Parameterized Algorithms

Exercise 3 – Sheet 2, Sheet 3, Kernelization

Elly, Jean-Pierre, Wendy

Sheet 2 – Weighted Vertex Cover

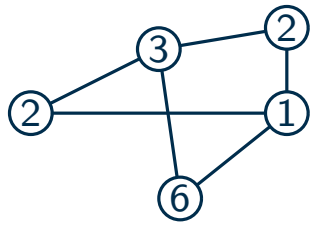
- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

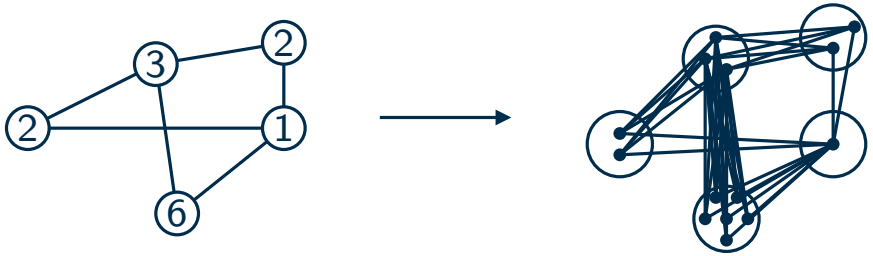
a) constant weights



Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

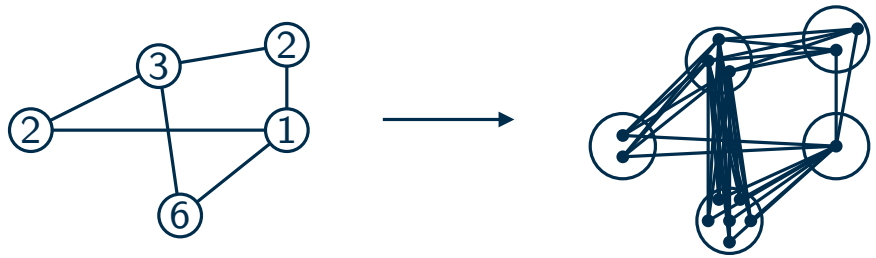
a) constant weights



Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights

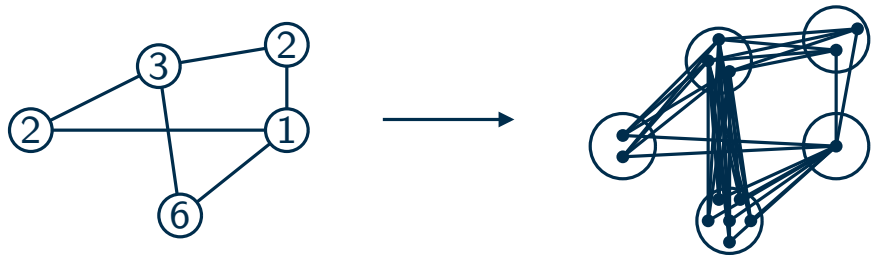


weight $\leq k$ $\iff$ size $\leq k$

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



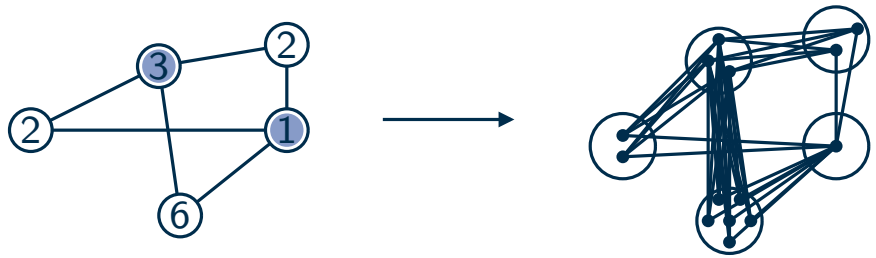
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



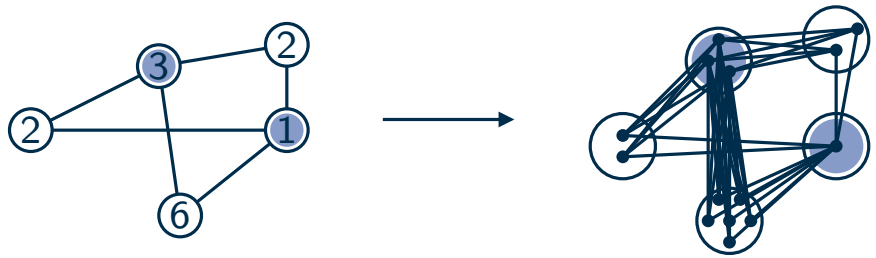
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



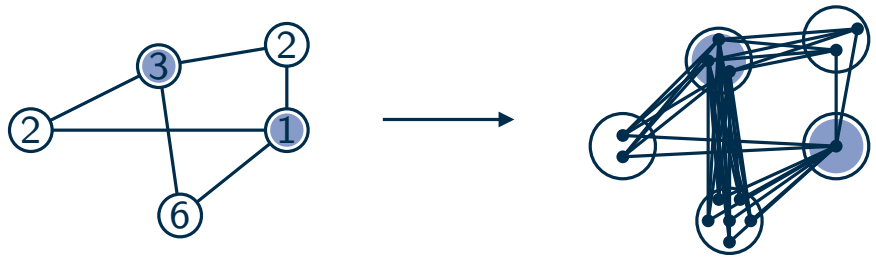
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



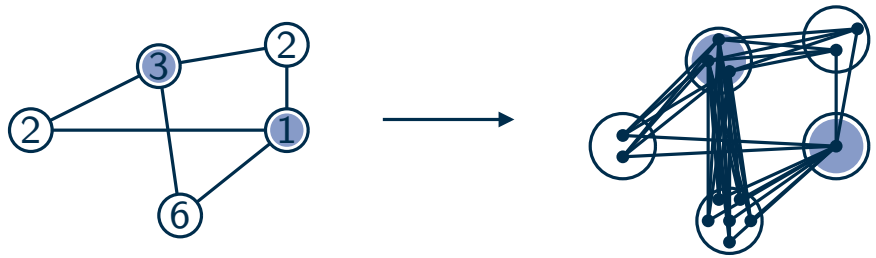
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



weight $\leq k$ $\iff$ size $\leq k$

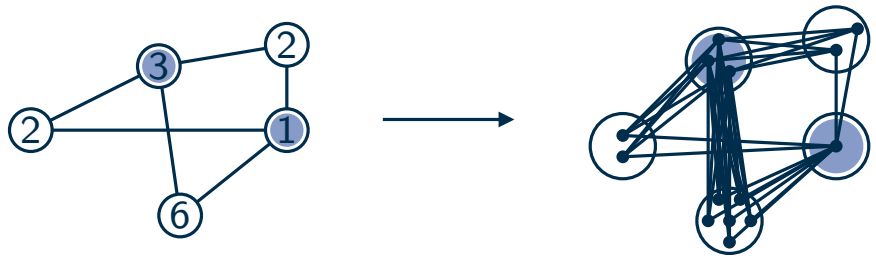
- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

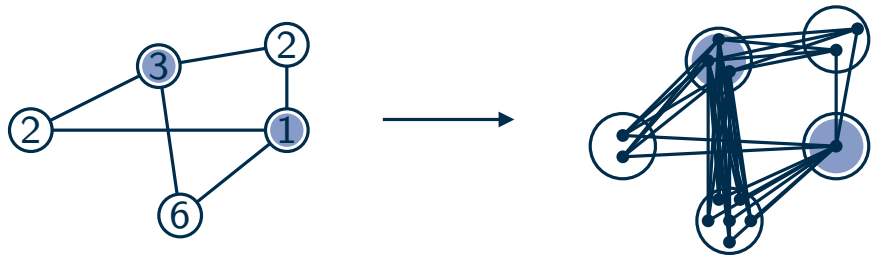
Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

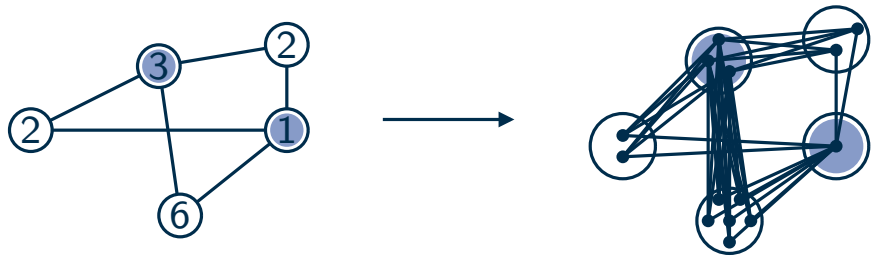
b) get rid of high weights

Is $O(1.342^k k(n+m))$ fast enough?

