Duality.

$$R = \max_y \min_x (x^t A y).$$

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:

Duality.

$$R = \max_y \min_x (x^t Ay).$$
$$C = \min_x \max_y (x^t Ay).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v$

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v$

Duality.

$$R = \max_y \min_x (x^t Ay).$$
$$C = \min_x \max_y (x^t Ay).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.
$\implies R \geq C$

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.
$\implies R \geq C$

Equilibrium $\implies R = C$!

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.
$\implies R \geq C$

Equilibrium $\implies R = C$!

Strong Duality: There is an equilibrium point!

Duality.

$$R = \max_y \min_x (x^t Ay).$$
$$C = \min_x \max_y (x^t Ay).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.
$\implies R \geq C$

Equilibrium $\implies R = C$!

Strong Duality: There is an equilibrium point! and $R = C$!

Duality.

$$R = \max_y \min_x (x^t A y).$$
$$C = \min_x \max_y (x^t A y).$$

Weak Duality: $R \leq C$.
Proof: Better to go second. $\square$
Note:
In situation R. y plays "Defense". x plays "Offense."
In situation C. x plays "Defense". y plays "Offense."

At Equilibrium (x^*, y^*), payoff v:
row payoffs (Ay^*) all $\geq v \implies R \geq v$.
column payoffs $((x^*)^t A)$ all $\leq v \implies v \geq C$.
$\implies R \geq C$

Equilibrium $\implies R = C$!

Strong Duality: There is an equilibrium point! and $R = C$!

Doesn't matter who plays first!

Summary and..

Zero sum game:

Summary and..

Zero sum game: $m \times n$ matrix A

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.
$\implies \min_i (Ay_2)_i > \min_i (Ay_1)_i$

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.
$\implies \min_i (Ay_2)_i > \min_i (Ay_1)_i$ Since x zero on non-best.
Best row is worse under y_2.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.
$\implies$ $\min_i (Ay_2)_i > \min_i (Ay_1)_i$ Since x zero on non-best.
Best row is worse under y_2.
$\implies$ Column player has incentive to change.

Summary and..

Zero sum game: $m \times n$ matrix A

row minimizes. strategy: m-dimensional vector x
... probability distribution over rows.

column maximizes. strategy: vector m-dimensional vector x
... probability distribution over columns.

Payoff (x, y): $x^T Ay$.

Nash equilibrium (x^*, y^*):
neither player has better response against others.

If there is an equilibrium: no disadvantage in announcing strategy!

All equilibrium points all have same payoff.

Why? Equilibriums: $x_1^T Ay_1 < x_2^T Ay_2$.
$\implies$ $\min_i (Ay_2)_i > \min_i (Ay_1)_i$ Since x zero on non-best.
Best row is worse under y_2.
$\implies$ Column player has incentive to change.
x_1, y_1 is not equilibrium.

An "asymptotic" game.

"Catch me."

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.
Row (“Catch me”): choose path from a to b.

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.
Row (“Catch me”): choose path from a to b.
Column(“Catcher”): choose edge.

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.
Row (“Catch me”): choose path from a to b.
Column(“Catcher”): choose edge.
Row pays if column chooses edge on path.

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.
Row (“Catch me”): choose path from a to b.
Column(“Catcher”): choose edge.
Row pays if column chooses edge on path.

Matrix:
row for each path: p

An “asymptotic” game.

“Catch me.”

Given: $G = (V, E)$.
Given $a, b \in V$.
Row (“Catch me”): choose path from a to b.
Column(“Catcher”): choose edge.
Row pays if column chooses edge on path.

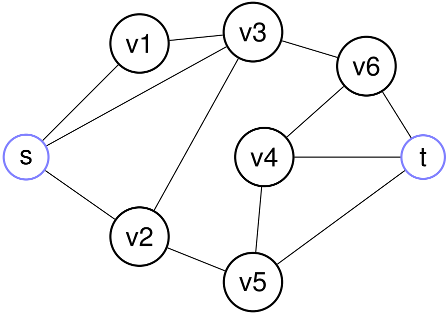
Matrix:
row for each path: p
column for each edge: e

An "asymptotic" game.

"Catch me."

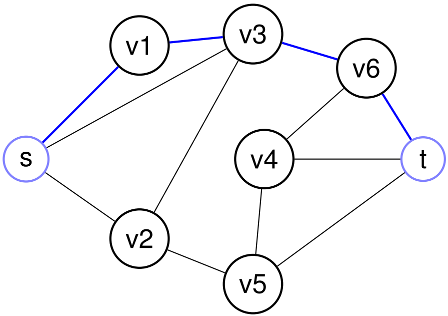
Given: $G = (V, E)$.
Given $a, b \in V$.
Row ("Catch me"): choose path from a to b.
Column("Catcher"): choose edge.
Row pays if column chooses edge on path.

Matrix:
row for each path: p
column for each edge: e
$A[p, e] = 1$ if $e \in p$.



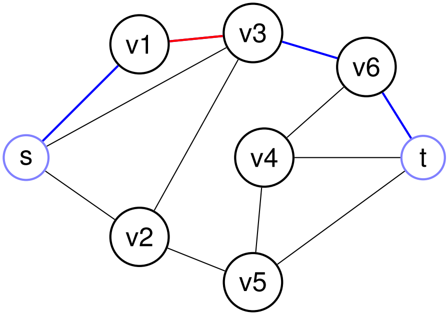
Catchme:

Catcher:



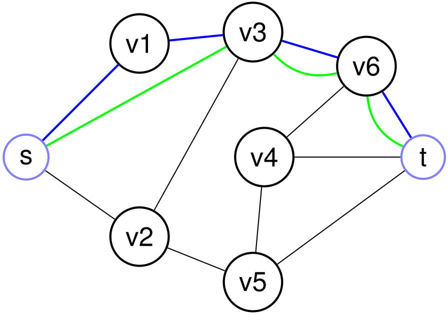
Catchme:
Use Blue Path.

