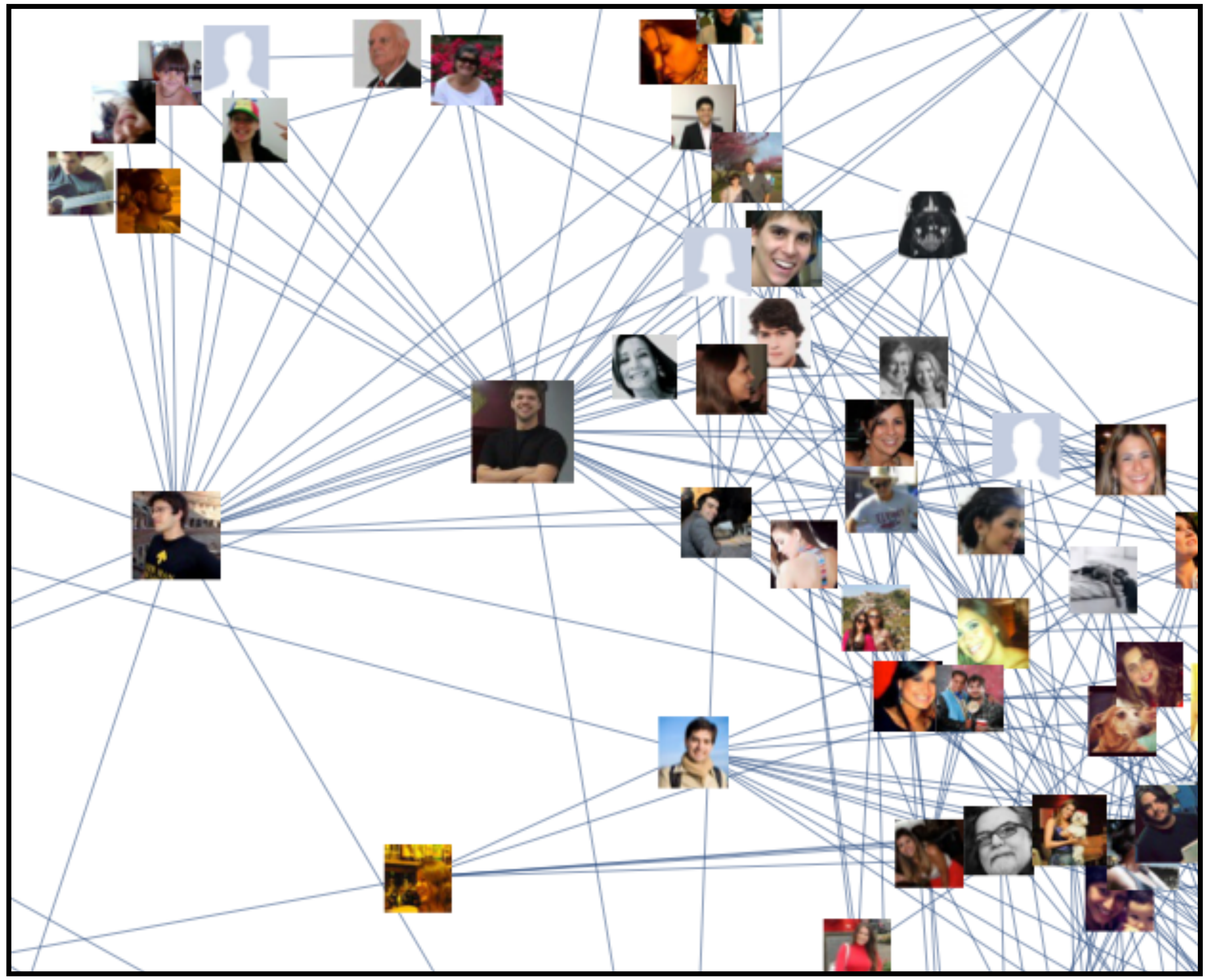
CS251

Great Ideas in *Theoretical* Computer Science

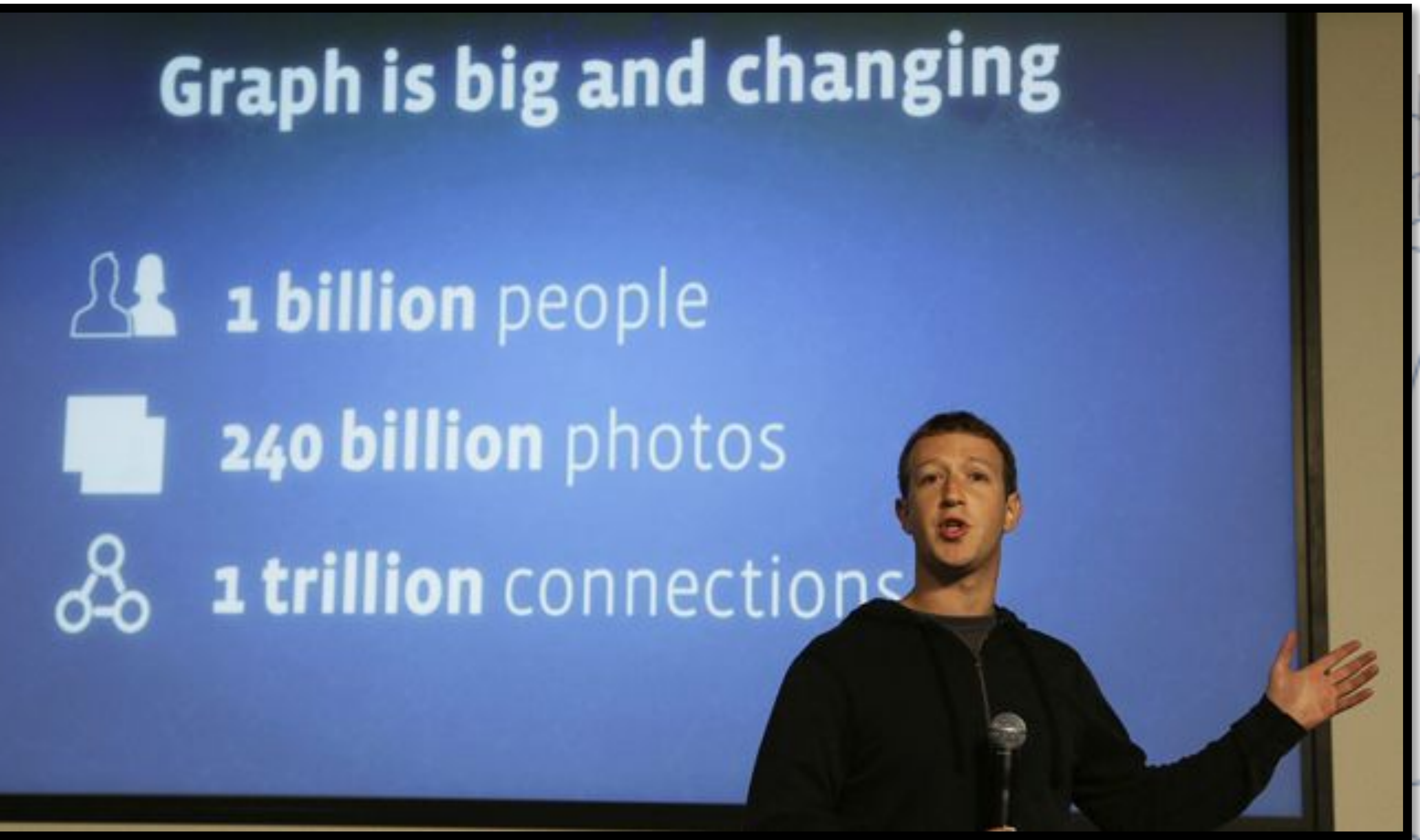


Graphs

Why graphs?

Why now?

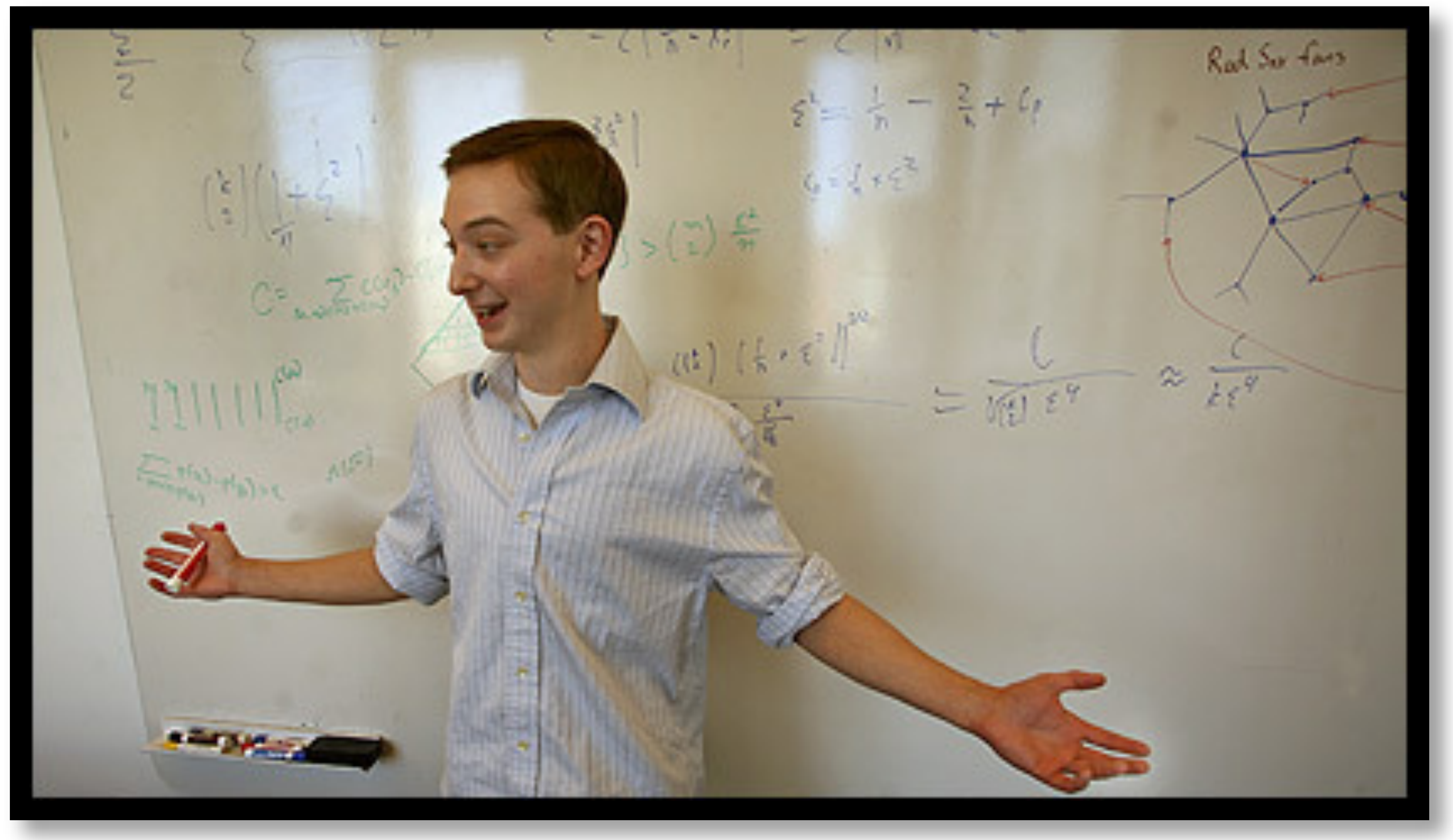
Facebook



Graph is big and changing
1 billion people
240 billion photos
1 trillion connections

Enemybook

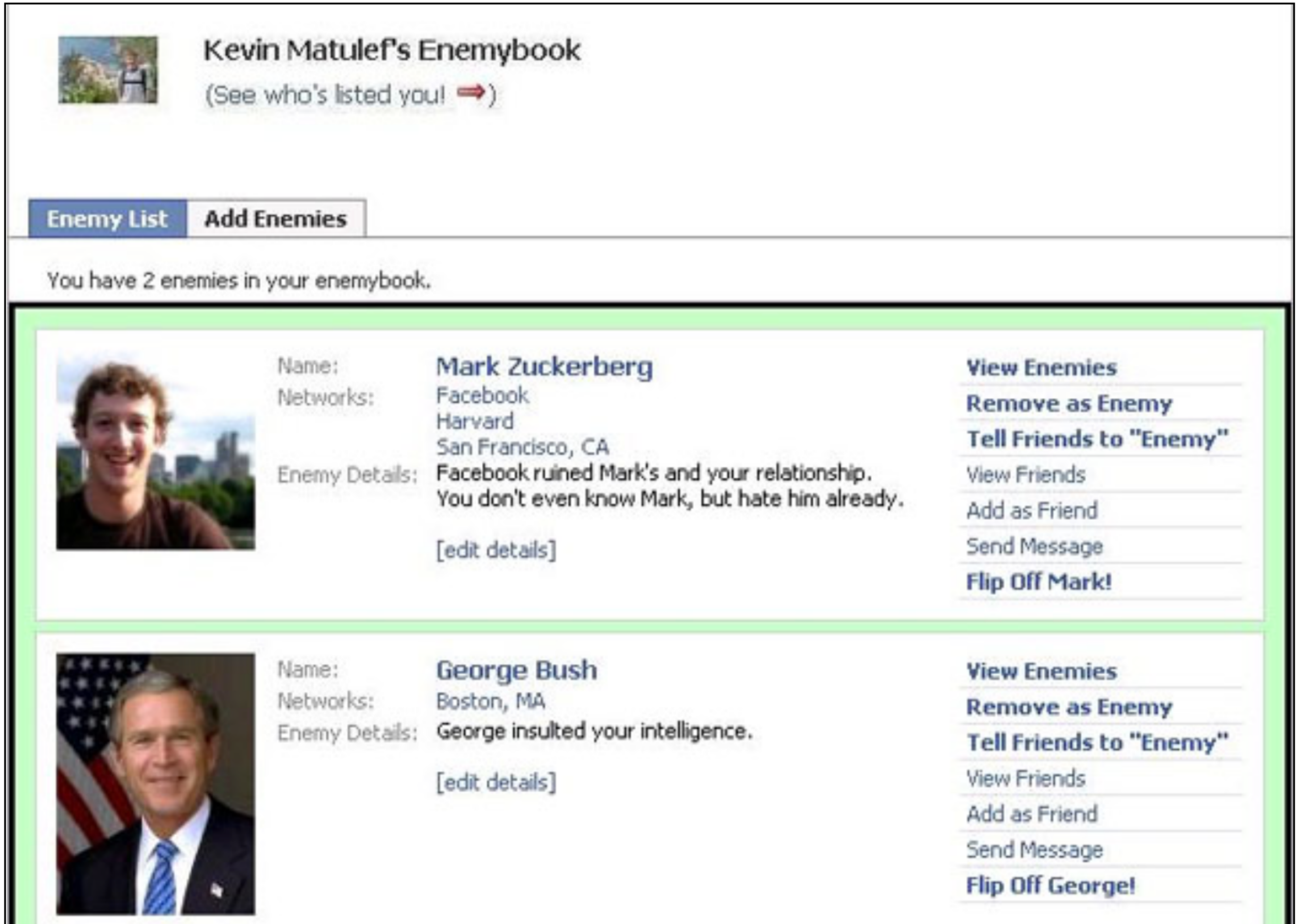
Kevin Matulef



Enemybook remedies the one-sided perspective of Facebook, by allowing you to manage enemies as well as friends. With Enemybook you can **add people** as Facebook enemies, **specify why** they are your enemies, **notify** your enemies, **see who lists you** as an enemy, and even **become friends with the enemies of your enemies**.

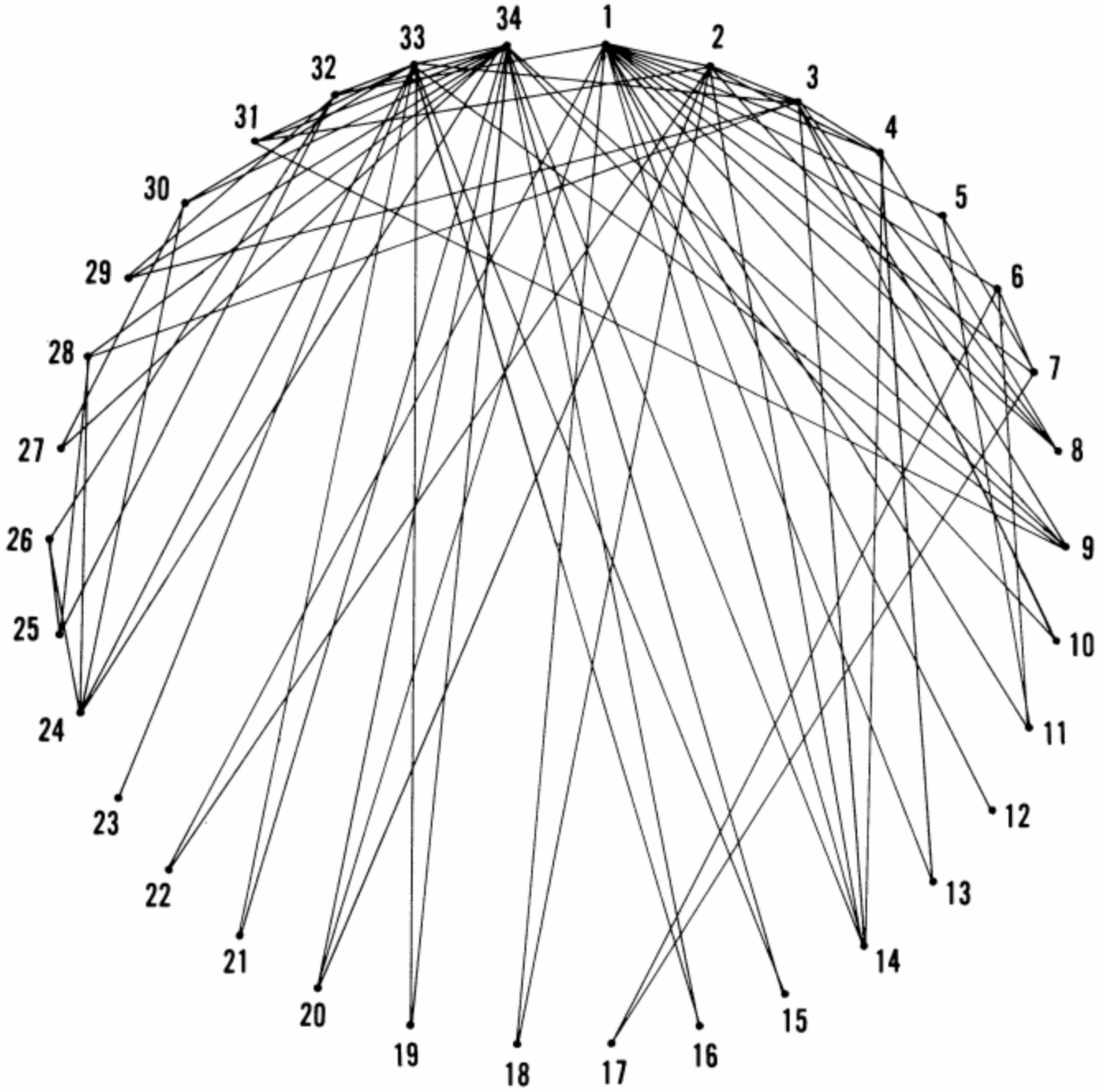
Enemybook

Browse Your Enemies:



Zachary Karate Club

FIGURE 1
Social Network Model of Relationships in the Karate Club



This is the graphic representation of the social relationships among the 34 individuals in the karate club. A line is drawn between two points when the two individuals being represented consistently interacted in contexts outside those of karate classes, workouts, and club meetings. Each such line drawn is referred to as an edge.

Zachary Karate Club CLUB

networkkarate.tumblr.com

Google - Page Rank

1998 paper

2.2 Link Structure of the Web

While estimates vary, the current graph of the crawlable Web has roughly $\boxed{\text{150 million nodes (pages)}}$ $\boxed{\text{and 1.7 billion edges (links).}}$ Every page has some number of forward links (outedges) and backlinks (inedges) (see Figure 1). We can never know whether we have found all the backlinks of a particular page but if we have downloaded it, we know all of its forward links at that time.

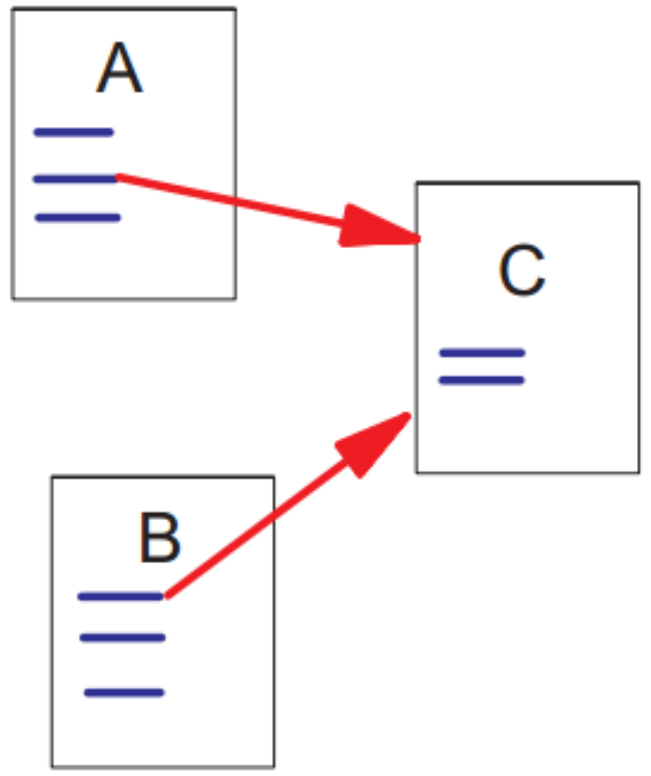


Figure 1: A and B are Backlinks of C

Web pages vary greatly in terms of the number of backlinks they have. For example, the Netscape home page has 62,804 backlinks in our current database compared to most pages which have just a few backlinks. Generally, highly linked pages are more "important" than pages with few links. Simple citation counting has been used to speculate on the future winners of the Nobel Prize [San95]. PageRank provides a more sophisticated method for doing citation counting.