Sheet 2 – Weighted Vertex Cover

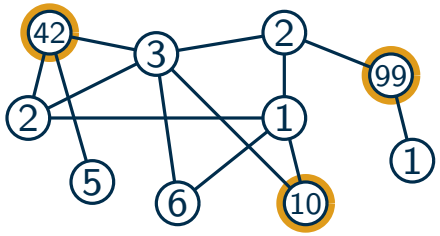
- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



weight $\leq k$ $\iff$ size $\leq k$

b) get rid of high weights



- choose corresponding vertex/group
- argue: choose whole group or none in group

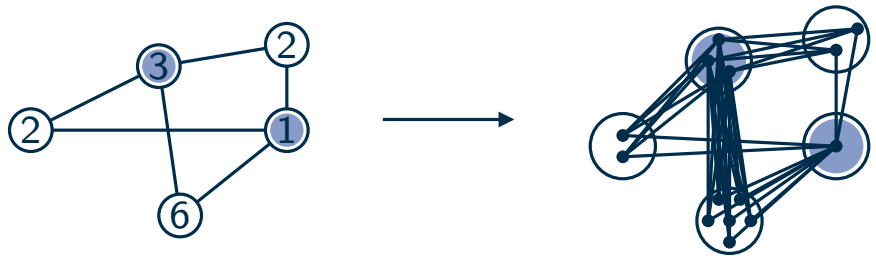
Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



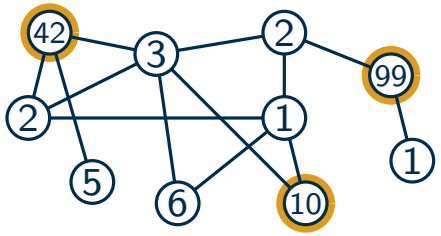
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

b) get rid of high weights

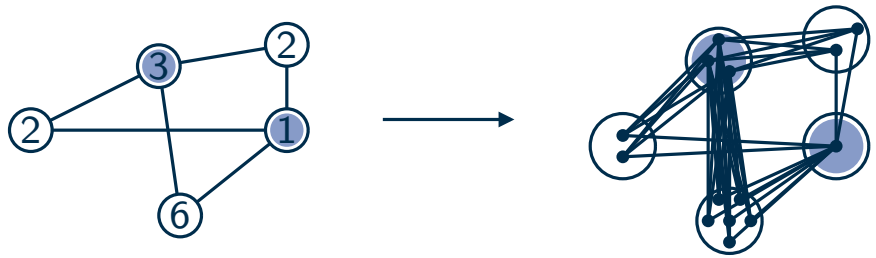


- branch on weights $\geq c$

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



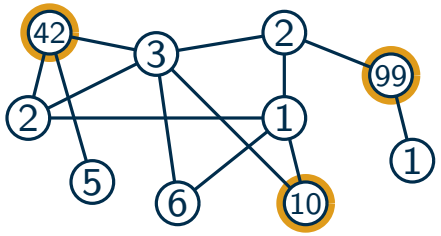
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

b) get rid of high weights

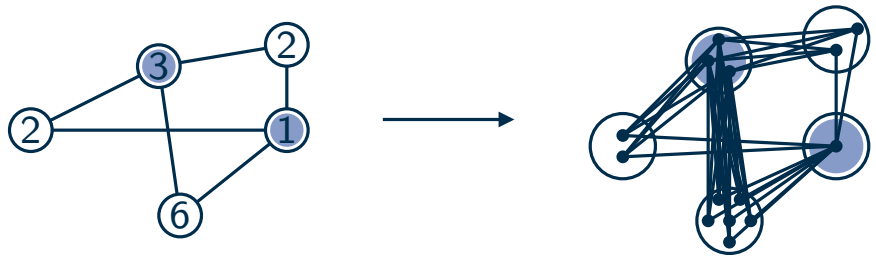


- branch on weights $\geq c$
- take either high weight or all neighbors $\Rightarrow$ branching vector $(c, 1)$

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



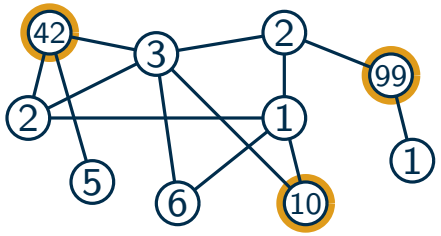
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

b) get rid of high weights



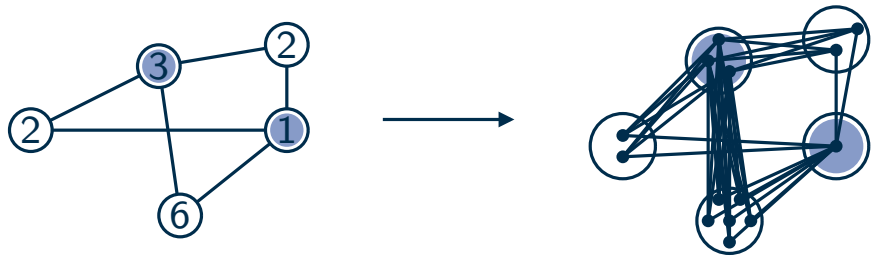
- branch on weights $\geq c$
- take either high weight or all neighbors $\Rightarrow$ branching vector $(c, 1)$

Running time: $O(1.2852^k(n+m))$ for $c = 6$

Sheet 2 – Weighted Vertex Cover

- find WVC with weight $\leq k$
- goal: same running time as VC with size $\leq k$ in $O(1.342^k(n+m))$ time

a) constant weights



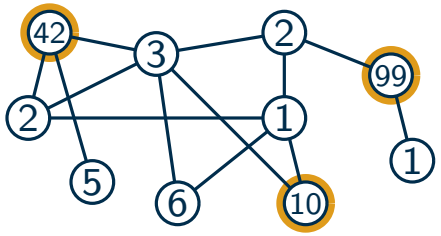
weight $\leq k$ $\iff$ size $\leq k$

- choose corresponding vertex/group
- argue: choose whole group or none in group

Running time: $O(1.342^k(n+m))$

Is $O(1.342^k k(n+m))$ fast enough?

b) get rid of high weights



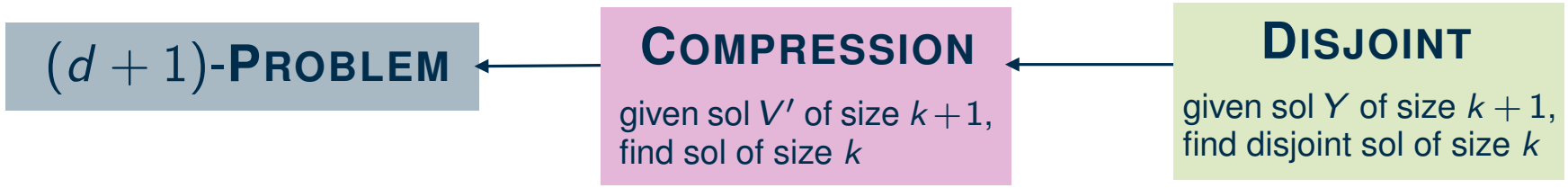
- branch on weights $\geq c$
- take either high weight or all neighbors
  $\Rightarrow$ branching vector $(c, 1)$

Running time: $O(1.2852^k(n+m))$ for $c = 6$

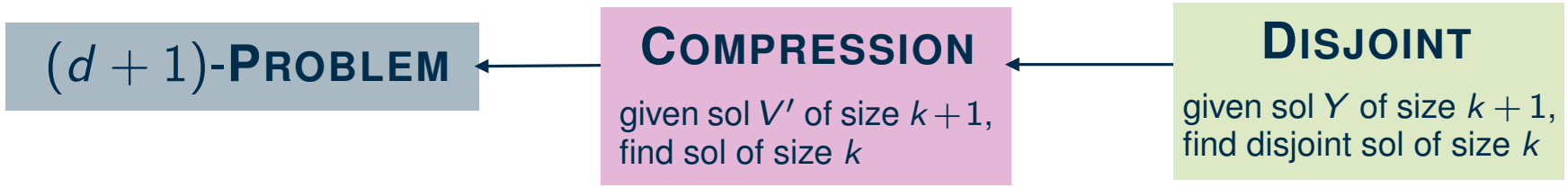
What if base for VC improves?

Sheet 2 – d-Hitting Set

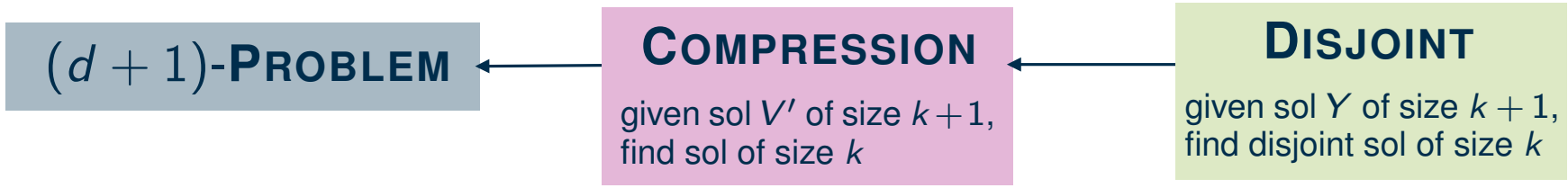
Sheet 2 – d-HITTING SET



Sheet 2 – d-Hitting Set

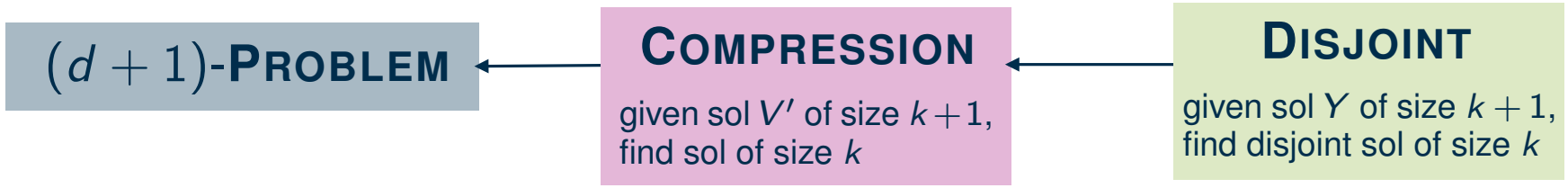


Sheet 2 – d-HITTING SET



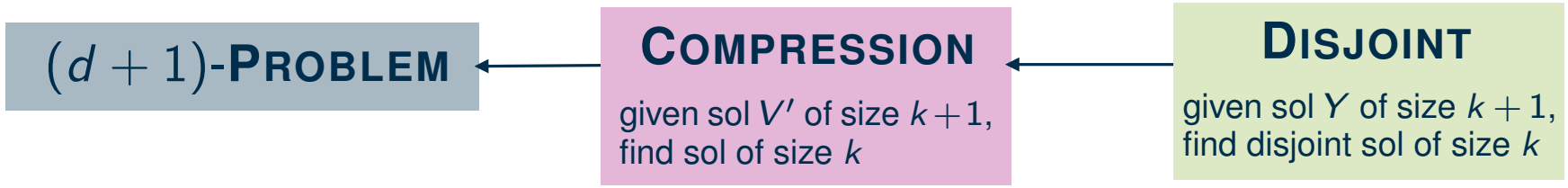
- a solution hits every set, has size $k+1$

