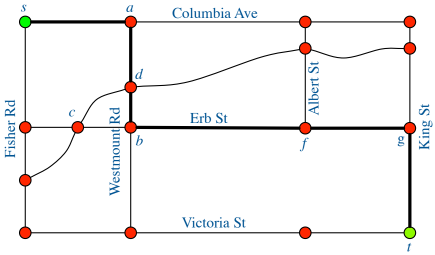
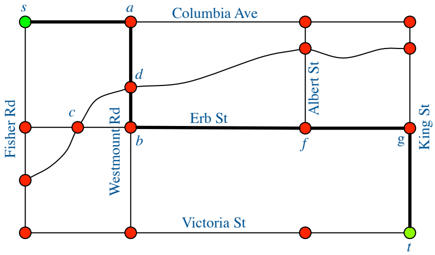
Module 1: Formulations (Optimization on Graphs)

Shortest Paths



- Familiar problem: Starting at location s, we wish to travel to t.
  What is the best (i.e., shortest) route?

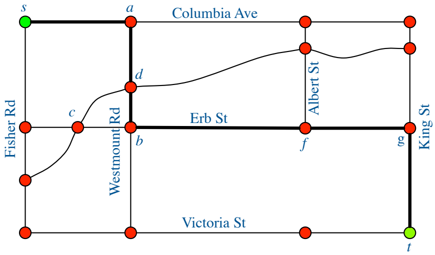
Shortest Paths



- Familiar problem: Starting at location s, we wish to travel to t. What is the best (i.e., shortest) route?

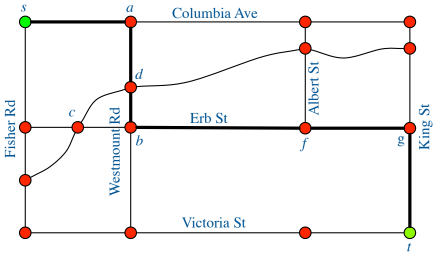
- In the figure above, such a route is indicated in bold.

Shortest Paths



- Goal: Write the problem of finding the shortest route between s and t as an integer program!

Shortest Paths



- Goal: Write the problem of finding the shortest route between s and t as an integer program!
  ... How?

Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

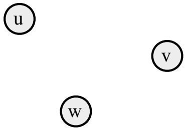
A graph G consists of ...

Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$

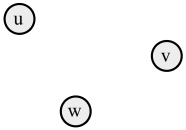


Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)



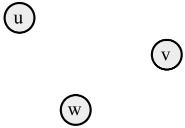
Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)

- edges $uw, wz, \ldots \in E$

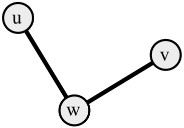


Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)
- edges $uw, wz, \ldots \in E$
  (drawn as lines connecting circles)



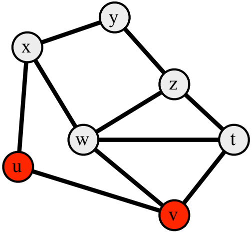
Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)
- edges $uw, wz, \ldots \in E$
  (drawn as lines connecting circles)

Two vertices u and v are adjacent if $uv \in E$.



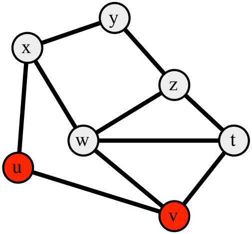
Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)
- edges $uw, wz, \ldots \in E$
  (drawn as lines connecting circles)

Two vertices u and v are adjacent if $uv \in E$. Vertices u and v are the endpoints of edge $uv \in E$,



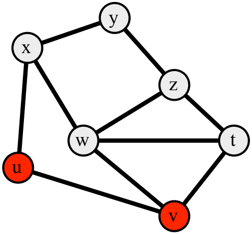
Graph Theory 101

Rephrasing this problem in the language of graph theory helps!

A graph G consists of ...

- vertices $u, w, \ldots \in V$
  (drawn as filled circles)
- edges $uw, wz, \ldots \in E$
  (drawn as lines connecting circles)

Two vertices u and v are adjacent if $uv \in E$. Vertices u and v are the endpoints of edge $uv \in E$, and edge $e \in E$ is incident to $u \in V$ if u is an endpoint of e.



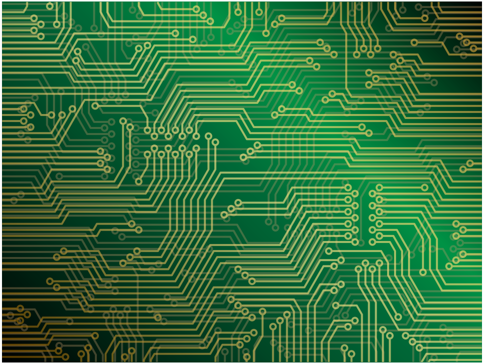
Graphs – Why do we care?