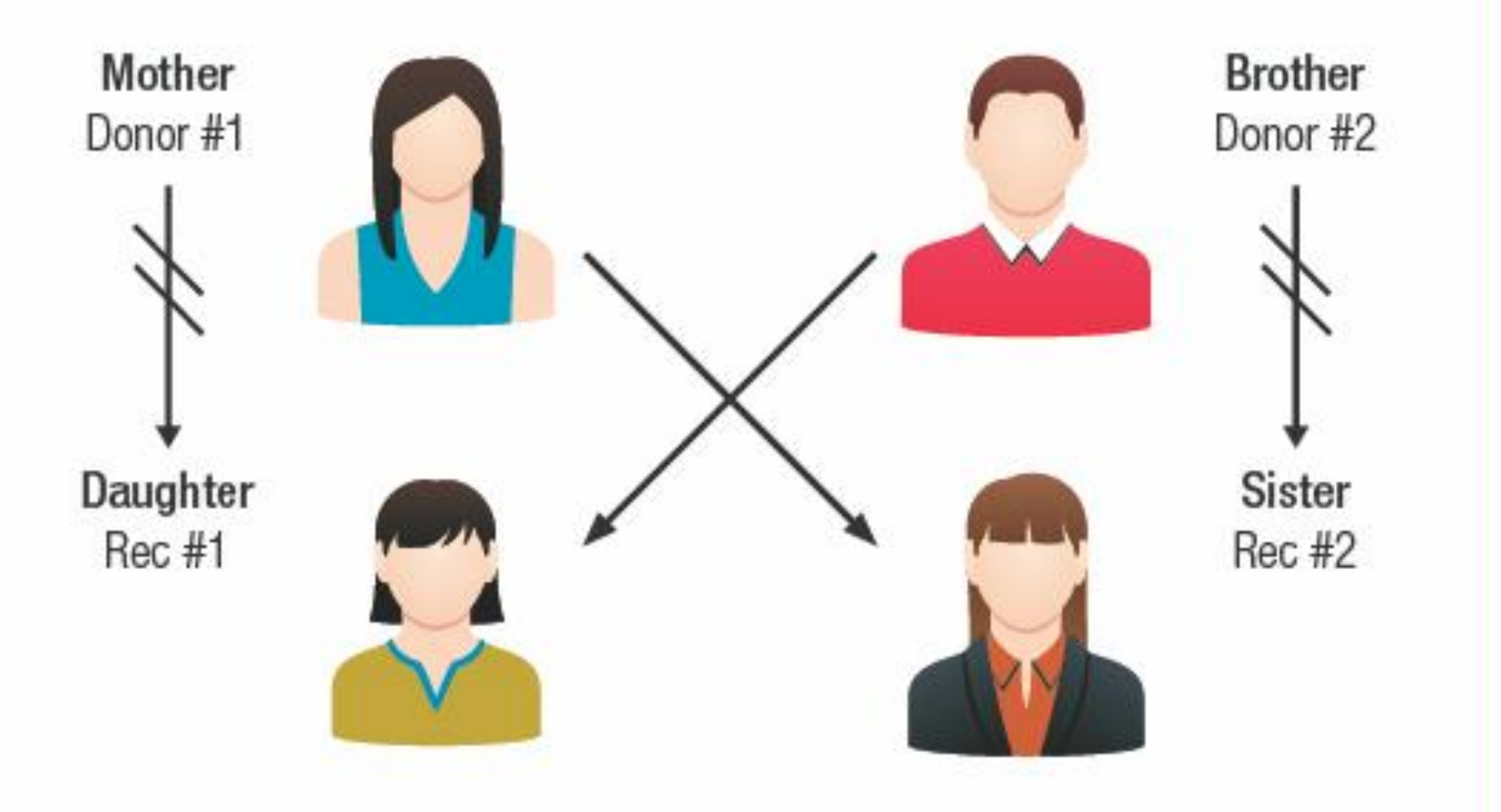
Larry Page

Sergey Brin

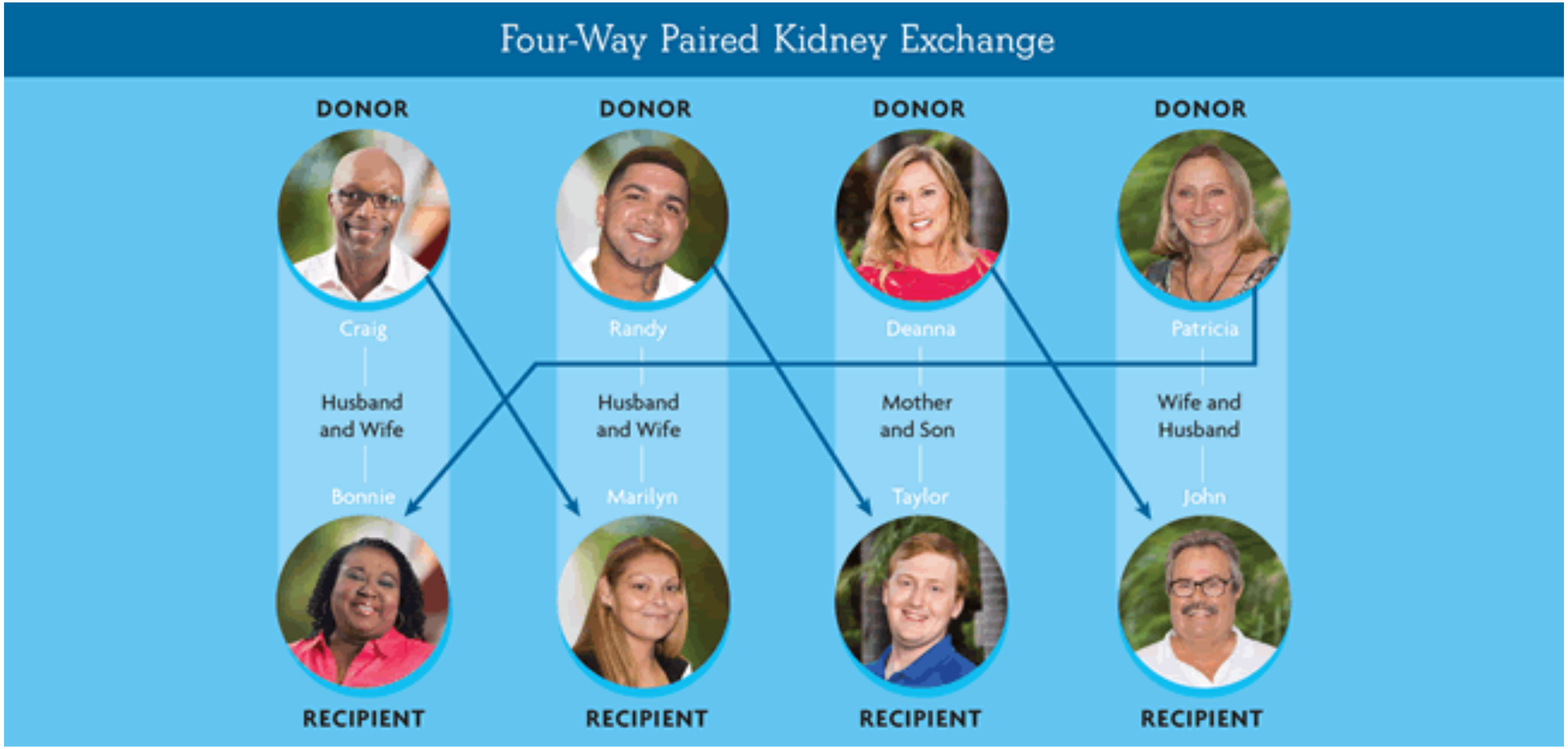
Google Maps


Caffe Nero
Noter Sokak 16-18
19 min
1.5 km
23 min
1.8 km
Marks & Spencer
MIGROS
Bağdat Cd.
Haldun Taner Sk.
Çilek Çk.
Caddebostan İskele Sk.
Selin Sk.
Erenköy Cami Sk.
Ethem Efendi Cd.
Şerafettin Sk.
Nurettin Ali Berkol Sk.
Sancak Sk.
Gözenç Sk.
Acun Sk.
İclal Karabekir Sk.
Kaptan Arif Sok
Tonozlu Sk.
Çolak İsmail Sk.
Kırmızı Sk.
Suadiye Cami Sk.
Çamlı Sk.
Hatboyu Sk.
Akın Hatboyu Çk.
Noter Sk.
Nihat Kızıltan Sk.
Abdulkadir Noyan Sk.
Çetin Emeç Blv.
Tarım İl Md.
Korusu

Kidney Exchange



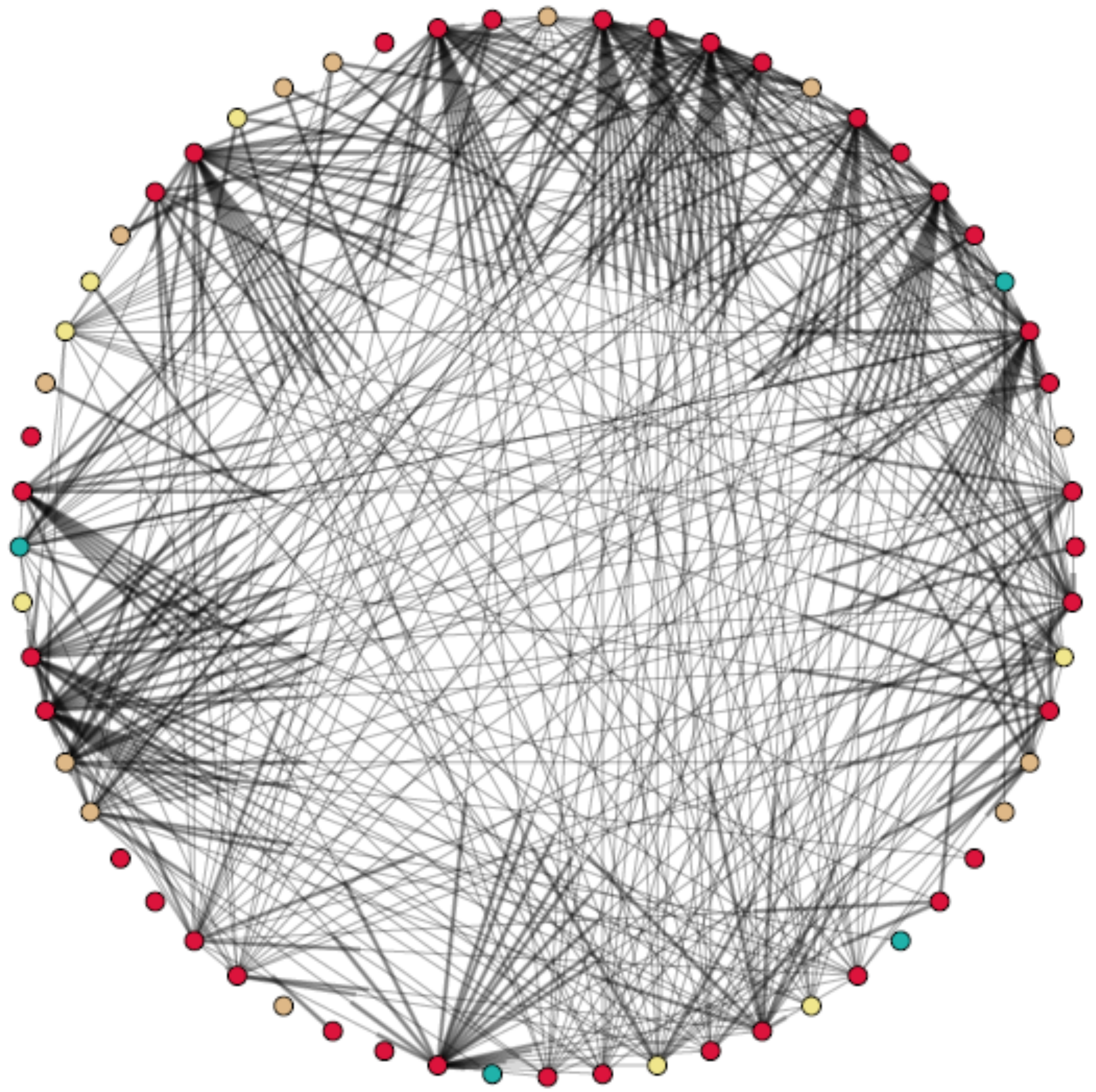
Kidney Exchange



Kidney Exchange

US national kidney exchange program

Vertices = patient-donor pairs, edges = compatibility



UNOS pool, Dec 2010 [Courtesy John Dickerson, CMU]

Tuomas Sandholm (CMU prof.)

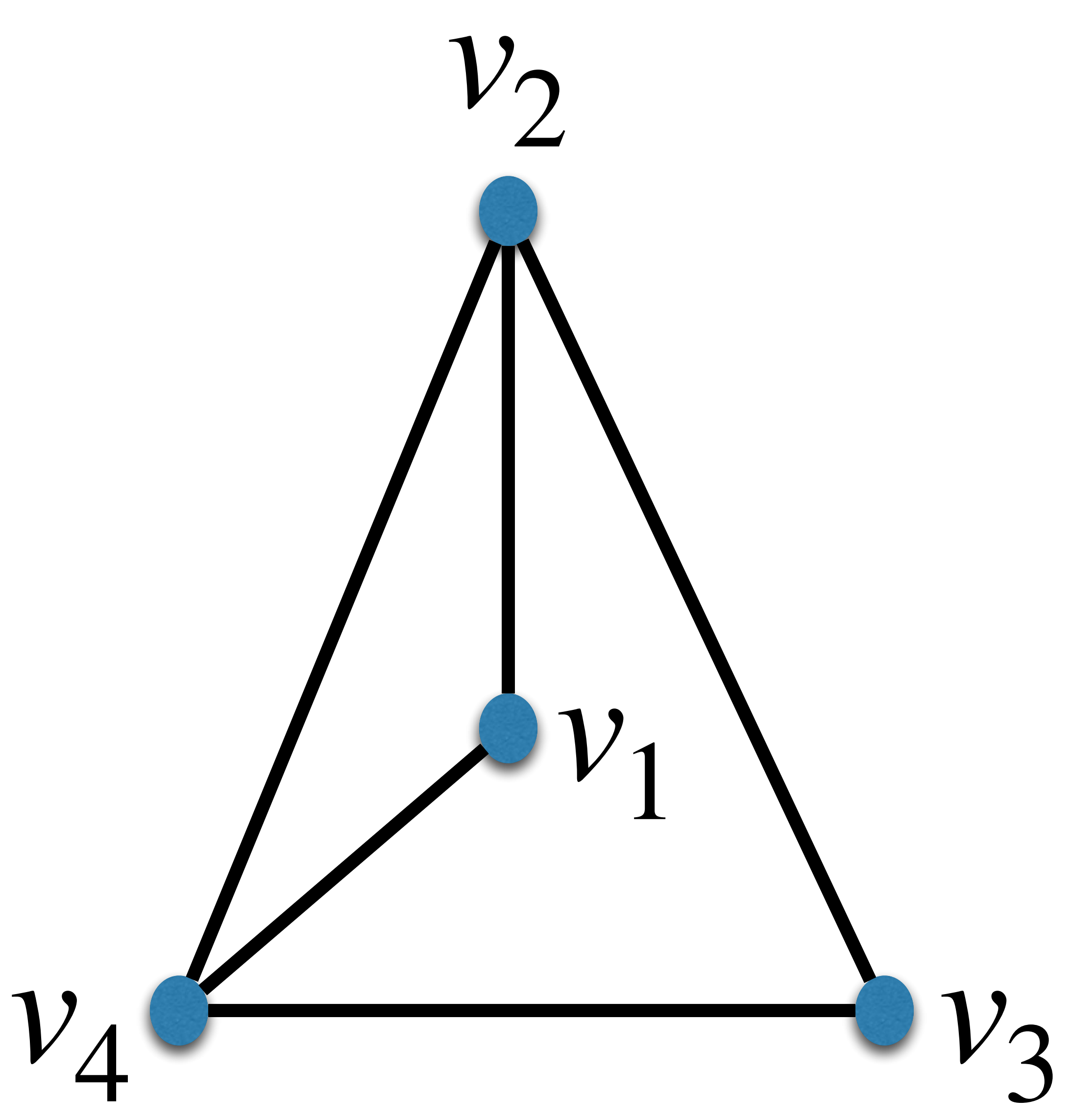
CS Life Lesson

If your problem has a graph, great!!!
If not, try to make it have a graph.

What is a graph?

(A hundred) definitions and basic properties

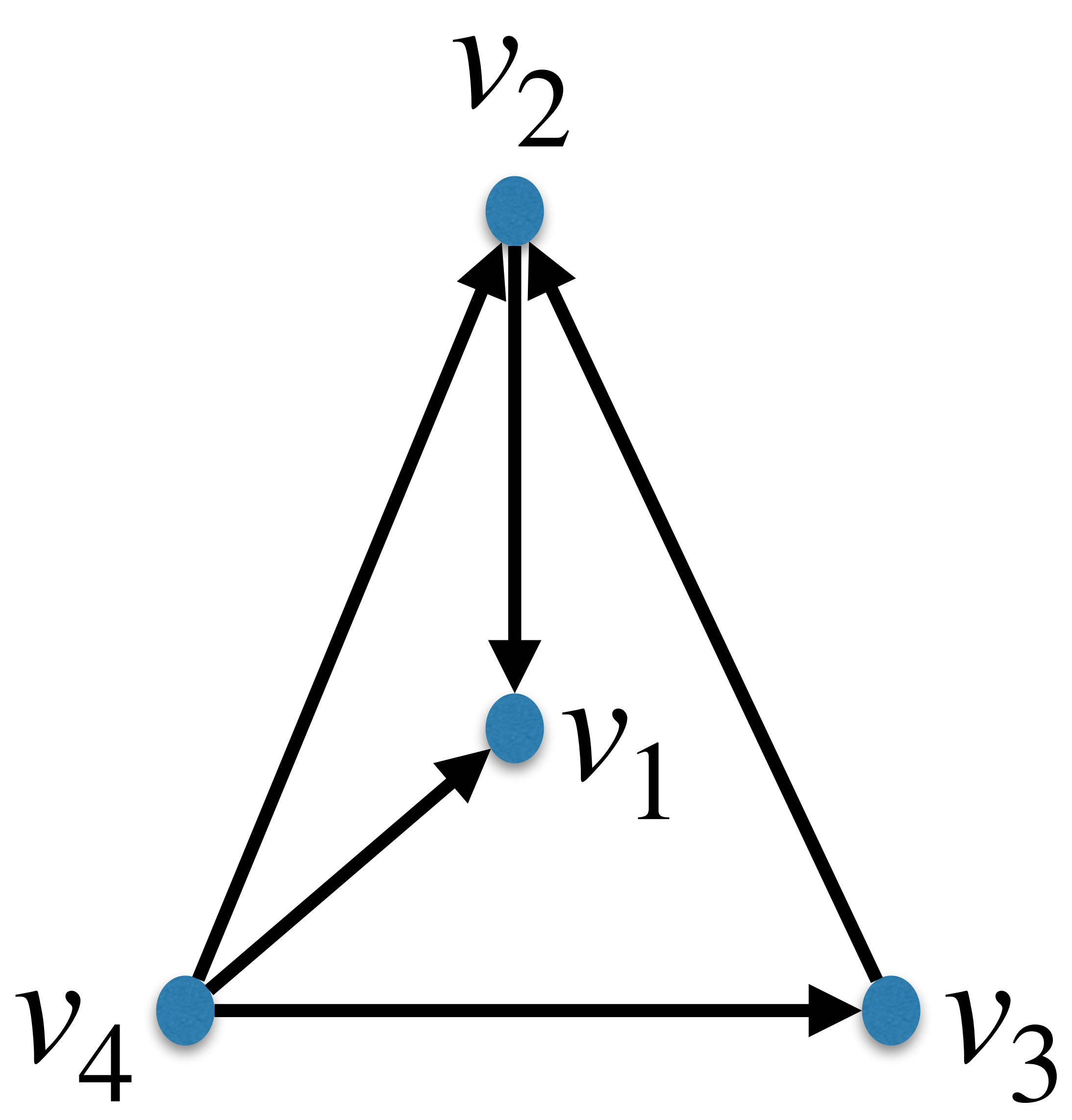
Types of Graphs



Simple

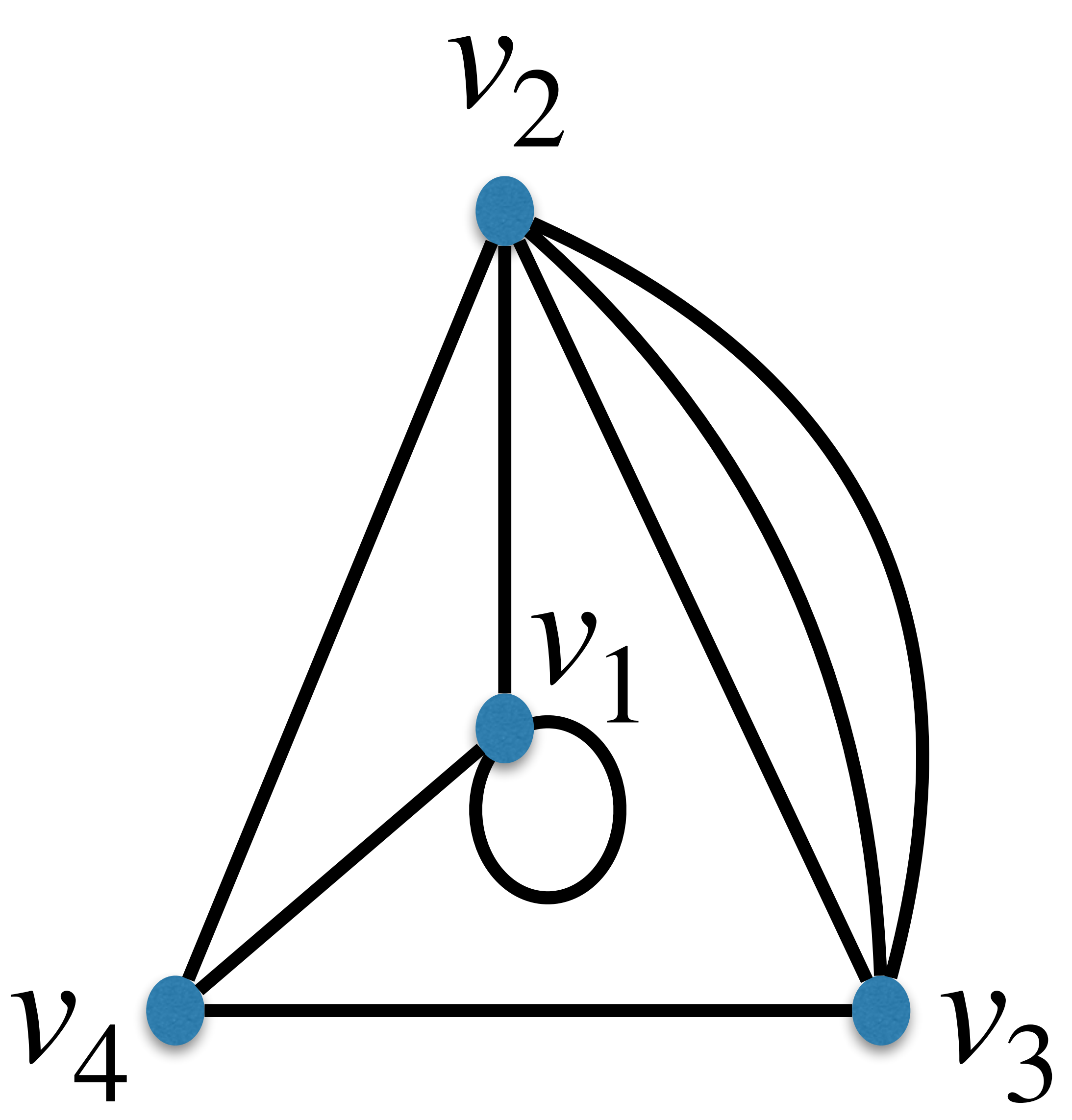
Undirected

Graph



Directed

Graph



Multigraph

Formal Definition (Simple Undirected Graph)

A (simple undirected) **graph** G is a tuple (V, E) where

- V is a finite set called the set of **vertices** (or **nodes**),
- E is a set called the set of **edges** such that
  every element of E is $\{u, v\}$ for distinct $u, v \in V$.