Sheet 2 – d-HITTING SET



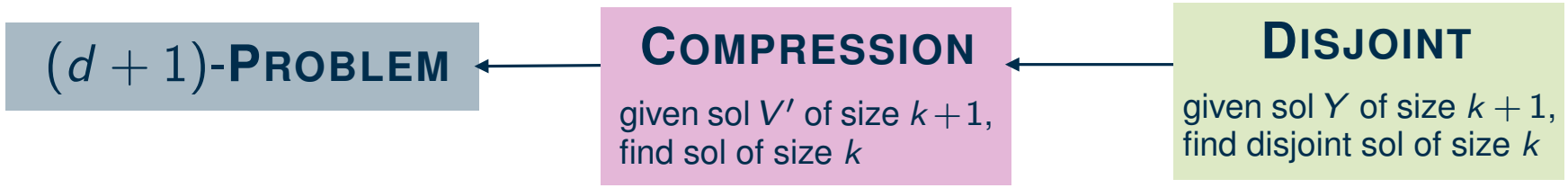
- a solution hits every set, has size $k+1$

Sheet 2 – d-HITTING SET



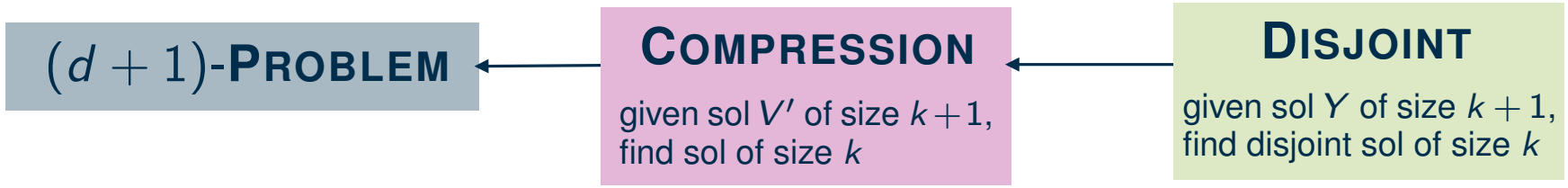
- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets

Sheet 2 – d-Hitting Set



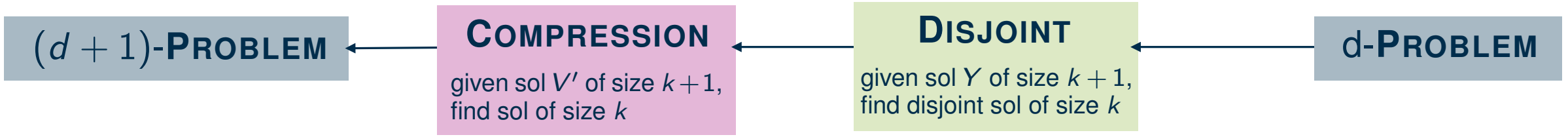
- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets

Sheet 2 – d-Hitting Set



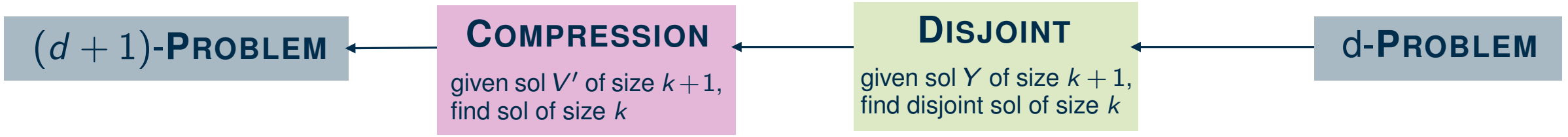
- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets

Sheet 2 – d-HITTING SET



- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets
- $\Rightarrow$ d-HITTING SET with parameter k

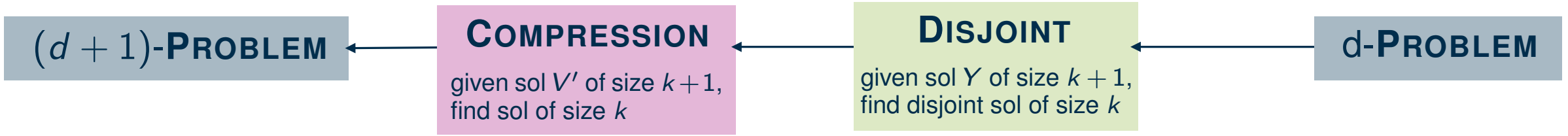
Sheet 2 – d-HITTING SET



- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets
- $\Rightarrow$ d-HITTING SET with parameter k

- 2-HITTING SET = VERTEX COVER, solvable in $O((2-0.658)^k n^c)$

Sheet 2 – d-HITTING SET

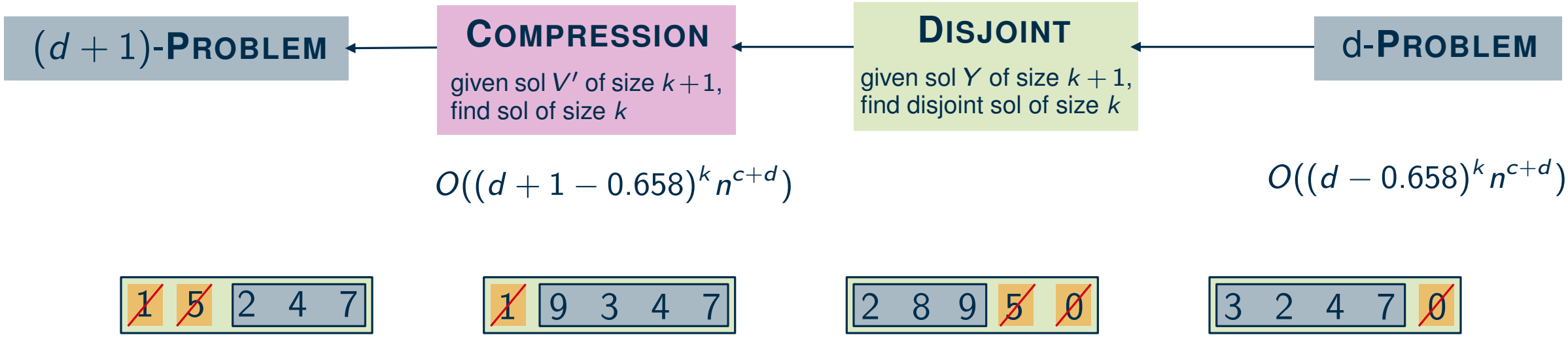


$O((d-0.658)^k n^{c+d})$

- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets
- $\Rightarrow$ d-HITTING SET with parameter k

- 2-HITTING SET = VERTEX COVER, solvable in $O((2-0.658)^k n^c)$

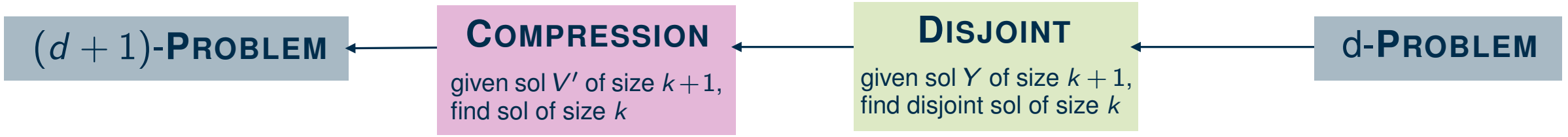
Sheet 2 – d-HITTING SET



- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets
- $\Rightarrow$ d-HITTING SET with parameter k

- 2-HITTING SET = VERTEX COVER, solvable in $O((2-0.658)^k n^c)$

Sheet 2 – d-HITTING SET



$O((d+1-0.658)^k n^{c+d+1})$ $O((d+1-0.658)^k n^{c+d})$ $O((d-0.658)^k n^{c+d})$

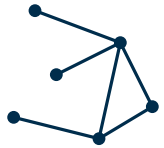
- a solution hits every set, has size $k+1$
- disjoint solution: find solution after removing Y from sets
- $\Rightarrow$ d-HITTING SET with parameter k

- 2-HITTING SET = VERTEX COVER, solvable in $O((2-0.658)^k n^c)$

Sheet 3

Odd Subgraph

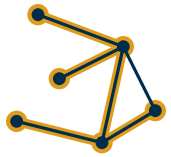
Find subgraph with exactly k edges where every vertex has odd degree.



Sheet 3

Odd Subgraph

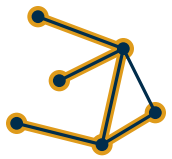
Find subgraph with exactly k edges where every vertex has odd degree.



Sheet 3

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.

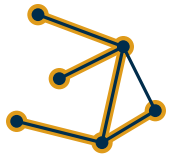


- $\geq k \cdot (2k - 1)$ edges, max deg $\leq k$?

Sheet 3

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.

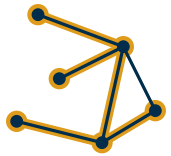


- $\geq k \cdot (2k - 1)$ edges, max deg $\leq k$?

Sheet 3

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.



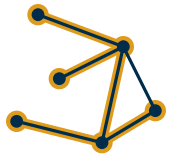
- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?

- $< k \cdot (2k-1)$ edges?

Sheet 3

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.

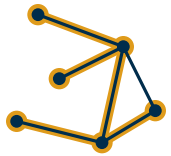


- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Sheet 3

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.



- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

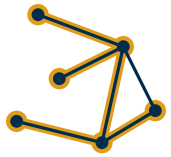
Edge Clique Cover

Cover all edges with k cliques.