Graphs are useful to compactly model many real-world entities.

Graphs – Why do we care?

Graphs are useful to compactly model many real-world entities. For example:

- Modeling circuits in chip design

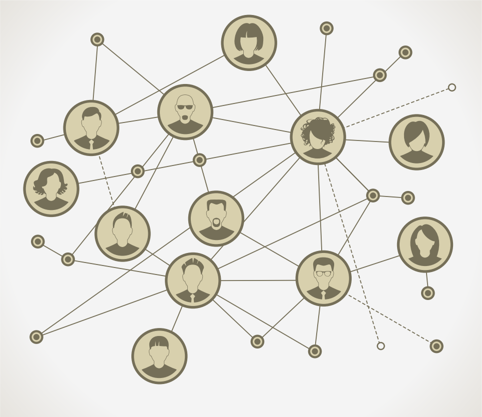


Eyematrix/iStock/Thinkstock

Graphs – Why do we care?

Graphs are useful to compactly model many real-world entities. For example:

- Modeling circuits in chip design
- Social networks

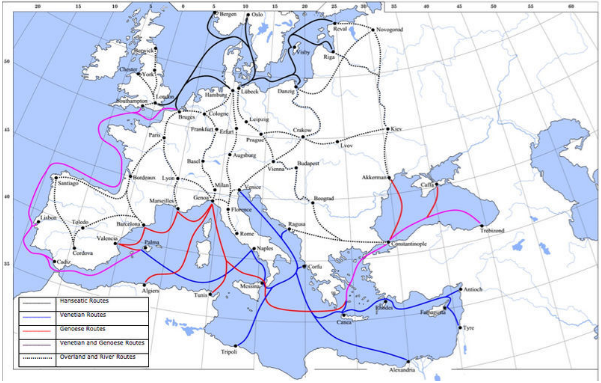


VLADGRIN/iStock/Thinkstock

Graphs – Why do we care?

Graphs are useful to compactly model many real-world entities. For example:

- Modeling circuits in chip design
- Social networks
- Trade networks

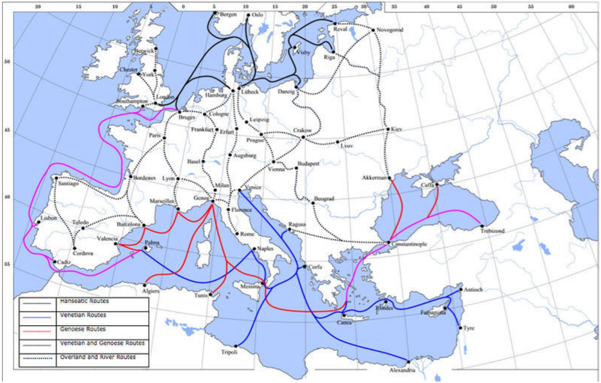


Lampman, 2008 [Online Image]. Late Medieval Trade Routes. Wikimedia Commons.
http://commons.wikimedia.org/wiki/File:Late_Medieval_Trade_Routes.jpg

Graphs – Why do we care?

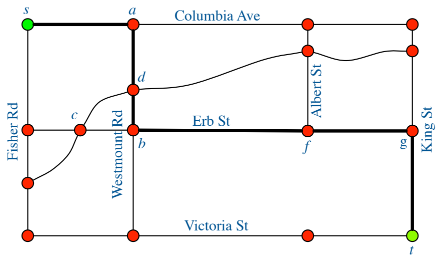
Graphs are useful to compactly model many real-world entities. For example:

- Modeling circuits in chip design
- Social networks
- Trade networks
- .... and many more!



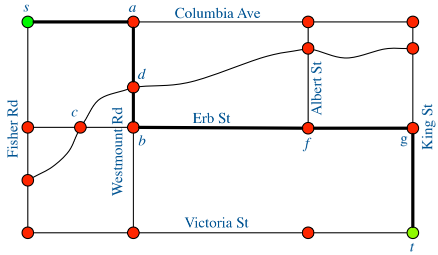
Lampman, 2008 [Online Image]. Late Medieval Trade Routes. Wikimedia Commons. http://commons.wikimedia.org/wiki/File:Late_Medieval_Trade_Routes.jpg

The Map of Water Town



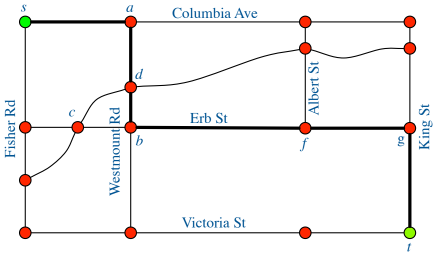
- We can think of the street map as a graph, G.