Catcher:



Catchme:
Use Blue Path.

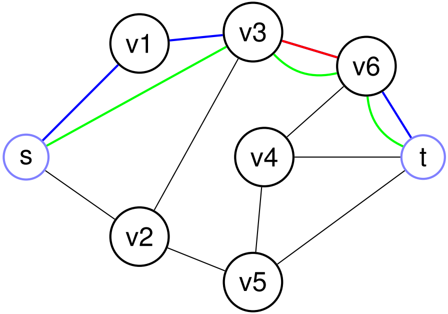
Catcher:
Caught!



Catchme:

Blue with prob. 1/2.
Green with prob. 1/2.

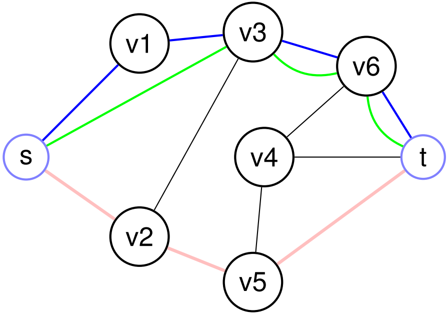
Catcher:



Catchme:

Blue with prob. 1/2.
Green with prob. 1/2.

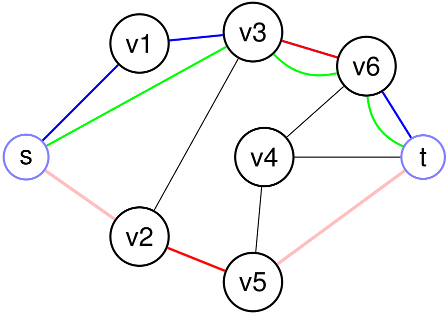
Catcher:
Caught!



Catchme:

Blue with prob. 1/3.
Green with prob. 1/6.
Pink with prob. 1/2.

Catcher:



Catchme:

Blue with prob. 1/3.
Green with prob. 1/6.
Pink with prob. 1/2.

Catcher:
Caught, sometimes.
With probability $1/2$.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path?

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.
Minimum cut allows the maximum toll on any edge!

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.
Minimum cut allows the maximum toll on any edge!

What should "catch me" do to avoid catcher?

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.
Minimum cut allows the maximum toll on any edge!

What should "catch me" do to avoid catcher?
minimize maximum load on any edge!

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.
Minimum cut allows the maximum toll on any edge!

What should "catch me" do to avoid catcher?
minimize maximum load on any edge!
Max-Flow Problem.

Example.

Row solution: $Pr[p_1] = 1/2$, $Pr[p_2] = 1/3$, $Pr[p_3] = 1/6$.
Edge solution: $Pr[e_1] = 1/2$, $Pr[e_2] = 1/2$

Offense (Best Response.):

Catch me: route along shortest path.
(Knows catcher's distribution.)
Catcher: raise toll on most congested edge.
(Knows catch me's distribution.)

Defense:

Where should "catcher" play to catch any path? a cut.
Minimum cut allows the maximum toll on any edge!

What should "catch me" do to avoid catcher?
minimize maximum load on any edge!
Max-Flow Problem.

Note: exponentially many strategies for "catch me"!

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \ldots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \ldots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.
Toll: charge most loaded edge.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \ldots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.
Toll: charge most loaded edge.

Defense: Toll: maximize shortest path under tolls.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \ldots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.
Toll: charge most loaded edge.

Defense: Toll: maximize shortest path under tolls.
Route: minimize max loaded on any edge.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.
Toll: charge most loaded edge.

Defense: Toll: maximize shortest path under tolls.
Route: minimize max loaded on any edge.

Toll/Congestion

Given: $G = (V, E)$.
Given $(s_1, t_1) \dots (s_k, t_k)$.
Row: choose routing of all paths.
Column: choose edge.
Row pays if column chooses edge on any path.

Matrix:
row for each routing: r
column for each edge: e

$A[r, e]$ is congestion on edge e by routing r

Offense: (Best Response.)
Router: route along shortest paths.
Toll: charge most loaded edge.

Defense: Toll: maximize shortest path under tolls.
Route: minimize max loaded on any edge.

Again: exponential number of paths for route player.

Summary...

You should now know about

Summary...

You should now know about

Games

Summary...

You should now know about

Games
Nash Equilibrium

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.
Checking Equilibrium.

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.
Checking Equilibrium.
Best Response.

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.
Checking Equilibrium.
Best Response.
Statement of Duality Theorem.

Summary...

You should now know about

Games
Nash Equilibrium
Pure Strategies
Zero Sum Two Person Games
Mixed Strategies.
Checking Equilibrium.
Best Response.
Statement of Duality Theorem.

Today

Today

Maximum Weight Matching

Today

Maximum Weight Matching
Undergraduate: saw maximum matching!

Today

Maximum Weight Matching
Undergraduate: saw maximum matching! (hopefully.)

Today

Maximum Weight Matching
Undergraduate: saw maximum matching! (hopefully.) Will review.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

Matching.

