

# LU Factorization

**Geometric Algorithms**  
**Lecture 12**

# Practice Problem

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}$$

*Determine the inverse of the above matrix in every way that we've discussed*

**Answer**

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}$$

# Objectives

- » Motivate **matrix factorization** in general, and the LU factorization in specific
- » Recall elementary row operations and connect them to matrices
- » Look at the **LU factorization**, how to find it, and how to use it

# LU Factorization

# Matrix Factorization

# Matrix Factorization

A **factorization** of a matrix  $A$  is an equation which expresses  $A$  as a product of one or more matrices, e.g.,

$$A = BC$$

# Matrix Factorization

A **factorization** of a matrix  $A$  is an equation which expresses  $A$  as a product of one or more matrices, e.g.,

$$A = BC$$

So far, we've been given two factors and asked to find their product

**Factorization is the harder direction**



# Reasons to Factorize

# Reasons to Factorize

Writing  $A$  as the product of multiple matrices can

# Reasons to Factorize

Writing  $A$  as the product of multiple matrices can

» make computing with  $A$  faster

# Reasons to Factorize

Writing  $A$  as the product of multiple matrices can

- » make computing with  $A$  faster

- » make working with  $A$  easier

# Reasons to Factorize

Writing  $A$  as the product of multiple matrices can

- » make computing with  $A$  faster

- » make working with  $A$  easier

- » expose important information about  $A$

# Reasons to Factorize

Writing  $A$  as the product of multiple matrices can

» make computing with  $A$  faster LU Decomposition

» make working with  $A$  easier

» expose important information about  $A$

# The Problem

**Question.** For an matrix  $A$ , solve the equations

$$A\mathbf{x} = \mathbf{b}_1 \quad , \quad A\mathbf{x} = \mathbf{b}_2 \quad \dots \quad A\mathbf{x} = \mathbf{b}_{k-1} \quad , \quad A\mathbf{x} = \mathbf{b}_k$$

**In other words:** we want to solve several matrix equations over the same matrix

# The Problem

**Question.** For a matrix  $A$ , solve (for  $X$ ) in the equation

$$AX = B$$

where  $X$  and  $B$  are matrices of appropriate dimension

**This is (essentially) the same question**



# The Problem

**Question.** Solve  $AX = B$

If  $A$  is *invertible*, then we have a solution:

Find  $A^{-1}$  and then  $X = A^{-1}B$

# The Problem

**Question.** Solve  $AX = B$

If  $A$  is *invertible*, then we have a solution:

Find  $A^{-1}$  and then  $X = A^{-1}B$

**What if  $A^{-1}$  is not invertible?**

**Even if it is, can we do it faster?**

# LU Factorization at a High Level

Given a  $m \times n$  matrix  $A$ , we are going to factorize  $A$  as

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_L \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\substack{\text{echelon form of } A \\ U}}$$

# LU Factorization at a High Level

Given a  $m \times n$  matrix  $A$ , we are going to factorize  $A$  as

echelon form of  $A$

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_L \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_U$$

**Note.** This applies to non-square matrices

# What are "L" and "U"?

L stands for "lower" as in *lower triangular*

U stands for "upper" as in *upper triangular*

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix} \begin{bmatrix} \blacksquare & * & * & * \\ 0 & \blacksquare & * & * \\ 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & \blacksquare \end{bmatrix}$$

$L$   $U$

# The Fundamental Question

$$A = LU$$

echelon form of  $A$

# The Fundamental Question

$$A = LU$$

echelon form of  $A$

We know how to build  $U$ , that's just the forward phase of Gaussian elimination

# The Fundamental Question

$$A = LU$$

echelon form of  $A$

We know how to build  $U$ , that's just the forward phase of Gaussian elimination

**How do we build  $L$ ?**



# The Fundamental Question

$$A = LU$$

echelon form of  $A$

We know how to build  $U$ , that's just the forward phase of Gaussian elimination

**How do we build  $L$ ?**

**The idea.**  $L$  "implements" the row operations of the forward phase

# Elementary Matrices

# Recall: Elementary Row Operations

scaling	multiply a row by a number
interchange	switch two rows
replacement	add a scaled equation to another

# The First Key Observation

# The First Key Observation

Elementary row operations are **linear transformations**  
(viewed as transformation on columns)

# The First Key Observation

Elementary row operations are **linear transformations**  
(viewed as transformation on columns)

**Example:** Scale row 2 by 5

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \xrightarrow{R_2 \leftarrow 5R_2} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 5a_{21} & 5a_{22} & 5a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

# Example: Scaling

Restricted to one column, we see this is the above linear transformation

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ 5v_2 \\ v_3 \end{bmatrix}$$

# Example: Scaling

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \mapsto \begin{bmatrix} v_1 \\ 5v_2 \\ v_3 \end{bmatrix}$$

Let's build the matrix which implements it:



# Another Example: Scaling + Replacement

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ (a_{31} - 2a_{11}) & (a_{32} - 2a_{12}) & (a_{33} - 2a_{13}) \end{bmatrix}$$

$$R_3 \leftarrow (R_3 - 2R_1)$$

# Another Example: Scaling + Replacement

$$R_3 \leftarrow (R_3 - 2R_1)$$

Elementary row operations are  
linear, so they are implemented  
by matrices

# General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform  $R_3 \leftarrow kR_3$  then we need the identity matrix but with the entry  $A_{33} = k$ .

# General Elementary Scaling Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform  $R_3 \leftarrow kR_3$  then we need the identity matrix but with the entry  $A_{33} = k$ .

If we want to perform  $R_i \leftarrow kR_i$  then we need the identity matrix but with the entry  $A_{ii} = k$ .

# General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

# General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform  $R_4 \leftarrow R_4 + kR_1$ , then we need the identity matrix but with the entry  $A_{41} = k$ .



# General Replacement Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ k & 0 & 0 & 1 \end{bmatrix}$$

If we want to perform  $R_4 \leftarrow R_4 + kR_1$ , then we need the identity matrix but with the entry  $A_{41} = k$ .

If we want to perform  $R_i \leftarrow R_i + kR_j$ , then we need the identity matrix but with the entry  $A_{ij} = k$ .

# General Swap Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

If we want to swap  $R_2$  and  $R_3$ , then we need the identity matrix, but with  $R_2$  and  $R_3$  swapped.

# Elementary Matrices

**Definition.** An **elementary matrix** is a matrix obtained by applying a single row operation to the identity matrix  $I$

**Example.**

# How To: Finding Elementary Matrices

**Question.** Find the matrix implementing the elementary row operation  $op$

**Solution.** Apply  $op$  to the identity matrix of the appropriate size

# Products of Elementary Matrices

# Products of Elementary Matrices

Taking stock:

# Products of Elementary Matrices

Taking stock:

» Elementary matrices implement elementary row operations

# Products of Elementary Matrices

Taking stock:

- » Elementary matrices implement elementary row operations
- » Remember that Matrix multiplication is transformation composition (i.e., do one then the other)



# Products of Elementary Matrices

Taking stock:

- » Elementary matrices implement elementary row operations
- » Remember that Matrix multiplication is transformation composition (i.e., do one then the other)

**So we can implement any sequence of row operations as a product of elementary matrices**

# How to: Matrices implementing Row Operations

**Question.** Find the matrix implementing a sequence of row operations  $op_1, op_2, \dots$

**Solution.** Apply the row operations in sequence to the identity matrix of the appropriate size

# Question

*Find the matrix implementing the following sequence of elementary row operations on a  $3 \times n$  matrix.*

$$R_2 \leftarrow 3R_2$$

$$R_1 \leftarrow R_1 + R_2$$

$$R_2 \leftrightarrow R_3$$

*Then multiply it with the all-ones  $3 \times 3$  matrix.*

# Answer

$$R_2 \leftarrow 3R_2$$

$$R_1 \leftarrow R_1 + R_2$$

$$R_2 \leftrightarrow R_3$$

# Second Key Observation

# Second Key Observation

Elementary row operations are **invertible** linear transformations

# Second Key Observation

Elementary row operations are **invertible** linear transformations

This also means the product of elementary matrices is invertible

$$(E_1 E_2 E_3 E_4)^{-1} = E_4^{-1} E_3^{-1} E_2^{-1} E_1^{-1}$$

**!! the order reverses !!**

# Question (Conceptual)

*Describe the inverse transformation for each elementary row operation*



# Question (Conceptual)

*Describe the inverse transformation for each elementary row operation*

The inverse of scaling by  $k$  is scaling by  $1/k$

# Question (Conceptual)

*Describe the inverse transformation for each elementary row operation*

The inverse of scaling by  $k$  is scaling by  $1/k$

The inverse of  $R_i \leftarrow R_i + kR_j$  is  $R_i \leftarrow R_i - kR_j$

# Question (Conceptual)

*Describe the inverse transformation for each elementary row operation*

The inverse of scaling by  $k$  is scaling by  $1/k$

The inverse of  $R_i \leftarrow R_i + kR_j$  is  $R_i \leftarrow R_i - kR_j$

The inverse of swapping is swapping again

# Recall: Elementary Row Operations

scaling	multiply a row by a number
interchange	switch two rows
replacement	add a scaled equation to another

# Recall: Elementary Row Operations

We only need these two for the forward phase

interchange

switch two rows

replacement

add a scaled equation to another

# Recall: Elementary Row Operations

We'll assume we only need this

replacement      add a scaled equation to another

# Reminder: LU Factorization at a High Level

Given a  $m \times n$  matrix  $A$ , we are going to factorize  $A$  as

$$A = \begin{matrix} & \begin{matrix} \text{Echelon form of } A \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix} & \begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ L & U \end{matrix}$$

# Gaussian Elimination and Elementary Matrices

$$A \sim A_1 \sim A_2 \sim \dots \sim A_k$$

Consider a sequence of elementary row operations from  $A$  to an echelon form

Each step can be represent as a **product with an elementary matrix**



# Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \dots \sim E_k E_{k-1} \dots E_2 E_1 A$$

# Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \dots \sim E_k E_{k-1} \dots E_2 E_1 A$$

This exactly tells us that if  $B$  is the final echelon form we get then

$$B = (E_k E_{k-1} \dots E_2 E_1) A = EA$$

where  $E$  implements a sequence of row operations. So:

# Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \dots \sim E_k E_{k-1} \dots E_2 E_1 A$$

This exactly tells us that if  $B$  is the final echelon form we get then

**Invertible**

$$B = (E_k E_{k-1} \dots E_2 E_1) A = EA$$

where  $E$  implements a sequence of row operations. So:

# Gaussian Elimination and Elementary Matrices

$$A \sim E_1 A \sim E_2 E_1 A \sim \dots \sim E_k E_{k-1} \dots E_2 E_1 A$$

This exactly tells us that if  $B$  is the final echelon form we get then

**Invertible**

$$B = (E_k E_{k-1} \dots E_2 E_1) A = EA$$

where  $E$  implements a sequence of row operations. So:

$$A = E^{-1} B = (E_1^{-1} E_2^{-1} \dots E_{k-1}^{-1} E_k^{-1}) B$$

# LU Factorization Algorithm

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization( $A$ ):  
2       $L \leftarrow$  identity matrix  
3       $U \leftarrow A$ 
```



# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps  
5      FOR each row operation OP in the prev step:
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps  
5      FOR each row operation OP in the prev step:  
6          E ← the matrix implementing OP
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps  
5      FOR each row operation OP in the prev step:  
6          E ← the matrix implementing OP  
7          L ← L @ E-1      # note the multiplication on the right
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps  
5      FOR each row operation OP in the prev step:  
6          E ← the matrix implementing OP  
7          L ← L @ E-1      # note the multiplication on the right  
8      RETURN (L, U)
```

# LU Factorization Algorithm

```
1  FUNCTION LU_Factorization(A):  
2      L ← identity matrix  
3      U ← A  
4      convert U to an echelon form by GE forward step # without swaps  
5      FOR each row operation OP in the prev step:  
6          E ← the matrix implementing OP  
7          L ← L @ E-1      # note the multiplication on the right  
8      RETURN (L, U)      we'll see how to do this more efficiently
```

The forward part of Gaussian  
elimination is matrix  
factorization

# The "L" Part

$$E = E_k E_{k-1} \dots E_2 E_1$$

This a product of elementary matrices

So  $L = E^{-1} = E_1^{-1} E_2^{-1} \dots E_{k-1}^{-1} E_k^{-1}$  **!! the order reverses !!**

We won't prove this, but it's worth thinking about: **why is this lower triangular?**

And can we build this in a more efficient way?



demo

# How To: LU Factorization by hand

**Question.** Find a LU Factorization for the matrix  $A$  (assuming no swaps)

**Solution.**

- » Start with  $L$  as the identity matrix
- » Find  $U$  by the forward part of GE
- » For each operation  $R_i \leftarrow R_i + kR_j$ , set  $L_{ij}$  to  $-k$

# Practice Problem

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}$$

Determine an LU factorization of the above matrix using this procedure

**Answer**

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix}$$

Determine an LU factorization of the above matrix using this procedure

# Analyzing LU Factorization

# Analyzing Linear Algebra Algorithms

# Analyzing Linear Algebra Algorithms

We will not use  $O(\cdot)$  notation!

# Analyzing Linear Algebra Algorithms

We will not use  $O(\cdot)$  notation!

For numerics, we care about number of **F**loating-oint  
**O**perations (FLOPs):

- >> addition
- >> subtraction
- >> multiplication
- >> division
- >> square root



# Analyzing Linear Algebra Algorithms

We will not use  $O(\cdot)$  notation!

For numerics, we care about number of **F**loating-oint  
**O**perations (FLOPs):

- >> addition
- >> subtraction
- >> multiplication
- >> division
- >> square root

$2n$  vs.  $n$  is very different  
when  $n \sim 10^{20}$

# Dominant Terms

# Dominant Terms

That said, we don't care about *exact* bounds

# Dominant Terms

That said, we don't care about *exact* bounds

A function  $f(n)$  is ***asymptotically equivalent*** to  $g(n)$  if

$$\lim_{i \rightarrow \infty} \frac{f(i)}{g(i)} = 1$$

# Dominant Terms

That said, we don't care about *exact* bounds

A function  $f(n)$  is ***asymptotically equivalent*** to  $g(n)$  if

$$\lim_{i \rightarrow \infty} \frac{f(i)}{g(i)} = 1$$

For polynomials, they are equivalent to their dominant term

# Dominant Terms

the dominant term of a polynomial is the monomial with the highest degree

$$\lim_{i \rightarrow \infty} \frac{3x^3 + 100000x^2}{3x^3} = 1$$

$3x^3$  dominates the function even though the coefficient for  $x^2$  is so large

# How To: Solving systems with the LU

**Question.** Solve the equation  $A\mathbf{x} = \mathbf{b}$  given that  $A = LU$  is a LU factorization.

**Solution.** First solve  $L\mathbf{x} = \mathbf{b}$  to get a solution  $\mathbf{c}$ , then solve  $U\mathbf{x} = \mathbf{c}$  to get a solution  $\mathbf{d}$ .

