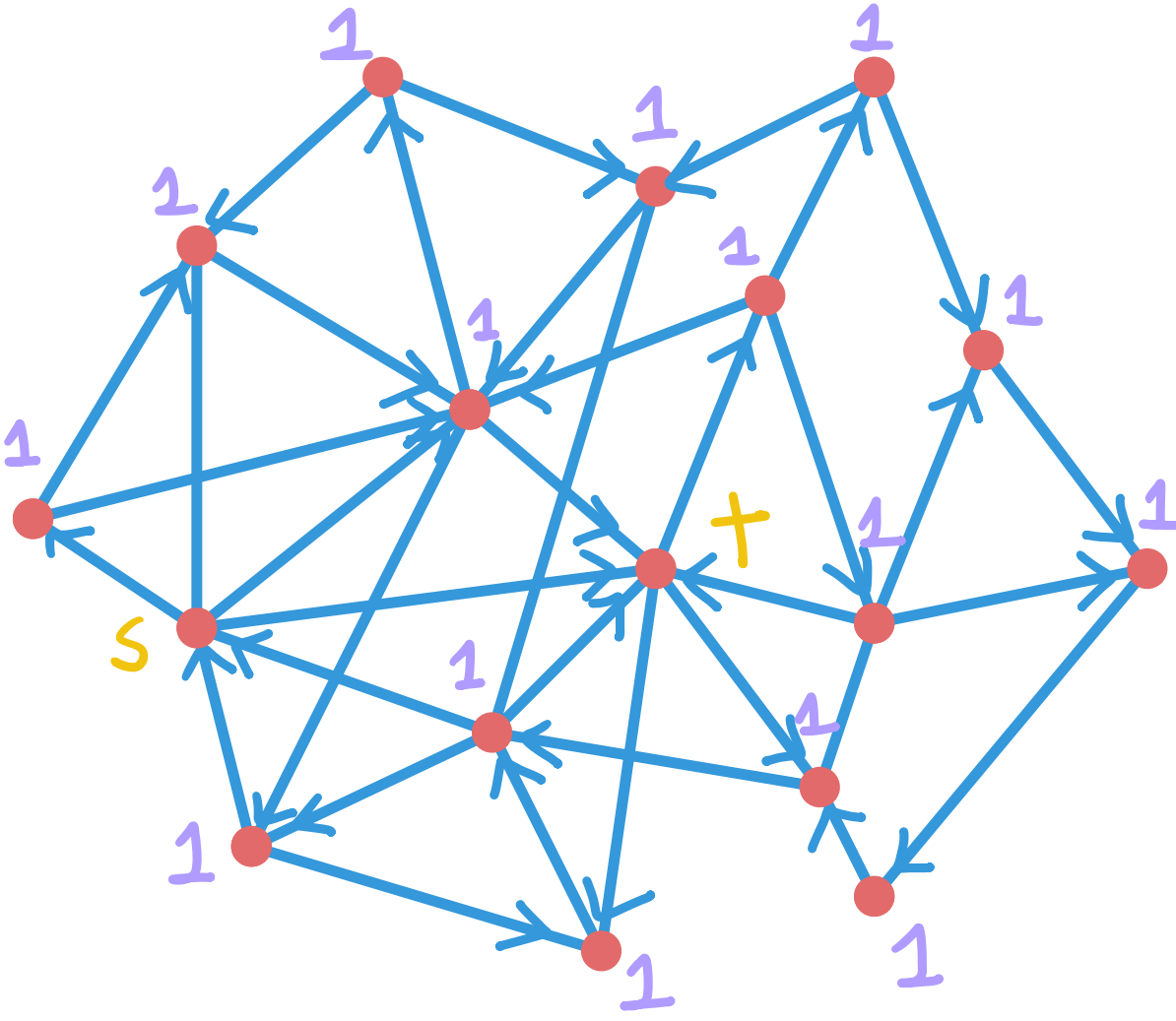
$c(v)$ = limit of flow through v

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

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

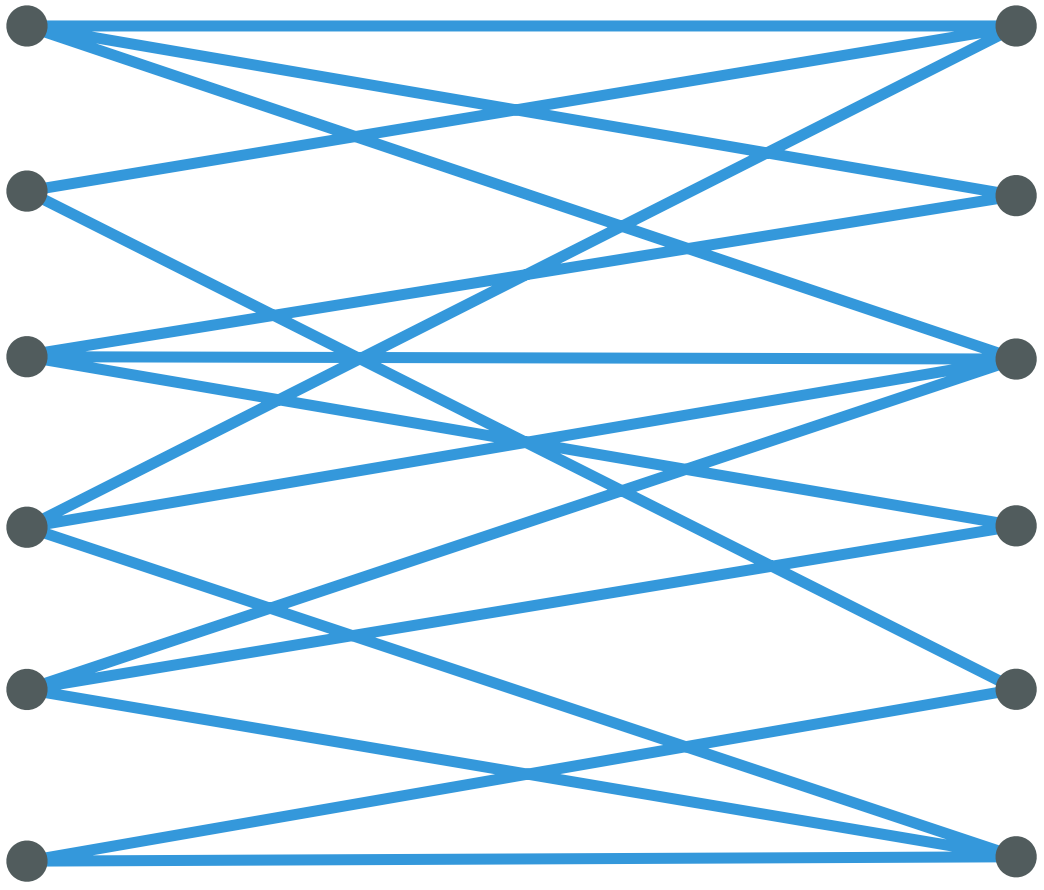
=> <u>vertex</u>

disjoint paths



Bipartite Matching

"Bipartite graph"

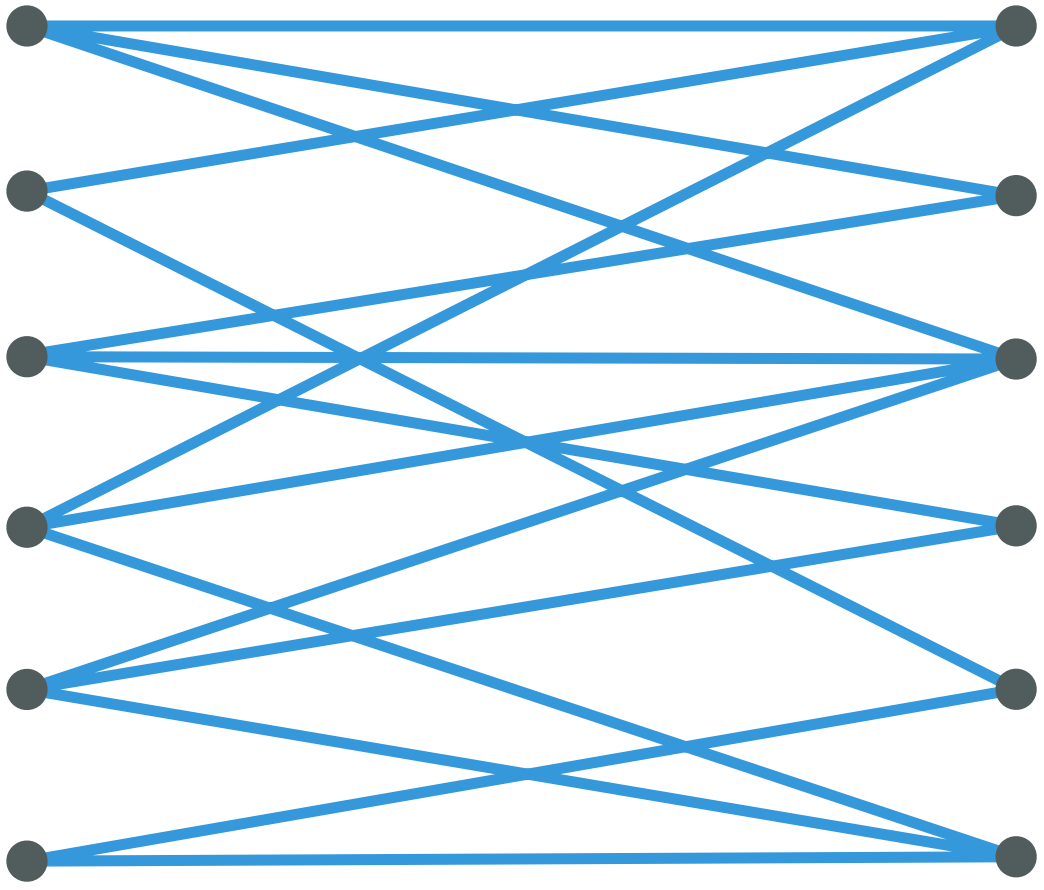


Goal: max set of edges w/ distinct endpoints

"matching"

Bipartite Vertex Cover

"Bipartite graph"

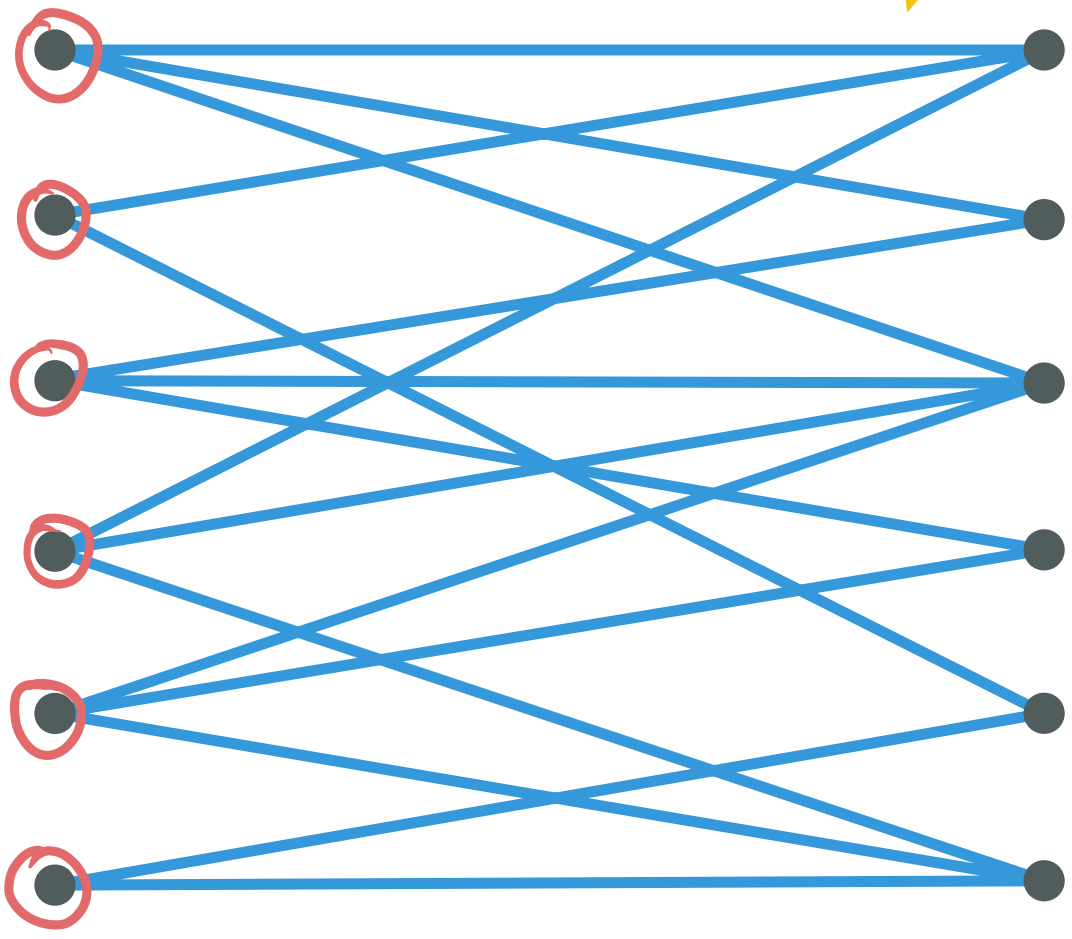


Goal: min set of vertices incident to all edges

"vertex cover"

Bipartite Vertex Cover

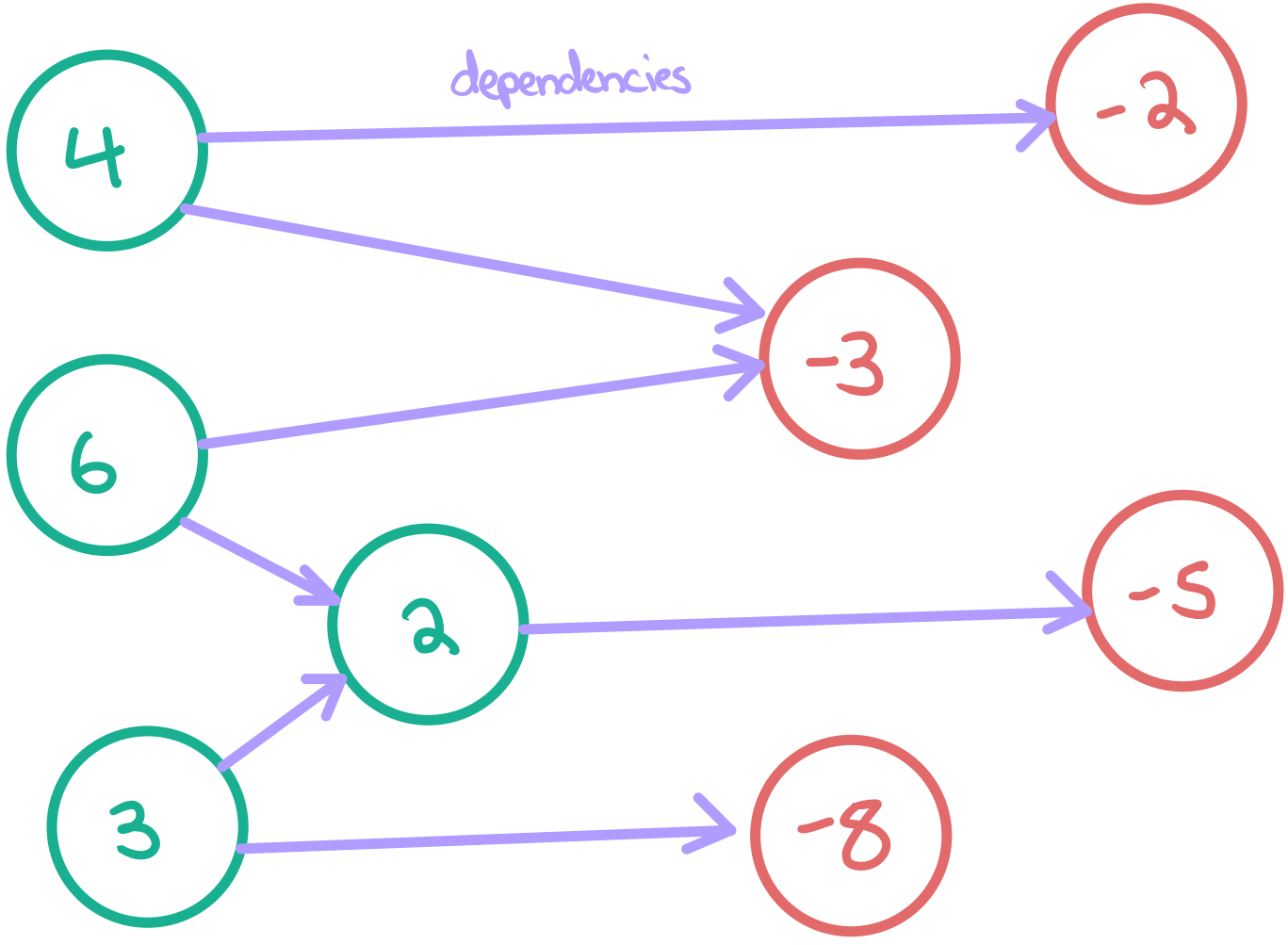
"Bipartite graph"



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?