The Map of Water Town



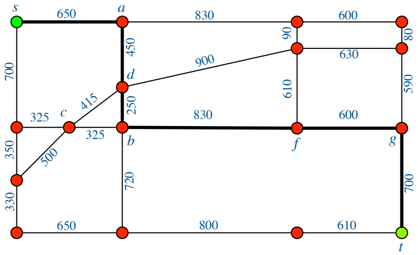
- We can think of the street map as a graph, G.
- Vertices: Road intersections

The Map of Water Town



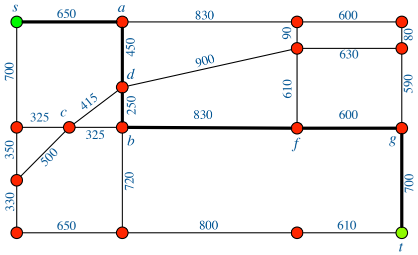
- We can think of the street map as a graph, G.
- Vertices: Road intersections
- Edges: Road segments connecting adjacent intersections

The Map of Water Town



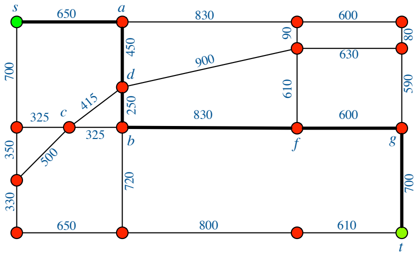
- Each edge $e \in E$ is labelled by its length $c_e \geq 0$.

The Map of Water Town



- Each edge $e \in E$ is labelled by its length $c_e \geq 0$.
- We are looking for a path connecting s and t of smallest total length!

Paths



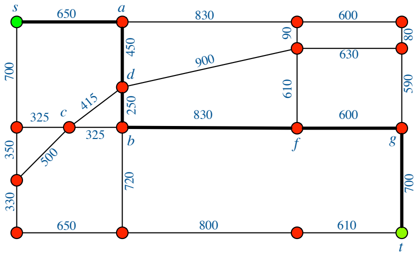
An s, t-path in $G = (V, E)$ is a sequence

$$v_1v_2, v_2v_3, v_3v_4, \ldots, v_{k-2}v_{k-1}, v_{k-1}v_k$$

where

- $v_i \in V$ and $v_iv_{i+1} \in E$ for all i, and

Paths



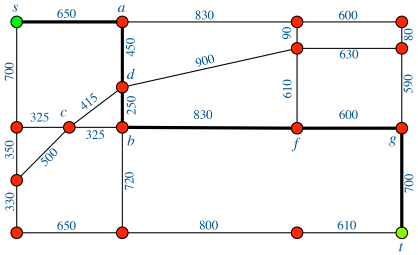
An s,t-path in $G = (V, E)$ is a sequence

$$v_1v_2, v_2v_3, v_3v_4, \ldots, v_{k-2}v_{k-1}, v_{k-1}v_k$$

where

- $v_i \in V$ and $v_iv_{i+1} \in E$ for all i, and
- $v_1 = s$, $v_k = t$, and $v_i \neq v_j$ for all $i \neq j$.

Paths



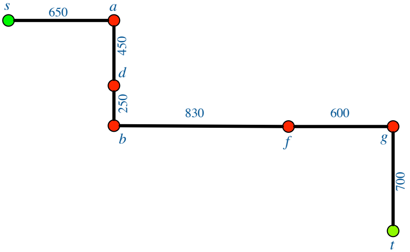
An s, t-path in $G = (V, E)$ is a sequence

$$v_1v_2, v_2v_3, v_3v_4, \ldots, v_{k-2}v_{k-1}, v_{k-1}v_k$$

where

- $v_i \in V$ and $v_iv_{i+1} \in E$ for all i, and
- $v_1 = s$, $v_k = t$, and $v_i \neq v_j$ for all $i \neq j$.
  (Without this, it is called an s, t-walk)