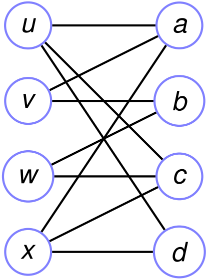
Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

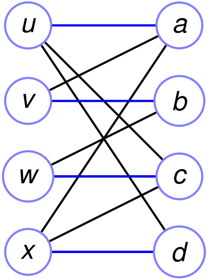
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G=(U,V,E)$, with edge weights $w: E \to R$, find a maximum weight matching.

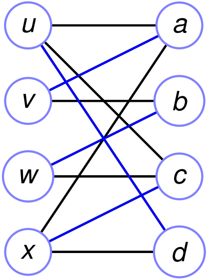
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

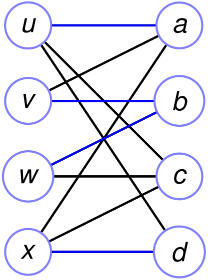
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

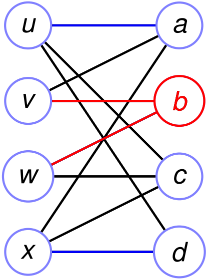
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

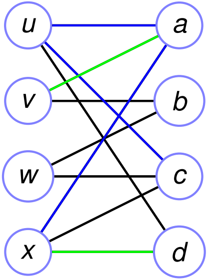
A matching is a set of edges where no two share an endpoint.



Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.

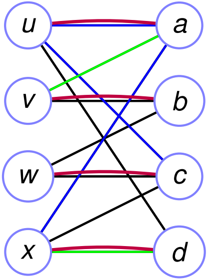


Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



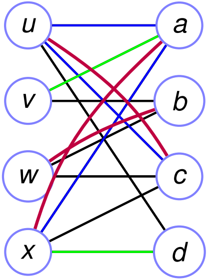
Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

Solution Value: 7.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



Blue – 3. Green - 2,
Black - 1, Non-edges - 0.

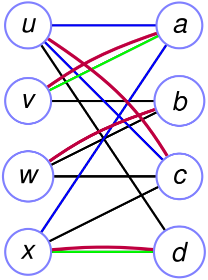
Solution Value: 7.

Solution Value: 7.

Matching.

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find a maximum weight matching.

A matching is a set of edges where no two share an endpoint.



Blue – 3. Green - 2, Black - 1, Non-edges - 0.

Solution Value: 7.

Solution Value: 7.

Solution Value: 8.

Applications

Jobs to workers.

Applications

Jobs to workers.

Teachers to classes.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

"The assignment problem"

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

“The assignment problem”

Min Weight Matching.

Applications

Jobs to workers.

Teachers to classes.

Classes to classrooms.

"The assignment problem"

Min Weight Matching.
  Negate values and find maximum weight matching.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, find an vertex cover function of minimum total value.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

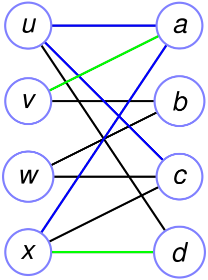
A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.

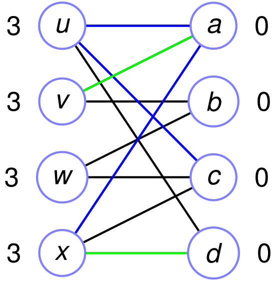


Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



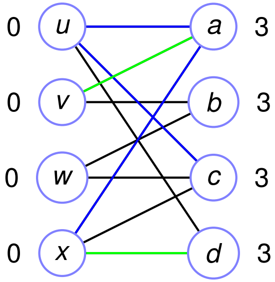
Solution Value: 12.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

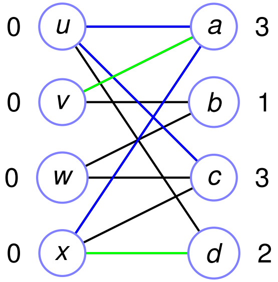
Solution Value: 12.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

Solution Value: 12.

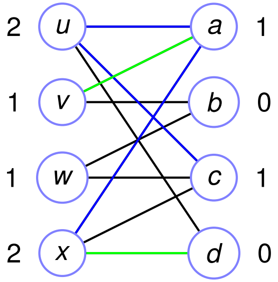
Solution Value: 9.

Vertex Cover

Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \to R$, find an vertex cover function of minimum total value.

A function $p : V \to R$, where for all edges, $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

Minimize $\sum_{v \in U \cup V} p(u)$.



Solution Value: 12.

Solution Value: 12.

Solution Value: 9.

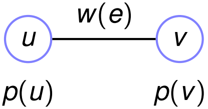
Solution Value: 8.

Cover is upper bound.

Feasible $p(\cdot)$,

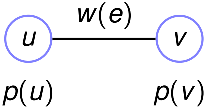
Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

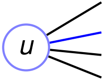


Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

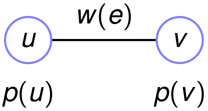


For a matching M, each u is the endpoint of at most one edge in M.

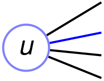


Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



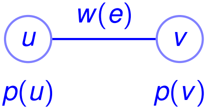
For a matching M, each u is the endpoint of at most one edge in M.



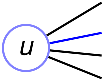
$$\sum_{e=(u,v)\in M} w(e)$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



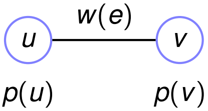
For a matching M, each u is the endpoint of at most one edge in M.



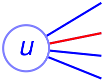
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v))$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



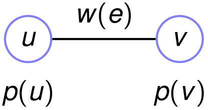
For a matching M, each u is the endpoint of at most one edge in M.