Example:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_5, v_6\}\}$$

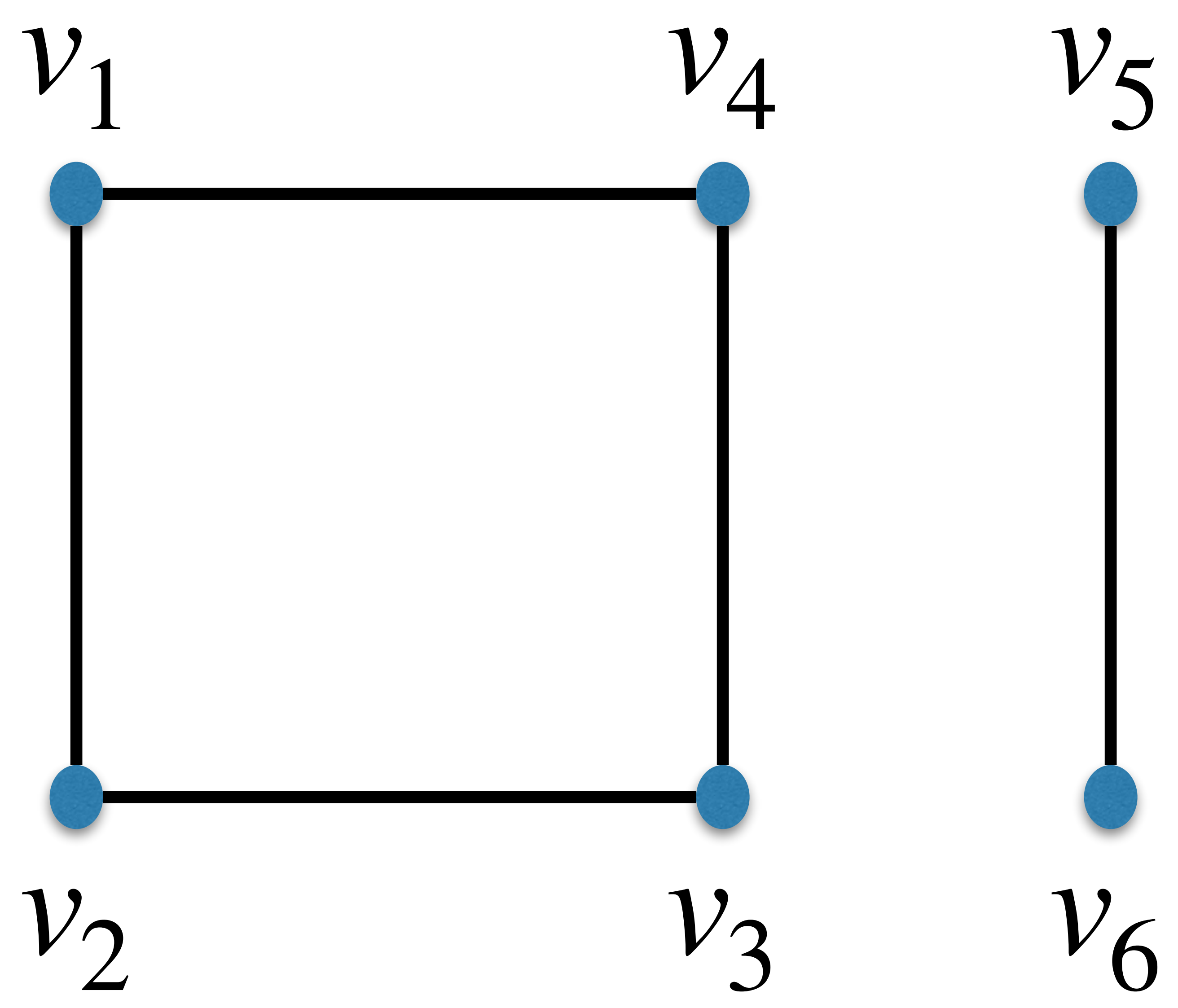
Formal Definition (Simple Undirected Graph)

Example:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_5, v_6\}\}$$

Graphs can be drawn:



Formal Definition (Simple Undirected Graph)

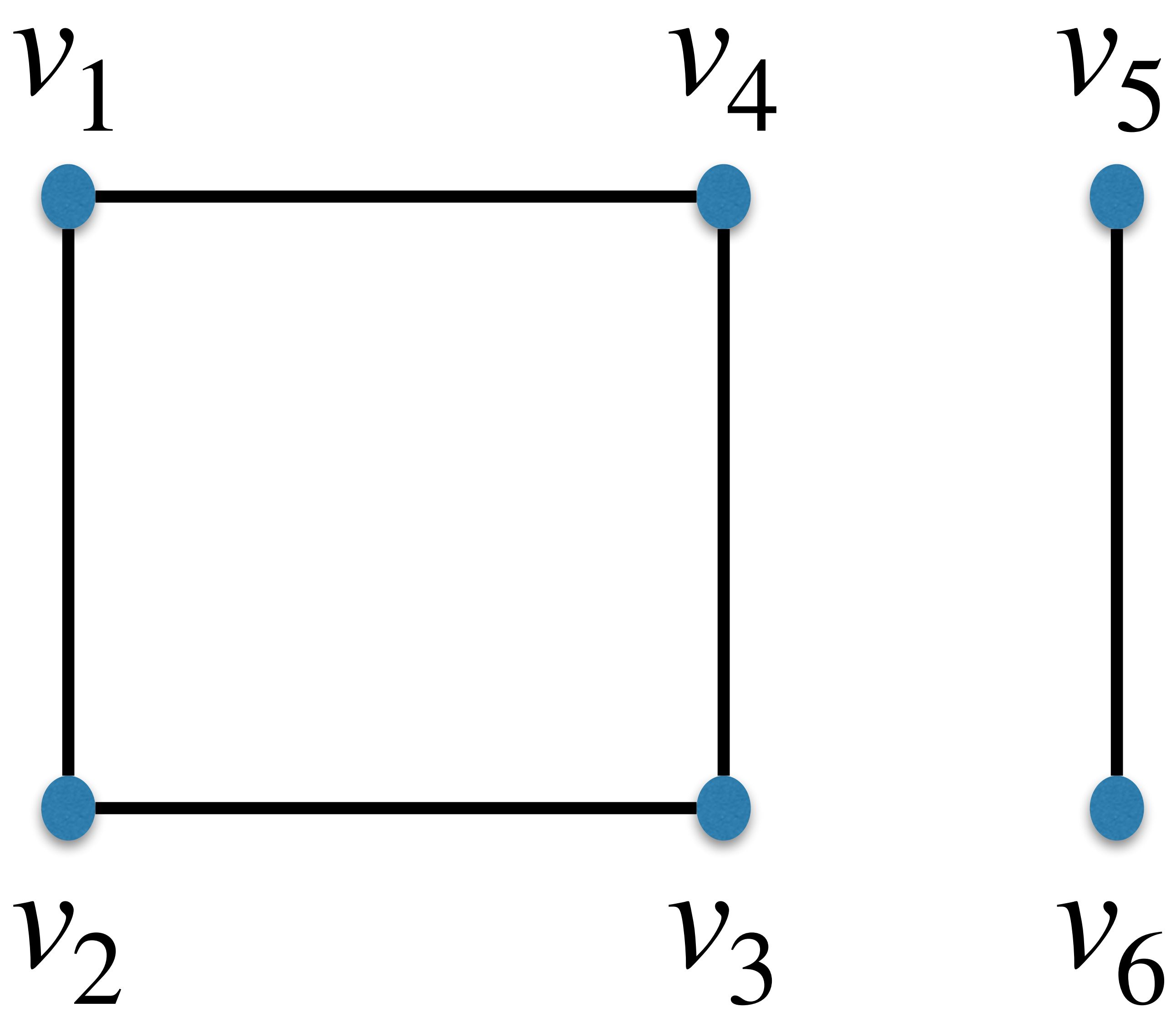
Example:

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_3, v_4\}, \{v_5, v_6\}\}$$

Adjacency Matrix Representation:

$$\begin{array}{c c}
 & \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} &
\begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}
\end{array}$$



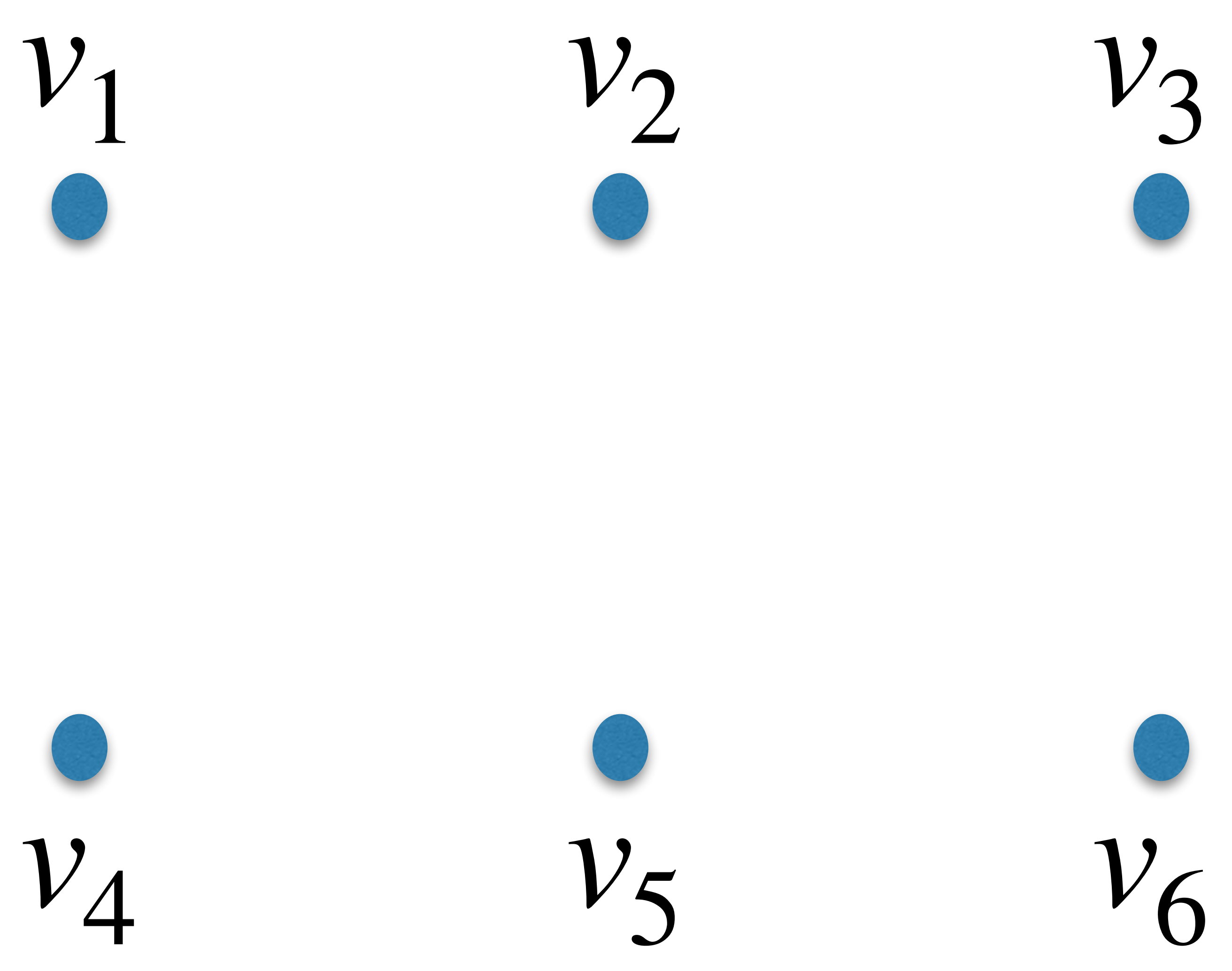
Almost always:

n = number of vertices in the graph, $|V|$

m = number of edges, $|E|$

Edge Cases

Is it possible that $E = \varnothing$?



6 "isolated" vertices.

Is it possible that $V = \varnothing$?

The Null Graph

IS THE NULL-GRAPH A POINTLESS CONCEPT?

Frank Harary
University of Michigan
and Oxford University

Ronald C. Read
University of Waterloo

ABSTRACT

The graph with no points and no lines is discussed critically. Arguments for and against its official admittance as a graph are presented. This is accompanied by an extensive survey of the literature. Paradoxical properties of the null-graph are noted. No conclusion is reached.

The Null Graph

Figure 1. The Null Graph

Other Definitions Related to Graphs

1st Challenge

Is it possible to have a party with 251 people in which everyone is friends with exactly 5 other people in the party?

Is it possible to have a graph with 251 vertices in which each vertex is adjacent to exactly 5 other vertices?

Terminology: Neighbor

Suppose $e = \{u, v\}$ is an edge.

We say:

u and v are **endpoints** of e

u and v are **adjacent**

u and v are **incident** on e

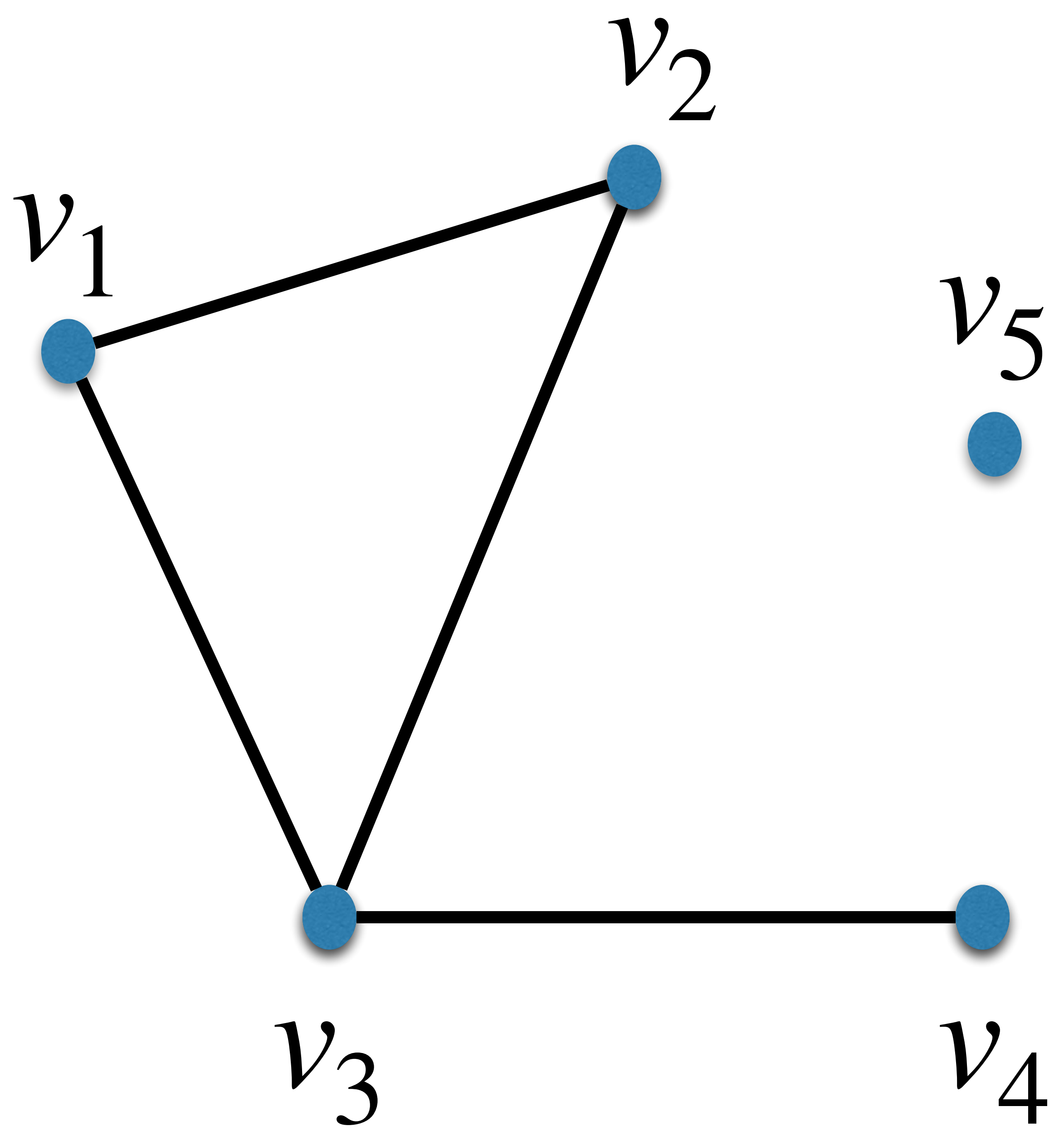
u is a **neighbor** of v

v is a **neighbor** of u

Terminology: Neighborhood

For $v \in V$, the **neighborhood** of v is defined as

$$N(v) = \{u \in V : \{v, u\} \in E\}.$$



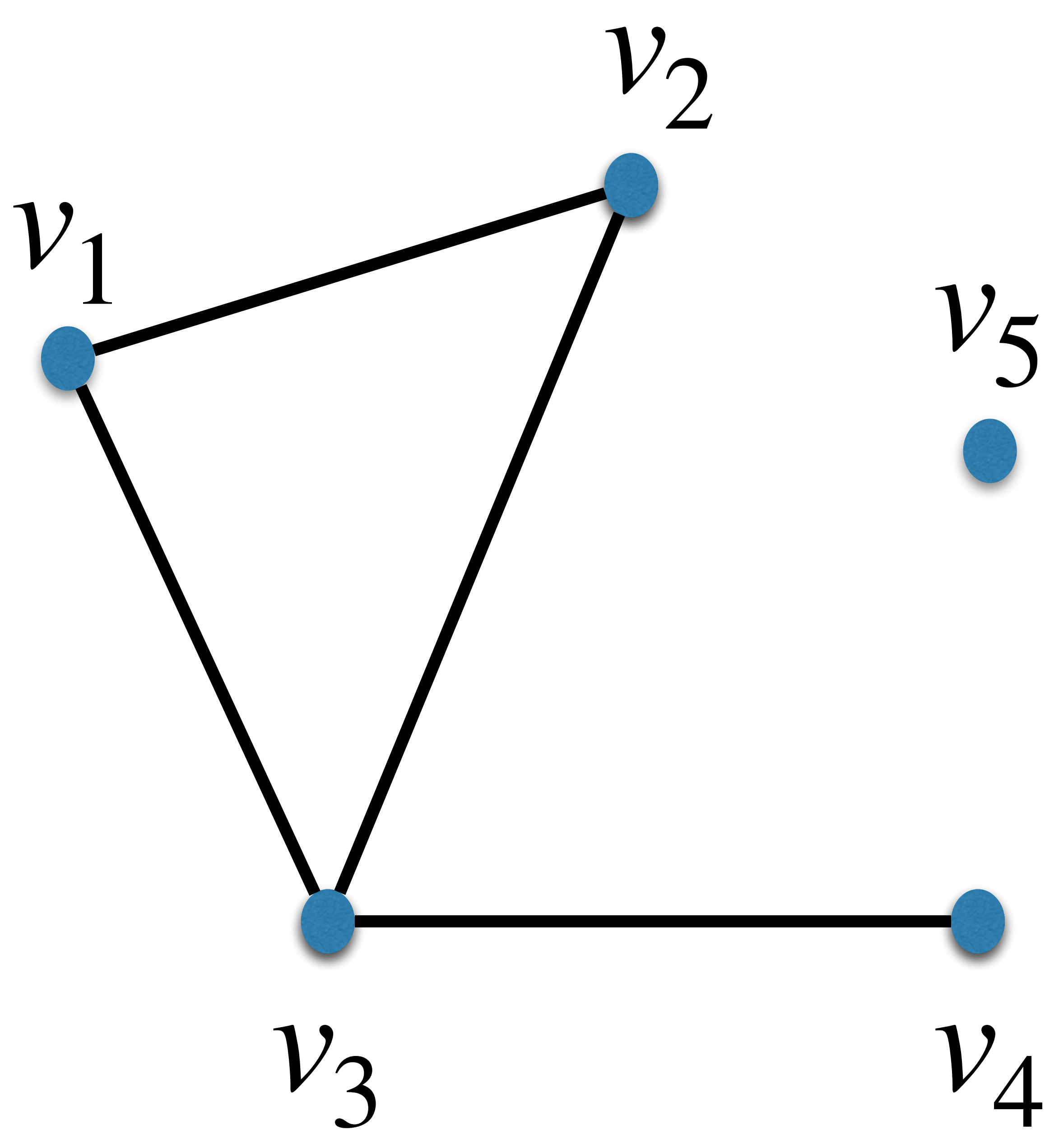
$$N(v_1) = \{v_2, v_3\}$$

$$N(v_3) = \{v_1, v_2, v_4\}$$

$$N(v_5) = \varnothing$$

Terminology: Degree

For $v \in V$, the **degree** of v is defined as $\deg(v) = |N(v)|$.