Paths



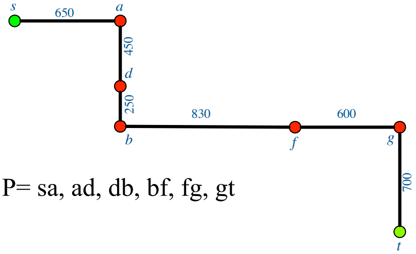
An s, t-path in $G = (V, E)$ is a sequence

$$v_1v_2, v_2v_3, v_3v_4, \ldots, v_{k-2}v_{k-1}, v_{k-1}v_k$$

where

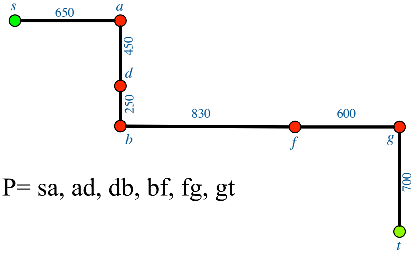
- $v_i \in V$ and $v_iv_{i+1} \in E$ for all i, and
- $v_1 = s$, $v_k = t$, and $v_i \neq v_j$ for all $i \neq j$.
  (Without this, it is called an s, t-walk)

Paths



P= sa, ad, db, bf, fg, gt

Paths

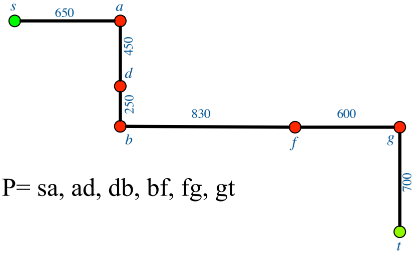


P= sa, ad, db, bf, fg, gt

The length of a path $P = v_1v_2, \ldots, v_{k-1}v_k$ is the sum of the lengths of the edges on P:

$$c(P) := \sum(c_e \ : \ e \in P).$$

Paths

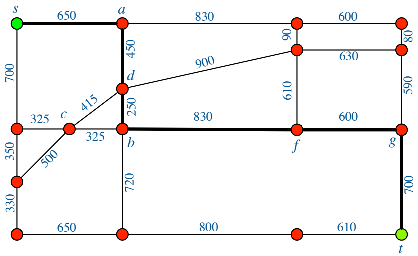


P= sa, ad, db, bf, fg, gt

$$
\begin{aligned}
c(P) &= c_{sa} + c_{ad} + c_{db} + c_{bf} + c_{fg} + c_{gt} \\
&= 650 + 490 + 250 + 830 + 600 + 700 \\
&= 3520
\end{aligned}
$$

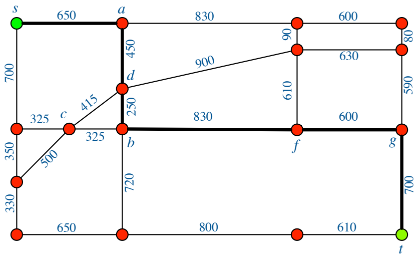
The length of a path $P = v_1v_2, \dots, v_{k-1}v_k$ is the sum of the lengths of the edges on P:

$$c(P) := \sum(c_e \, : \, e \in P).$$



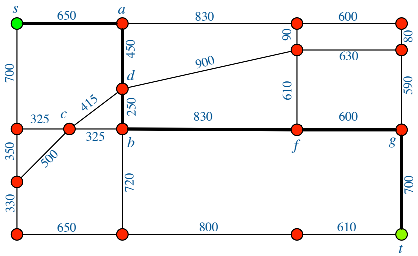
Shortest Path Problem

- Given: Graph $G = (V, E)$, lengths $c_e \geq 0$ for all $e \in E$, $s, t \in V$



Shortest Path Problem

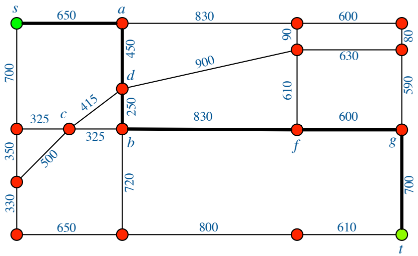
- Given: Graph $G = (V, E)$, lengths $c_e \geq 0$ for all $e \in E$, $s, t \in V$
- Find: Minimum-length s, t-path P



Shortest Path Problem

- Given: Graph $G = (V, E)$, lengths $c_e \geq 0$ for all $e \in E$, $s, t \in V$
- Find: Minimum-length s, t-path P

Goal: Write an IP whose solutions are the shortest s, t-paths!



Shortest Path Problem

- Given: Graph $G = (V, E)$, lengths $c_e \geq 0$ for all $e \in E$, $s, t \in V$
- Find: Minimum-length s, t-path P

Goal: Write an IP whose solutions are the shortest s, t-paths!

$\longrightarrow$ Later!