Verify:

# How To: Solving systems with the LU

**Question.** Solve the equation  $A\mathbf{x} = \mathbf{b}$  given that  $A = LU$  is a LU factorization.

**Solution.** First solve  $L\mathbf{x} = \mathbf{b}$  to get a solution  $\mathbf{c}$ , then solve  $U\mathbf{x} = \mathbf{c}$  to get a solution  $\mathbf{d}$ .

**Why is this better than just solving  $A\mathbf{x} = \mathbf{b}$ ?**



# FLOPs for Solving General Systems

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

GE Backward:  $\sim 2n^2$  FLOPS

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

GE Backward:  $\sim 2n^2$  FLOPS

Matrix Inversion:  $\sim 2n^3$  FLOPS

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

GE Backward:  $\sim 2n^2$  FLOPS

Matrix Inversion:  $\sim 2n^3$  FLOPS

Matrix-Vector Multiplication:  $\sim 2n^2$  FLOPS

# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

GE Backward:  $\sim 2n^2$  FLOPS

Matrix Inversion:  $\sim 2n^3$  FLOPS

Matrix-Vector Multiplication:  $\sim 2n^2$  FLOPS

**Solving by matrix inversion:  $\sim 2n^3$  FLOPS**



# FLOPs for Solving General Systems

The following FLOP estimates are based on  $n \times n$  matrices

Gaussian Elimination:  $\sim \frac{2n^3}{3}$  FLOPS

GE Forward:  $\sim \frac{2n^3}{3}$  FLOPS

GE Backward:  $\sim 2n^2$  FLOPS

Matrix Inversion:  $\sim 2n^3$  FLOPS

Matrix-Vector Multiplication:  $\sim 2n^2$  FLOPS

**Solving by matrix inversion:  $\sim 2n^3$  FLOPS**

**Solving by Gaussian elimination:  $\sim \frac{2n^3}{3}$  FLOPS**

# FLOPS for solving LU systems

LU Factorization:  $\sim \frac{2n^3}{3}$  FLOPS

Solving  $L\mathbf{x} = \mathbf{b}$ :  $\sim 2n^2$  FLOPS (by "forward" elimination)

Solving  $U\mathbf{x} = \mathbf{c}$ :  $\sim 2n^2$  FLOPS (already in echelon form)

**Solving by LU Factorization:**  $\sim \frac{2n^3}{3}$  FLOPS

If you solve several matrix equations for the same matrix, LU factorization is faster than matrix inversion on the first equation, and the same (asymptotically) in later equations (and it works for rectangular matrices).

# Other Considerations: Density

# Other Considerations: Density

If  $A$  doesn't have too many entries ( $A$  is **sparse**), then it's likely that  $L$  and  $U$  won't either.

# Other Considerations: Density

If  $A$  doesn't have too many entries ( $A$  is **sparse**), then it's likely that  $L$  and  $U$  won't either.

But  $A^{-1}$  may have *many* entries ( $A^{-1}$  is **dense**)

# Other Considerations: Density

If  $A$  doesn't have too many entries ( $A$  is **sparse**), then it's likely that  $L$  and  $U$  won't either.

But  $A^{-1}$  may have *many* entries ( $A^{-1}$  is **dense**)

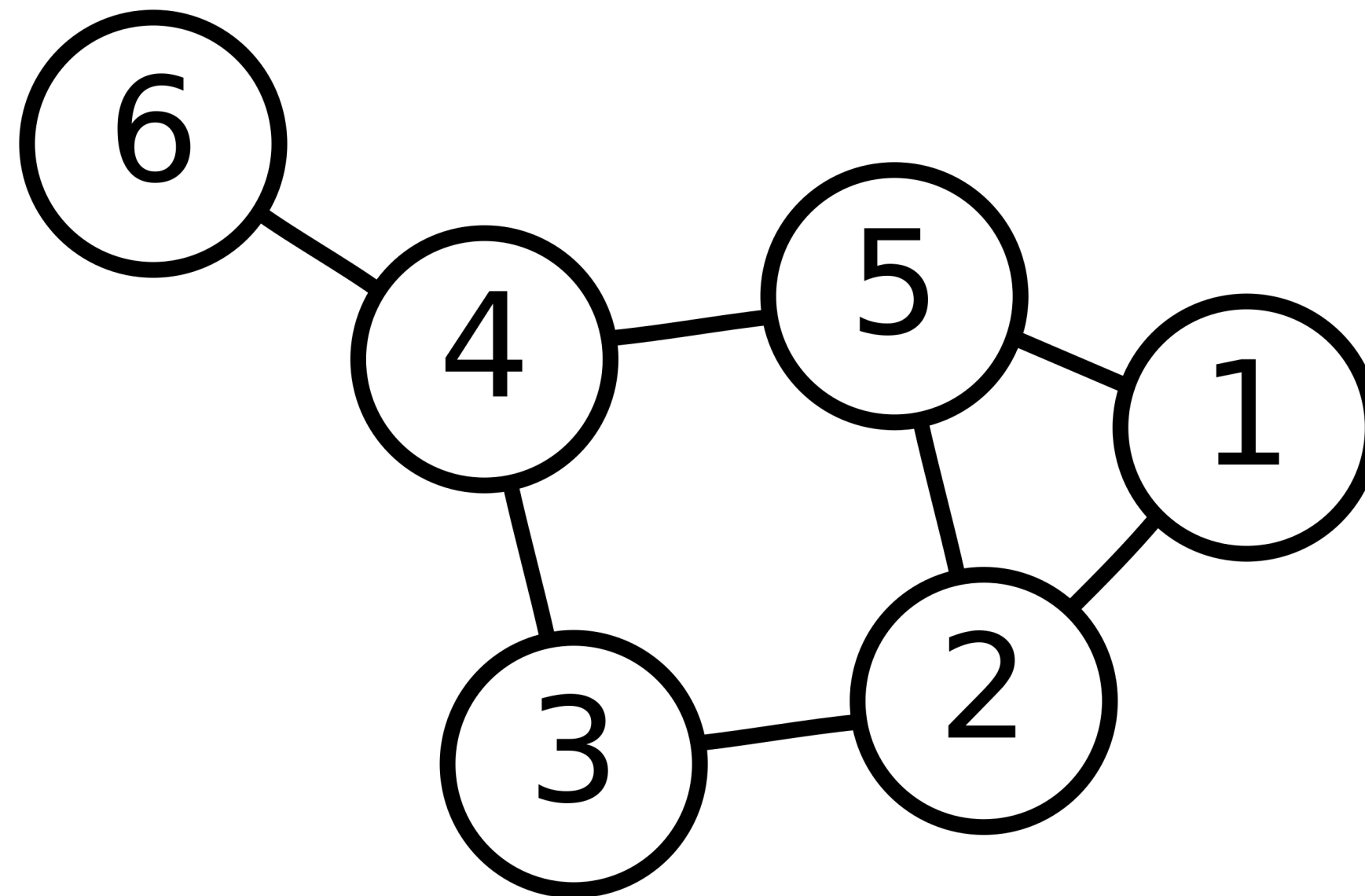
Sparse matrices are faster to compute with and better with respect to storage.

# **Algebraic Graph Theory**



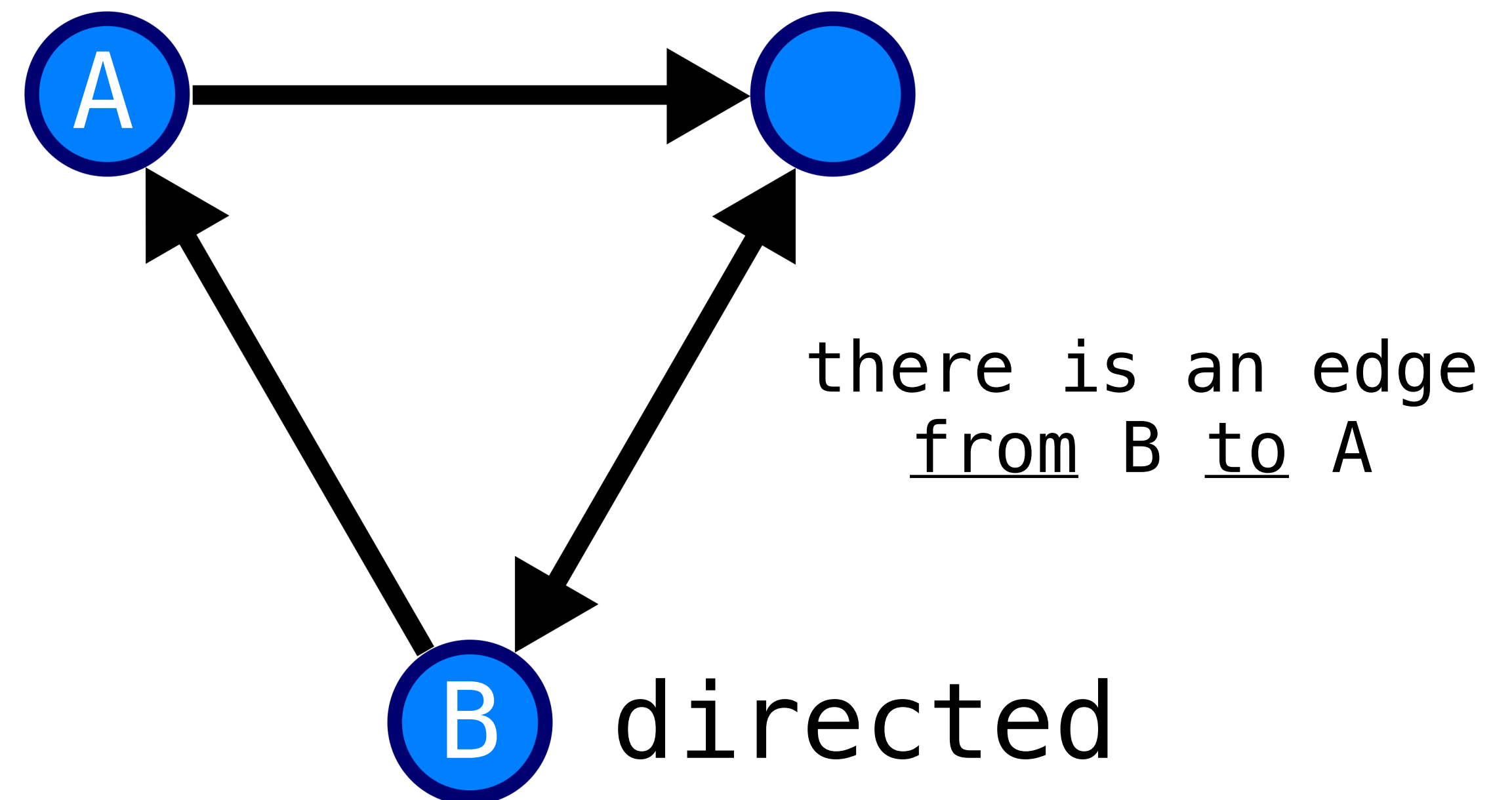
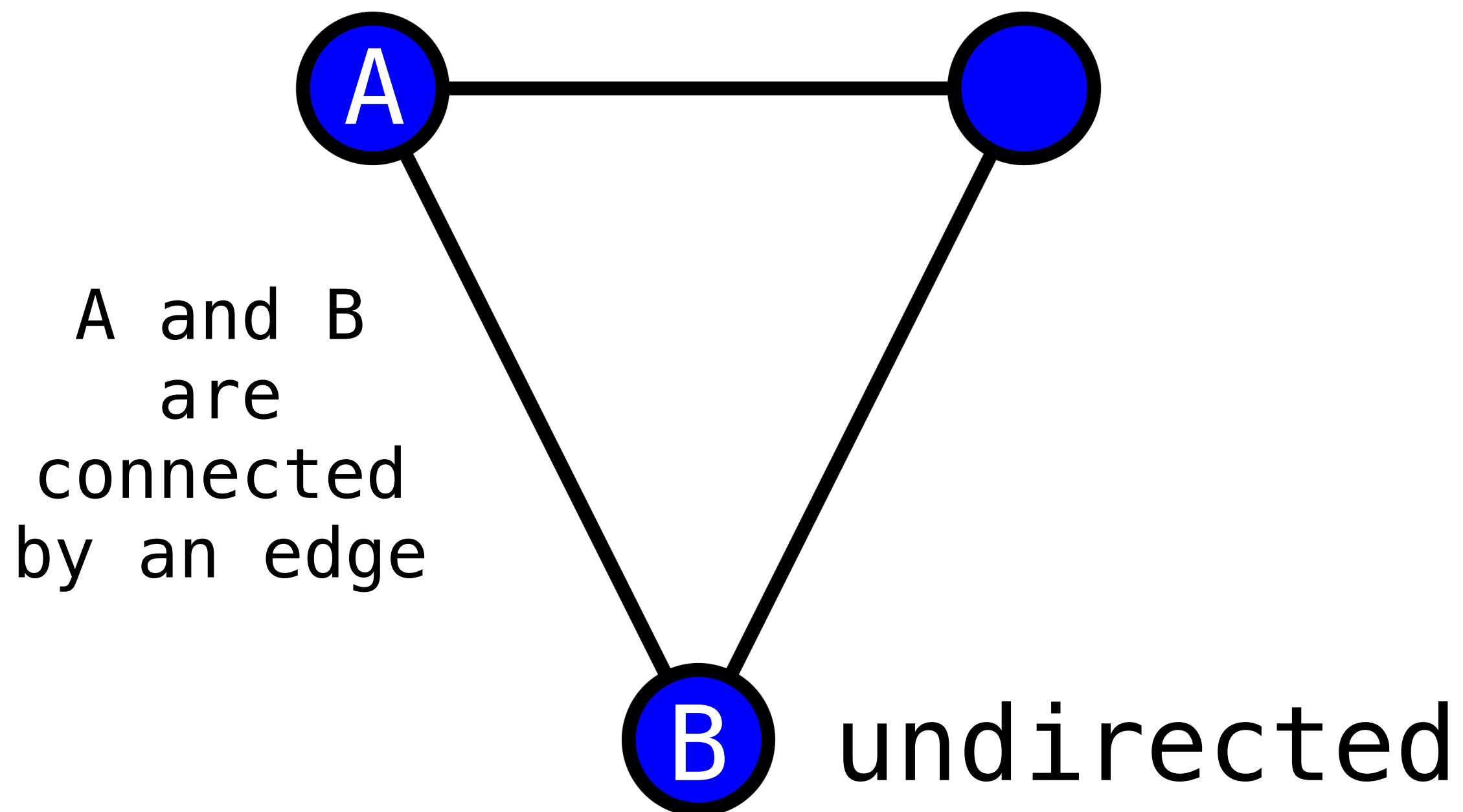
# Graphs

