Parameterized Algorithms

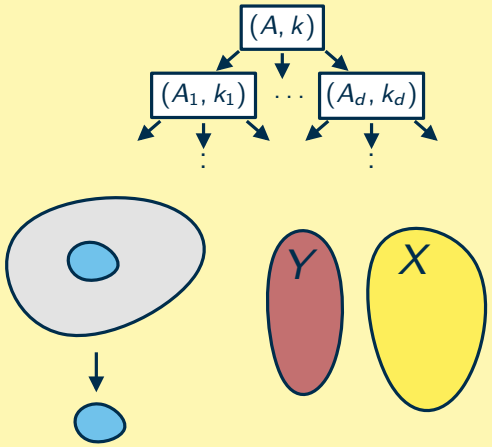
Iterative Compression: Feedback Vertex Set

Thomas Bläsius

Content

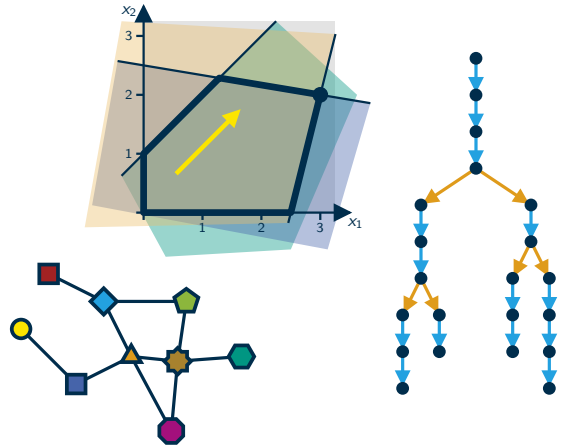
Basic toolbox

- bounded search trees
- kernelization
- iterative compression



Extended toolbox

- linear programs
- branch-and-reduce
- color coding

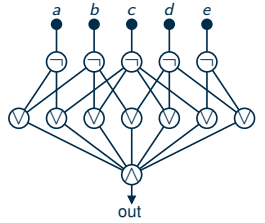


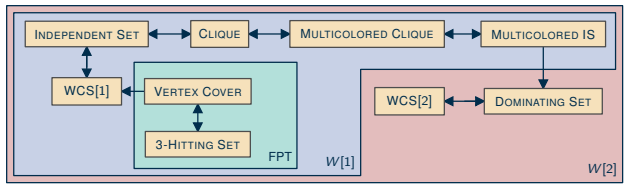
Tree width

- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

Lower bonds

- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH

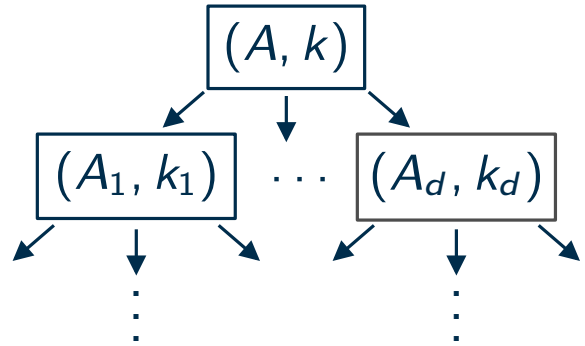




Recap: Basic FPT-techniques

Bounded search tree

- for instance (A, k) compute in time n^c instances $(A_1, k_1), \ldots, (A_d, k_d)$, such that: (A, k) yes instance $\Leftrightarrow$ (A_i, k_i) yes instance for an $i \in [1, d]$
- bound d by a function $f_1(k)$
- bound tree height by $f_2(k)$ (example: parameter shrinks in each step)
- $\Rightarrow$ FPT-algo with running time $O(f_1(k)^{f_2(k)} n^c)$



Kernelization

- successively apply safe reduction rules
- show: what remains is a core, whose size only depends on k

Iterative compression

- compression algo: solve the problem under the assumption of knowing a slightly larger solution
- grow the instance step by step and maintain a small solution via repeated compression

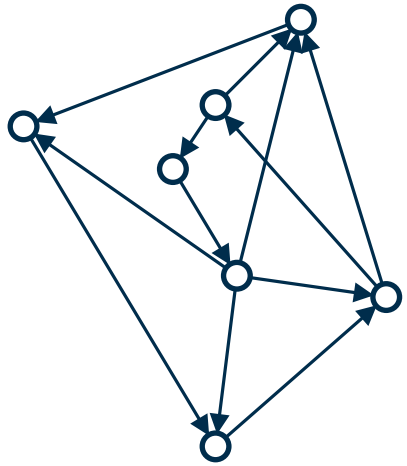
Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

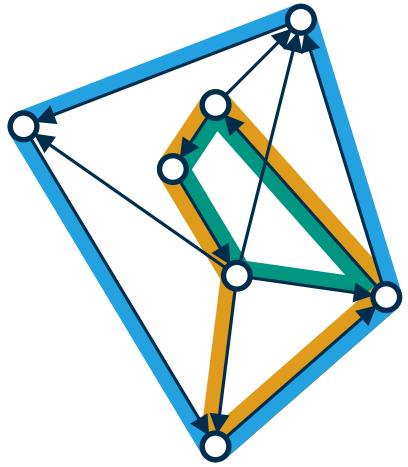


Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

- not acyclic



Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

- not acyclic
- all directed cycles include ●

Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

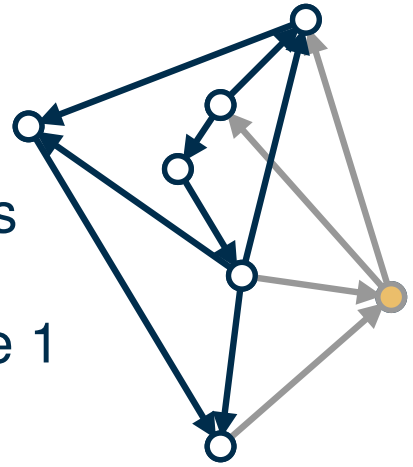
- not acyclic
- all directed cycles include ●
- {●} is FVS of size 1

Feedback Vertex Set and tournament graphs

Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

- not acyclic
- all directed cycles include ●
- {●} is FVS of size 1

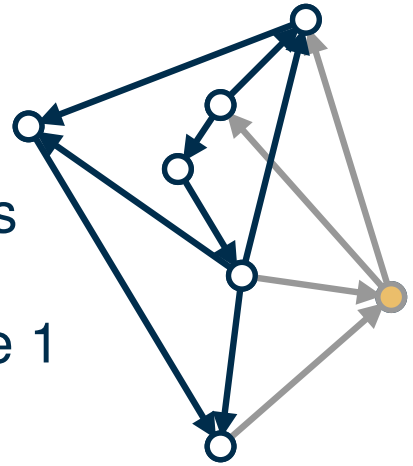


Feedback Vertex Set and tournament graphs

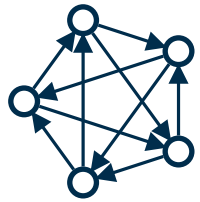
Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

- not acyclic
- all directed cycles include ●
- {●} is FVS of size 1



Definition
A directed graph $G = (V, E)$ is a **tournament graph** if for every two different $u, v \in V$ either $uv \in E$ or $vu \in E$.

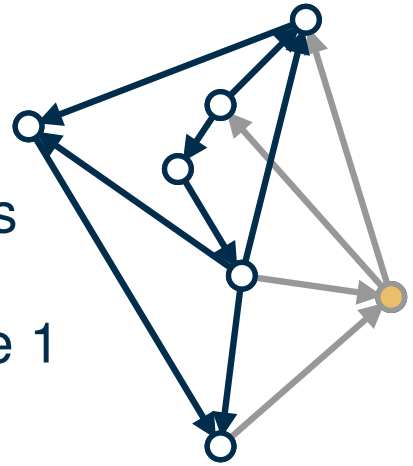


Feedback Vertex Set and tournament graphs

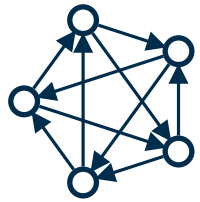
Problem: Feedback Vertex Set
Given a directed graph $G = (V, E)$ and a parameter k.
Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

- not acyclic
- all directed cycles include ●
- {●} is FVS of size 1



Definition
A directed graph $G = (V, E)$ is a **tournament graph** if for every two different $u, v \in V$ either $uv \in E$ or $vu \in E$.



Goal

- FPT algorithm for Feedback Vertex Set on tournament graphs
- use iterative compression

Compression problem

Problem: FEEDBACK VERTEX SET COMPRESSION
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

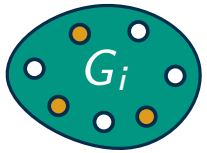
Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

FVS of size k in G_i

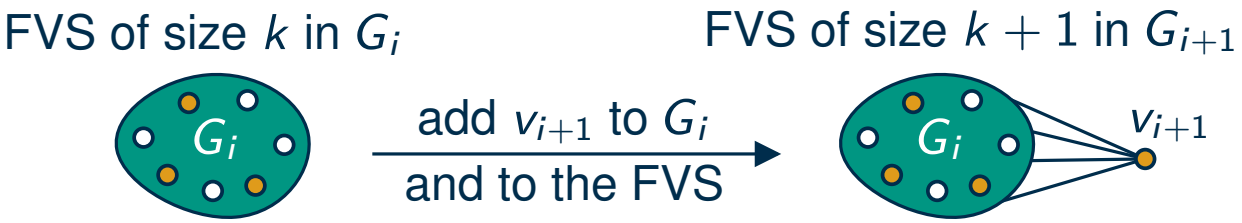


Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

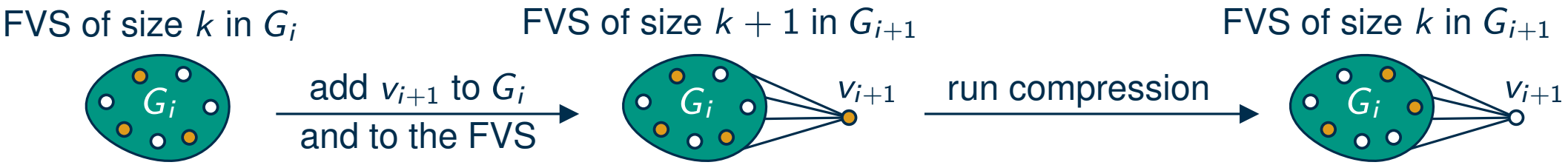


Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

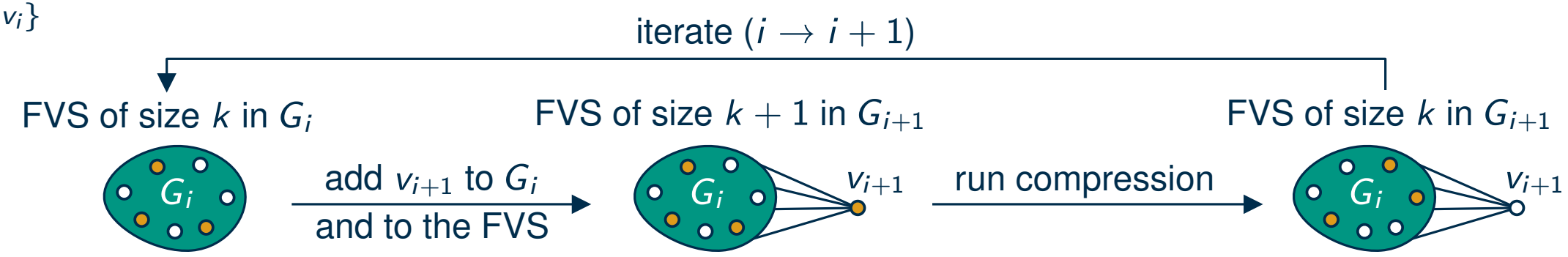


Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

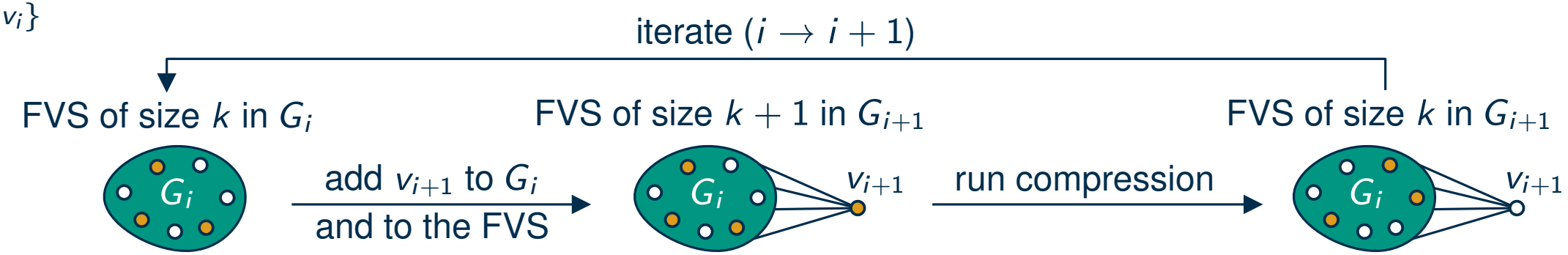


Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

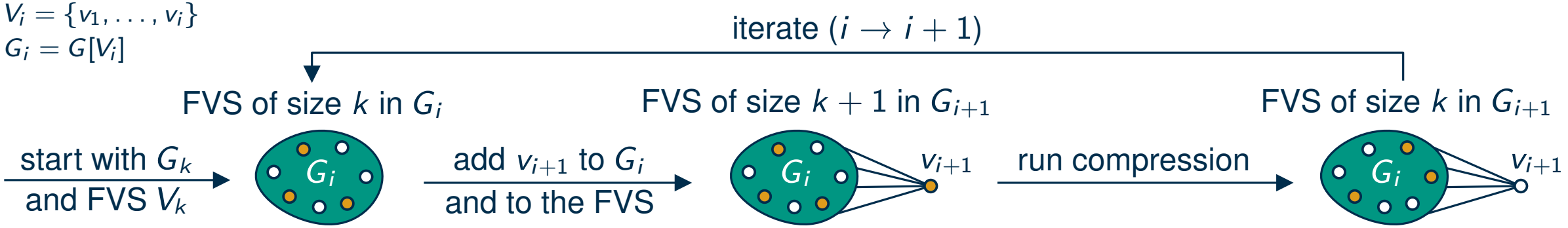


What is missing for a complete algorithm?

Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.



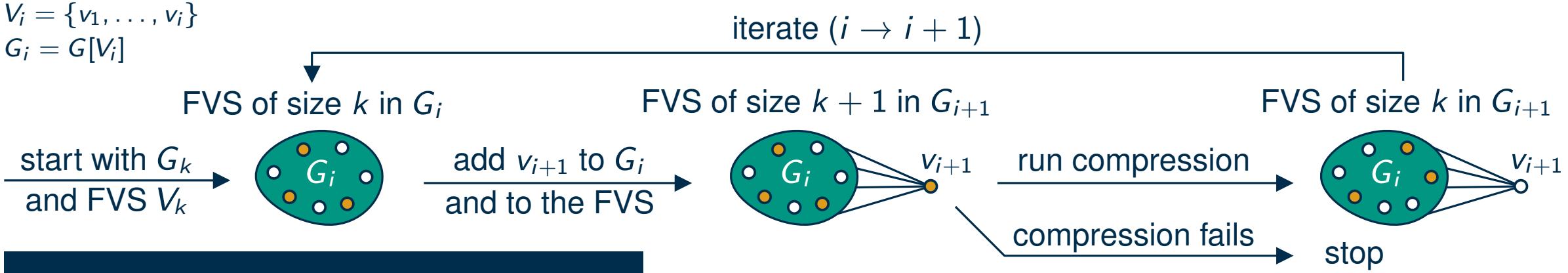
What is missing for a complete algorithm?

Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

$V_i = \{v_1, \ldots, v_i\}$
$G_i = G[V_i]$

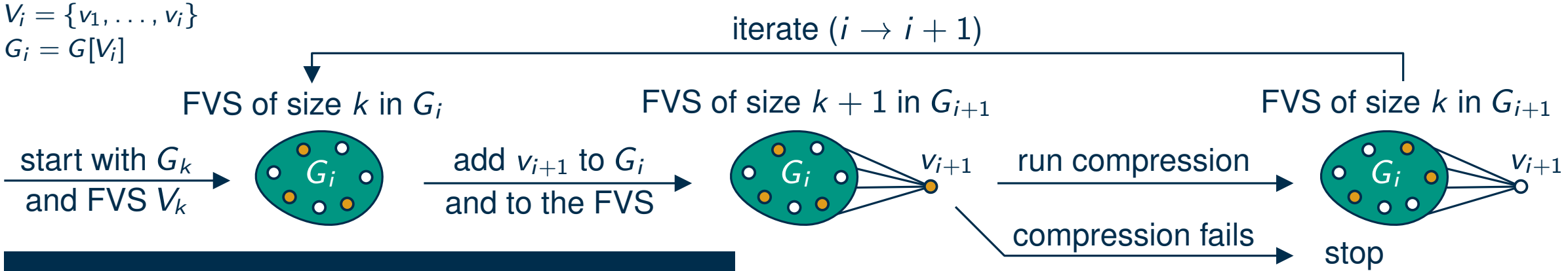


What is missing for a complete algorithm?

Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.

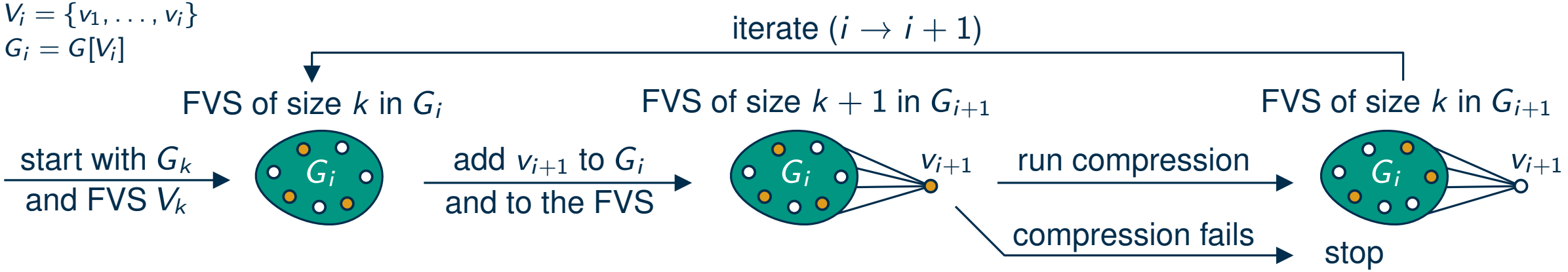


Where do we need problem-specific arguments?

Compression problem

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS of size $k+1$. Is there a FVS of size k?

Lemma
If we can solve Feedback Vertex Set Compression in $f(k) \cdot g(n)$ time, we can solve Feedback Vertex Set in $O(f(k) \cdot g(n) \cdot n)$ time.



selecting all vertices is a solution

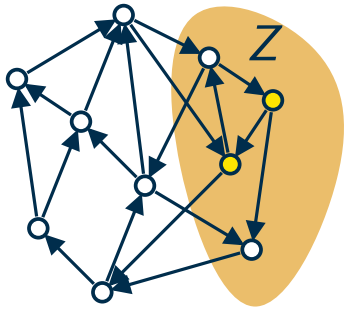
adding a vertex to the graph and to the solution yields a solution

no solution of size k for a subgraph $\Rightarrow$ no solution of size k for the whole graph

Disjoint solutions

Problem: FEEDBACK VERTEX SET COMPRESSION
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

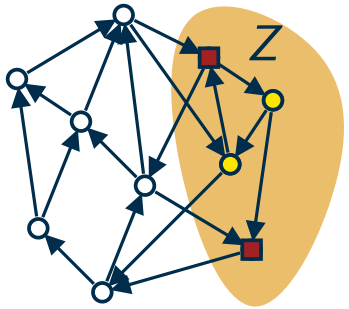


Disjoint solutions

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

- guess a subset $X \subseteq Z$; $Y = Z \setminus X$
- is there a FVS Z' with $|Z'| \leq k$ and $Z' \cap Z = X$?

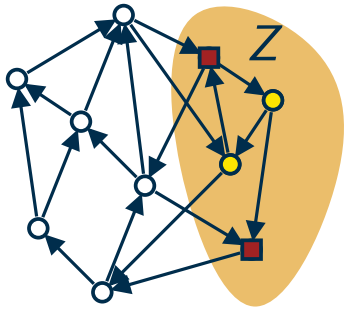


Disjoint solutions

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

- guess a subset $X \subseteq Z$; $Y = Z \setminus X$
- is there a FVS Z' with $|Z'| \leq k$ and $Z' \cap Z = X$?
- equivalent: is there a FVS F in $G - X$ with $|F| \leq k - |X|$ and $F \cap Y = \emptyset$?

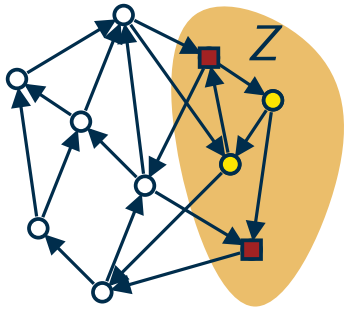


Disjoint solutions

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

- guess a subset $X \subseteq Z$; $Y = Z \setminus X$
- is there a FVS Z' with $|Z'| \leq k$ and $Z' \cap Z = X$?
- equivalent: is there a FVS F in $G - X$ with $|F| \leq k - |X|$ and $F \cap Y = \emptyset$?
- note: Y is a FVS of size $k - |X| + 1$ in $G - X$

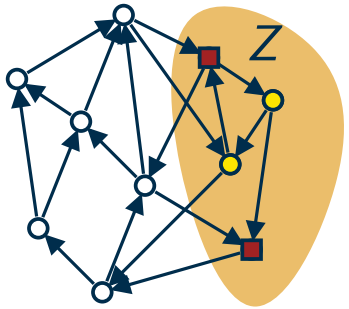


Disjoint solutions

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

- guess a subset $X \subseteq Z$; $Y = Z \setminus X$
- is there a FVS Z' with $|Z'| \leq k$ and $Z' \cap Z = X$?
- equivalent: is there a FVS F in $G - X$ with $|F| \leq k - |X|$ and $F \cap Y = \emptyset$?
- note: Y is a FVS of size $k - |X| + 1$ in $G - X$



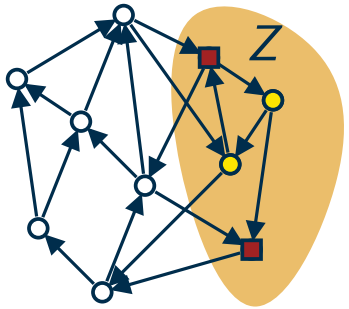
Problem: Disjoint Feedback Vertex Set
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Disjoint solutions

Problem: Feedback Vertex Set Compression
Given a directed graph, a parameter k, and a FVS Z of size $k+1$. Is there a FVS of size k?

Idea

- guess a subset $X \subseteq Z$; $Y = Z \setminus X$
- is there a FVS Z' with $|Z'| \leq k$ and $Z' \cap Z = X$?
- equivalent: is there a FVS F in $G - X$ with $|F| \leq k - |X|$ and $F \cap Y = \emptyset$?
- note: Y is a FVS of size $k - |X| + 1$ in $G - X$



Problem: Disjoint Feedback Vertex Set
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

$f(k) \cdot g(n)$-algo for Disjoint FVS $\Rightarrow O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$-algo for FVS Compression

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

DISJOINT-(PROBLEM) solvable in time $\alpha^k \cdot g(n)$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

DISJOINT-(PROBLEM) solvable in time $\alpha^k \cdot g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} \alpha^{k-i} \leq (k+1) \cdot \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

DISJOINT-(PROBLEM) solvable in time $\alpha^k \cdot g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} \alpha^{k-i} \leq (k+1) \cdot \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}$

Reminder
$(a+b)^k =$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

DISJOINT-(PROBLEM) solvable in time $\alpha^k \cdot g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} \alpha^{k-i} \leq (k+1) \cdot \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}$

Reminder
$(a+b)^k = \sum_{i=0}^{k} \binom{k}{i} a^i b^{k-i}$

Side-quest: Specific running times

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

DISJOINT-(PROBLEM) solvable in $g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} = 2^{k+1} - 1$
- resulting running time: $O\left(\sum_{i=0}^{k} \binom{k+1}{i} \cdot g(n)\right) = O(2^k \cdot g(n))$

DISJOINT-(PROBLEM) solvable in time $\alpha^k \cdot g(n)$

- it holds that: $\sum_{i=0}^{k} \binom{k+1}{i} \alpha^{k-i} \leq (k+1) \cdot \sum_{i=0}^{k} \binom{k}{i} \alpha^{k-i}$
- resulting running time: $O(k(1+\alpha)^k \cdot g(n))$

Reminder
$(a+b)^k = \sum_{i=0}^{k} \binom{k}{i} a^i b^{k-i}$

What did we do so far?

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$ time.

What did we do so far?

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$ time.

Goal

- FPT-algo for FVS in tournament graphs

What did we do so far?

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$ time.

Goal

- FPT-algo for FVS in tournament graphs
- sufficient: FPT-algo for DISJOINT FVS in tournament graphs

What did we do so far?

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$ time.

Goal

- FPT-algo for FVS in tournament graphs
- sufficient: FPT-algo for DISJOINT FVS in tournament graphs

Note

- so far, we did not use that G is a tournament graph

What did we do so far?

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$ time.

Goal

- FPT-algo for FVS in tournament graphs
- sufficient: FPT-algo for DISJOINT FVS in tournament graphs

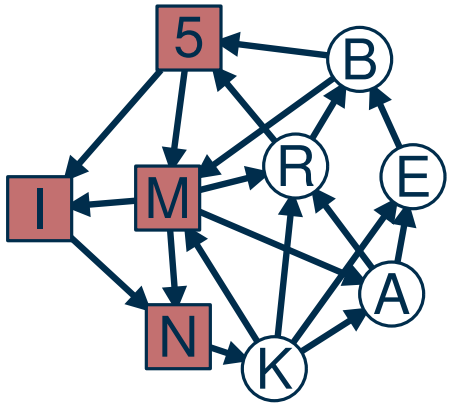
Note

- so far, we did not use that G is a tournament graph
- every induced subgraph of a tournament graph is a tournament graph

Find a minimum FVS disjoint to Y

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

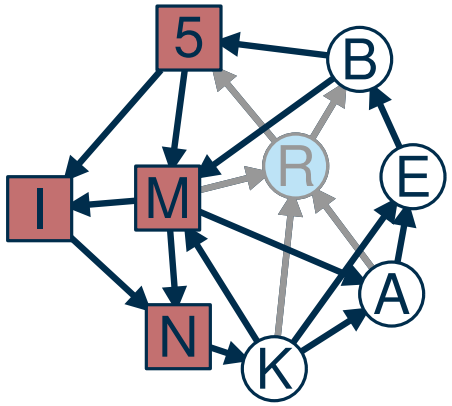
$Y = \{5, M, I, N\}$



Find a minimum FVS disjoint to Y

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

$Y = \{5, M, I, N\}$



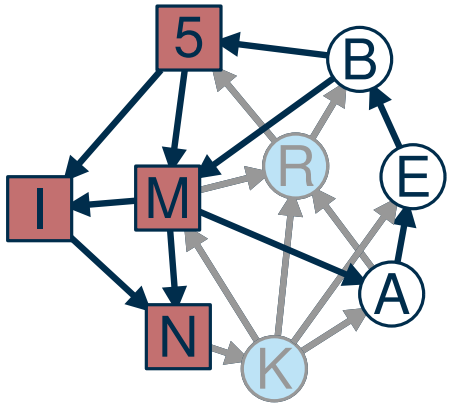
Solution

- R must be chosen (because of the triangle $5MR$)

Find a minimum FVS disjoint to Y

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

$$Y = \{5, M, I, N\}$$



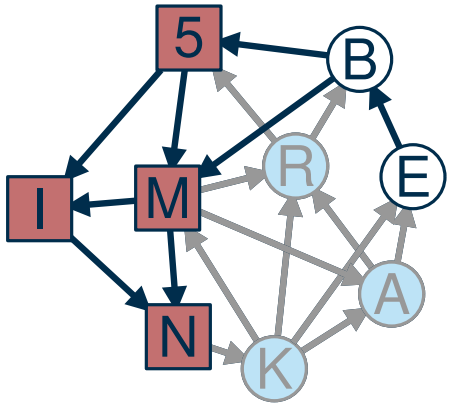
Solution

- R must be chosen (because of the triangle $5MR$)
- K must be chosen (because of the triangle MNK)

Find a minimum FVS disjoint to Y

Problem: DISJOINT FEEDBACK VERTEX SET
Given a directed graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

$$Y = \{5, M, I, N\}$$



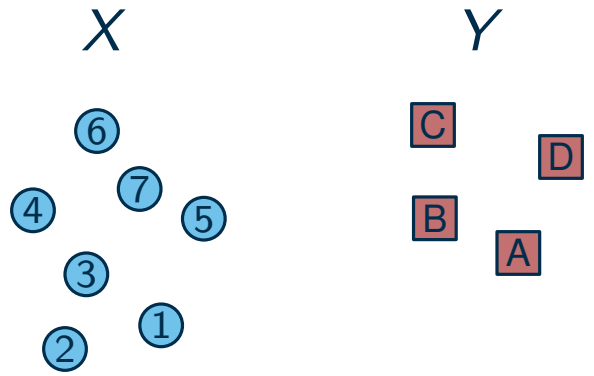
Solution

- R must be chosen (because of the triangle $5MR$)
- K must be chosen (because of the triangle MNK)
- additionally, one must choose B, E, or A

DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic

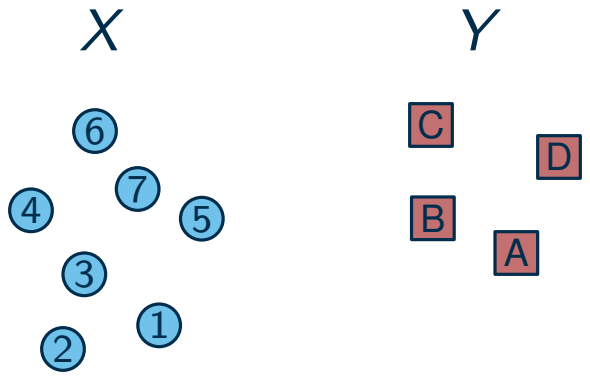


DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic

Why?