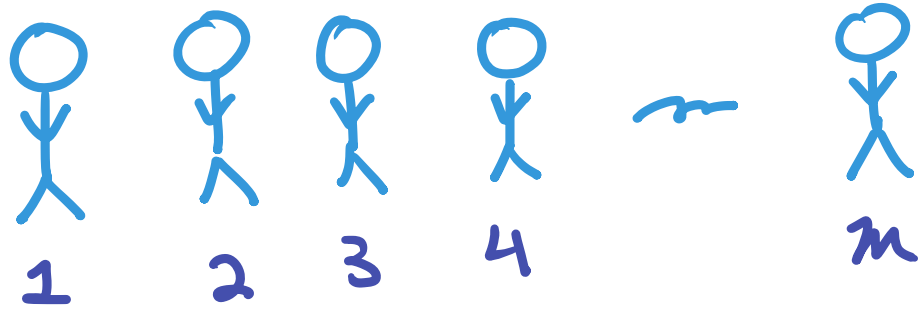
Team	Wins	Games left
New York	92	2
Baltimore	91	3
Toronto	91	3
Boston	90	2

Remaining schedule

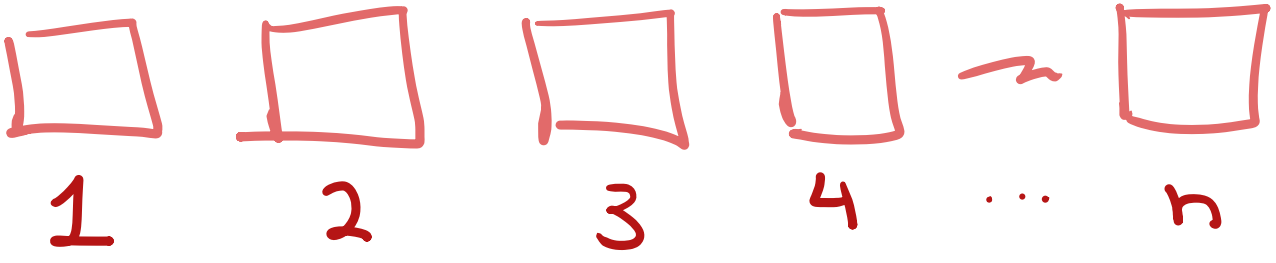
(wins)	(92) New York	(91) Baltimore	(91) Toronto	(90) Boston
New York	X	1	1	0
Baltimore	1	X	1	1
Toronto	1	1	X	1
Boston	0	1	1	X

Assignment Problems

m individuals



n jobs

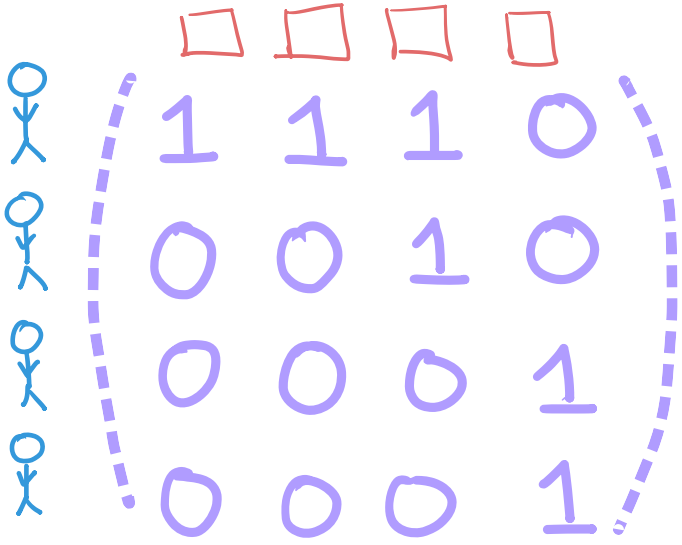


goal: fill as many jobs as possible w/ individuals

the catch: not everyone is qualified for all jobs

Assignment Problems

"Qualification matrix"