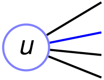
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u,v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.

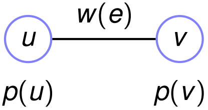


$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u \in U} p(u) + \sum_{v \in V} p(v)$$

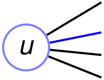
Holds with equality if

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.



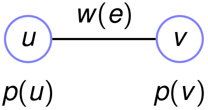
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u\in U} p(u) + \sum_{v\in V} p(v)$$

Holds with equality if

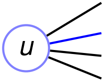
for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

Cover is upper bound.

Feasible $p(\cdot)$, for edge $e = (u, v)$, $p(u) + p(v) \geq w(e)$.



For a matching M, each u is the endpoint of at most one edge in M.



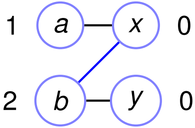
$$\sum_{e=(u,v)\in M} w(e) \leq \sum_{e=(u,v)\in M} (p(u) + p(v)) \leq \sum_{u\in U} p(u) + \sum_{v\in V} p(v)$$

Holds with equality if

for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

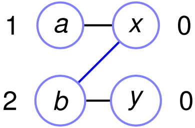
perfect matching.

Simple example.



Blue edge – 2, Others – 1.

Simple example.

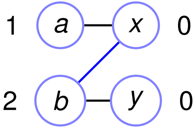


Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.

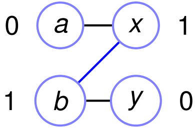
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

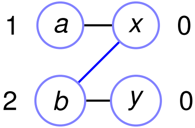
Value: 3.
Using max incident edge.



Value: 2.
Same as optimal matching!

Proof of optimality.

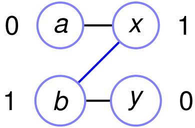
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.

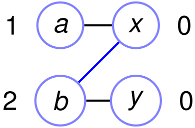


Matching and cover are optimal,

Value: 2.
Same as optimal matching!

Proof of optimality.

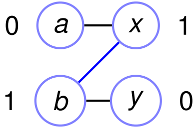
Simple example.



Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.

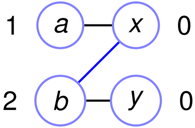


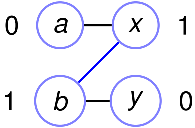
Value: 2.
Same as optimal matching!

Proof of optimality.

Matching and cover are optimal,
edges in matching have $w(e) = p(u) + p(v)$. Tight edge.

Simple example.





Blue edge – 2, Others – 1.

Using max incident edge.

Value: 3.
Using max incident edge.

Value: 2.
  Same as optimal matching!

Proof of optimality.

Matching and cover are optimal,
  edges in matching have $w(e) = p(u) + p(v)$. Tight edge.
  all nodes are matched.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

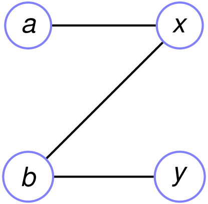
Example:

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

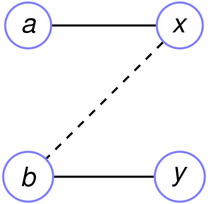


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

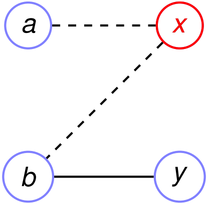


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

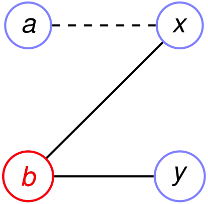


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



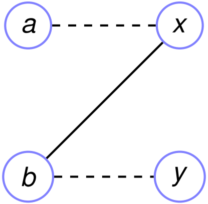
Start at unmatched node(s),
follow unmatched edge(s),
follow matched.
Repeat until an unmatched node.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:

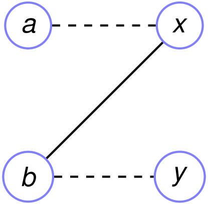


Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



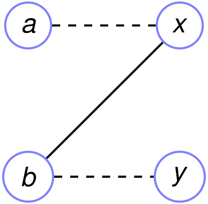
Start at unmatched node(s),

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



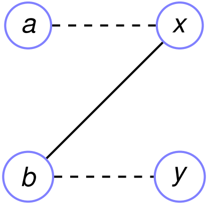
Start at unmatched node(s),
follow unmatched edge(s),

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

Example:



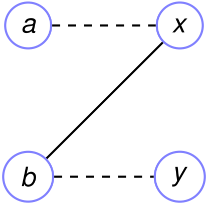
Start at unmatched node(s),
follow unmatched edge(s),
follow matched.

Maximum Matching

Given a bipartite graph, $G = (U, V, E)$, find a maximum sized matching.

Key Idea: Augmenting Alternating Paths.

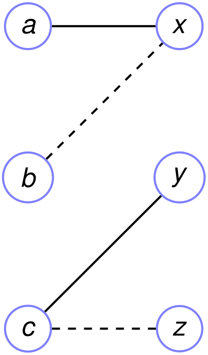
Example:



Start at unmatched node(s),
follow unmatched edge(s),
follow matched.
Repeat until an unmatched node.