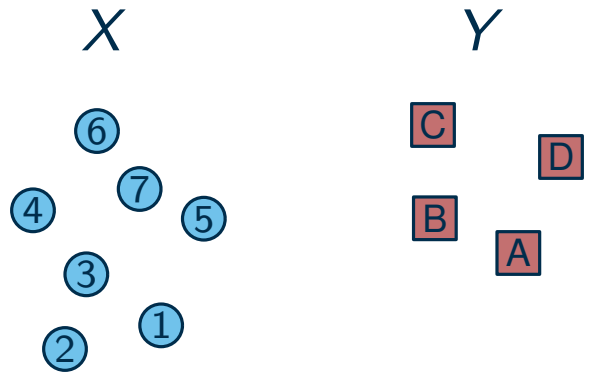


DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
  (otherwise: no-instance)

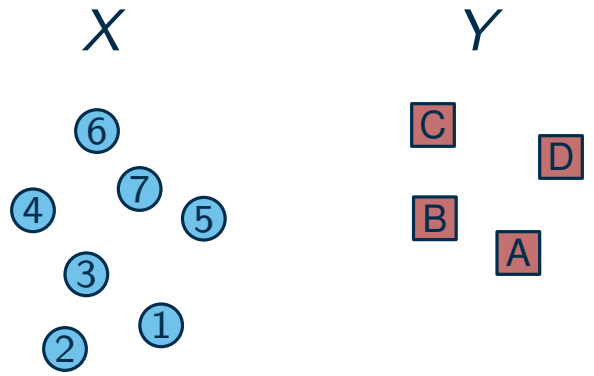
Why?



Disjoint FVS in tournament graphs

Situation and basic observations

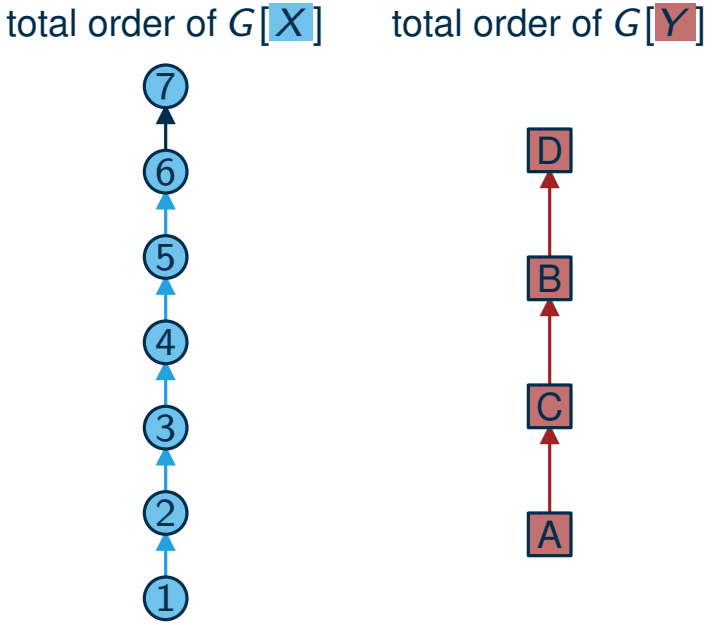
- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



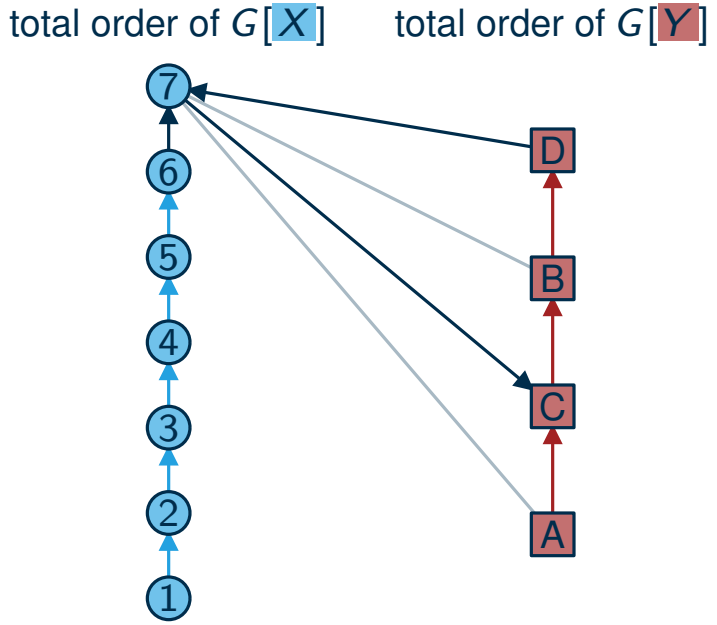
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1



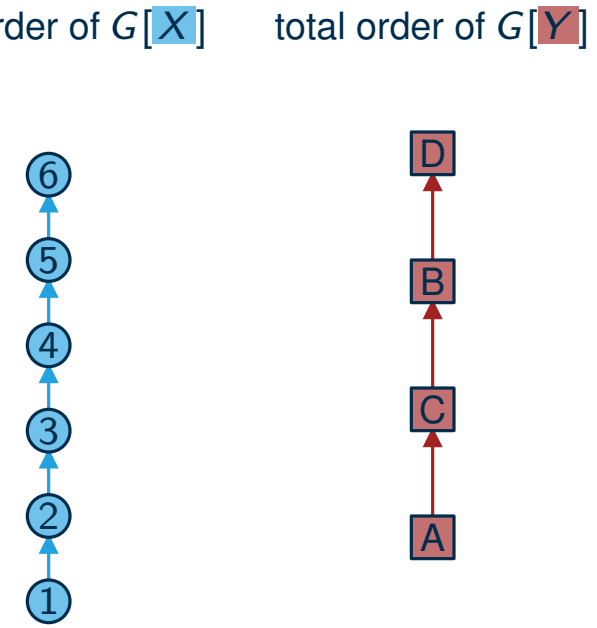
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1



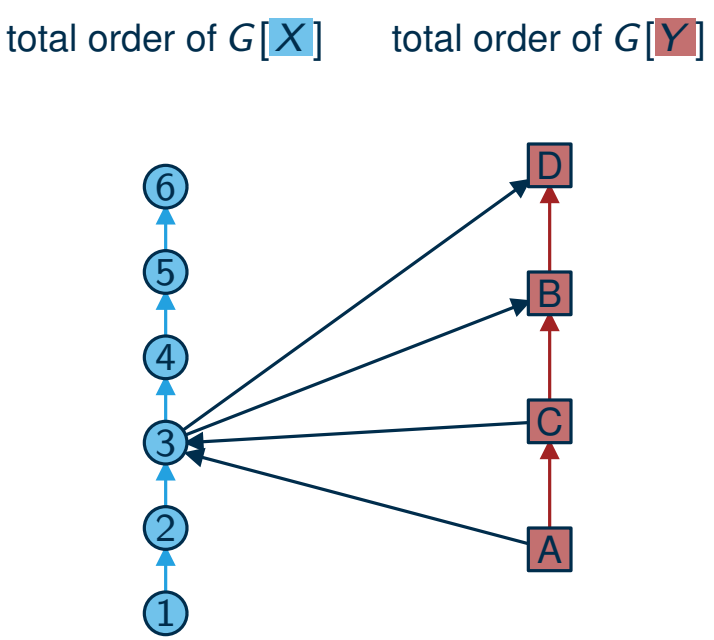
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

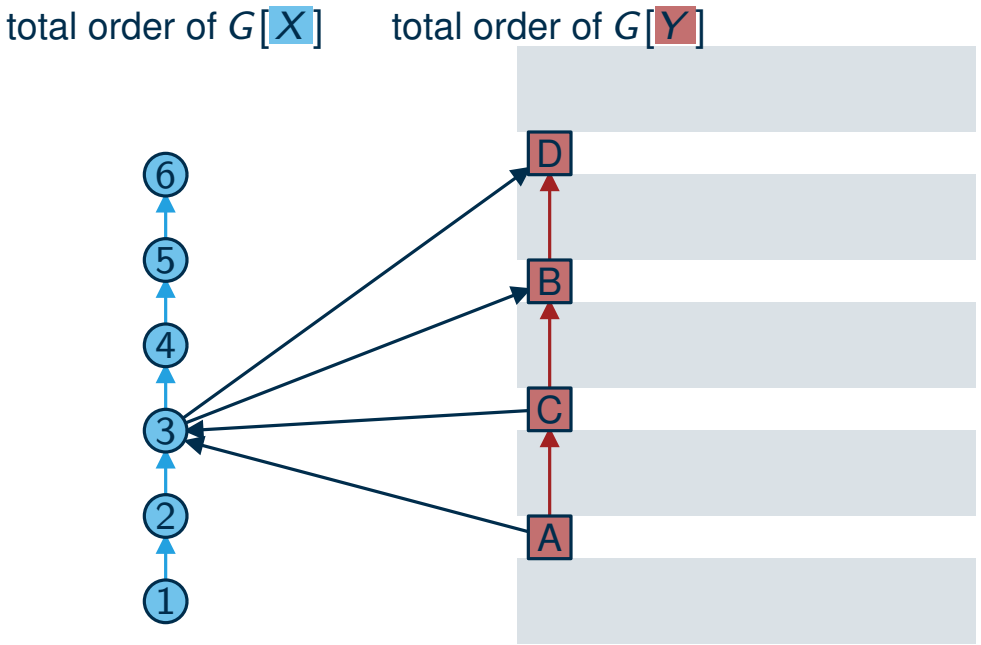
- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

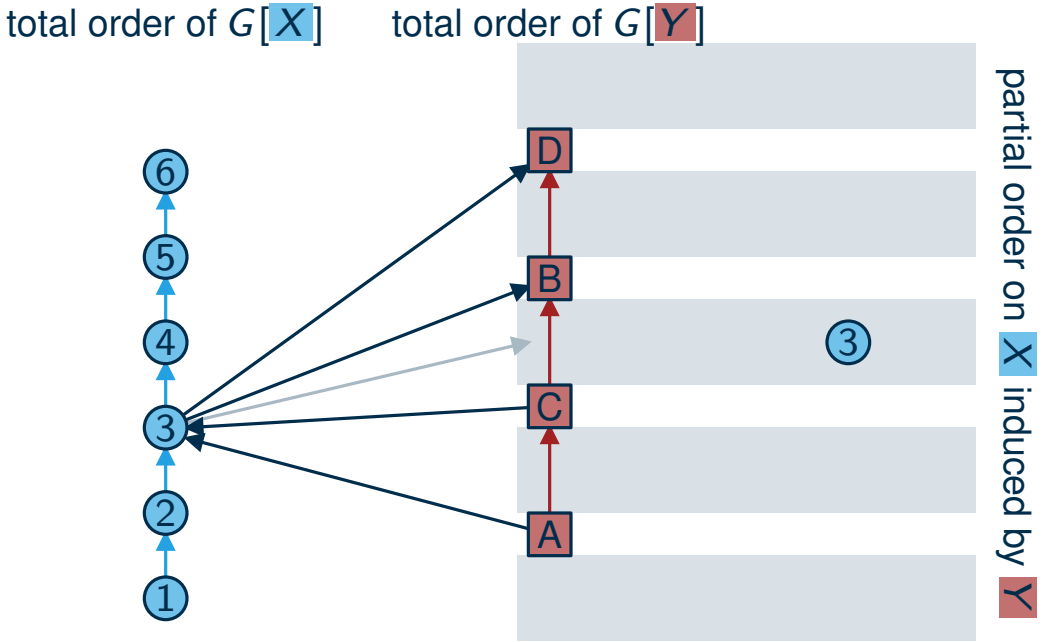
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

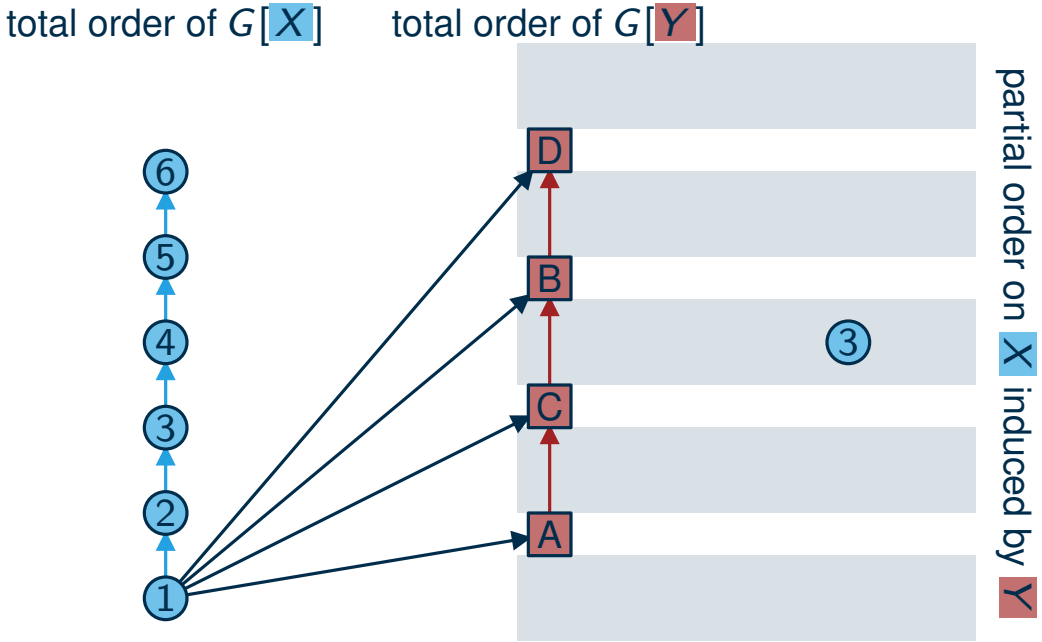
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