**Definition (Informal).** A graph is a collection of nodes with edges between them.



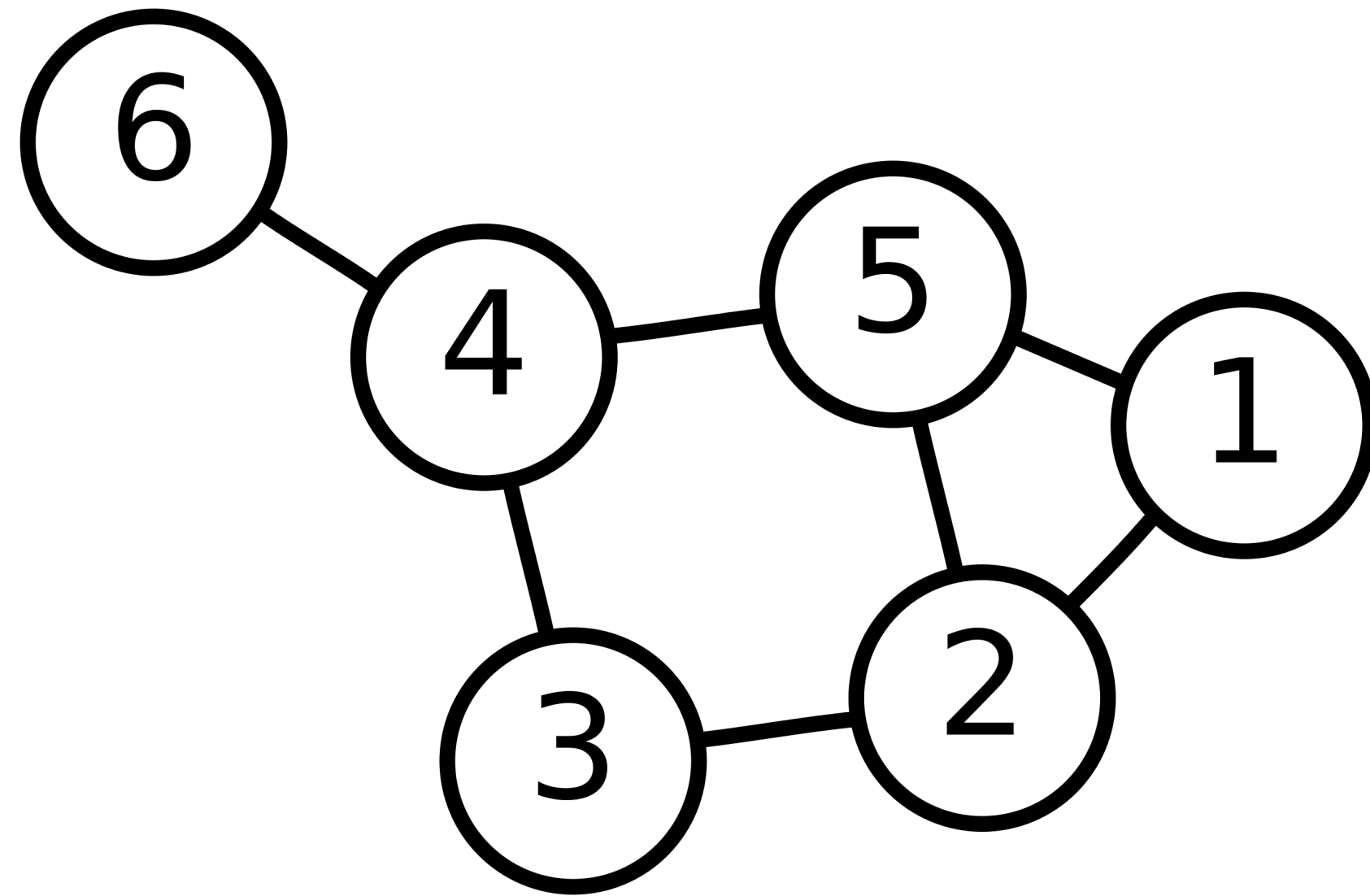
# Directed vs. Undirected Graphs

A graph is **directed** if its edges have a direction.

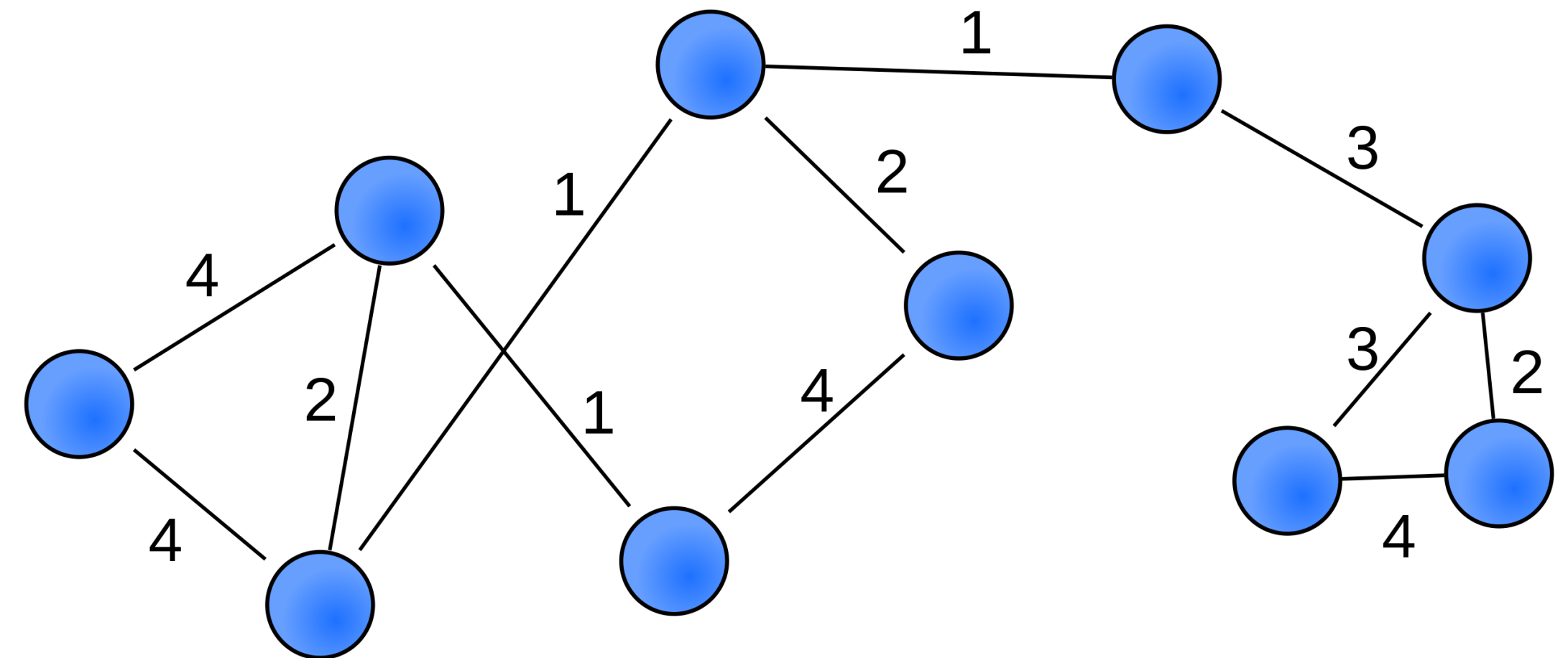


# Weighted vs Unweighted graphs

A graph is **weighted** if its edges have associated values.



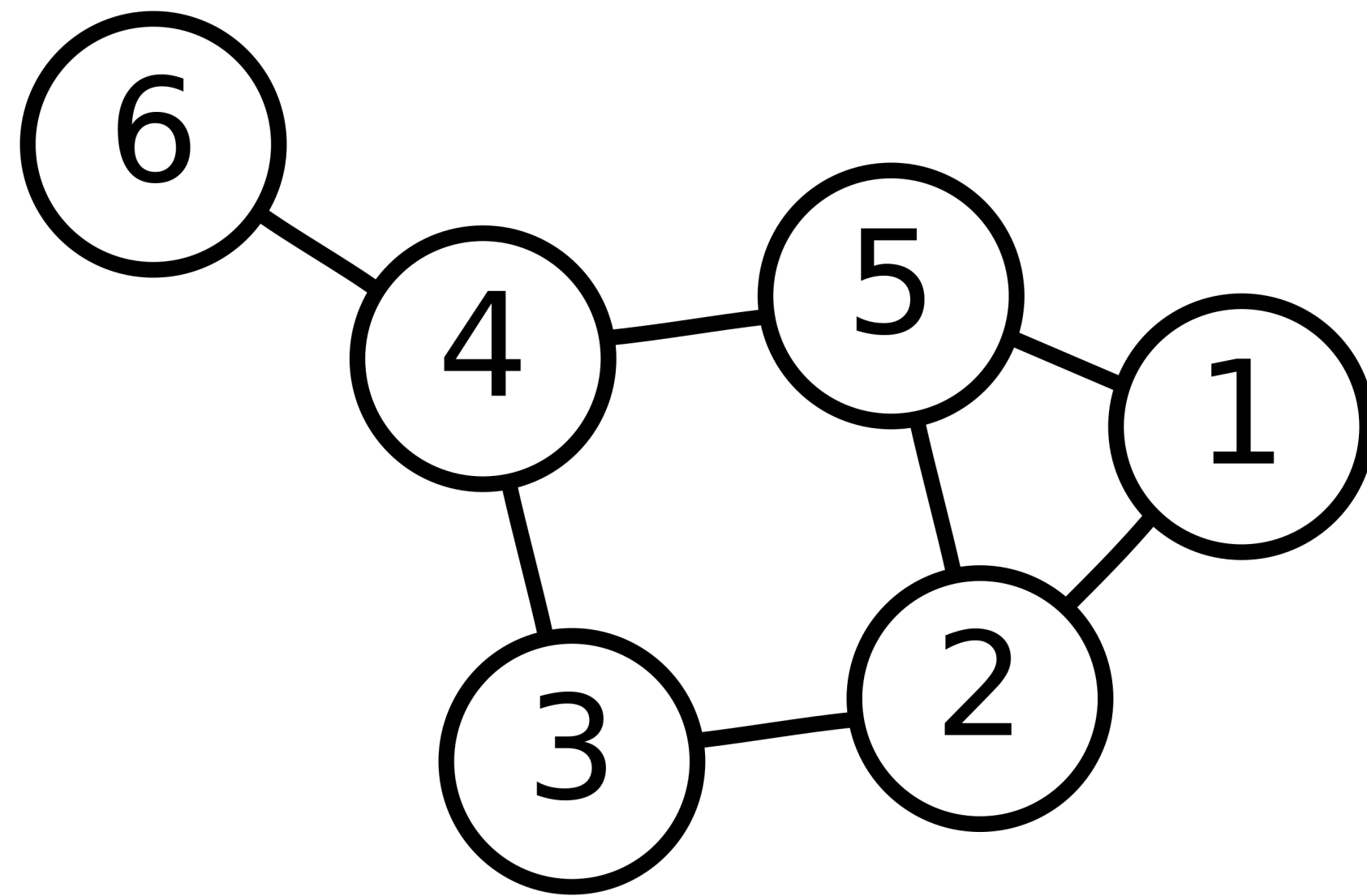
unweighted



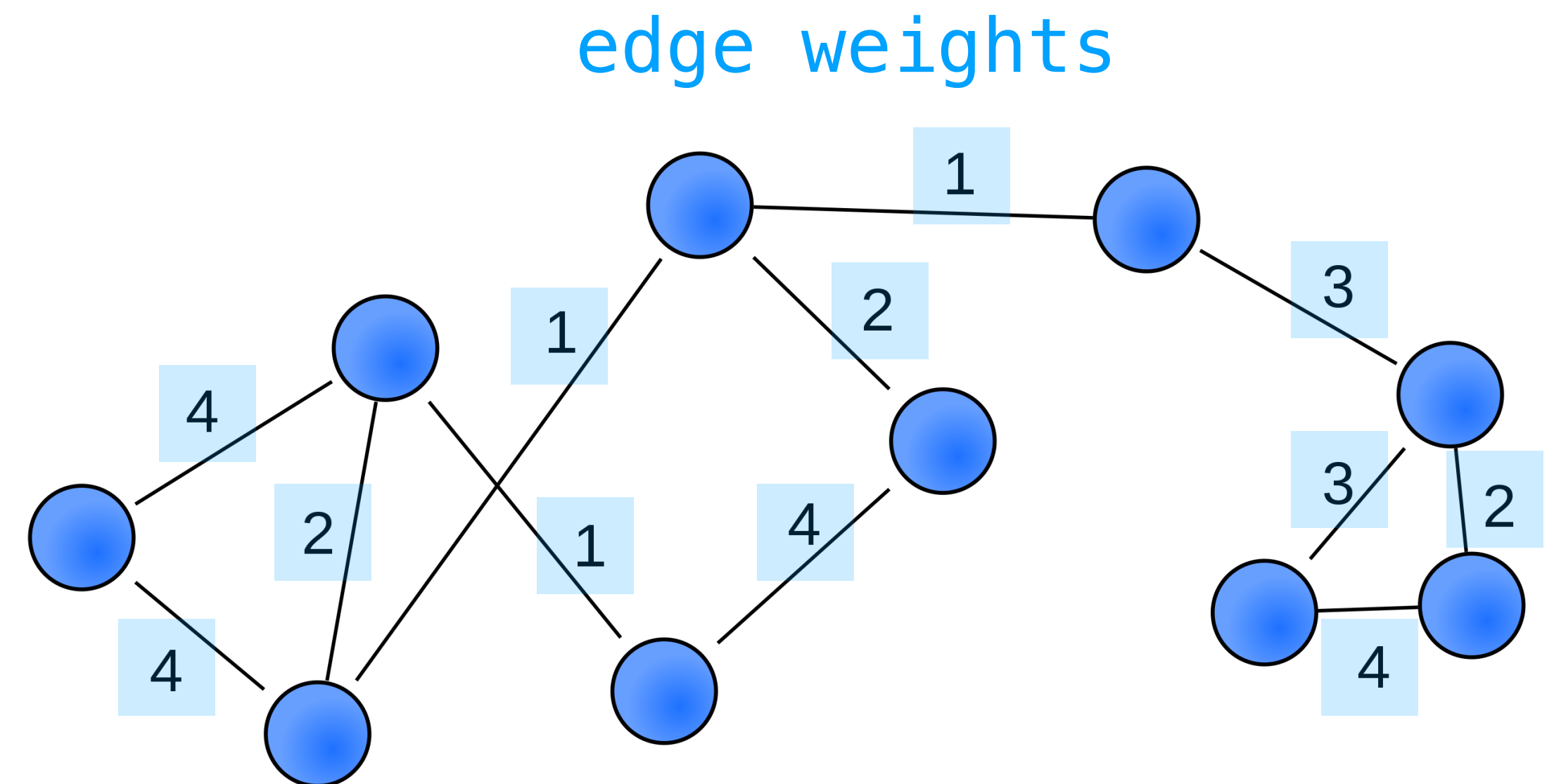
weighted

# Weighted vs Unweighted graphs

A graph is **weighted** if its edges have associated values.



