LU Factorization

Geometric Algorithms
Lecture 12

CAS CS 132

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

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

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bd} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

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

$$A\vec{x} = \vec{e}_1$$

$$A\vec{x} = e_2$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 3^{-3} & 7^{-6} & 0^{-3} \end{bmatrix} \sim \begin{bmatrix} 1 & \overset{-2}{2} & \overset{+6}{1} \\ 0 & 1 & -3 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 7 \\ 0 & 1 & -3 \end{bmatrix}$$

$$\vec{x} = \boxed{\begin{bmatrix} 7 \\ -3 \end{bmatrix}}$$

$$\begin{bmatrix} 1 & 2 & 0 \\ 3^{-3} & 7^{-6} & 1^{-0} \end{bmatrix} \sim \begin{bmatrix} 1 & \overset{-2}{2} & \overset{-2}{0} \\ 0 & 1 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \end{bmatrix}$$

Answer

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

$$\left[\begin{array}{cccc} 1 & 2 & 1 & 0 \\ \overset{-3}{3} & \overset{-6}{7} & \overset{-3}{0} & \overset{-0}{1} \end{array}\right] \sim \begin{array}{c} \begin{array}{cccc} & -2 & \times 6 & -2 \end{array} \\ \left[\begin{array}{cccc} 1 & 2 & 1 & 0 \\ 0 & 1 & -3 & 1 \end{array}\right] \end{array} \sim \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \boxed{\begin{array}{cc} 7 & -2 \\ -3 & 1 \end{array}}\right]$$

$$A\vec{x} = \begin{bmatrix} 3 \\ 10 \end{bmatrix} \qquad \left[\begin{array}{ccc} 1 & 2 & 3 \\ \overset{-3}{3} & \overset{-6}{7} & \overset{-9}{10} \end{array}\right] \sim \begin{array}{c} \begin{array}{ccc} & -2 & -2 \end{array} \\ \left[\begin{array}{ccc} 1 & 2 & 3 \\ 0 & 1 & 1 \end{array}\right] \end{array} \sim \left[\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \end{array}\right]$$

$$\cancel{A^{-1}}\cancel{A}\vec{x} = A^{-1}\vec{b}$$

$$\vec{x} = \boxed{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}$$

$$\vec{x} = A^{-1}\vec{b}$$

$$x = \begin{bmatrix} 7 & -2 \\ -3 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 10 \end{bmatrix} = \begin{bmatrix} 21 & -20 \\ -9 & +10 \end{bmatrix} = \boxed{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}$$

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

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

In other words: we want to solve <u>several</u> 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

echelon form of A

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

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`*

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

The Fundamental Question

$$A = L\boxed{U}$$ echelon form of A

The Fundamental Question

$$A = L\boxed{U}$$ echelon form of A

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

The Fundamental Question

$$A = L\boxed{U}$$ 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

$EA = U$

$A = E^{-1}U$

$$A = L\boxed{U}$$ 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 (in reverse)

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:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 20 & 25 & 30 \\ 7 & 8 & 9 \end{bmatrix}$$

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

$R_3 \leftarrow (R_3 - 2R_1)$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \mapsto \begin{bmatrix} x_1 \\ x_2 \\ x_3 - 2x_1 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \mapsto \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix}$$

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 then 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.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 + 3R_1} \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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

<u>Taking stock:</u>

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,...

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 3×3 *matrix.*

Answer

$$R_2 \leftarrow 3R_2$$

$$R_1 \leftarrow R_1 + R_2$$

$$R_2 \leftrightarrow R_3$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 3 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & \underline{1} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0 & 1 \\ 0 & 3 & 0 \end{bmatrix}$$

check:

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

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

$$(E_1E_2E_3E_4)^{-1} = E_4^{-1}E_3^{-1}E_2^{-1}E_1^{-1}$$

!! 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

Echelon form of A

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

Gaussian Elimination and Elementary Matrices

$$A \sim A_1 \sim A_2 \sim \ldots \sim \boxed{A_k}$$

echelon form

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

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

Gaussian Elimination and Elementary Matrices

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

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

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

where E implements a <u>sequence</u> of row operations. So:

Gaussian Elimination and Elementary Matrices

$$A \sim E_1A \sim E_2E_1A \sim \ldots \sim E_kE_{k-1}\ldots E_2E_1A$$

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

Invertible

$$B = \boxed{(E_kE_{k-1}\ldots E_2E_1)}A = EA$$

where E implements a <u>sequence</u> of row operations. So:

Gaussian Elimination and Elementary Matrices

$$A \sim E_1A \sim E_2E_1A \sim \ldots \sim E_kE_{k-1}\ldots E_2E_1A$$

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

Invertible

$$B = (E_kE_{k-1}\ldots E_2E_1)A = EA$$

where E implements a sequence of row operations. So:

L

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

U

LU Factorization Algorithm

LU Factorization Algorithm

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

LU Factorization Algorithm

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

LU Factorization Algorithm

```
FUNCTION LU_Factorization(A):
    L ← identity matrix
    U ← A
```

LU Factorization Algorithm

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

LU Factorization Algorithm

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

LU Factorization Algorithm

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

LU Factorization Algorithm

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

LU Factorization Algorithm

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

LU Factorization Algorithm

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

we'll see how to do this more efficiently

The forward part of Gaussian elimination <u>is</u> matrix factorization

The "L" Part

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

This a product of elementary matrices

So $L = E^{-1} = E_1^{-1} E_2^{-1} \ldots E_{k-1}^{-1} E_k^{-1}$ **!! 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 **FL**oating–oint **OP**erations (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 **FL**oating-oint **OP**erations (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

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

Dominant Terms

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

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

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

For polynomials, they are equivalent to their dominant term

Dominant Terms

$$\lim_{i \to \infty} \frac{3i^2}{4i^2} \neq 1$$

$$= 3/4$$

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

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

$3x^3$ 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: $(LU)\vec{x} = \vec{b}$ $\quad L(U\vec{x}) = \vec{b}$ $\quad A\vec{d} =$

$LU\vec{d} =$

$L\vec{y} = \vec{b} \Rightarrow y = \vec{c}$ $\quad U\vec{x} = \vec{c}$ $\quad x = \vec{d}$ $\quad L\vec{c} = \vec{b}$

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

<u>The following FLOP estimates are based on $n \times n$ matrices</u>

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

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

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

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

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

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

If you solve several matrix equations for the same matrix, LU factorization is <u>faster</u> 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

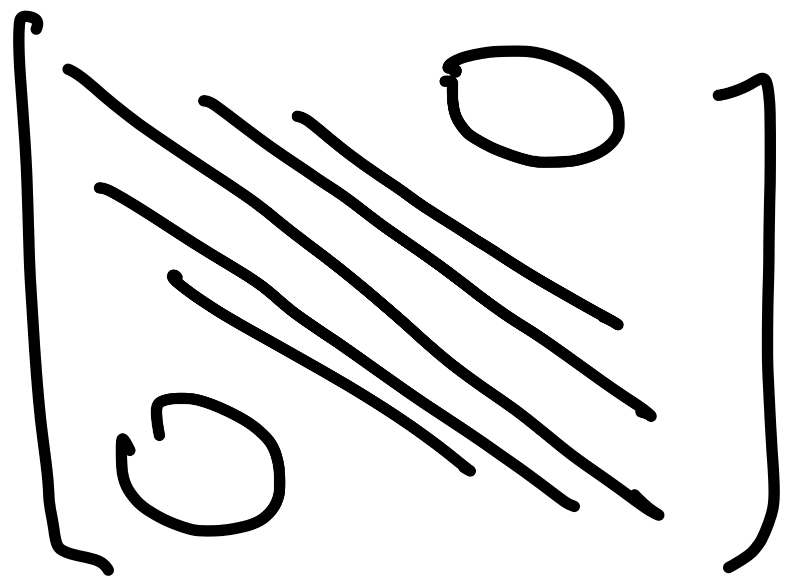
If A doesn't have to many entries (A is **sparse)**, then its likely that L and U won't either.