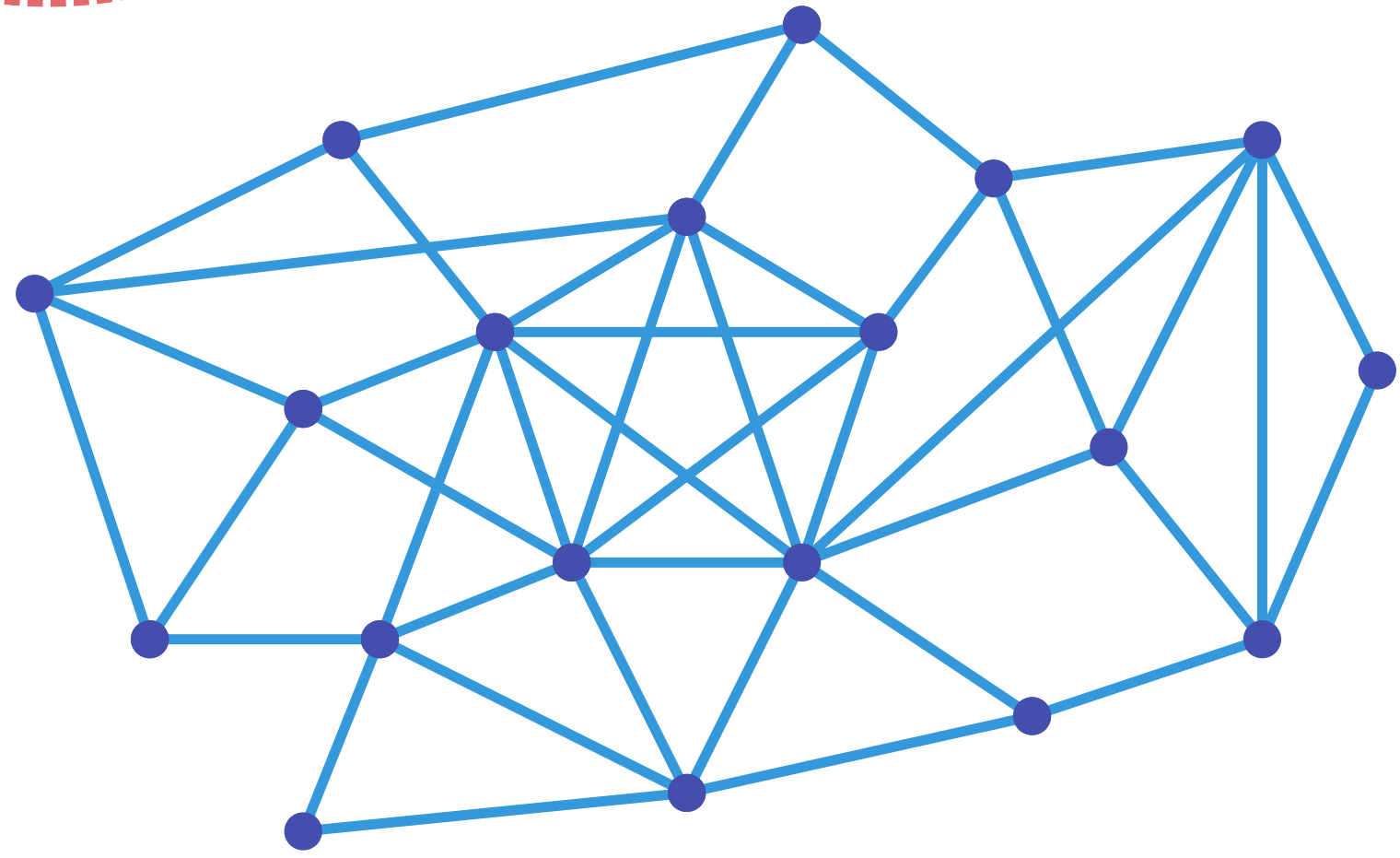


"1" = qualified

"0" = not qualified

Edge Orientation

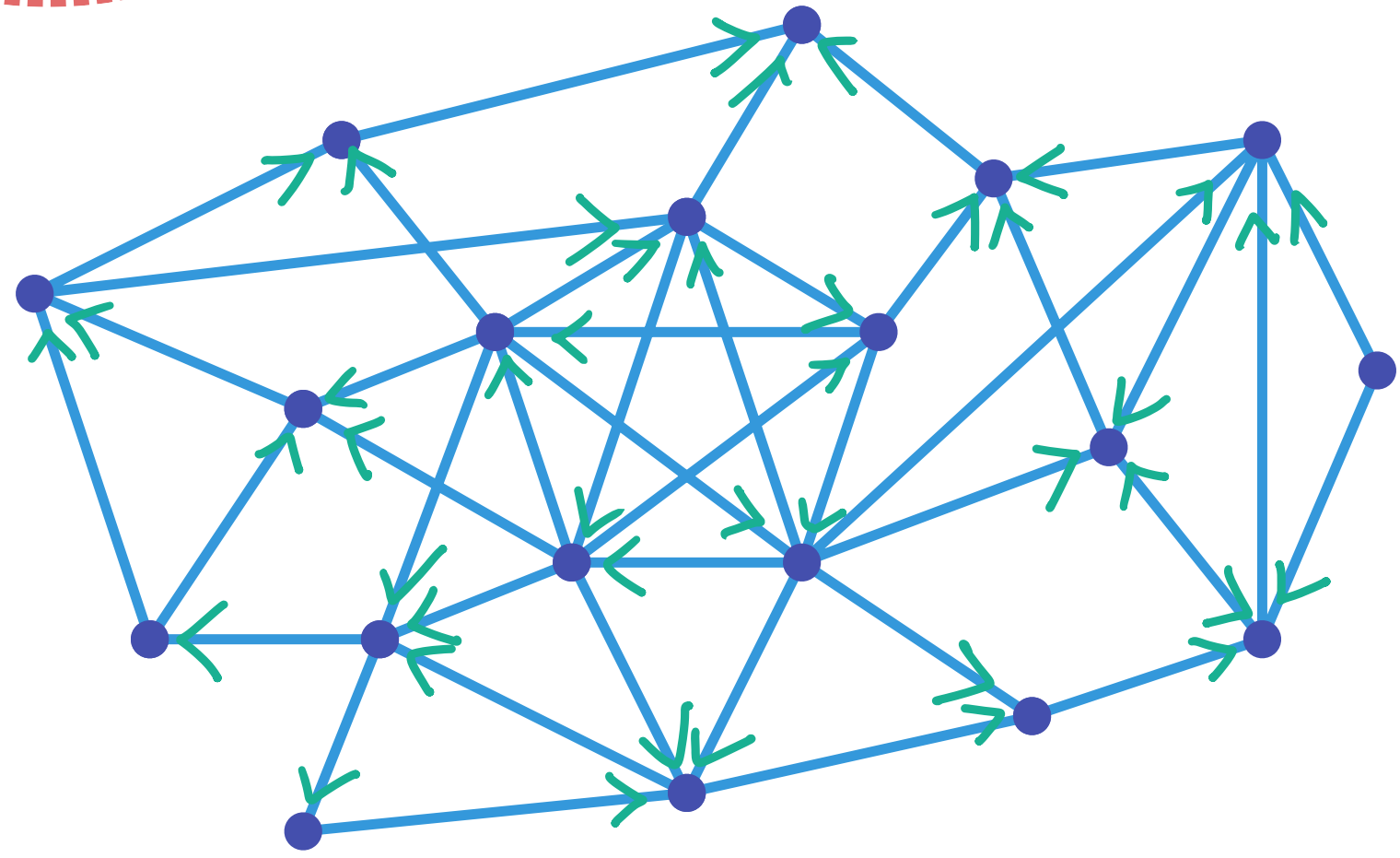
input: undirected graph



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

Edge Orientation

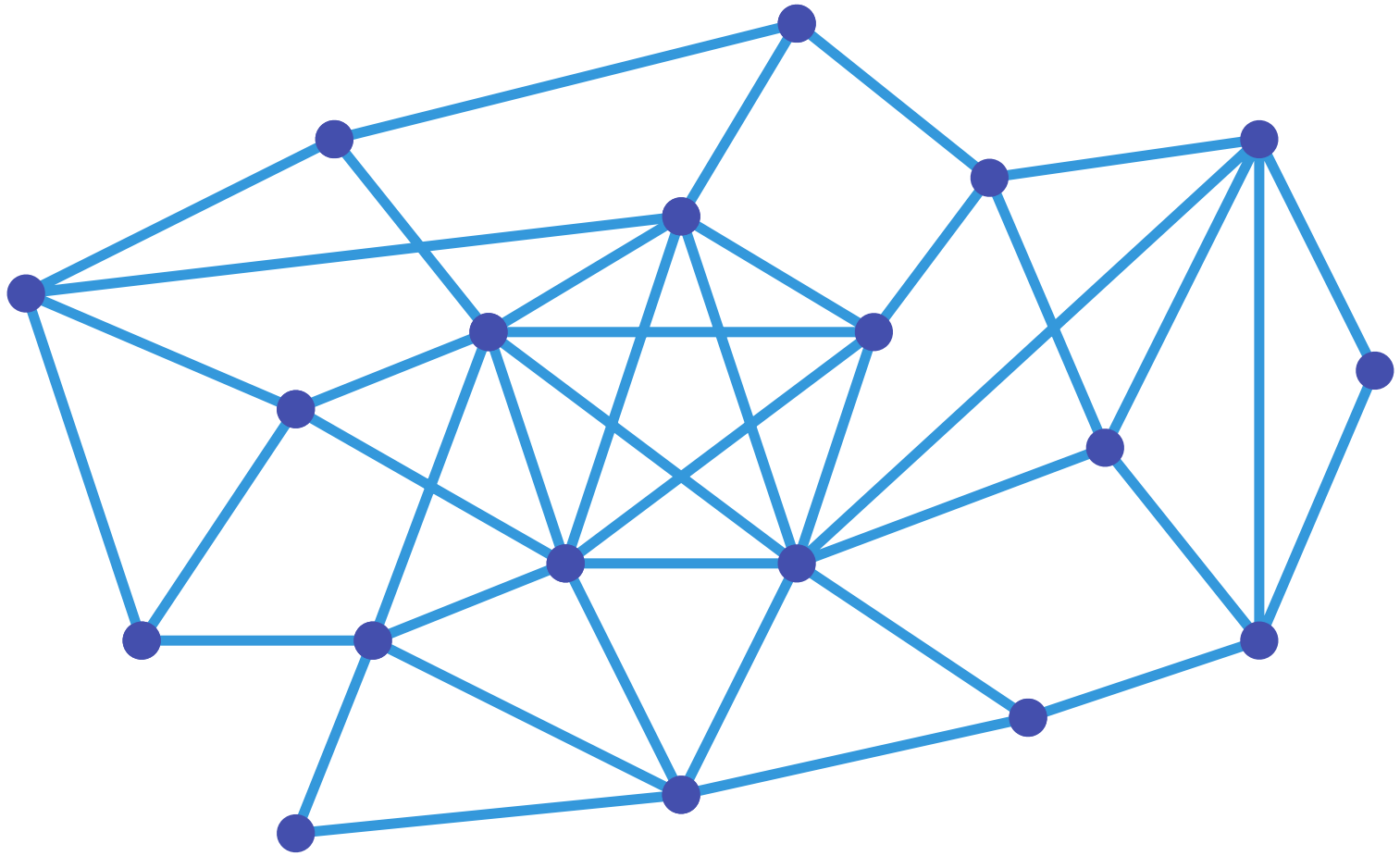
input: undirected graph



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

Densest Subgraph

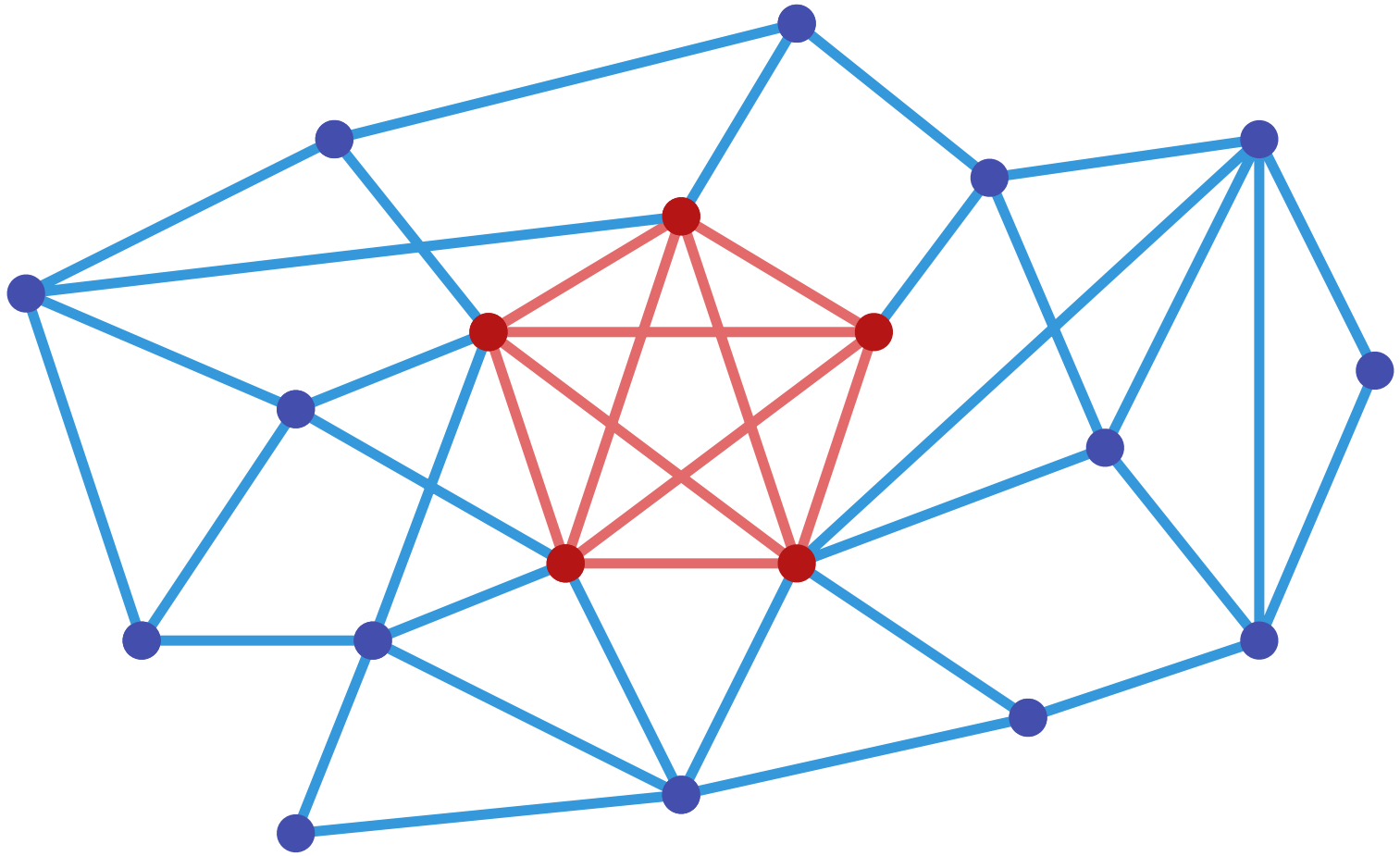
"density" = $\frac{\#\text{ edges}}{\#\text{ vertices}}$



Goal: compute densest subgraph

Densest Subgraph

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

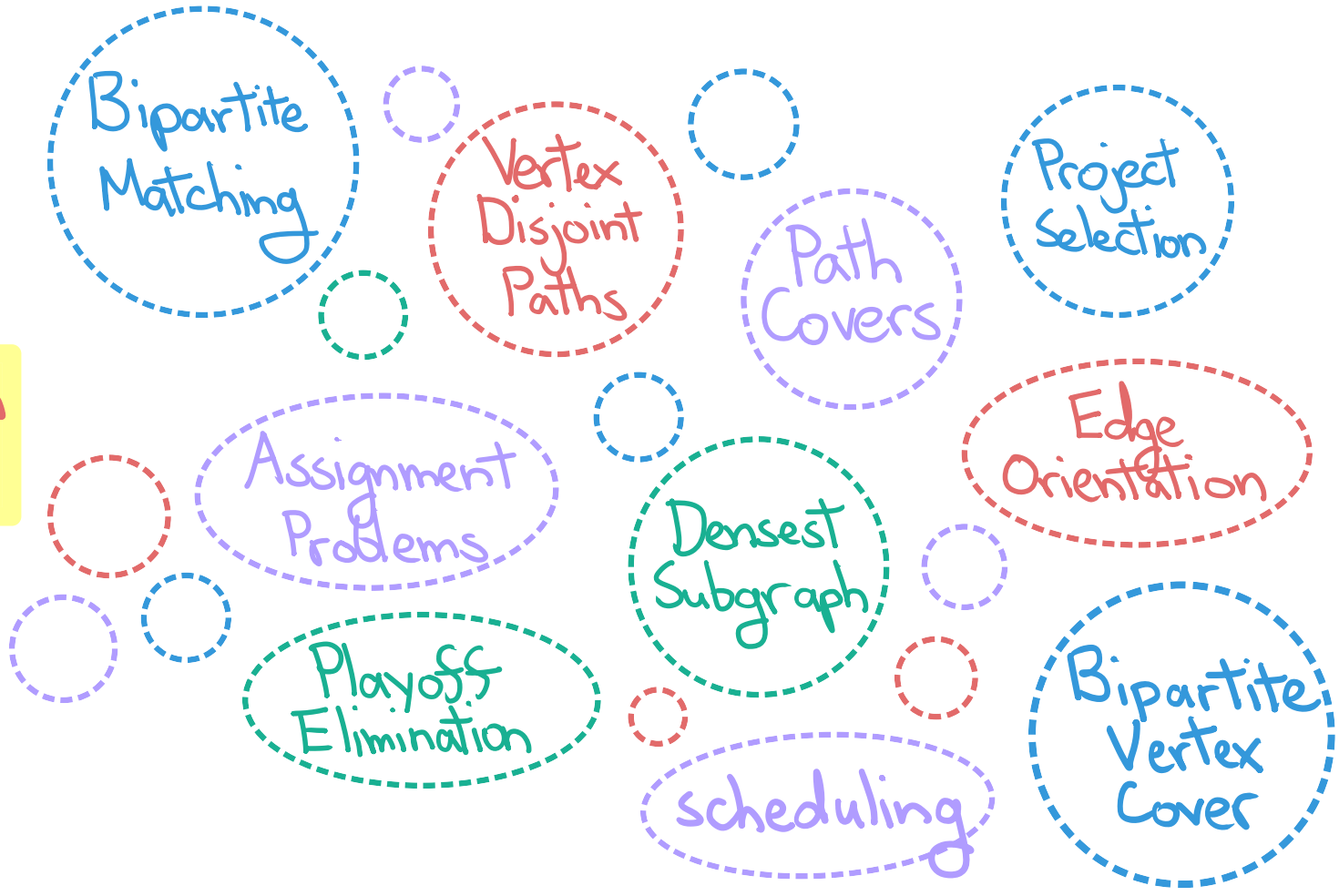


Goal: compute densest subgraph

Applications of Flows + Cuts

Gameplan

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



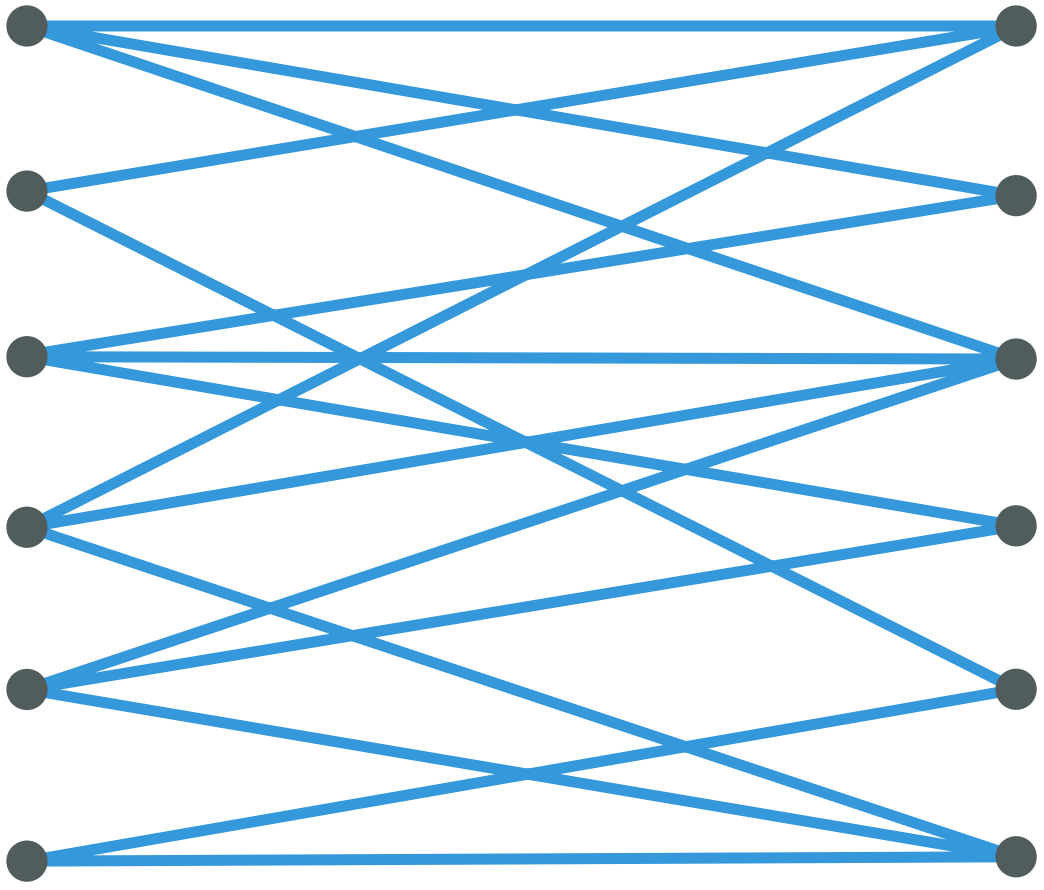
"only two algorithms"*

① identify subproblems

② change the input

Bipartite Matching

"Bipartite graph"

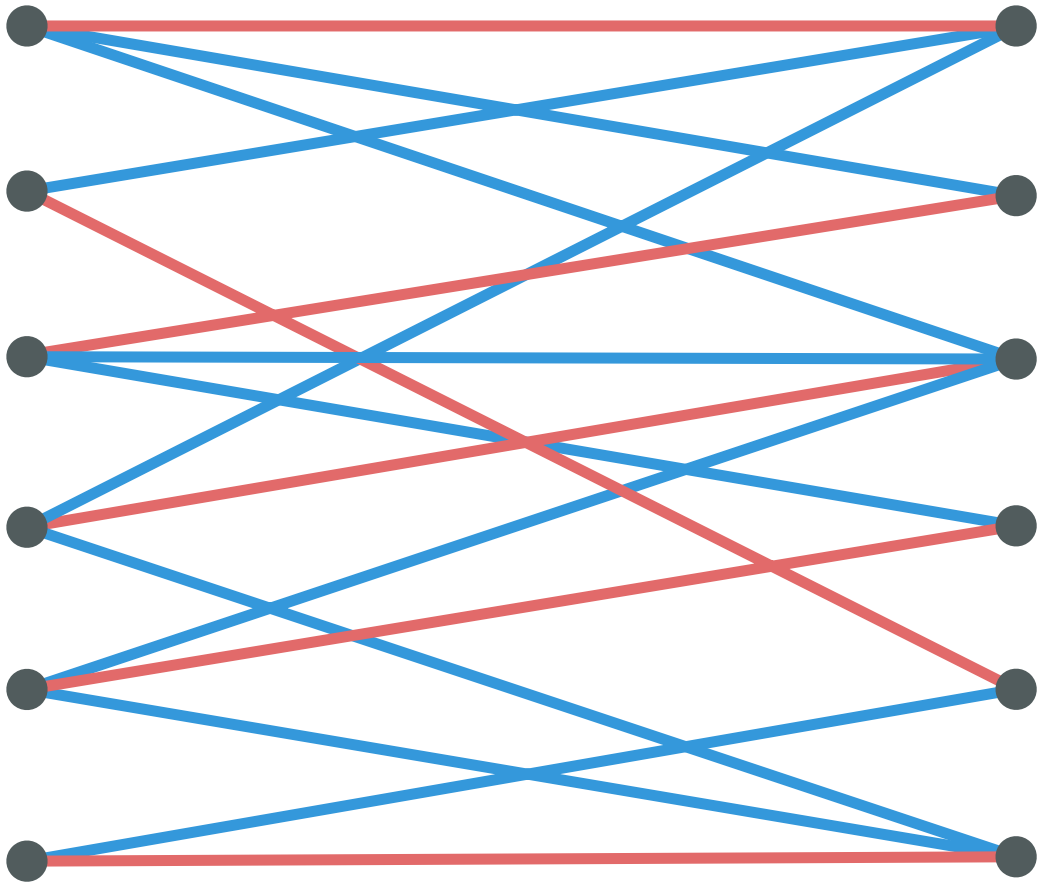


Goal: max set of edges w/ distinct endpoints

"matching"