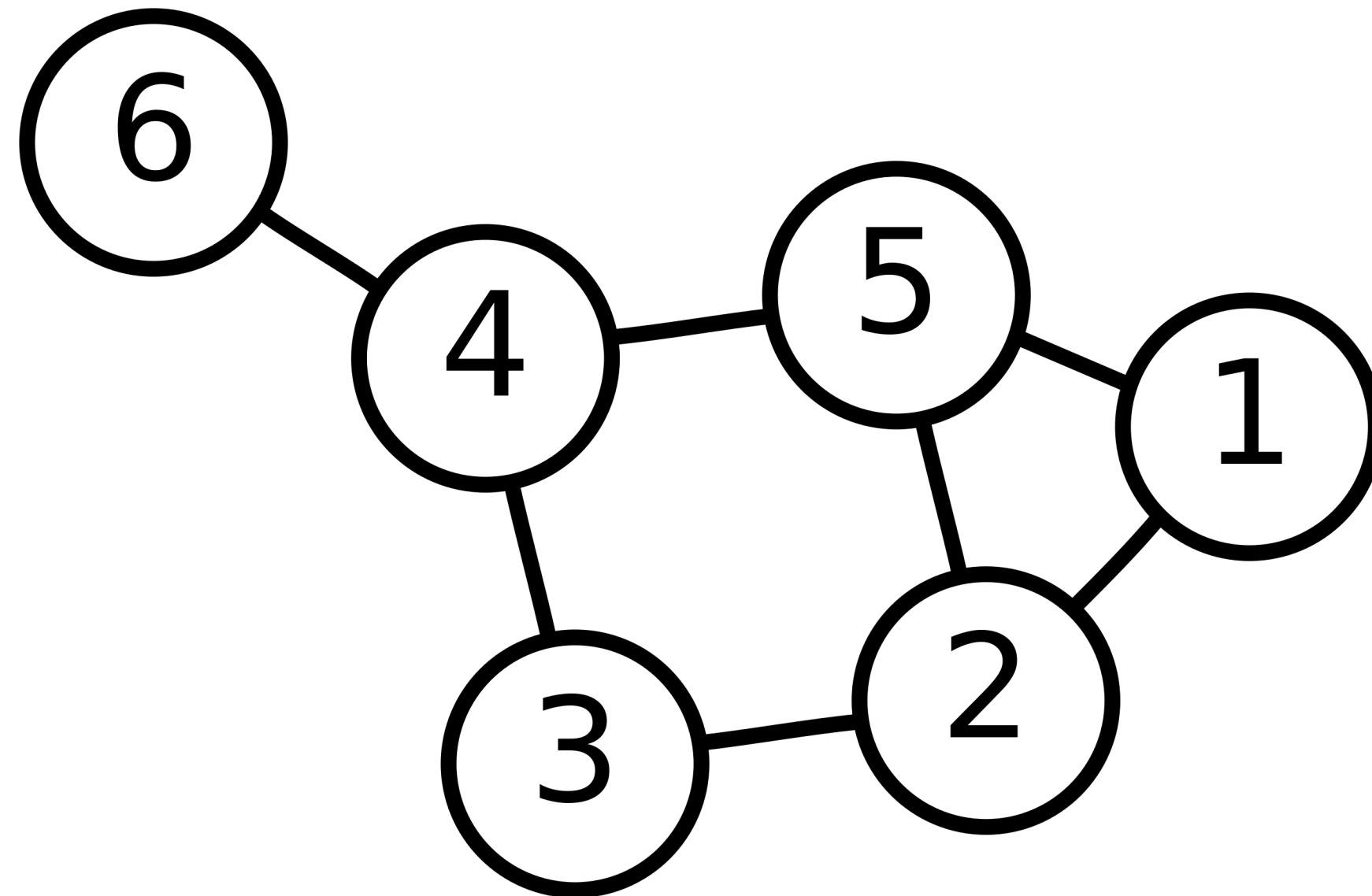
unweighted



weighted

# Simple Graphs

A graph is **simple** if it is undirected, has no self loops, and no multi-edges.



# Four Kinds of Graphs

directed

undirected

weighted

nodes are traffic lights  
edges are streets  
weights are number of lanes

nodes are musicians  
edges are collaborations  
weights are number of collaborations

unweighted

nodes are instagram users  
edges are follows

nodes are bodies of land  
edges are pedestrian bridges

# Four Kinds of Graphs

directed

undirected

weighted

nodes are traffic lights  
edges are streets  
weights are number of lanes

nodes are musicians  
edges are collaborations  
weights are number of collaborations

unweighted

nodes are instagram users  
edges are follows

nodes are bodies of land  
edges are pedestrian bridges

Today

# Four Kinds of Graphs

directed

undirected

weighted

nodes are traffic lights  
edges are streets  
weights are number of lanes

Markov Chains

nodes are musicians  
edges are collaborations  
weights are number of collaborations

unweighted

nodes are instagram users  
edges are follows

nodes are bodies of land  
edges are pedestrian bridges

Today



# Fundamental Question

# Fundamental Question

How do we represent a graph  
formally in a computer?

# Fundamental Question

How do we represent a graph formally in a computer?

There are a couple ways, but one way is to use matrices.

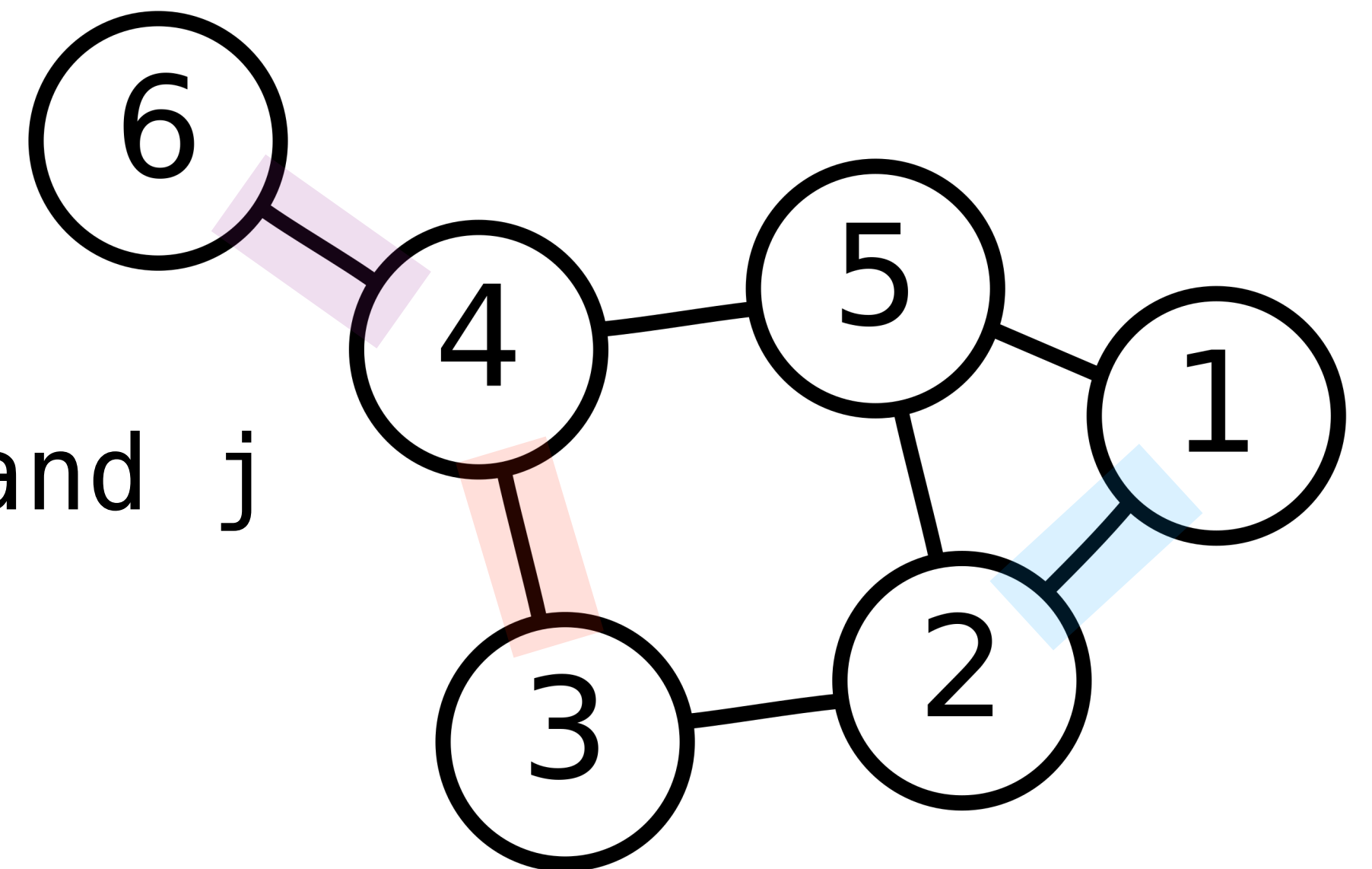
# Adjacency Matrices

Let  $G$  be an simple graph with its nodes labeled by numbers 1 through  $n$ .

We can create the **adjacency matrix**  $A$  for  $G$  as follows.

$$A_{ij} = \begin{cases} 1 & \text{there is an edge between } i \text{ and } j \\ 0 & \text{otherwise} \end{cases}$$

$$\begin{matrix} & & A_{12} & & A_{34} & & A_{46} \\ A_{21} & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix}$$



# Symmetric Matrices

**Definition.** A  $n \times n$  matrix is **symmetric** if

$$A^T = A$$

**Example.**

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

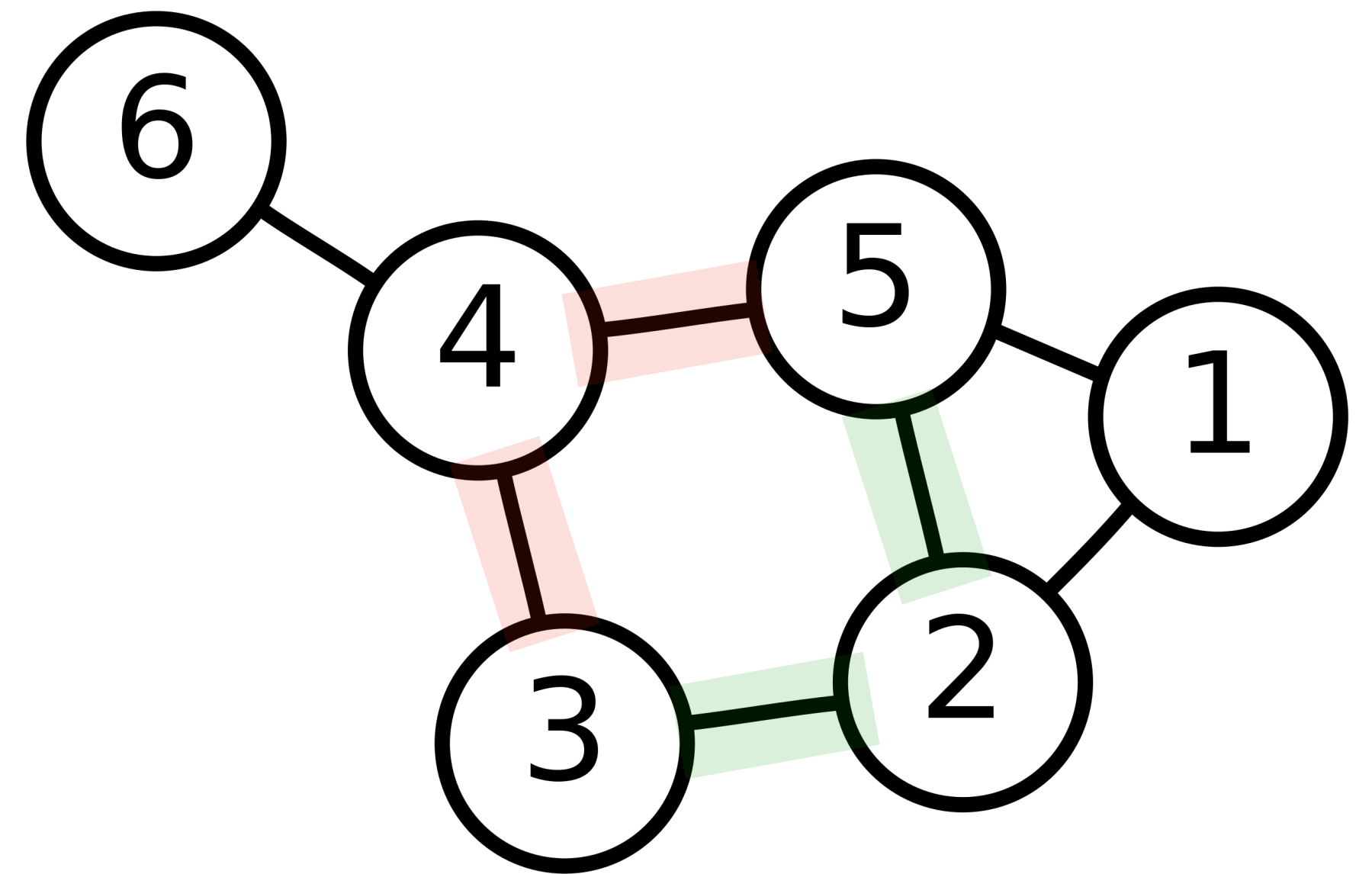
Once we have an adjacency matrix,  
we can do linear algebra on  
graphs.