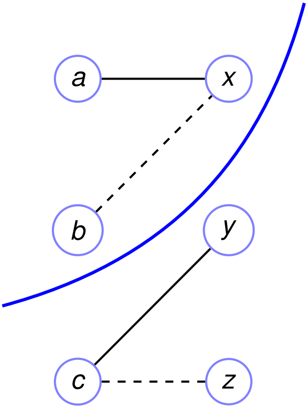
No perfect matching

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

No perfect matching

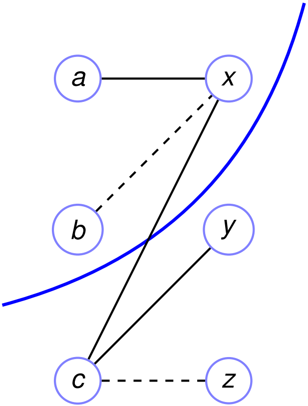


Can’t increase matching size.

No alternating path from (a) to (y).

Cut!

No perfect matching

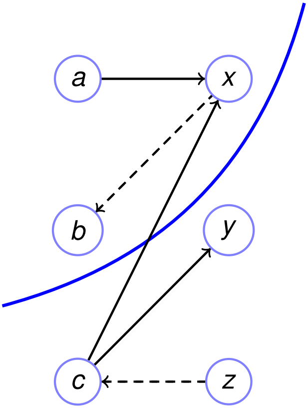


Can’t increase matching size.
  No alternating path from (a) to (y).

Cut!

Still no augmenting path.
  Still Cut?

No perfect matching



Algorithm:

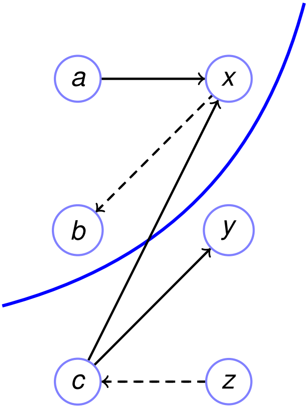
Can't increase matching size.
No alternating path from (a) to (y).

Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

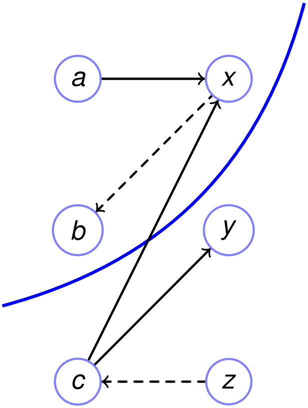
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.

No perfect matching



Can't increase matching size.
No alternating path from (a) to (y).

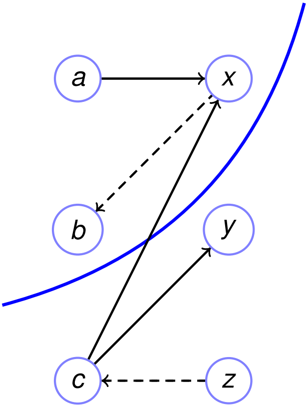
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

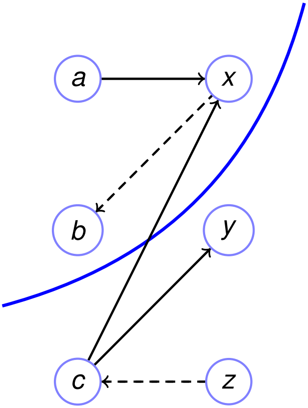
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

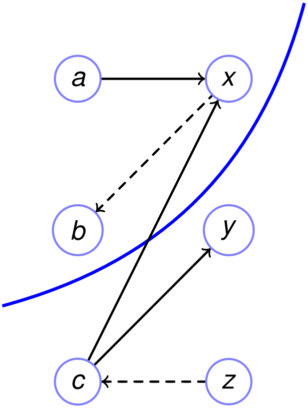
Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched

No perfect matching



Can’t increase matching size.
No alternating path from (a) to (y).

Cut!

Still no augmenting path.
Still Cut?

Use directed graph!
Cut in this graph.

Algorithm:
Given matching.
Direct unmatched edges U to V, matched V to U.
Find path between unmatched nodes on left to right. (BFS, DFS).
Until everything matched ... or output a cut.

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to both if

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to <u>both</u> if
for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and

Back to Maximum Weight Matching.

Want vertex cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to both if

for $e \in M$, $w(e) = p(u) + p(v)$ (Defn: tight edge.) and
perfect matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.

Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

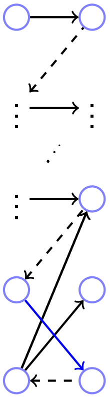
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.

Use alt./aug. paths of tight edges.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

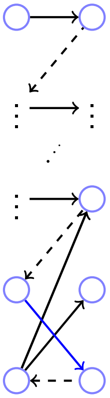
Add tight edges to matching.

Use alt./aug. paths of tight edges.

"maximum matching algorithm."

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

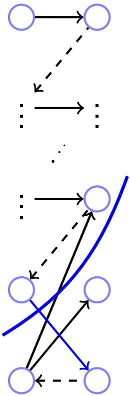
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

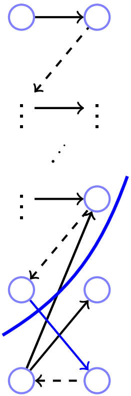
Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

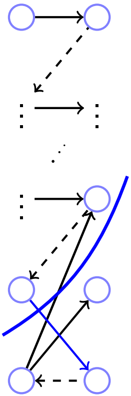
Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
 Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    ”maximum matching algorithm.”