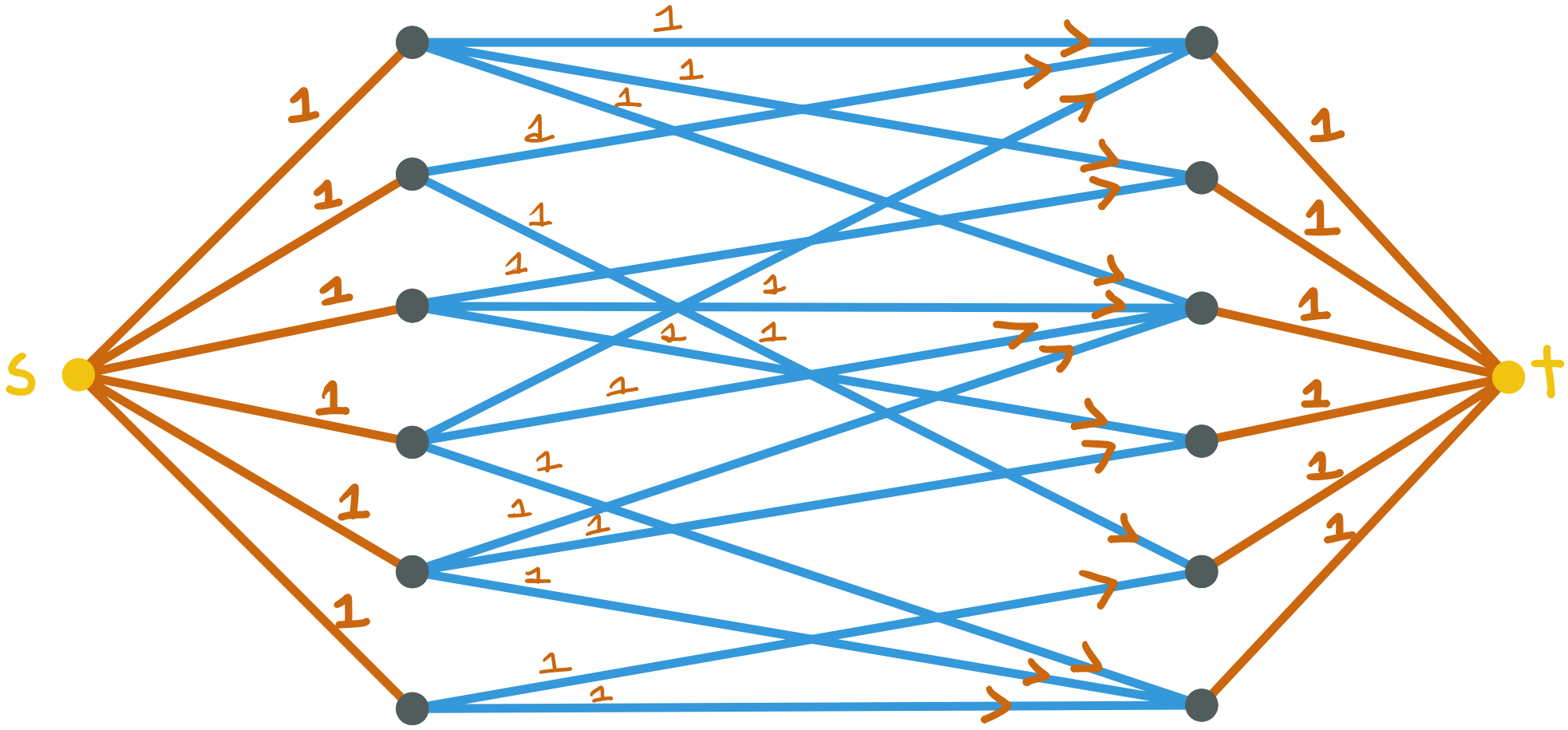
Bipartite Matching

(bipartite graph)



Goal: max set of edges w/ distinct endpoints

"matching"



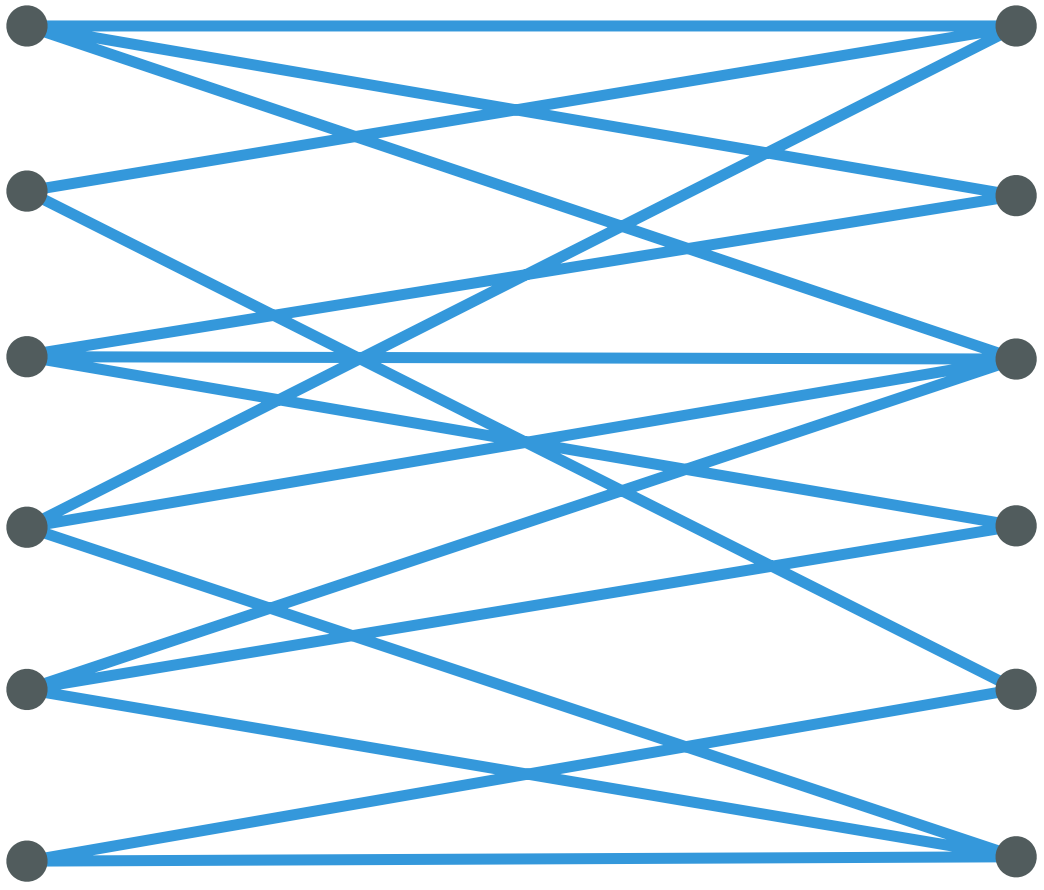
Matching w/ k edges

k edge-disjoint (s,t)-paths

(see lecture notes for detailed proof)

Bipartite Vertex Cover

"Bipartite graph"

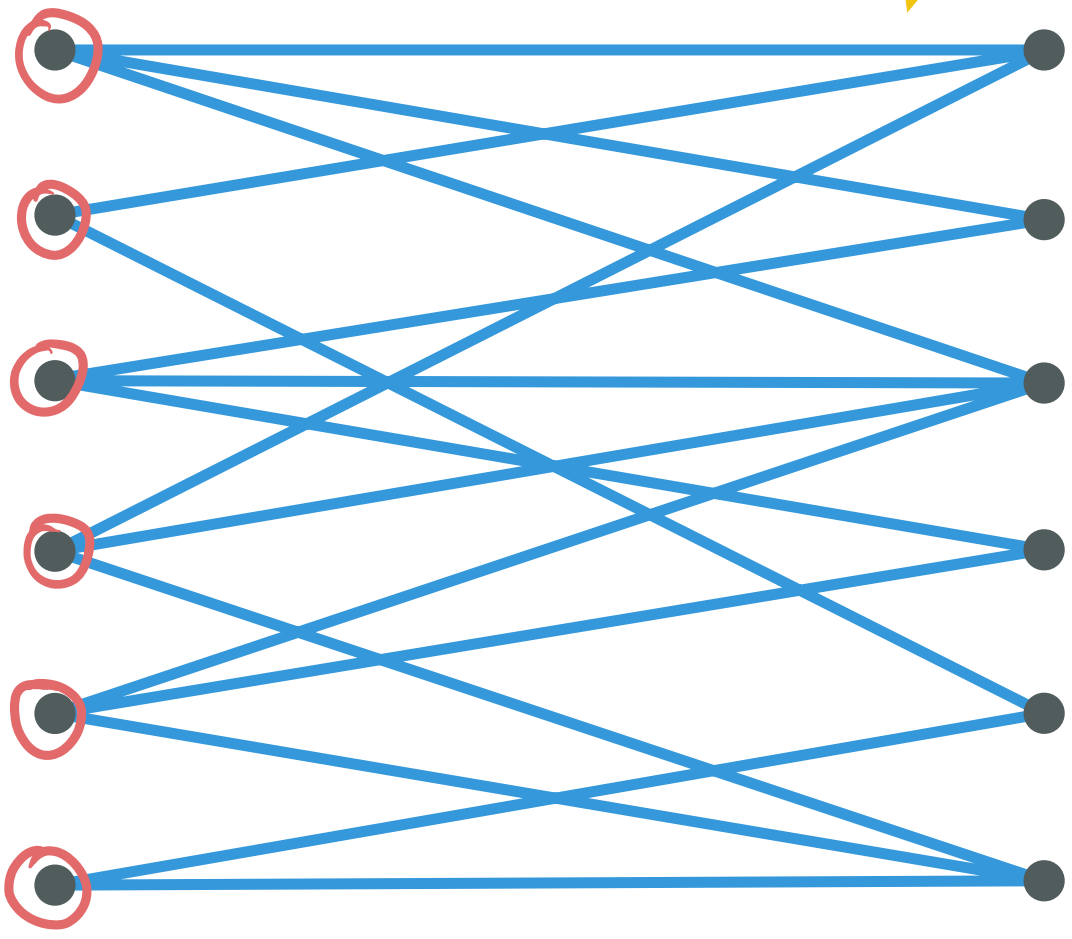


Goal: min set of vertices incident to all edges

"vertex cover"

Bipartite Vertex Cover

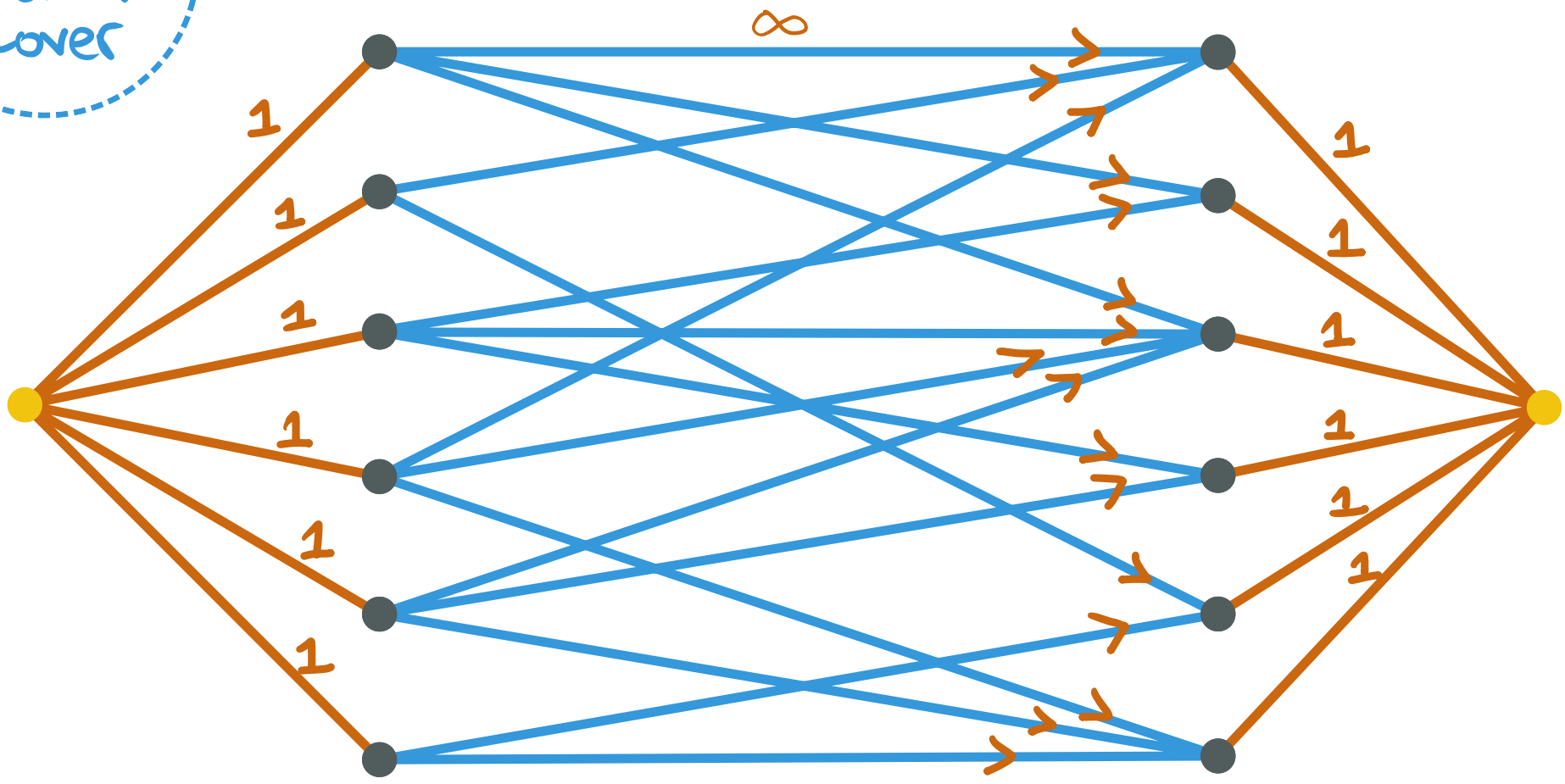
"Bipartite graph"



Goal: min set of vertices incident to all edges

"vertex cover"