$\deg(v_1) = 2$

$\deg(v_3) = 3$

$\deg(v_5) = 0$

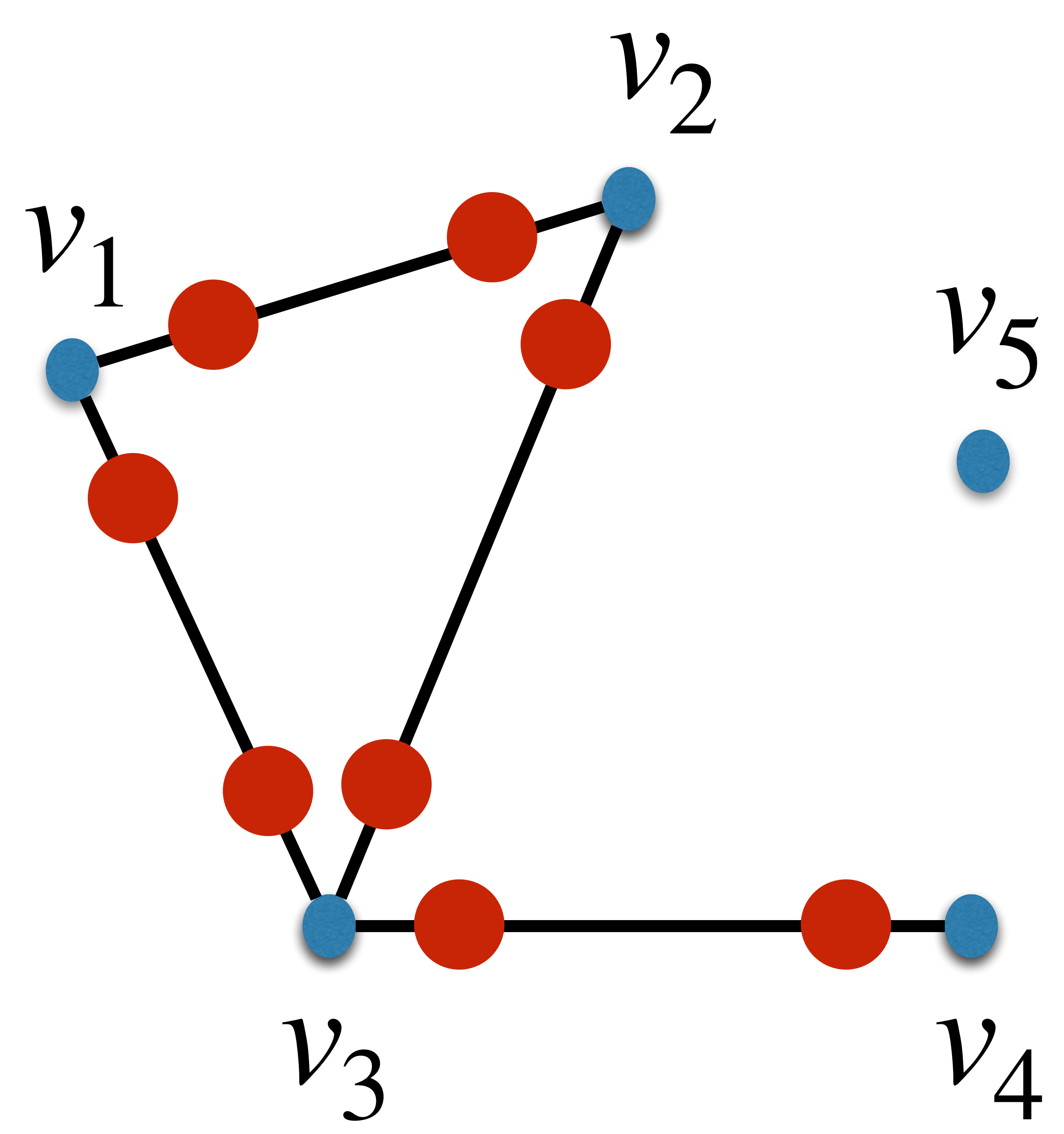
A graph is called **d-regular** if for all $v \in V$, $\deg(v) = d$.

Handshake Lemma

Lemma: Let $G = (V, E)$ be a graph. Then,

$$\sum_{v \in V} \deg(v) = 2m.$$

Proof:



Place tokens on edges:

- each vertex puts a token on each edge it is incident to.

Observations:

- vertex v puts $\deg(v)$ tokens.

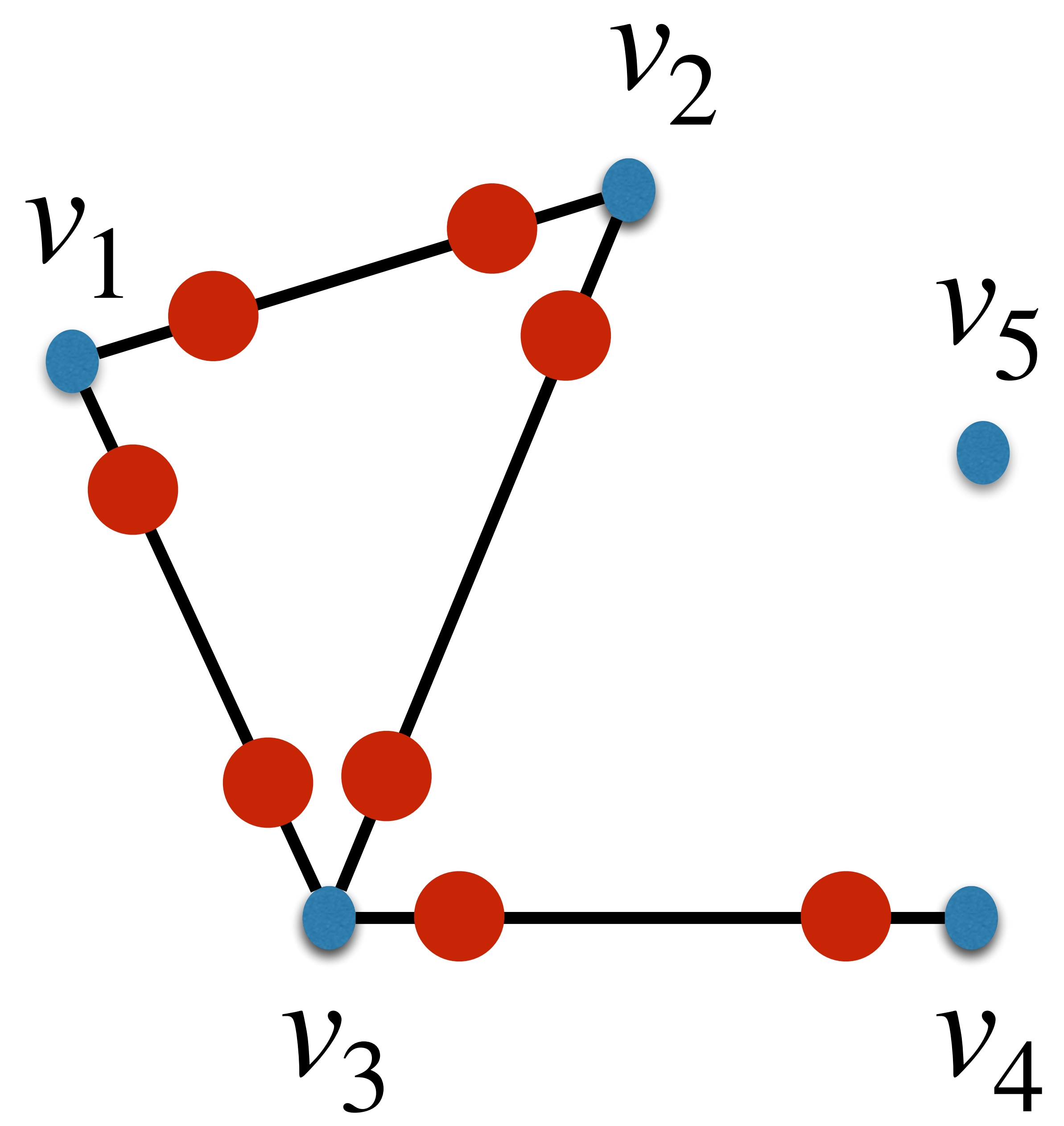
- each edge gets 2 tokens.

Handshake Lemma

Lemma: Let $G = (V, E)$ be a graph. Then,

$$\sum_{v \in V} \deg(v) = 2m.$$

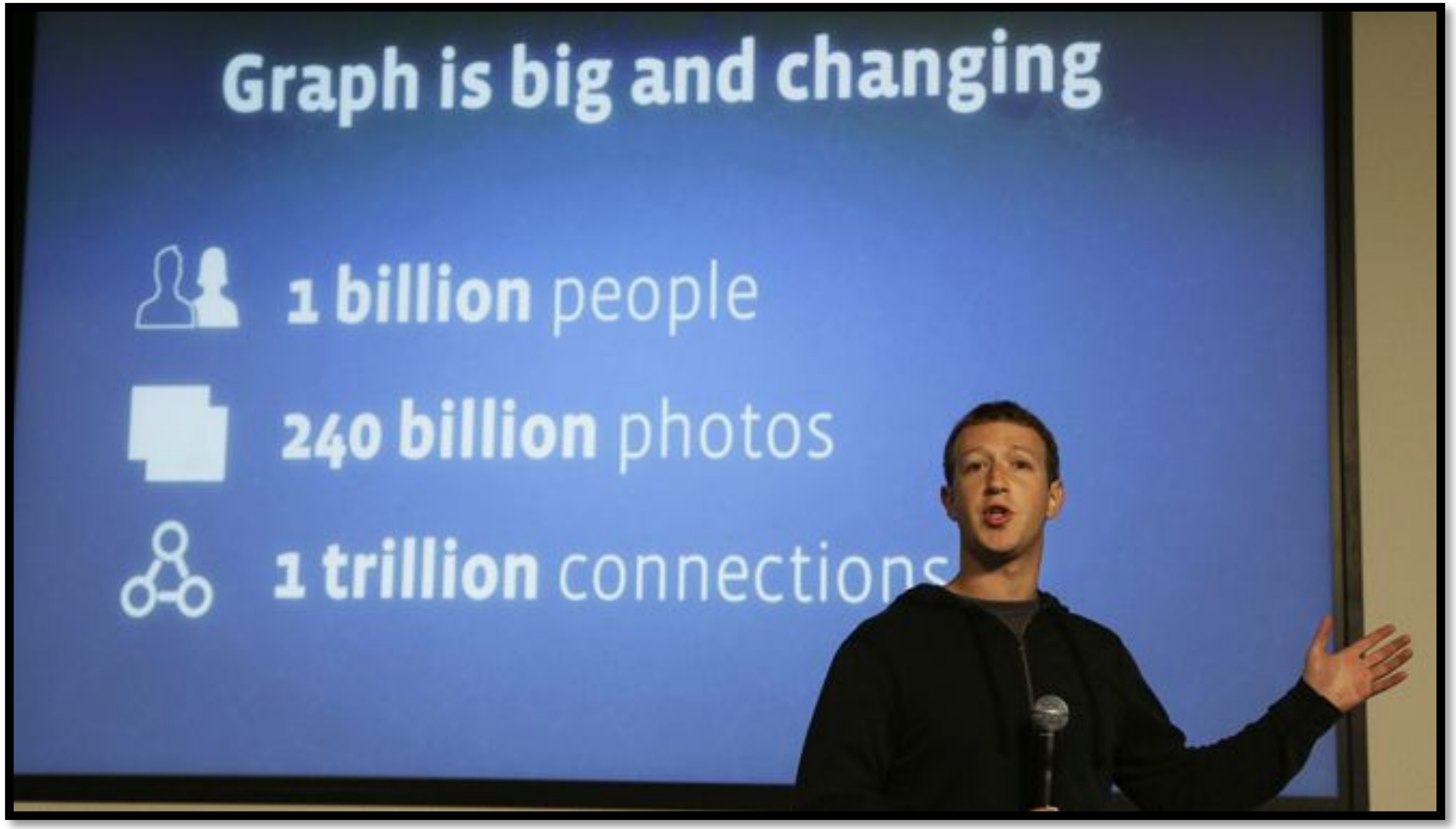
Proof (continued):



Count the total # tokens:

1st way: $\sum_{v \in V} \deg(v)$

2nd way: $2m$ $\blacksquare$



$$m = 1000000000000 \quad n = 1000000000$$

$$2m = 2000000000000 = \sum_{v \in V} \deg(v)$$

$\implies$ on average, people have 2000 friends.

poll.cs251.com

Is it possible to have a graph with 251 vertices in which each vertex is adjacent to exactly 5 other vertices?

2nd Challenge

We have n computers that we want to connect.

We can put a link between any two computers,
but the links are expensive.

What is the least number of links we can use?

What is the least number of edges needed to connect n vertices?

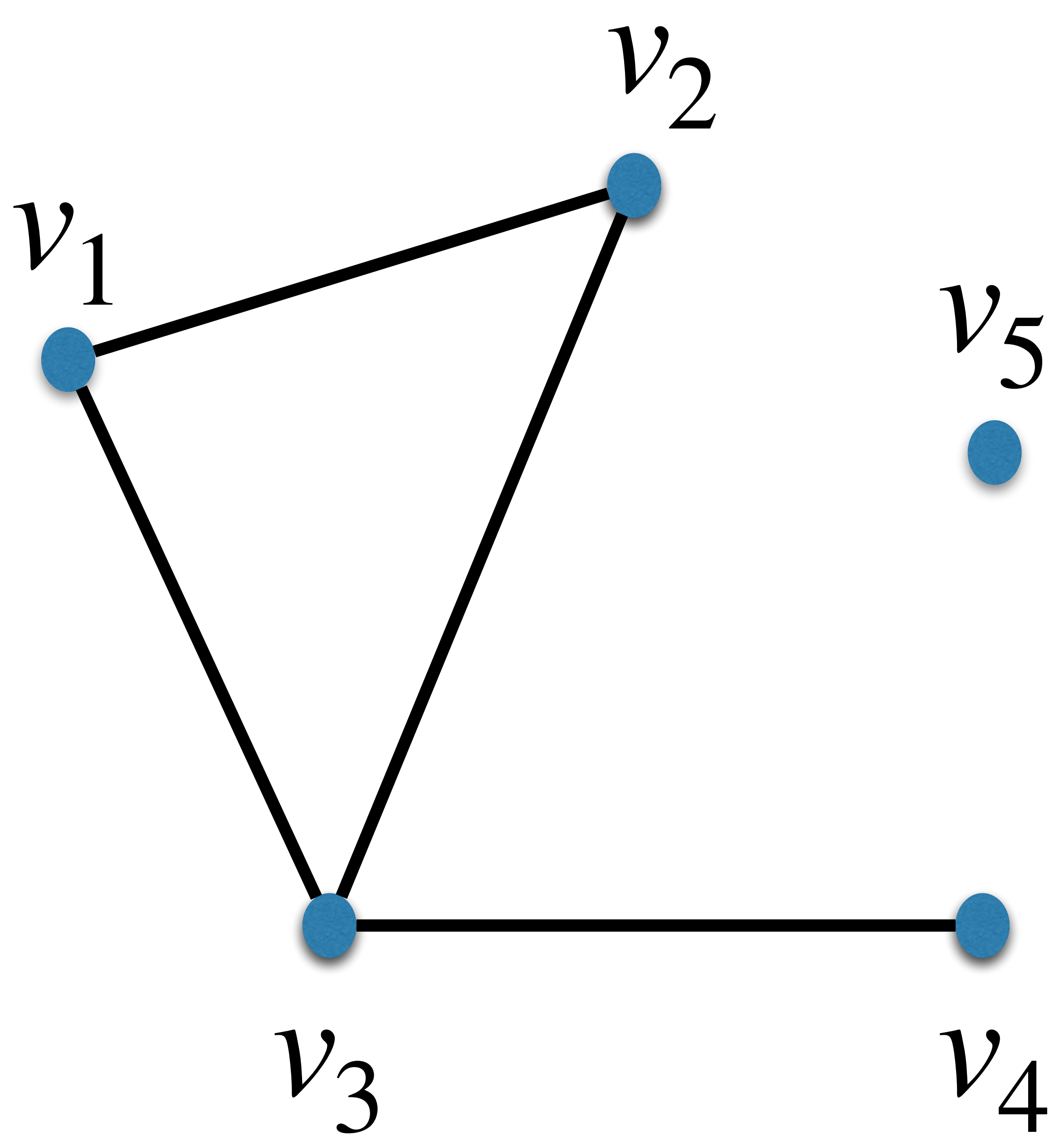
Terminology: Walks & Paths

A **walk** in a graph $G = (V, E)$ is a sequence of vertices

$$u_0, u_1, u_2, \ \ldots, u_k \qquad (k \geq 0)$$

such that $\{u_{i-1}, u_i\} \in E$ for all $i \in \{1,2,\ldots,k\}$.

This is a walk (of length k) from u_0 to u_k.



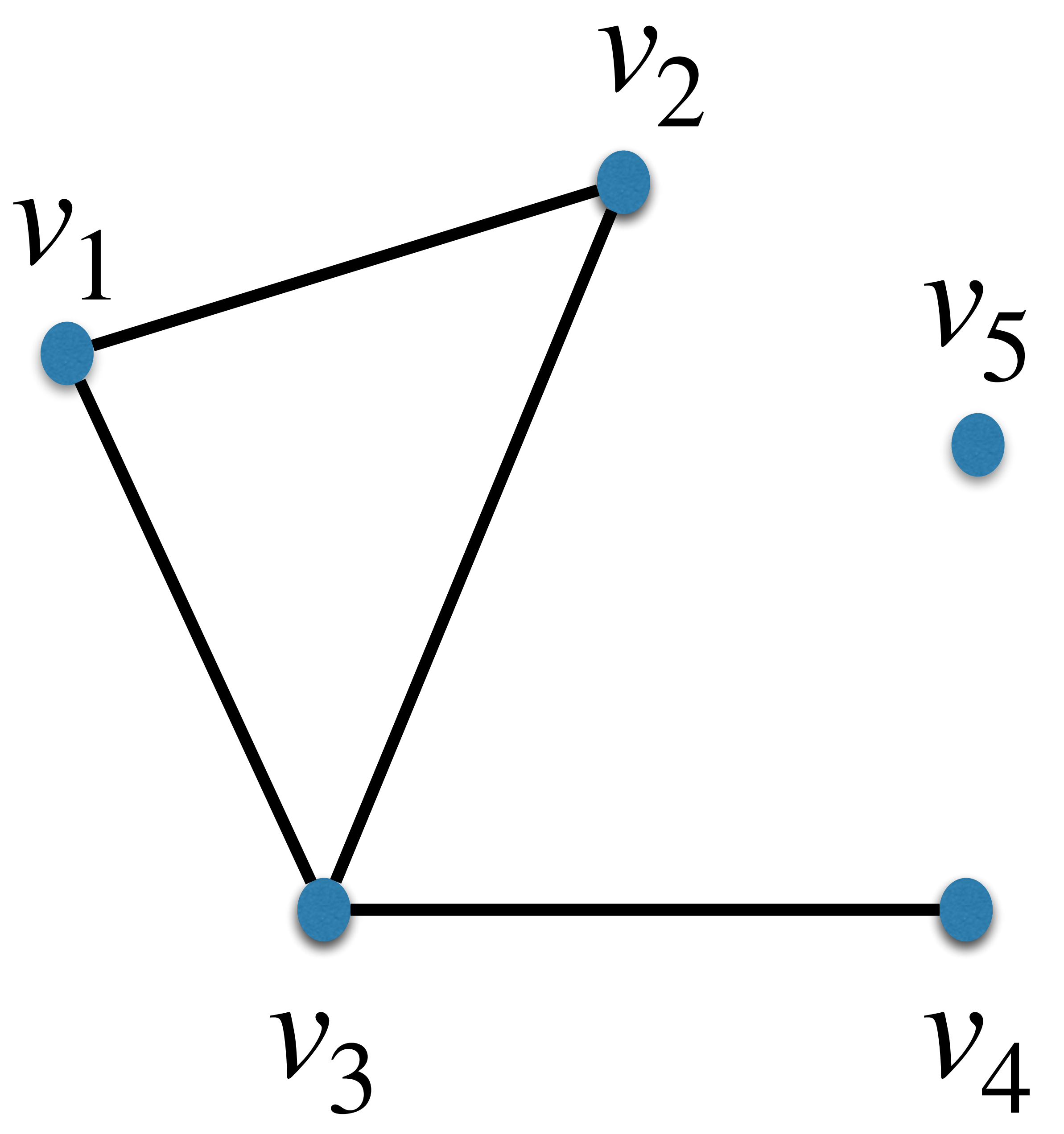
$(v_1, v_2, v_3, v_1, v_3, v_4)$

is a walk from v_1 to v_4
of length 5.

Terminology: Walks & Paths

A **path** in a graph $G = (V, E)$ is a walk with <u>no repeated vertices</u>.

<u>Fact:</u> There is a path from v to v' iff there is a walk from v to v'.



