Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Advanced Algorithms

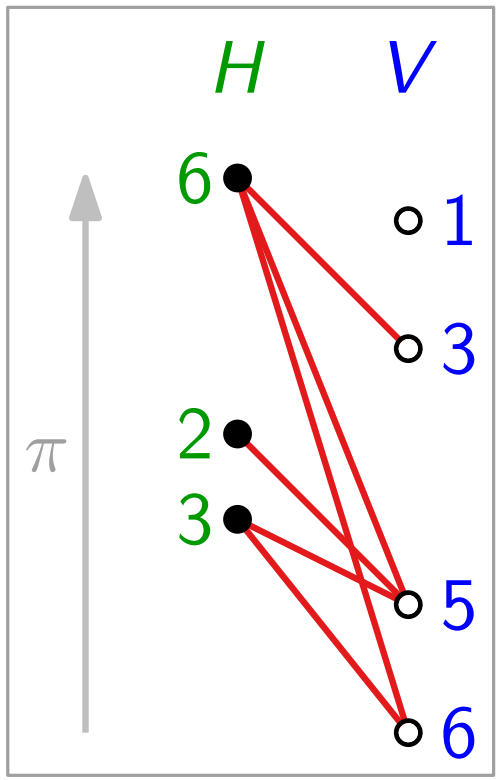
Winter term 2019/20

Lecture 7. Shortest Paths in Graphs with Negative Weights

Steven Chaplick & Alexander Wolff *Chair for Computer Science I*

Motivation

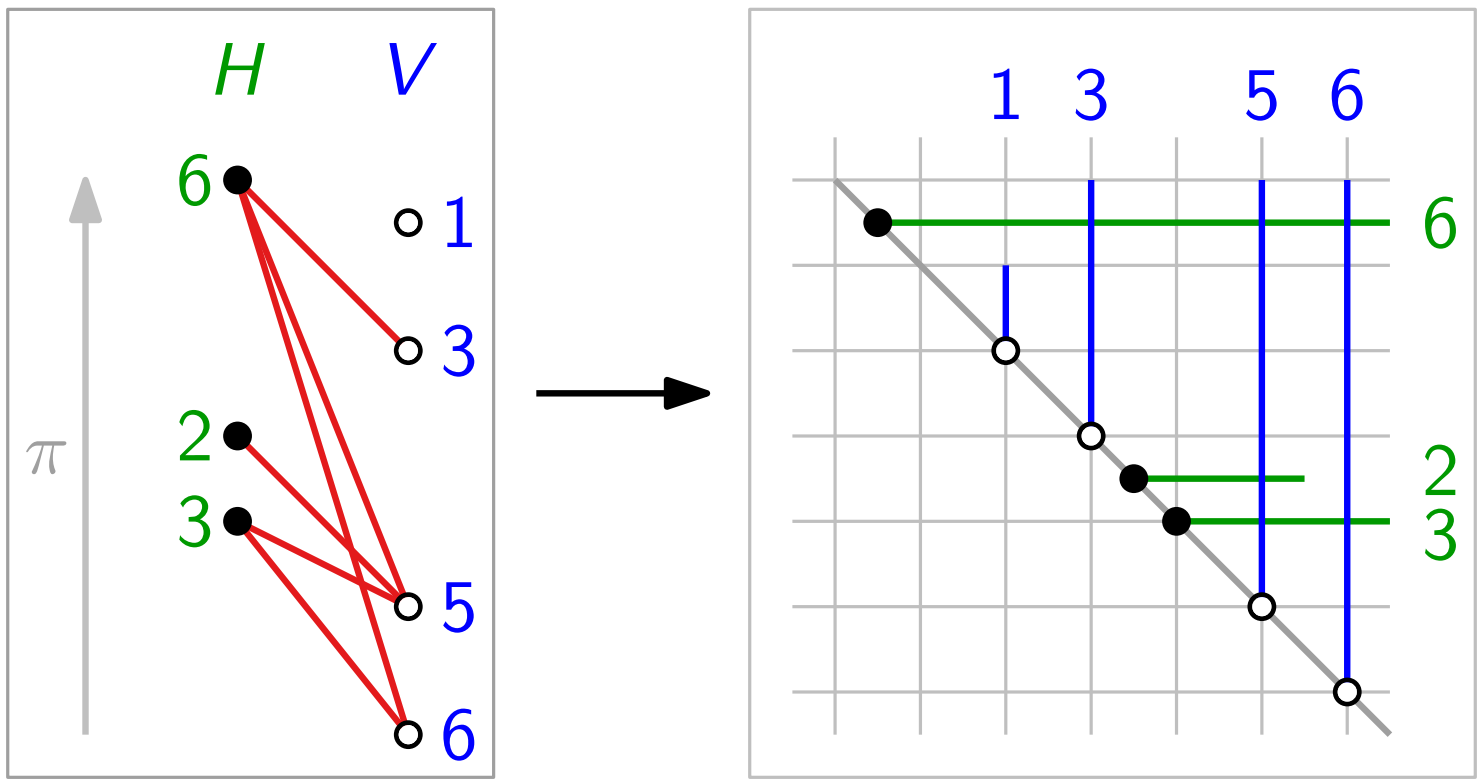
Problem. Let $G = (H \cup V, E)$ be a bipartite graph with positive vertex weights $\ell\colon H \cup V \to \mathbb{N}$ and a permutation π of $H \cup V$.



Motivation

Problem. Let $G = (H \cup V, E)$ be a bipartite graph with positive vertex weights $\ell\colon H \cup V \to \mathbb{N}$ and a permutation π of $H \cup V$.

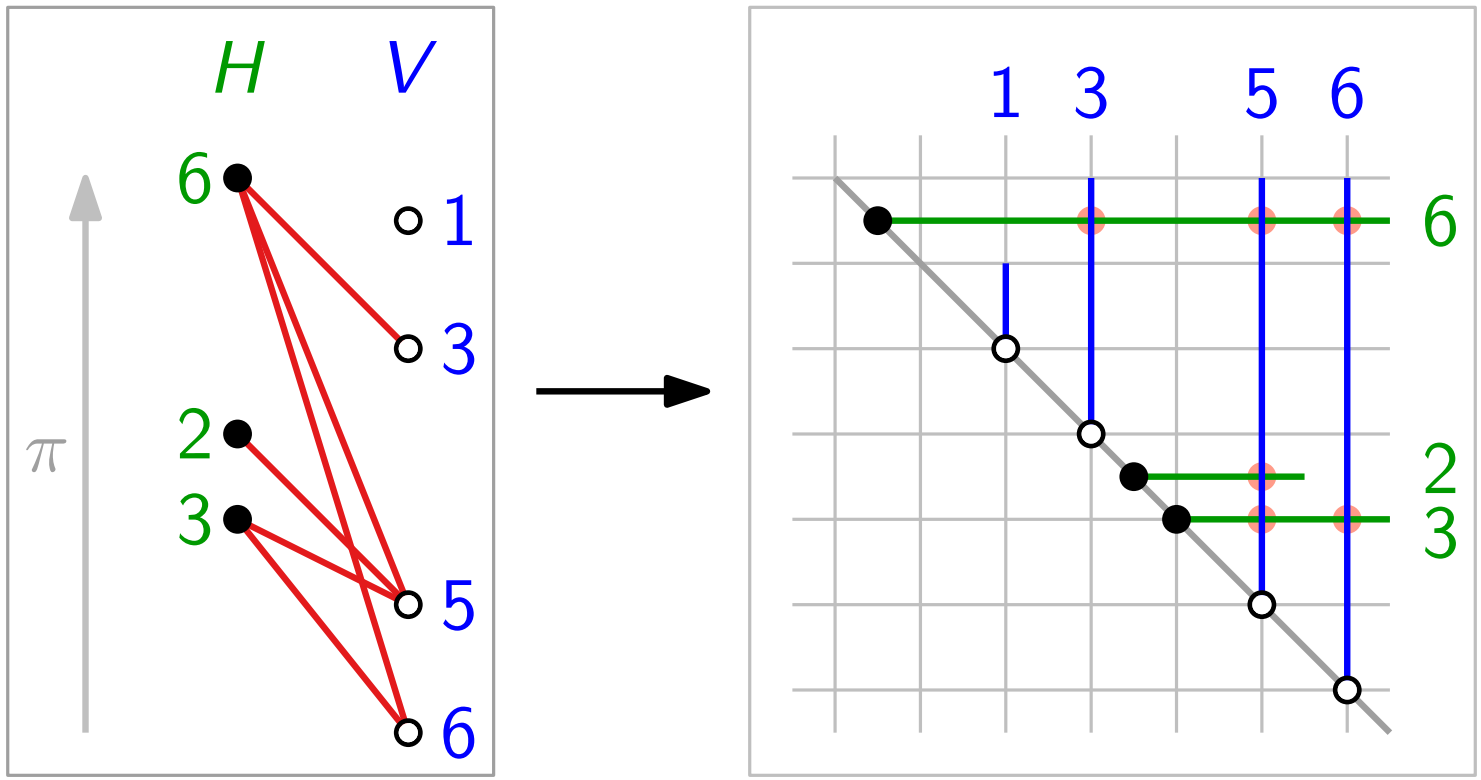
Does G admit a *stick representation* that respects the given stick order π and stick lengths ℓ?



Motivation

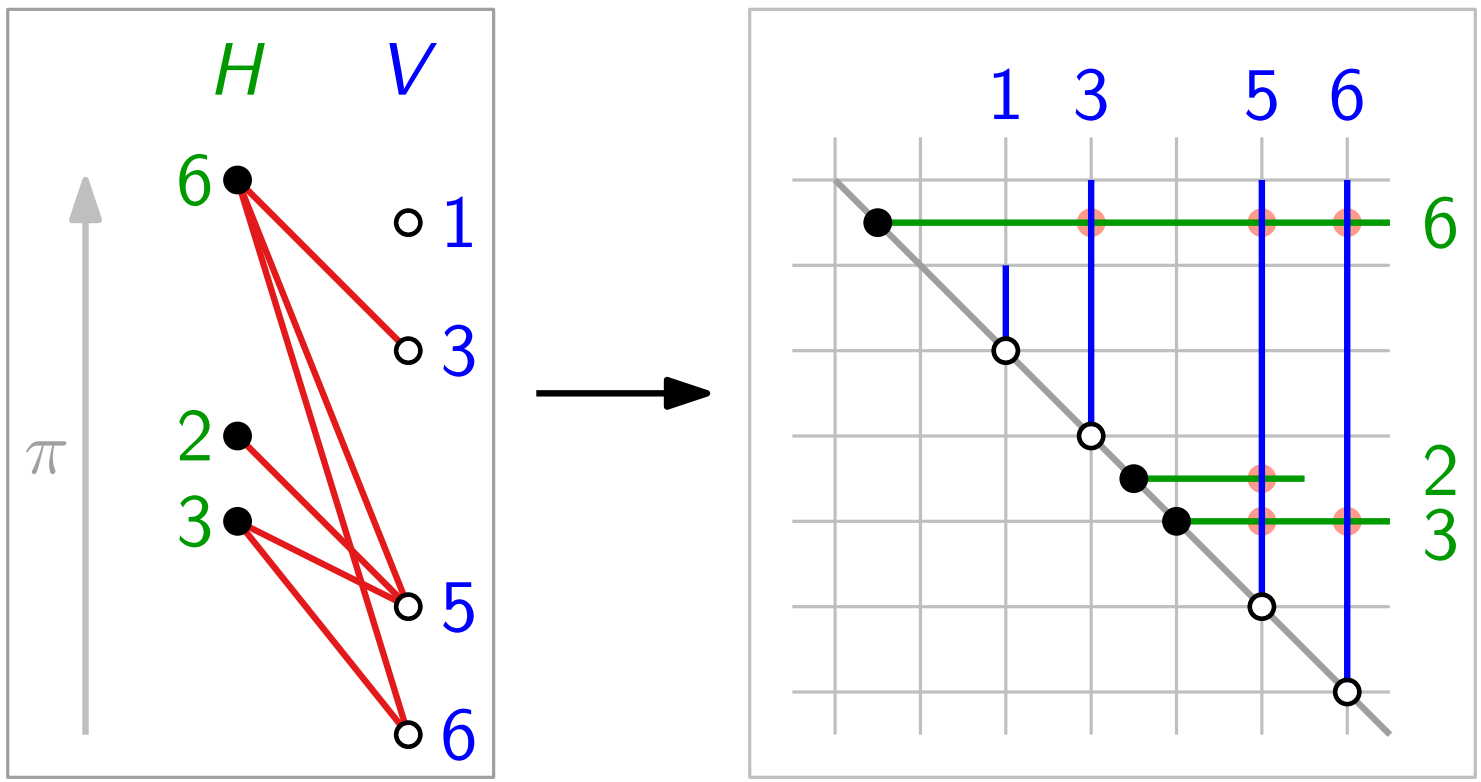
Problem. Let $G = (H \cup V, E)$ be a bipartite graph with positive vertex weights $\ell\colon H \cup V \to \mathbb{N}$ and a permutation π of $H \cup V$.

Does G admit a *stick representation* that respects the given stick order π and stick lengths ℓ?



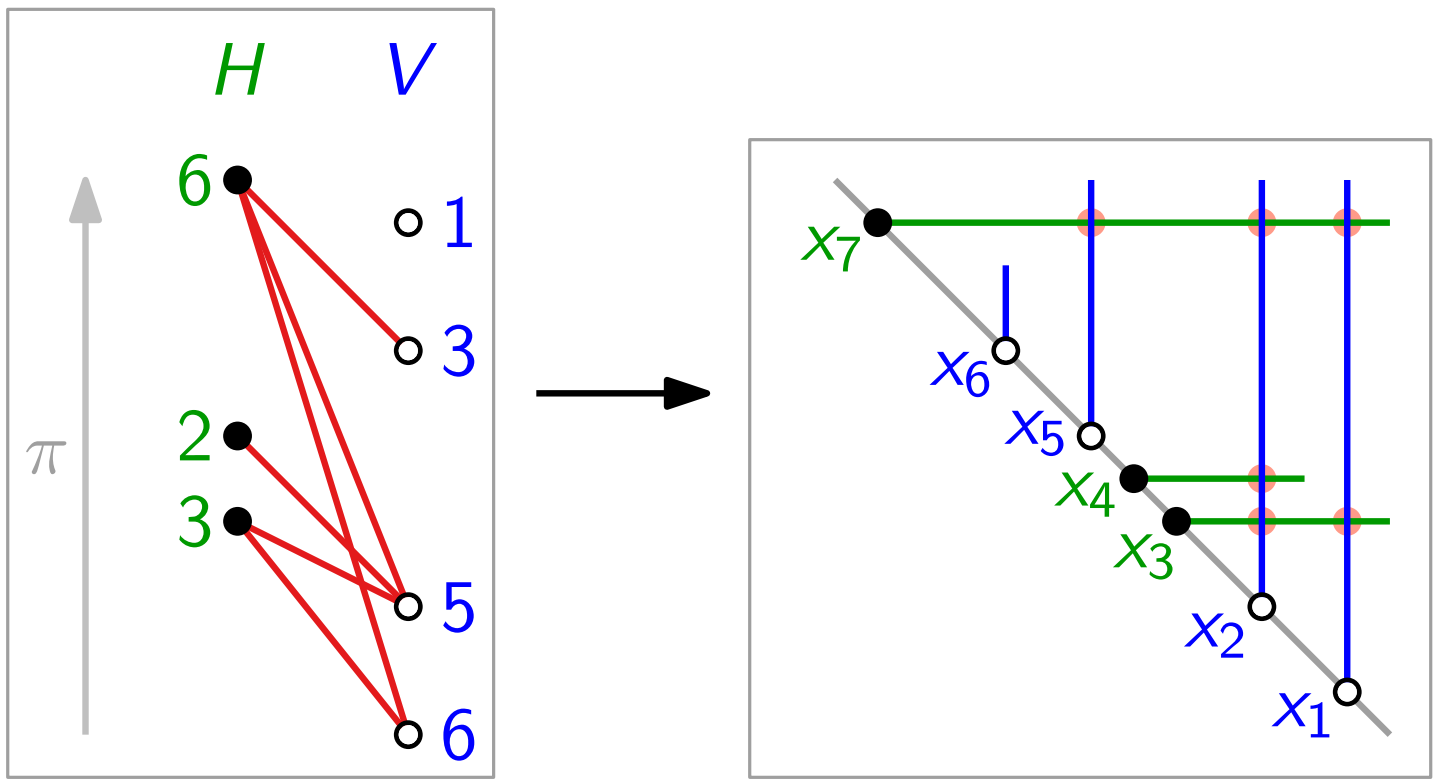
Motivation

Algorithm.