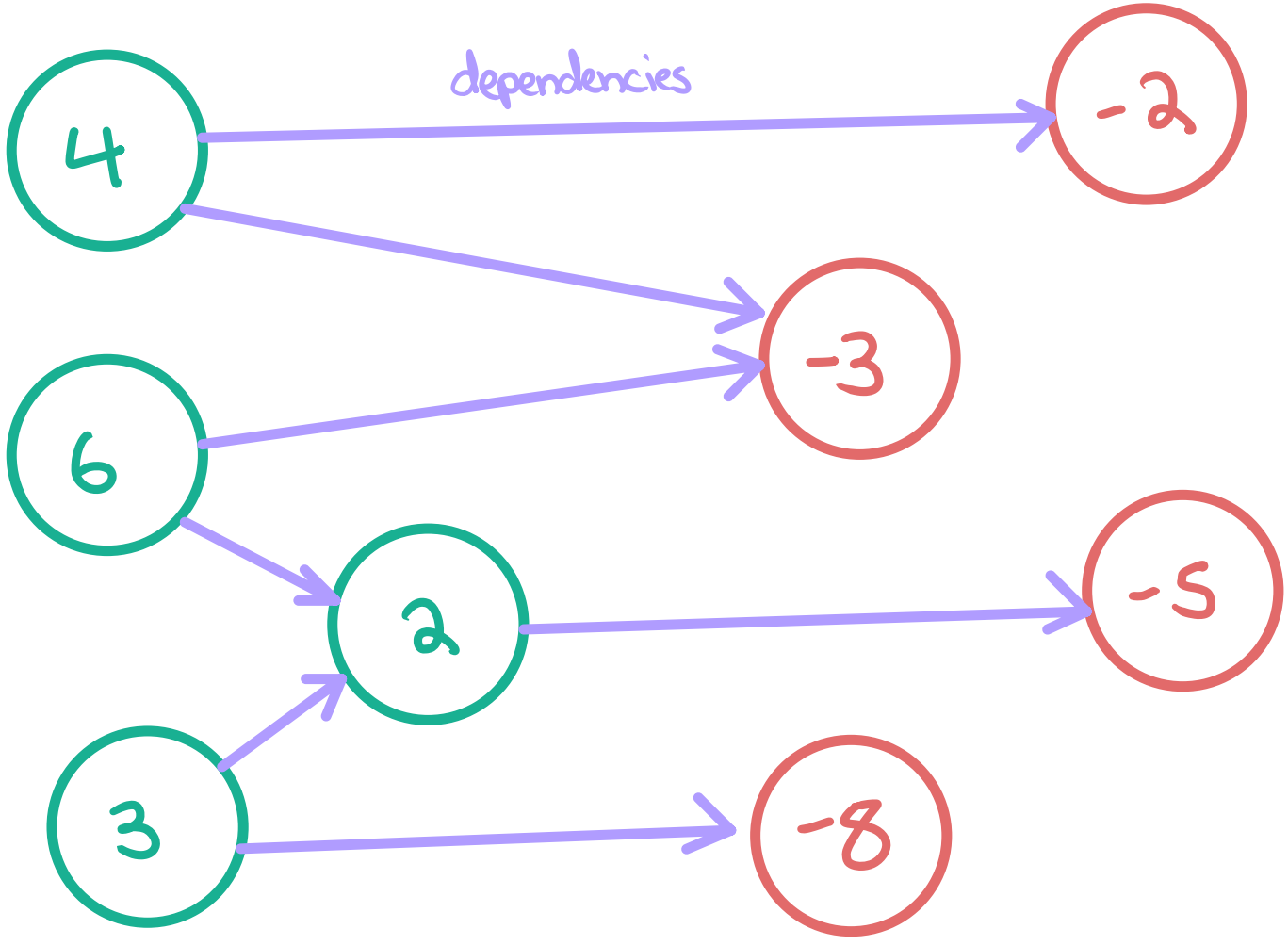
Bipartite Vertex Cover



(s,t)-cut of size k

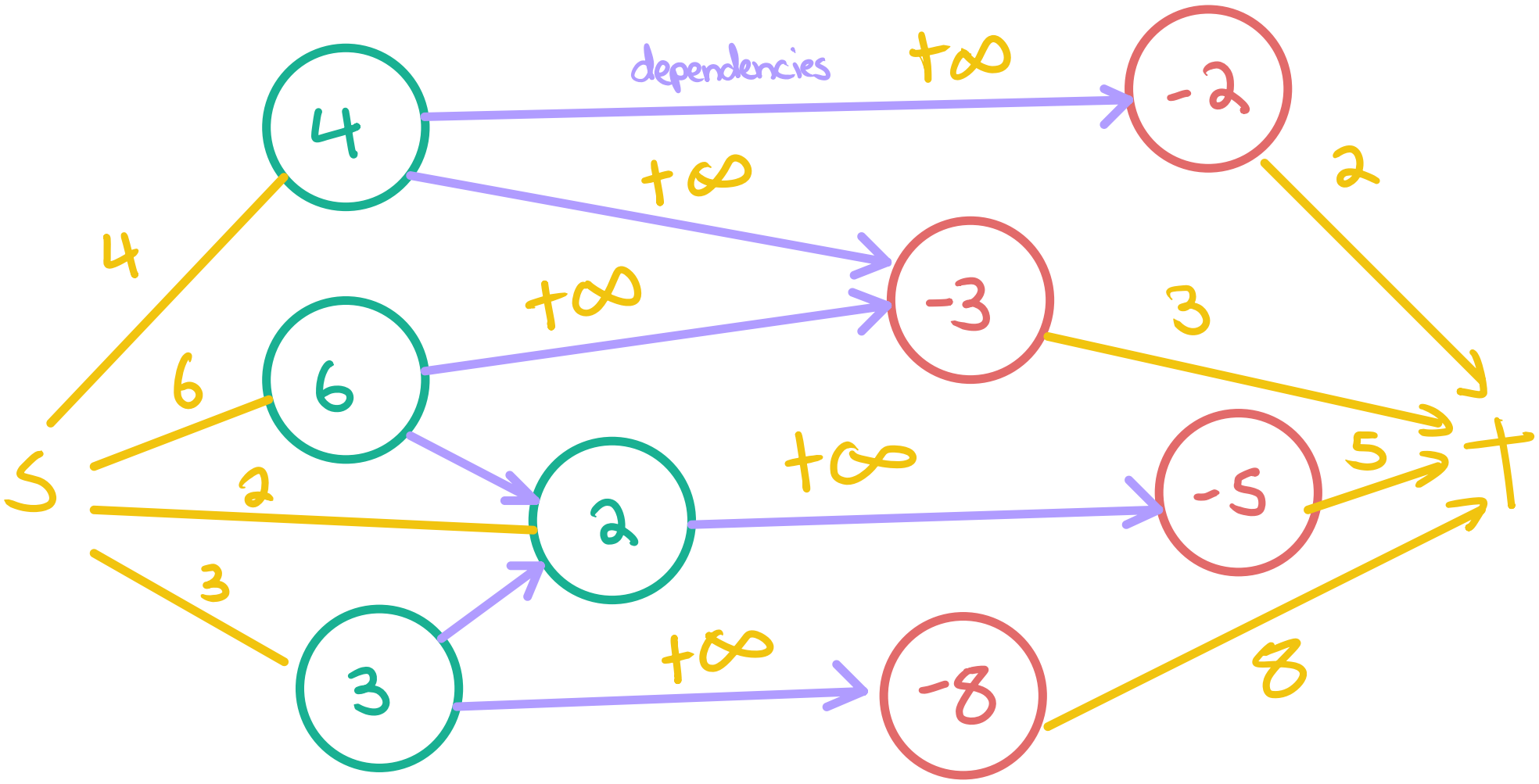
vertex cover of size k

Project Selection

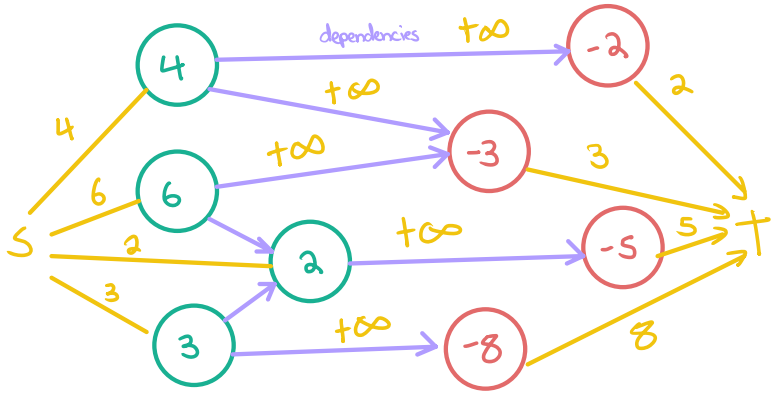


Projects have net profits (can be <0 for costs) and dependencies

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



claim: min-cut => best project selection


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

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

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

NewYork vs Baltimore

NewYork vs Toronto

Baltimore vs Toronto

Baltimore vs Boston

Toronto vs Boston

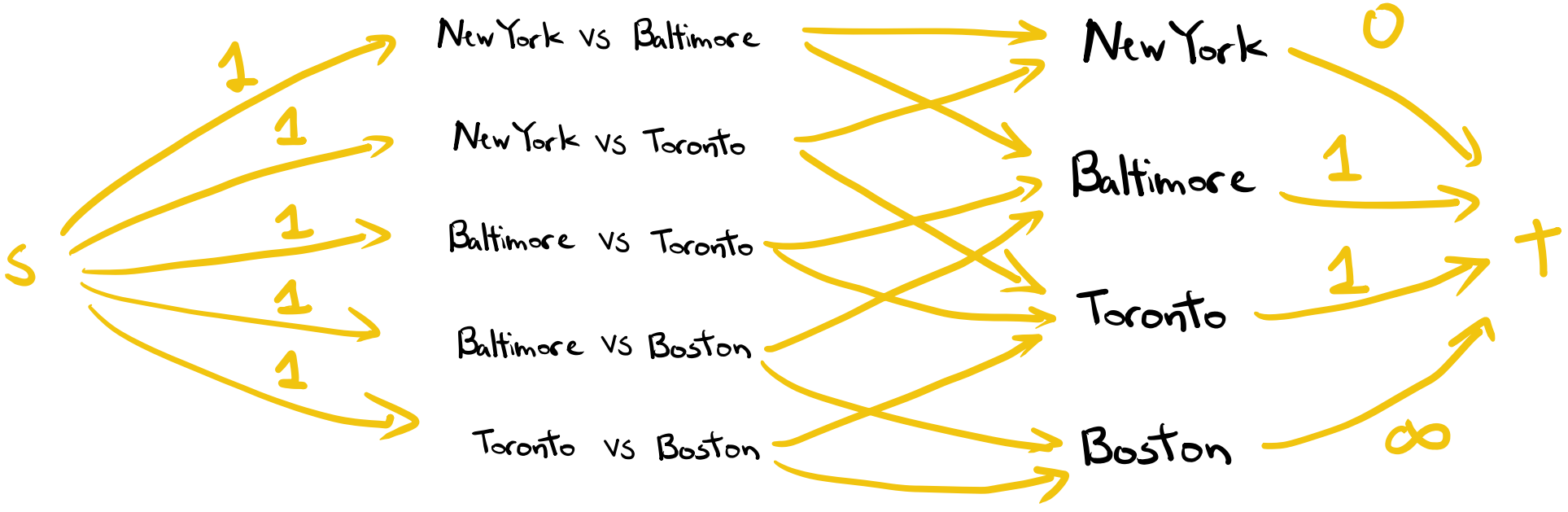
NewYork

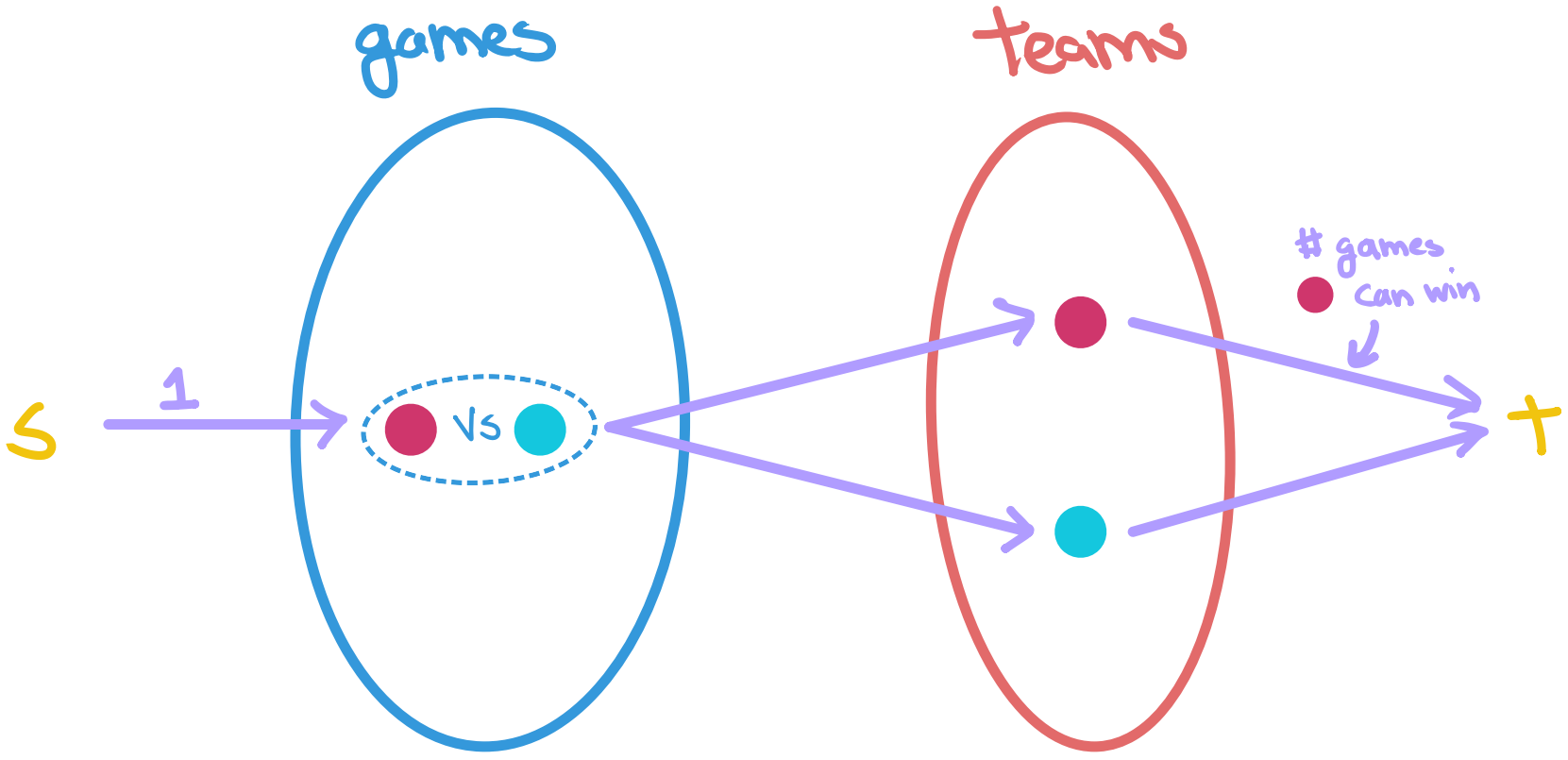
Baltimore

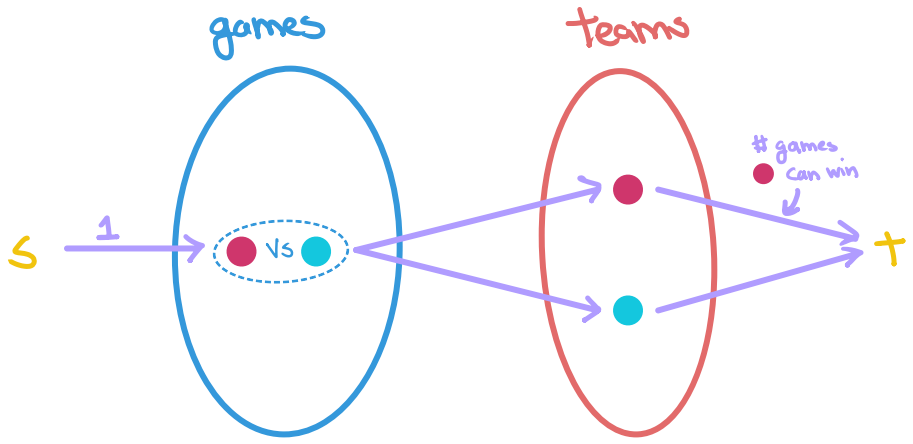
Toronto

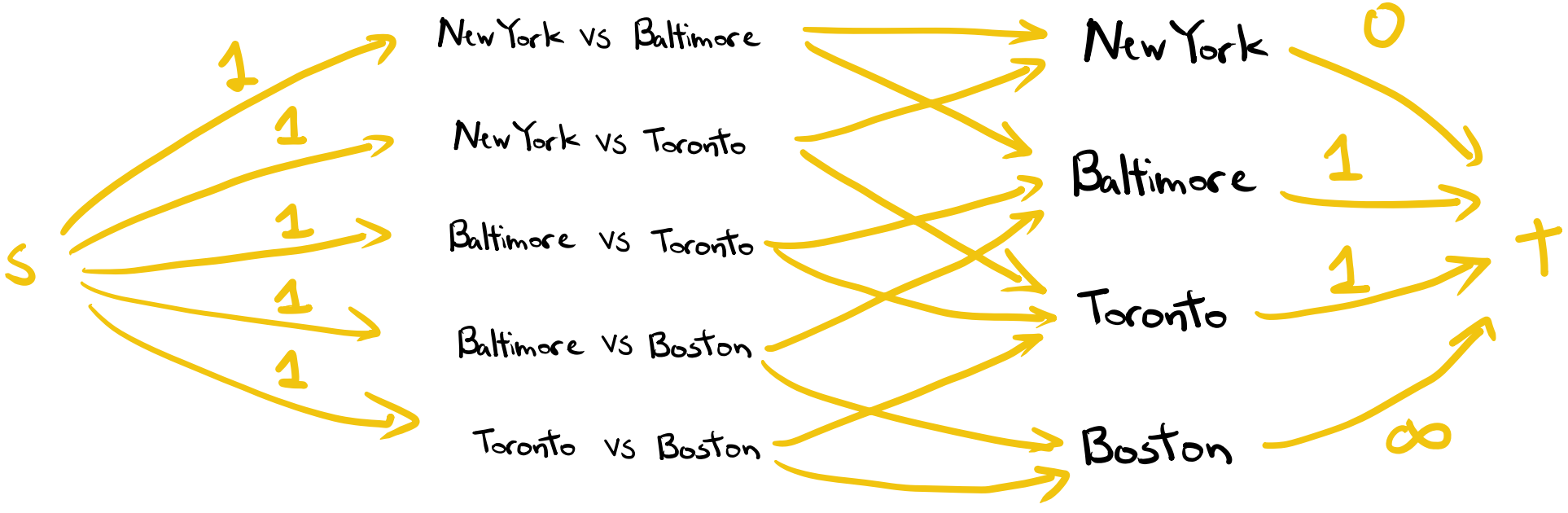