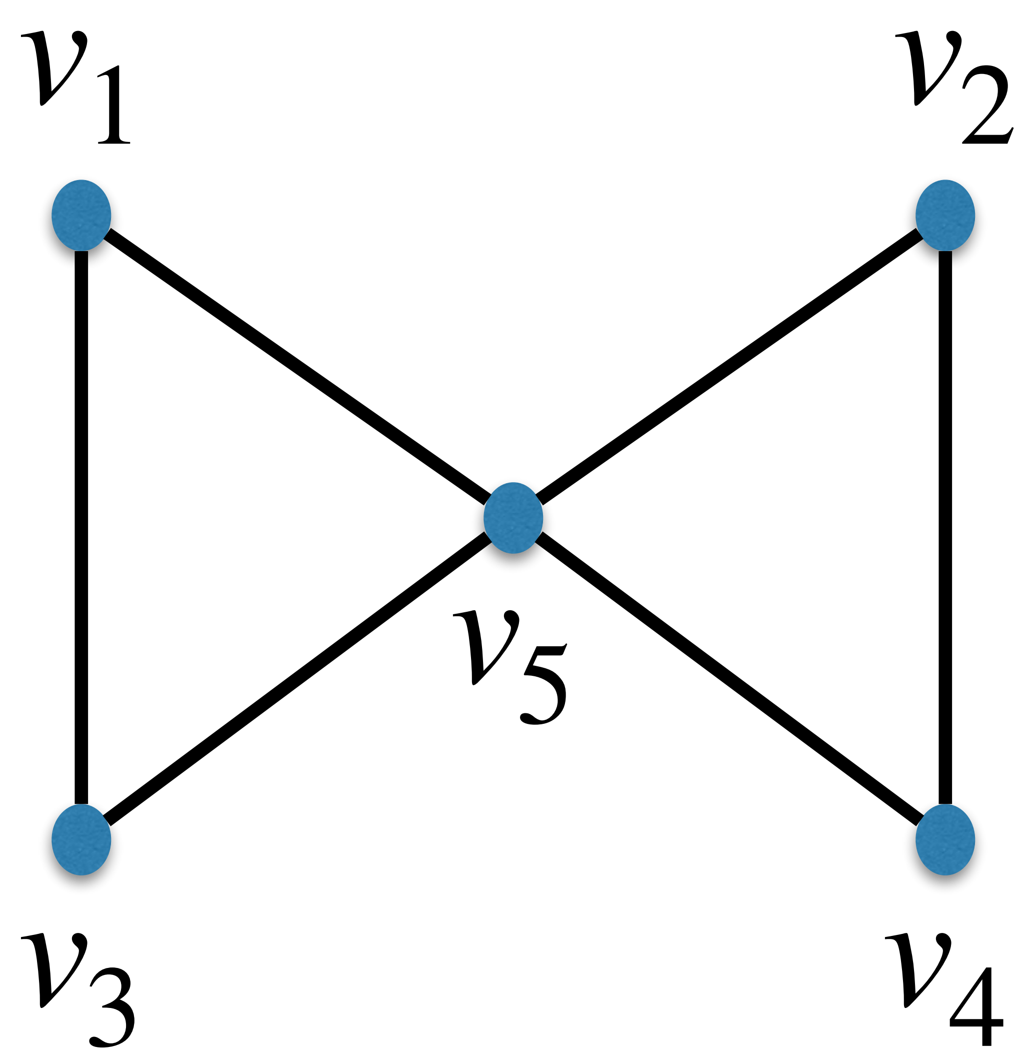
$(v_1, v_2, v_3, v_1, v_3, v_4)$

"shortcut" repeated vertices

(v_1, v_2, v_3, v_4)

Terminology: Circuits & Cycles

A **circuit** in a graph $G = (V, E)$ is a walk from u to u (for some vertex u).



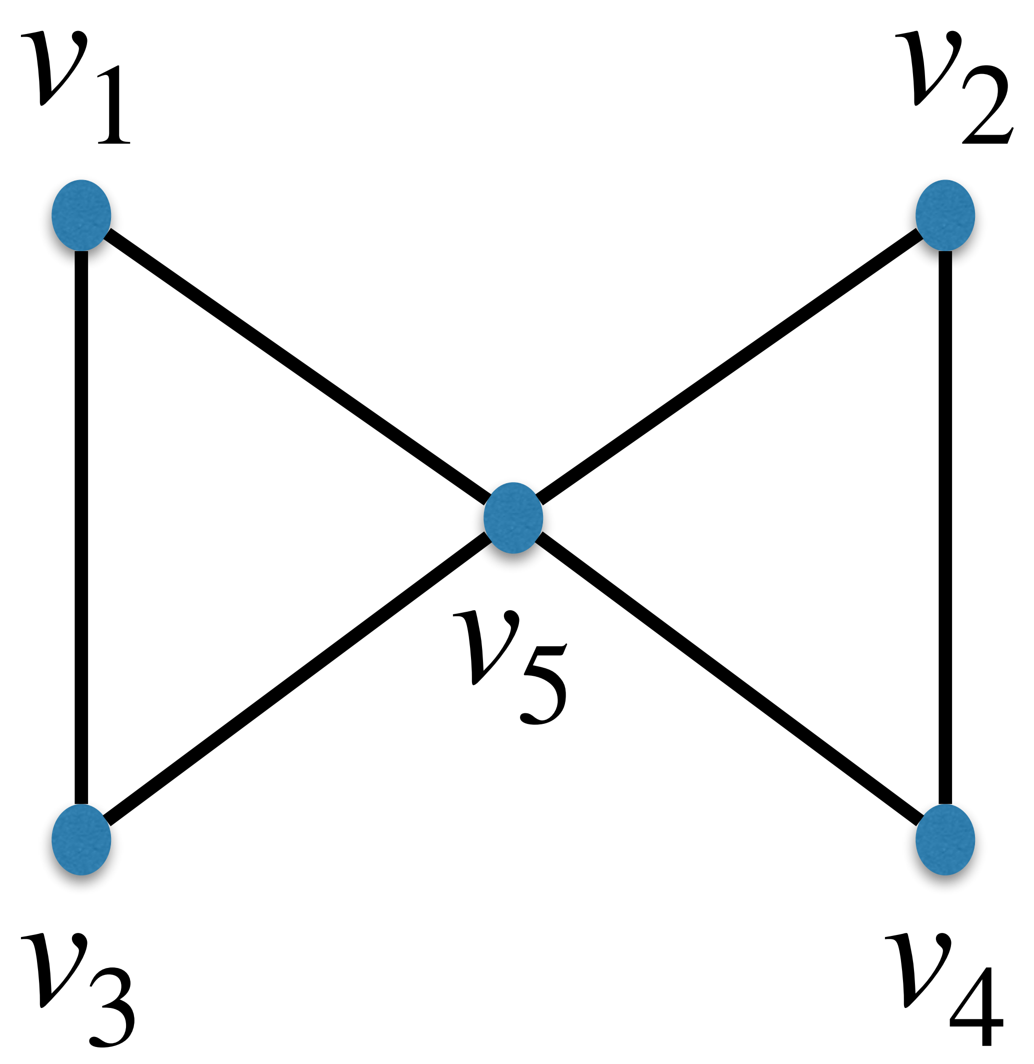
$(v_1, v_5, v_2, v_4, v_5, v_3, v_1)$

is a circuit

Terminology: Circuits & Cycles

A **cycle** in a graph $G = (V, E)$ is a circuit with <u>no repeated vertices</u>. (length ≥ 3)

except the start & end



(v_1, v_3, v_5, v_1) is a cycle.

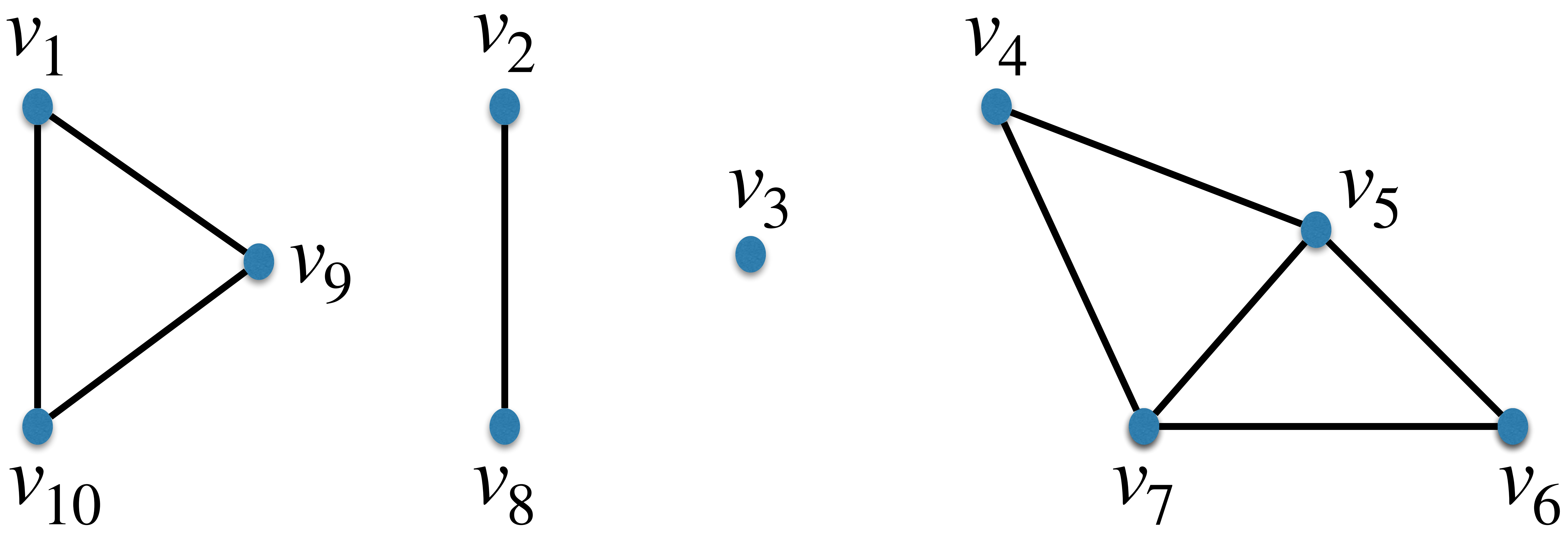
(v_1, v_5, v_3, v_1) is considered the same cycle.

(v_5, v_1, v_3, v_5) is considered the same cycle.

A graph with no cycles is called **acyclic**.

Terminology: Connected Graph

A graph is **connected** if there is a path between any two vertices in the graph.



This 10-vertex graph is **<u>not</u>** connected.

It has 4 **connected components**:

$\{v_1, v_9, v_{10}\}$ $\{v_2, v_8\}$ $\{v_3\}$ $\{v_4, v_5, v_6, v_7\}$

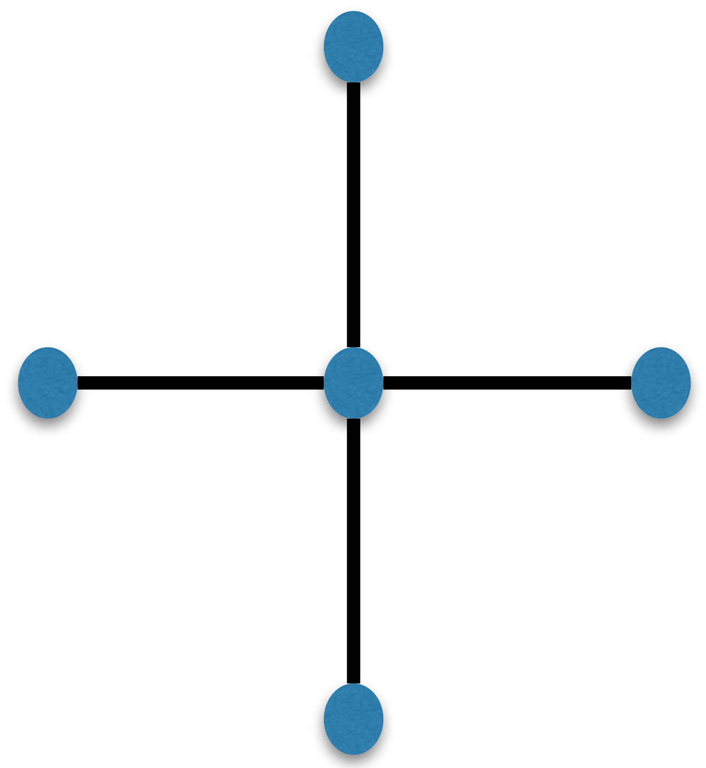
A graph is connected **iff** it has 1 connected component.

Back to the Challenge

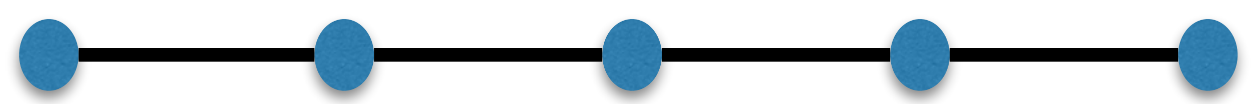
What is the least number of edges needed to connect **n** vertices?

n-1 edges are always sufficient

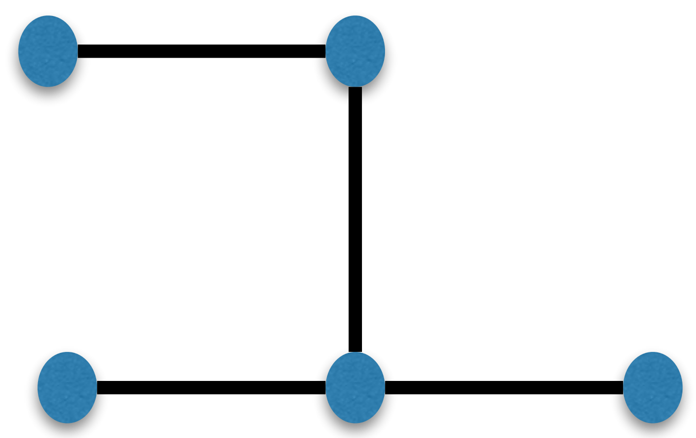
"star graph"



"path graph"



"something else"



n-1 edges always necessary?

Connected $\implies m \geq n-1$

Theorem: Let $G = (V, E)$ be a connected graph.

Then $m \geq n - 1$.

Furthermore: $m = n - 1 \iff G$ is acyclic.

Proof:

Imagine the following process:

- remove all the edges of G.
- add them back one by one (in an arbitrary order).

n isolated vertices → G

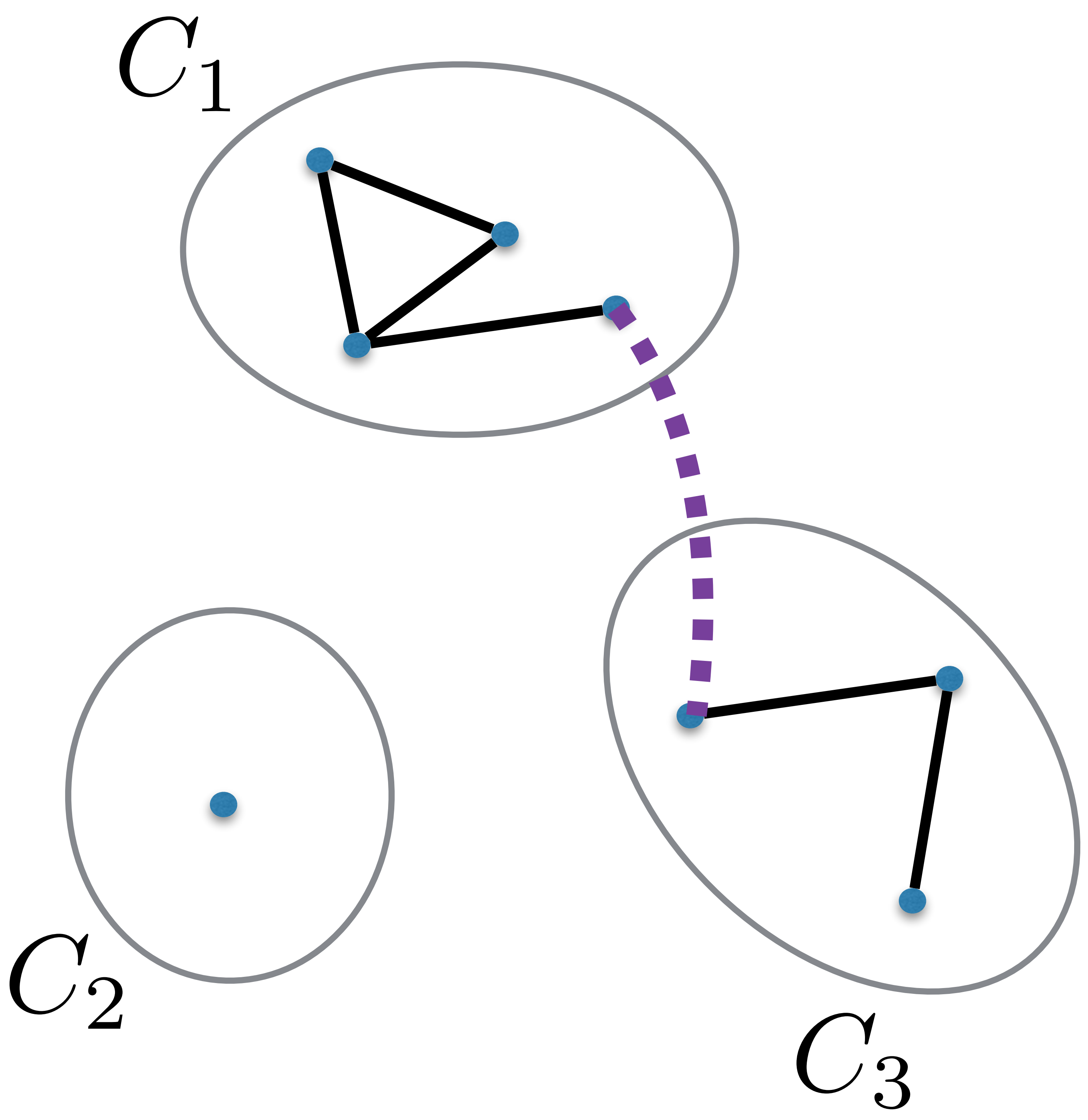
n CCs → **1** CC

CC = connected component

Connected $\implies m \geq n - 1$

Proof (continued):

Consider a step of adding an edge back.



2 possibilities:

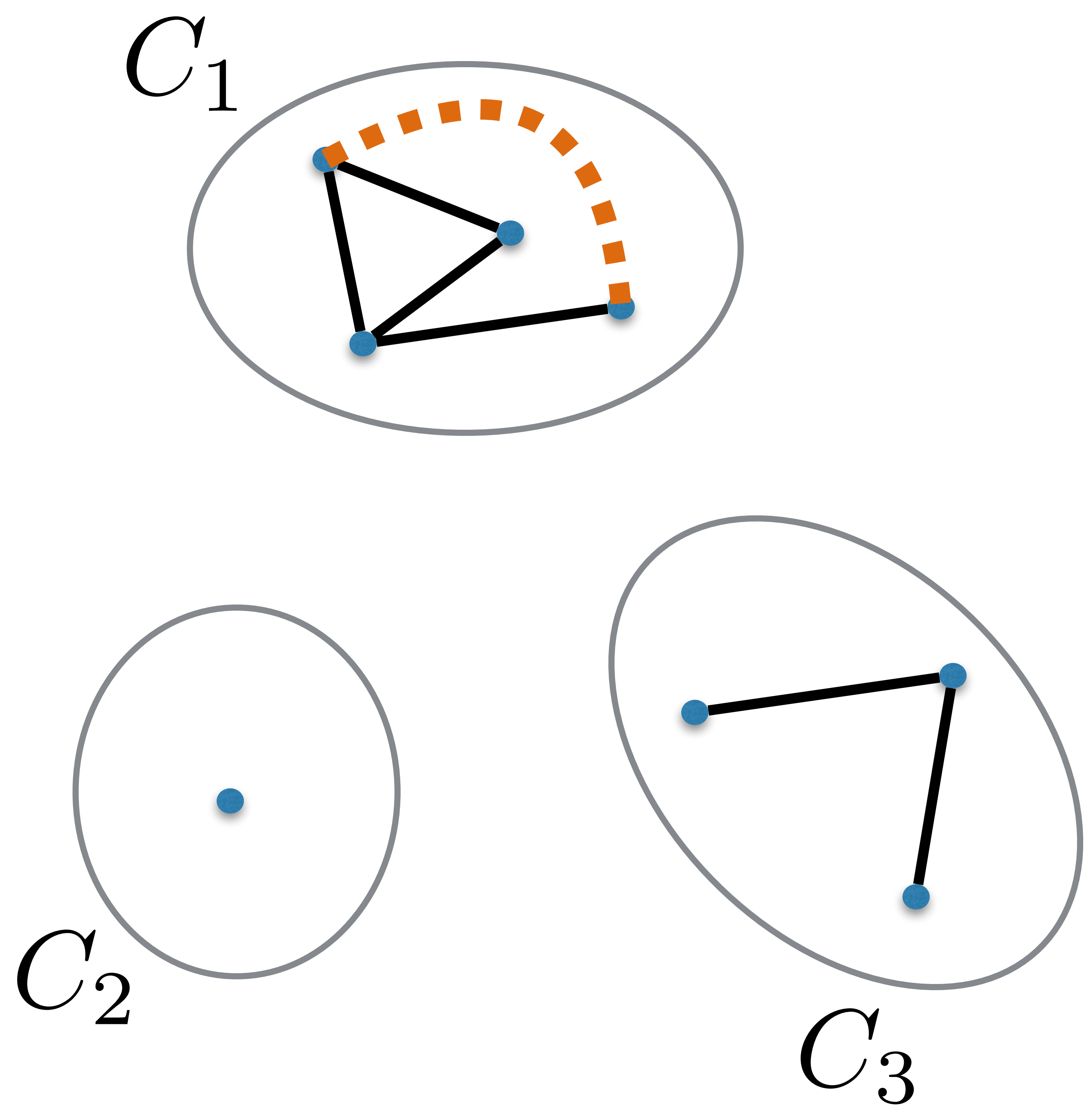
(i) connector edge

- connects 2 CCs.
- # CCs goes down by 1.
- cannot create a new cycle.