Motivation

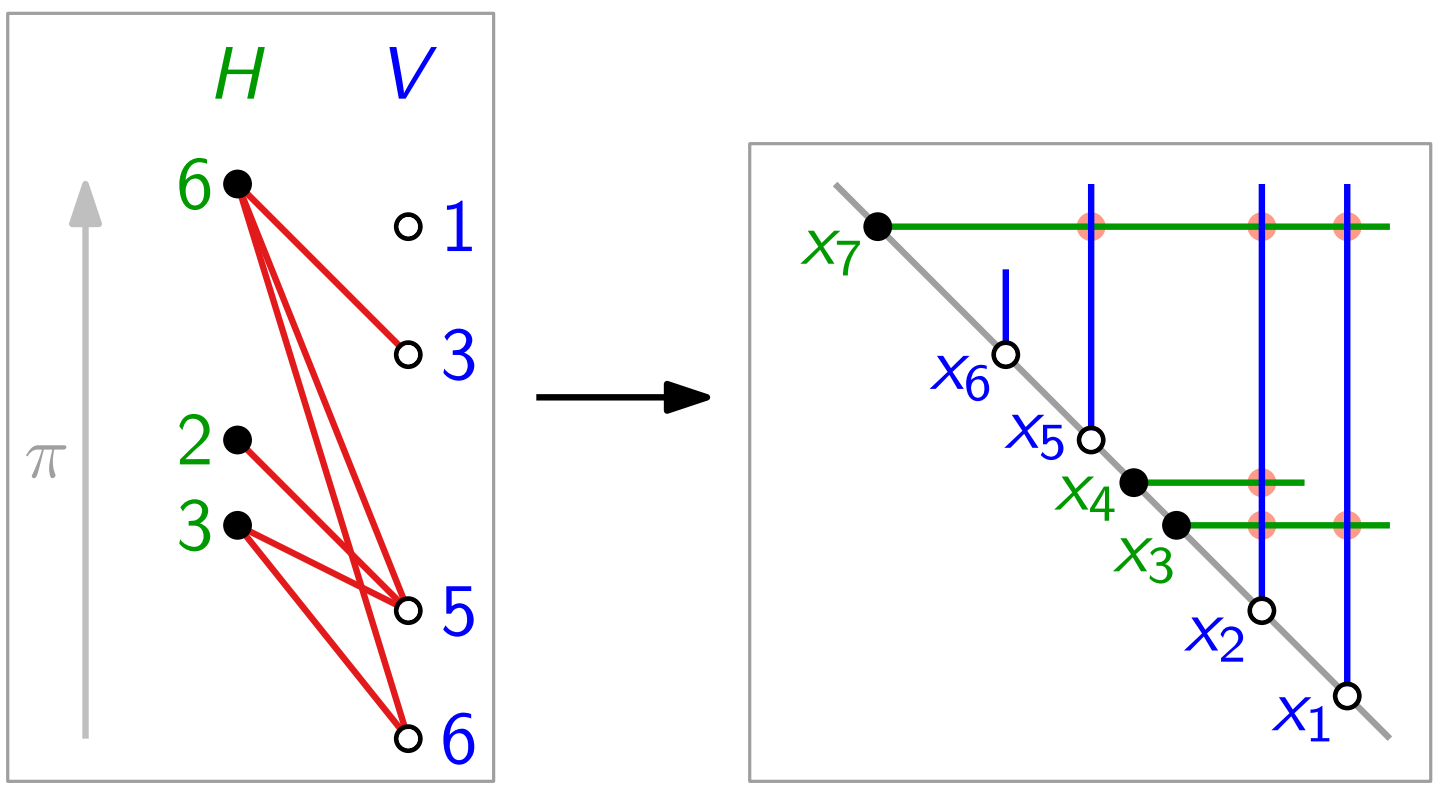
Algorithm.



Motivation

$n = |H| + |V|$

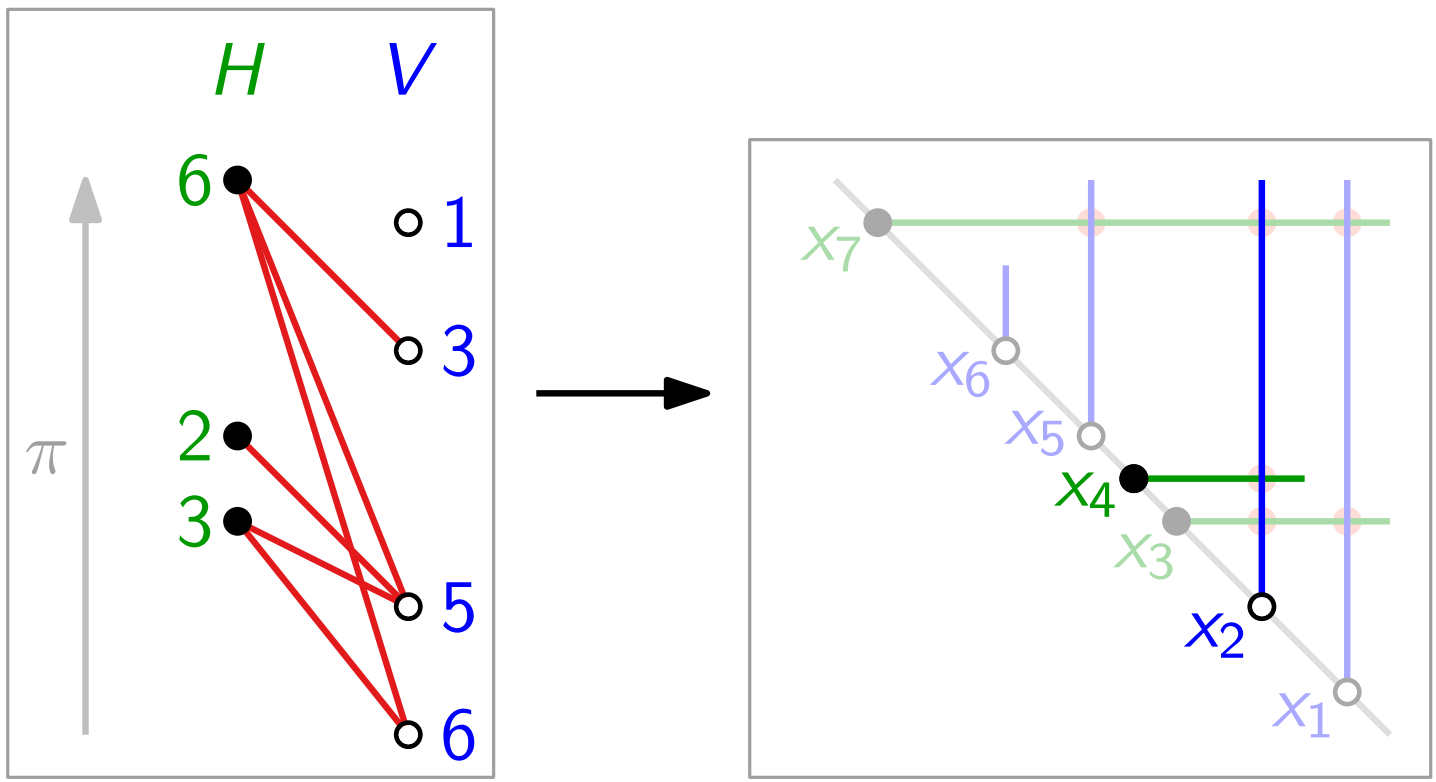
Algorithm. $x_n < \cdots < x_2 < x_1 = 0$



Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

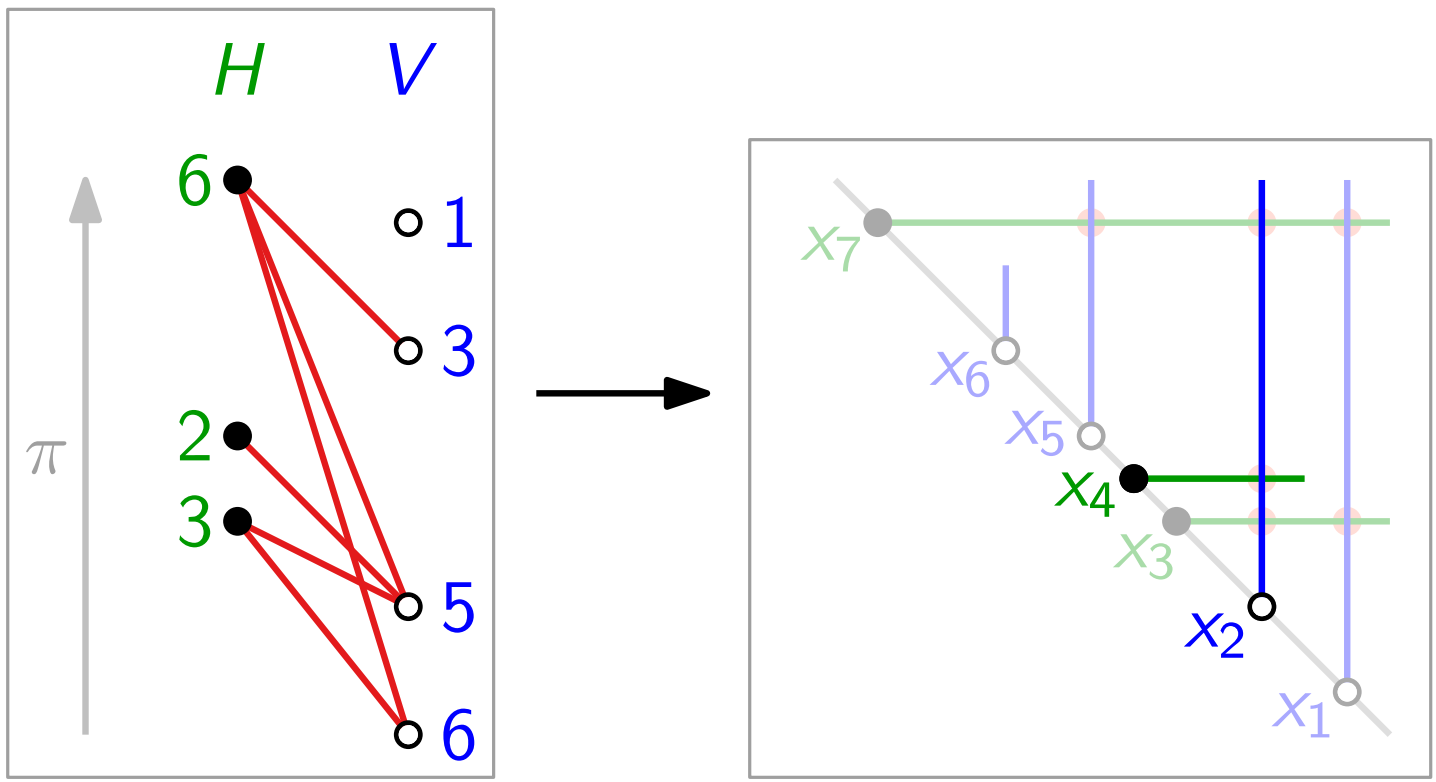


Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$:

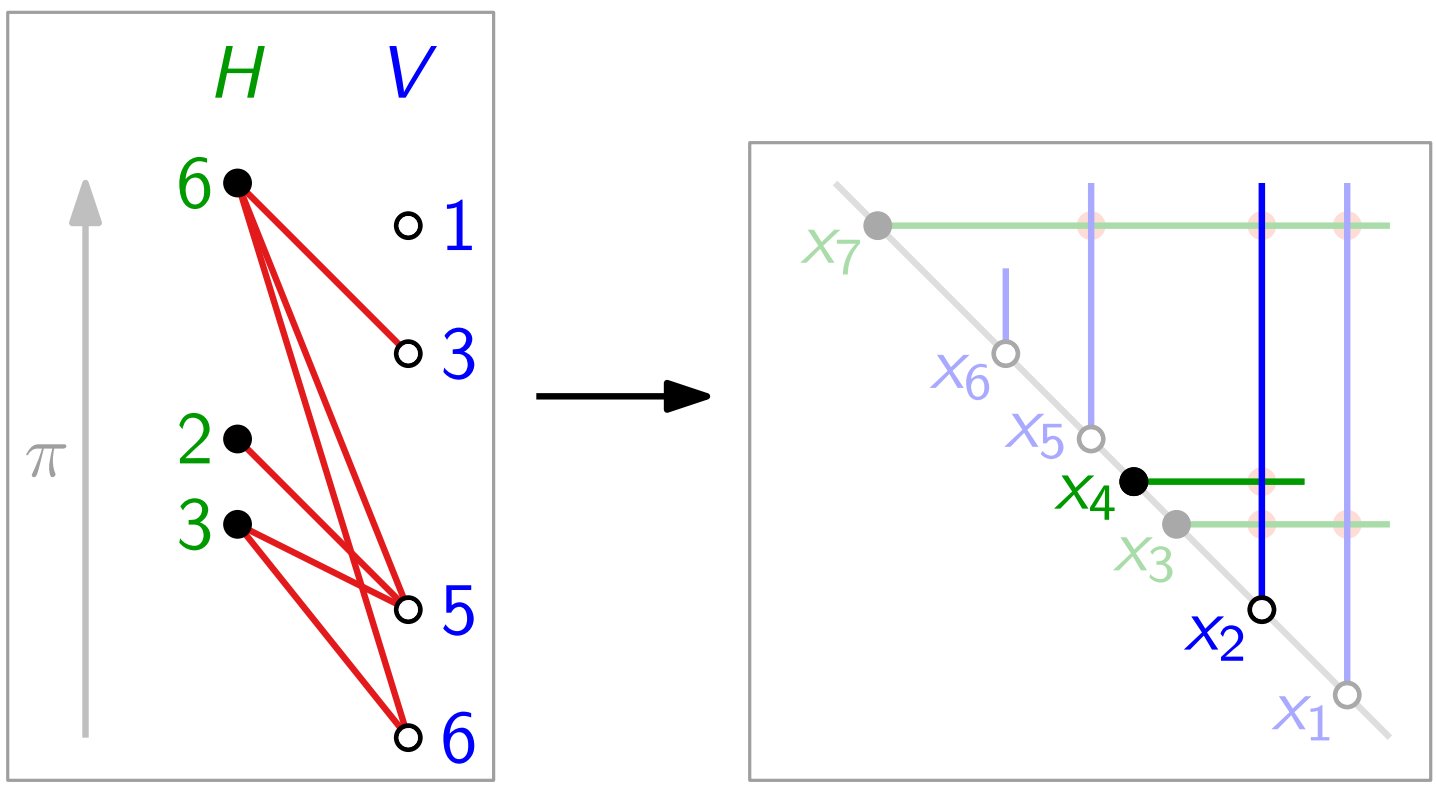


Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$



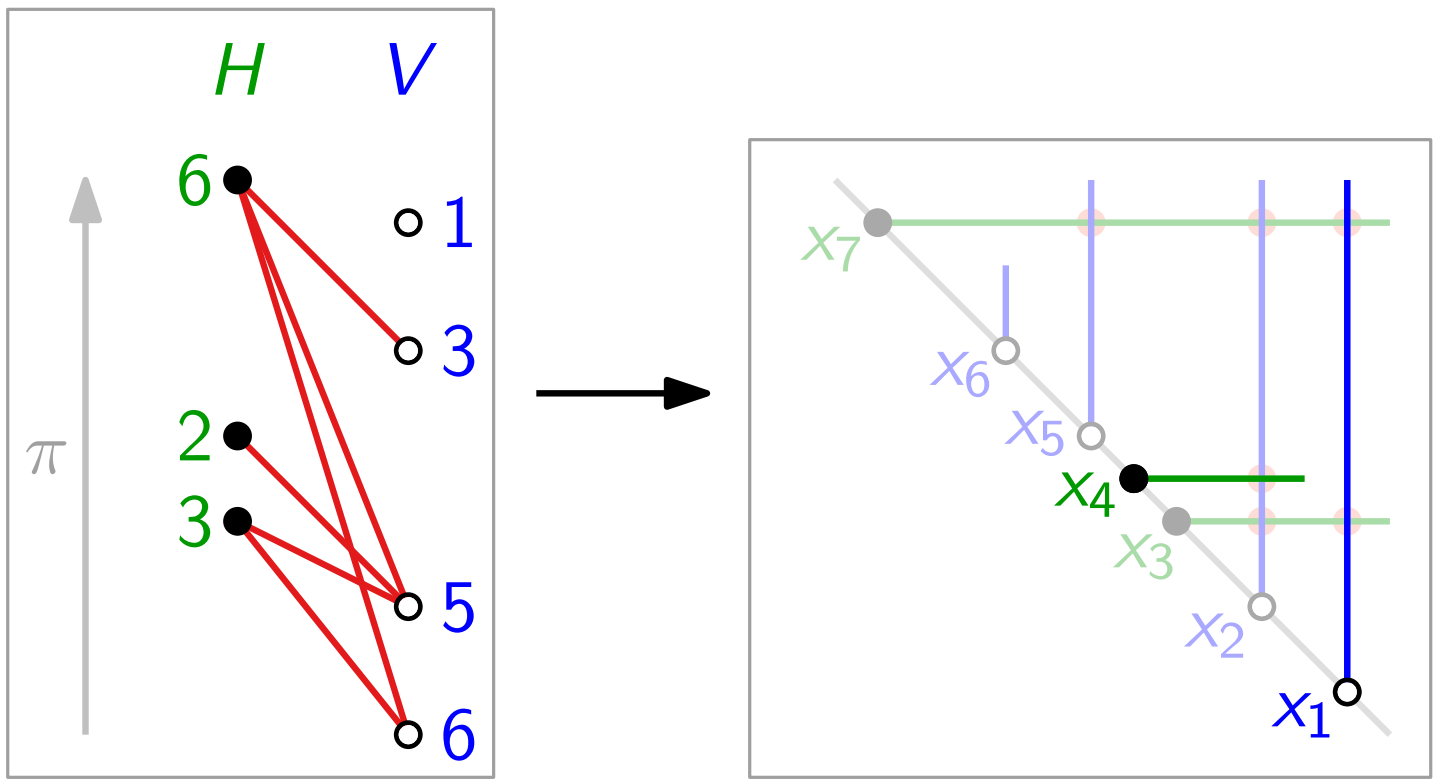
Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$:



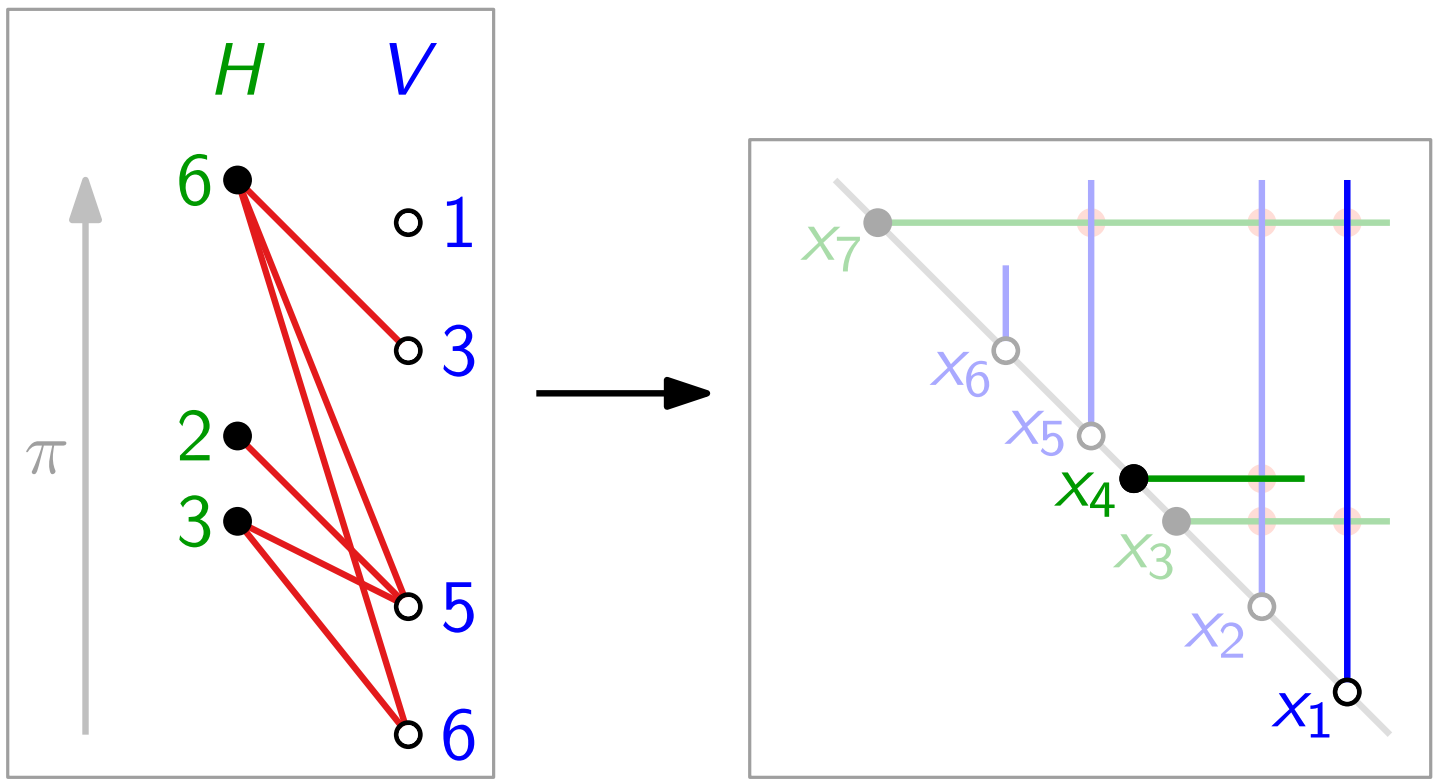
Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$



Motivation

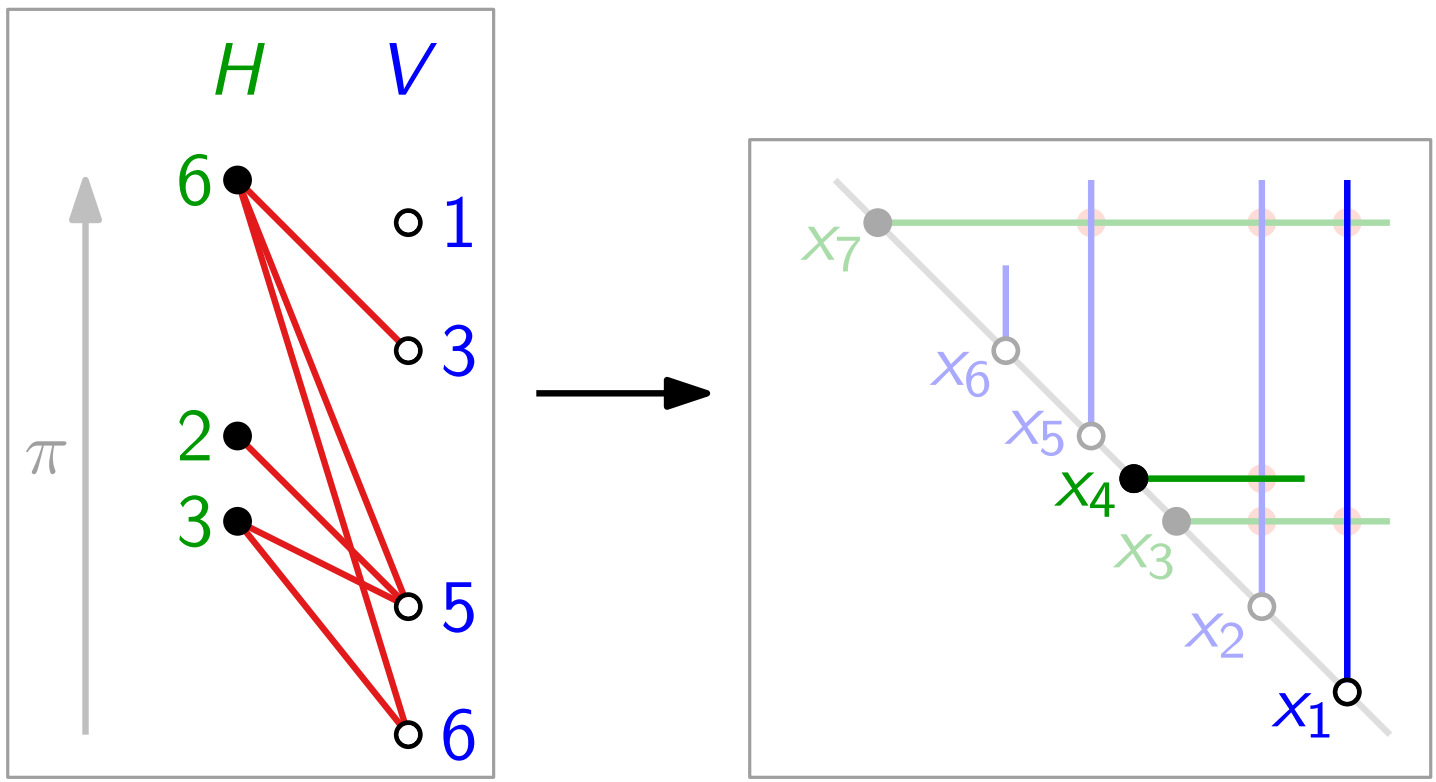
$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Solve LP



Motivation

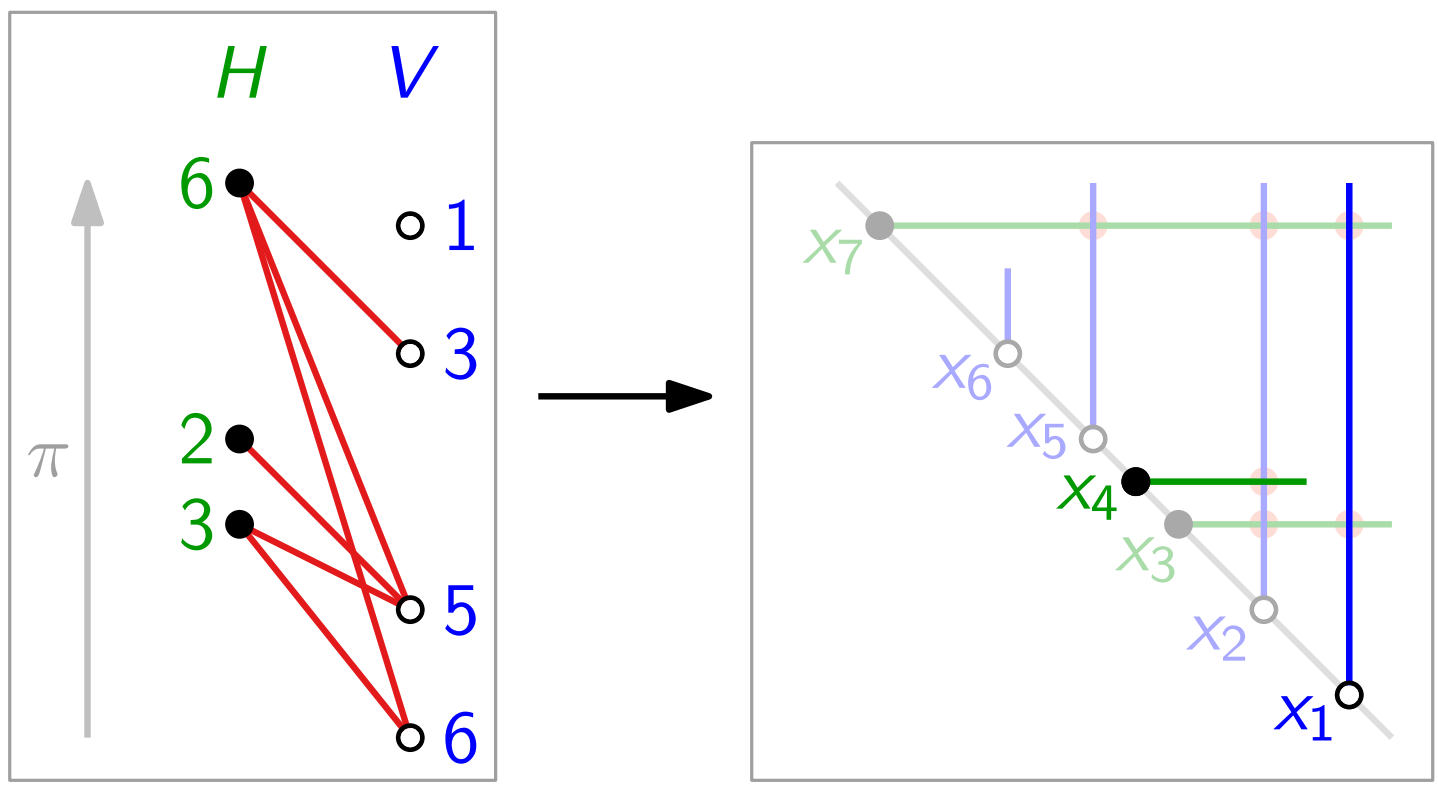
$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Solve LP (without objective function):



Motivation

$n = |H| + |V|$

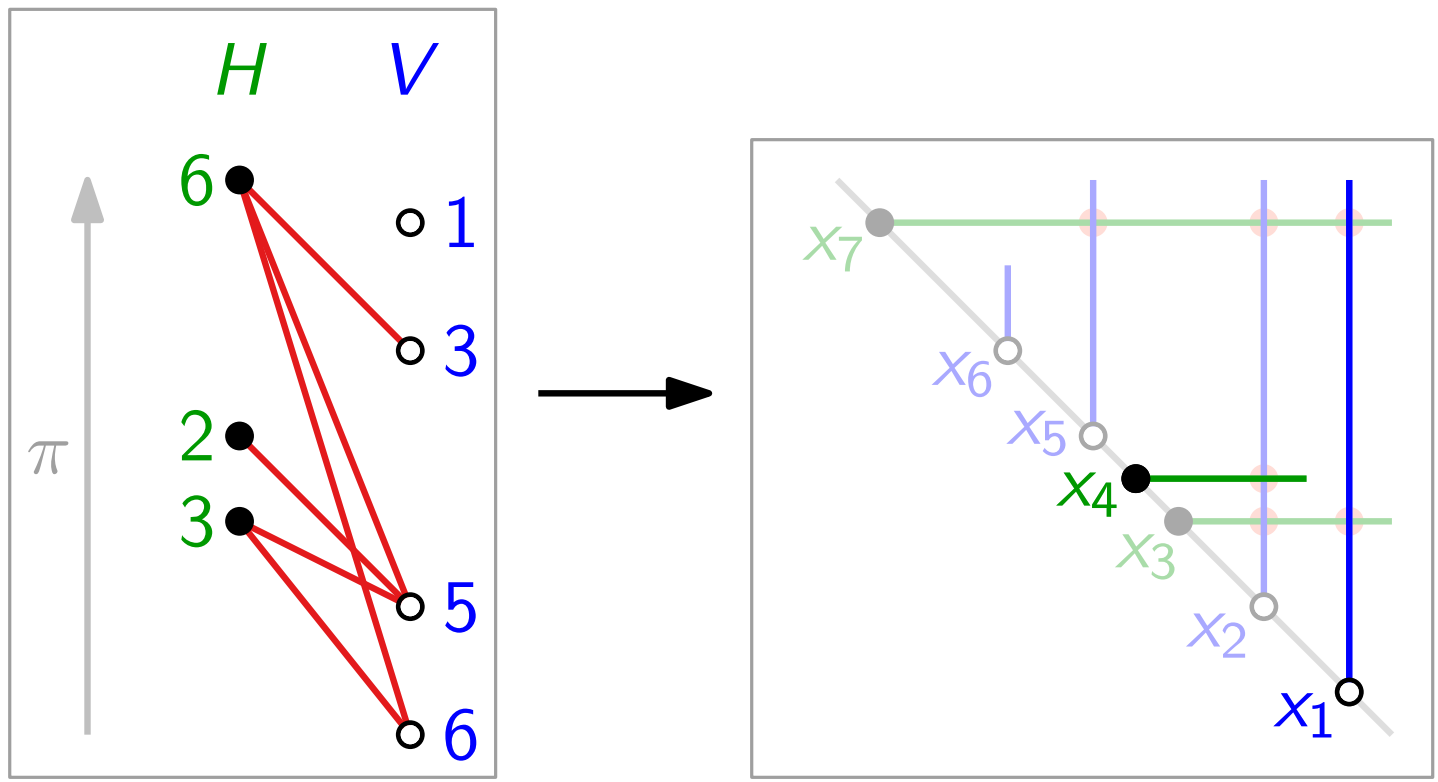
Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Solve LP (without objective function):