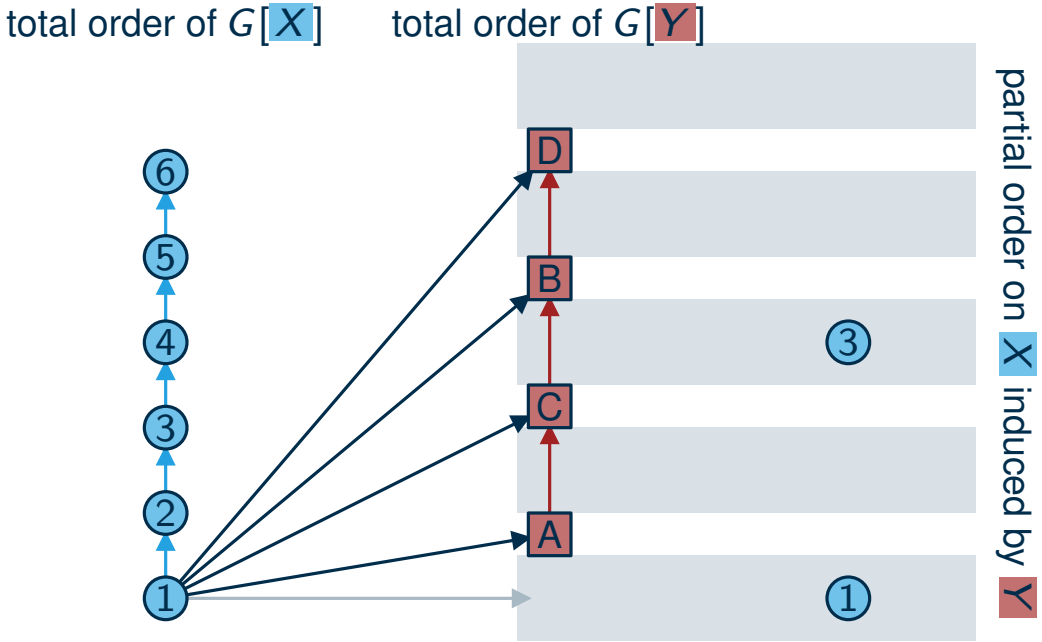
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

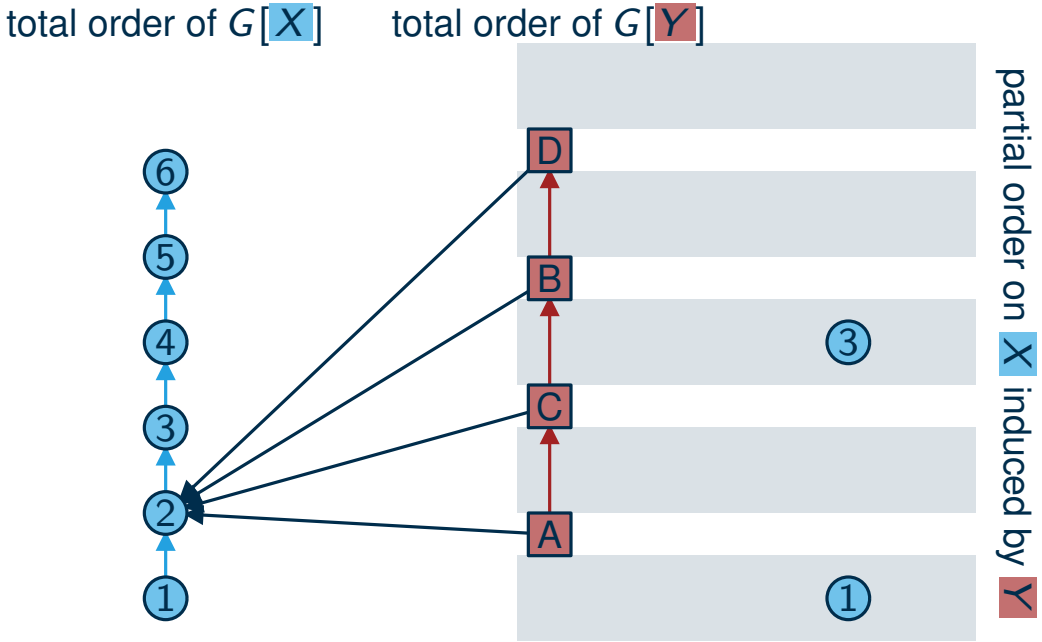
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

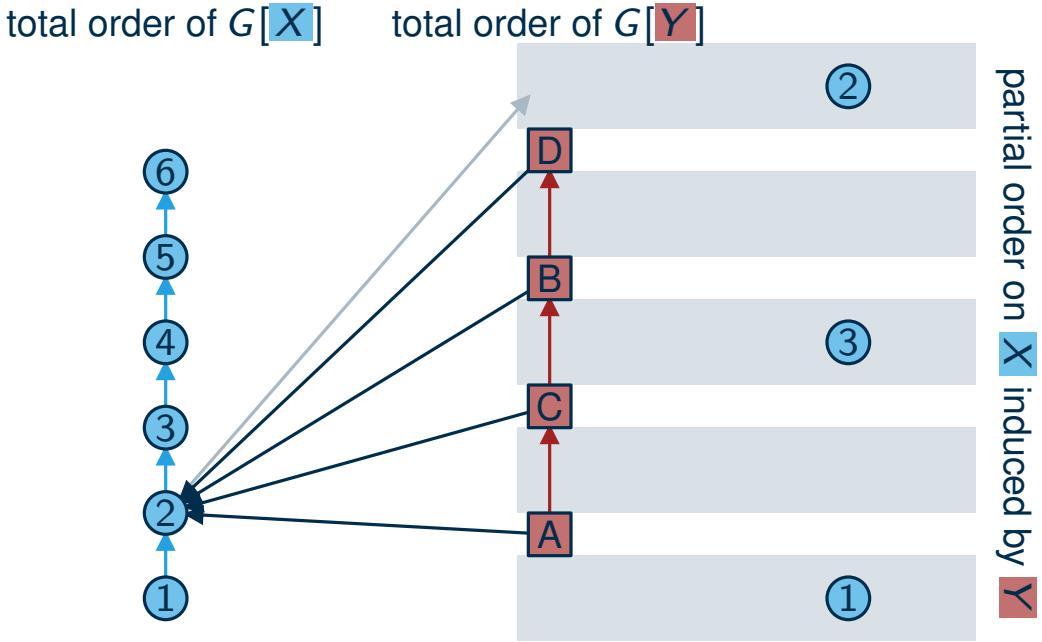
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



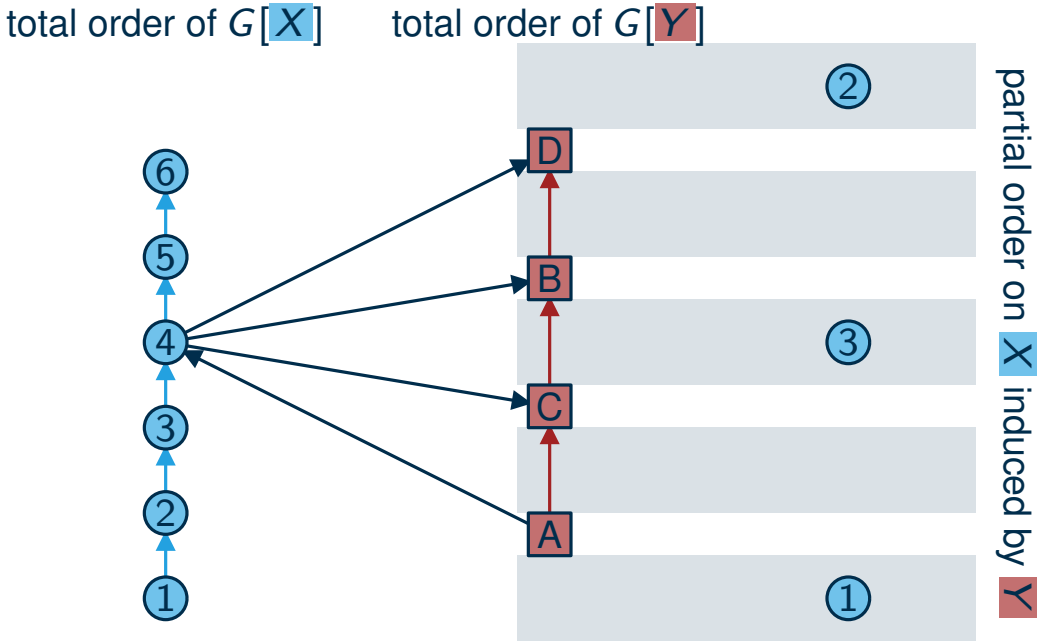
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

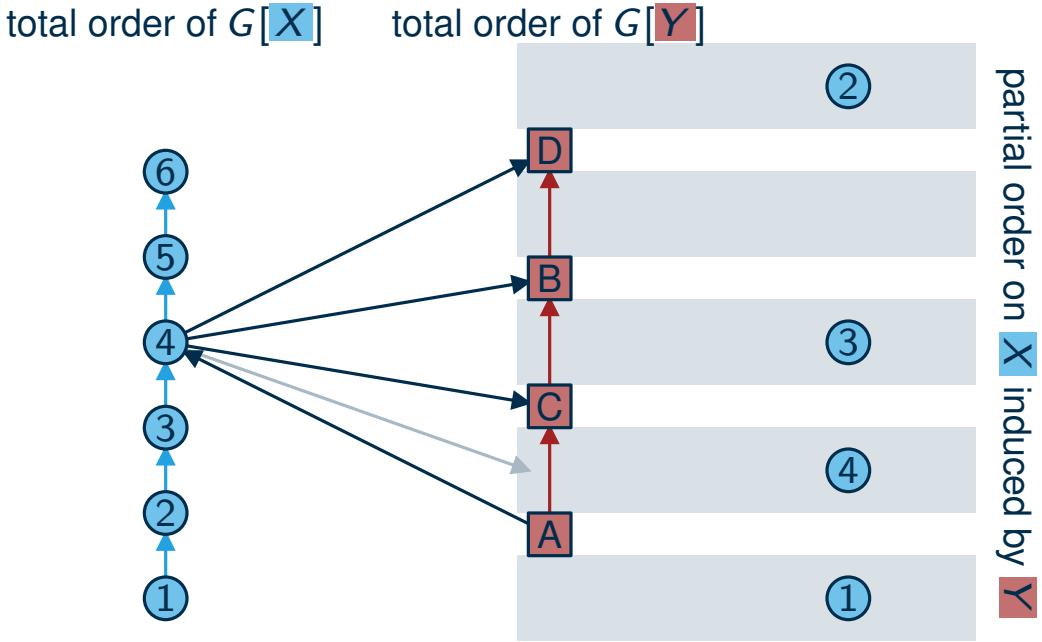
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



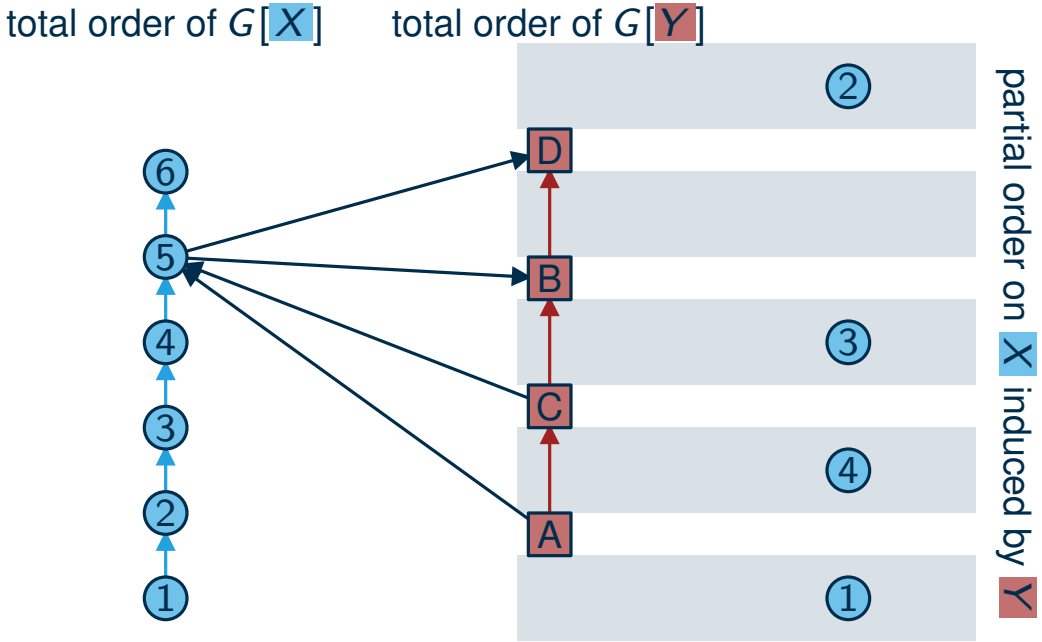
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

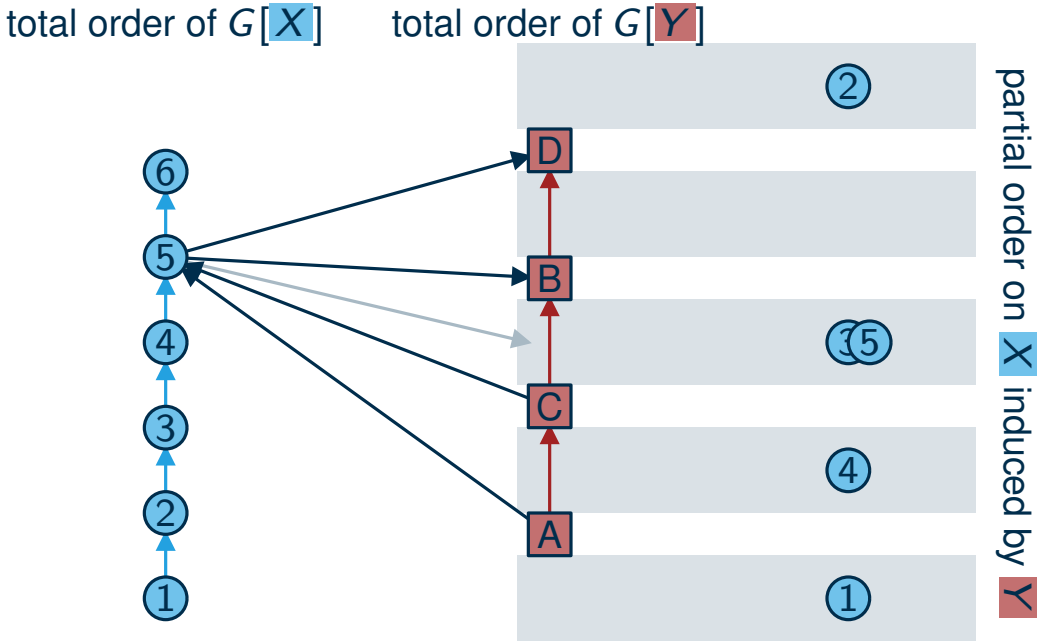
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

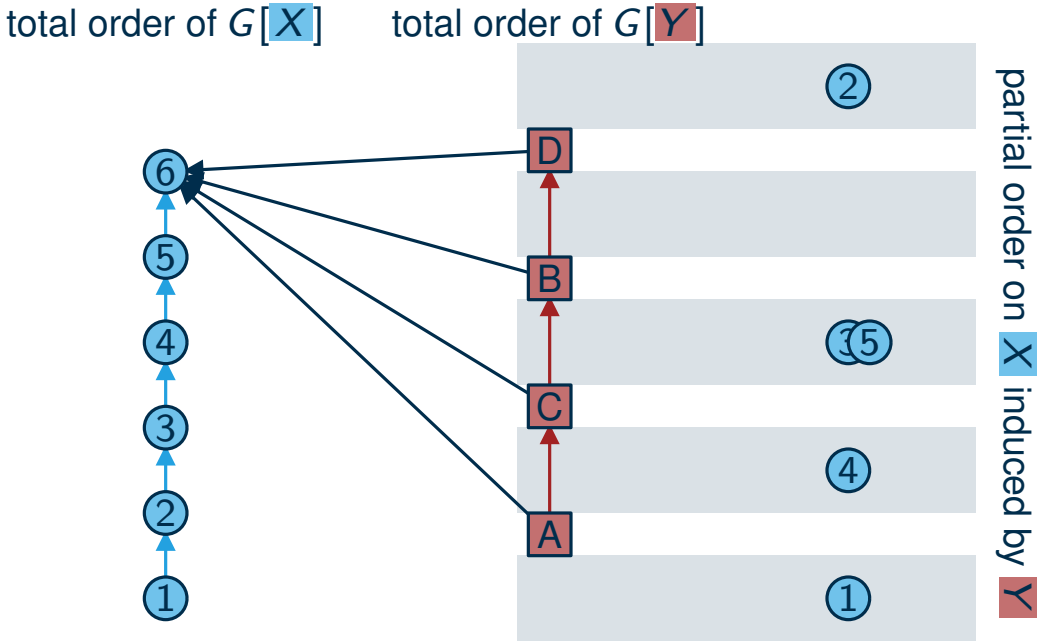
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