Connected $\implies m \geq n - 1$

Proof (continued):

Consider a step of adding an edge back.



2 possibilities:

(i) connector edge

- connects 2 CCs.
- # CCs goes down by 1.
- cannot create a new cycle.

(ii) cycle-creator edge

- an edge within a CC.
- # CCs stays the same.
- creates a new cycle.

Connected $\implies m \geq n-1$

<u>Proof (continued):</u>

Consider a step of adding an edge back.

<u>2 possibilities:</u>

(i) connector edge # CCs goes down by 1.

(ii) cycle-creator edge # CCs stays the same.

n CCs $\xrightarrow{\text{(i)}}$ **n-1** CCs $\xrightarrow{\text{(i)}}$ **n-2** CCs $\xrightarrow{\text{(i)}}$ **...** $\xrightarrow{\text{(i)}}$ **1** CC

So we must add at least $n-1$ edges. (at least $n-1$ type **(i)** edges)

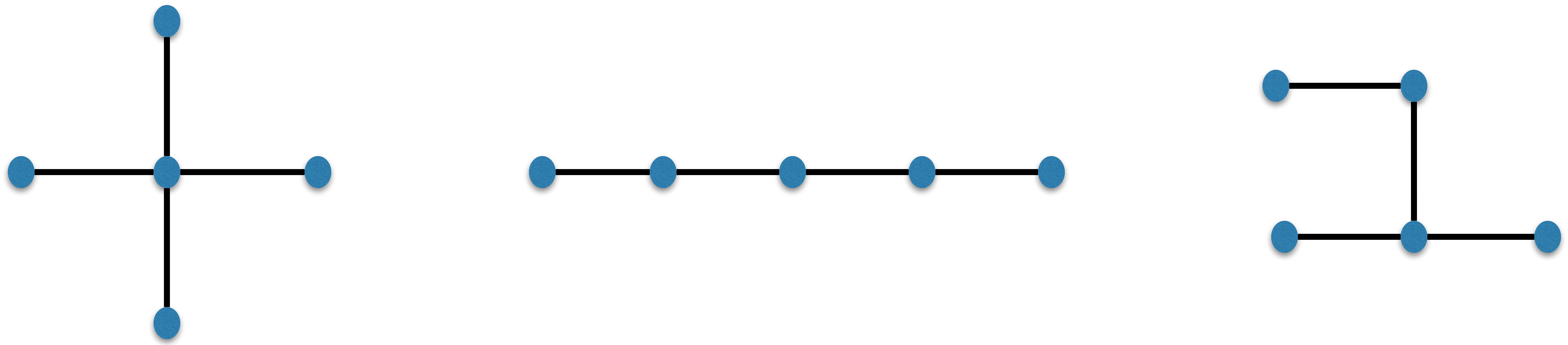
i.e. we must have $m \geq n-1$.

If $m = n-1$: all type **(i)** edges $\implies$ no cycles.

If $m > n-1$: at least one type **(ii)** edge $\implies$ a cycle. $\blacksquare$

Trees

Some examples with 5 vertices



An n-vertex **tree** is any graph with at least 2 of the following 3 properties:

(i) connected

(ii) $m = n - 1$

(iii) acyclic

Exercise:
If a graph has two of the properties, it automatically has the third too.

Basic Graph Algorithms

Graph Search Algorithms

- **Depth-First Search (DFS)**
- **Breadth-First Search (BFS)**

Minimum Spanning Tree (MST) Algorithm

Minimum Spanning Tree (MST) Algorithm

Motivating Question

Year: 1926

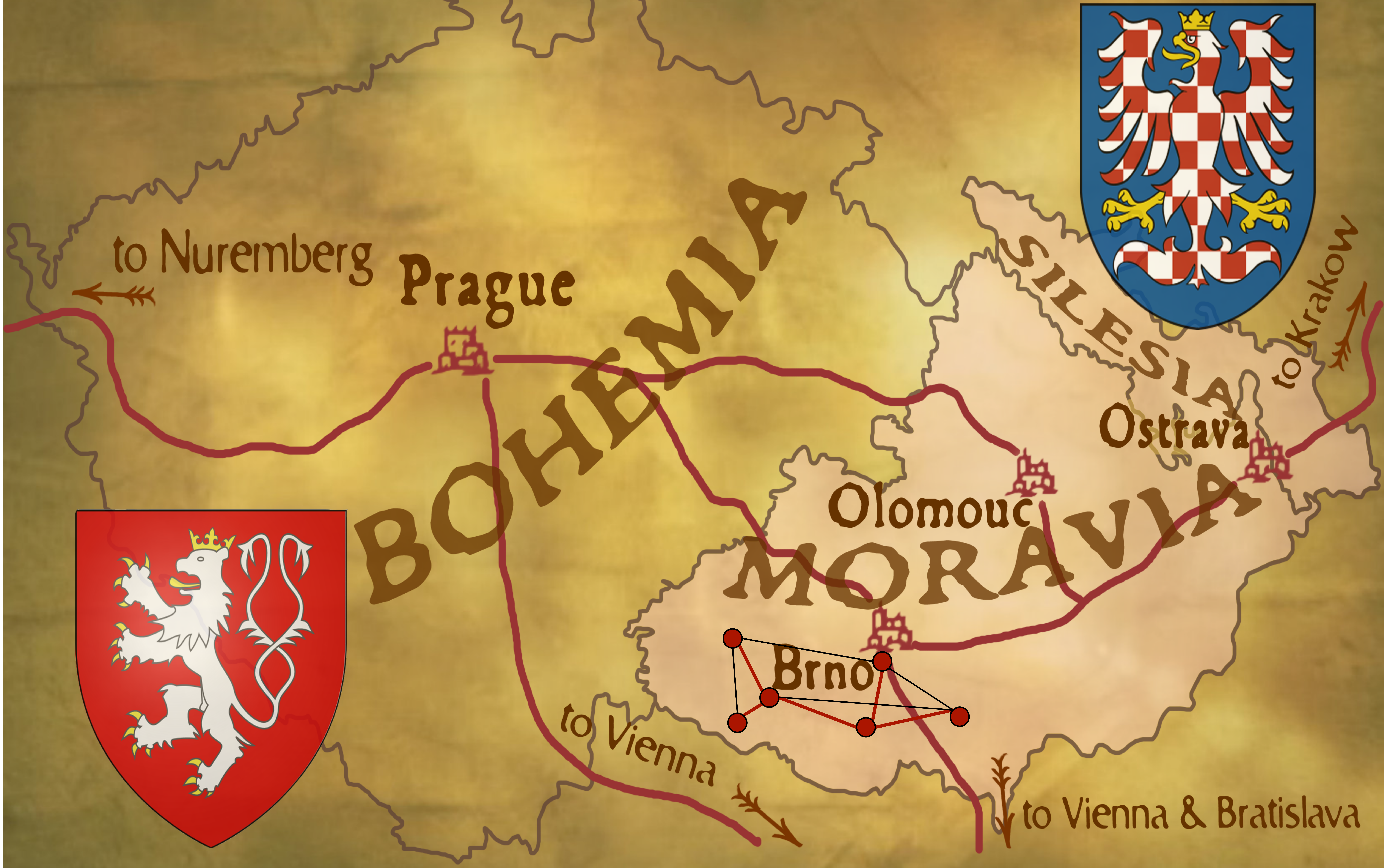
Place: Brno, Moravia

Our Hero: Otakar Boruvka

Boruvka's pal Jindrich Saxel was working for West Moravian Power Plant company.

Saxel asked:

What is the least cost way to electrify southwest Moravia?

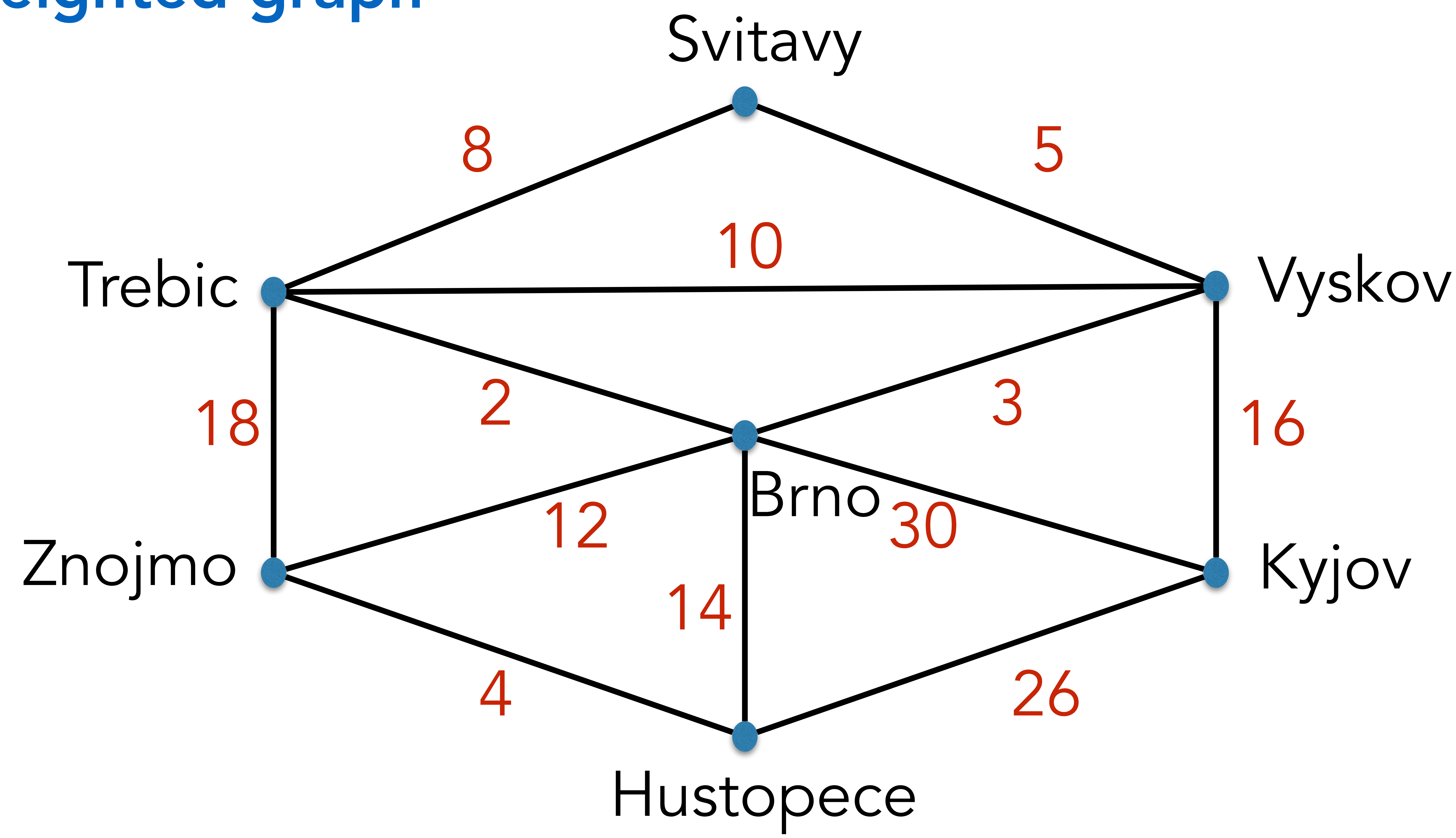

to Nuremberg
Prague
BOHEMIA
SILESIA
to Krakow
Ostrava
Olomouc
MORAVIA
Brno
to Vienna
to Vienna & Bratislava

CS Life Lesson

If your problem has a graph, great!!!
If not, try to make it have a graph.

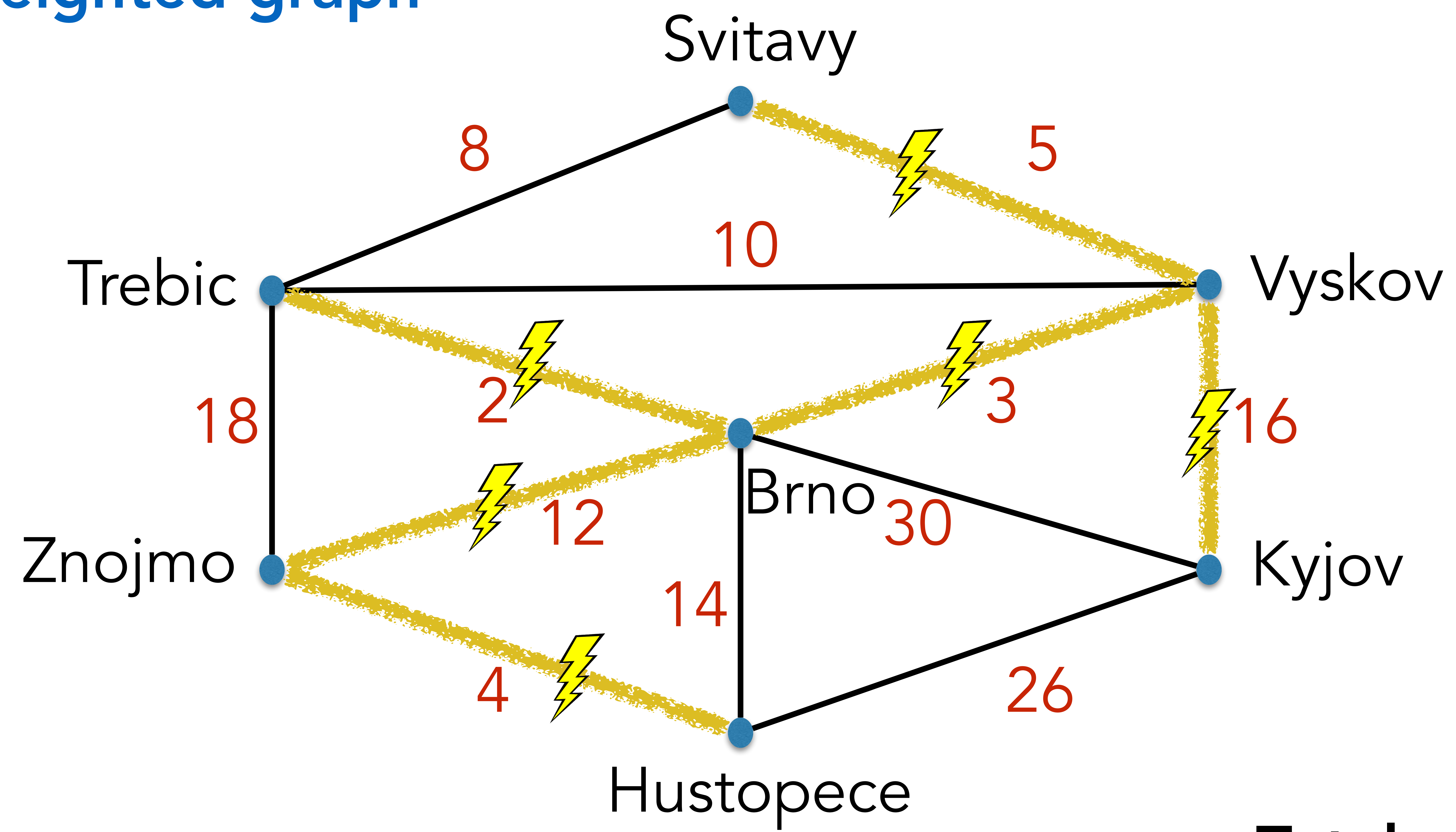
Graph Representation of Problem

weighted graph



Graph Representation of Problem

weighted graph



Total weight/cost: 42

Minimum Spanning Tree (MST) Problem

Input: A connected graph $G = (V, E)$, and a cost function $c : E \to \mathbb{R}^+$.

Output: An MST. I.e., subset of edges with <u>minimum total cost</u> such that all vertices are connected.

<u>**Observation:**</u> The output must be a **tree** (i.e. connected, acyclic).

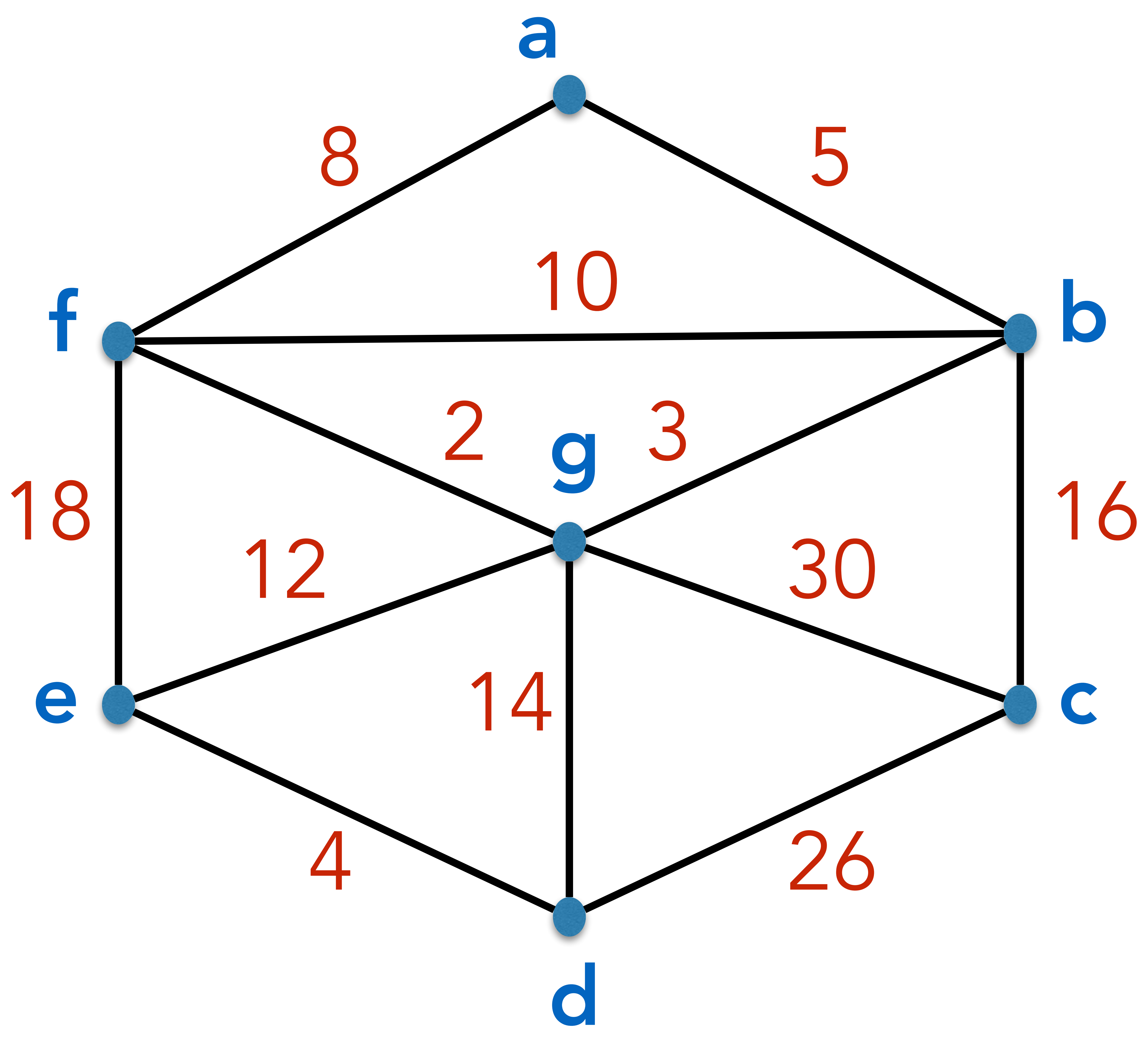
<u>**Convenient Assumption:**</u> Edges have distinct costs.

<u>**Exercise:**</u>

In this case, the MST is unique.

"Whether the distance from Brno to Breclav is 50km or 50km and 1cm is a matter of conjecture."