Other Considerations: Density

If A doesn't have to many entries (A is **sparse)**, then its 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 to many entries (A is **sparse)**, then its likely that L and U won't either.

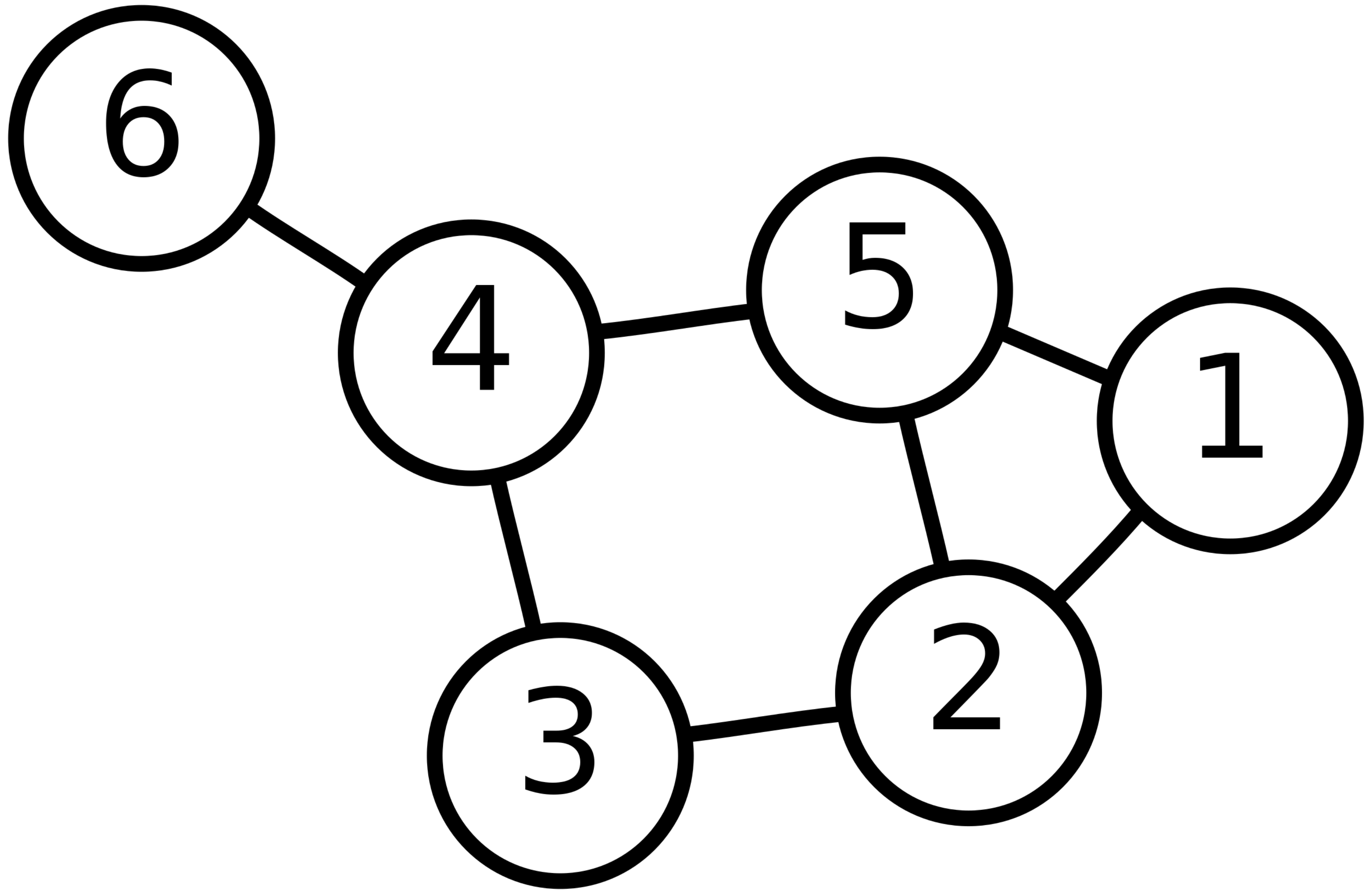
But A^{-1} may have ***many*** entries (A^{-1} is **dense**)

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

Algebraic Graph Theory

Graphs

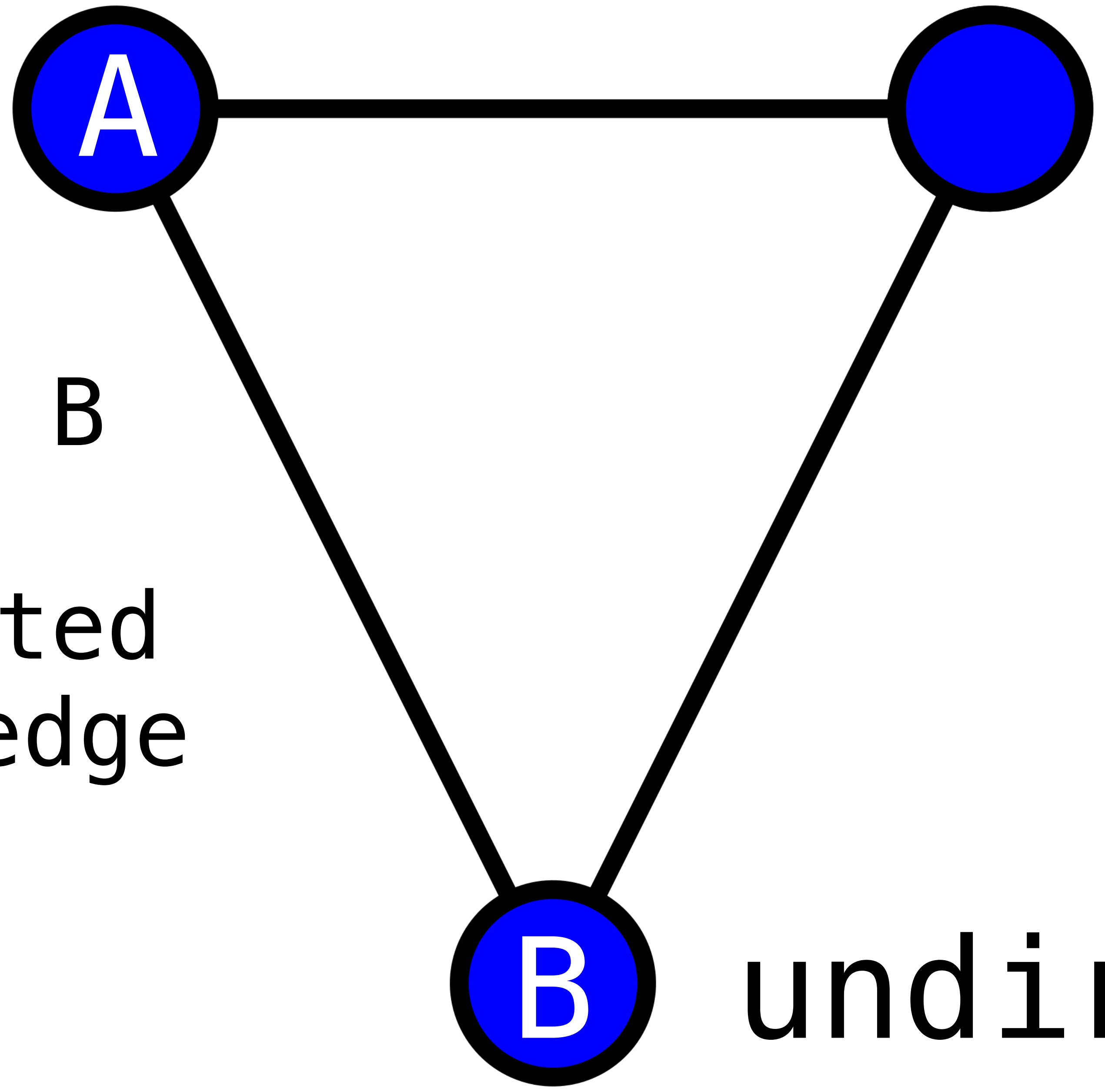
Definition (Informal). A **graph** is a collection of nodes with edges between them.