Sheet 3

Odd Subgraph

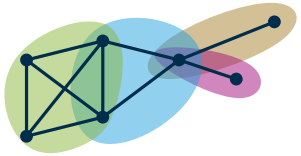
Find subgraph with exactly k edges where every vertex has odd degree.



- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

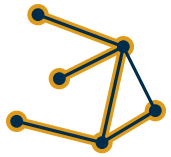
Edge Clique Cover

Cover all edges with k cliques.



Sheet 3

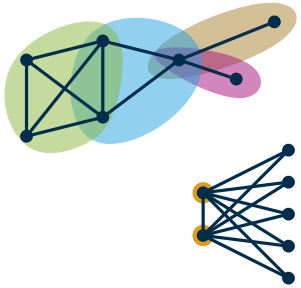
Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

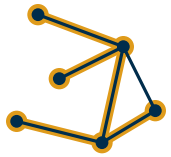


Cover all edges with k cliques.

- get rid of twins

Sheet 3

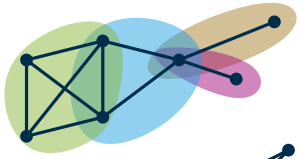
Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover



Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Sheet 3

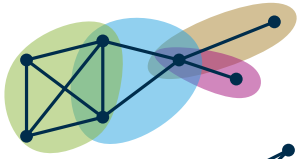
Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph

Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover



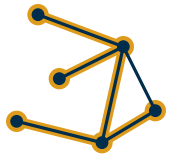
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

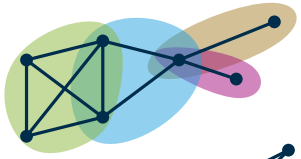
ODD SUBGRAPH



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

EDGE CLIQUE COVER



Cover all edges with k cliques.

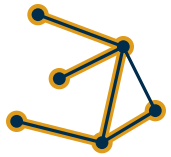
- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

VERTEX COVER in Bipartite Graphs

Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

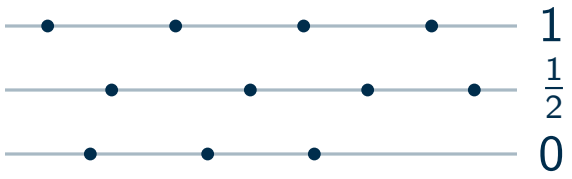
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Vertex Cover in Bipartite Graphs

a) LP for VC has an optimal integer solution on bipartite graphs

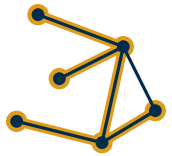
- half-integral solution for general graphs



Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

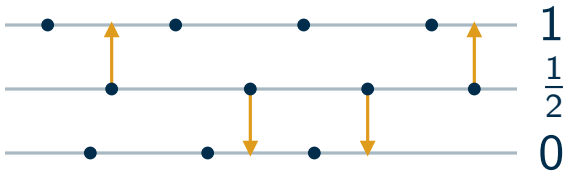
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Vertex Cover in Bipartite Graphs

a) LP for VC has an optimal integer solution on bipartite graphs

- half-integral solution for general graphs

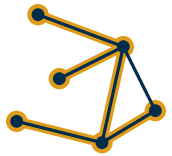


- assign integers to $\frac{1}{2}$ vertices (how?)

Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

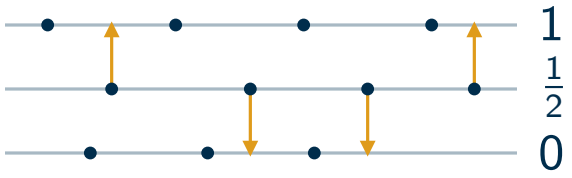
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Vertex Cover in Bipartite Graphs

a) LP for VC has an optimal integer solution on bipartite graphs

- half-integral solution for general graphs



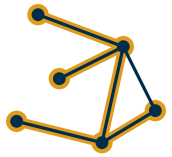
- assign integers to $\frac{1}{2}$ vertices (how?)

b) $|\min \text{VC}| = |\max \text{matching}|$

Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

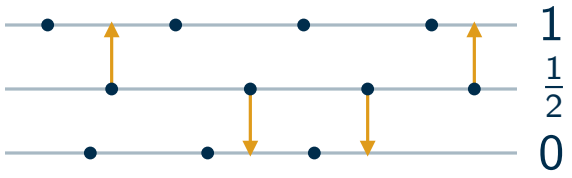
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Vertex Cover in Bipartite Graphs

a) LP for VC has an optimal integer solution on bipartite graphs

- half-integral solution for general graphs



- assign integers to $\frac{1}{2}$ vertices (how?)

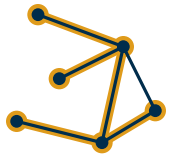
b) $|\min \text{VC}| = |\max \text{matching}|$

- use duality of VC and matching (next lecture)

Sheet 3

Problem 3 changed! (Old Problem 3 moved to Sheet 4)

Odd Subgraph



Find subgraph with exactly k edges where every vertex has odd degree.

- $\geq k \cdot (2k-1)$ edges, max deg $\leq k$?
- $< k \cdot (2k-1)$ edges?
- max deg $> k$? k even/odd?

Edge Clique Cover

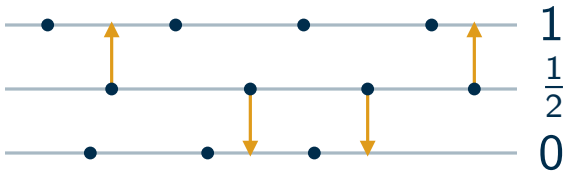
Cover all edges with k cliques.

- get rid of twins
- # vertices if no twins and all edges can be covered with k cliques?

Vertex Cover in Bipartite Graphs

a) LP for VC has an optimal integer solution on bipartite graphs

- half-integral solution for general graphs



- assign integers to $\frac{1}{2}$ vertices (how?)

b) $|\min \text{VC}| = |\max \text{matching}|$

- use duality of VC and matching (next lecture)

Closest String (Implementation)

- use a bounded search tree
- find rules online, e.g. see book

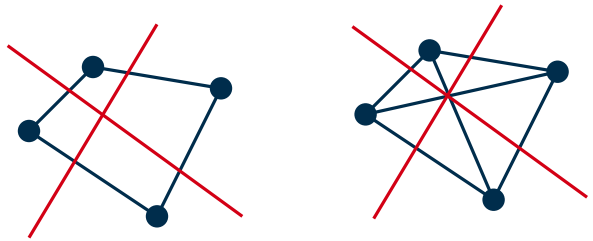
Kernelization

Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

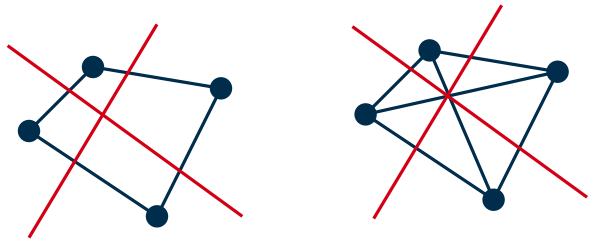
Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



Kernelization

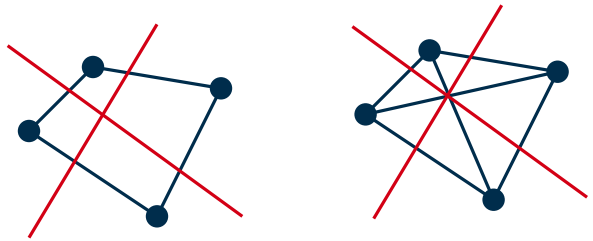
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



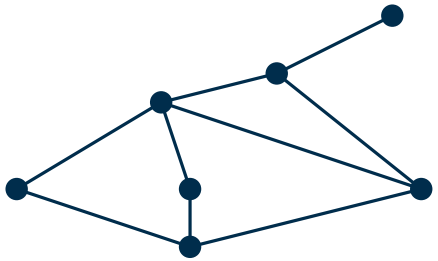
CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

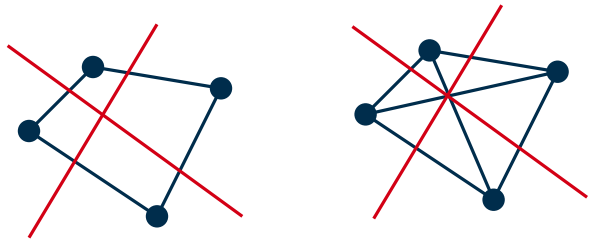


CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

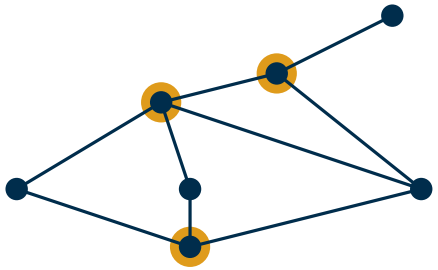


Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

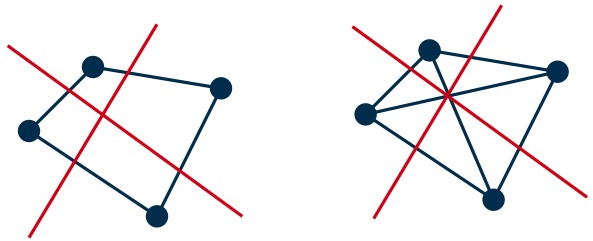


CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

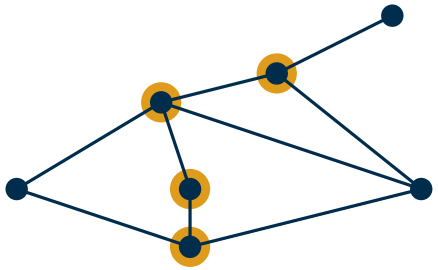


Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

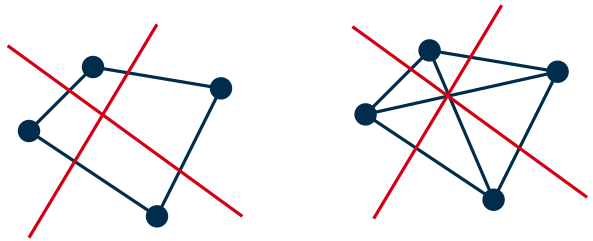


CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

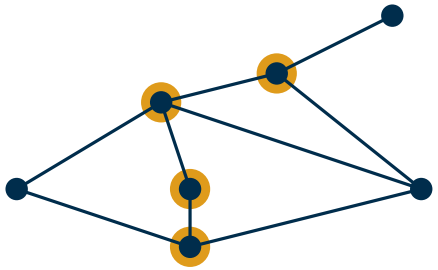


Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



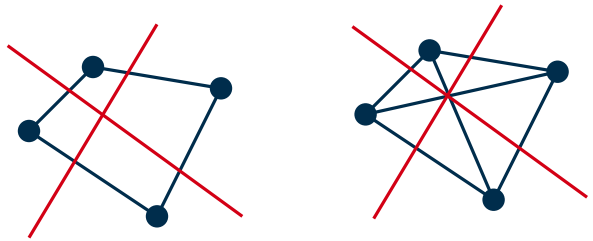
CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected



Why can we not remove vertices with degree $> k$?