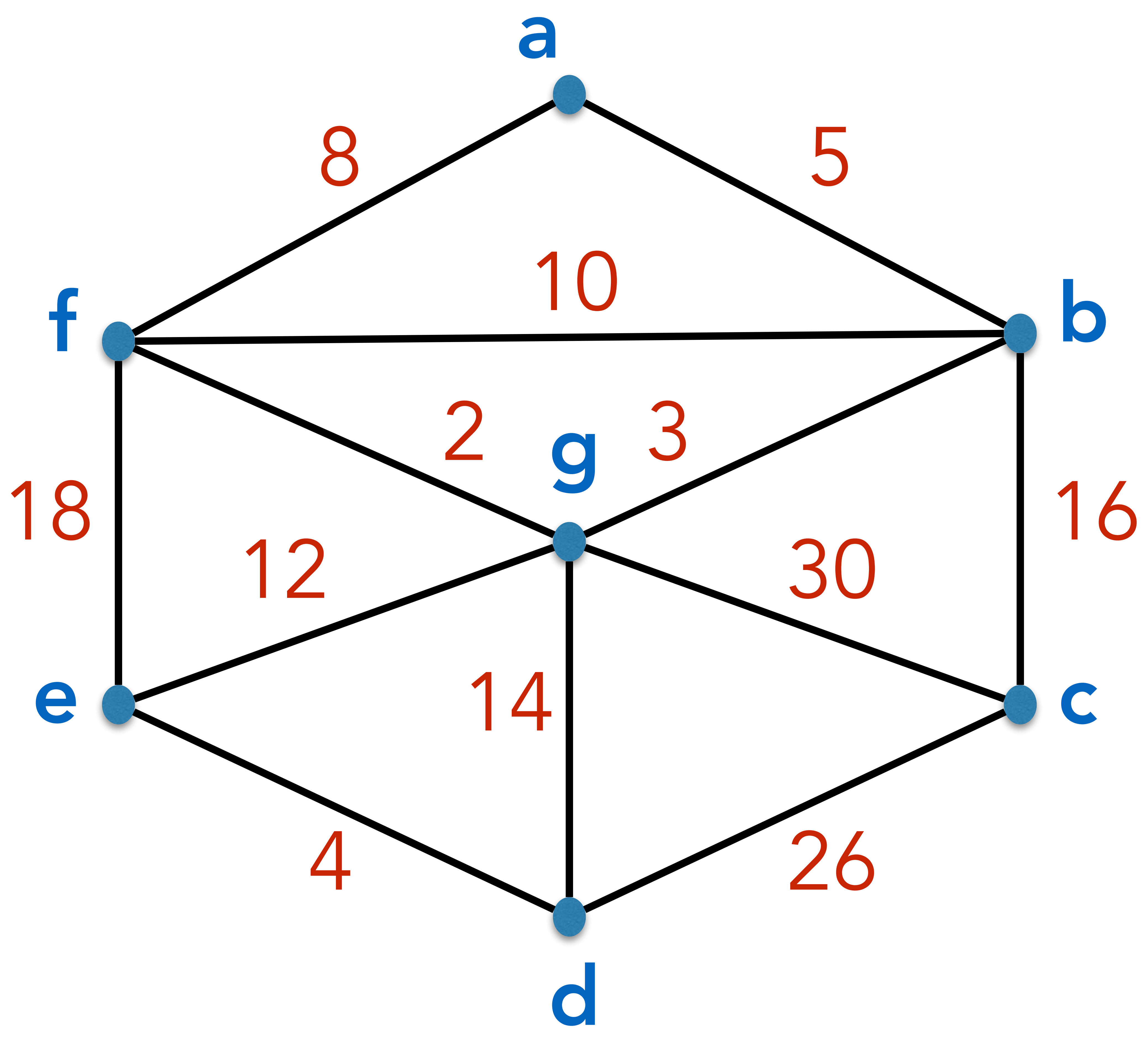
Jarník-Prim Algorithm



S = vertices connected so far

T = edges in the solution so far

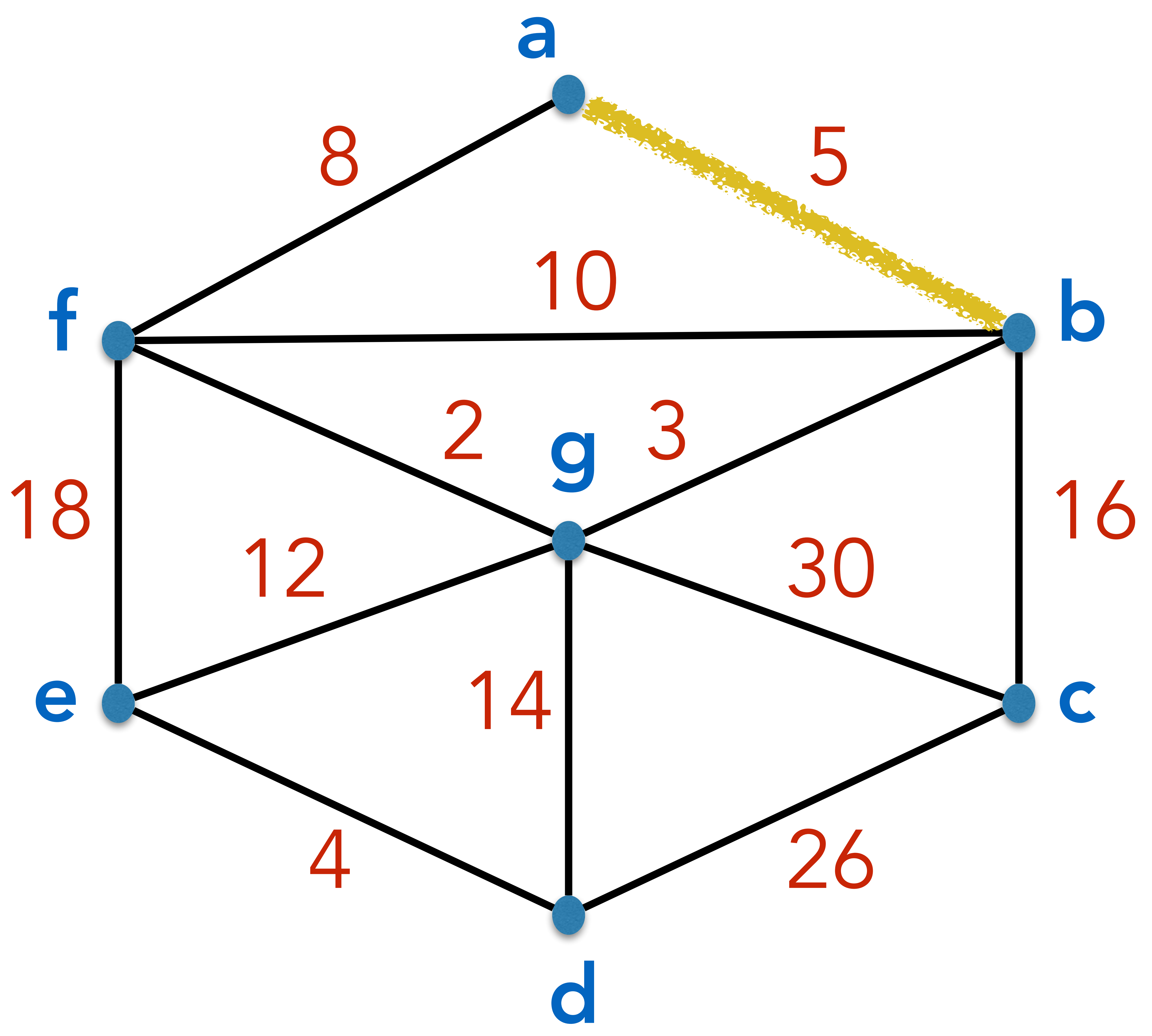
Jarník-Prim Algorithm



S = {a} (start with an arbitrary node)

T = { }

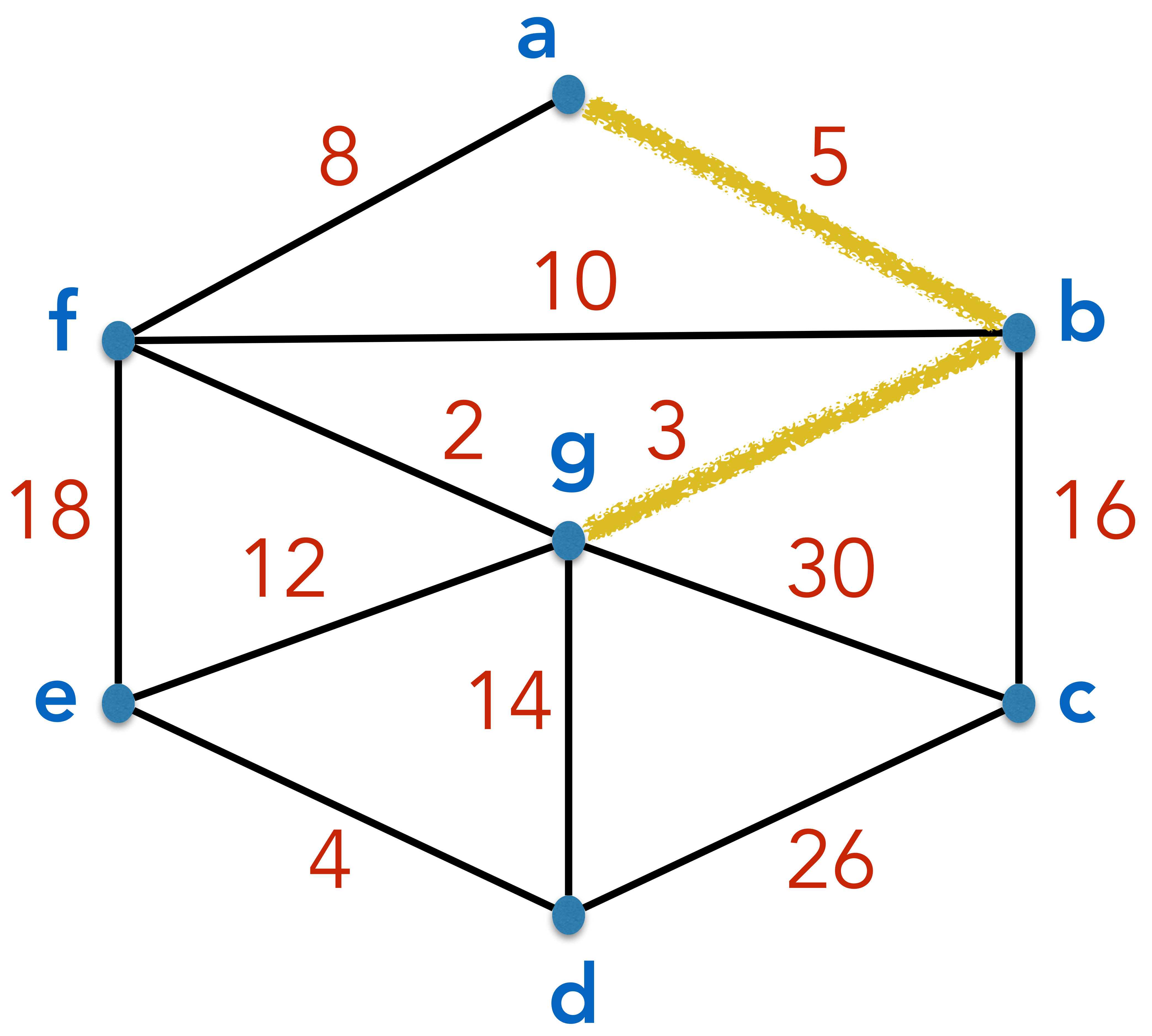
Jarník-Prim Algorithm



S = {a, b}

T = {{a, b}}

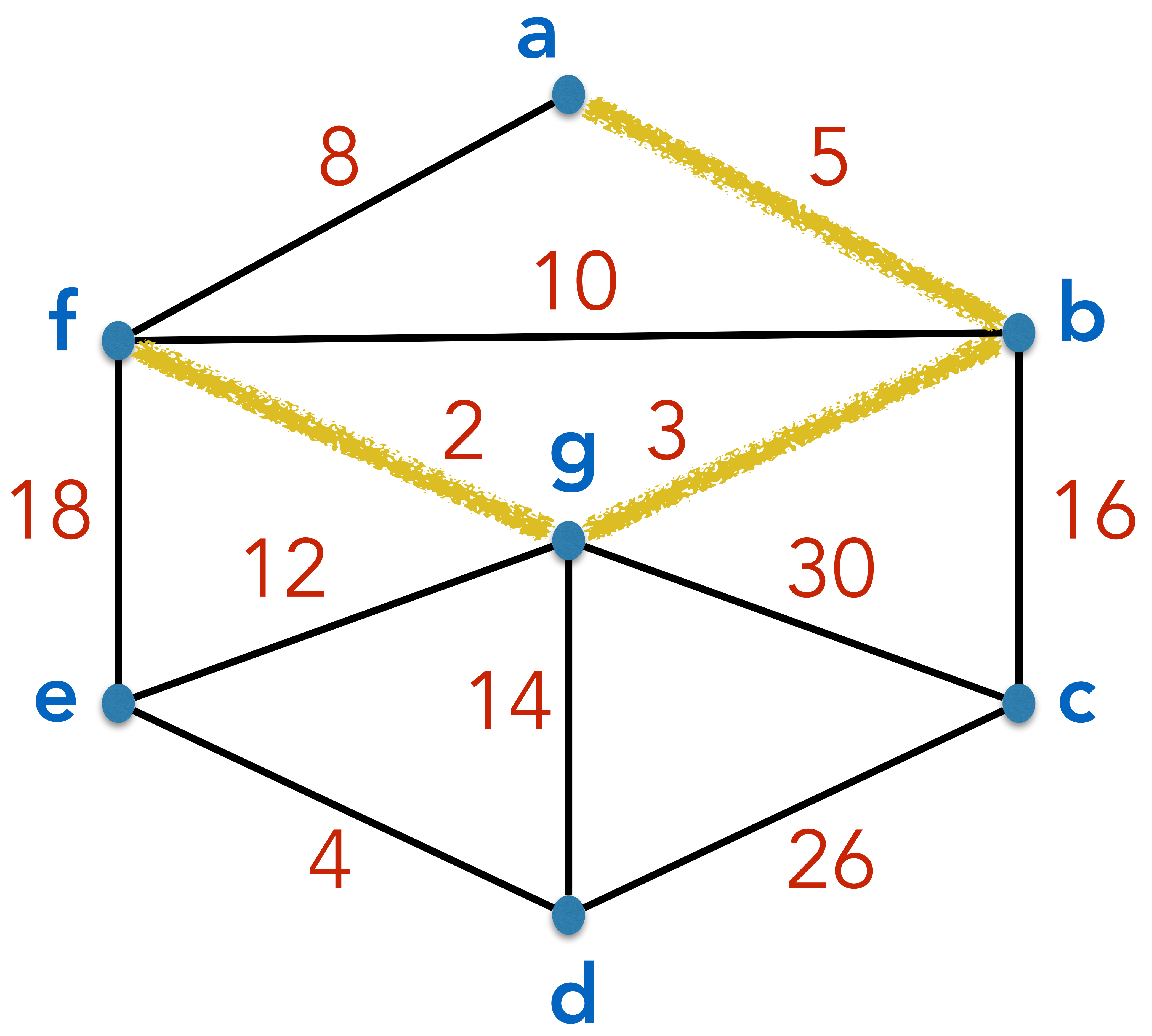
Jarník-Prim Algorithm



S = {a, b, g}

T = {{a, b}, {b, g}}

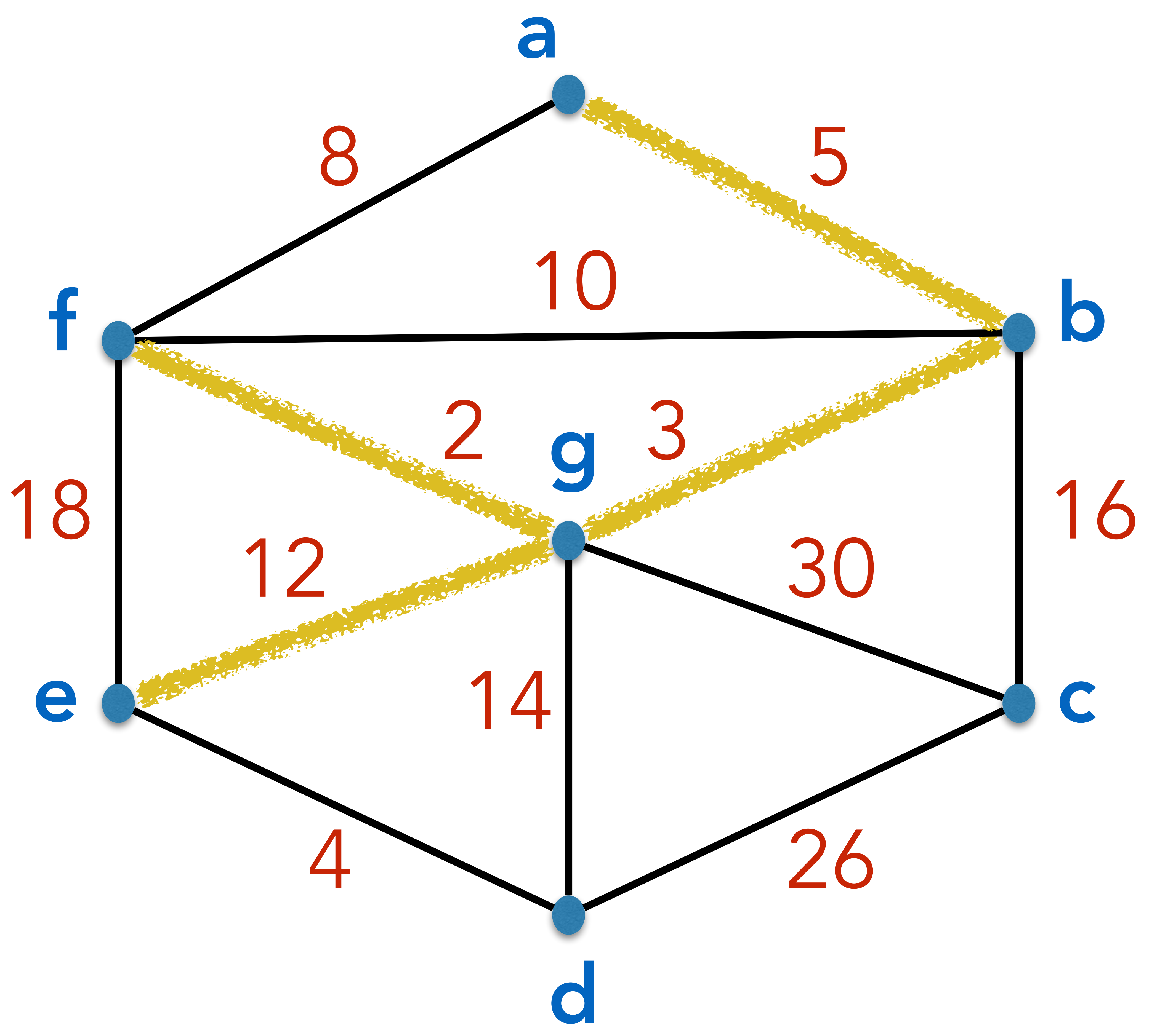
Jarník-Prim Algorithm



S = {a, b, g, f}

T = {{a, b}, {b, g}, {g, f}}

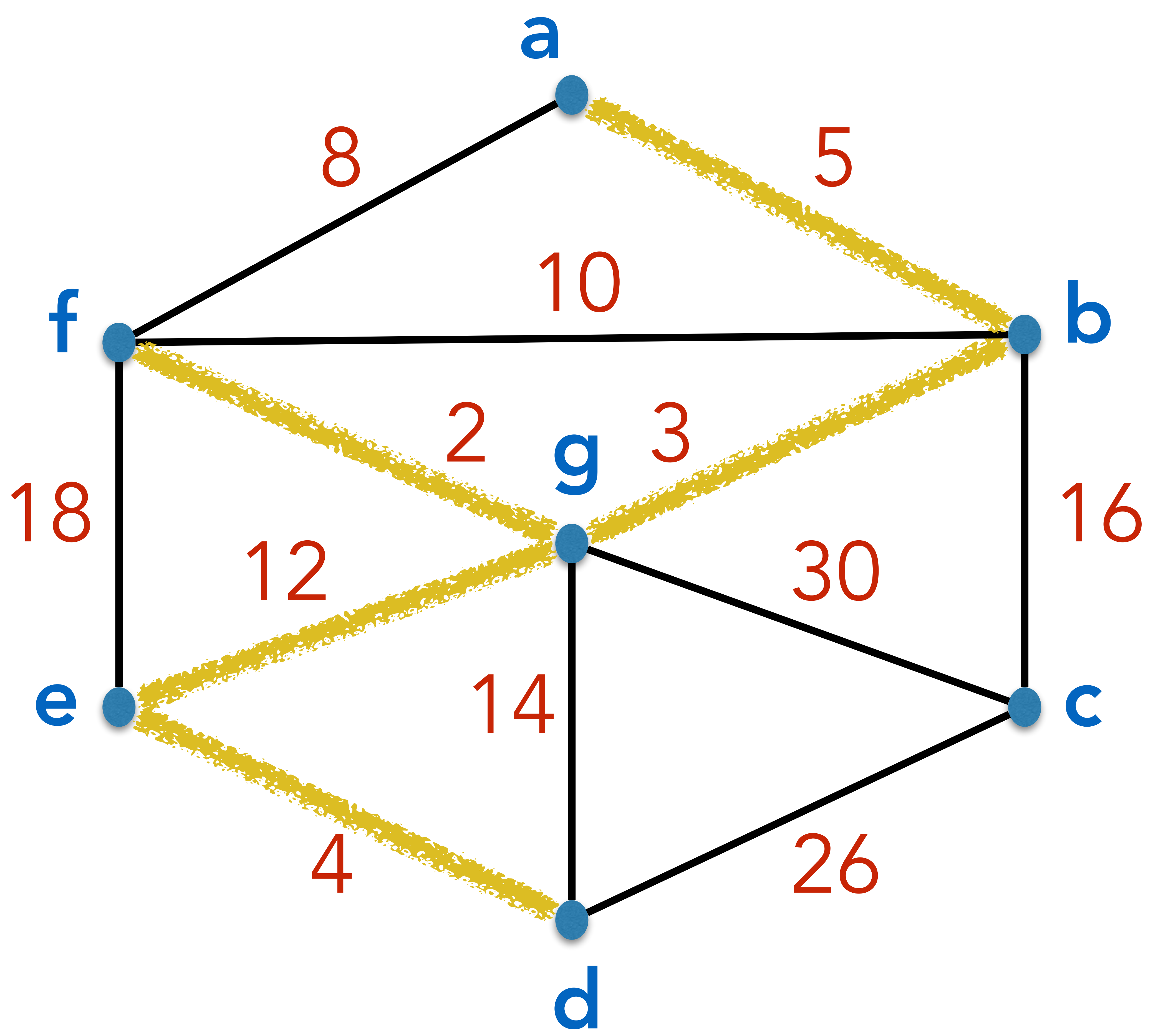
Jarník-Prim Algorithm



S = {a, b, g, f, e}

T = {{a, b}, {b, g}, {g, f}, {g, e}}

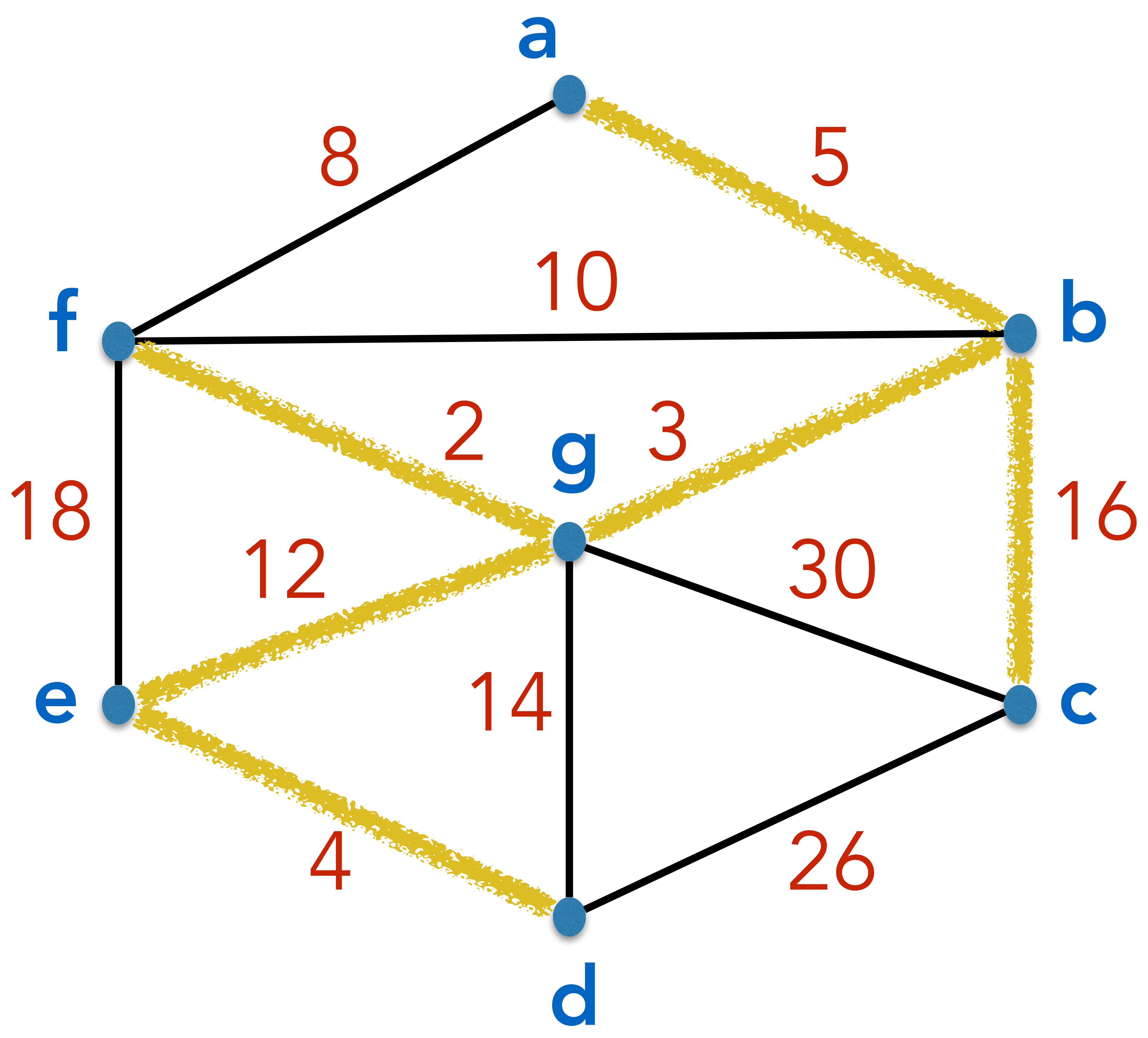
Jarník-Prim Algorithm



S = {a, b, g, f, e, d}

T = {{a, b}, {b, g}, {g, f}, {g, e}, {e, d}}

Jarník-Prim Algorithm



Total cost: 42

S = {a, b, g, f, e, d, c}

T = {{a, b}, {b, g}, {g, f}, {g, e}, {e, d}, {b, c}}

Jarník-Prim Algorithm

On input a weighted & connected graph **G** = (**V**, **E**):