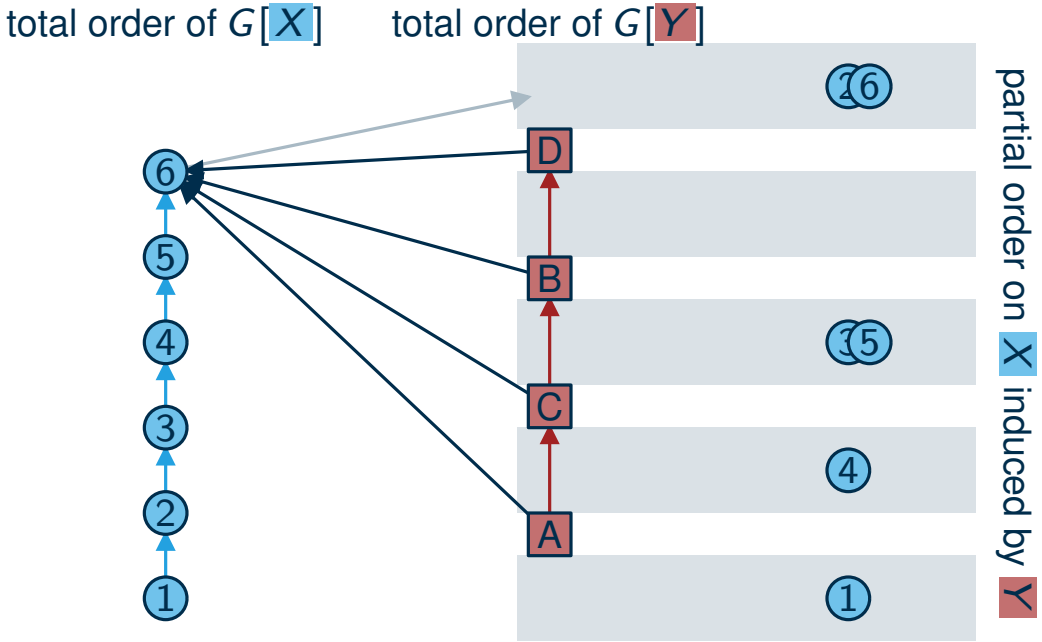
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y

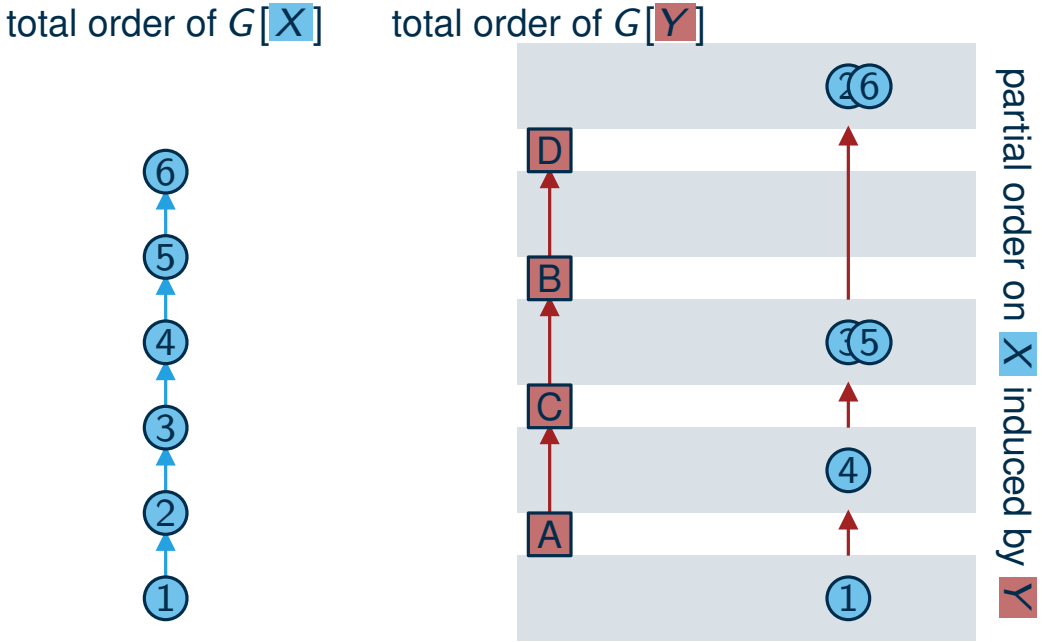
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y $\Rightarrow$ partial order on X

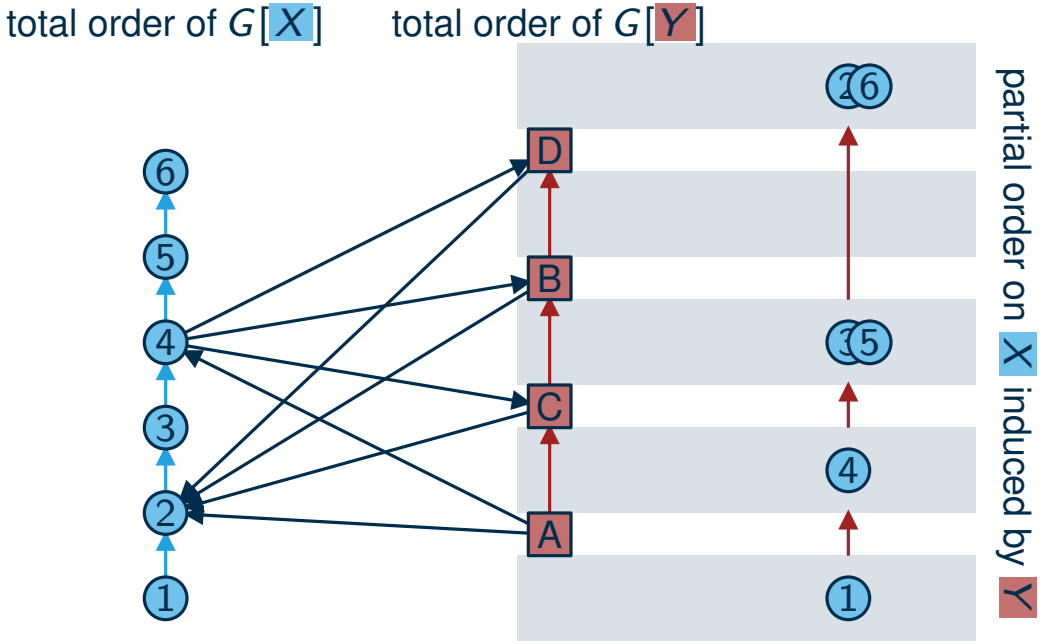
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y $\Rightarrow$ partial order on X
- conflict if this partial order on X disagrees with total order on X

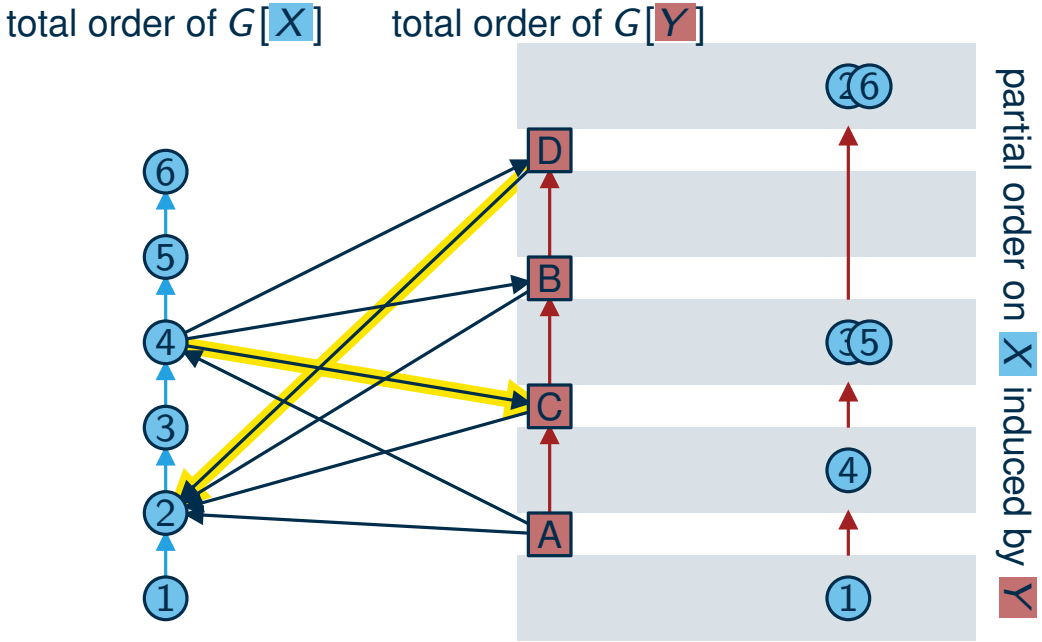
DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order

Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges



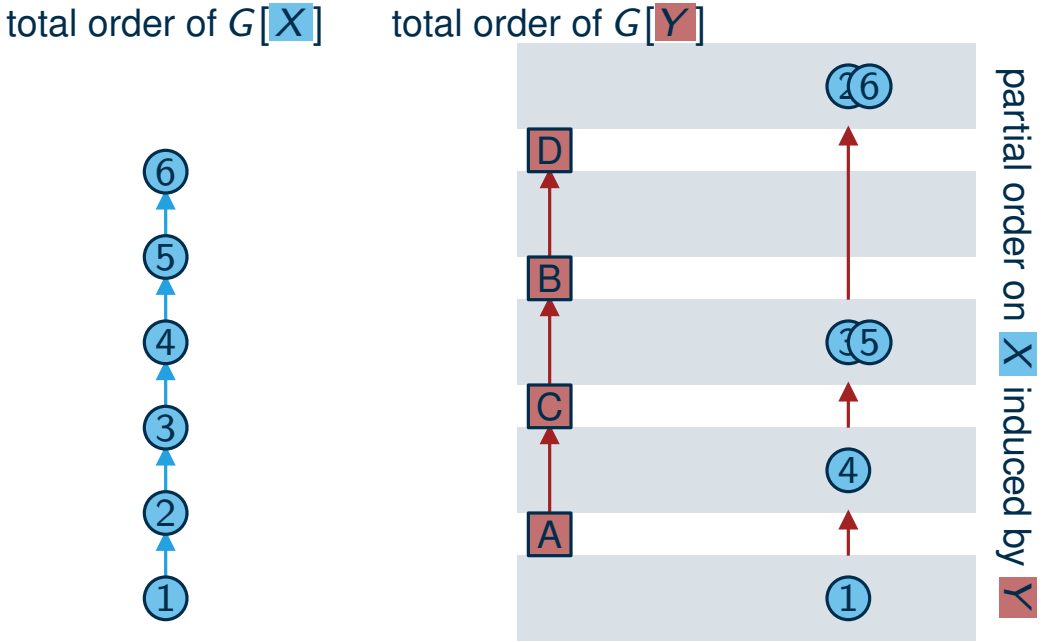
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y $\Rightarrow$ partial order on X
- conflict if this partial order on X disagrees with total order on X

DISJOINT FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

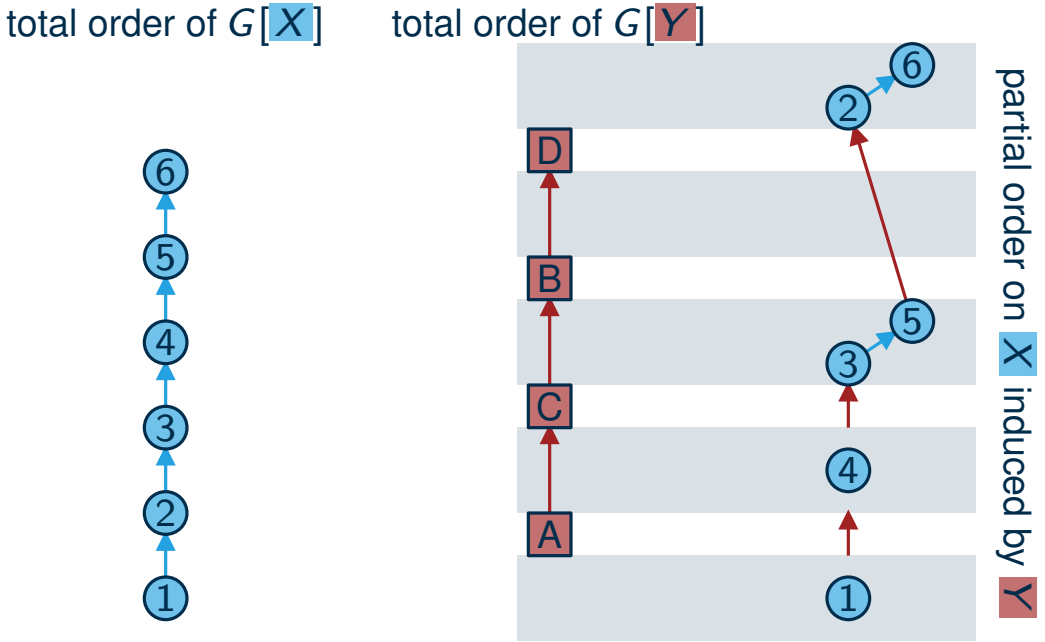
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y $\Rightarrow$ partial order on X
- conflict if this partial order on X disagrees with total order on X
- goal: keep as many vertices from X as possible such that the two orders agree

Disjoint FVS in tournament graphs

Situation and basic observations

- Y is given FVS of size $k+1 \Rightarrow G[X]$ is acyclic
- X is also a FVS $\Rightarrow G[Y]$ is acyclic
- acyclic tournament graphs define total order



Single-vertex conflicts for $x \in X$

- $G[Y] + x$ cyclic $\rightarrow$ delete x and reduce k by 1
- no conflict $\Leftrightarrow$ first incoming then outgoing edges

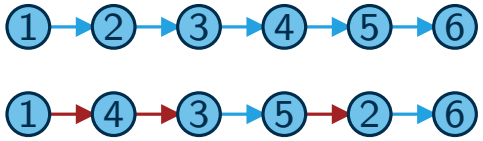
(Partial) order on X induced by Y

- $x \in X$ not selected for deletion $\Rightarrow$ clear where x fits in total order of Y $\Rightarrow$ partial order on X
- conflict if this partial order on X disagrees with total order on X
- goal: keep as many vertices from X as possible such that the two orders agree
- break ties in the partial order according to the total order in $G[X] \rightarrow$ two total orders on X