Example: Matchings

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

Employees have different skill-sets and may take different amounts of time to execute a job.

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

Employees have different skill-sets and may take different amounts of time to execute a job.

Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	2	-
3	-	1	-	-
4	8	-	3	-

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

Employees have different skill-sets and may take different amounts of time to execute a job.

Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	2	-
3	-	1	-	-
4	8	-	3	-

Note: Some workers are not able to handle certain jobs!

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

Employees have different skill-sets and may take different amounts of time to execute a job.

Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	2	-
3	-	1	-	-
4	8	-	3	-

Note: Some workers are not able to handle certain jobs!

Goal: Assign each worker to exactly one task so that the total execution time is smallest!

WaterTech - Job Assignment

WaterTech has a collection of important jobs:

$$J = \{1', 2', 3', 4'\}$$

that it needs to handle urgently.

It also has 4 employees:

$$E = \{1, 2, 3, 4\}$$

that need to handle these jobs.

Employees have different skill-sets and may take different amounts of time to execute a job.

Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	2	-
3	-	1	-	-
4	8	-	3	-

Note: Some workers are not able to handle certain jobs!

Goal: Assign each worker to exactly one task so that the total execution time is smallest!

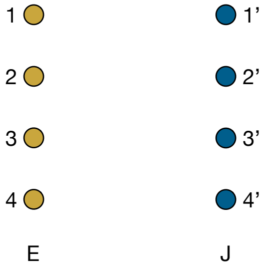
$\longrightarrow$ We will rephrase this in the language of graphs

Matchings

Create a graph with one vertex for each employee and job.

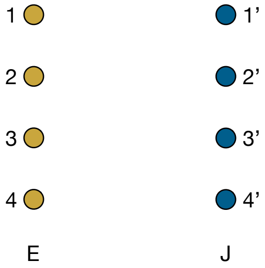
Matchings

Create a graph with one vertex for each employee and job.



Matchings

Create a graph with one vertex for each employee and job.

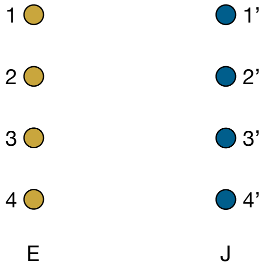


Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

Matchings

Create a graph with one vertex for each employee and job.

Add an edge ij for $i \in E$ and $j \in J$ if employee i can handle job j.

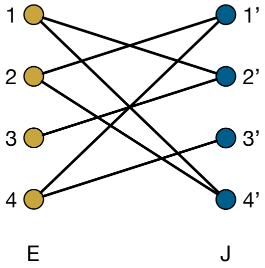


Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

Matchings

Create a graph with one vertex for each employee and job.

Add an edge ij for $i \in E$ and $j \in J$ if employee i can handle job j.



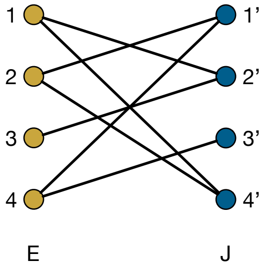
Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

Matchings

Create a graph with one vertex for each employee and job.

Add an edge ij for $i \in E$ and $j \in J$ if employee i can handle job j.

Let the cost c_{ij} of edge ij be the amount of time needed by i to complete j.



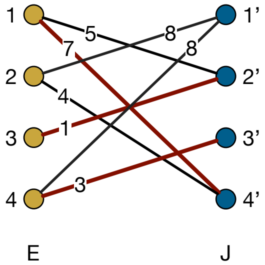
Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

Matchings

Create a graph with one vertex for each employee and job.

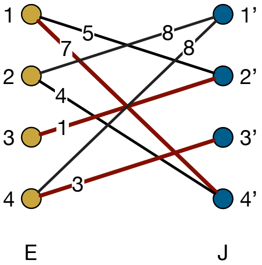
Add an edge ij for $i \in E$ and $j \in J$ if employee i can handle job j.

Let the cost c_{ij} of edge ij be the amount of time needed by i to complete j.



Employees	Jobs			
	$1'$	$2'$	$3'$	$4'$
1	-	5	-	7
2	8	-	-	4
3	-	1	-	-
4	8	-	3	-

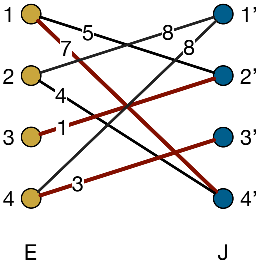
Matchings



Definition

A collection $M \subseteq E$ is a matching if no two edges $ij, i'j' \in M$ $(ij \neq i'j')$ share an endpoint;

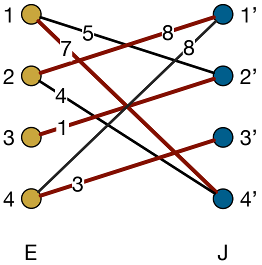
Matchings



Definition

A collection $M \subseteq E$ is a matching if no two edges $ij, i'j' \in M$ $(ij \neq i'j')$ share an endpoint; i.e., $\{i, j\} \cap \{i', j'\} = \emptyset$.