$\#$ variables $=$ $\#$ constraints $=$



Motivation

$n = |H| + |V|$

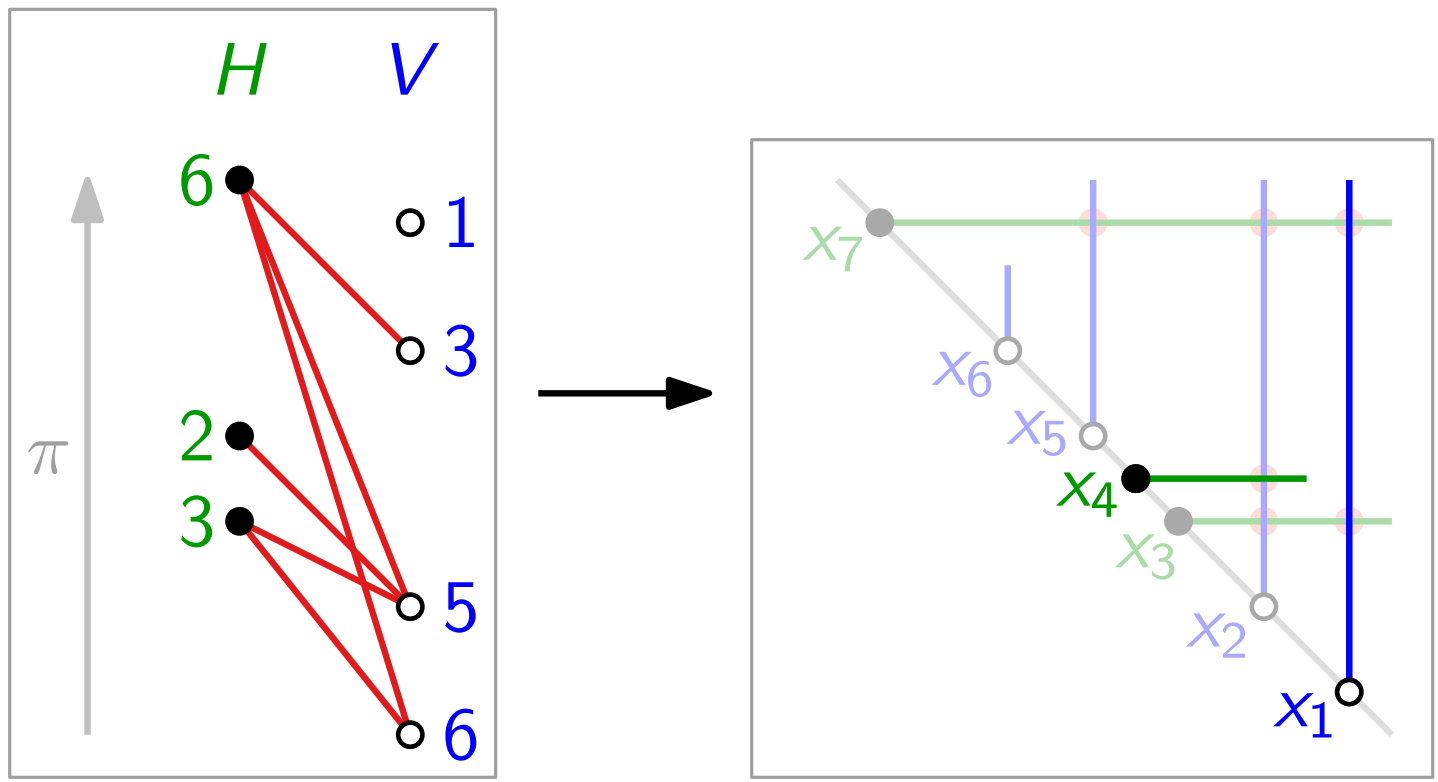
Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Solve LP (without objective function):

$\#$ variables $= n-1$ $\#$ constraints $=$



Motivation

$n = |H| + |V|$

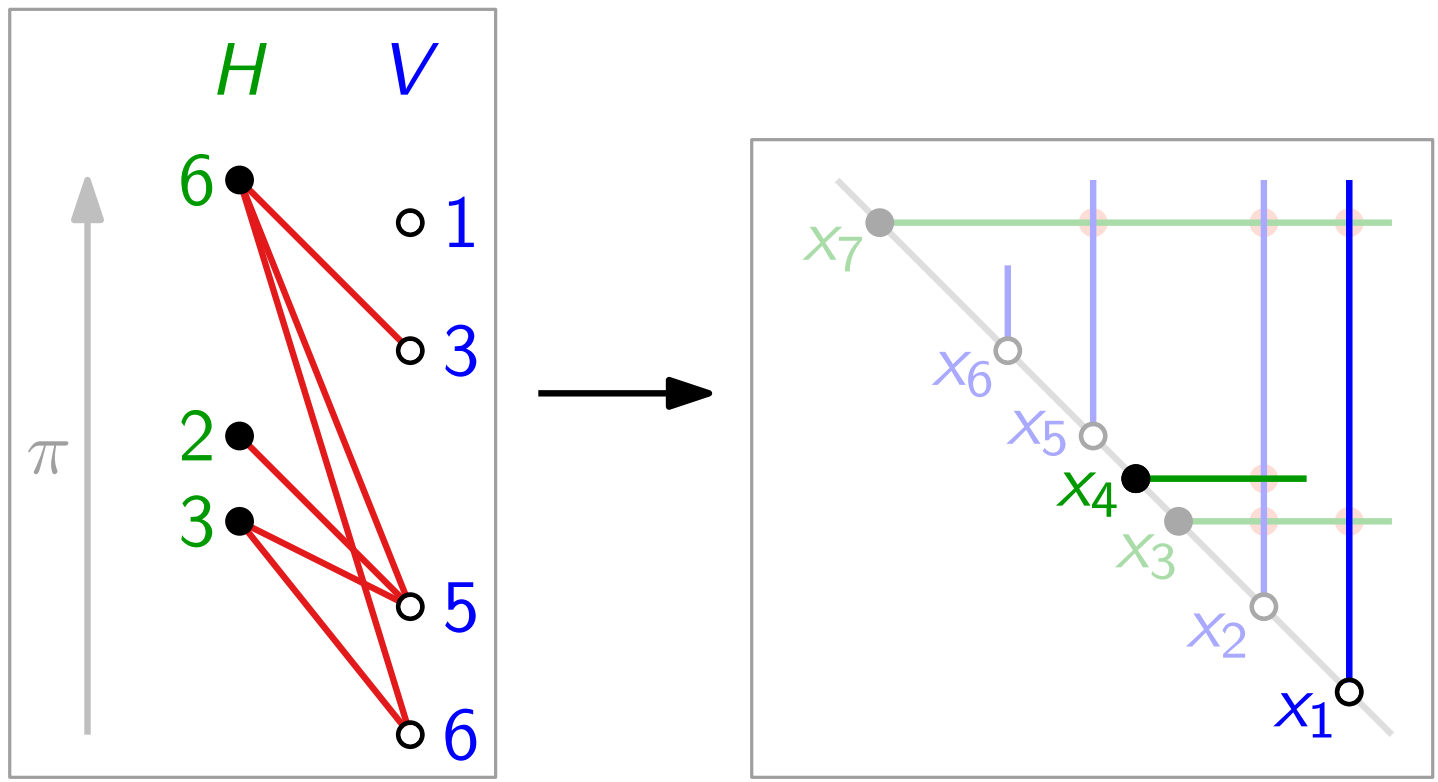
Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$



Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

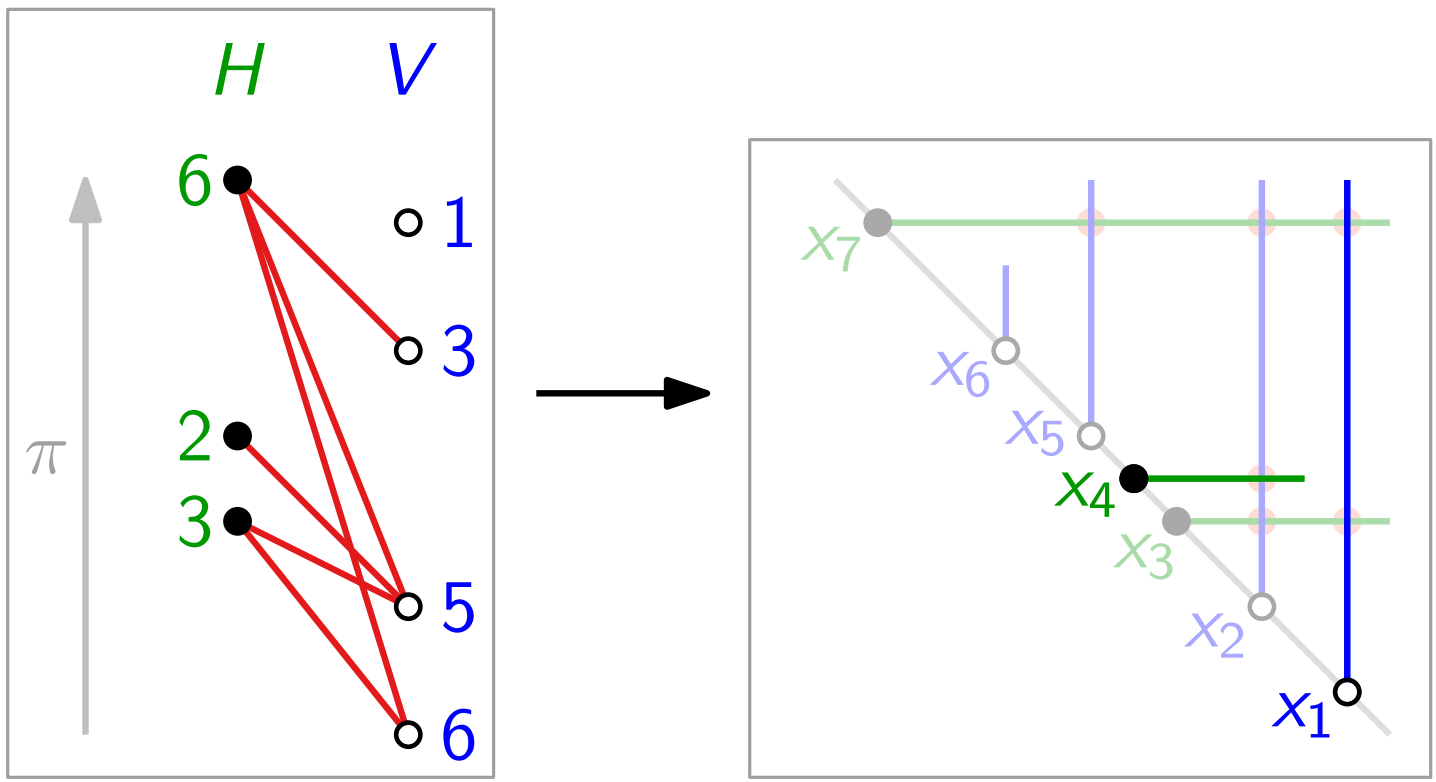
for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Exercise: Improve this to $O(|E| + n)$!

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$



Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

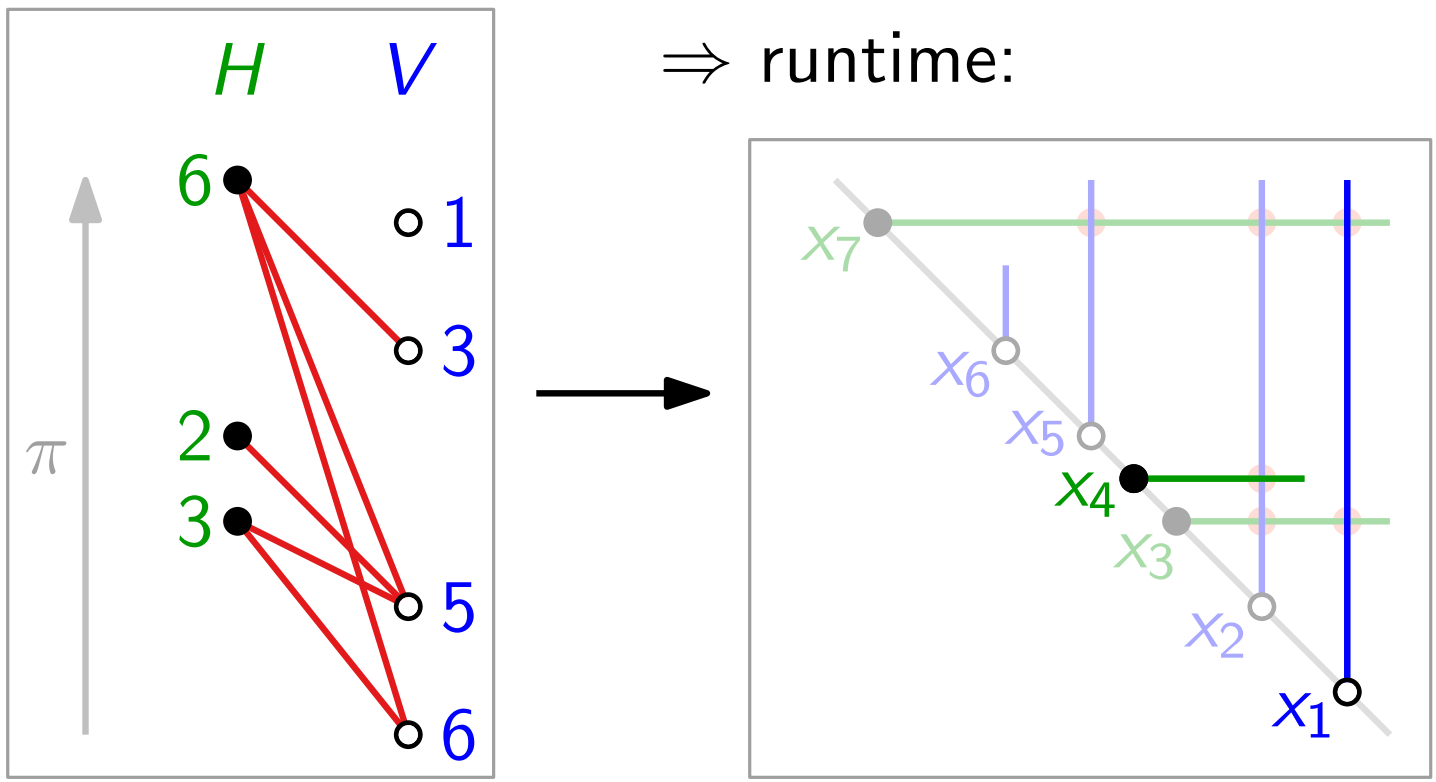
for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Exercise: Improve this to $O(|E| + n)$!