Longest Increasing Subsequence

Recap

- two orders of the vertices in X:
  - one given by the graph $G[X]$
  - one induced by where they fit into the order of Y given by $G[Y]$

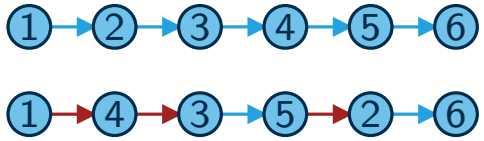


- just seen: F is FVS $\Leftrightarrow$ both orders agree on $X \setminus F$

Longest Increasing Subsequence

Recap

- two orders of the vertices in X:
  - one given by the graph $G[X]$
  - one induced by where they fit into the order of Y given by $G[Y]$



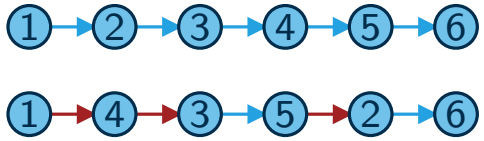
- just seen: F is FVS $\Leftrightarrow$ both orders agree on $X \setminus F$

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

Longest Increasing Subsequence

Recap

- two orders of the vertices in X:
  - one given by the graph $G[X]$
  - one induced by where they fit into the order of Y given by $G[Y]$



- just seen: F is FVS $\Leftrightarrow$ both orders agree on $X \setminus F$

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

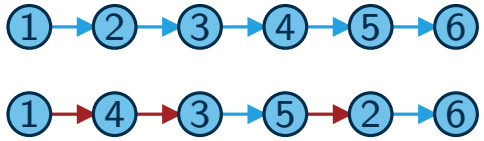
Algorithms for Longest Increasing Subsequence

- simple dynamic program (like Longest Common Subsequence): $O(n^2)$

Longest Increasing Subsequence

Recap

- two orders of the vertices in X:
  - one given by the graph $G[X]$
  - one induced by where they fit into the order of Y given by $G[Y]$



- just seen: F is FVS $\Leftrightarrow$ both orders agree on $X \setminus F$

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

Algorithms for Longest Increasing Subsequence

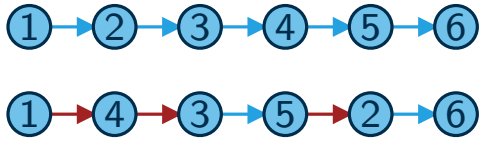
- simple dynamic program (like Longest Common Subsequence): $O(n^2)$
- dynamic program with binary search: $O(n \log n)$

$\rightarrow$ next slide

Longest Increasing Subsequence

Recap

- two orders of the vertices in X:
  - one given by the graph $G[X]$
  - one induced by where they fit into the order of Y given by $G[Y]$



- just seen: F is FVS $\Leftrightarrow$ both orders agree on $X \setminus F$

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

Algorithms for Longest Increasing Subsequence

- simple dynamic program (like Longest Common Subsequence): $O(n^2)$
- dynamic program with binary search: $O(n \log n)$ $\rightarrow$ next slide
- input is a permutation of the integers $1, \ldots, n$: $O(n \log \log n)$ https://dl.acm.org/doi/10.1145/359581.359603

DP for LONGEST INCREASING SUBSEQUENCE

Problem: LONGEST INCREASING SUBSEQUENCE
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer

DP for LONGEST INCREASING SUBSEQUENCE

Problem: LONGEST INCREASING SUBSEQUENCE
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

$\ell = 1$ | 1 | 4 | 3 | 5 | 2 | 6 |

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer

DP for LONGEST INCREASING SUBSEQUENCE

Problem: LONGEST INCREASING SUBSEQUENCE
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer

$\ell = 1$	**1**	4	3	5	2	6
$\ell = 2$	**1**	4	3	5	**2**	6

DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer

$\ell = 1$	**1**	4	3	5	2	6
$\ell = 2$	**1**	4	3	5	**2**	6
$\ell = 3$	**1**	4	**3**	**5**	2	6

DP for LONGEST INCREASING SUBSEQUENCE

Problem: LONGEST INCREASING SUBSEQUENCE
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer

$\ell = 1$	**1**	4	3	5	2	6
$\ell = 2$	**1**	4	3	5	**2**	6
$\ell = 3$	**1**	4	**3**	**5**	2	6
$\ell = 4$	**1**	4	**3**	**5**	2	**6**

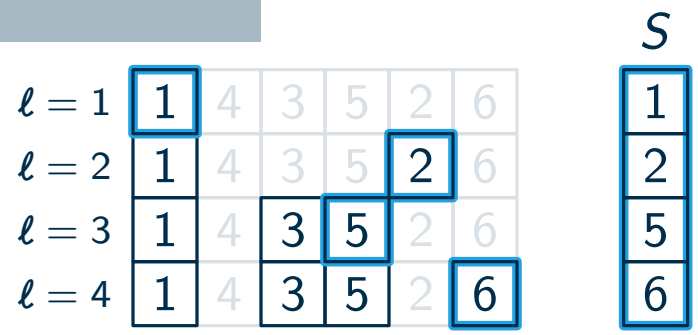
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



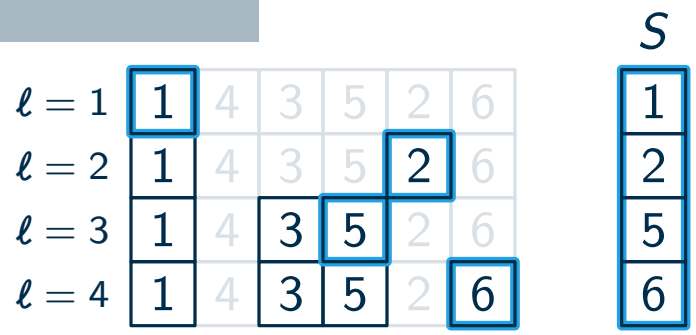
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Why is S sorted increasingly?

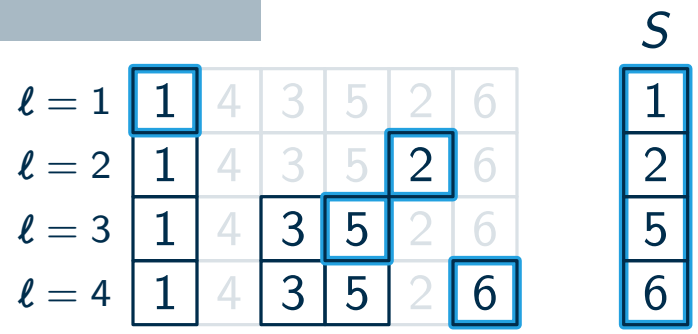
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}

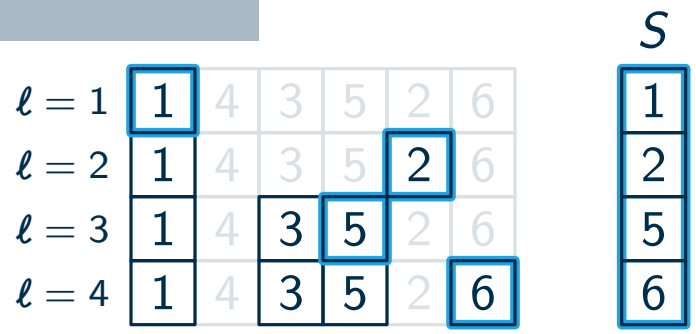
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

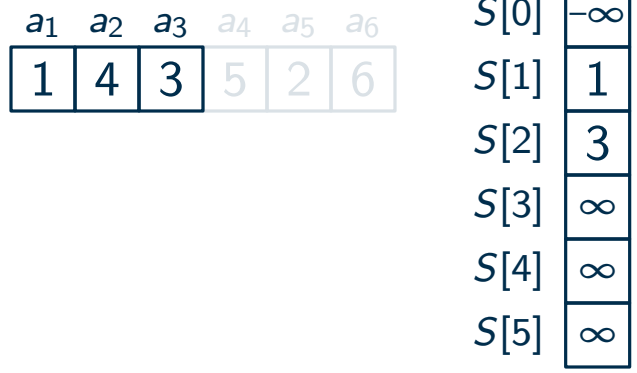
Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}



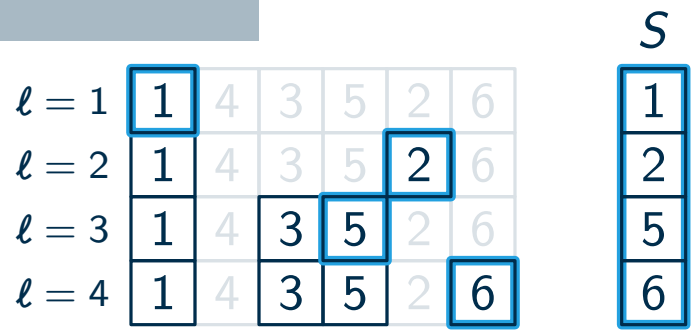
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

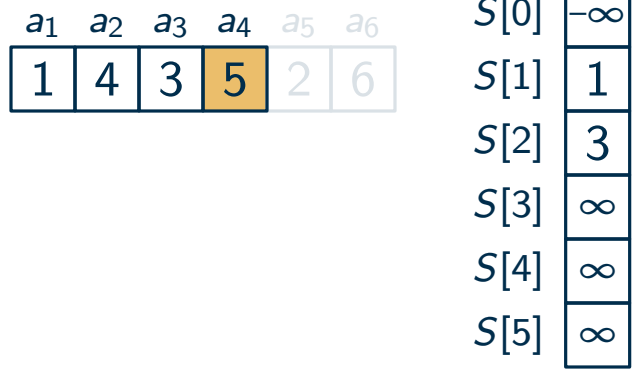
Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



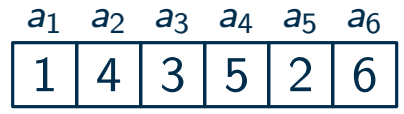
Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}



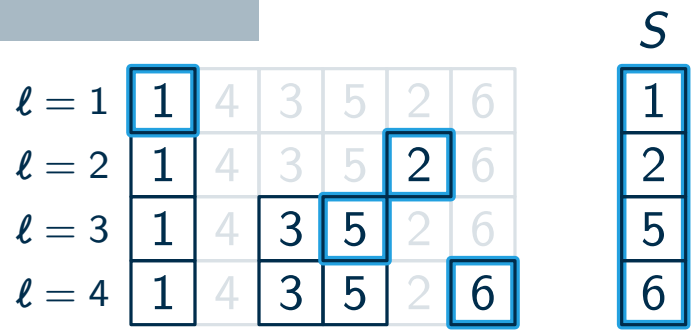
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.



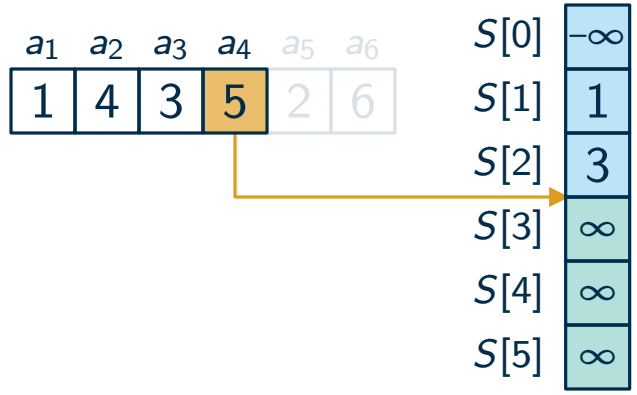
Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$



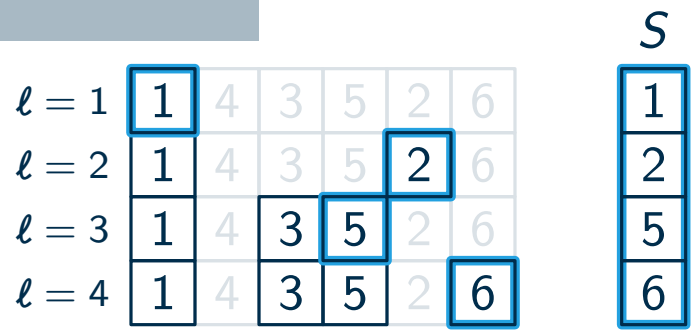
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

$S[0]$	$-\infty$
$S[1]$	1
$S[2]$	3
$S[3]$	5
$S[4]$	∞
$S[5]$	∞

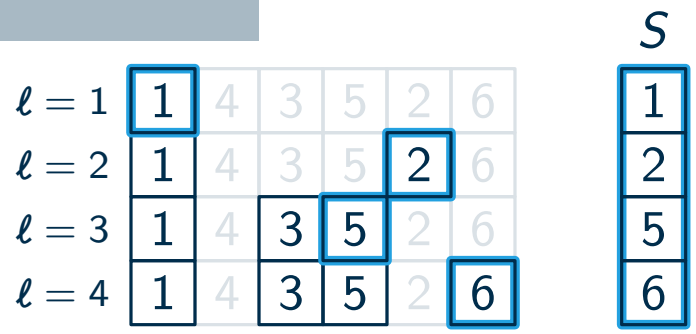
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

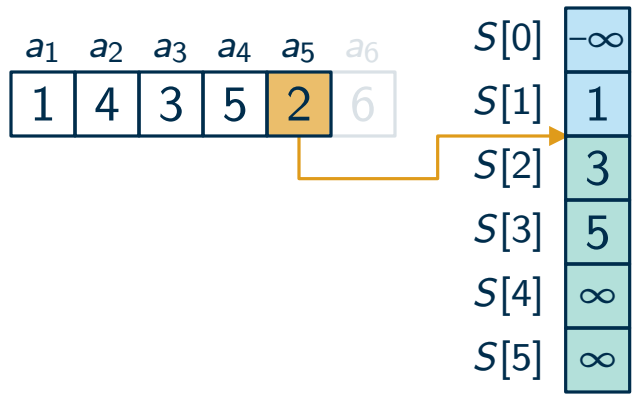
Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$



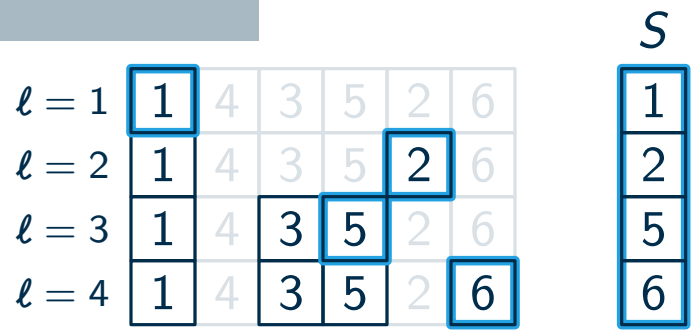
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