S = {**w**} (for an arbitrary **w** in **V**)

T = ∅

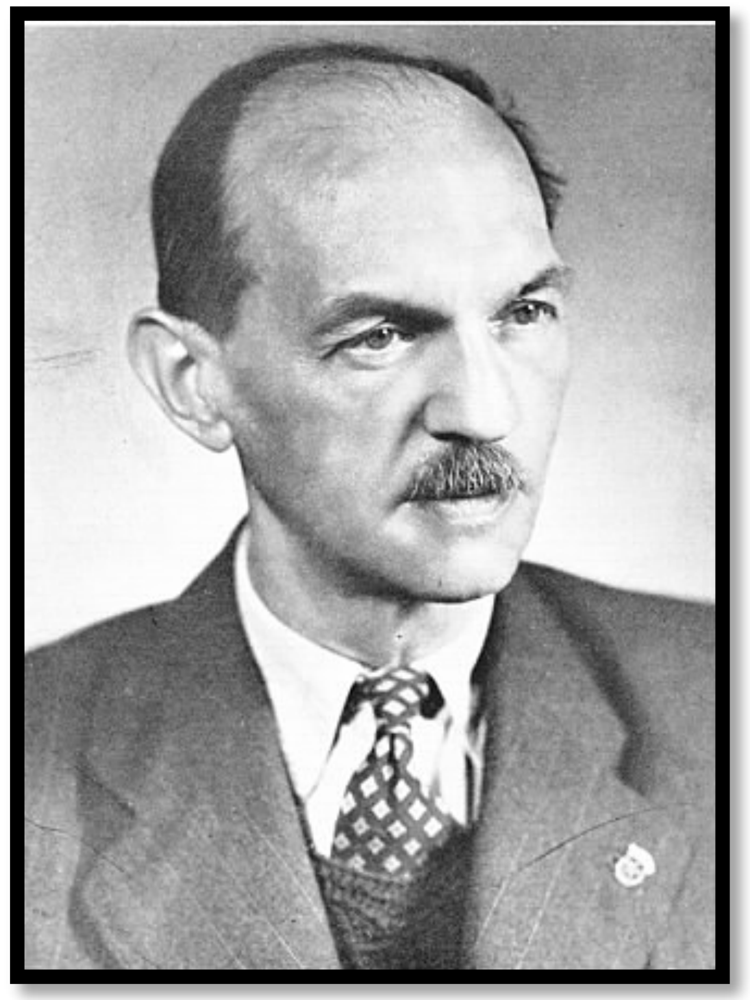
While S ≠ **V**:

- Let {**u**,**v**} be the min cost edge such that **u** is in S, **v** is not in S.
- T = T + {**u**,**v**}
- S = S + **v**

Output T

Usually known as Prim's algorithm.
(due to a 1957 publication by Robert Prim)

First discovered by Vojtech Jarník,
who described it in a letter to Boruvka,
and later published it in 1930.



Boruvka himself had published a different
algorithm in 1926.

Correctness of Algorithm

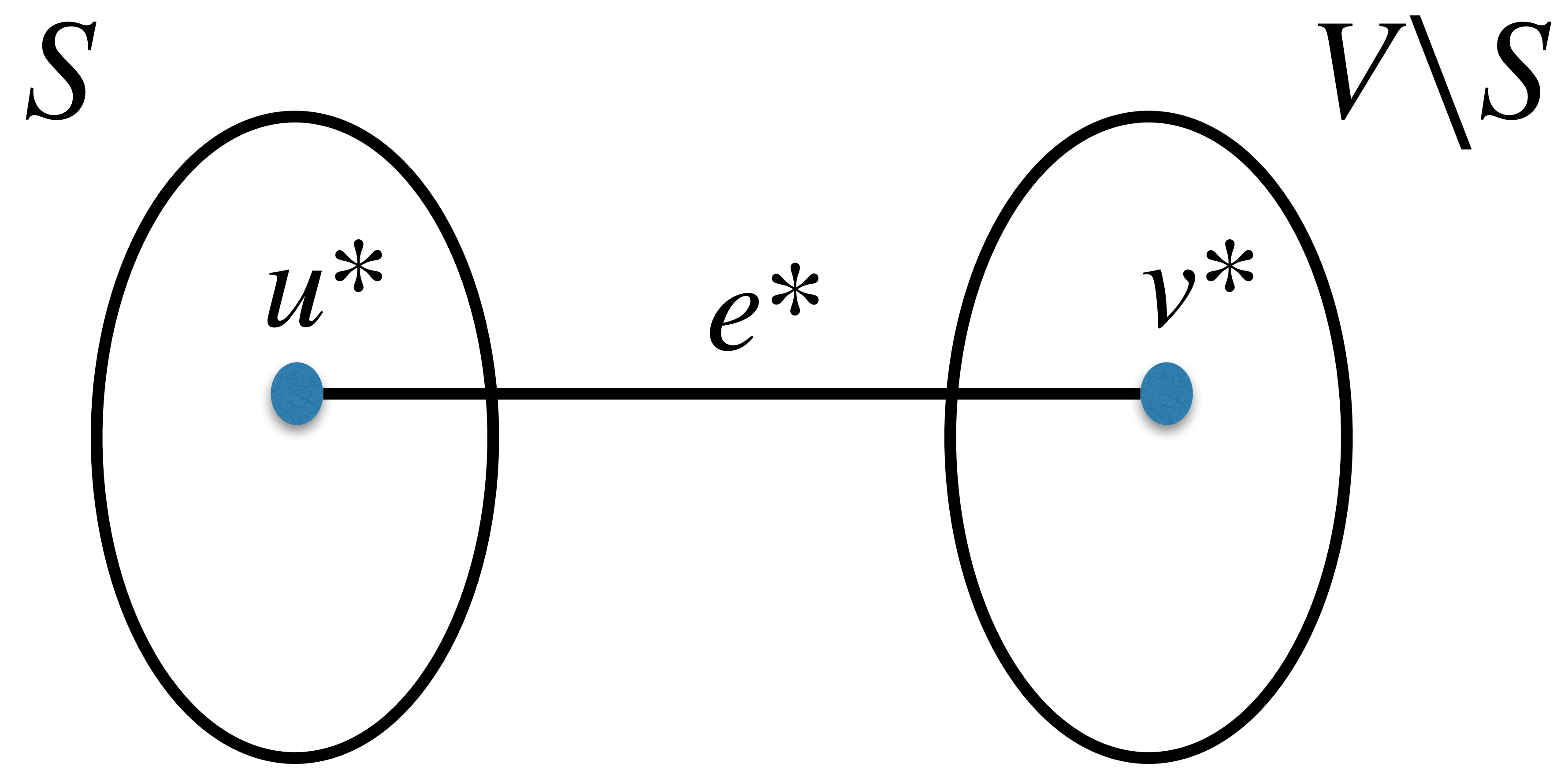
Lemma (MST Cut Property):

Let $G = (V, E)$ be a connected graph with distinct positive edge costs.

Let $S \subset V \quad (S \neq \emptyset,\ S \neq V)$.

Let $e^* = \{u^*, v^*\}$ be the cheapest edge with $u^* \in S,\ v^* \notin S$.

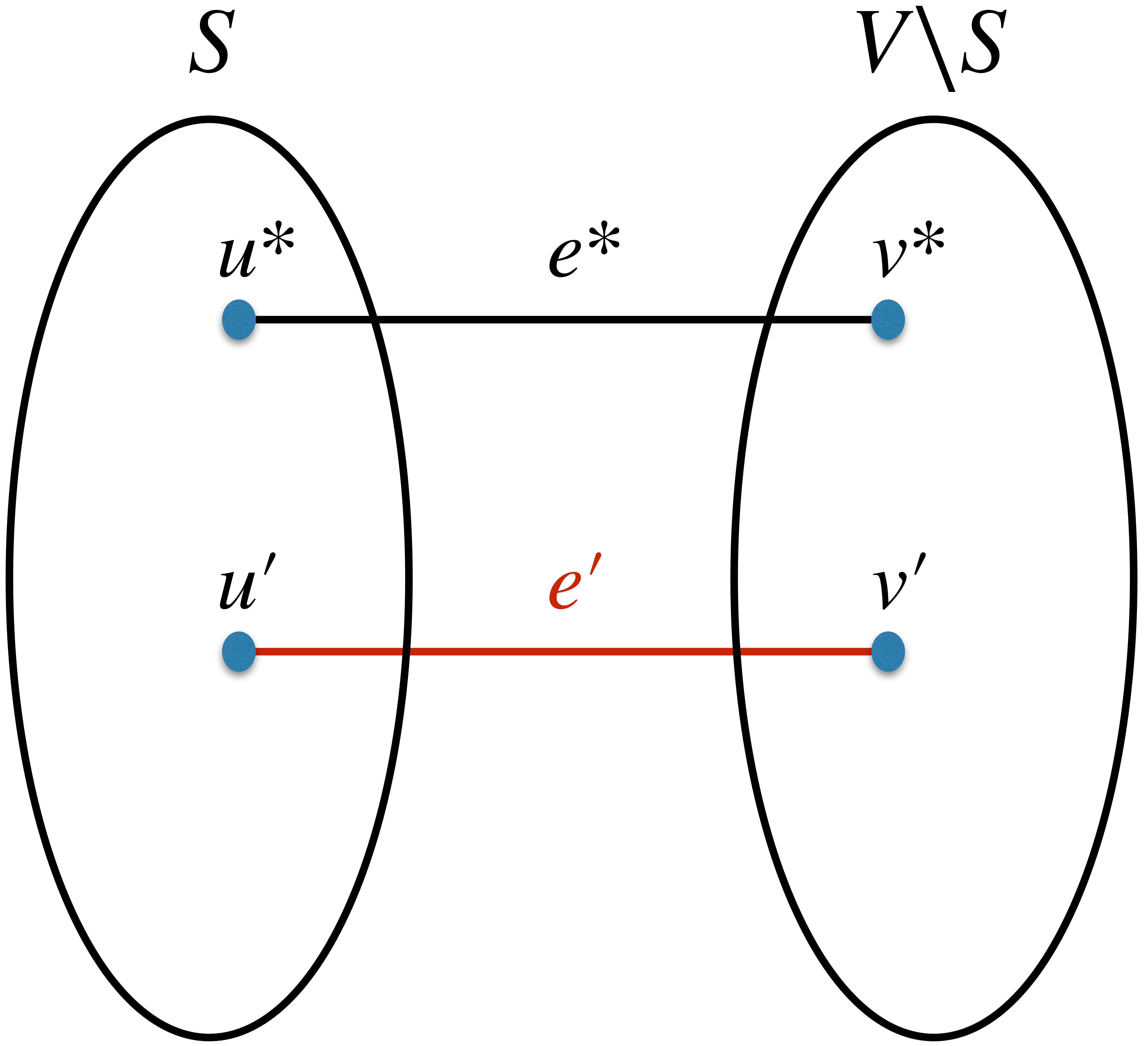
Then the MST must contain e^*.



Correctness of algorithm:

Every time alg. adds an edge, we know it must be in MST.

Correctness of Algorithm



Proof Idea: (proof by contradiction)

Let **T** be the MST.

Suppose $e^* = \{u^*, v^*\}$ is not in **T**.

Pick $e' = \{u', v'\}$ in **T** $(u' \in S, v' \in V\backslash S)$.

(e' chosen carefully)

$c(e') > c(e^*)$

T* = **T** - e' + e^* is a spanning tree with smaller cost.

Why? **T*** has $n-1$ edges. Argue it must be connected.

Race for World Record

A naïve implementation of Jarník-Prim runs in time $O(m^2)$.

A better implementation runs in time $O(m \log m)$.

In practice, this is pretty good!

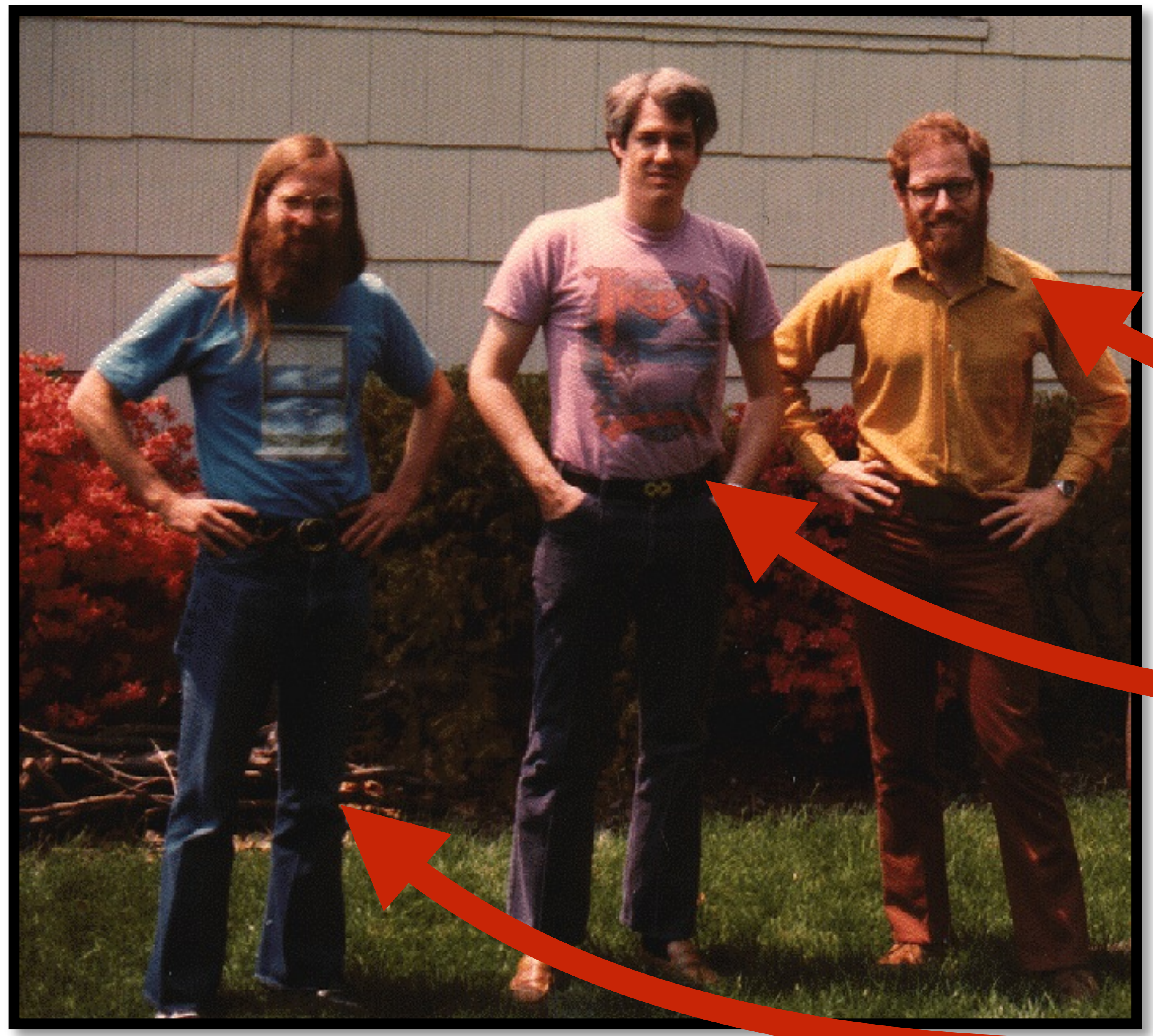
But a good algorithm designer always thinks:

Can we do better?

Race for World Record

1984: Fredman & Tarjan invent the "Fibonacci heap" data structure.

Running time improved from $O(m \log m)$ to $O(m \log^* m)$.



Race for World Record

1986: Gabow, Galil, T. Spencer, Tarjan improved the alg.

Running time improved from $O(m \log^* m)$ to $O(m \log(\log^* m))$.

Gabow

Galil

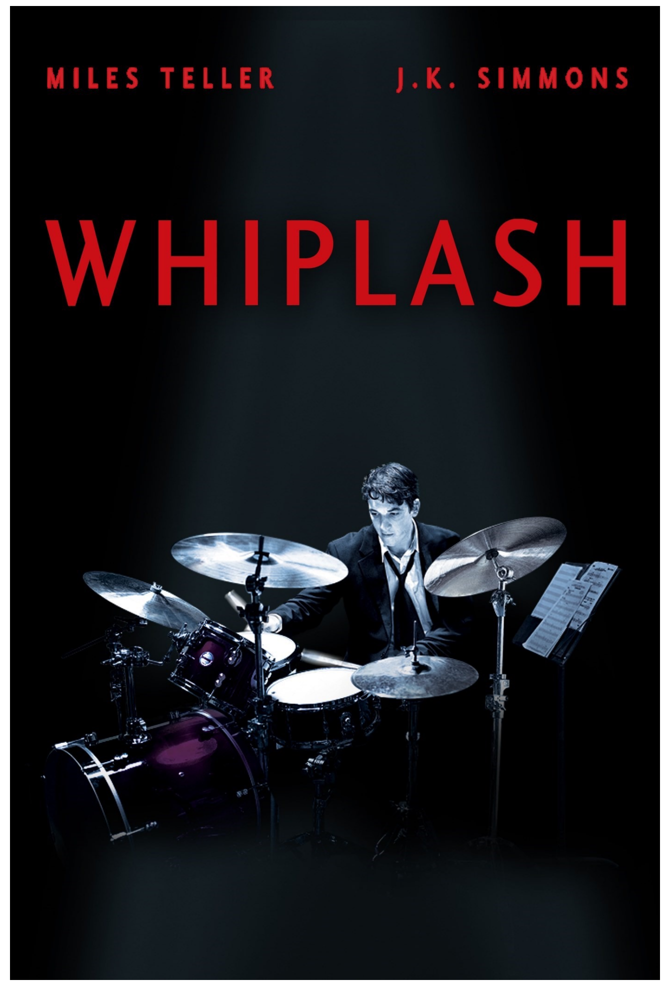
Tarjan & Not-Spencer

Race for World Record

1997: Chazelle invents "soft heap" data structure.

Running time improved from $O(m \log^* m)$ to $O(m\,\alpha(m)\,\log \alpha(m))$

What is $\alpha(m)$???



Bernard Chazelle   Damien Chazelle   (writer & director)

Race for World Record

What is $\alpha(m)$???

It is known as the Inverse-Ackermann function.

$\log^*(m)$ # times you do $\log$ to go down to 2.

$\log^{**}(m)$ # times you do $\log^*$ to go down to 2.

$\log^{***}(m)$ # times you do $\log^{**}$ to go down to 2.

$\alpha(m)$ # $*$'s you need so that $\log^{***\cdots***}(m) \leq 2$

Incomprehensibly small!

Race for World Record

2002: Pettie & Ramachandran gave a new algorithm.

They proved it is running time is O(optimal).

Would you like to know the running time?

So would we! It is **unknown**.
All we know is: whatever it is, it's optimal.

Pettie

Ramachandran

Next Time: Matching Algorithms