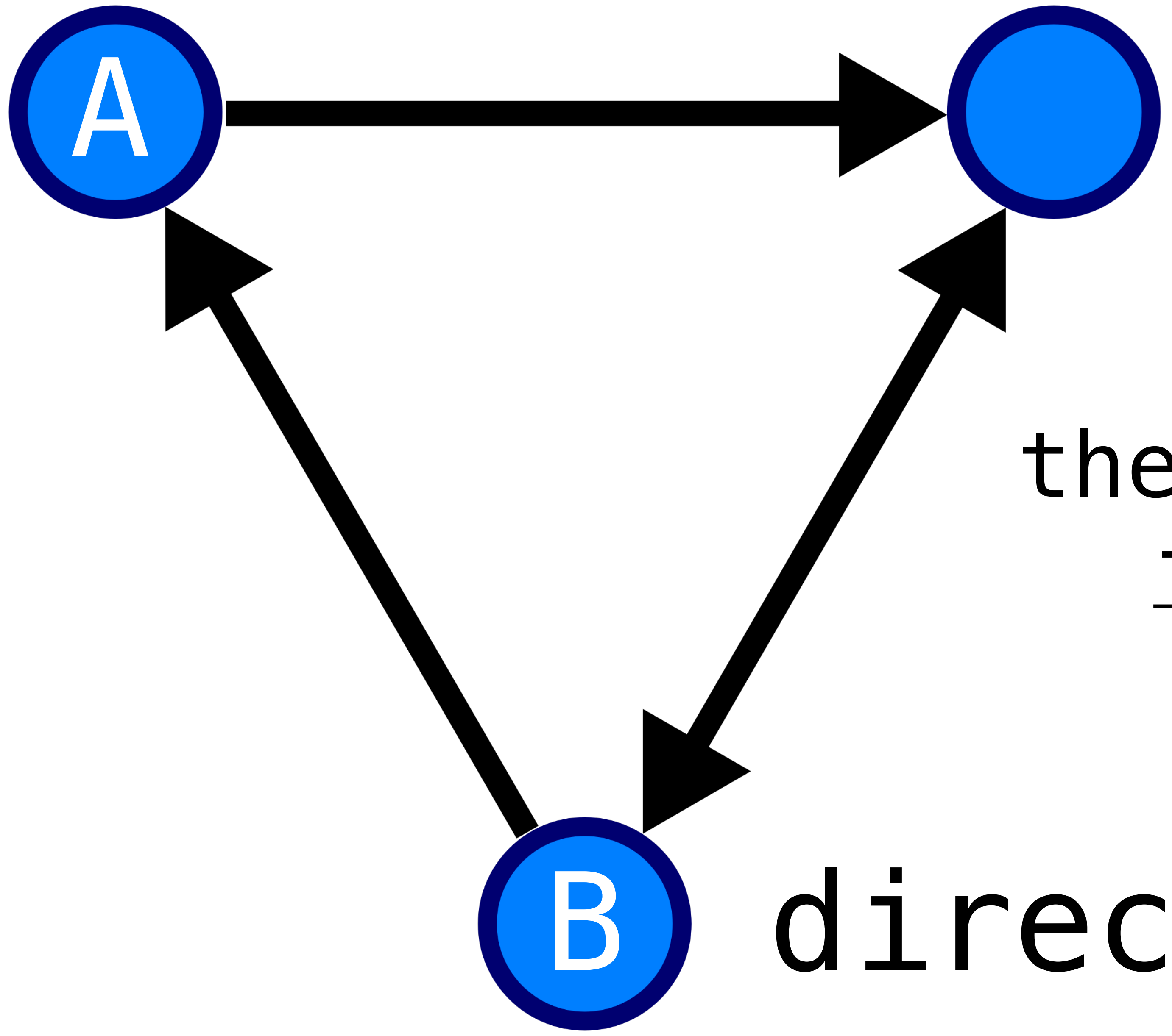
Directed vs. Undirected Graphs

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

A and B are connected by an edge



undirected

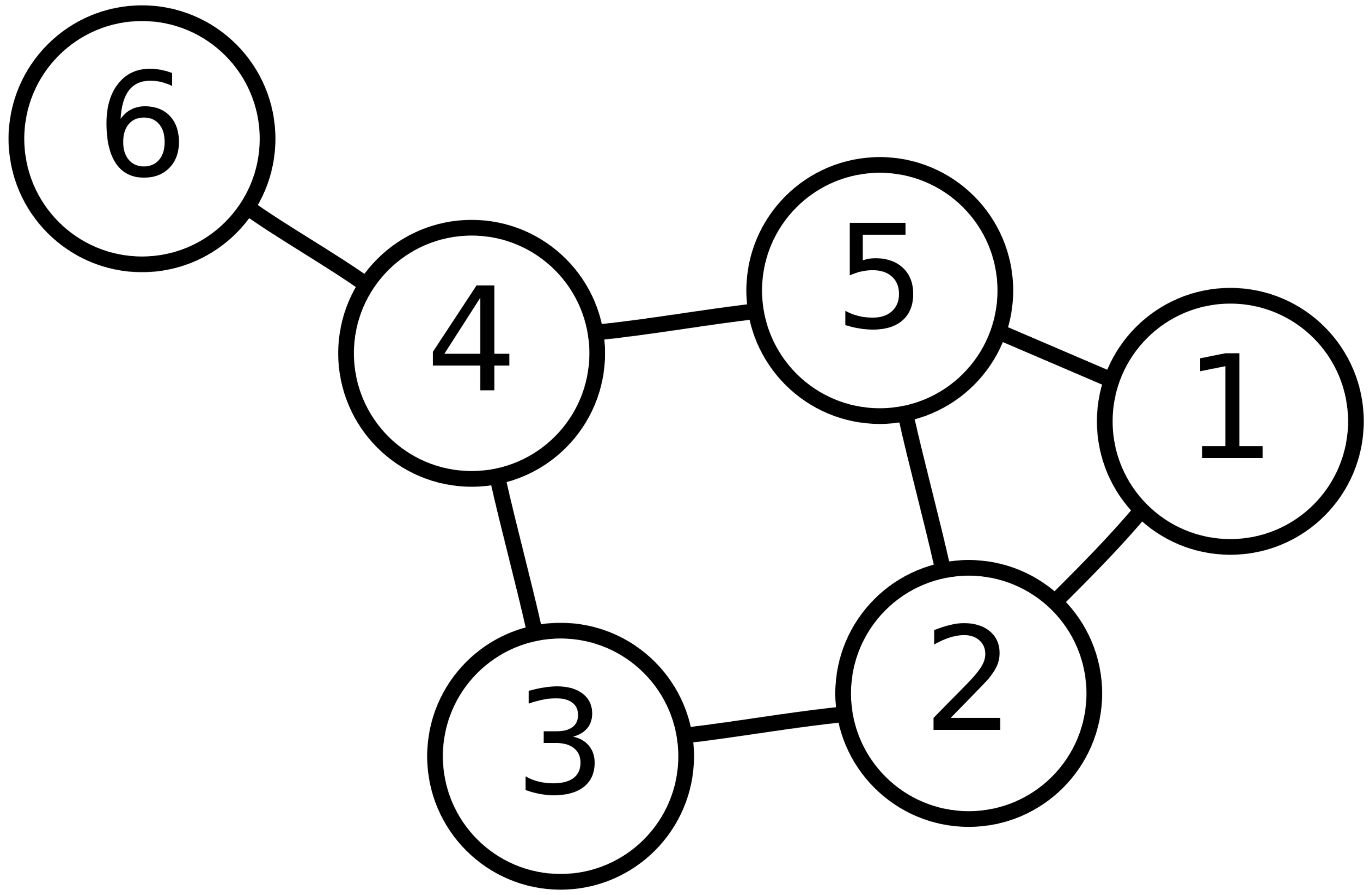


there is an edge from B to A

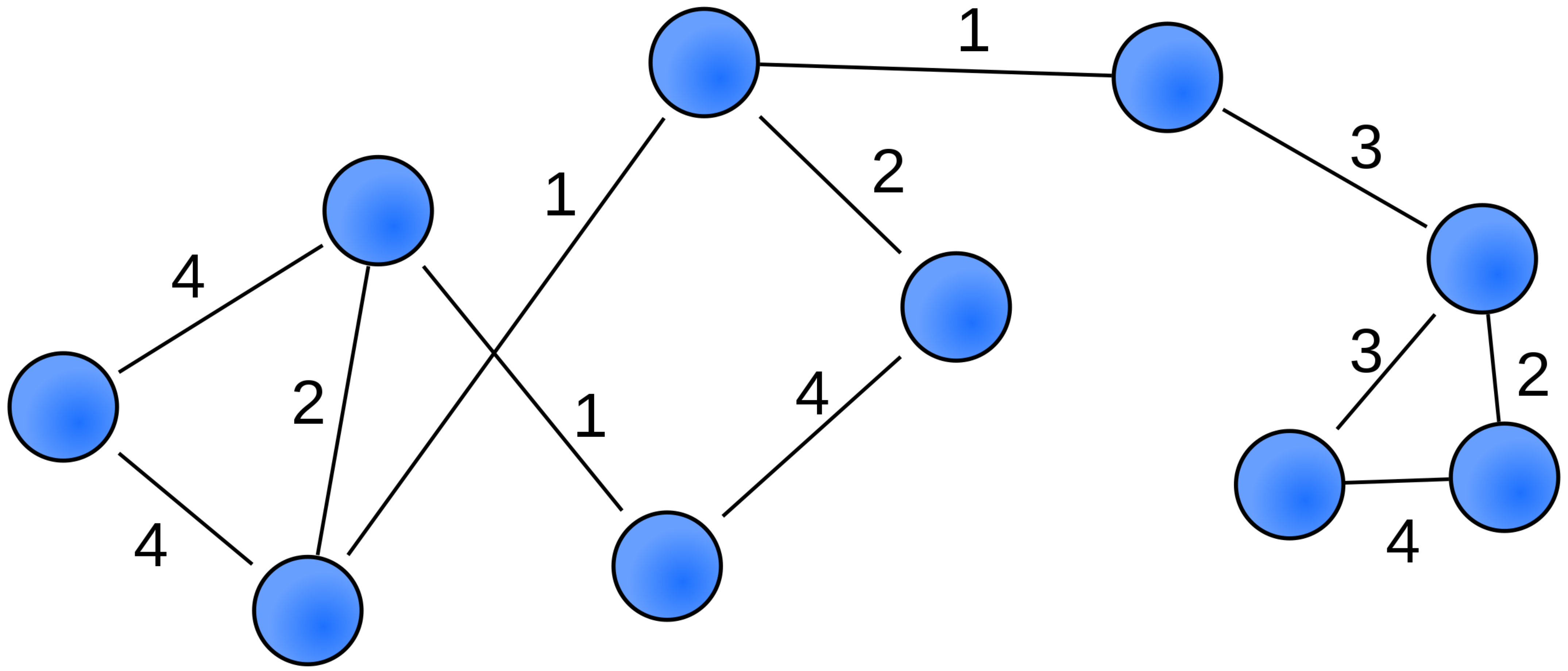
directed

