Given a bipartite graph, $G = (U, V, E)$, with edge weights $w : E \rightarrow R$, 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 $w : E \rightarrow R$, 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.
Matching.

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

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

![Graph with matching edges]

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

Solution Value: 7.

Solution Value: 7.

Solution Value: 8.
Matching.

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

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

\[
\begin{align*}
\text{u} & \quad \text{a} \\
\text{v} & \quad \text{b} \\
\text{w} & \quad \text{c} \\
\text{x} & \quad \text{d}
\end{align*}
\]
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.

\[ \begin{align*}
\text{Solution Value: 7.} \\
\text{Solution Value: 8.}
\end{align*} \]
Matching.

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

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

A function $p : V \rightarrow R$, 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 $w : E \rightarrow R$, find an vertex cover function of minimum total value.

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

Minimize $\sum_{v \in U \cup V} p(u)$.
Given a bipartite graph, \( G = (U, V, E) \), with edge weights \( w : E \rightarrow R \), find an vertex cover function of minimum total value.

A function \( p : V \rightarrow R \), 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 $w : E \rightarrow R$, find an vertex cover function of minimum total value.

A function $p : V \rightarrow R$, 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 $w : E \rightarrow R$, find an vertex cover function of minimum total value.

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

A function $p : V \rightarrow R$, 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 $e = (u, v)$, $p(u) + p(v) \geq w(e)$.

\begin{center}
\begin{tikzpicture}
  \node[circle, draw] (u) at (0,0) {$u$};
  \node[circle, draw] (v) at (1,0) {$v$};
  \draw (u) -- node[above] {$w(e)$} (v);
  \node at (0,-0.5) {$p(u)$};
  \node at (1,-0.5) {$p(v)$};
\end{tikzpicture}
\end{center}
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 $M$. 
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 \( M \).

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

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

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

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 $M$.

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

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.

Proof of optimality. Matching and cover are optimal, edges in matching have $w(e) = p(u) + p(v)$. Tight edge. All nodes are matched.
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.
Value: 2.
    Same as optimal matching!
Proof of optimality.

\[
\begin{array}{c}
\text{1} & a & x & 0 \\
\text{2} & b & y & 0 \\
\text{0} & a & x & 1 \\
\text{1} & b & y & 0 \\
\end{array}
\]
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.
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.
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:

![Graph Example](attachment:graph.png)
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),

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.
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?

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.
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:
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 vector cover (price function) $p(\cdot)$ and matching where.

$\text{Optimal solutions to both if } e \in M, w(e) = p(u) + p(v)$ (Defn: tight edge.) and perfect matching.
Want vector cover (price function) $p(\cdot)$ and matching where. Optimal solutions to both if
Want vector 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
Want vector cover (price function) $p(\cdot)$ and matching where.

Optimal solutions to both if

for $e \in M$, $w(e) = p(u) + p(v)$ \textbf{(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
Start with empty matching, feasible cover function \( \rho(\cdot) \)
Maximum Weight Matching

Goal: perfect matching on tight edges.

Algorithm

Start with empty matching, feasible cover function ($\rho(\cdot)$)
Add tight edges to matching.
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
Add tight edges to matching.
  Use alt./aug. paths of tight edges.

\[ \delta = \min_{e \in (S \cup T \times T \cup S)} \{ w(e) - \rho(u) - \rho(v) \}. \]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((p(\cdot))\)
Add tight edges to matching.
Use alt./aug. paths of tight edges.
"maximum matching algorithm."

\[ \delta = \min_{e \in (S \cup T \times S \cup T)} \left( w(e) - p(u) - p(v) \right) \]

... and get new tight edge!
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function ($\rho(\cdot)$)
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
Start with empty matching, feasible cover function \((\rho(\cdot))\)
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!

\[ \delta = \min_{e \in (S \times U \times T \times V)} \{w(e) - \rho(u) - \rho(v)\} \]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \( (\rho(\cdot)) \)
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**

Start with empty matching, feasible cover function \((\rho(\cdot))\)

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 \(S_U, T_V\).

\[
\delta = \min_{e \in (S \cup S') \times T'} \{\omega(e) - \rho(u) - \rho(v)\}.
\]
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with empty matching, feasible cover function \((\rho(\cdot))\)
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
Start with empty matching, feasible cover function \( \rho(\cdot) \)
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

Start with 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
Start with 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,
   backward edges still feasible
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with 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,
backward edges still feasible
... and get new tight edge!
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm

Start with empty matching, feasible cover function \( (\rho(\cdot)) \)

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,
backward edges still feasible

... and get new tight edge!

What’s delta?
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with 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, $\mathcal{C}(S, T)$, in directed graph of tight edges!
All edges across cut are not tight. (loose?)
Nontight edges leaving cut, go from $S \cup T$.
Lower prices in $S$, raise prices in $S \cup T$, all explored edges still tight,
backward edges still feasible ...
and get new tight edge!

What's delta?

$$w(e) > p(u) + p(v) \rightarrow \delta = \min_{e \in \mathcal{C}(S \cup T \times T)} w(e) - p(u) - p(v).$$
Maximum Weight Matching
Goal: perfect matching on tight edges.

Algorithm
Start with 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,
backward edges still feasible
... and get new tight edge!
What’s delta? \( w(e) > p(u) + p(v) \) →
\( \delta = \min_{e \in (S_U \times T_V)} w(e) - p(u) - p(v) \).
Lecture 2 ended in the middle of the previous slide. A question was asked why don’t we just drop prices around the blue (loose) edge in the figure. Why not?
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!
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: $\rho(u) = \text{maximum incident edge for } u \in U$, 0 otherwise.
Some details

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

Beginning “Matcher” Solution: $M = \emptyset$.

Feasible! Value = 0.

Beginning “Coverer” Solution: $\rho(u) =$ 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: $M = \emptyset$.
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.
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

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.
Example

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

Notice: no weights on the right problem. Retains edges in failed search through price changes.
Example

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.
Example

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.
Example

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

Notice: no weights on the right problem. Retains edges in failed search through price changes.
Example

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.
Example

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

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

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

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

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

Notice: no weights on the right problem. Retains edges in failed search through price changes.
Example

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.
Example

All matched edges tight.

Perfect matching.

Feasible price function.

Optimal!

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

All matched edges tight.
Perfect matching.

Notice:
- no weights on the right problem.
- retains edges in failed search through price changes.
Example

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

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

Notice:
Example

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

Notice:
  no weights on the right problem.
Example

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

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.