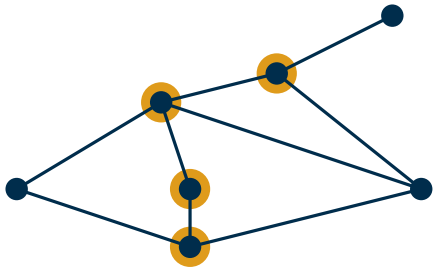
Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

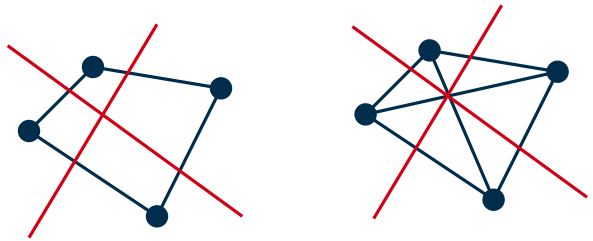
Kernel size $2^k + O(k^2)$



Why can we not remove vertices with degree $> k$?

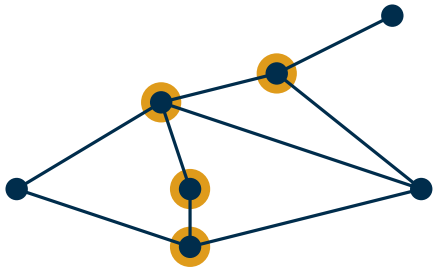
Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$

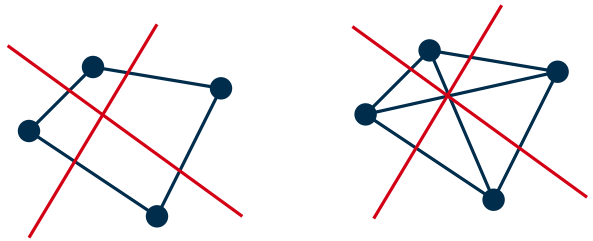


Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

Kernelization

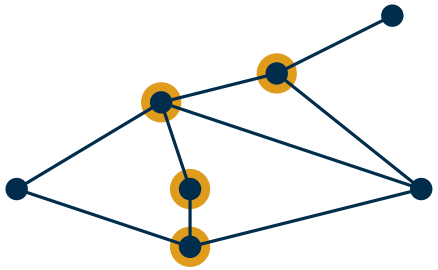
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



1. Bound kernel for graphs with max degree $f(k)$

CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$

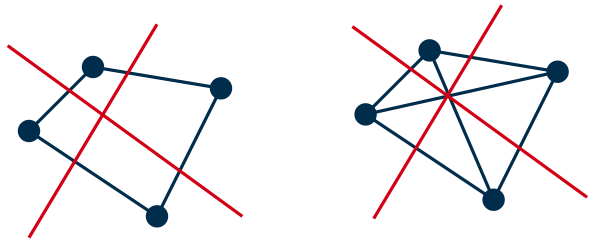


Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

Kernelization

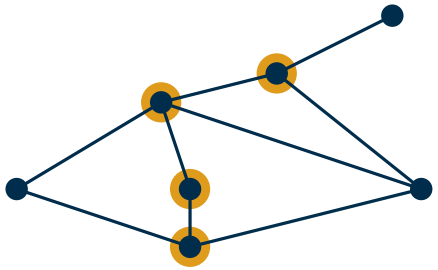
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



1. Bound kernel for graphs with max degree $f(k)$
2. What if G has high max degree?

CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$

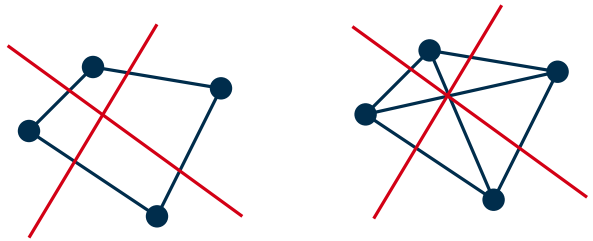


Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

Kernelization

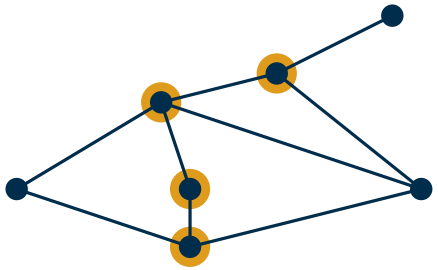
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



1. Bound kernel for graphs with max degree $f(k)$
2. What if G has high max degree?

CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



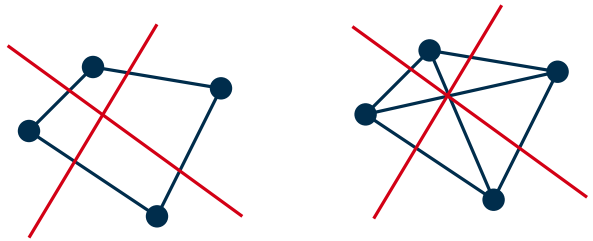
1. Consider high-degree vertices with degree $> k$ (H) and low-degree vertices (L) separately.

Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

Kernelization

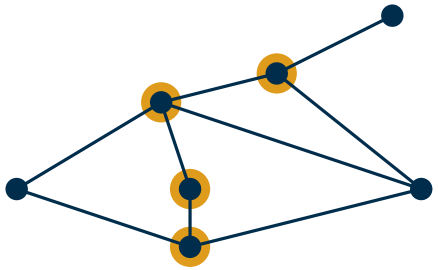
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



1. Bound kernel for graphs with max degree $f(k)$
2. What if G has high max degree?

CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



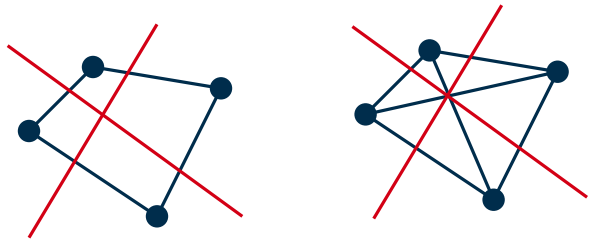
1. Consider high-degree vertices with degree $> k$ (H) and low-degree vertices (L) separately.
2. How many edges are in $L \times L$?

Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

Kernelization

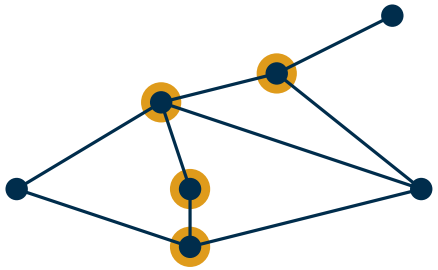
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



1. Bound kernel for graphs with max degree $f(k)$
2. What if G has high max degree?

CONNECTED VERTEX COVER
given: graph G, parameter k
Find a vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



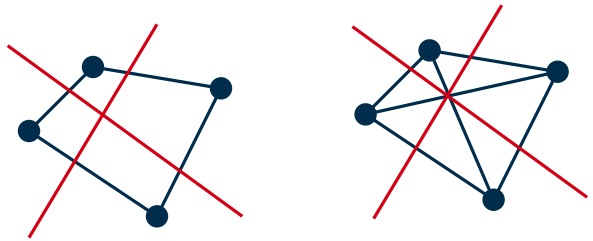
1. Consider high-degree vertices with degree $> k$ (H) and low-degree vertices (L) separately.
2. How many edges are in $L \times L$?
3. Reduce number of vertices in L that are only adjacent to vertices in H. Why can we not just remove all these vertices? Make sure that H still needs to be selected.

Why can we not remove vertices with degree $> k$?

Poly kernel for C_4-free? For $K_{d,d}$-free? For planar?