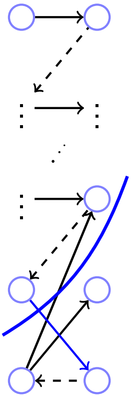
No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

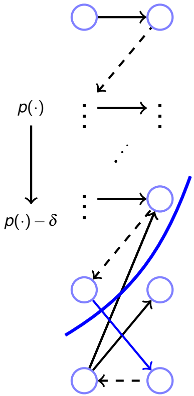
No augmenting path.
Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

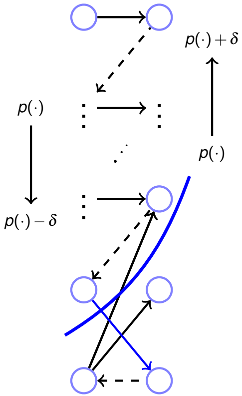
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U,

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
    Use alt./aug. paths of tight edges.
        "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

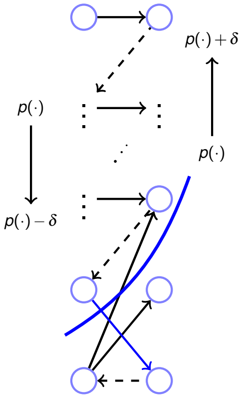
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

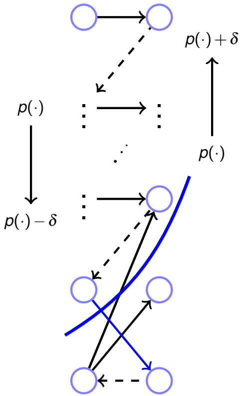
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

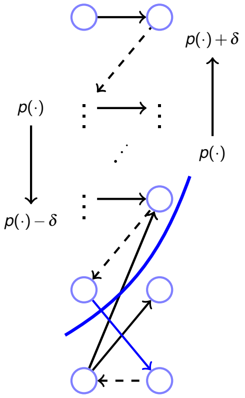
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight
... and get new tight edge!

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

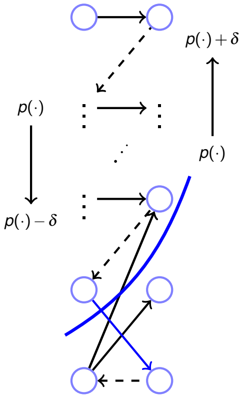
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight
... and get new tight edge!
What's delta?

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
  Use alt./aug. paths of tight edges.
    "maximum matching algorithm."

No augmenting path.
  Cut, (S, T), in directed graph of tight edges!

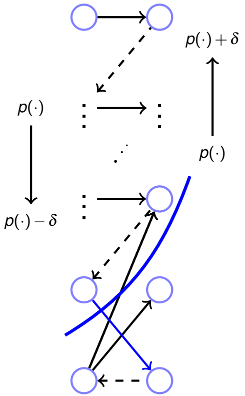
All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
  all explored edges still tight,
  matched edges still tight
... and get new tight edge!
What's delta? $w(e) < p(u) + p(v) \rightarrow$

Maximum Weight Matching

Goal: perfect matching on tight edges.



Algorithm

Init: empty matching, feasible cover function ($p(\cdot)$)

Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

No augmenting path.
Cut, (S, T), in directed graph of tight edges!

All edges across cut are not tight. (loose?)

Nontight edges leaving cut, go from S_U, T_V.

Lower prices in S_U, raise prices in S_V,
all explored edges still tight,
matched edges still tight

... and get new tight edge!

What's delta? $w(e) < p(u) + p(v) \rightarrow$
$\delta = min_{e \in (S_U \times T_V)} p(u) + p(v) - w(e)$.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible!

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$,

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
  $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
  Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to S in next cut.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
  $p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

Main Work:
  breadth first search from unmatched nodes finds cut.
  Update prices (find minimum delta.)

Simple Implementation:
  Each bfs either augments or adds node to S in next cut.
  $O(n)$ iterations per augmentation.
  $O(n)$ augmentations.

Some details

Add 0 value edges, so that optimal solution contains perfect matching.

Beginning "Matcher" Solution: $M = \{\}$.

Feasible! Value = 0.

Beginning "Coverer" Solution:
$p(u) =$ maximum incident edge for $u \in U$, 0 otherwise.

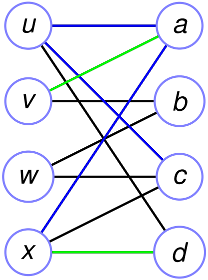
Main Work:
breadth first search from unmatched nodes finds cut.
Update prices (find minimum delta.)

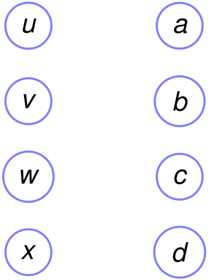
Simple Implementation:
Each bfs either augments or adds node to S in next cut.
$O(n)$ iterations per augmentation.
$O(n)$ augmentations.

$O(n^2 m)$ time.

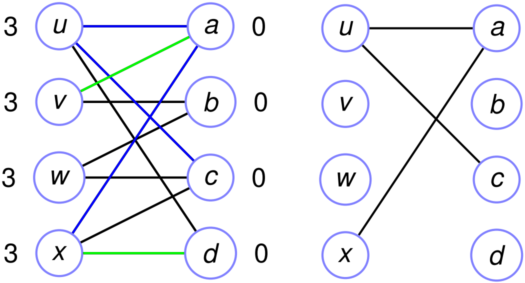
Example

Weight legend:
black 1, green 2, blue 3



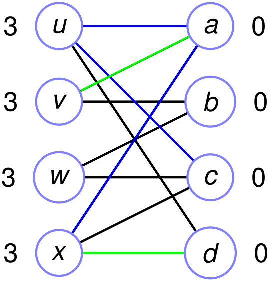


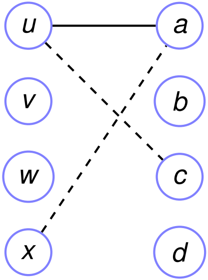
Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.