Matchings



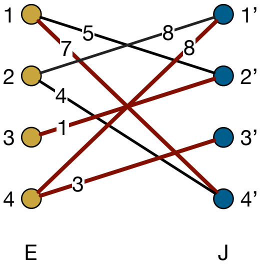
Definition

A collection $M \subseteq E$ is a matching if no two edges $ij, i'j' \in M$ $(ij \neq i'j')$ share an endpoint; i.e., $\{i, j\} \cap \{i', j'\} = \emptyset$.

Examples

1. $M = \{14', 21', 32', 43'\}$ is a matching.

Matchings



Definition

A collection $M \subseteq E$ is a matching if no two edges $ij, i'j' \in M$ $(ij \neq i'j')$ share an endpoint; i.e., $\{i, j\} \cap \{i', j'\} = \emptyset$.

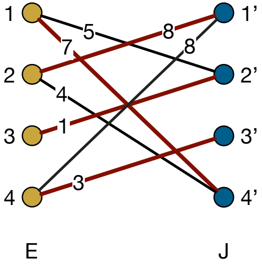
Examples

1. $M = \{14', 21', 32', 43'\}$ is a matching.
2. $M = \{14', 32', 41', 43'\}$ is not a matching.

Matchings

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e \, : \, e \in M)$$

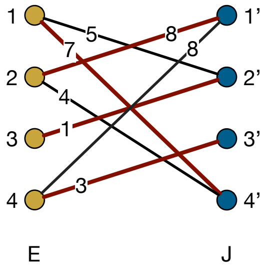


Matchings

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e \,:\, e \in M)$$

e.g., $M = \{14', 21', 32', 43'\}$
$\longrightarrow c(M) = 19$

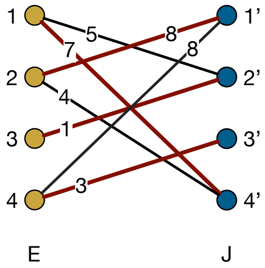


Matchings

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e \, : \, e \in M)$$

e.g., $M = \{14', 21', 32', 43'\}$
$\longrightarrow c(M) = 19$



Definition

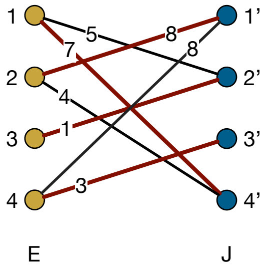
A matching M is perfect if every vertex v in the graph is incident to an edge in M.

Matchings

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e \,:\, e \in M)$$

e.g., $M = \{14', 21', 32', 43'\}$
$\longrightarrow c(M) = 19$



Definition

A matching M is perfect if every vertex v in the graph is incident to an edge in M.

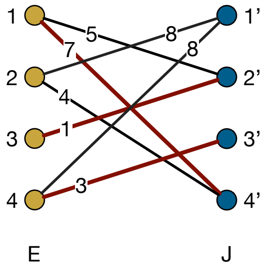
E.g., matching in figure is perfect,

Matchings

The cost of a matching M is the sum of costs of its edges:

$$c(M) = \sum (c_e \,:\, e \in M)$$

e.g., $M = \{14', 21', 32', 43'\}$
$\longrightarrow c(M) = 19$



Definition

A matching M is perfect if every vertex v in the graph is incident to an edge in M.

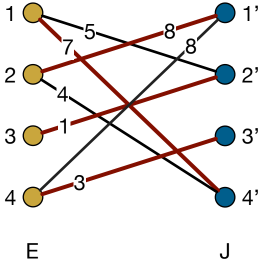
E.g., matching in figure is perfect, and this one is not!

Restating the Assignment Problem

Definition

A matching M is perfect if every vertex v in the graph is incident to an edge in M.

Note: Perfect matchings correspond to feasible assignments of workers to jobs!



Restating the Assignment Problem

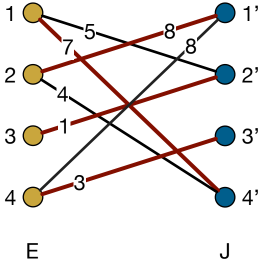
Definition

A matching M is perfect if every vertex v in the graph is incident to an edge in M.