Weighted vs Unweighted graphs

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



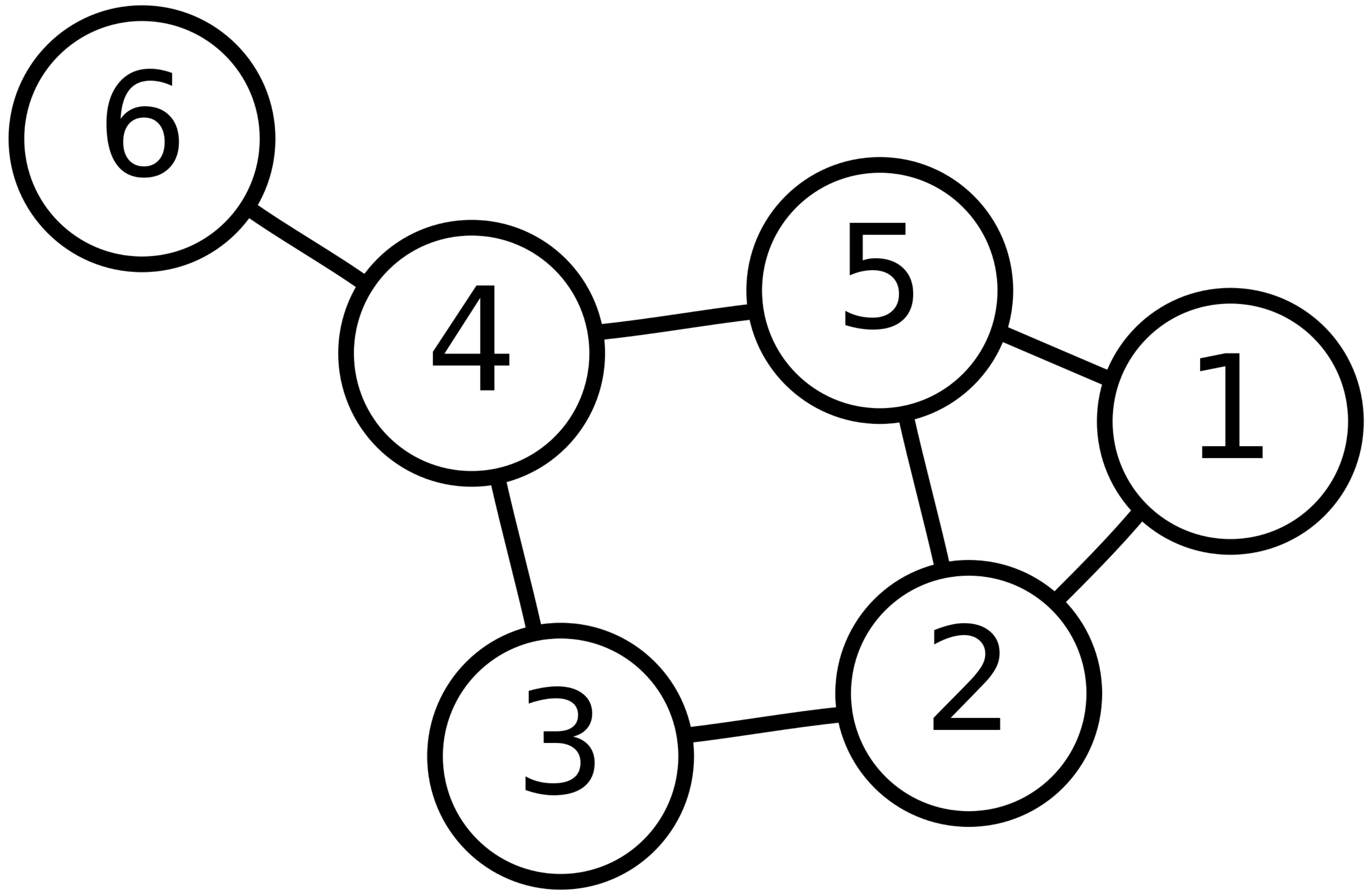
unweighted



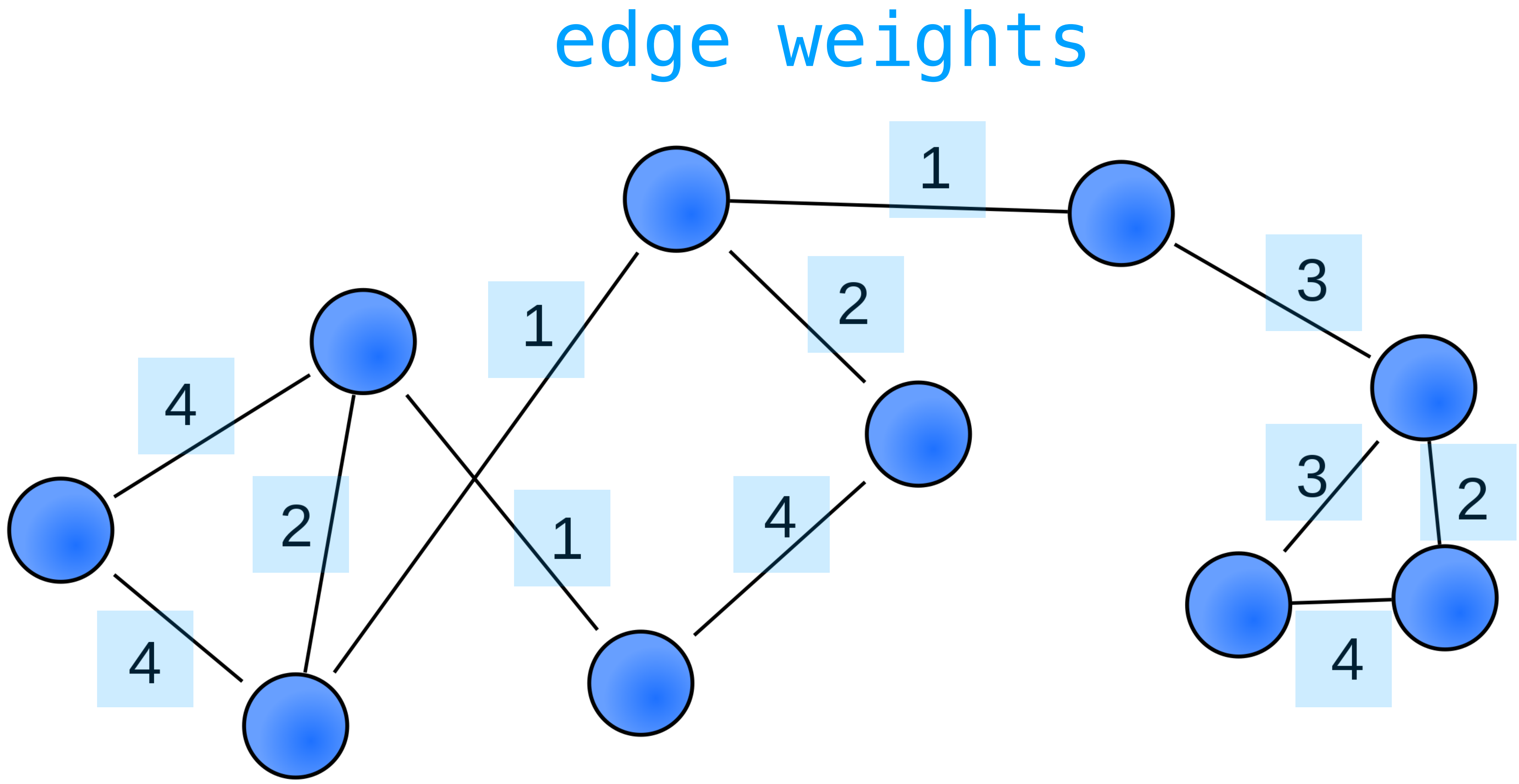
weighted

Weighted vs Unweighted graphs

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