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$

$\Rightarrow$ runtime:



Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

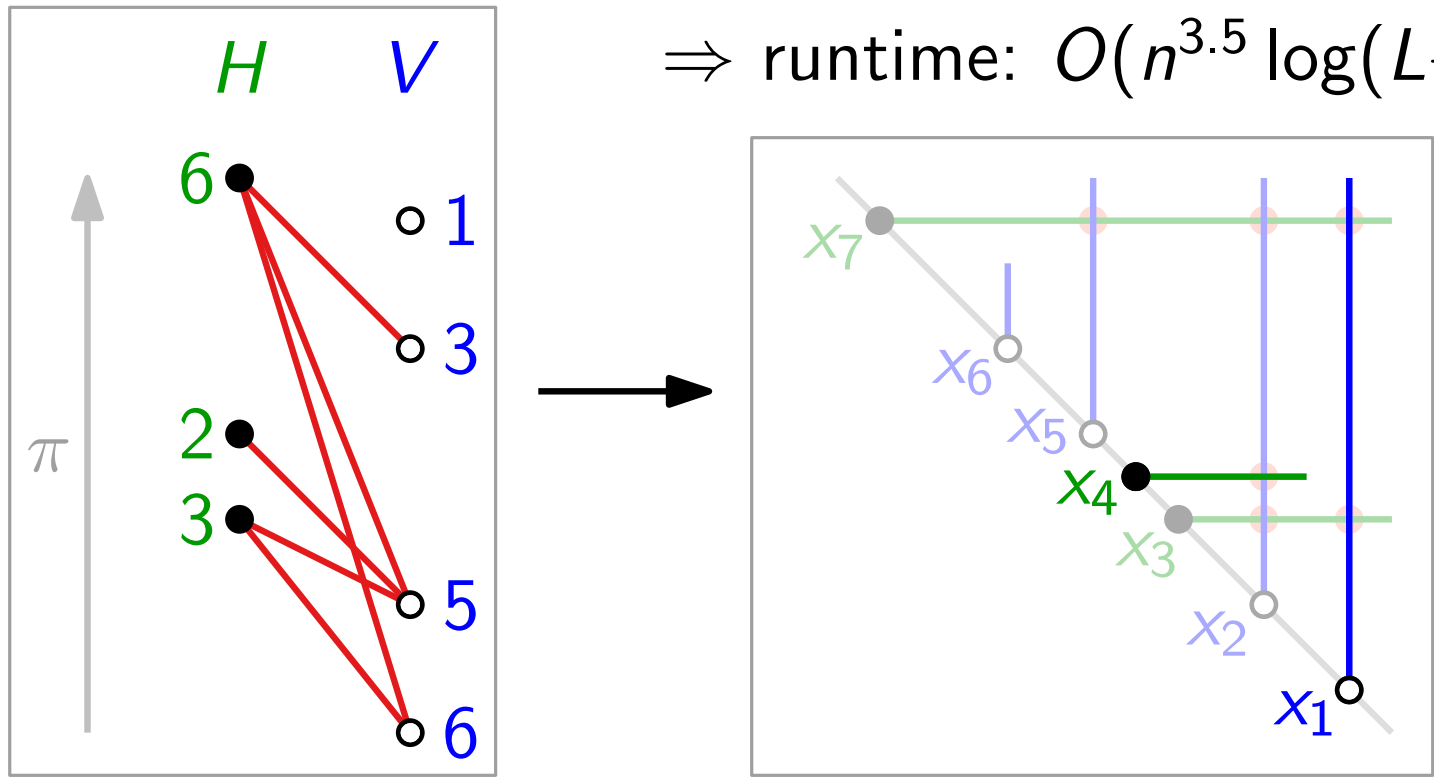
for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Exercise: Improve this to $O(|E| + n)$!

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$

$\Rightarrow$ runtime: $O(n^{3.5}\log(L+n))$



Motivation

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

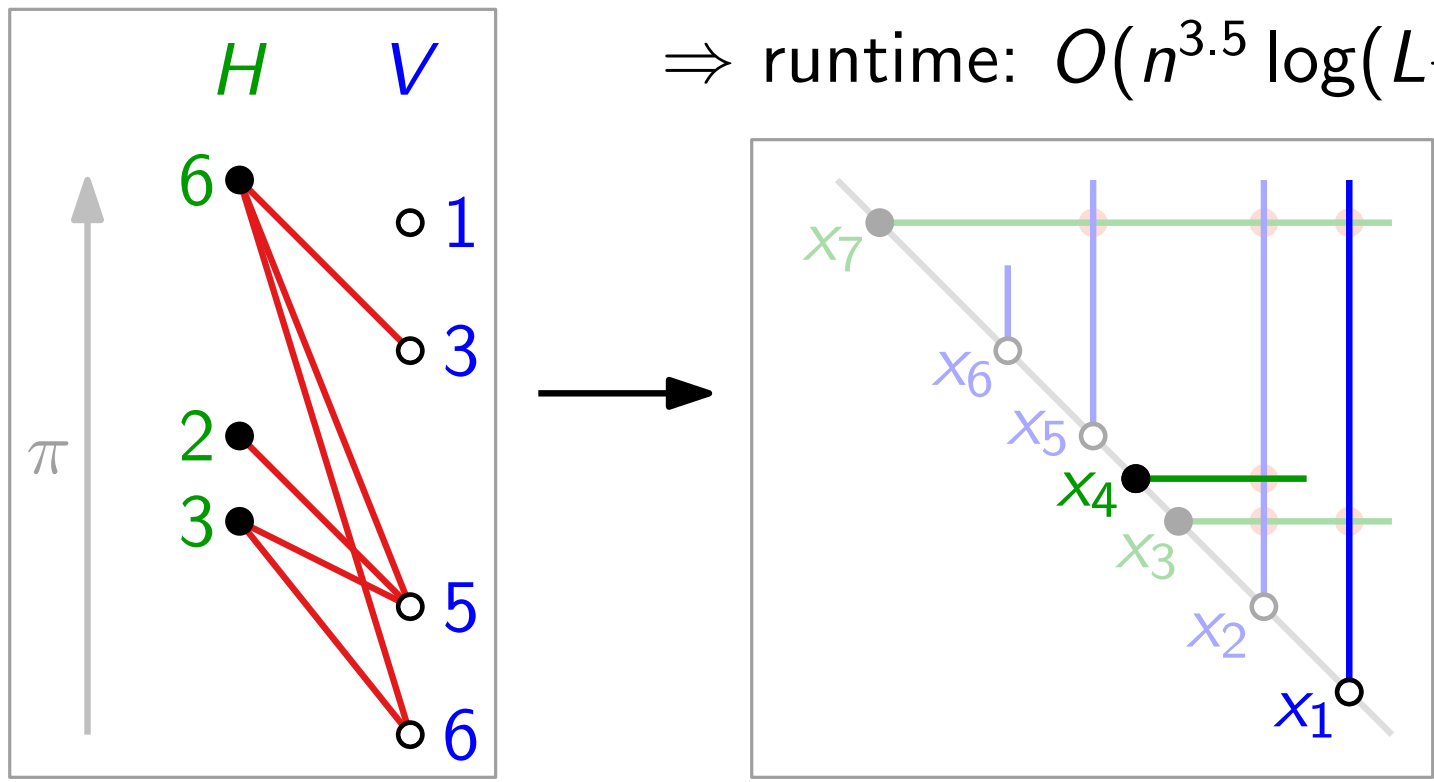
for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Exercise: Improve this to $O(|E| + n)$!

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$

$\Rightarrow$ runtime: $O(n^{3.5}\log(L+n))$



Motivation

difference constraints

$n = |H| + |V|$

Algorithm. $x_n < \cdots < x_2 < x_1 = 0$. Let $n \geq h > v \geq 1$.

for each $hv \in E$: $x_v - x_h < \min\{\ell_h, \ell_v\}$

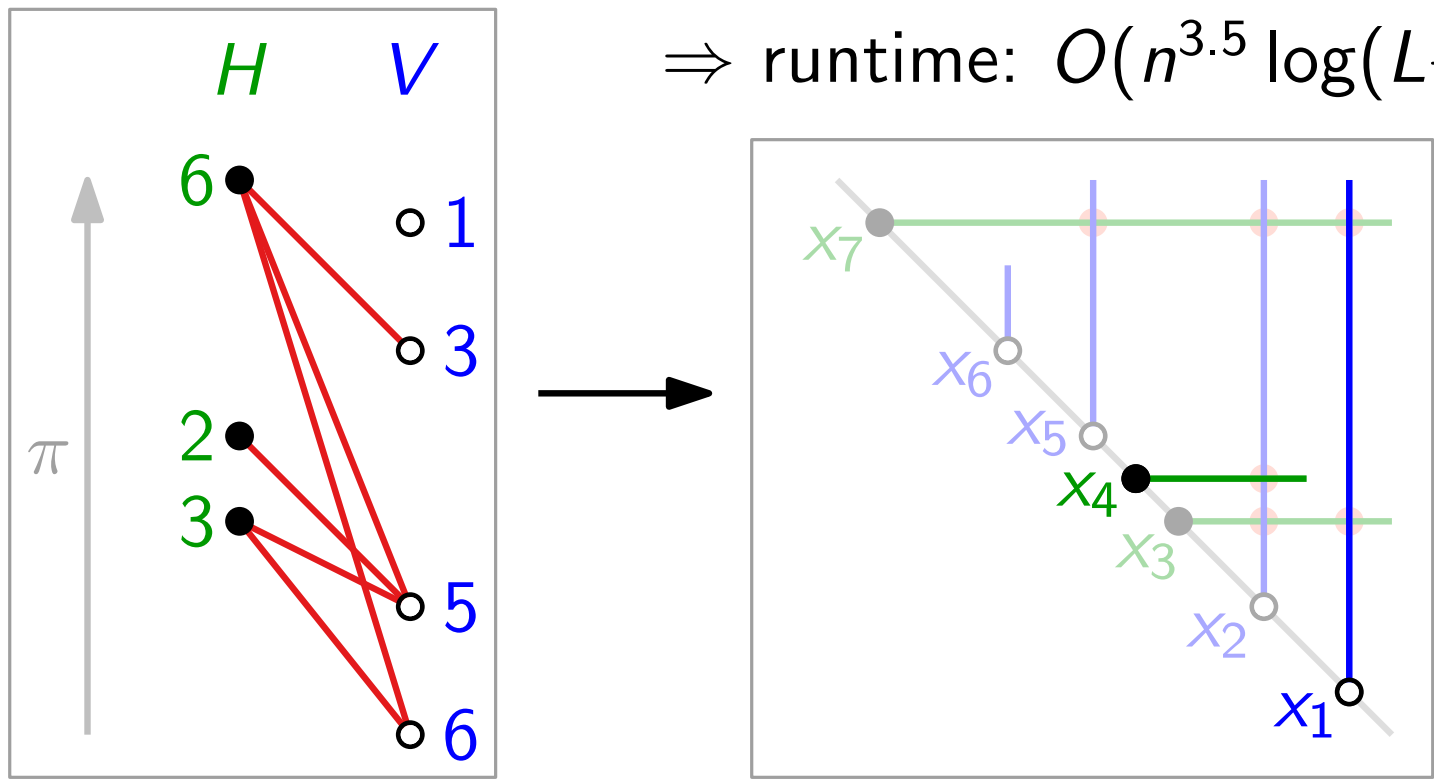
for each $hv \notin E$: $x_v - x_h > \min\{\ell_h, \ell_v\}$

Exercise: Improve this to $O(|E| + n)$!

Solve LP (without objective function):

$\#\,\text{variables} = n-1 \quad \#\,\text{constraints} = |H|\cdot|V|+n-1$

$\Rightarrow$ runtime: $O(n^{3.5}\log(L+n))$



Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$

Yes: Take $x = (-5, -3, 0, -1, 4)$

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$

Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Solving Systems of Difference Constraints

Is this system feasible?

$$\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}$$

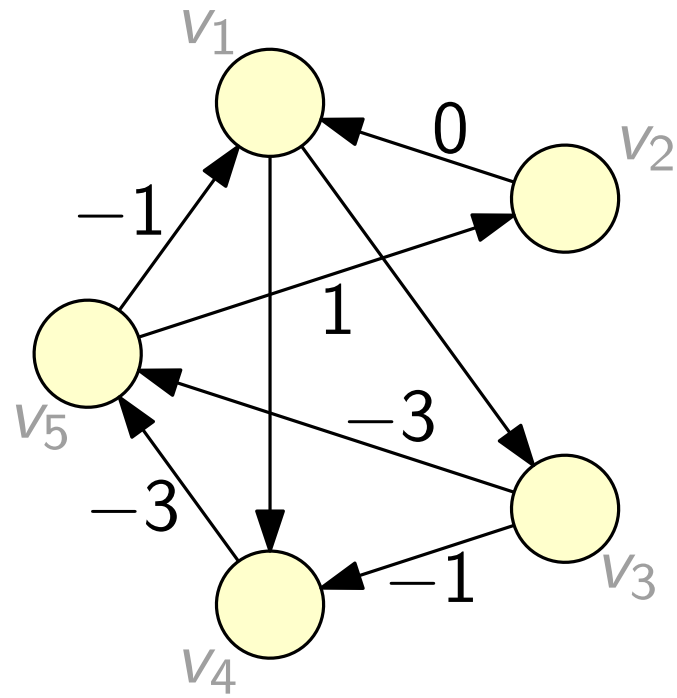
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



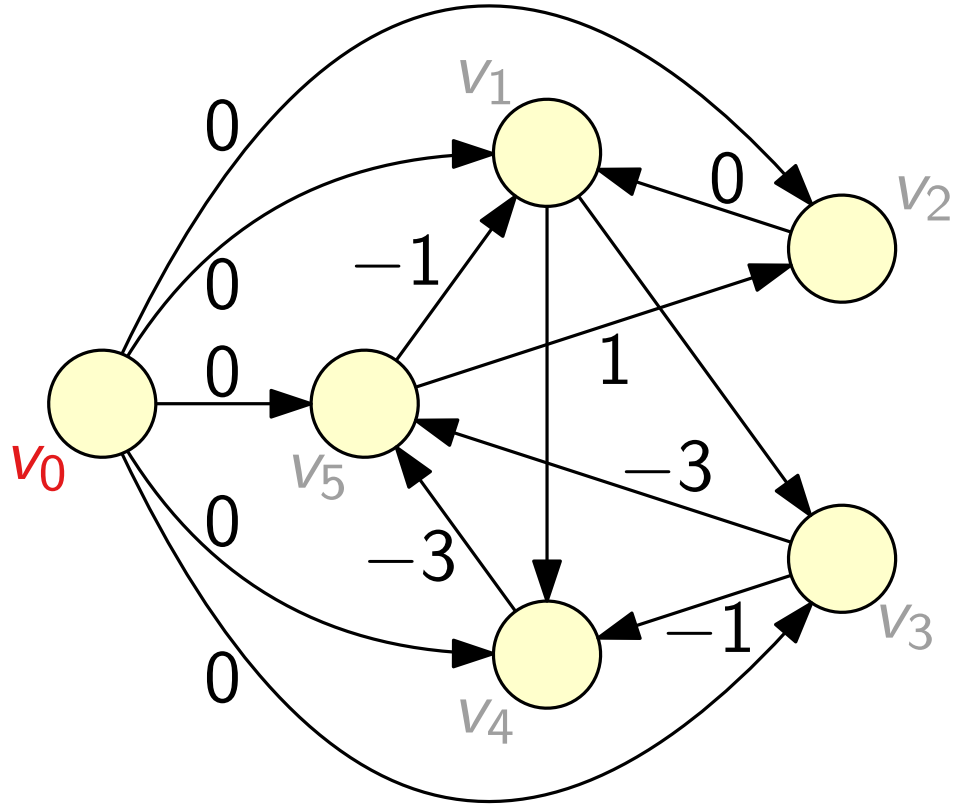
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