Note: Perfect matchings correspond to feasible assignments of workers to jobs!

E.g., the matching shown corresponds to the following assignment:

$$1 \to 4',\ 2 \to 1',\ 3 \to 2',\ \text{and } 4 \to 3'$$



Restating the Assignment Problem

Definition

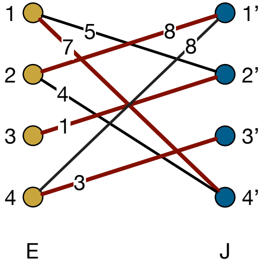
A matching M is perfect if every vertex v in the graph is incident to an edge in M.

Note: Perfect matchings correspond to feasible assignments of workers to jobs!

E.g., the matching shown corresponds to the following assignment:

$$1 \to 4',\ 2 \to 1',\ 3 \to 2',\ \text{and } 4 \to 3'$$

whose execution time equals $c(M) = 19$!



Restating the Assignment Problem

Definition

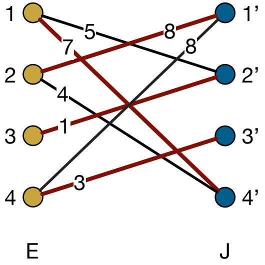
A matching M is perfect if every vertex v in the graph is incident to an edge in M.

Note: Perfect matchings correspond to feasible assignments of workers to jobs!

E.g., the matching shown corresponds to the following assignment:

$$1 \to 4',\, 2 \to 1',\, 3 \to 2',\, \text{and } 4 \to 3'$$

whose execution time equals $c(M) = 19$!



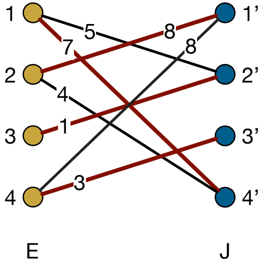
Restatement of original question:

Find a perfect matching M in our graph of smallest cost.

A Little More Notation...

Notation: Use $\delta(v)$ to denote the set of edges incident to v; i.e.,

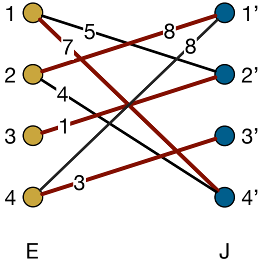
$$\delta(v) = \{e \in E \,:\, e = vu \text{ for some } u \in V\}.$$



A Little More Notation...

Notation: Use $\delta(v)$ to denote the set of edges incident to v; i.e.,

$$\delta(v) = \{e \in E \,:\, e = vu \text{ for some } u \in V\}.$$



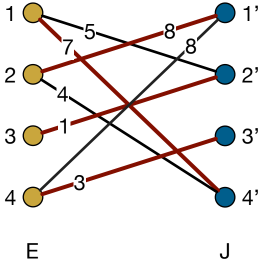
Examples

- $\delta(2) = \{21', 24'\}$

A Little More Notation...

Notation: Use $\delta(v)$ to denote the set of edges incident to v; i.e.,

$$\delta(v) = \{e \in E \,:\, e = vu \text{ for some } u \in V\}.$$



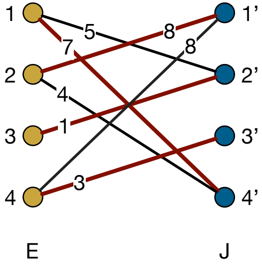
Examples

- $\delta(2) = \{21', 24'\}$
- $\delta(3') = \{43'\}$

A Little More Notation...

Notation: Use $\delta(v)$ to denote the set of edges incident to v; i.e.,

$$\delta(v) = \{e \in E \,:\, e = vu \text{ for some } u \in V\}.$$



Definition

Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.

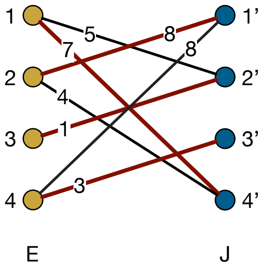
Examples

- $\delta(2) = \{21', 24'\}$
- $\delta(3') = \{43'\}$

An IP for Perfect Matchings

Definition

Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.

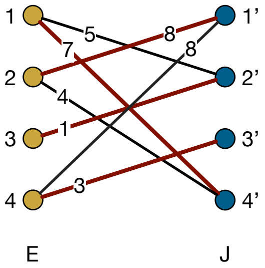


An IP for Perfect Matchings

Definition

Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.

The IP will have a binary variable x_e for every edge $e \in E$.