# Example: Squared Adjacency Matrices

*Given an adjacency matrix  $A$ , can we interpret anything meaningful from  $A^2$ ?*

# Example: Squared Adjacency Matrices

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

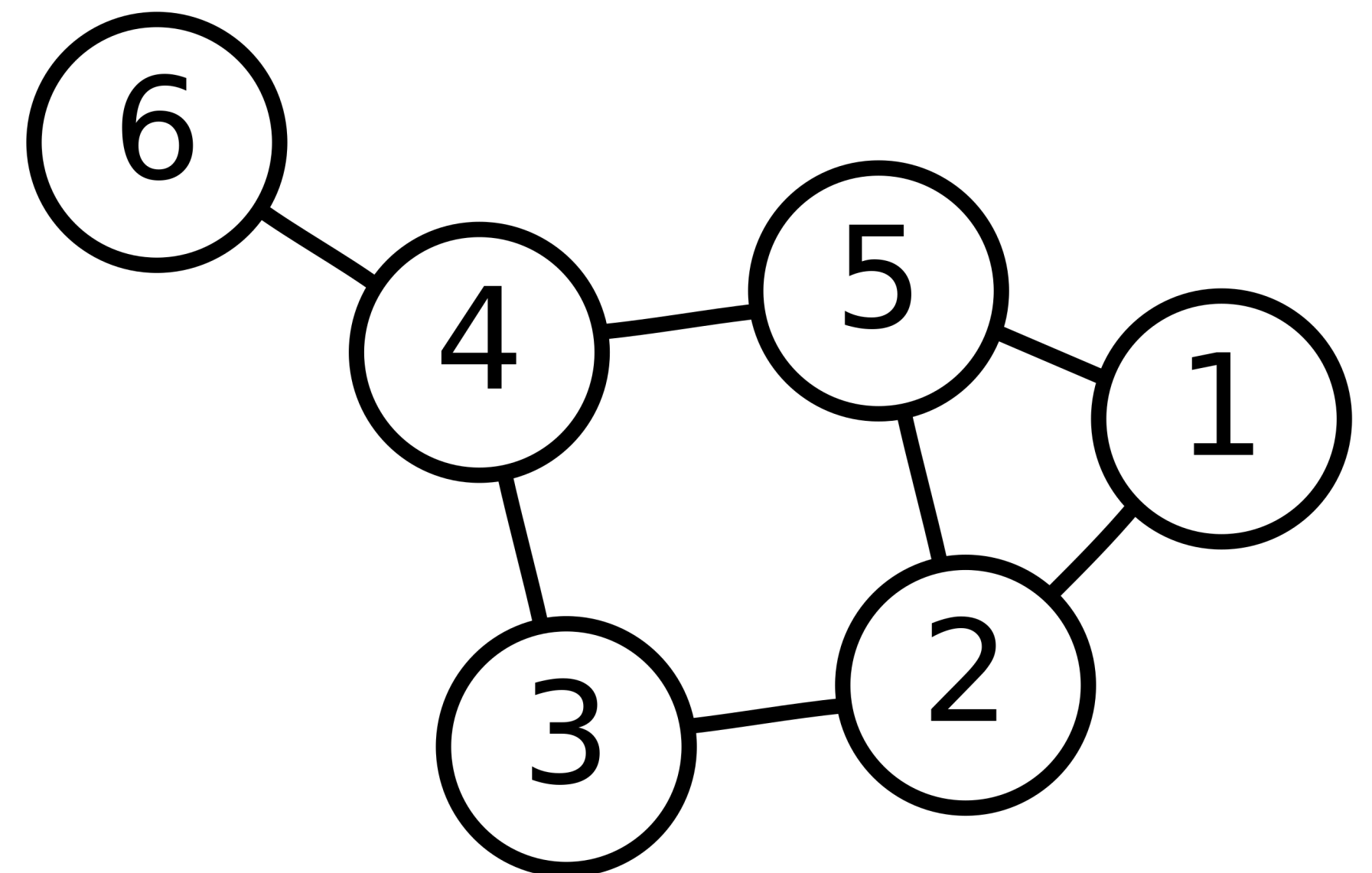


$$(A^2)_{53} = 1(0) + 1(1) + 0(0) + 1(1) + 0(0) + 0(0) = 2$$



# Example: Squared Adjacency Matrices

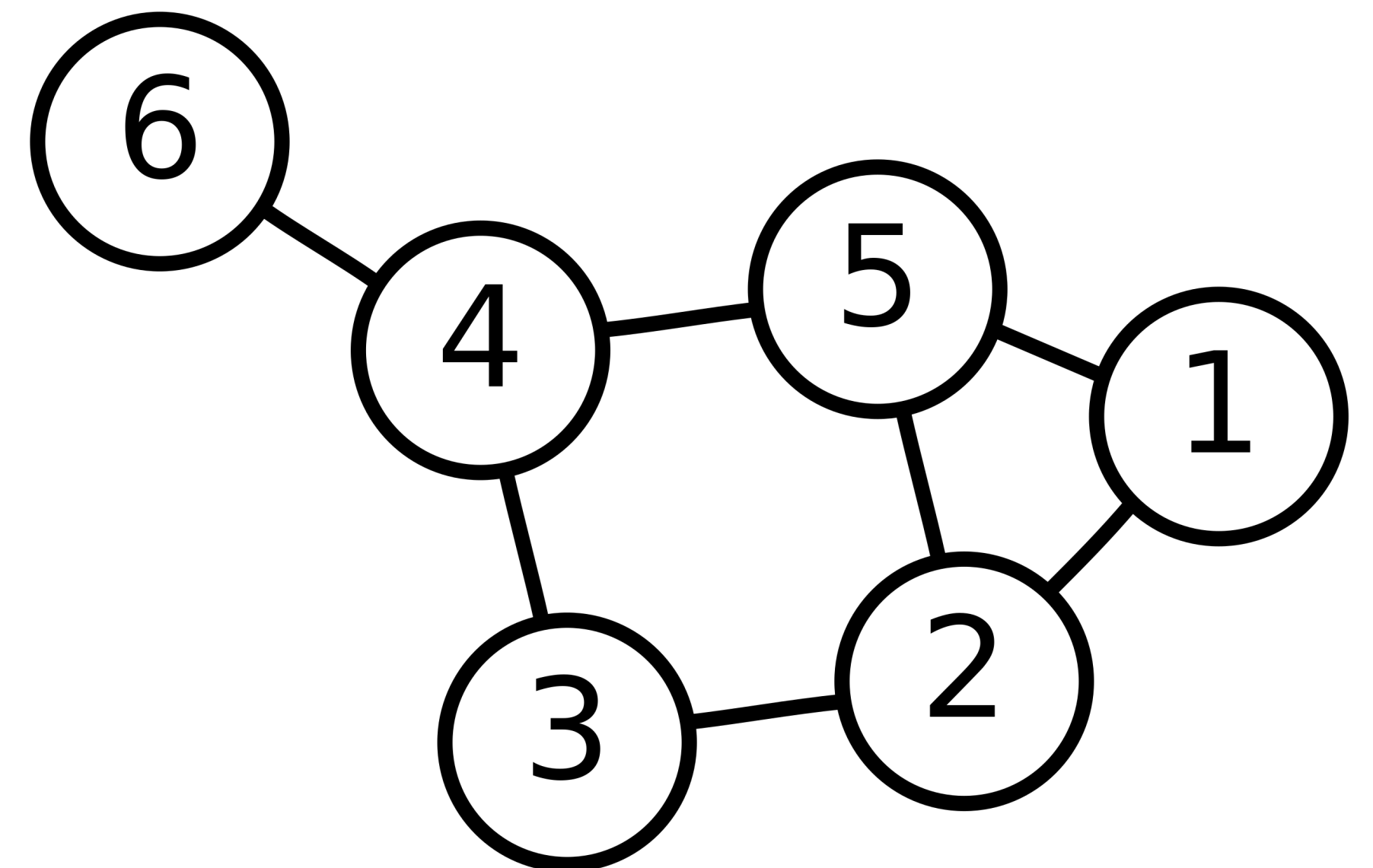
$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \dots + A_{in}A_{nj}$$



# Example: Squared Adjacency Matrices

$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \dots + A_{in}A_{nj}$$

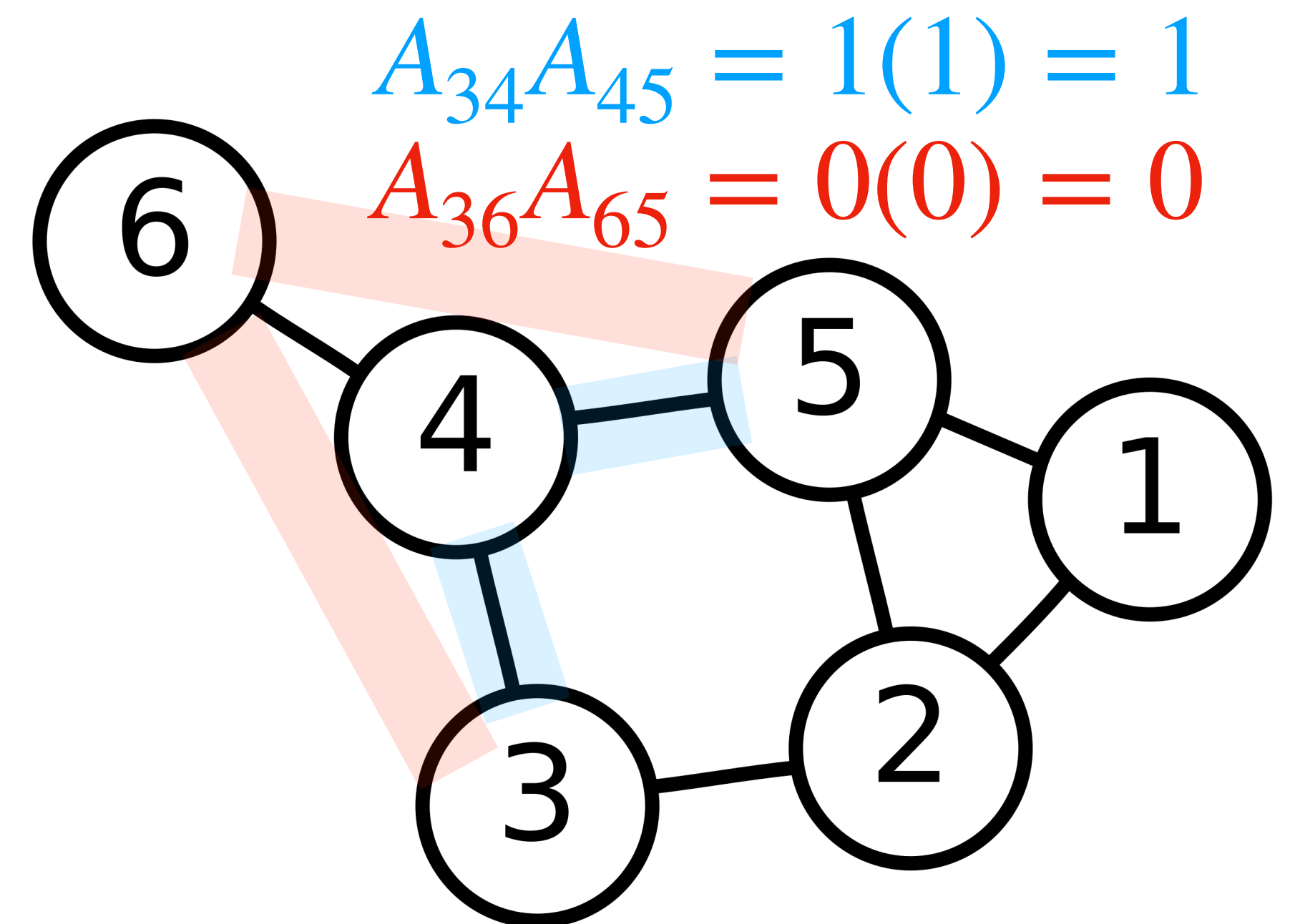
$$A_{ik}A_{kj} = \begin{cases} 1 & \text{there are edges } i \text{ to } k \text{ and } k \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$



# Example: Squared Adjacency Matrices

$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \dots + A_{in}A_{nj}$$

$$A_{ik}A_{kj} = \begin{cases} 1 & \text{there are edges } i \text{ to } k \text{ and } k \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

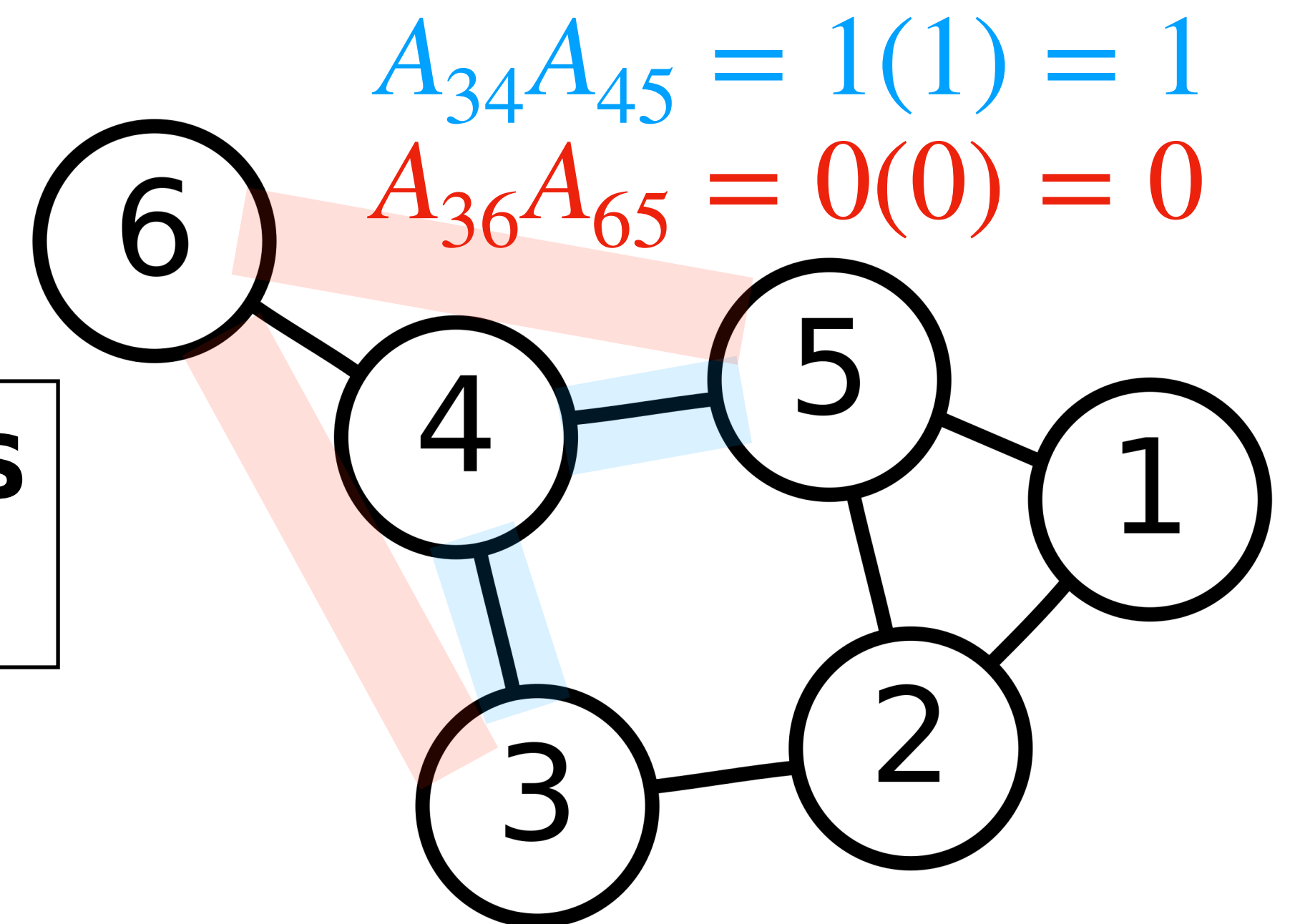


# Example: Squared Adjacency Matrices

$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \dots + A_{in}A_{nj}$$

$$A_{ik}A_{kj} = \begin{cases} 1 & \text{there are edges } i \text{ to } k \text{ and } k \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