Boston

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



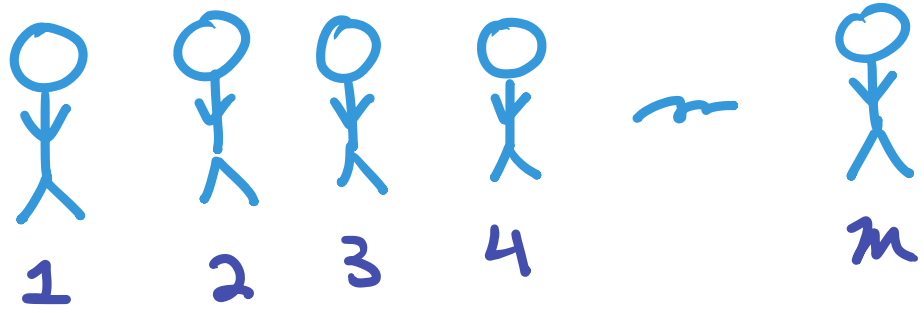

games
teams
games
can win
s
1
vs
t


games
teams
games can win
s
1
vs
t

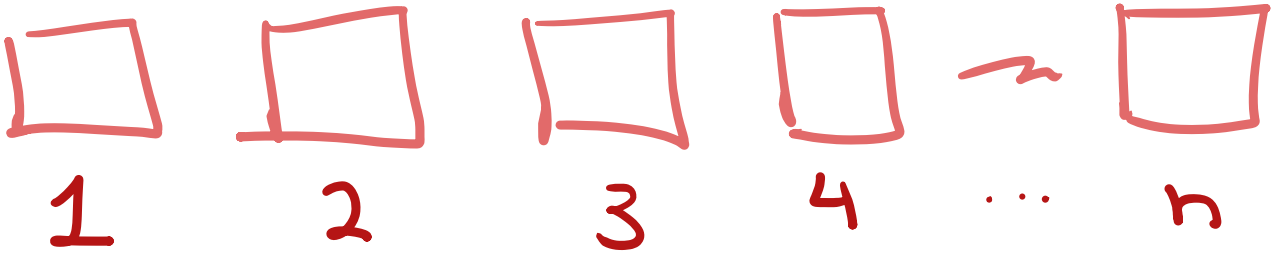

s
1
1
1
1
1
NewYork vs Baltimore
NewYork vs Toronto
Baltimore vs Toronto
Baltimore vs Boston
Toronto vs Boston
NewYork
Baltimore
Toronto
Boston
0
1
1
∞
t

Assignment Problems

m individuals



n jobs

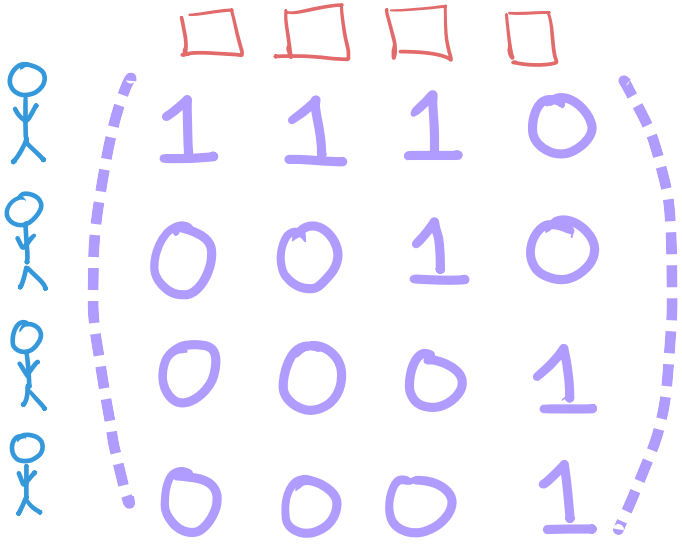


goal: fill as many jobs as possible w/ individuals

the catch: not everyone is qualified for all jobs

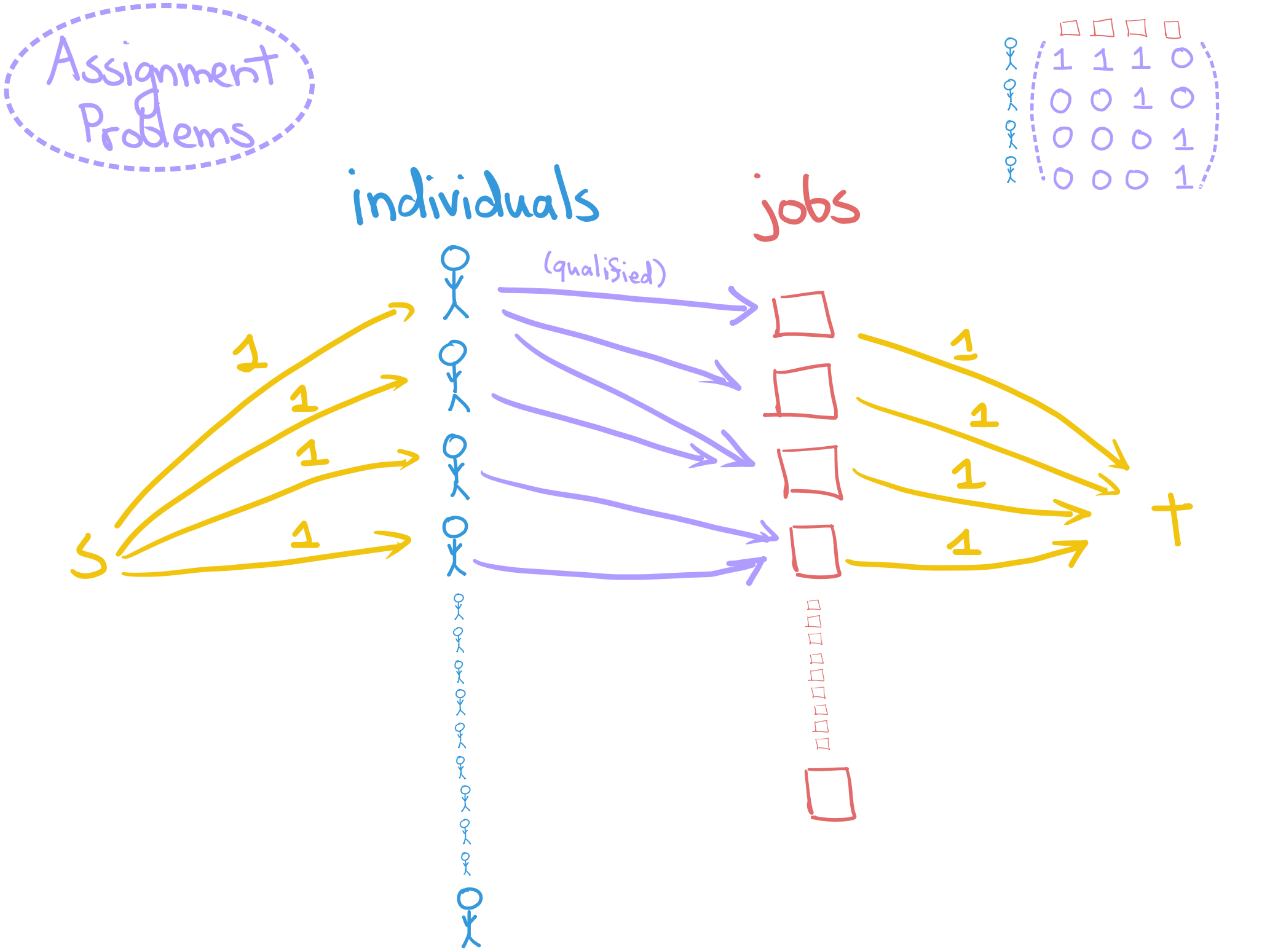
Assignment Problems

"Qualification matrix"



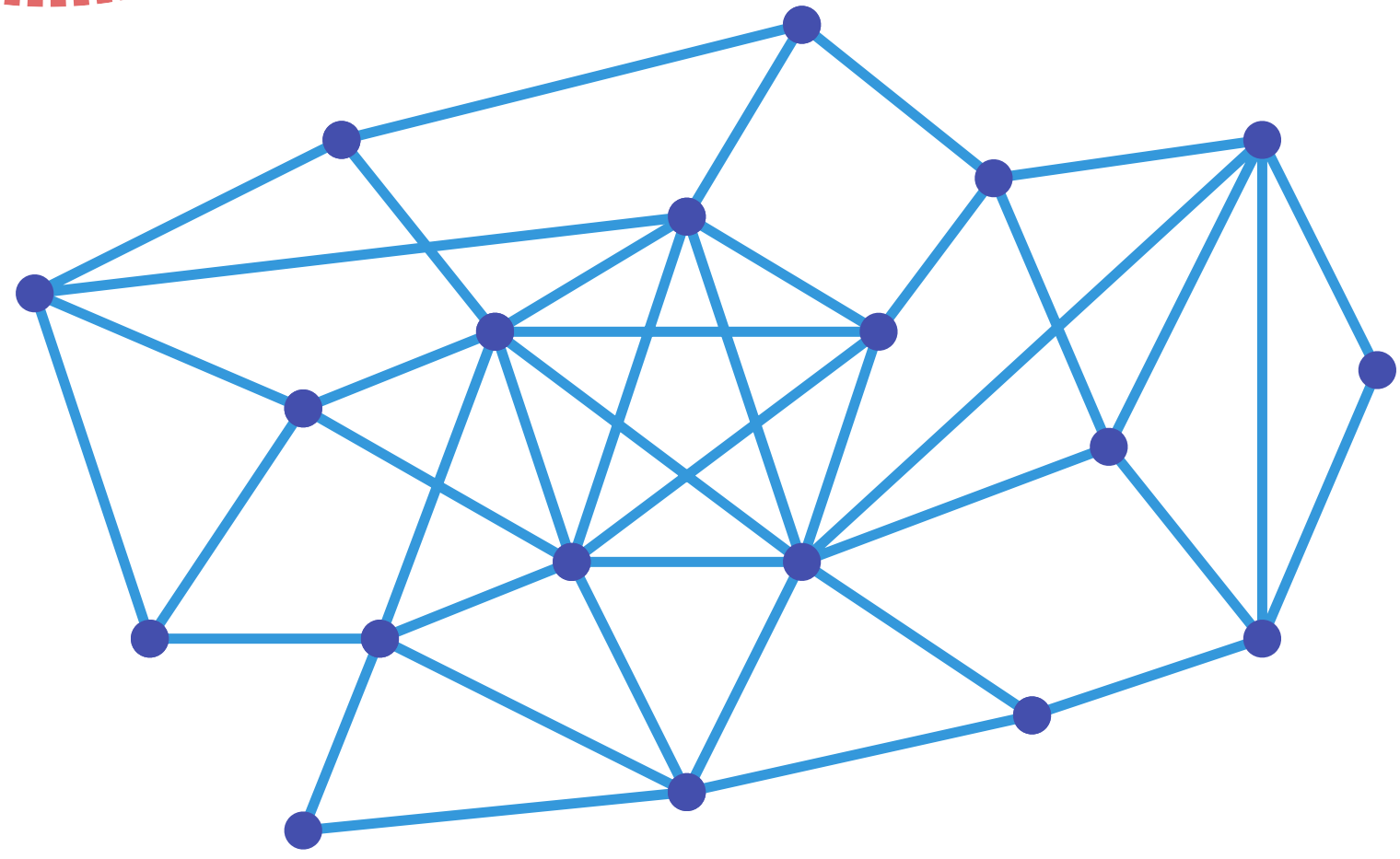
"1" = qualified

"0" = not qualified


Assignment Problems
1 1 1 0
0 0 1 0
0 0 0 1
0 0 0 1
individuals
jobs
(qualified)
1
1
1
1
1
1
1
1
s
t