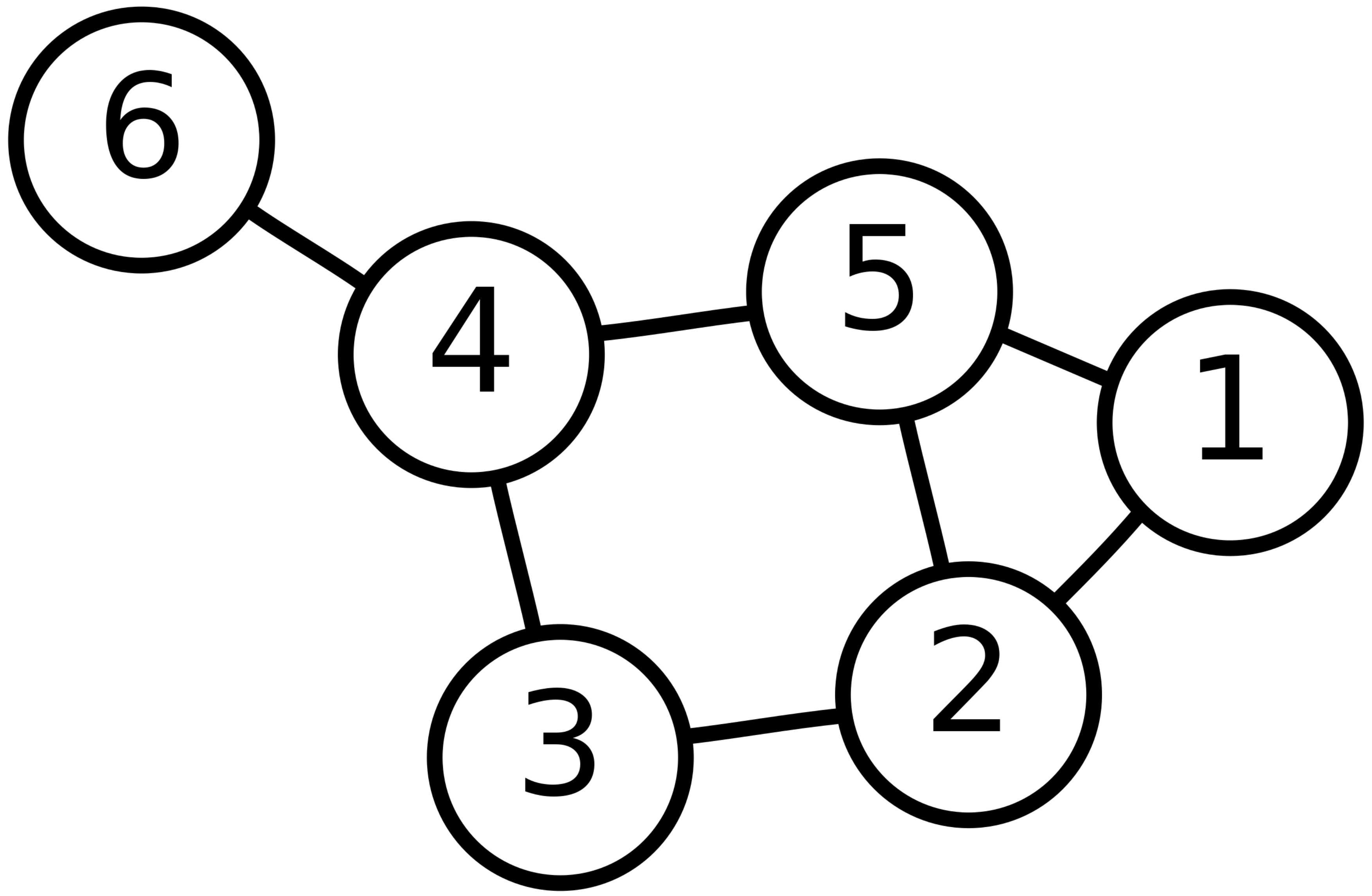
unweighted



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 <u>matrices</u>.

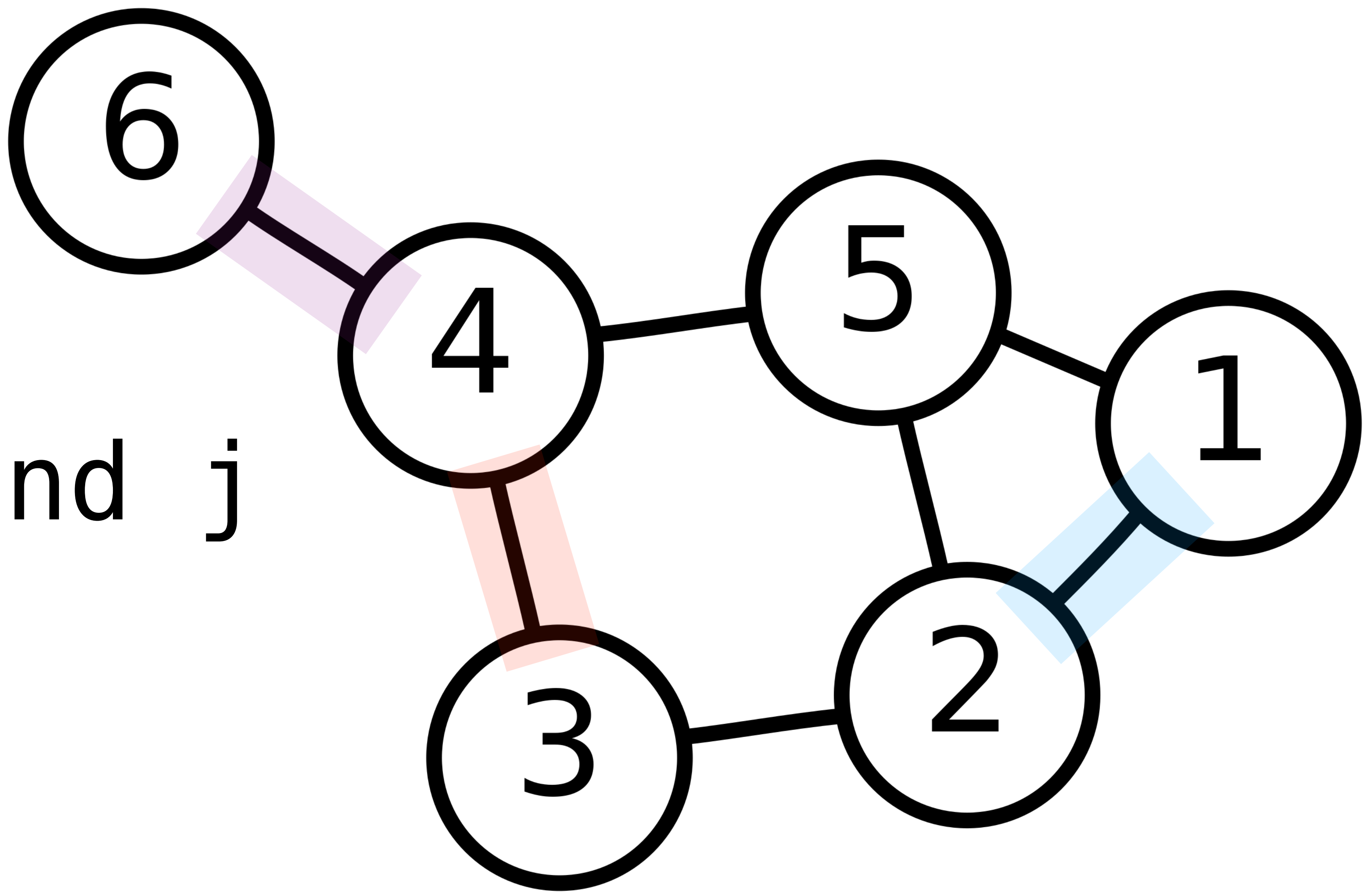
Adjacency Matrices

$$
\begin{array}{c c}
 & \begin{array}{cccccc} & A_{12} & & A_{34} & & A_{46} \end{array} \\
\begin{array}{c} \\ A_{21} \\ \\ A_{43} \\ \\ A_{64} \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\end{array}
$$

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.

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



Symmetric Matrices

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

$$A^T = A$$

Example.