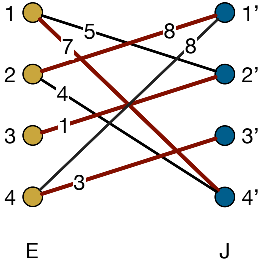
An IP for Perfect Matchings

Definition

Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.

The IP will have a binary variable x_e for every edge $e \in E$. Idea:

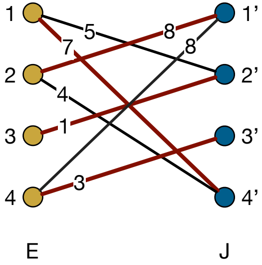
$$x_e = 1 \quad \leftrightarrow \quad e \in M$$



An IP for Perfect Matchings

Definition

Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.



The IP will have a binary variable x_e for every edge $e \in E$. Idea:

$$x_e = 1 \quad \leftrightarrow \quad e \in M$$

Constraints: For all $v \in V$, need

$$\sum (x_e \, : \, e \in \delta(v)) = 1$$

An IP for Perfect Matchings

Definition

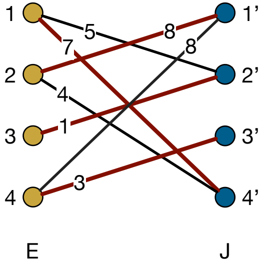
Given $G = (V, E)$, $M \subseteq E$ is a perfect matching iff $M \cap \delta(v)$ contains a single edge for all $v \in V$.

The IP will have a binary variable x_e for every edge $e \in E$. Idea:

$$x_e = 1 \quad \leftrightarrow \quad e \in M$$

Constraints: For all $v \in V$, need

$$\sum (x_e \, : \, e \in \delta(v)) = 1$$

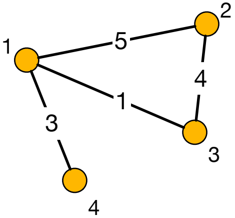


Objective:

$$\sum (c_e x_e \, : \, e \in E)$$

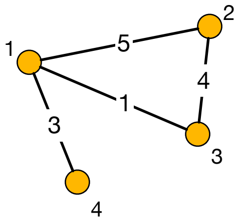
An IP for Perfect Matchings

$$\begin{aligned}
\min \ & \sum(c_e x_e \ : \ e \in E) \\
\text{s.t.} \ & \sum(x_e \ : \ e \in \delta(v)) = 1 \ (v \in V) \\
& x \geq \mathbb{0}, \ x \text{ integer}
\end{aligned}$$



An IP for Perfect Matchings

$$\begin{aligned} \min\ & \sum (c_e x_e \;:\; e \in E) \\ \text{s.t.}\ & \sum (x_e \;:\; e \in \delta(v)) = 1 \;\; (v \in V) \\ & x \geq \mathbb{0},\ x \text{ integer} \end{aligned}$$



$$\min \quad (5,1,3,4)x$$

$$\text{s.t.} \quad \begin{array}{c c} & \begin{array}{cccc} 12 & 13 & 14 & 23 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} x = \mathbb{1}$$

$$x \geq \mathbb{0} \quad \text{integer}$$

Recap

- Graphs consist of vertices V and edges E ... and are very useful in modeling many practical problems.

Recap

- Graphs consist of vertices V and edges E ... and are very useful in modeling many practical problems.
- In particular, graphs can be used to model road networks, where roads are edges and street intersections are vertices.

Recap

- Graphs consist of vertices V and edges E ... and are very useful in modeling many practical problems.
- In particular, graphs can be used to model road networks, where roads are edges and street intersections are vertices.
- In the shortest path problem, each edge $e \in E$ has an associated weight c_e, and we are looking for a path connecting two specific vertices of smallest total weight.

Recap

- Graphs consist of vertices V and edges E ... and are very useful in modeling many practical problems.
- In particular, graphs can be used to model road networks, where roads are edges and street intersections are vertices.
- In the shortest path problem, each edge $e \in E$ has an associated weight c_e, and we are looking for a path connecting two specific vertices of smallest total weight.
- A matching is a collection of edges, no two of which share an endpoint.

Recap

- Graphs consist of vertices V and edges E ... and are very useful in modeling many practical problems.
- In particular, graphs can be used to model road networks, where roads are edges and street intersections are vertices.
- In the shortest path problem, each edge $e \in E$ has an associated weight c_e, and we are looking for a path connecting two specific vertices of smallest total weight.
- A matching is a collection of edges, no two of which share an endpoint. A perfect matching is a matching that *covers* all vertices in V.