Edge Orientation

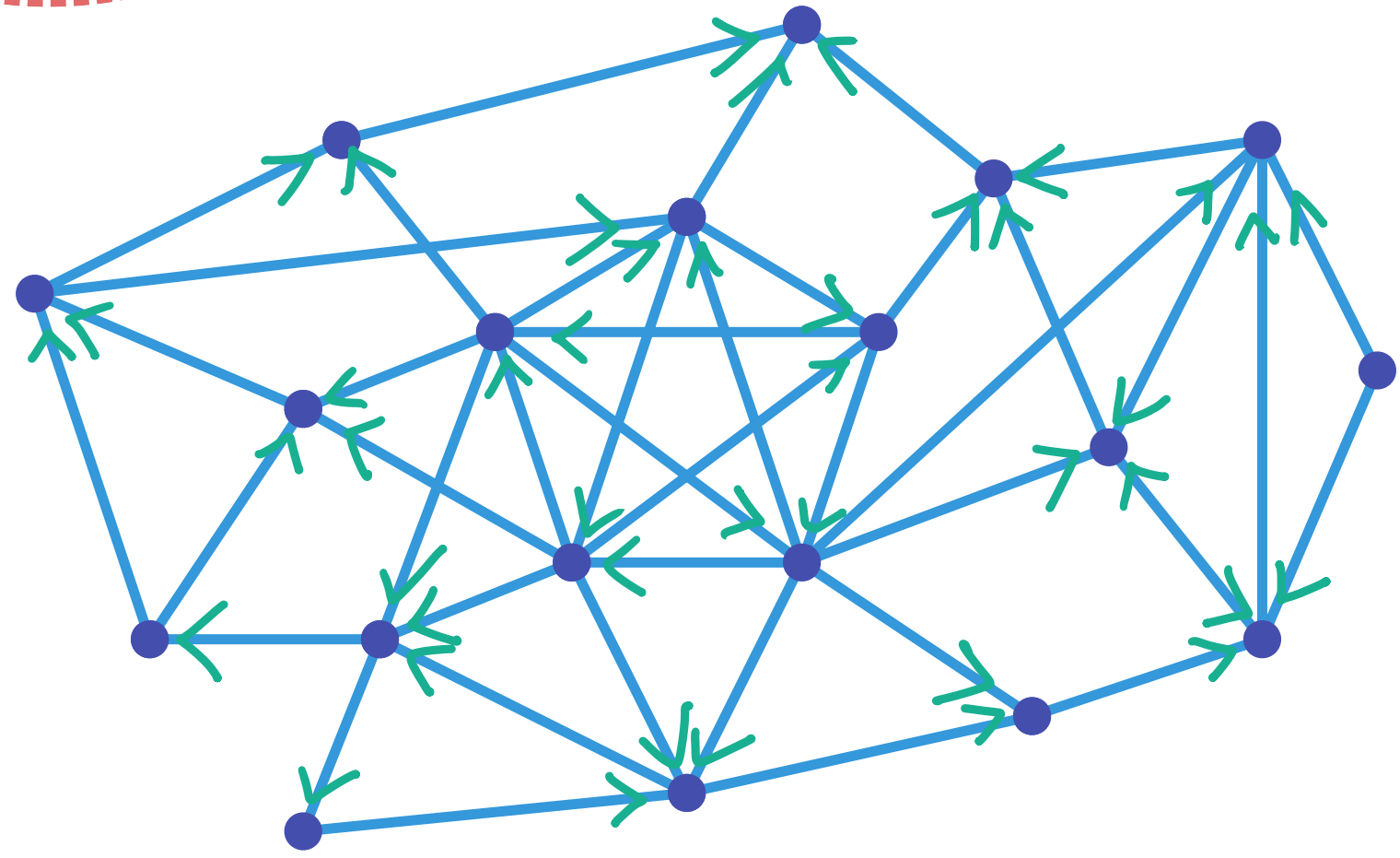
input: undirected graph



Goal: direct the edges to minimize the max density

Edge Orientation

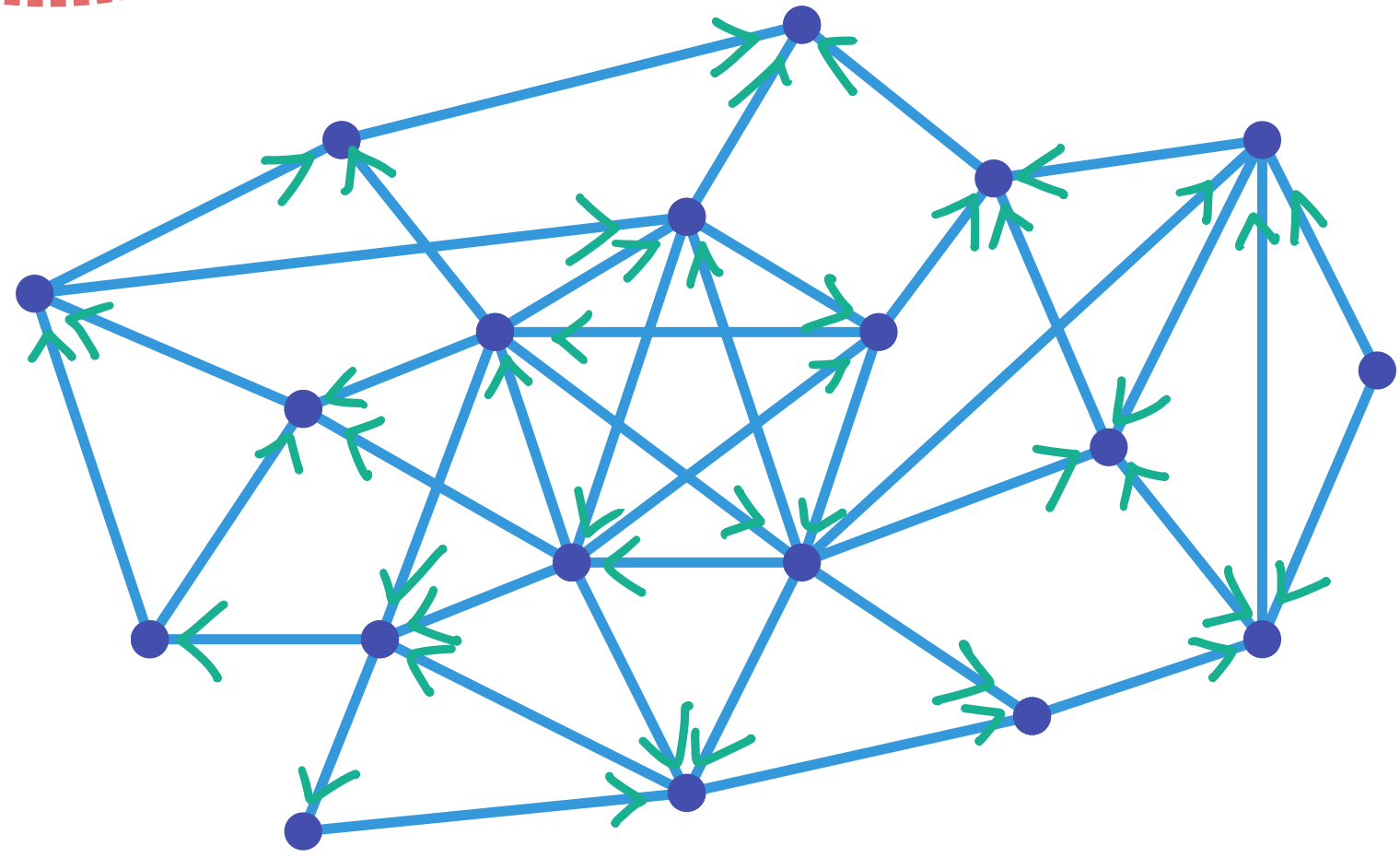
input: undirected graph



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

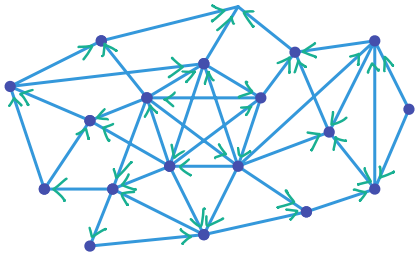
Edge Orientation

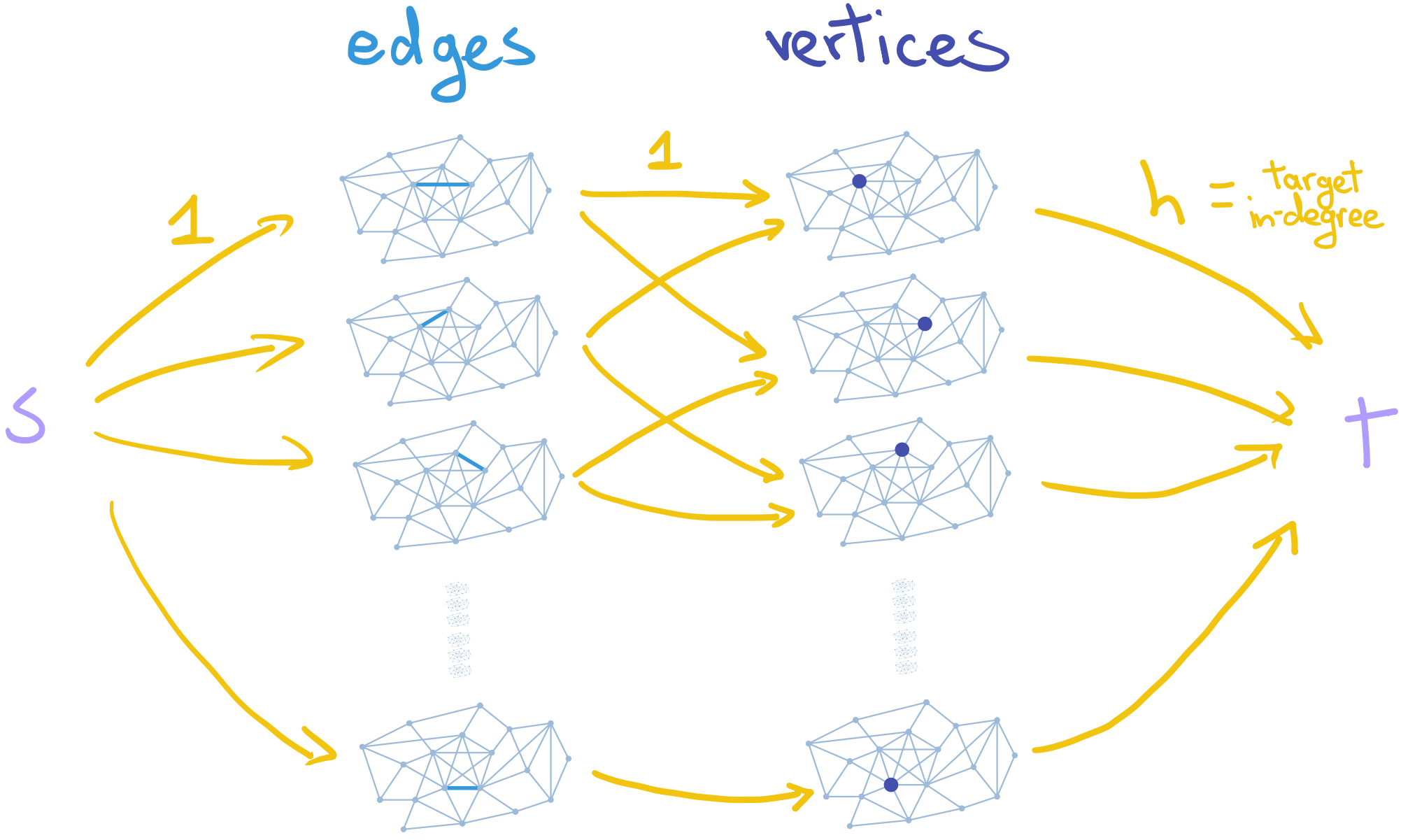
input: undirected graph



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

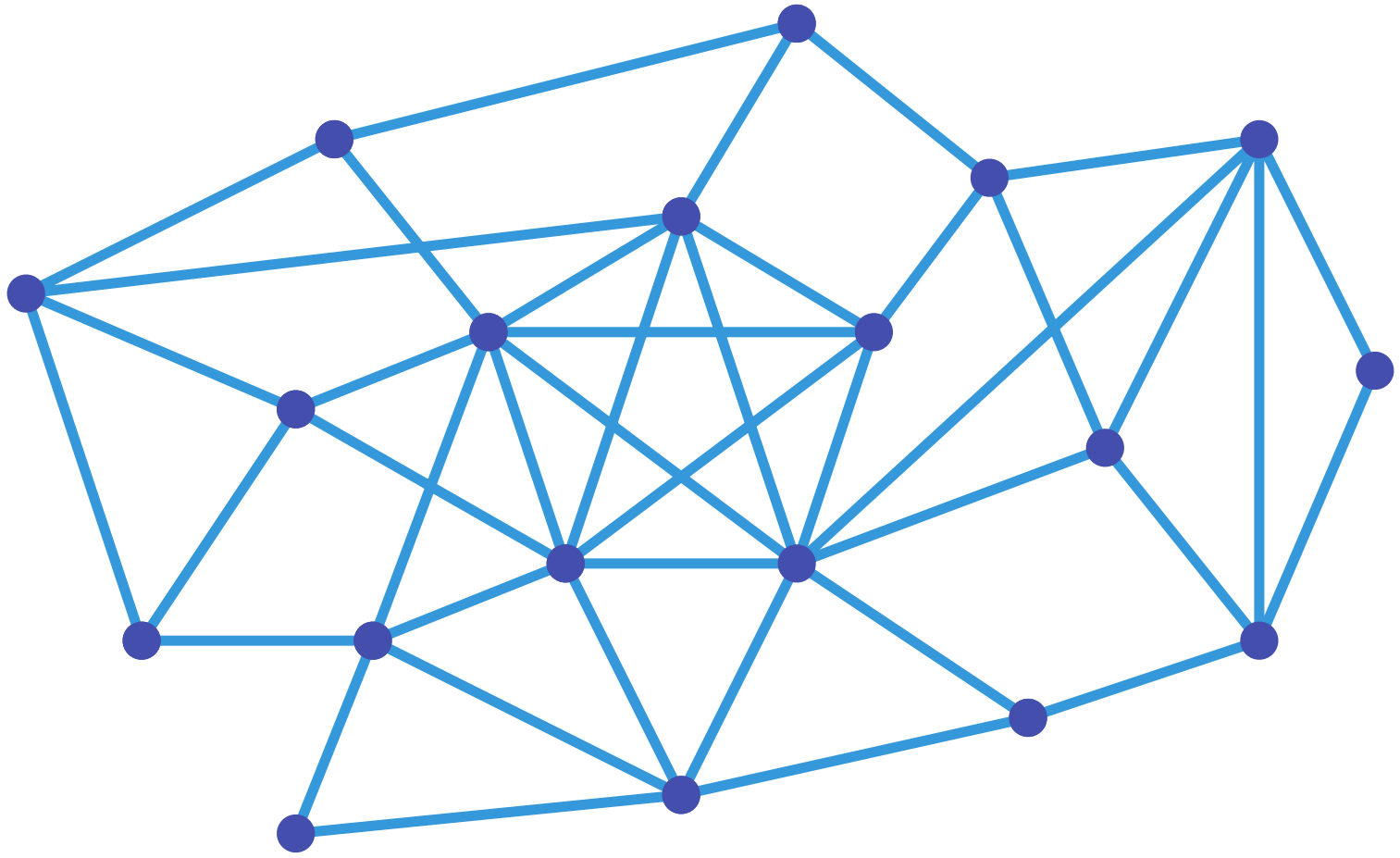
Edge Orientation





Densest Subgraph

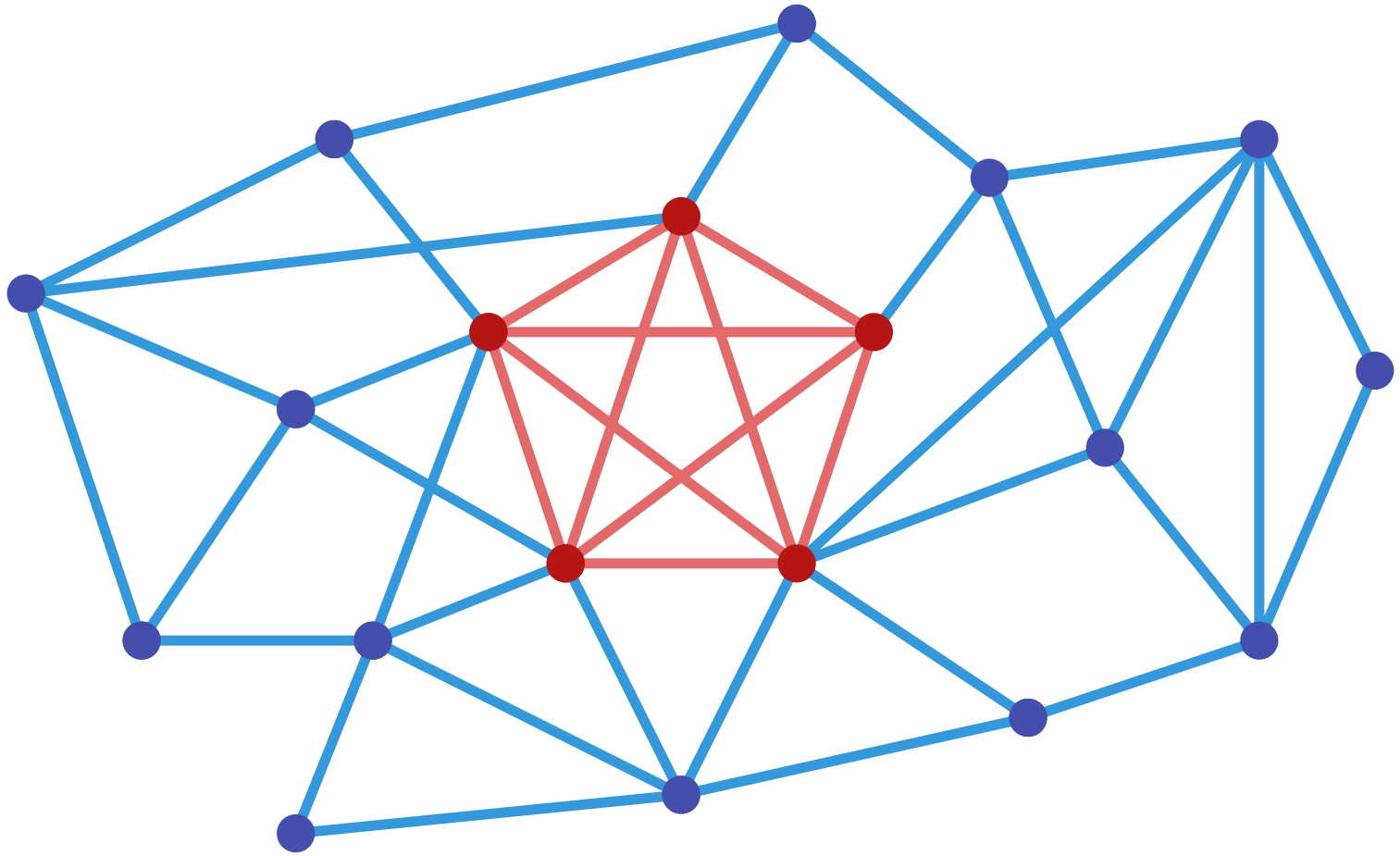
"density" = $\frac{\#\text{ edges}}{\#\text{ vertices}}$



Goal: compute densest subgraph

Densest Subgraph

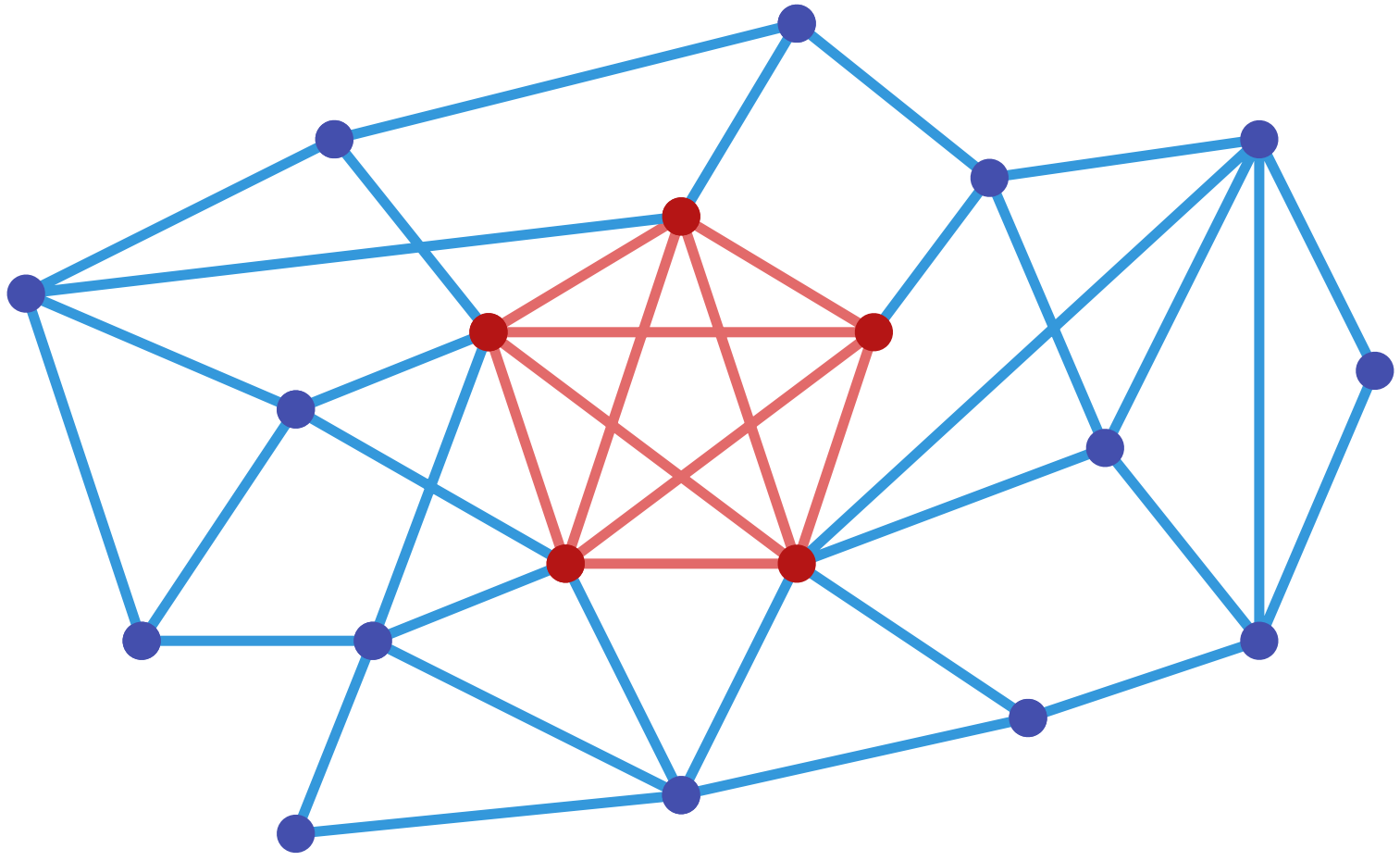