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

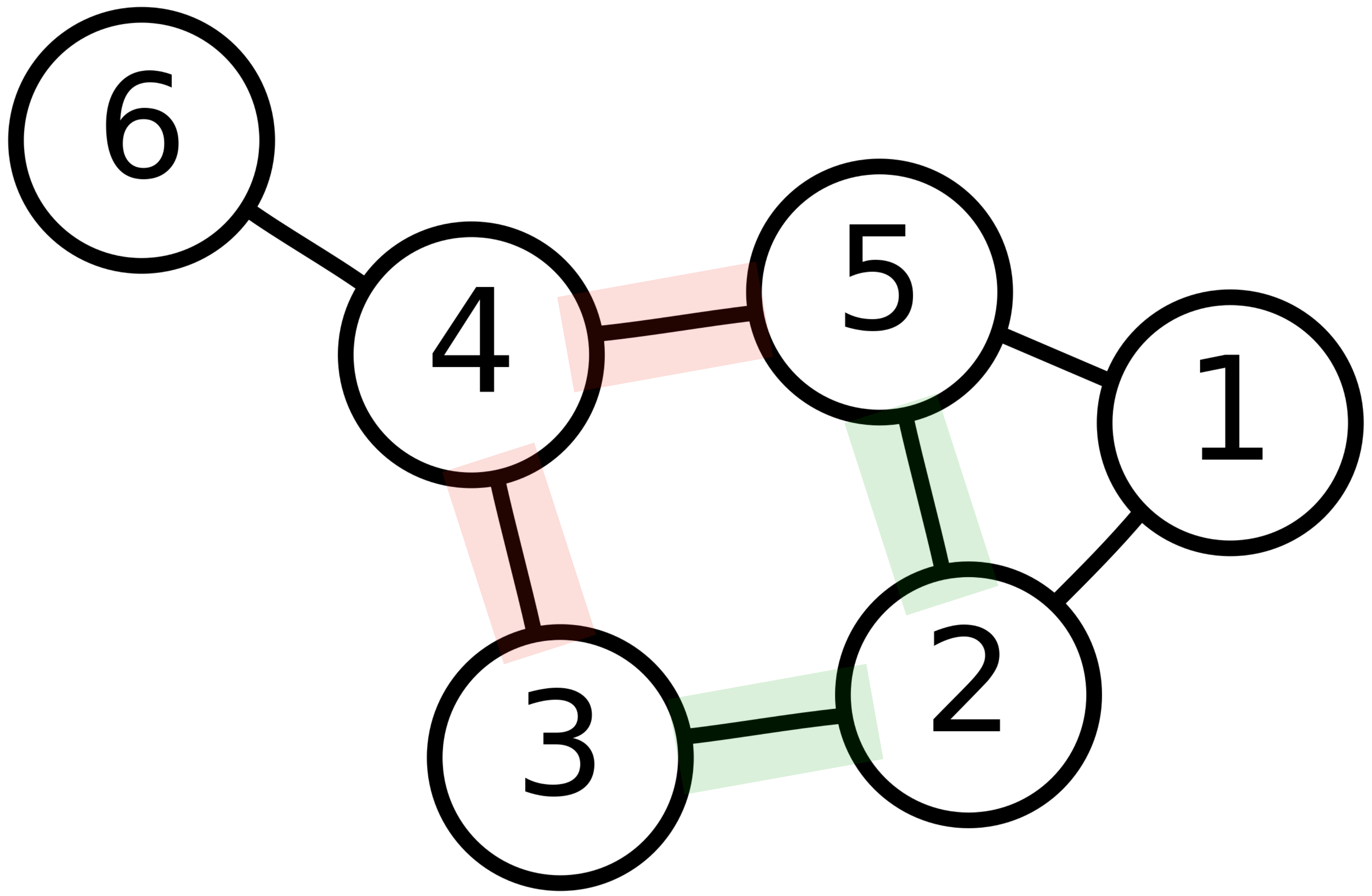
Once we have an adjacency matrix, we can do linear algebra on graphs.

Example: Squared Adjacency Matrices

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

Example: Squared Adjacency Matrices

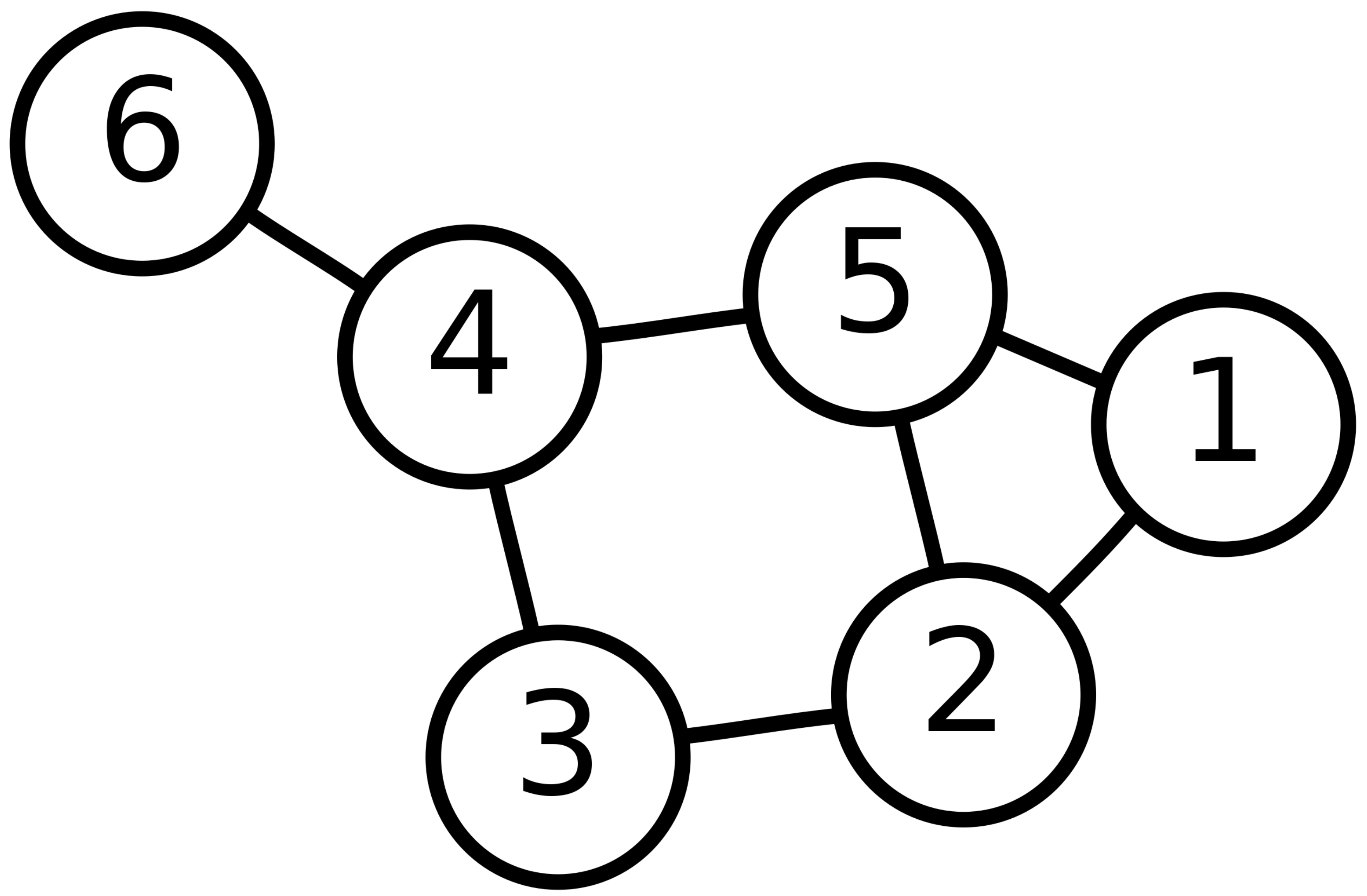
$$\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1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