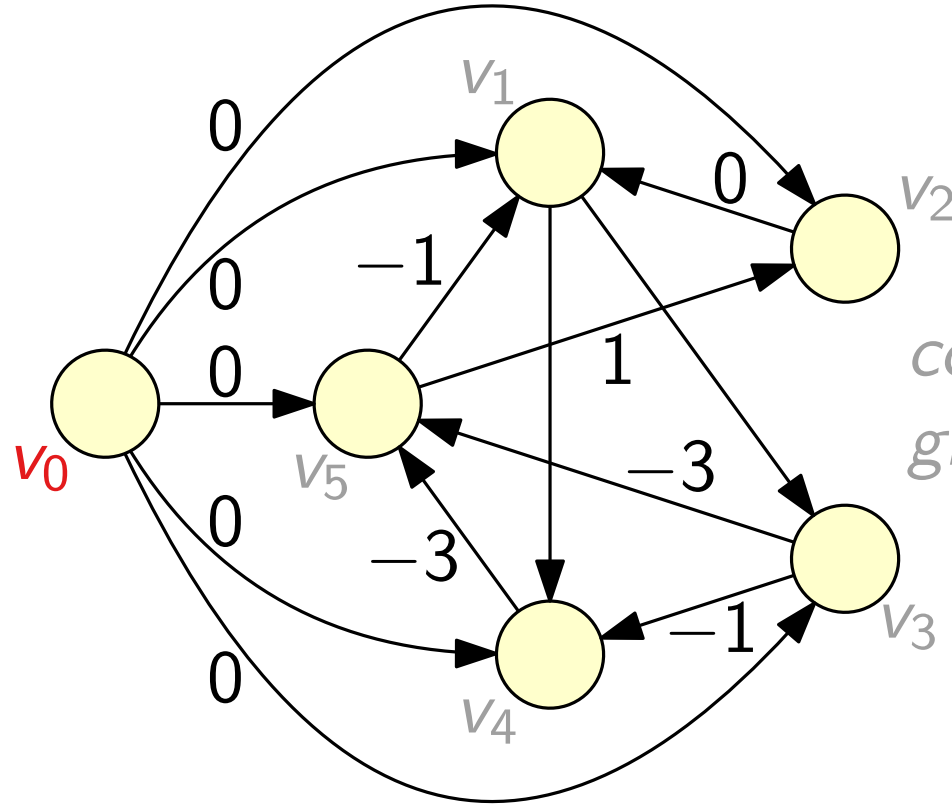
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



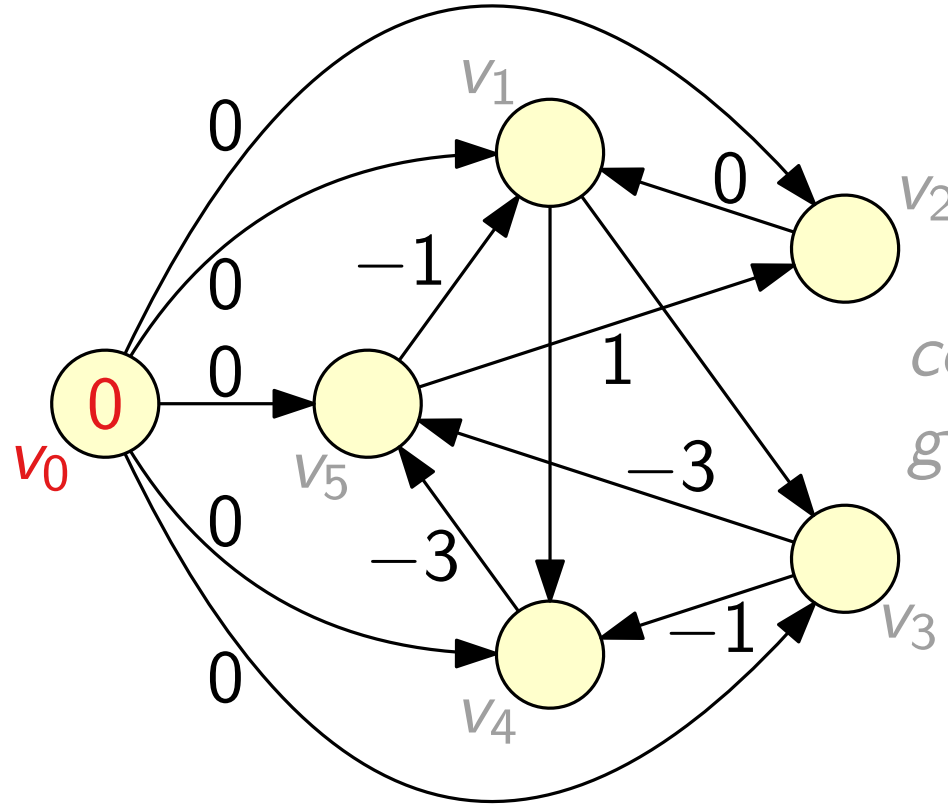
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



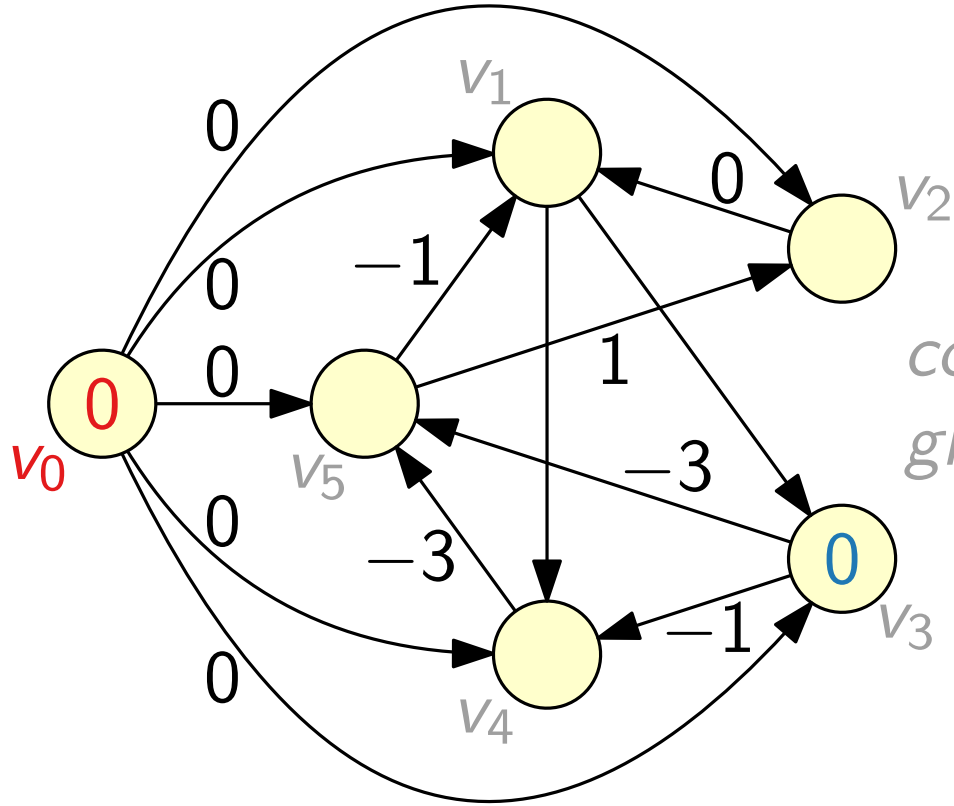
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



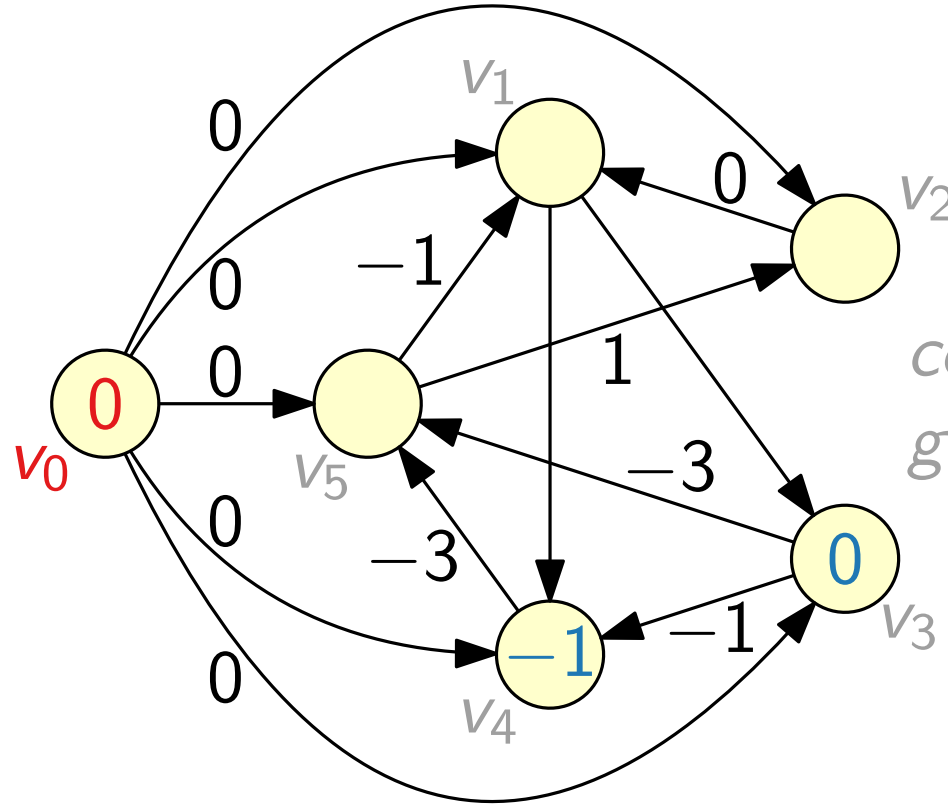
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



constraint graph $G_{A,b}$

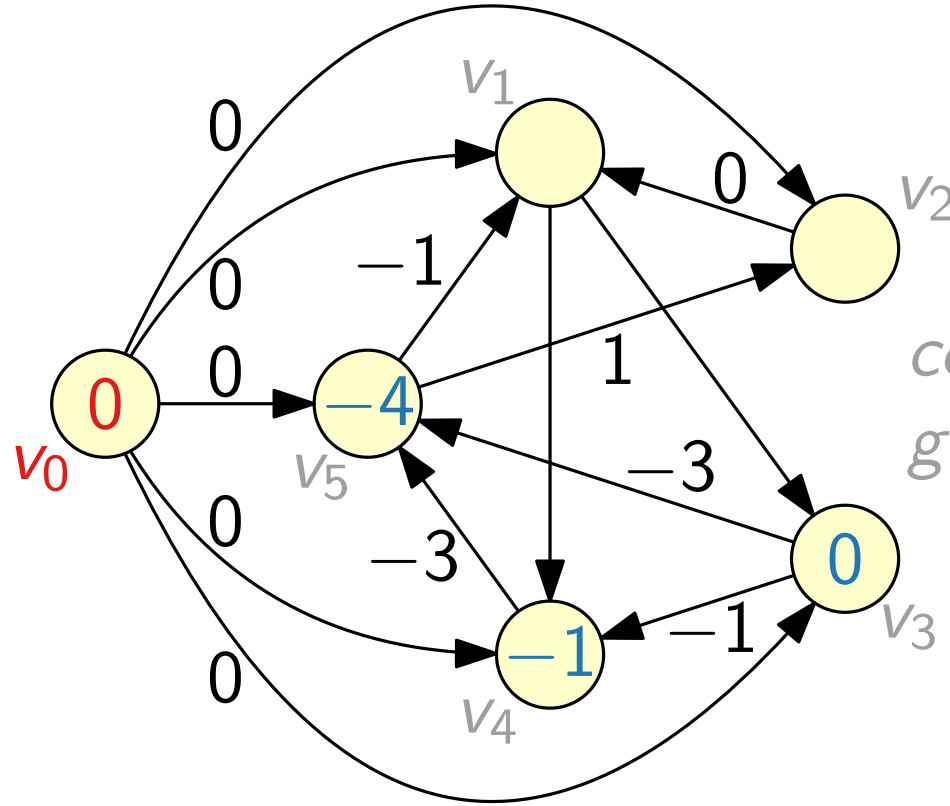
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



constraint graph $G_{A,b}$

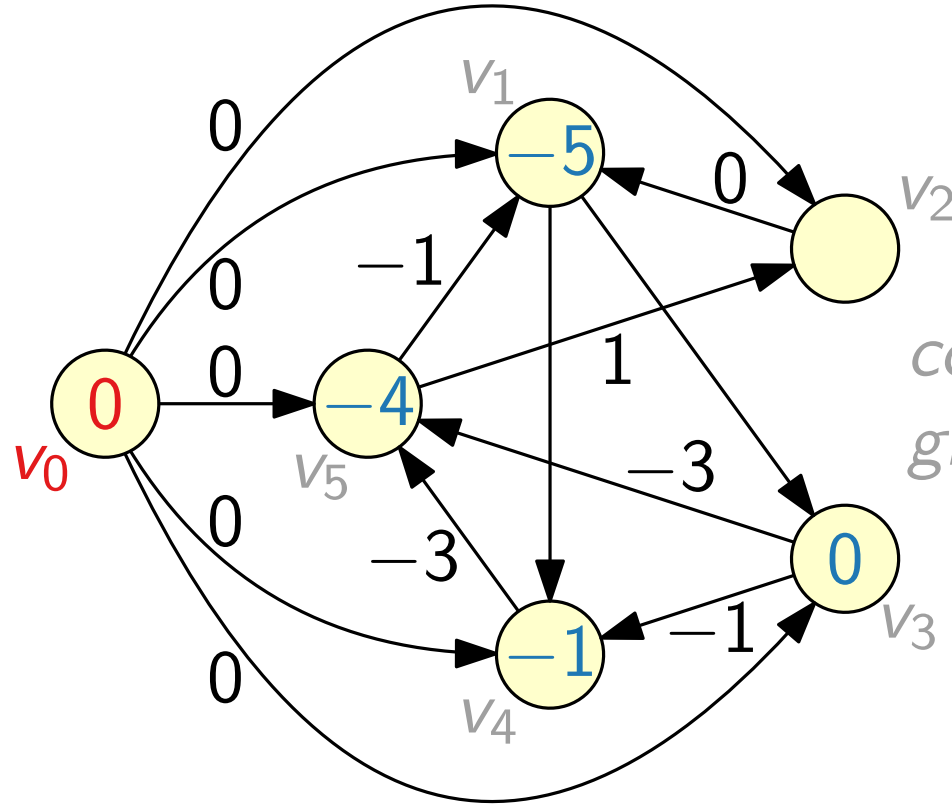
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\le 0 \\
x_1 - x_5 &\le -1 \\
x_2 - x_5 &\le 1 \\
x_3 - x_1 &\le 5 \\
x_4 - x_1 &\le 4 \\
x_4 - x_3 &\le -1 \\
x_5 - x_3 &\le -3 \\
x_5 - x_4 &\le -3
\end{aligned}
$$



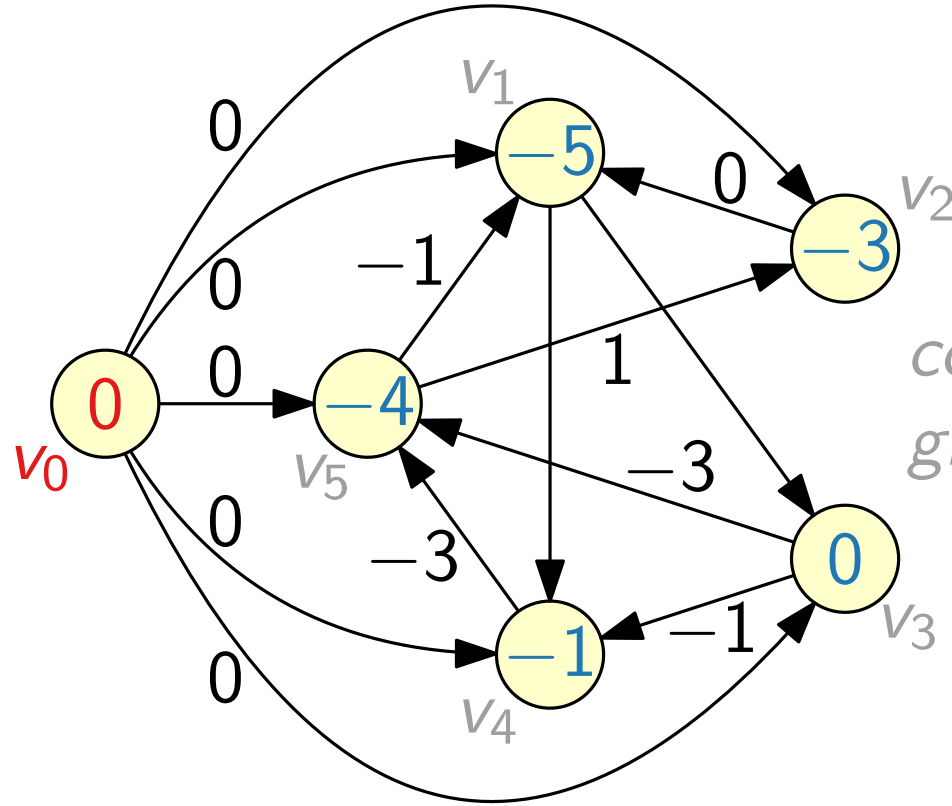
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \le b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\leq 0 \\
x_1 - x_5 &\leq -1 \\
x_2 - x_5 &\leq 1 \\
x_3 - x_1 &\leq 5 \\
x_4 - x_1 &\leq 4 \\
x_4 - x_3 &\leq -1 \\
x_5 - x_3 &\leq -3 \\
x_5 - x_4 &\leq -3
\end{aligned}
$$