"density" = $\frac{\#\text{ edges}}{\#\text{ vertices}}$



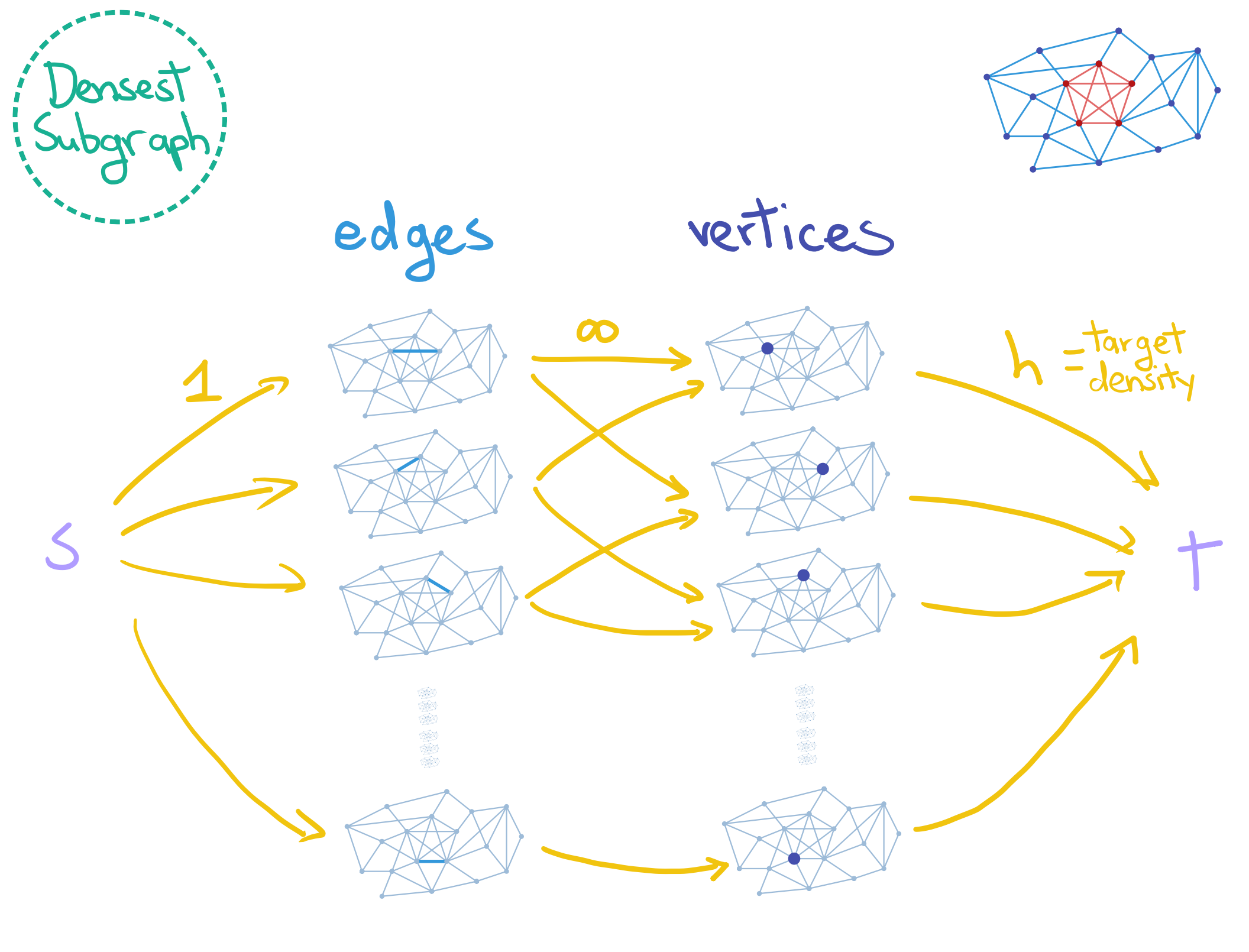
Goal: compute densest subgraph

Densest Subgraph

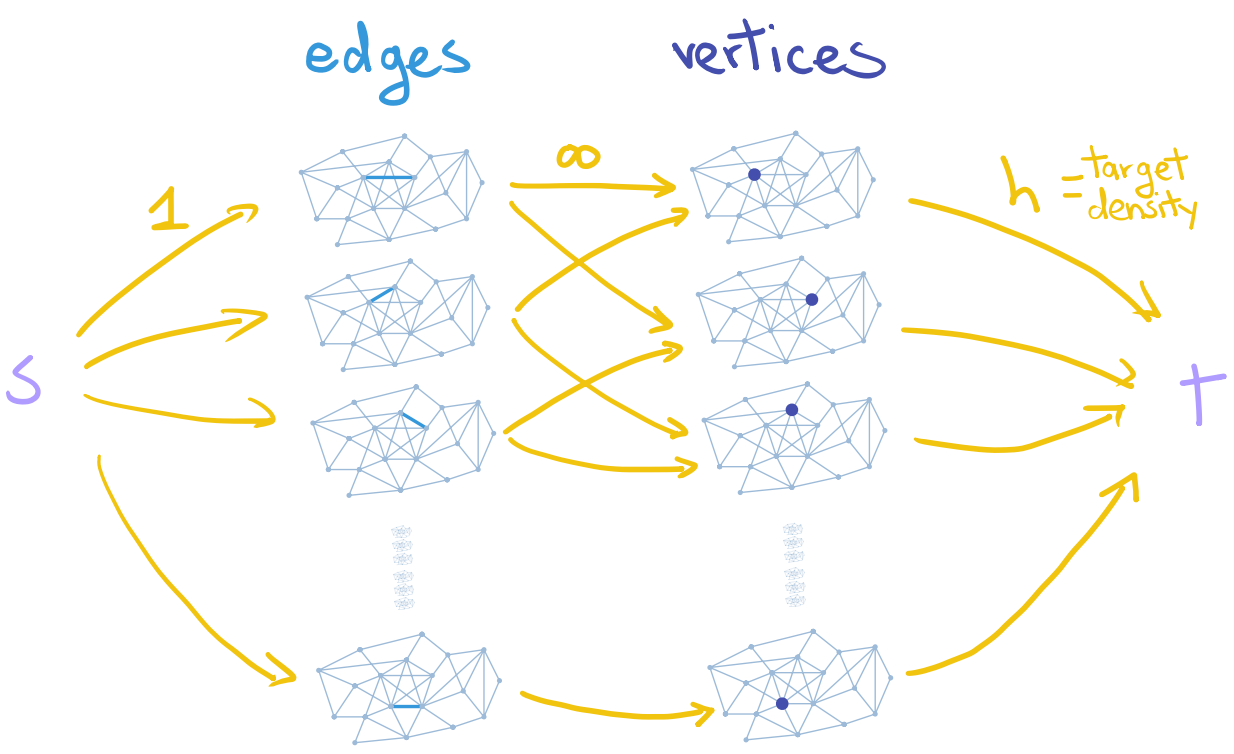
"density" = $\frac{\#\text{ edges}}{\#\text{ vertices}}$

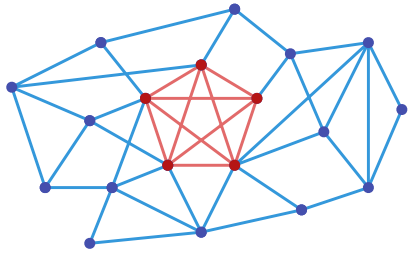


Decision version: Given $h>0$, is there a subgraph w/ density $>h$?


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

Densest Subgraph

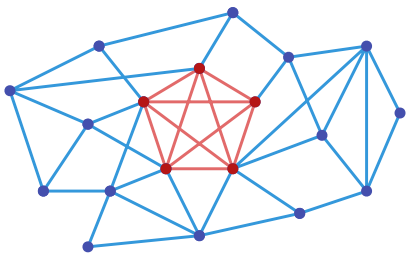


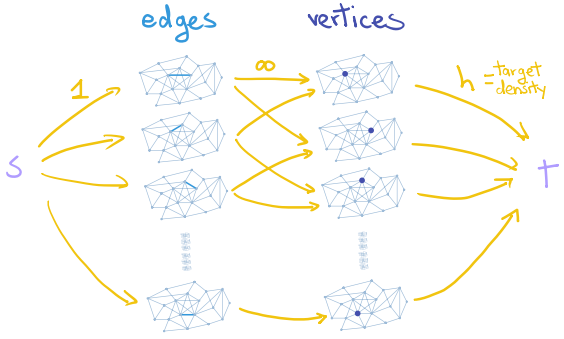


claim: if max-flow = m, then all subgraphs have density ≤ h

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

claim: if max-flow = m, then all subgraphs have density ≤ h





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