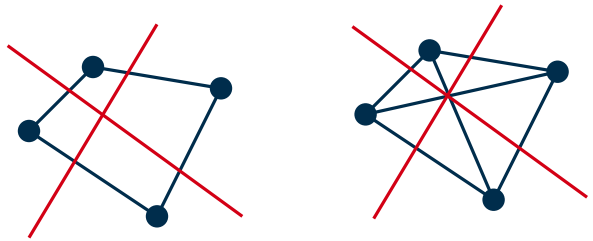
Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

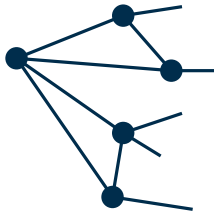


Kernelization

INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$

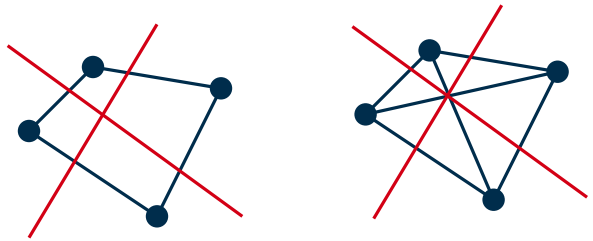


max degree $\geq 2k$



Kernelization

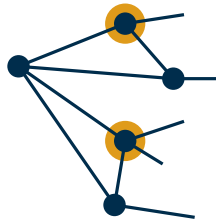
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

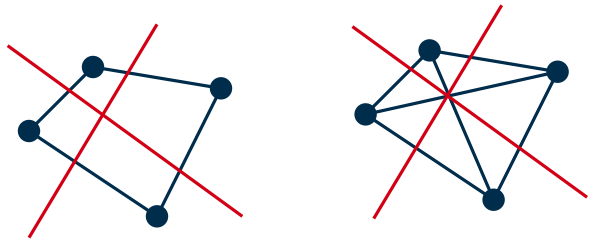
- at least half of the neighbors can be chosen

Why?



Kernelization

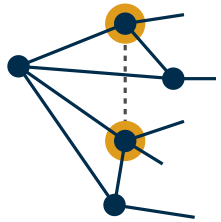
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

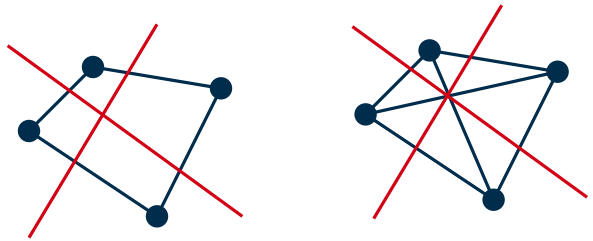
- at least half of the neighbors can be chosen

Why?



Kernelization

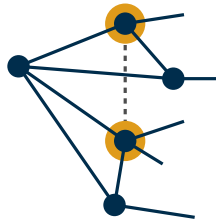
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

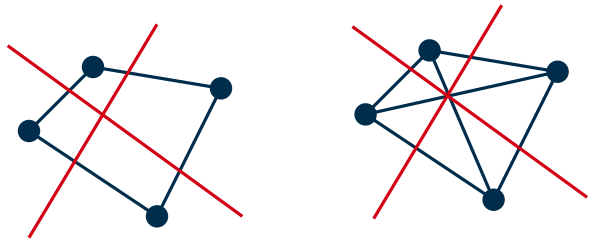
- at least half of the neighbors can be chosen
- yes-instance

Why?



Kernelization

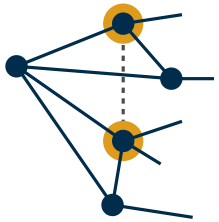
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



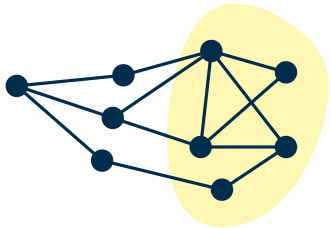
max degree $\geq 2k$

- at least half of the neighbors can be chosen
- yes-instance

Why?

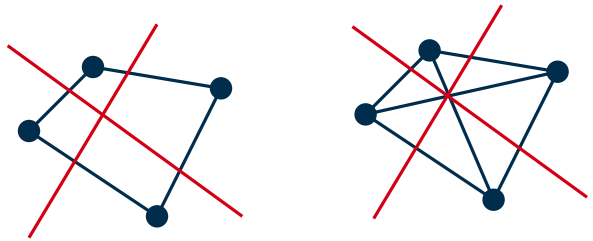


max degree $< 2k$



Kernelization

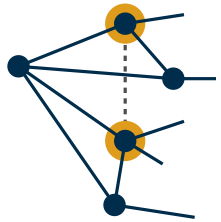
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

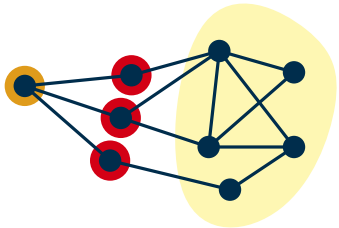
- at least half of the neighbors can be chosen
- yes-instance

Why?



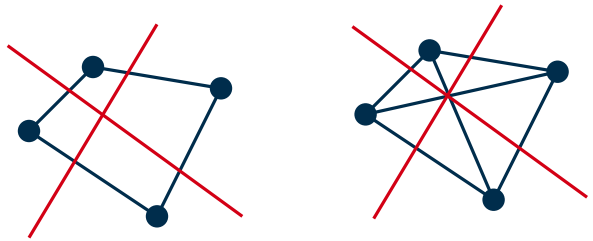
max degree $< 2k$

- choosing a vertex removes at most $2k$ vertices



Kernelization

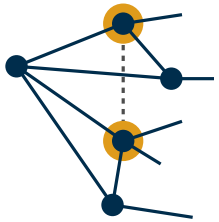
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

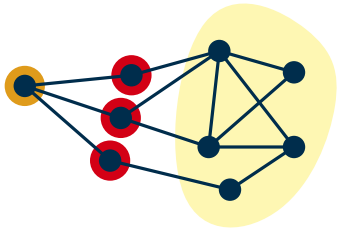
- at least half of the neighbors can be chosen
- yes-instance

Why?



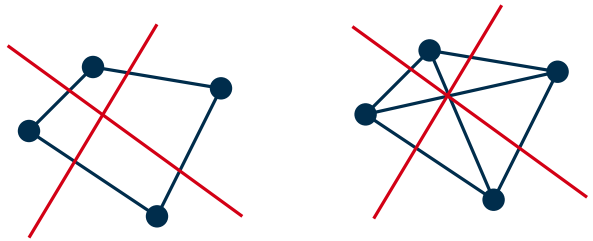
max degree $< 2k$

- choosing a vertex removes at most $2k$ vertices
- at least $2k^2$ vertices $\Rightarrow$ yes-instance



Kernelization

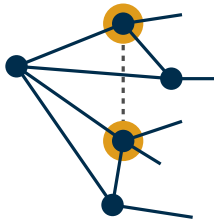
INDEPENDENT SET (Graphs without C_4)
given: graph G with no C_4 as subgraph, parameter k
Find an independent set A with $|A| = k$



max degree $\geq 2k$

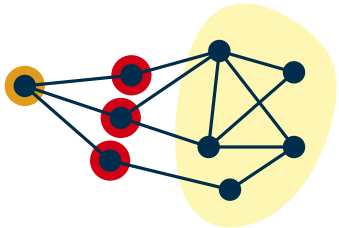
- at least half of the neighbors can be chosen
- yes-instance

Why?



max degree $< 2k$

- choosing a vertex removes at most $2k$ vertices
- at least $2k^2$ vertices $\Rightarrow$ yes-instance



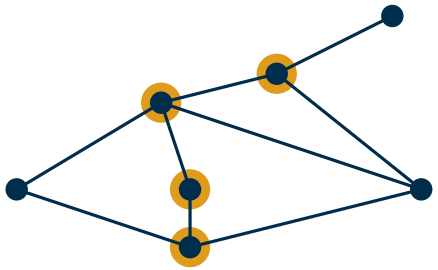
$\Longrightarrow$ kernel with $2k^2$ vertices