$S[0]$	$-\infty$
$S[1]$	1
$S[2]$	2
$S[3]$	5
$S[4]$	∞
$S[5]$	∞

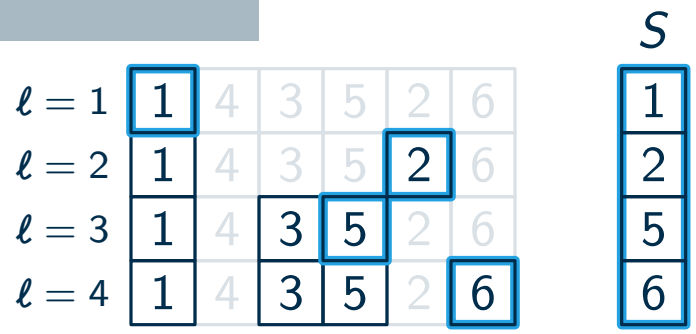
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

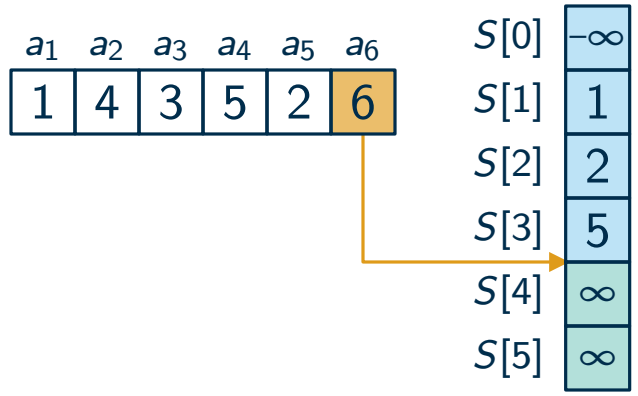
Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$



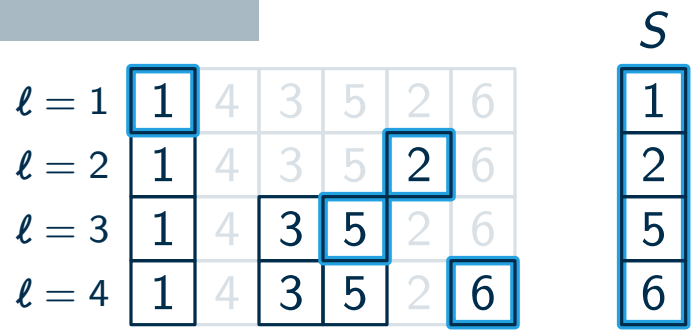
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1}
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

$S[0]$	$-\infty$
$S[1]$	1
$S[2]$	2
$S[3]$	5
$S[4]$	6
$S[5]$	∞

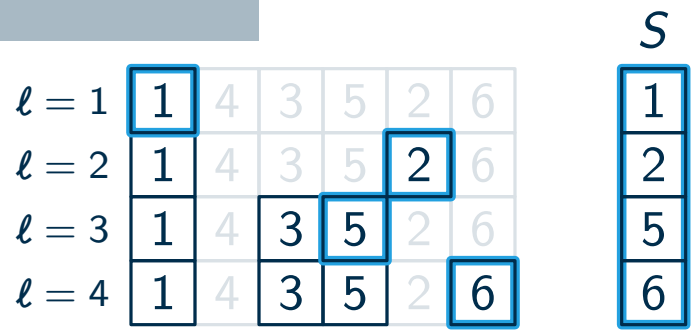
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1} binary search $\rightarrow O(\log n)$
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

$S[0]$	$-\infty$
$S[1]$	1
$S[2]$	2
$S[3]$	5
$S[4]$	6
$S[5]$	∞

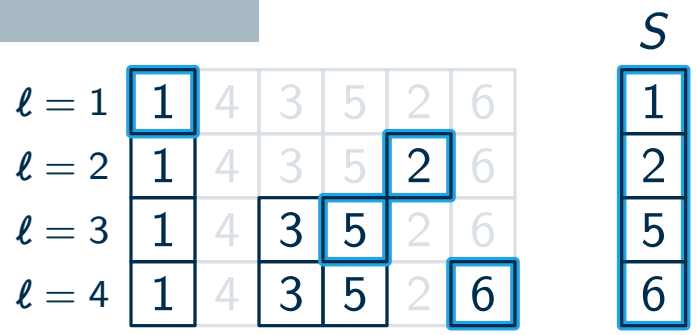
DP for Longest Increasing Subsequence

Problem: Longest Increasing Subsequence
Given a sequence of numbers $a_1, \ldots, a_n$. Find the longest increasing subsequence, i.e., if a_i and a_j are in the subsequence, then $i < j$ implies $a_i < a_j$.

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

Computing more than necessary

- compute for each ℓ the best increasing subsequence of length ℓ
- a sequence is better if it ends in a smaller integer
- $S[\ell]$: smallest integer a length-ℓ increasing subsequence ends in



Dynamic program

$\rightarrow O(n \log n)$

- let S_i be this array S, but for the input $a_1, \ldots, a_i$
- compute S_i based on S_{i-1} binary search $\rightarrow O(\log n)$
  - a_i can extend sequence of length ℓ if $S[\ell] < a_i$
  - results in better sequence of length $\ell + 1$ if $a_i < S[\ell + 1]$

a_1	a_2	a_3	a_4	a_5	a_6
1	4	3	5	2	6

$S[0]$	$-\infty$
$S[1]$	1
$S[2]$	2
$S[3]$	5
$S[4]$	6
$S[5]$	∞

FVS on tournament graphs

DISJOINT FEEDBACK VERTEX SET: $O(n \log n)$ (via LONGEST INCREASING SUBSEQUENCE)

Lemma
If we can solve DISJOINT FVS in $f(k) \cdot g(n)$ time, then we can solve FVS COMPRESSION in $O\left(\sum_{i=0}^{k} \binom{k+1}{i} f(k-i) \cdot g(n)\right)$ time.

FEEDBACK VERTEX SET COMPRESSION: $O(2^k \cdot n \log n)$

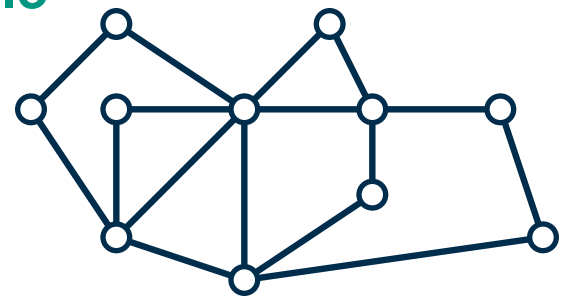
Lemma
If we can solve FVS COMPRESSION in $f(k) \cdot g(n)$ time, we can solve FVS in $O(f(k) \cdot g(n) \cdot n)$.

Theorem
FEEDBACK VERTEX SET can be solved in $O(2^k \cdot n^2 \log n)$ time on tournament graphs.

Undirected Feedback Vertex Set

Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

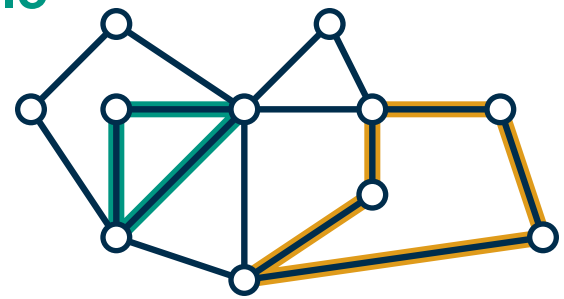
Example



Undirected Feedback Vertex Set

Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example

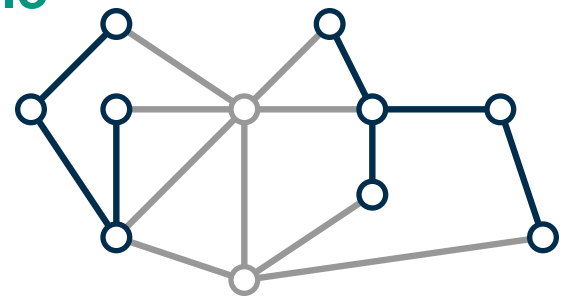


- we must hit all cycles
- deleting two vertices is necessary

Undirected Feedback Vertex Set

Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Example



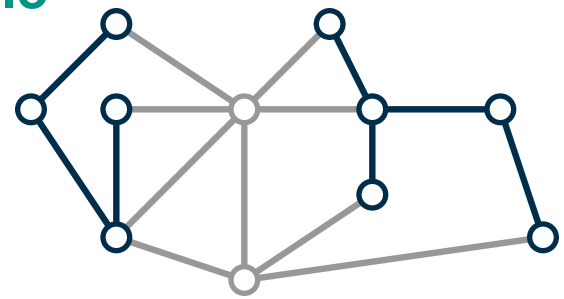
- we must hit all cycles
- deleting two vertices is necessary
- and sufficient

Undirected Feedback Vertex Set

Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Goal: FPT-Algo for Feedback Vertex Set

Example



- we must hit all cycles
- deleting two vertices is necessary
- and sufficient

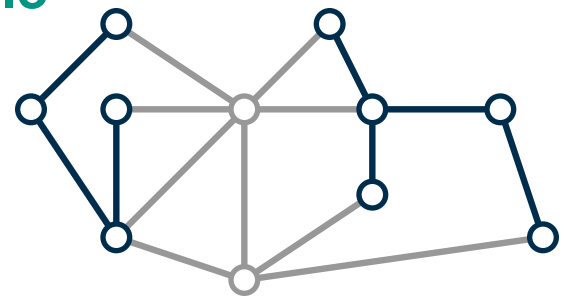
Undirected Feedback Vertex Set

Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Goal: FPT-Algo for Feedback Vertex Set

Problem: Disjoint FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Example



- we must hit all cycles
- deleting two vertices is necessary
- and sufficient

Undirected Feedback Vertex Set

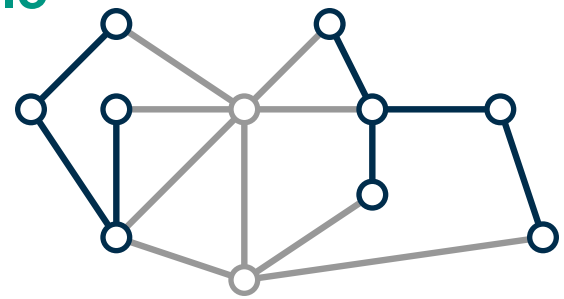
Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Goal: FPT-Algo for Feedback Vertex Set

Problem: Disjoint FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Does an FPT-algo for Disjoint FVS yield an FPT-Algo für FVS?

Example



- we must hit all cycles
- deleting two vertices is necessary
- and sufficient

Undirected Feedback Vertex Set

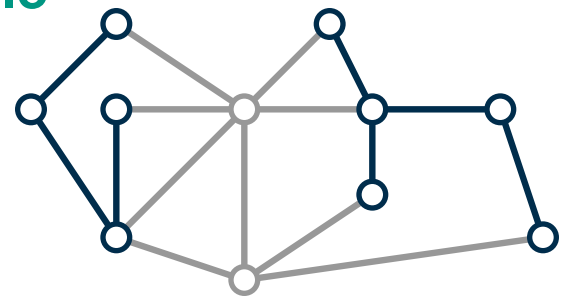
Problem: Feedback Vertex Set (undirected)
Given an undirected graph $G = (V, E)$ and a parameter k. Is there a feedback vertex set of size k?
($F \subseteq V$ is a *feedback vertex set* if $G - F$ is acyclic)

Goal: FPT-Algo for Feedback Vertex Set

Problem: Disjoint FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Does an FPT-algo for Disjoint FVS yield an FPT-Algo für FVS?

Example



- we must hit all cycles
- deleting two vertices is necessary
- and sufficient

Yes!

selecting all vertices is a solution

adding a vertex to the graph and to the solution yields a solution

no solution of size k for a subgraph
$\Rightarrow$ no solution of size k for the whole graph

An algorithm for DISJOINT FVS

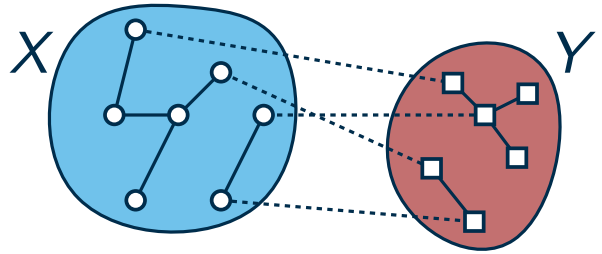
Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

Plan

- choose vertices from $X = V \setminus Y$ and none from Y

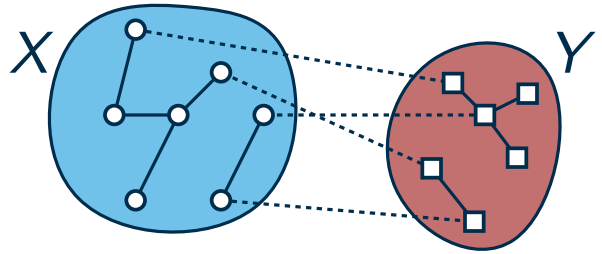


Note: $G[X]$ and $G[Y]$ are acyclic

Why?

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

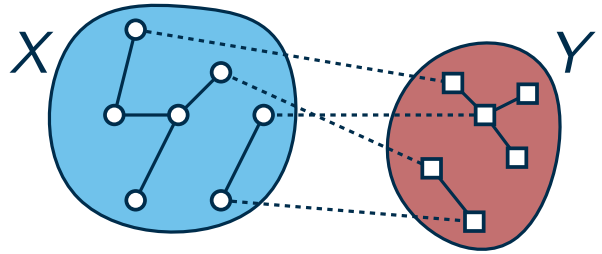
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$

Option 1: choose $v \in F$

Option 2: choose $v \notin F$

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$

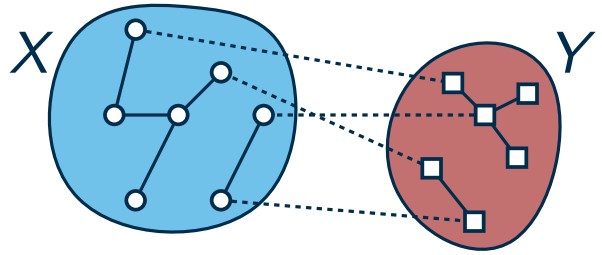
Option 1: choose $v \in F$

- choosing v reduces k by 1
- bounds the height of the tree

Option 2: choose $v \notin F$

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

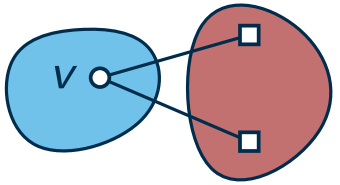
Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y**

Option 1: choose $v \in F$

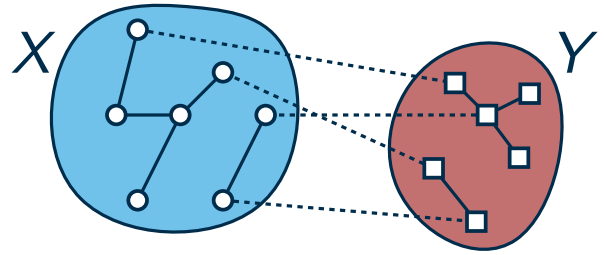
- choosing v reduces k by 1
- bounds the height of the tree

Option 2: choose $v \notin F$



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

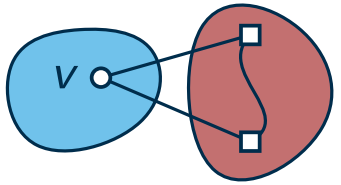
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y**