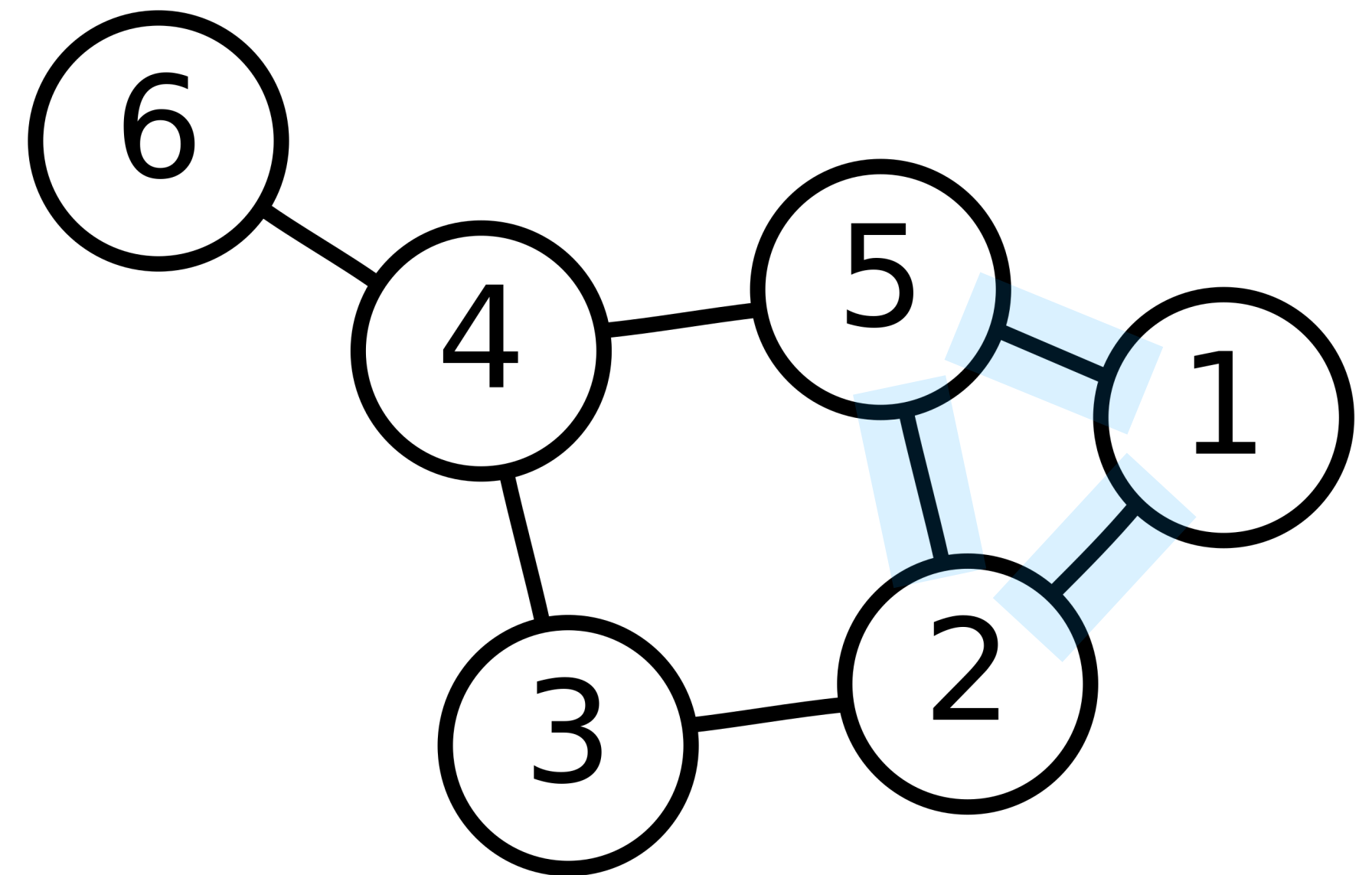
$$(A^2)_{ij} = \text{number of 2-step paths from } i \text{ to } j$$



# Application: Triangle Counting

A **triangle** in an undirected graph is a set of three distinct nodes with edges between every pair of nodes.

Triangles in a social network represent mutual friends and tight cohesion (among other things)



# Application: Triangle Counting (Naive)

```
FUNCTION tri_count_naive(A):
```

```
    count = 0
```

```
    for i from 1 to n:
```

```
        for j from i + 1 to n:
```

```
            for k from j + 1 to n:
```

```
                if  $A_{ij} = 1$  and  $A_{jk} = 1$  and  $A_{ki} = 1$ : # an edge between each pair
```

```
                    count += 1:
```

```
RETURN count
```

# Application: Triangle Counting

**Theorem.** For an adjacency matrix  $A$ , the number of triangle containing the edge  $(i,j)$  is

$$(A^2)_{ij} * A_{ij}$$

Verify:

# Application: Triangle Counting

**FUNCTION** tri\_count( $A$ ):

    compute  $A^2$

    count  $\leftarrow$  sum of  $(A^2)_{ij} * A_{ij}$  for all distinct  $i$  and  $j$

**RETURN** count / 6      # why divided by 6?



# Application: Triangle Counting

**FUNCTION** tri\_count( $A$ ):

# in NumPy '\*' is entry-wise multiplication

# also called the HADAMARD PRODUCT

count  $\leftarrow$  sum of the entries of  $A^2 * A$

**RETURN** count / 6

# Application: Triangle Counting

```
FUNCTION tri_count(A):
```

```
    # in NumPy '*' is entry-wise multiplication
```

```
    #      also called the HADAMARD PRODUCT
```

```
    # and 'np.sum' sums the entry of a matrix
```

```
RETURN np.sum( (A @ A) * A ) / 6
```

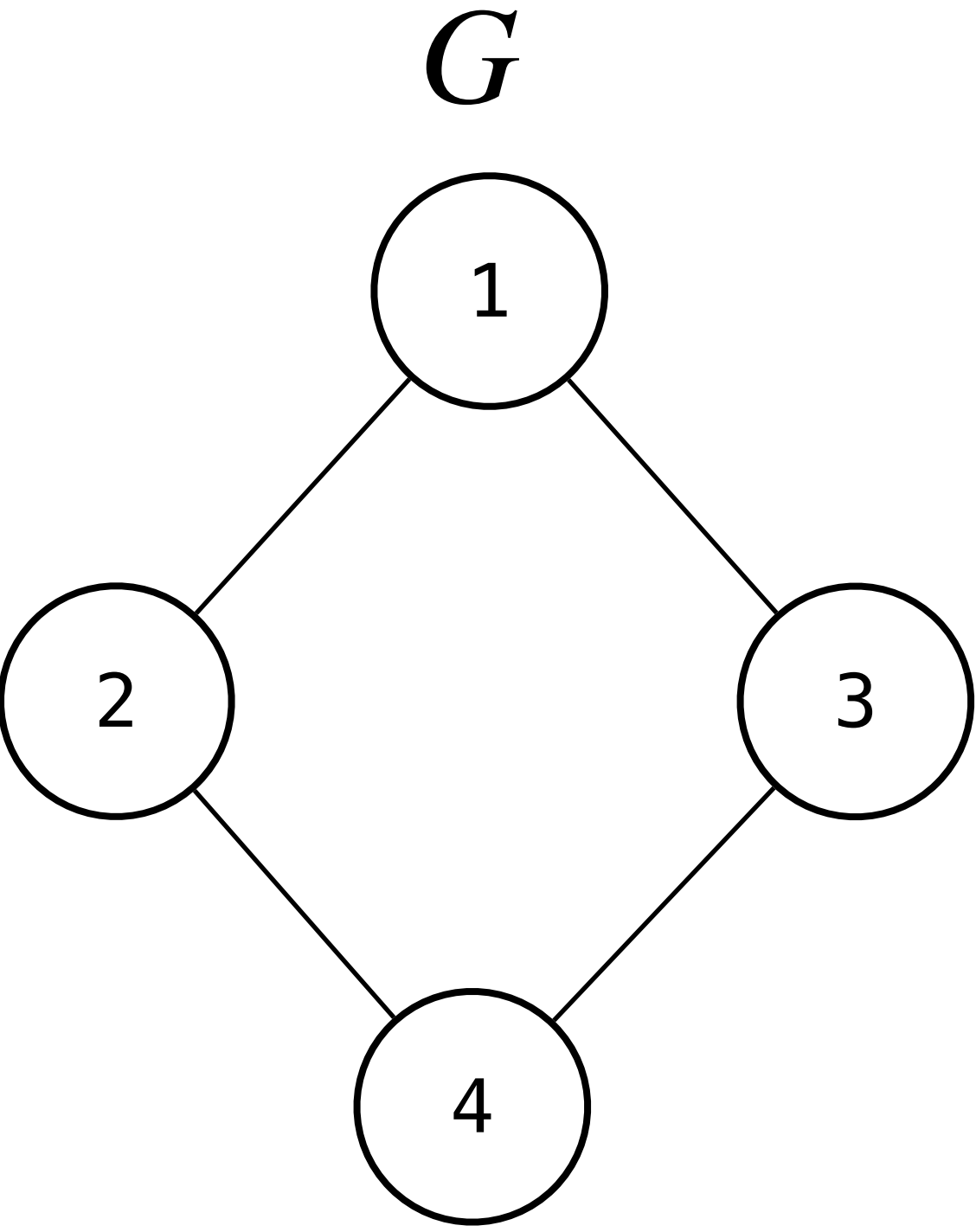
demo

# Another Application: Reachability

**Question:** If  $A^2$  gives us information about length 2 paths, then what about  $A^k$ ?

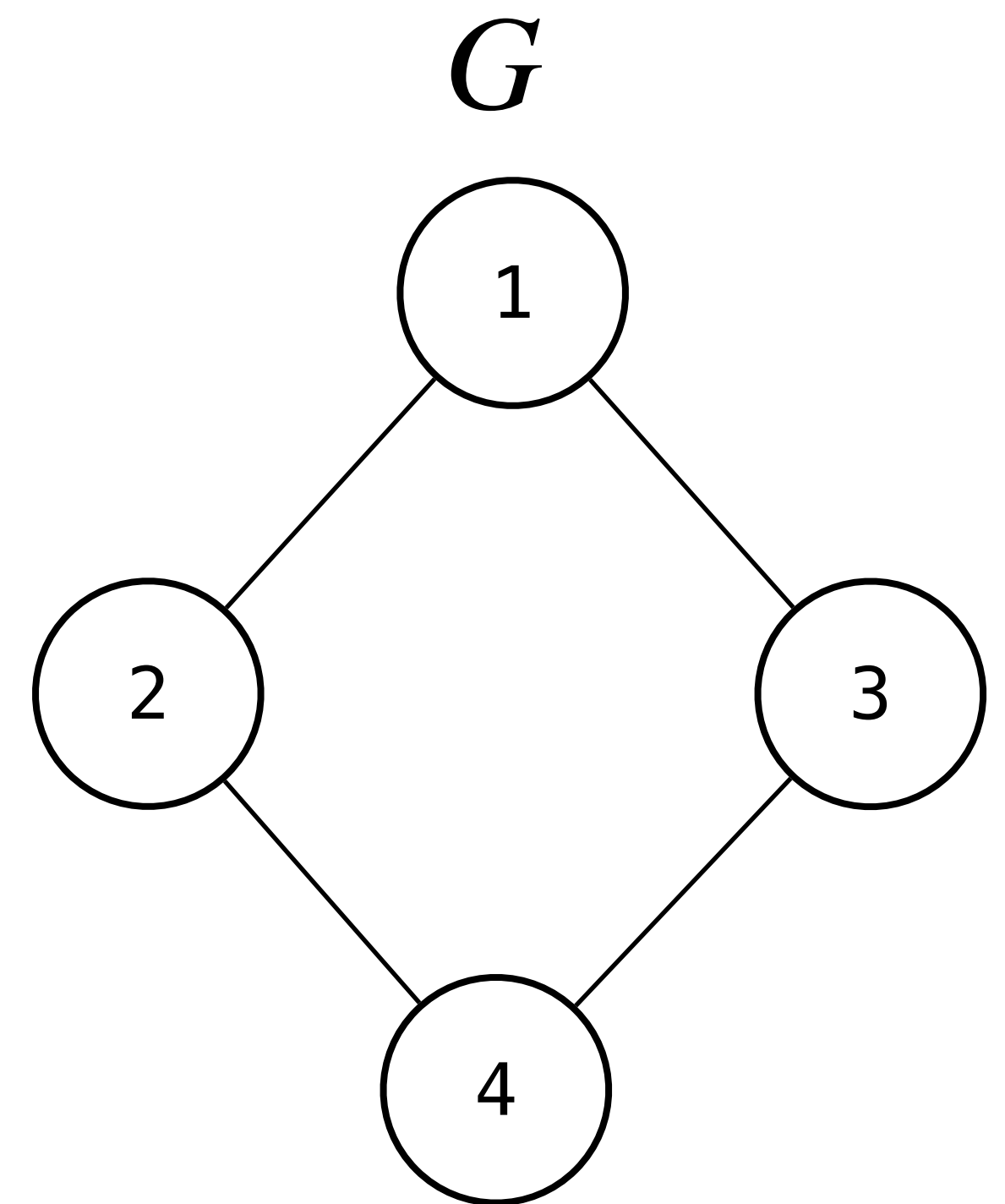
$A^k$  gives us information about  $k$ -length paths.