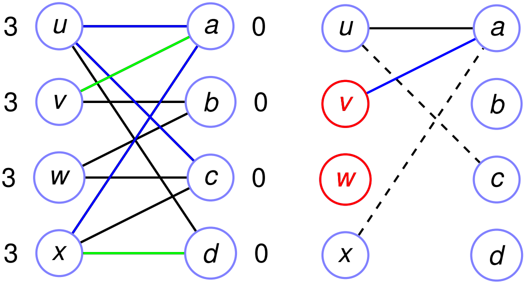
Example





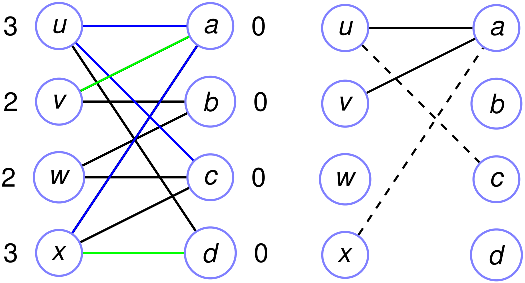
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.

Example



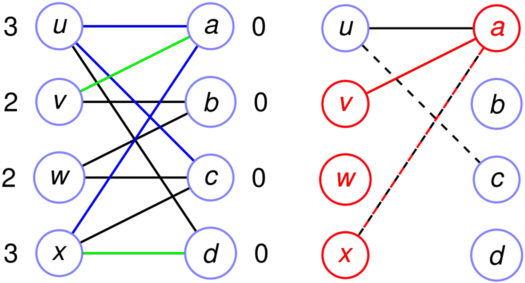
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!

Example



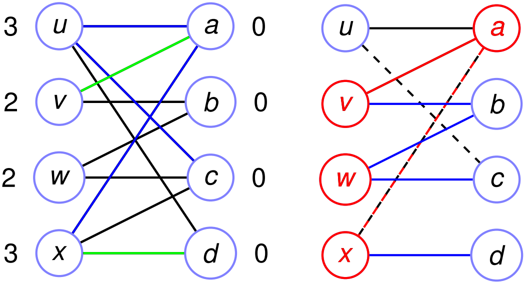
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.

Example



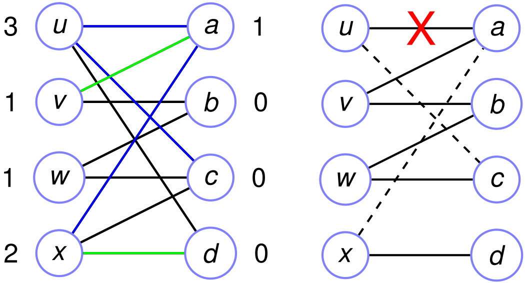
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices... $\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$

Example



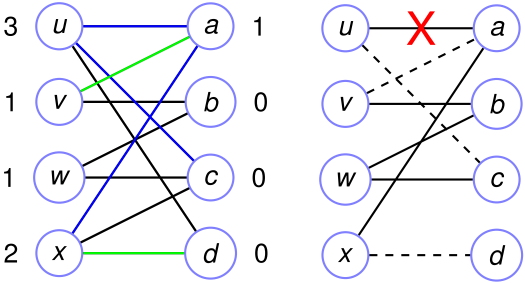
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.

Example



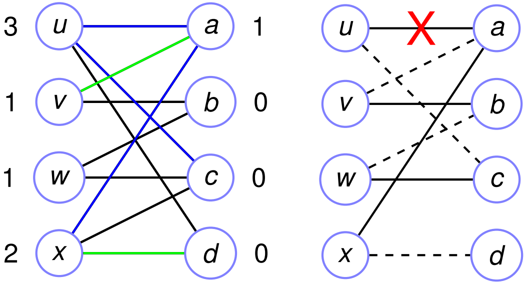
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.

Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

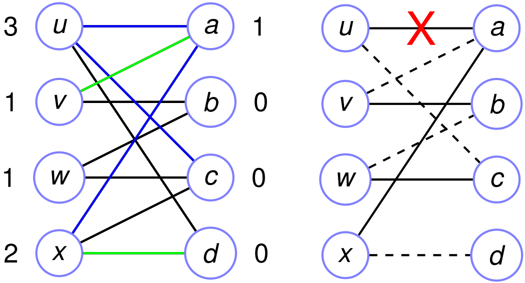
Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.
..and another augmentation...

..and finally: a perfect matching.

Example

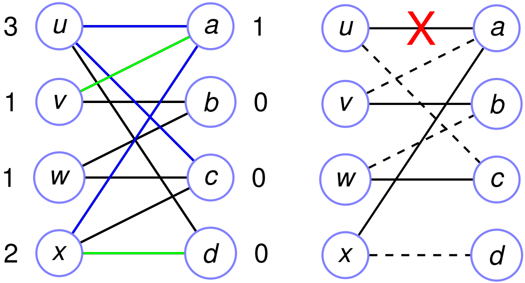


All matched edges tight.

Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a “new” nontight edge.
..and another augmentation...

..and finally: a perfect matching.