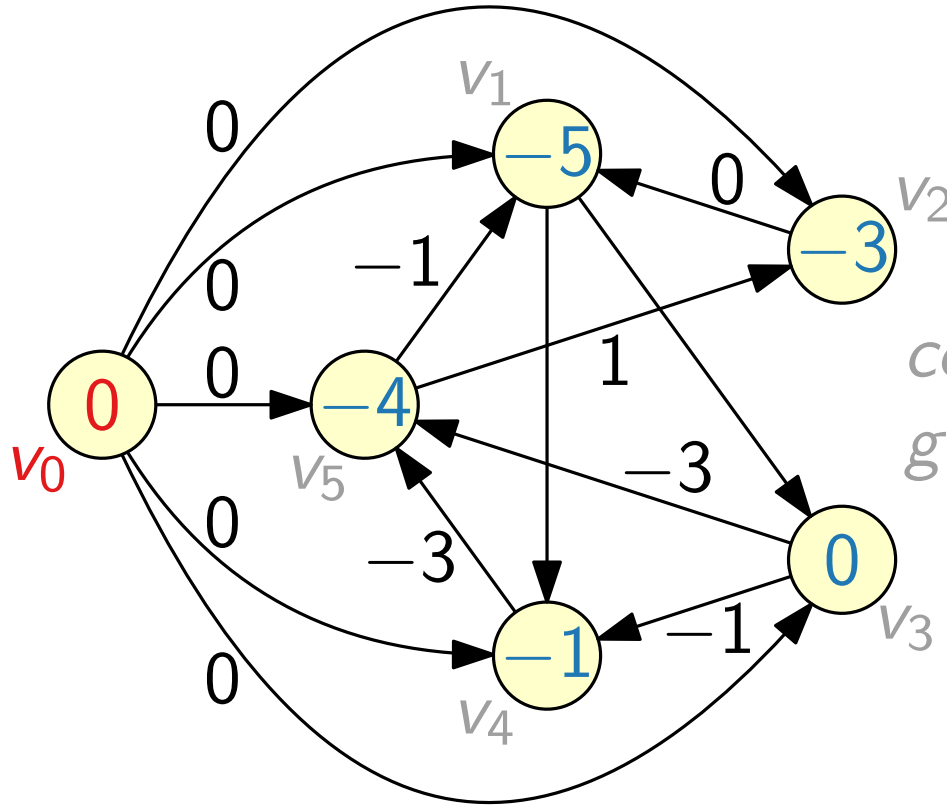
Yes: Take $x = (-5, -3, 0, -1, 4)$ or $x' = (0, 2, 5, 4, 9)$.

Lemma. Let x be a solution to a system $Ax \leq b$ of difference constraints, and let $d \in \mathbb{R}$. Then $x + d = (x_1 + d, \ldots, x_n + d)$ is a solution, too.

Solving Systems of Difference Constraints

Is this system feasible?

$$
\begin{aligned}
x_1 - x_2 &\le 0 \\
x_1 - x_5 &\le -1 \\
x_2 - x_5 &\le 1 \\
x_3 - x_1 &\le 5 \\
x_4 - x_1 &\le 4 \\
x_4 - x_3 &\le -1 \\
x_5 - x_3 &\le -3 \\
x_5 - x_4 &\le -3
\end{aligned}
$$



Definition. The constraint graph $G_{A,b}$ is a weighted digraph with vertex set $V_A = \{v_0, v_1, \ldots, v_n\}$ and edge set

$$E_A = \{v_i v_j : x_j - x_i \le b_{ij} \text{ is a constraint}\} \cup \{v_0 v_k : 1 \le k \le n\}.$$

The weight of $v_i v_j$ is b_{ij} if $i > 0$ and 0 otherwise.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow$

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}$,

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}$, $x_3 - x_2 \leq b_{23}, \ldots,$

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}$, $x_3 - x_2 \leq b_{23}, \ldots,$ $x_k - x_{k-1} \leq b_{k-1,k}$.
$\Rightarrow$

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}$, $x_3 - x_2 \leq b_{23}, \ldots,$ $x_k - x_{k-1} \leq b_{k-1,k}$.
$\Rightarrow$
Σ

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}, \quad x_3 - x_2 \leq b_{23}, \ldots, \quad x_k - x_{k-1} \leq b_{k-1,k}.$
$\underset{\Sigma}{\Rightarrow} 0 \leq b_{12} + b_{23} + \cdots + b_{k-1,k}$

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}, \quad x_3 - x_2 \leq b_{23}, \ldots, \quad x_k - x_{k-1} \leq b_{k-1,k}.$
$\underset{\Sigma}{\Rightarrow} 0 \leq b_{12} + b_{23} + \cdots + b_{k-1,k} = w(C)$

Shortest Paths Do the Job

Theorem. Let $Ax \leq b$ be a system of difference constraints, and let $\delta_k = \delta(v_0, v_k)$ be the length of a shortest v_0–v_k path for $k = 1, \ldots, n$.
If $G_{A,b}$ contains no negative cycles,
then $x = (\delta_1, \ldots, \delta_n)$ is a feasible solution.
If $G_{A,b}$ contains a negative cycle,
then there is no feasible solution.

Proof. Assume no neg. cycles. Consider $v_i v_j \in E_A$ with $i > 0$.
Δ-inequality $\Rightarrow$ $\delta_j \leq \delta_i + b_{ij}$, or $\delta_j - \delta_i \leq b_{ij}$.
Letting $x_i = \delta_i$ and $x_j = \delta_j$ satisfies $x_j - x_i \leq b_{ij}$.

Now assume $\exists$ neg. cycle and $Ax \leq b$ has a solution x.
Wlog., let $C = \langle v_1, v_2, \ldots, v_k = v_1 \rangle$ be a neg. cycle.
$\Rightarrow x_2 - x_1 \leq b_{12}, \quad x_3 - x_2 \leq b_{23}, \ldots, \quad x_k - x_{k-1} \leq b_{k-1,k}.$
$\underset{\Sigma}{\Rightarrow} 0 \leq b_{12} + b_{23} + \cdots + b_{k-1,k} = w(C)$ ↯ □

Shortest Paths & Negative Edge Weights

Ideas?

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:

Initialize(graph G, vtx s)
  foreach $u \in V$ **do**
    $u.d = \infty$
    $u.\pi = nil$
  $s.d = 0$

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:

```
Initialize(graph G, vtx s)
  foreach u ∈ V do
      u.d = ∞      estimate for δ(s, u)
      u.π = nil
  s.d = 0
```

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:

Initialize(graph G, vtx s)
  foreach $u \in V$ **do**
    $u.d = \infty$ — estimate for $\delta(s, u)$
    $u.\pi = nil$ — pointer to predecessor on some "currently" shortest s–u path
  $s.d = 0$

Shortest Paths & Negative Edge Weights

Ideas?

Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:

Initialize(graph G, vtx s)
  foreach $u \in V$ **do**
    $u.d = \infty$ — estimate for $\delta(s, u)$
    $u.\pi = nil$ — pointer to predecessor on some "currently" shortest s–u path
  $s.d = 0$

Relax(vtx u, vtx v, weights w)
  if $v.d > u.d + w(u, v)$ **then**
    $v.d = u.d + w(u, v)$
    $v.\pi = u$
    Q.DecreaseKey($v, v.d$)

Shortest Paths & Negative Edge Weights

Ideas?

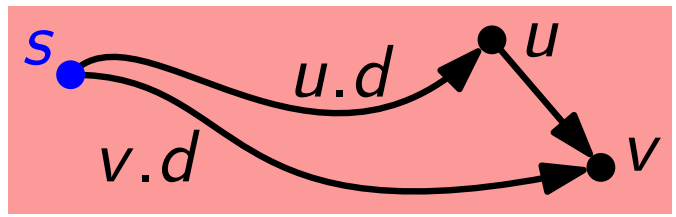
Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:



Initialize(graph G, vtx s)
  foreach $u \in V$ **do**
    $u.d = \infty$ — estimate for $\delta(s, u)$
    $u.\pi = nil$ — pointer to predecessor on some "currently" shortest s–u path
  $s.d = 0$

Relax(vtx u, vtx v, weights w)
  if $v.d > u.d + w(u, v)$ **then**
    $v.d = u.d + w(u, v)$
    $v.\pi = u$
    Q.DecreaseKey($v, v.d$)

Shortest Paths & Negative Edge Weights

Ideas?

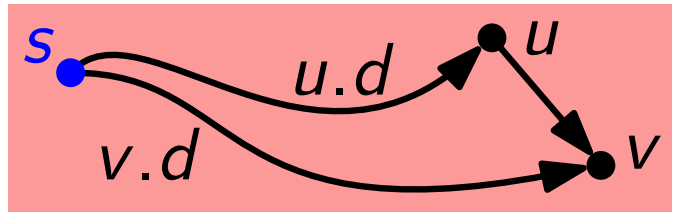
Dijkstra can handle only graphs with non-negative edge weights.

But maybe we can reduce to this problem?

What about adding the same constant c to each edge weight?

Problem: Paths with few edges get relatively cheaper :-(

Recall initialization and main subroutine of Dijkstra:



Initialize(graph G, vtx s)
  foreach $u \in V$ **do**
    $u.d = \infty$ — estimate for $\delta(s, u)$
    $u.\pi = nil$ — pointer to predecessor on some "currently" shortest s–u path
  $s.d = 0$

Relax(vtx u, vtx v, weights w)
  if $v.d > u.d + w(u, v)$ **then**
    $v.d = u.d + w(u, v)$
    $v.\pi = u$
    [Q.DecreaseKey(v, $v.d$)]

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

runtime?

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

runtime?

$O(E + V \log V)$

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

runtime?

$O(E + V \log V)$

runtime?

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

The Bellman–Ford Algorithm

```
Dijkstra(graph G, weights w, vtx s)
  Initialize(G, s)
  Q = new PriorityQueue(G.V, d)
  while not Q.Empty() do
      u = Q.ExtractMin()
      foreach v ∈ Adj[u] do
          Relax(u, v; w)
```

runtime?

$O(E + V \log V)$

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

runtime?

$O(V \cdot E)$

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization,

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) =$

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) = \delta(s, v_i)$

(δ s.p.)

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \underset{\delta \text{ s.p.}}{=} \delta(s, v_i) \underset{\text{Relax}}{\Rightarrow} v_i.d = \delta(s, v_i)$.

Suppose v is *not* reachable from s.

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \underset{\delta \text{ s.p.}}{=} \delta(s, v_i) \underset{\text{Relax}}{\Rightarrow} v_i.d = \delta(s, v_i)$.

Suppose v is *not* reachable from s. $\Rightarrow$ Initially, $v.d = \infty$

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \underset{\delta \text{ s.p.}}{=} \delta(s, v_i) \underset{\text{Relax}}{\Rightarrow} v_i.d = \delta(s, v_i)$.

Suppose v is *not* reachable from s. $\Rightarrow$ Initially, $v.d = \infty$

$\Rightarrow$ During execution, $v.d$ remains ∞ (otherwise $\exists$ s–v path)

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \underset{\delta \text{ s.p.}}{=} \delta(s, v_i) \underset{\text{Relax}}{\Rightarrow} v_i.d = \delta(s, v_i)$.

Suppose v is *not* reachable from s. $\Rightarrow$ Initially, $v.d = \infty$

$\Rightarrow$ During execution, $v.d$ remains ∞ (otherwise $\exists$ s–v path)

$\Rightarrow$ At termination, $v.d = \infty = \delta(s, v)$. $\square$

Correctness of Bellman–Ford

If G contains no neg. cycle reachable from s, after the for-i loop, for every vertex v, $v.d = \delta(s, v)$ and Bellman–Ford returns true.

Suppose v is reachable from s.

$\Rightarrow \exists$ shortest s–v path $\delta = \langle s = v_0, v_1, \ldots, v_k = v \rangle$.

G has no negative cycle, δ shortest path $\Rightarrow$ length $k \leq n - 1$.

After initialization, $v_0.d = \delta(s, v_0) = 0$.

In phase i of the alg., $v_{i-1}v_i$ is relaxed.

$\Rightarrow v_i.d \leq v_{i-1}.d + w(v_{i-1}, v_i)$. By induction, $v_{i-1}.d = \delta(s, v_{i-1})$.

$\Rightarrow v_i.d \leq \delta(s, v_{i-1}) + w(v_{i-1}, v_i) \underset{\delta \text{ s.p.}}{=} \delta(s, v_i) \underset{\text{Relax}}{\Rightarrow} v_i.d = \delta(s, v_i)$.

Suppose v is *not* reachable from s. $\Rightarrow$ Initially, $v.d = \infty$

$\Rightarrow$ During execution, $v.d$ remains ∞ (otherwise $\exists$ s–v path)

$\Rightarrow$ At termination, $v.d = \infty = \delta(s, v)$. $\square$

The Bellman–Ford Algorithm (overview)

```
Initialize(graph G, vtx s)
  foreach u ∈ V do
      u.d = ∞
      u.π = nil
  s.d = 0
```

```
Relax(vtx u, vtx v, weights w)
  if v.d > u.d + w(u, v) then
      v.d = u.d + w(u, v)
      v.π = u
```

```
Bellman–Ford(graph G, weights w, vtx s)
  Initialize(G, s)
  for i = 1 to |G.V| − 1 do
      foreach uv ∈ G.E do
          Relax(u, v; w)
  foreach uv ∈ G.E do
      if v.d > u.d + w(u, v) return false
  return true
```

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow$

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow \; v_1.d \leq v_0.d + w(v_0, v_1), \ldots,$

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow \; v_1.d \leq v_0.d + w(v_0, v_1), \ldots, \; v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\Rightarrow$

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow \; v_1.d \leq v_0.d + w(v_0, v_1), \ldots, \; v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\underset{\Sigma}{\Rightarrow}$

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow \; v_1.d \leq v_0.d + w(v_0, v_1), \ldots, \; v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\underset{\Sigma}{\Rightarrow} \; 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) =$

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow \; v_1.d \leq v_0.d + w(v_0, v_1), \ldots, \; v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\underset{\Sigma}{\Rightarrow} \; 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C)$ 

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow\ v_1.d \leq v_0.d + w(v_0, v_1), \ldots,\ v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\overset{\Rightarrow}{\Sigma}\ 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C)$ 

For this implication we additionally need that $\sum_i v_i.d < \infty$.

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow\ v_1.d \leq v_0.d + w(v_0, v_1), \ldots,\ v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\overset{\Rightarrow}{\Sigma}\ 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C)$ 

For this implication we additionally need that $\sum_i v_i.d < \infty$.

(True since C is reachable from s, plus the previous proof.)

Correctness (cont'd)

If G contains a negative cycle that is reachable from s, then Bellman–Ford returns false.

Let $C = \langle v_0, v_1, \ldots, v_k = v_0 \rangle$ be such a negative cycle.

Assume that Bellman-Ford returns true.

$\Rightarrow\ v_1.d \leq v_0.d + w(v_0, v_1), \ldots,\ v_k.d \leq v_{k-1}.d + w(v_{k-1}, v_k),$

$\overset{\Rightarrow}{\Sigma}\ 0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i) = w(C)$ 

For this implication we additionally need that $\sum_i v_i.d < \infty$.

(True since C is reachable from s, plus the previous proof.)

Improvement: $O(\sqrt{V}E \log W)$, where $W = \max_{uv \in E} w(u, v)$.
[Goldberg, SIAM J. Comput. 1995]

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \leq i,j \leq n}$.