# Example



# Example

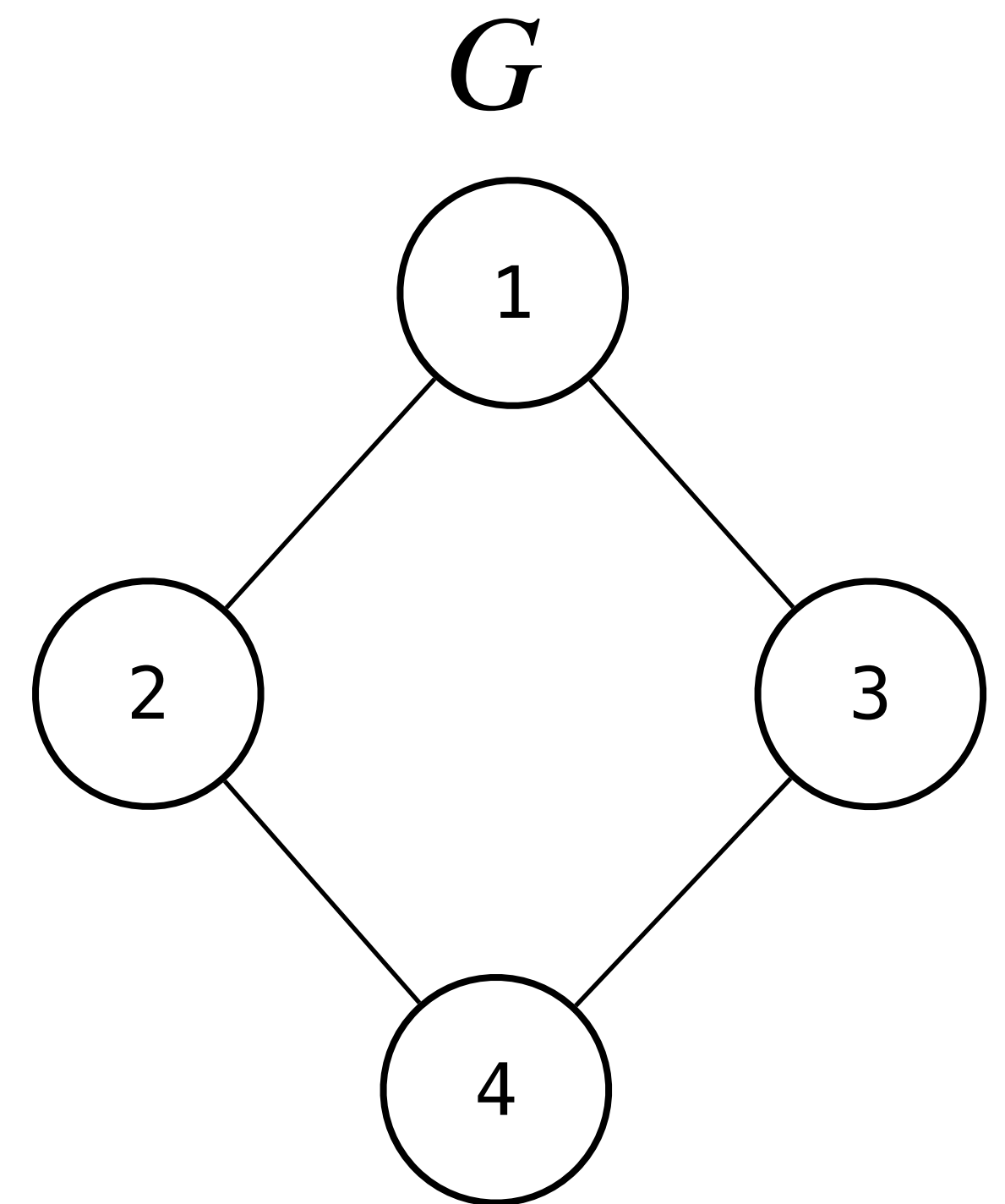
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$



# Example

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

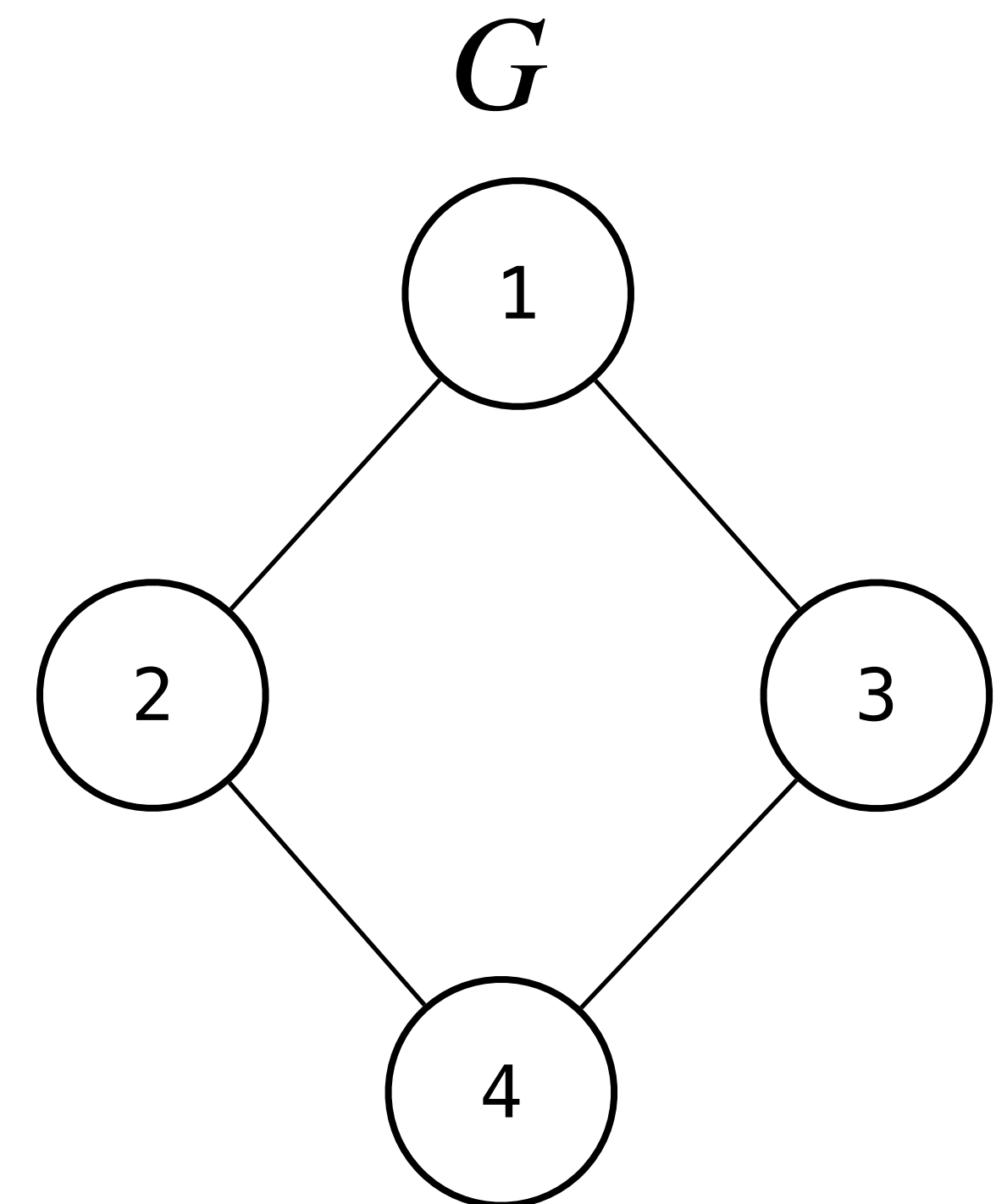


# Example

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$

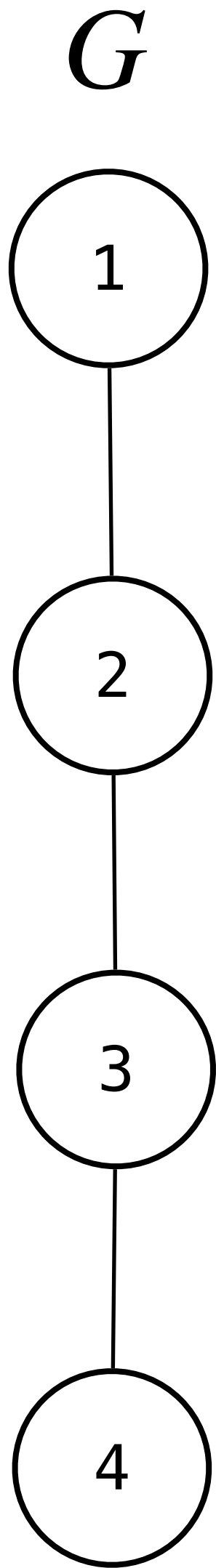
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 4 & 4 & 0 \\ 4 & 0 & 0 & 4 \\ 4 & 0 & 0 & 4 \\ 0 & 4 & 4 & 0 \end{bmatrix}$$



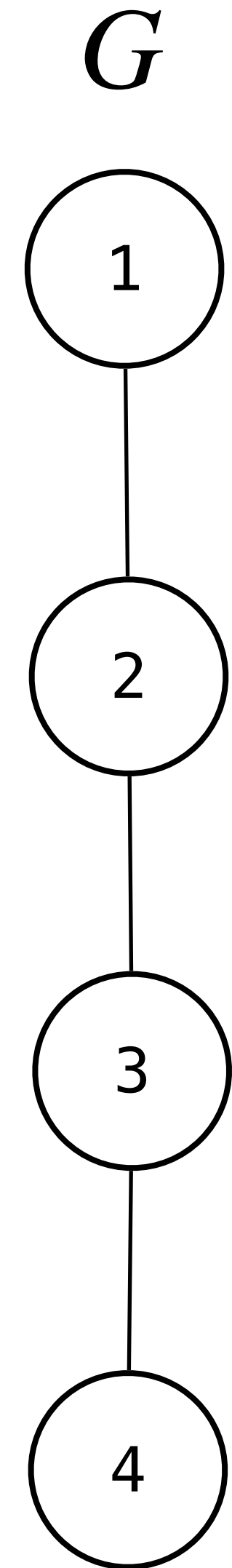


# Example



# Example

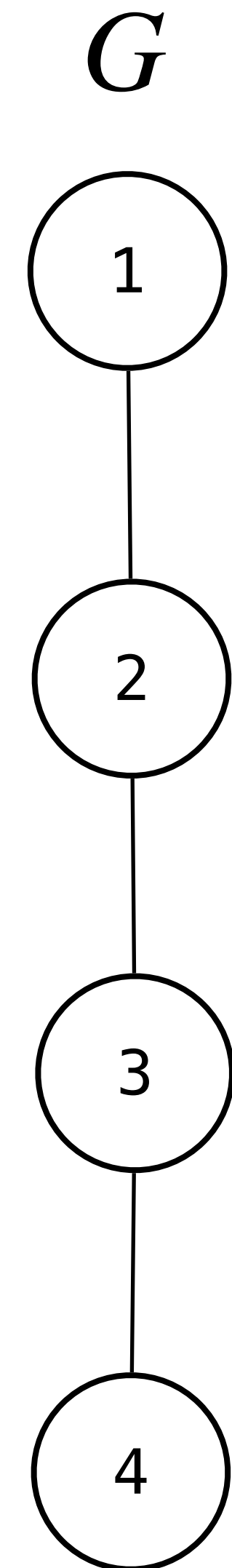
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$



# Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

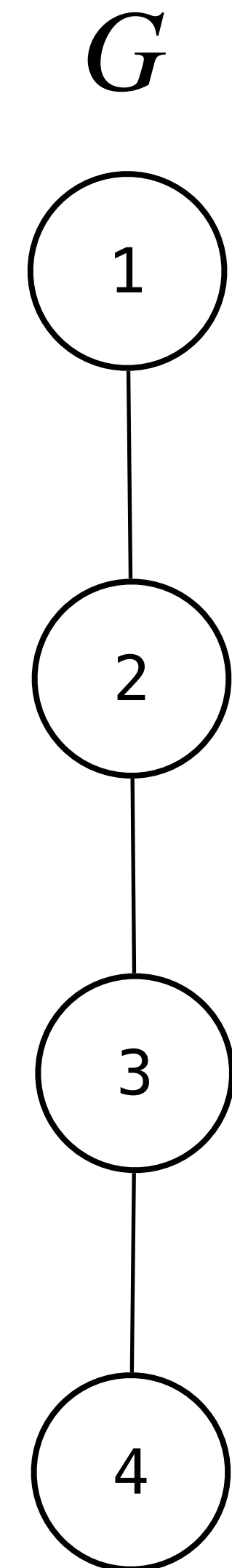


# Example

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \text{adjacency matrix for } G$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 1 \\ 2 & 0 & 3 & 0 \\ 0 & 3 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