Example

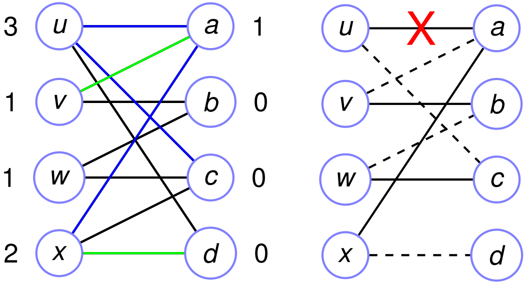


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching.

Example

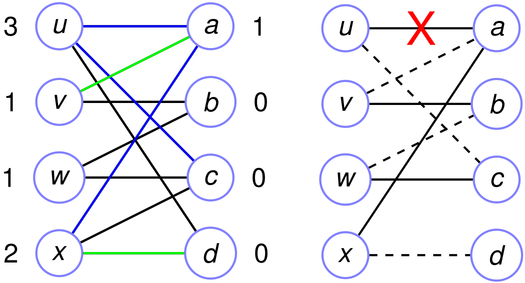


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\to$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function.

Example

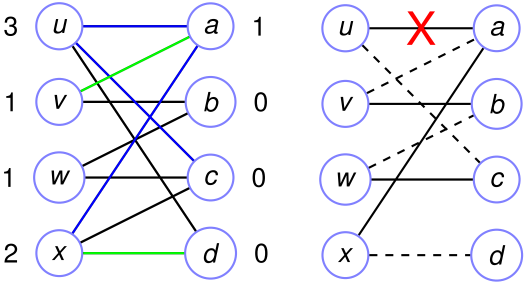


Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same.

Example

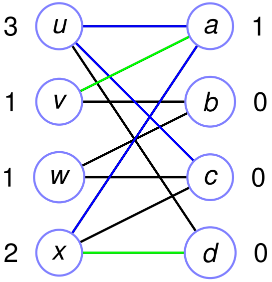


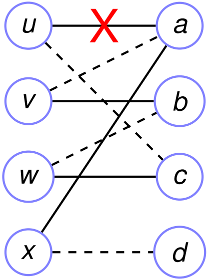
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Example





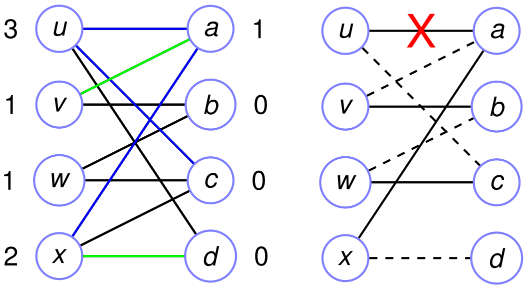
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:

Example



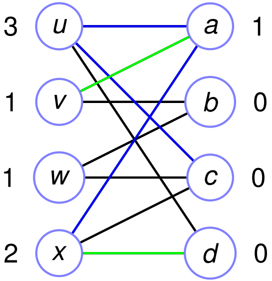
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

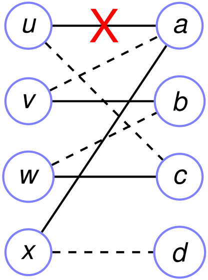
..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.

Example





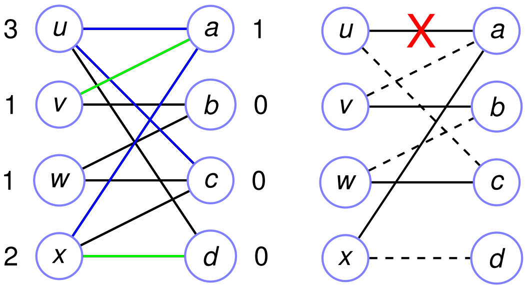
Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.

Example



Weight legend:
black 1, green 2, blue 3
Tight edges for inital prices.
Max matching in tight edges.
dashed means matched.
No augmenting path $\rightarrow$
reachable: $S = \{u, v\}$
Blue edge on right soon
to be tight!
Adjust prices...$\delta = 1$
new tight edges.
Still no augmenting path.
Reachable $S = \{v, w, x, a\}$
Blue edges minimally non-tight.
Adjust prices.
Some more tight edges.
And X shows
a "new" nontight edge.
..and another augmentation...

..and finally: a perfect matching.

All matched edges tight.
Perfect matching. Feasible price function. Values the same. Optimal!

Notice:
no weights on the right problem.
retain previous matching through price changes.
retains edges in failed search through price changes.

Some thoughts..

Unweighted matching algorithm to weighted.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.
  Primal infeasible.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.
  Primal infeasible.
Primal only “plays” tight constraints.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.
  Primal infeasible.
Primal only “plays” tight constraints. Best offense.
Terminate when perfect matching.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
Dual feasible.
Primal infeasible.
Primal only “plays” tight constraints. Best offense.
Terminate when perfect matching.
$\rightarrow$ Dual only plays tight constraints.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
 Dual feasible.
 Primal infeasible.
Primal only "plays" tight constraints. Best offense.
Terminate when perfect matching.
  $\rightarrow$ Dual only plays tight constraints.
   Dual's best offense.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.
  Primal infeasible.
Primal only “plays” tight constraints. Best offense.
Terminate when perfect matching.
   $\rightarrow$ Dual only plays tight constraints.
      Dual’s best offense.

Some thoughts..

Unweighted matching algorithm to weighted.

How?

Use duality.

In this case:
  Dual feasible.
  Primal infeasible.
Primal only "plays" tight constraints. Best offense.
Terminate when perfect matching.
  $\rightarrow$ Dual only plays tight constraints.
    Dual's best offense.

Equilibrium.

...see you on Tuesday