Kernelization

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

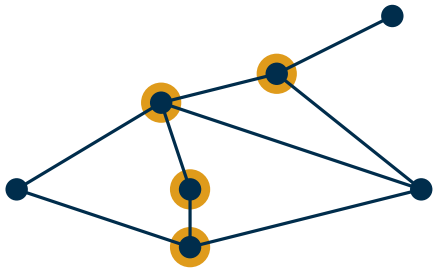
Kernel size $2^k + O(k^2)$

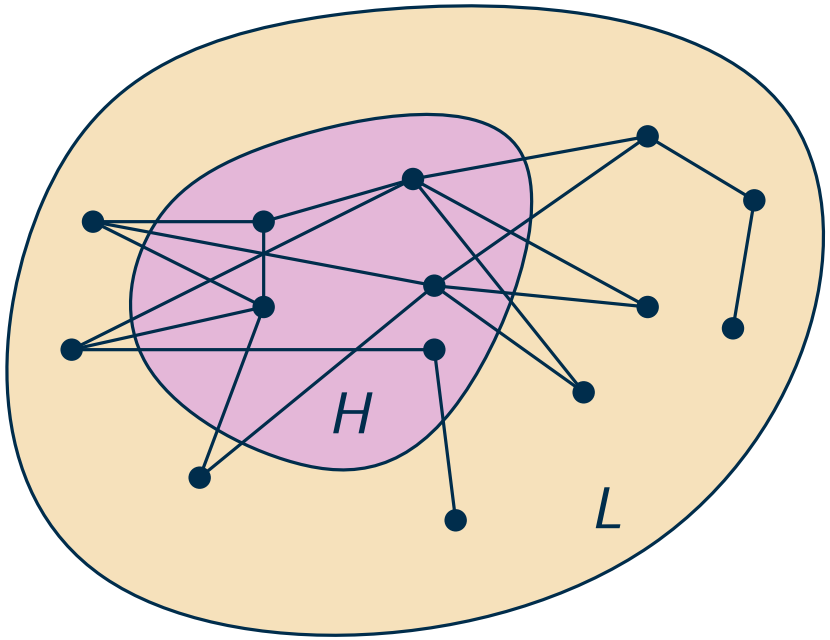


Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$

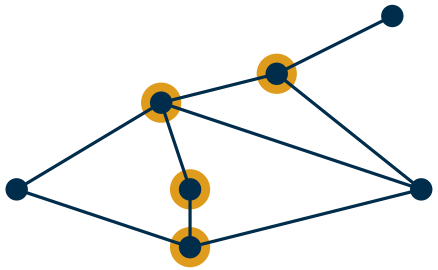


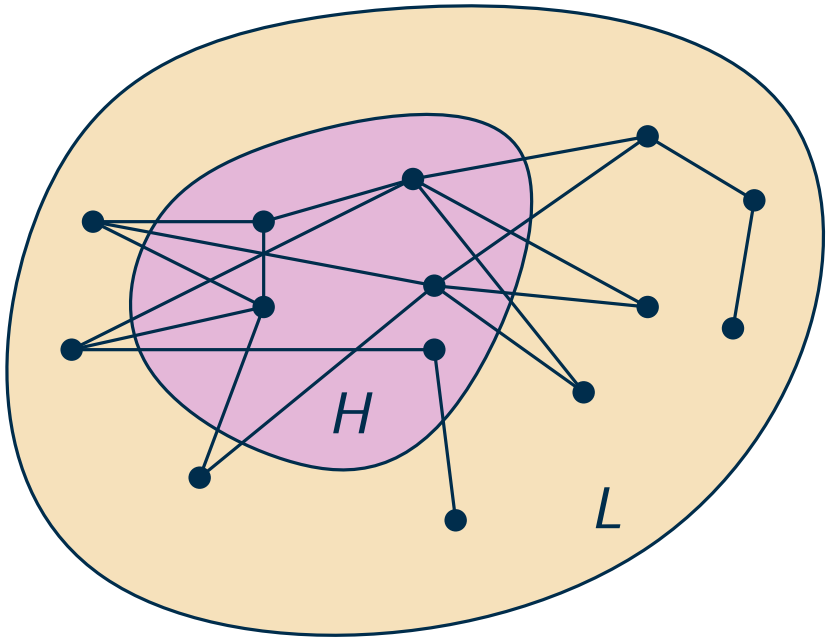


Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



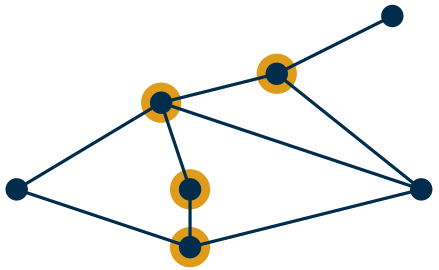


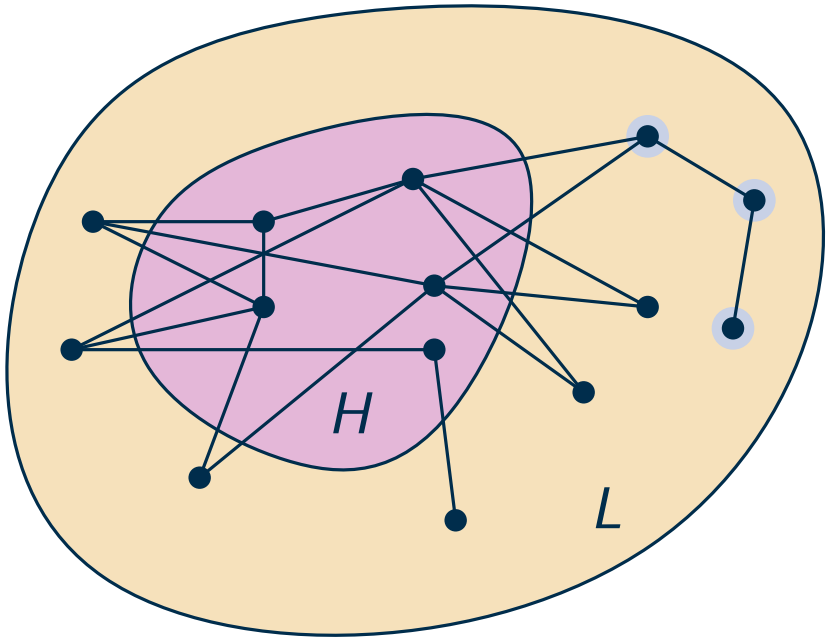
- at most k vertices in H

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$





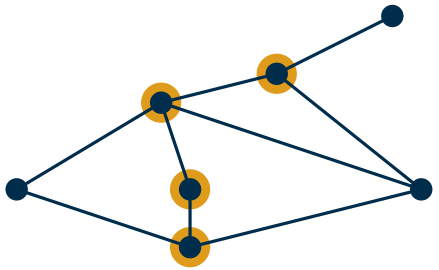
- at most k vertices in H
- at most k^2 vertices in L that have a neighbor in L

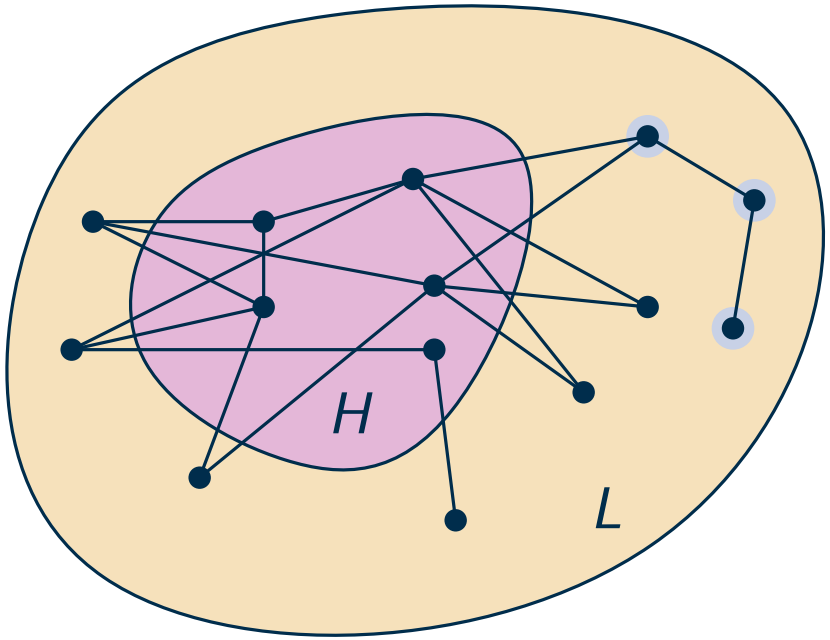
Why?

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



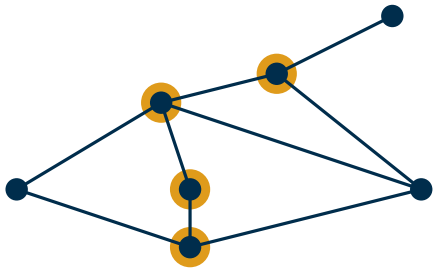


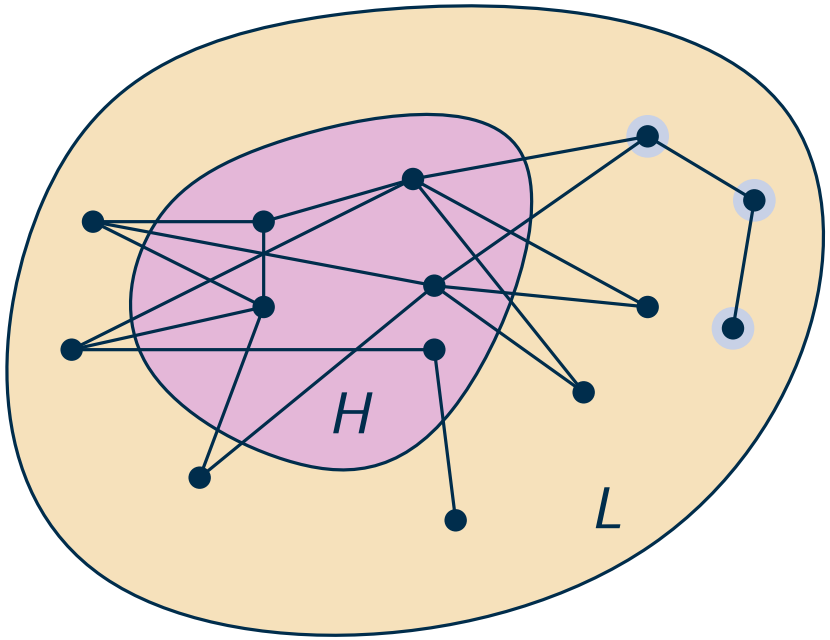
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



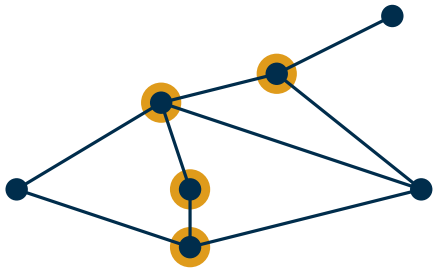


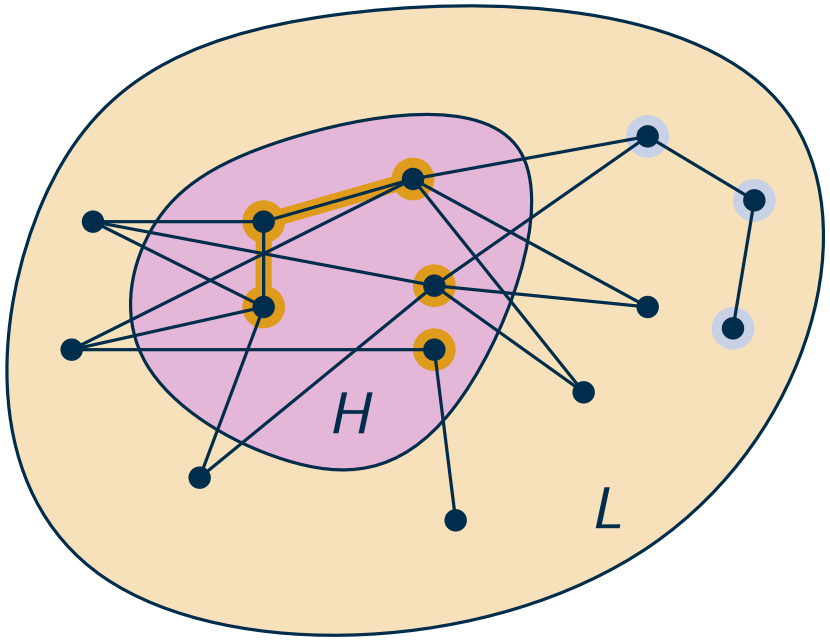
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