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} =$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} =$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n}$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \ \delta(i,j) =$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} =$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} =$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \dots$

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

Extend-Shortest-Paths(L, W)
 $L' = (\ell'_{ij} = \infty)$ new $n \times n$ matrix
 for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
   for $k = 1$ **to** n **do**
    $\ell'_{ij} = \min\{\ell'_{ij}, \ell_{ik} + w_{kj}\}$
 return L'

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

Extend-Shortest-Paths(L, W)

```
L' = (ℓ'_ij = ∞) new n × n matrix
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            ℓ'_ij = min{ℓ'_ij, ℓ_ik + w_kj}
return L'
```

Slow-All-Pairs-SP(W)

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

Extend-Shortest-Paths(L, W)
```
L' = (ℓ'_ij = ∞) new n × n matrix
for i = 1 to n do
    for j = 1 to n do
        for k = 1 to n do
            ℓ'_ij = min{ℓ'_ij, ℓ_ik + w_kj}
return L'
```

Slow-All-Pairs-SP(W)
```
L^(1) = W
for m = 2 to n − 1 do
    L^(m) = new matrix
    L^(m) = ESP(L^(m−1), W)
return L^(n−1)
```

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

Extend-Shortest-Paths(L, W)

$L' = (\ell'_{ij} = \infty)$ new $n \times n$ matrix
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    for $k = 1$ **to** n **do**
      $\ell'_{ij} = \min\{\ell'_{ij}, \ell_{ik} + w_{kj}\}$
return L'

Slow-All-Pairs-SP(W)

$L^{(1)} = W$
for $m = 2$ **to** $n - 1$ **do**
  $L^{(m)} =$ new matrix
  $L^{(m)} = \text{ESP}(L^{(m-1)}, W)$
return $L^{(n-1)}$

Runtime:

All-Pairs Shortest Paths

$w_{jj} = 0$

Assume that the graph is given by a matrix $W = (w_{ij})_{1 \le i,j \le n}$.

Let $\ell_{ij}^{(m)}$ be the length of a shortest i–j path with $\le m$ edges.

Then $\ell_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$ and $\ell_{ij}^{(m)} = \min_{1 \le k \le n} \{\ell_{ik}^{(m-1)} + w_{kj}\}$.

$\Rightarrow \delta(i,j) = \ell_{ij}^{(n-1)} = \ell_{ij}^{(n)} = \ell_{ij}^{(n+1)} = \cdots$ since shortest paths are simple (if there are no neg. cycles)!

Extend-Shortest-Paths(L, W)

$L' = (\ell'_{ij} = \infty)$ new $n \times n$ matrix
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    for $k = 1$ **to** n **do**
      $\ell'_{ij} = \min\{\ell'_{ij}, \ell_{ik} + w_{kj}\}$
return L'

Slow-All-Pairs-SP(W)

$L^{(1)} = W$
for $m = 2$ **to** $n - 1$ **do**
  $L^{(m)}$ = new matrix
  $L^{(m)} = \text{ESP}(L^{(m-1)}, W)$
return $L^{(n-1)}$

Runtime: $O(n^4)$

Faster APSP

Faster-All-Pairs-SP($n \times n$ matrix W)

$L^{(1)} = W$

$m = 1$

while $m < n - 1$ **do**

    L = new $n \times n$ matrix

    L = Extend-Shortest-Path( )

    $m =$

return $L^{(m)}$

Runtime: $O(n^3 \log n)$

Faster APSP

Faster-All-Pairs-SP($n \times n$ matrix W)

$L^{(1)} = W$

$m = 1$

while $m < n - 1$ **do**

  $L^{(2m)} =$ new $n \times n$ matrix

  $L^{(2m)} =$ Extend-Shortest-Path($L^{(m)}, L^{(m)}$)

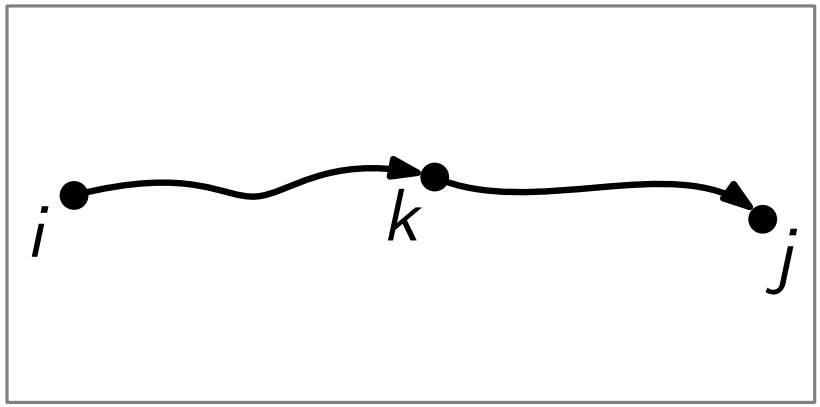
  $m = 2m$

return $L^{(m)}$

Runtime: $O(n^3 \log n)$

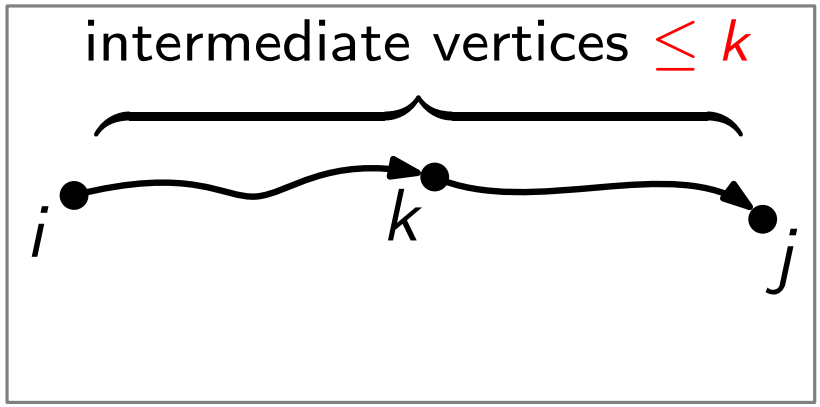
The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



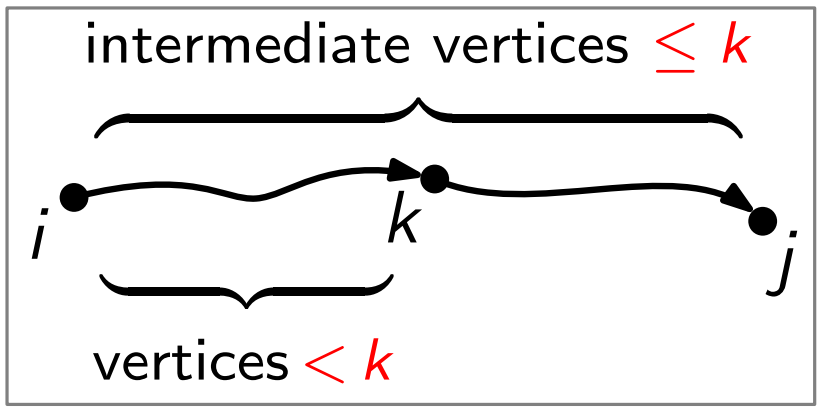
The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



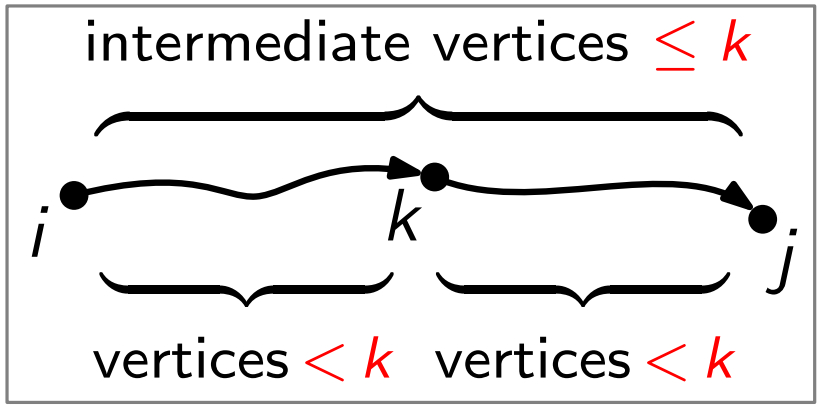
The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



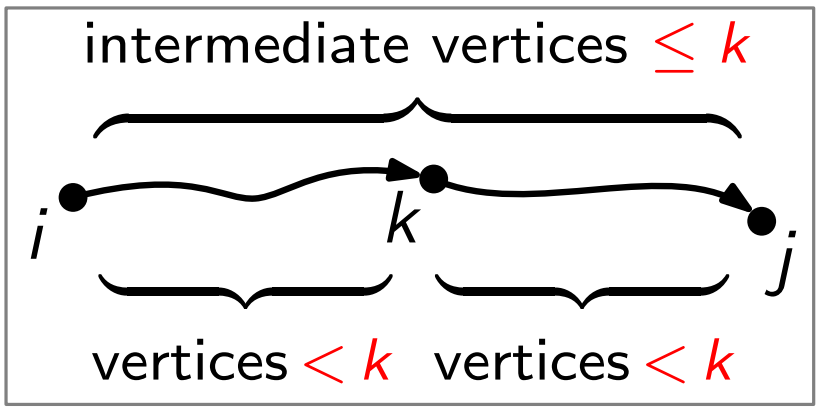
The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



The Floyd–Warshall Algorithm

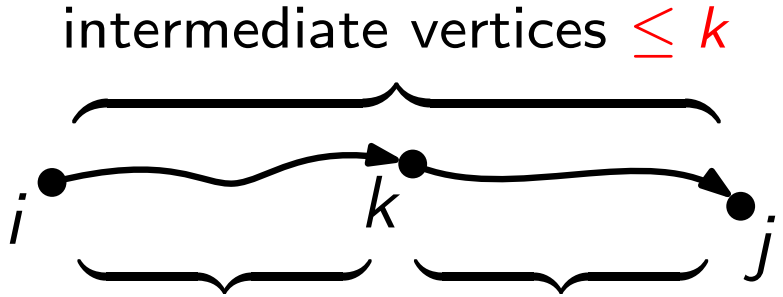
[W., J. ACM 1962]
[F., Comm. ACM 1977]



$d_{ij}^{(0)} = w_{ij}$; if $k > 0$ then

The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]

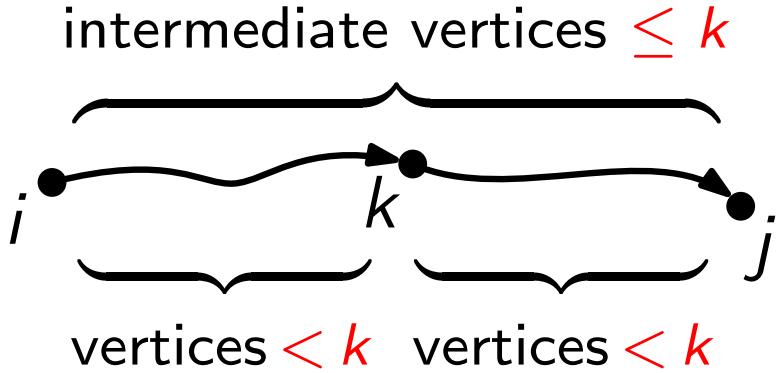


$$d_{ij}^{(0)} = w_{ij}\,; \quad \text{if } k > 0 \text{ then}$$

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



$$d_{ij}^{(0)} = w_{ij}\,; \quad \text{if } k > 0 \text{ then}$$

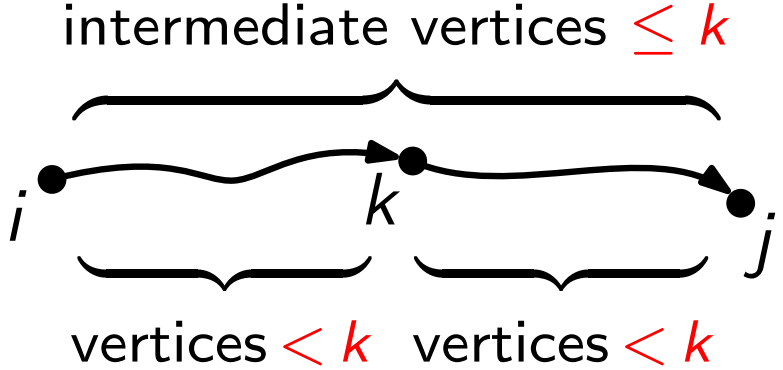
$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

Floyd–Warshall($n \times n$ matrix W)

$D^{(0)} = W$
for $k = 1$ **to** n **do**
  $D^{(k)} =$ new $n \times n$ matrix
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      $d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$
return $D^{(n)}$

The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



$$d_{ij}^{(0)} = w_{ij}\,; \quad \text{if } k > 0 \text{ then}$$

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

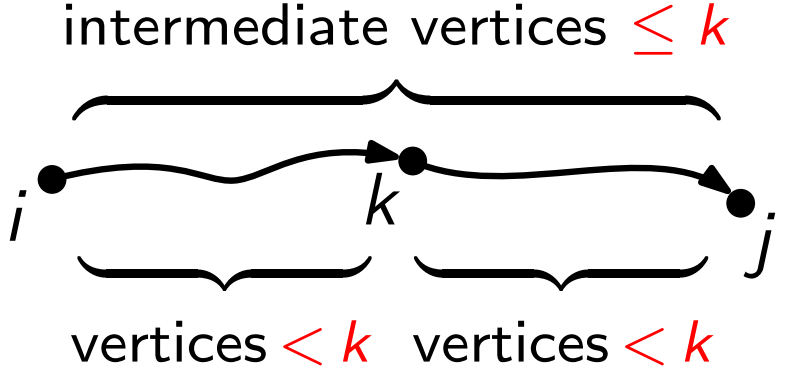
Floyd–Warshall($n \times n$ matrix W)

```
D^(0) = W
for k = 1 to n do
    D^(k) = new n × n matrix
    for i = 1 to n do
        for j = 1 to n do
            d_ij^(k) = min{d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1)}
return D^(n)
```

Runtime:

The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



$$d_{ij}^{(0)} = w_{ij}\,; \quad \text{if } k > 0 \text{ then}$$

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

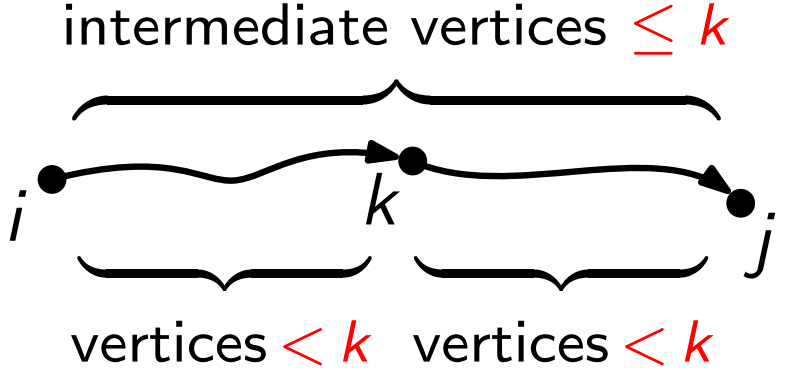
Floyd–Warshall($n \times n$ matrix W)

```
D^(0) = W
for k = 1 to n do
    D^(k) = new n × n matrix
    for i = 1 to n do
        for j = 1 to n do
            d_ij^(k) = min{d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1)}
return D^(n)
```

Runtime:
$O(n^3)$

The Floyd–Warshall Algorithm

[W., J. ACM 1962]
[F., Comm. ACM 1977]



$$d_{ij}^{(0)} = w_{ij}; \quad \text{if } k > 0 \text{ then}$$

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$$

Floyd–Warshall($n \times n$ matrix W)

$D^{(0)} = W$
for $k = 1$ **to** n **do**
  $D^{(k)} =$ new $n \times n$ matrix
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      $d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\}$
return $D^{(n)}$

Improvement:
$O(V(V \log V + E))$
Johnson's algorithm
[J. ACM 1977]

Runtime:
$O(n^3)$