Option 1: choose $v \in F$

- choosing v reduces k by 1
- bounds the height of the tree

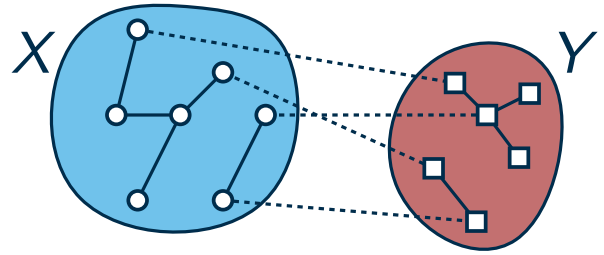
Option 2: choose $v \notin F$

- case1: neighbors are connected
  $\Rightarrow v \notin F$ is actually not an option



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

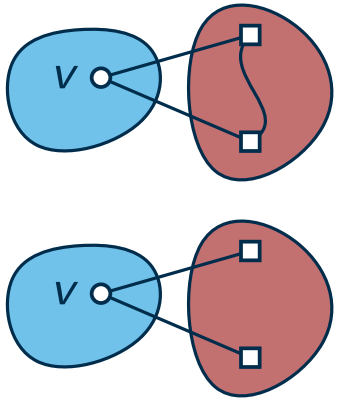
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y**

Option 1: choose $v \in F$

- choosing v reduces k by 1
- bounds the height of the tree

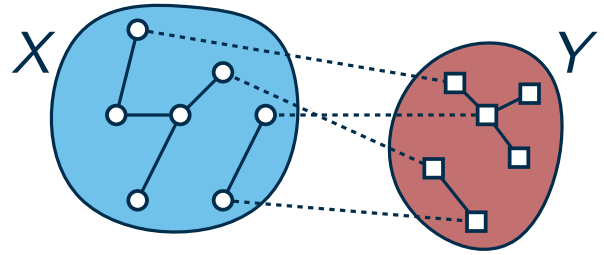
Option 2: choose $v \notin F$

- case1: neighbors are connected
  $\Rightarrow v \notin F$ is actually not an option
- case2: neighbors not connected



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

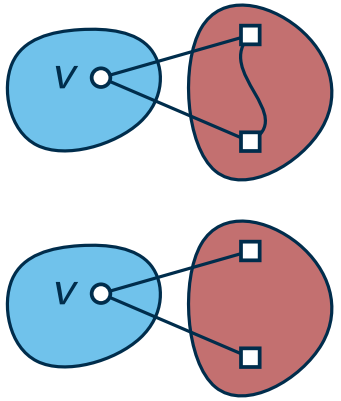
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y**

Option 1: choose $v \in F$

- choosing v reduces k by 1
- bounds the height of the tree

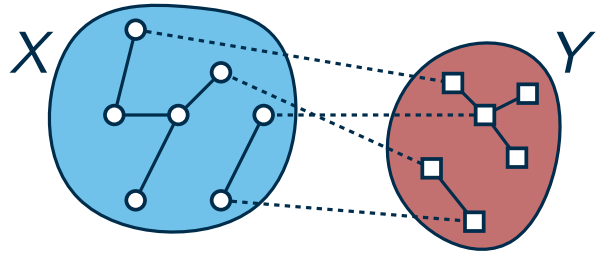
Option 2: choose $v \notin F$

- case1: neighbors are connected
  $\Rightarrow v \notin F$ is actually not an option
- case2: neighbors not connected
  $\Rightarrow$ reduces number of components in Y
- happens at most k times (as $|Y| = k+1$)



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

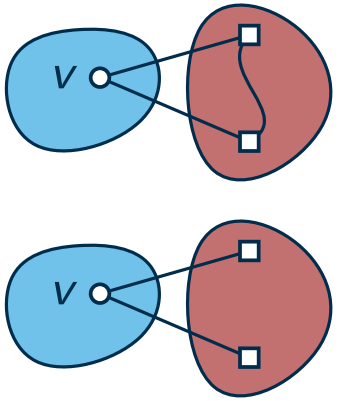
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible

Option 1: choose $v \in F$

- choosing v reduces k by 1
- bounds the height of the tree

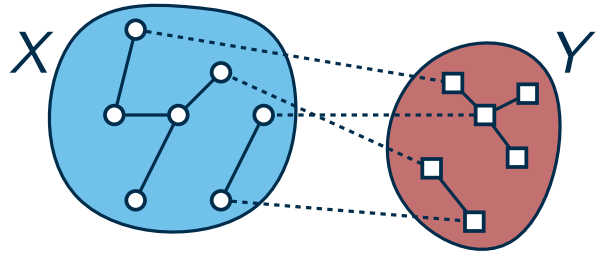
Option 2: choose $v \notin F$

- case1: neighbors are connected
  $\Rightarrow v \notin F$ is actually not an option
- case2: neighbors not connected
  $\Rightarrow$ reduces number of components in Y
- happens at most k times (as $|Y| = k+1$)



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

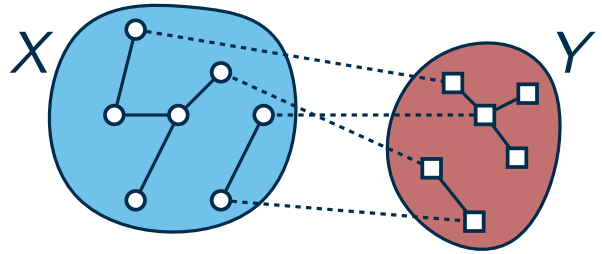
Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ← **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Why does such a vertex u exist?

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?

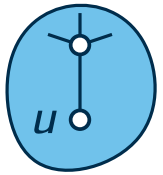


Note: $G[X]$ and $G[Y]$ are acyclic

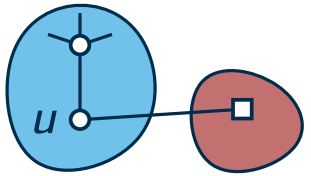
Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Case 1: u has no neighbor in Y

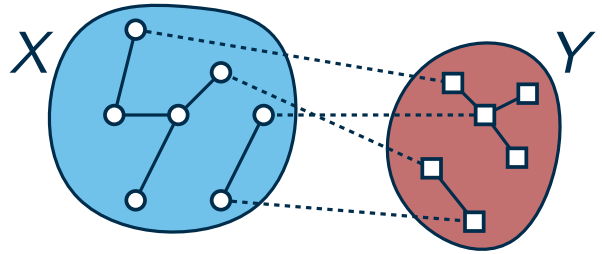


Case 2: u has one neighbor in Y



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



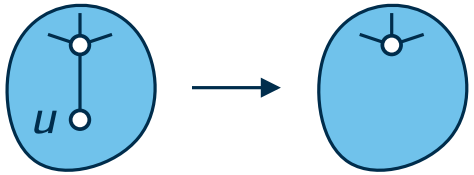
Note: $G[X]$ and $G[Y]$ are acyclic

Plan

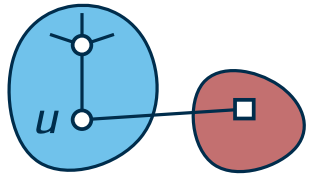
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Case 1: u has no neighbor in Y

- u does not lie on a cycle
- **reduction rule:** delete u

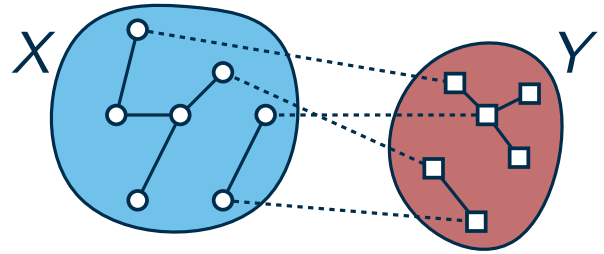


Case 2: u has one neighbor in Y



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



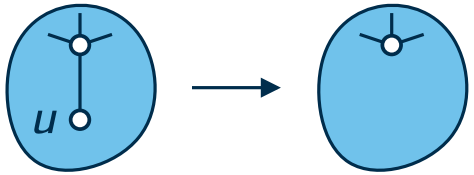
Note: $G[X]$ and $G[Y]$ are acyclic

Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

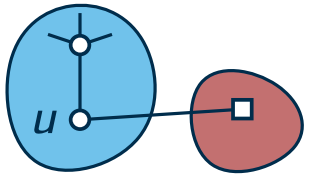
Case 1: u has no neighbor in Y

- u does not lie on a cycle
- **reduction rule:** delete u



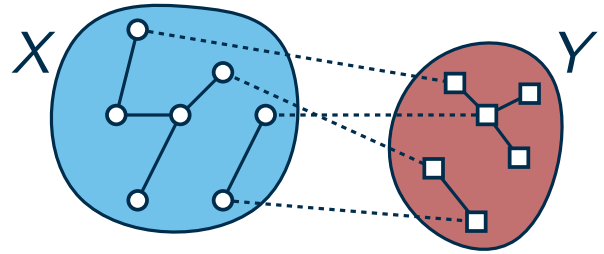
Case 2: u has one neighbor in Y

- u has degree 2 in G



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



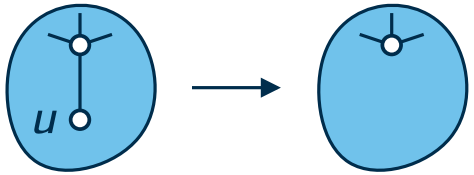
Note: $G[X]$ and $G[Y]$ are acyclic

Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

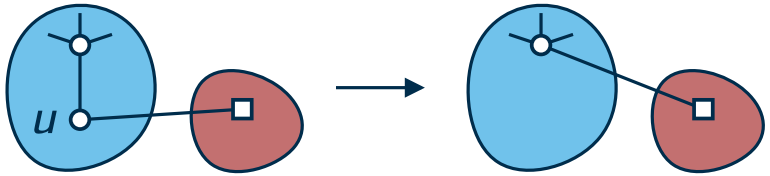
Case 1: u has no neighbor in Y

- u does not lie on a cycle
- **reduction rule:** delete u



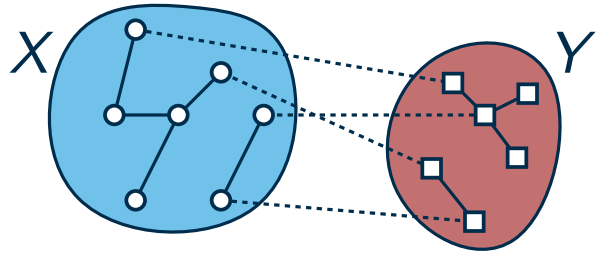
Case 2: u has one neighbor in Y

- u has degree 2 in G
- **reduction rule:** replace u by an edge



An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

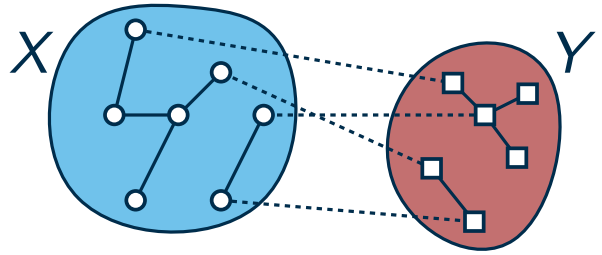
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ← **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Size of the search tree

- $v \in F$ reduces k, $v \notin F$ reduces number of components in Y

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

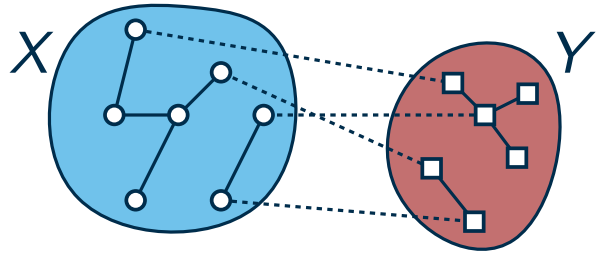
- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Size of the search tree

- $v \in F$ reduces k, $v \notin F$ reduces number of components in Y $\Rightarrow$ height $\leq 2k$ $\Rightarrow \leq 4^k$ leaves

An algorithm for DISJOINT FVS

Problem: DISJOINT FVS (undirected)
Given an undirected graph, a parameter k, and a FVS Y of size $k+1$. Is there a FVS F of size k with $F \cap Y = \emptyset$?



Note: $G[X]$ and $G[Y]$ are acyclic

Plan

- choose vertices from $X = V \setminus Y$ and none from Y
- branch on a vertex $v \in X$ **with $\geq$ 2 neighbors in Y** ⟵ **todo:** show that this is always possible
- or: reduce a vertex $u \in X$ with only one neighbor in X

Size of the search tree

- $v \in F$ reduces k, $v \notin F$ reduces number of components in Y $\Rightarrow$ height $\leq 2k$ $\Rightarrow \leq 4^k$ leaves

Theorem
FEEDBACK VERTEX SET (undirected) can be solved in $5^k \cdot n^{O(1)}$ time.

Wrap-Up

Theorem
FEEDBACK VERTEX SET can be solved in $O(2^k \cdot n^2 \log n)$ time on tournament graphs.

Theorem
FEEDBACK VERTEX SET (undirected) can be solved in $5^k \cdot n^{O(1)}$ time.

Wrap-Up

Theorem
FEEDBACK VERTEX SET can be solved in $O(2^k \cdot n^2 \log n)$ time on tournament graphs.

Theorem
FEEDBACK VERTEX SET (undirected) can be solved in $5^k \cdot n^{O(1)}$ time.

Technique: iterative compression

- show: PROBLEM $\in$ FPT $\Leftarrow$ PROBLEM COMPRESSION $\in$ FPT $\Leftarrow$ DISJOINT PROBLEM $\in$ FPT
- then solve DISJOINT PROBLEM (this is the difficult part!)

Wrap-Up

Theorem
FEEDBACK VERTEX SET can be solved in $O(2^k \cdot n^2 \log n)$ time on tournament graphs.

Theorem
FEEDBACK VERTEX SET (undirected) can be solved in $5^k \cdot n^{O(1)}$ time.

Technique: iterative compression

- show: PROBLEM $\in$ FPT $\Leftarrow$ PROBLEM COMPRESSION $\in$ FPT $\Leftarrow$ DISJOINT PROBLEM $\in$ FPT
- then solve DISJOINT PROBLEM (this is the difficult part!)

Why is DISJOINT PROBLEM easier than PROBLEM?

Wrap-Up

Theorem
FEEDBACK VERTEX SET can be solved in $O(2^k \cdot n^2 \log n)$ time on tournament graphs.

Theorem
FEEDBACK VERTEX SET (undirected) can be solved in $5^k \cdot n^{O(1)}$ time.

Technique: iterative compression

- show: PROBLEM $\in$ FPT $\Leftarrow$ PROBLEM COMPRESSION $\in$ FPT $\Leftarrow$ DISJOINT PROBLEM $\in$ FPT
- then solve DISJOINT PROBLEM (this is the difficult part!)

Why is DISJOINT PROBLEM easier than PROBLEM?

- solution Y of size $k+1$ is known
- new solution must be disjoint to Y
- complement of Y is a solution