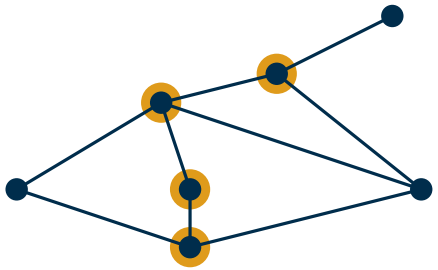


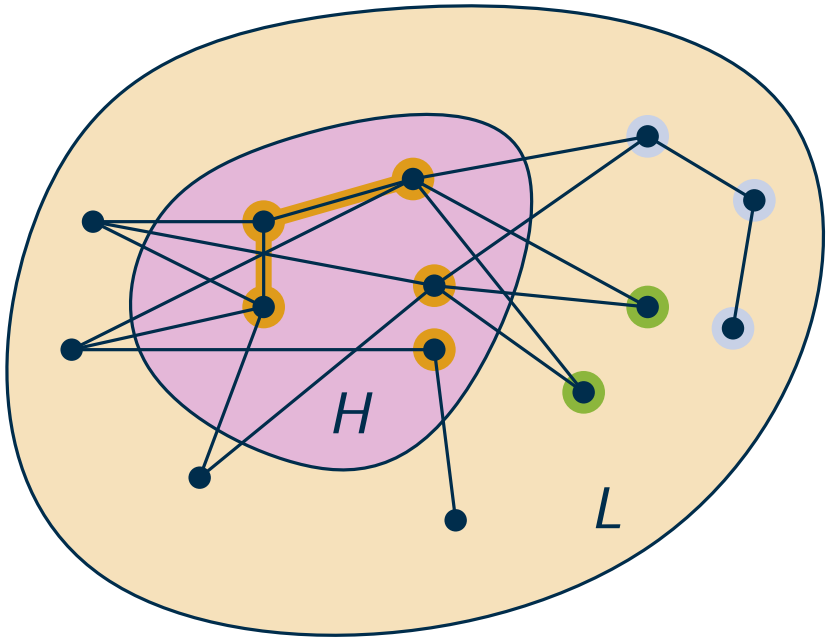
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H
  - edges are covered, but vertices might be needed to make solution connected

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



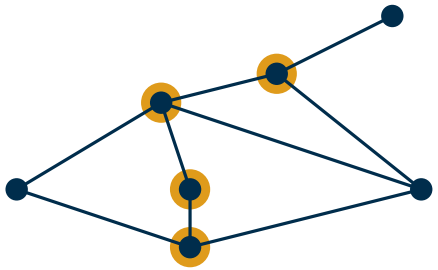


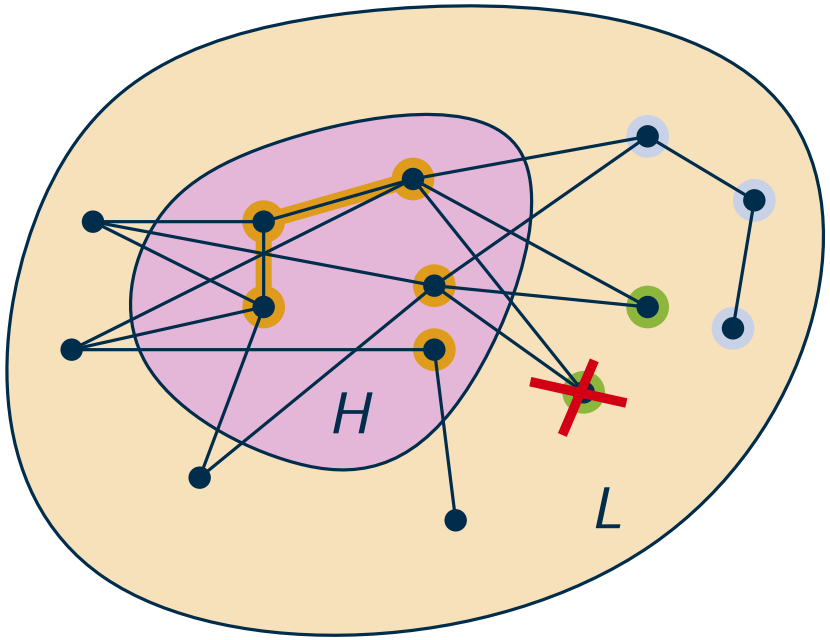
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H
  - edges are covered, but vertices might be needed to make solution connected
  - same neighborhood in $H \Rightarrow$ only need one of them

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$



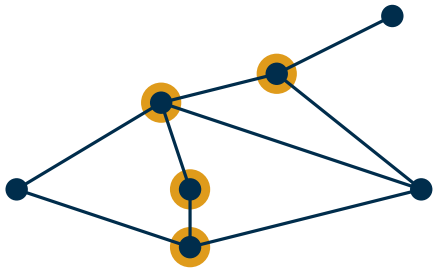


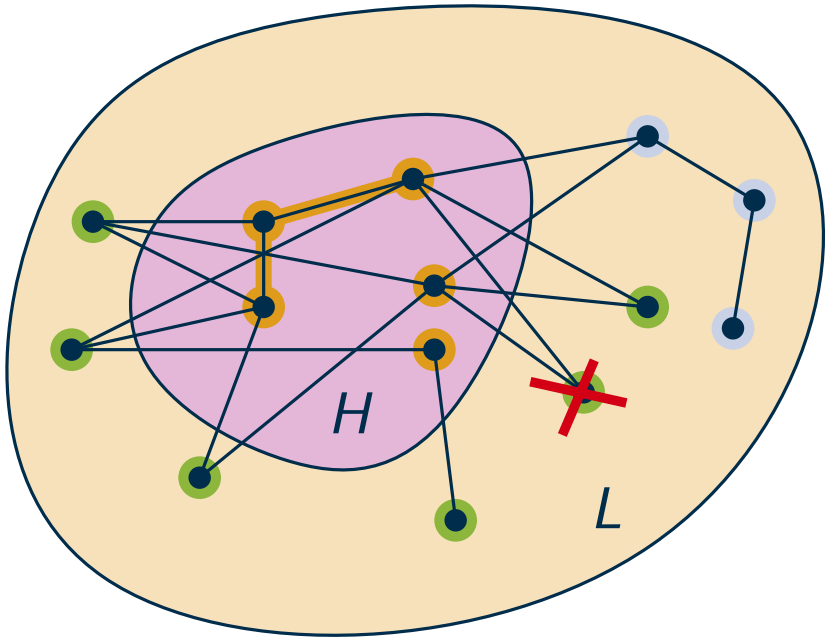
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H
  - edges are covered, but vertices might be needed to make solution connected
  - same neighborhood in $H \Rightarrow$ only need one of them

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$





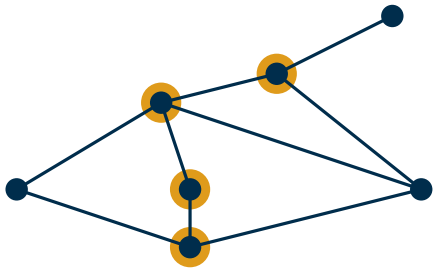
- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H
  - edges are covered, but vertices might be needed to make solution connected
  - same neighborhood in $H \Rightarrow$ only need one of them

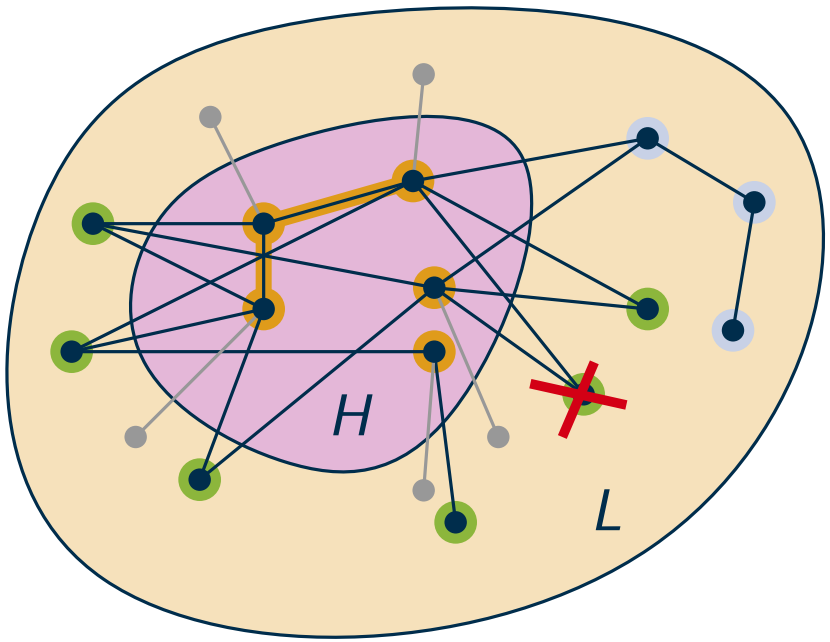
$\Rightarrow$ at most 2^k vertices in L with only neighbors in H

Kernelization

CONNECTED VERTEX COVER
given: graph G, parameter k
Find vertex cover A with $|A| = k$ such that $G[A]$ is connected

Kernel size $2^k + O(k^2)$





- at most k vertices in H — Why?
- at most k^2 vertices in L that have a neighbor in L — Why?
- vertices in L with only neighbors in H
  - edges are covered, but vertices might be needed to make solution connected
  - same neighborhood in $H \Rightarrow$ only need one of them

$\Rightarrow$ at most 2^k vertices in L with only neighbors in H

- ensure H is chosen: add k leaves (one per vertex in H)