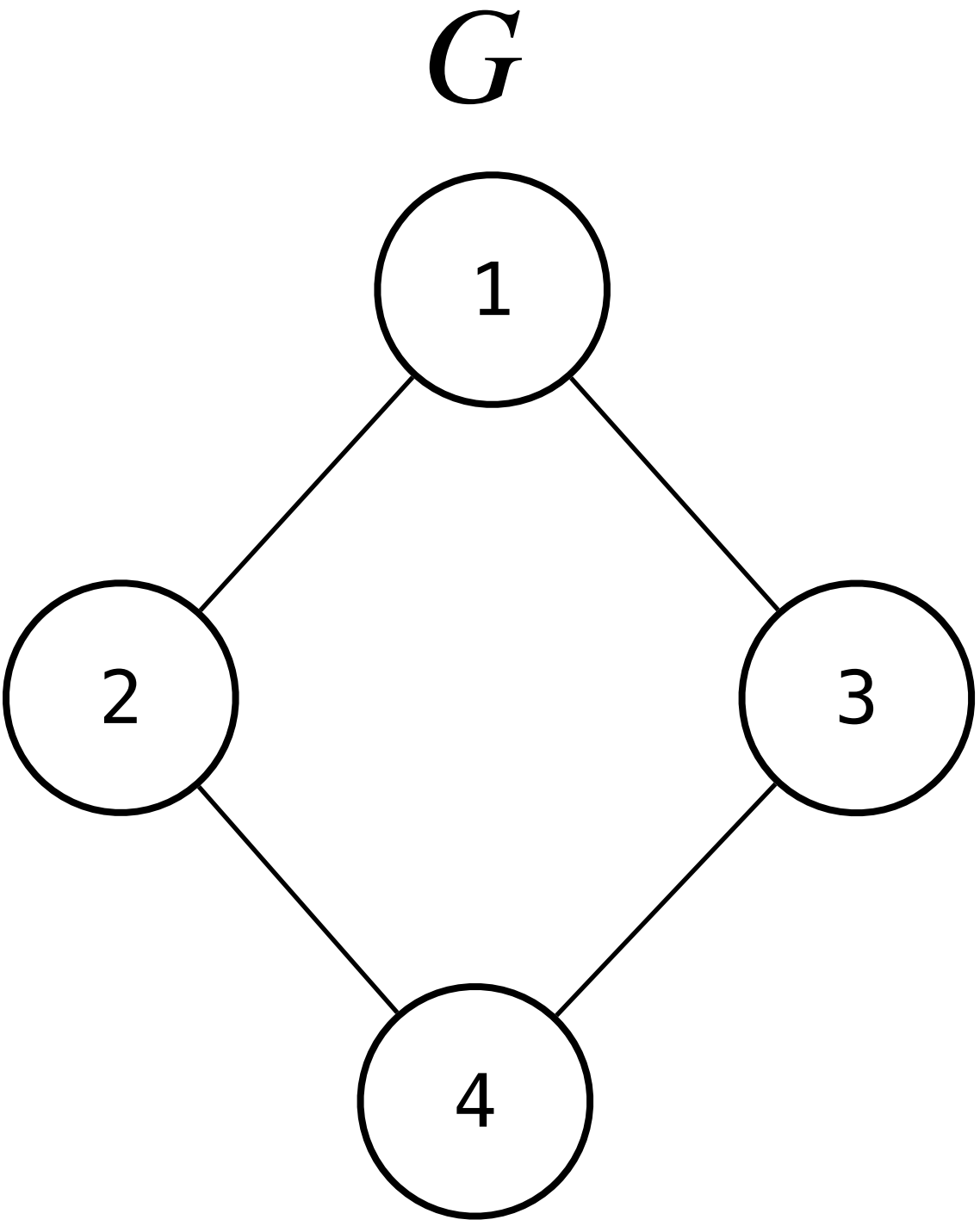


# Another Application: Reachability

**Theorem:** Let  $G$  be a simple graph.

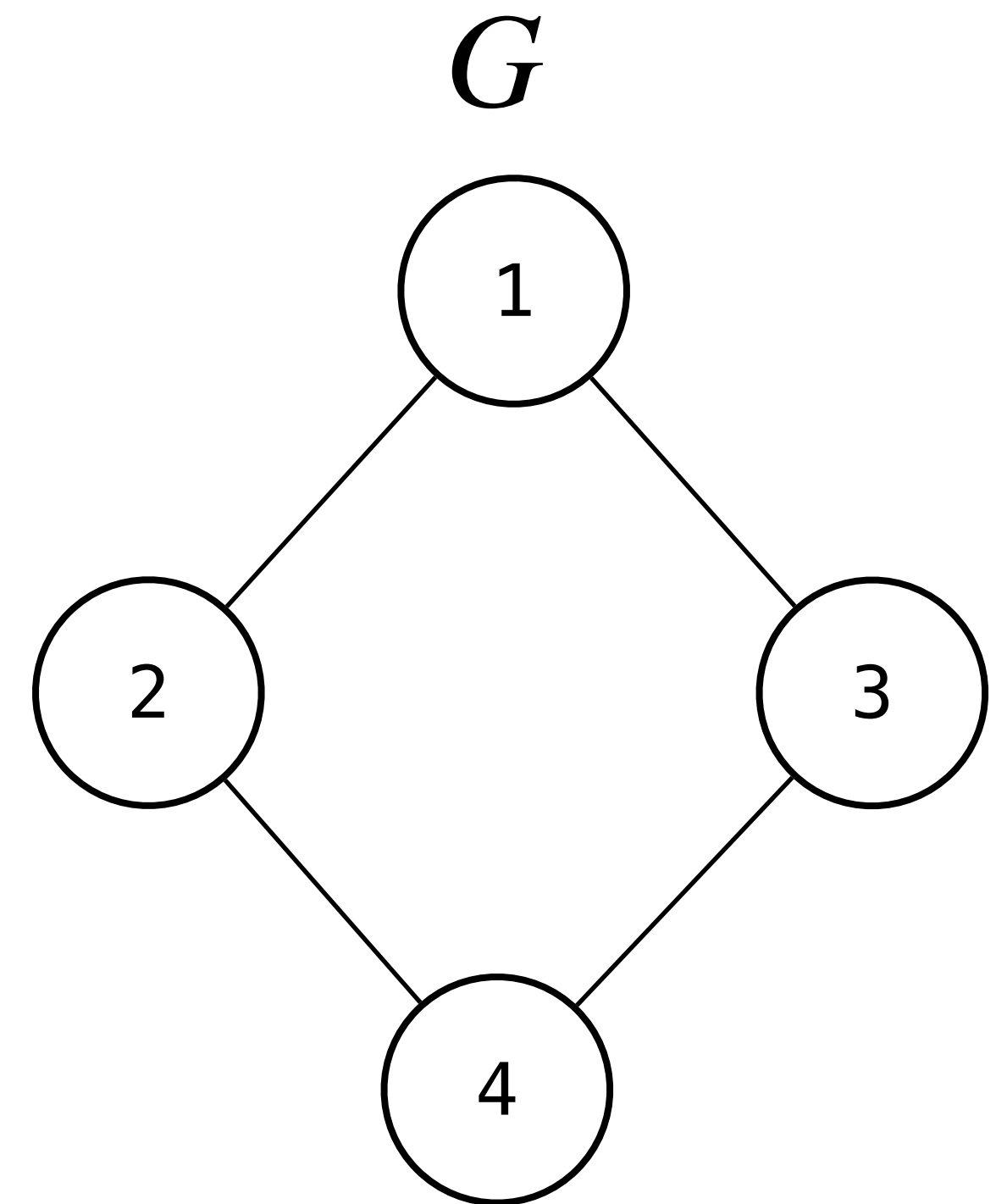
- $(A_G^k)_{ij}$  is the number of paths of length **exactly**  $k$  from  $v_i$  to  $v_j$ .
- $((A_G + I)^k)_{ij}$  is nonzero if and only if there is a path of length at **at most**  $k$  from  $v_i$  to  $v_j$ .

# Example



# Example

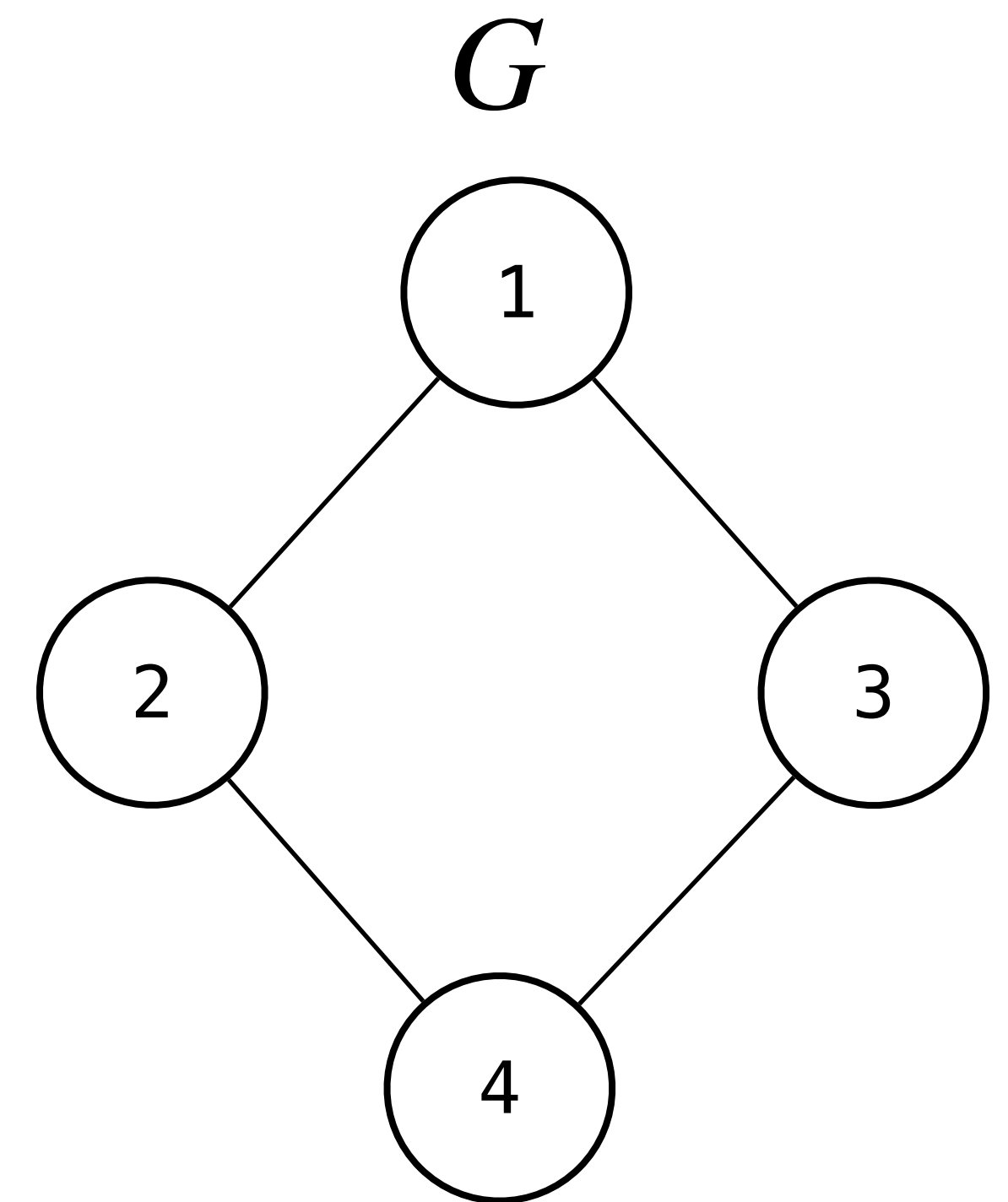
$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = (\text{adjacency matrix for } G) + I$$



# Example

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = (\text{adjacency matrix for } G) + I$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}^2 = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix}$$



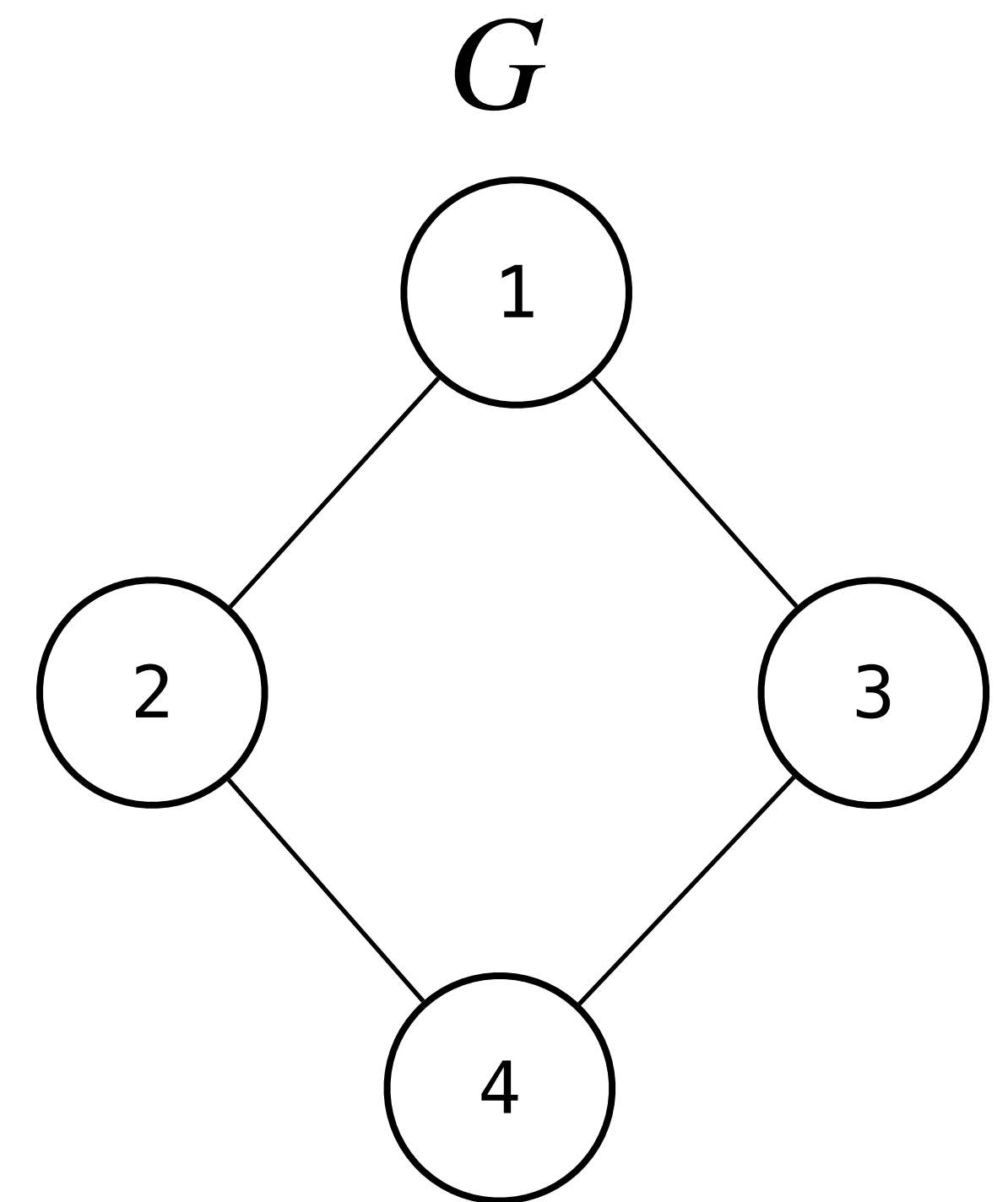


# Example

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = (\text{adjacency matrix for } G) + I$$

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}^2 = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^3 = \begin{bmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 6 & 6 & 7 \\ 6 & 6 & 7 & 6 \\ 6 & 7 & 6 & 6 \\ 7 & 6 & 6 & 6 \end{bmatrix}$$



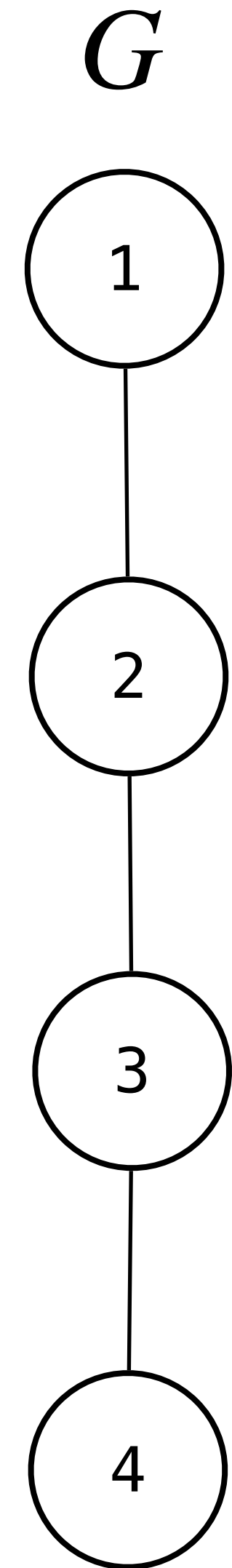
# How To: Reachability

**Question:** Given a simple graph  $G$  determine how many nodes  $v_i$  can reach in at least  $k$  steps.

**Answer:** Find  $(A_G + I)^k$  and count the number of nonzero elements in column  $i$ .

# Question

*Determine the  $(A_G + I)^2$  and  $(A_G + I)^3$  and interpret the results.*



# Summary

Matrix inverses allow us to easily solve many matrixes equations over the same  $A$

LU Factorizations allows us to do the same, but more generally more efficiently

Adjacency matrices are linear algebraic representations of graphs