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

Example: Squared Adjacency Matrices

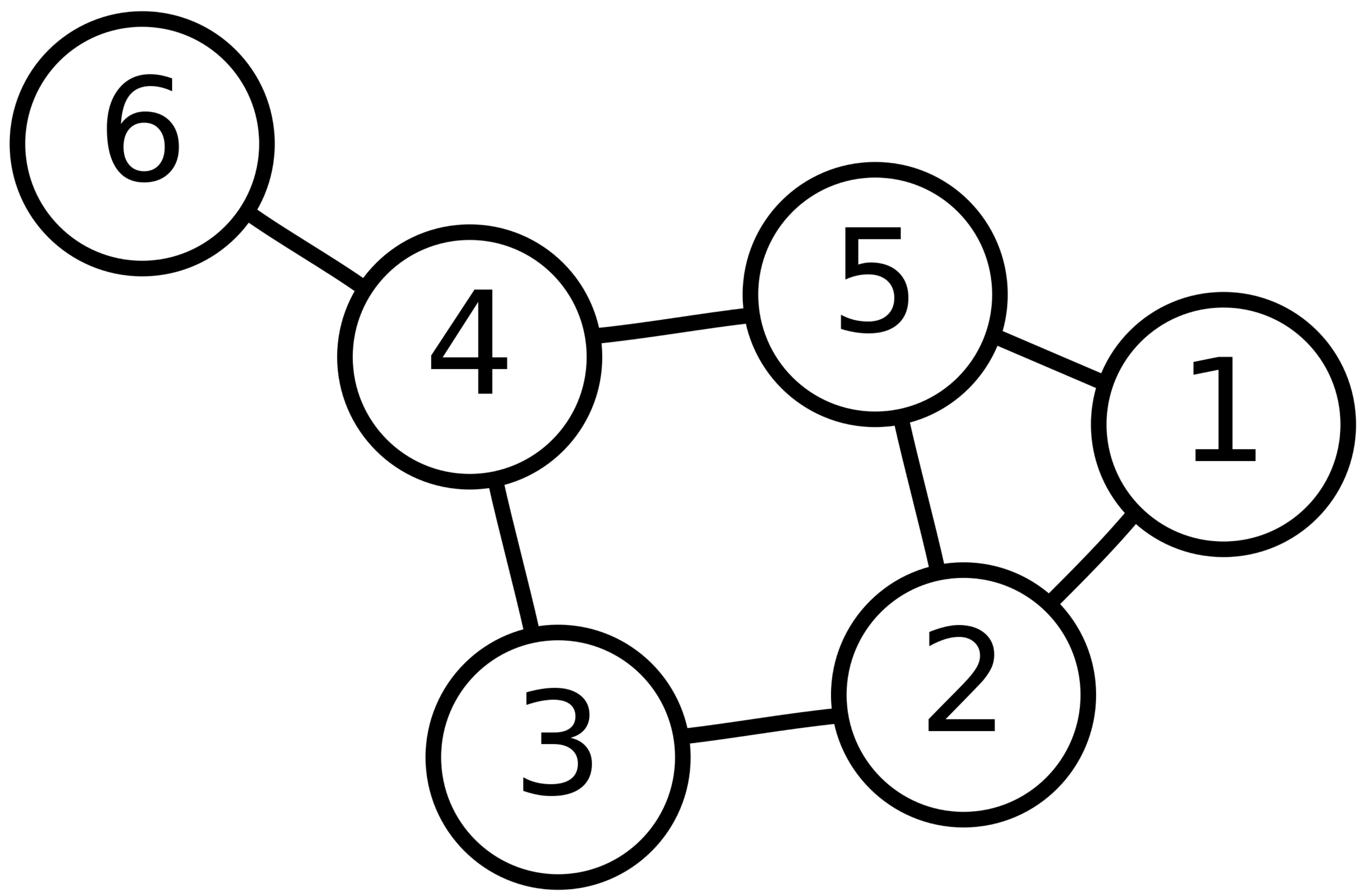
$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \ldots + A_{in}A_{nj}$$



Example: Squared Adjacency Matrices

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

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



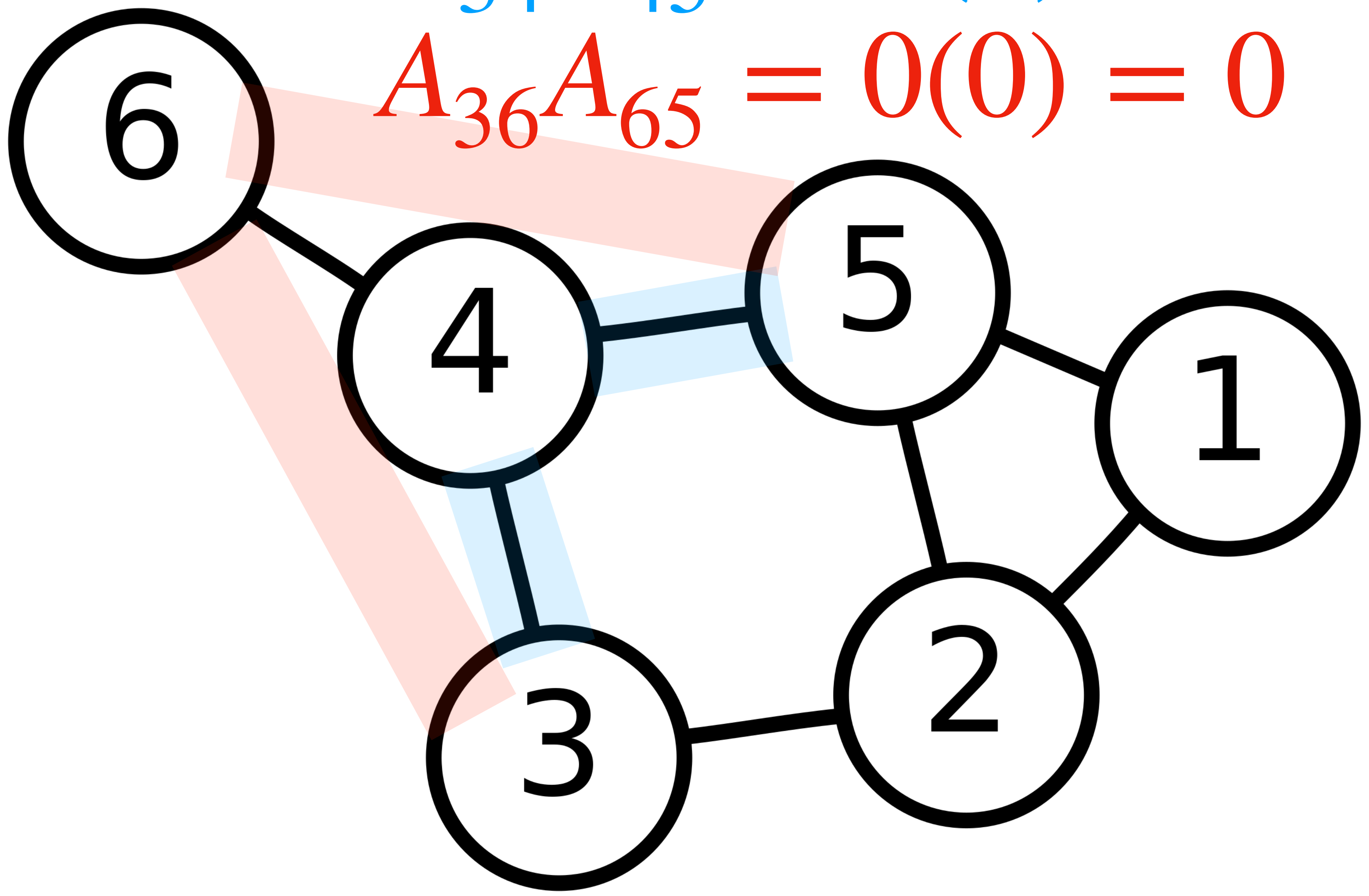
Example: Squared Adjacency Matrices

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

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

$A_{34}A_{45} = 1(1) = 1$

$A_{36}A_{65} = 0(0) = 0$



Example: Squared Adjacency Matrices

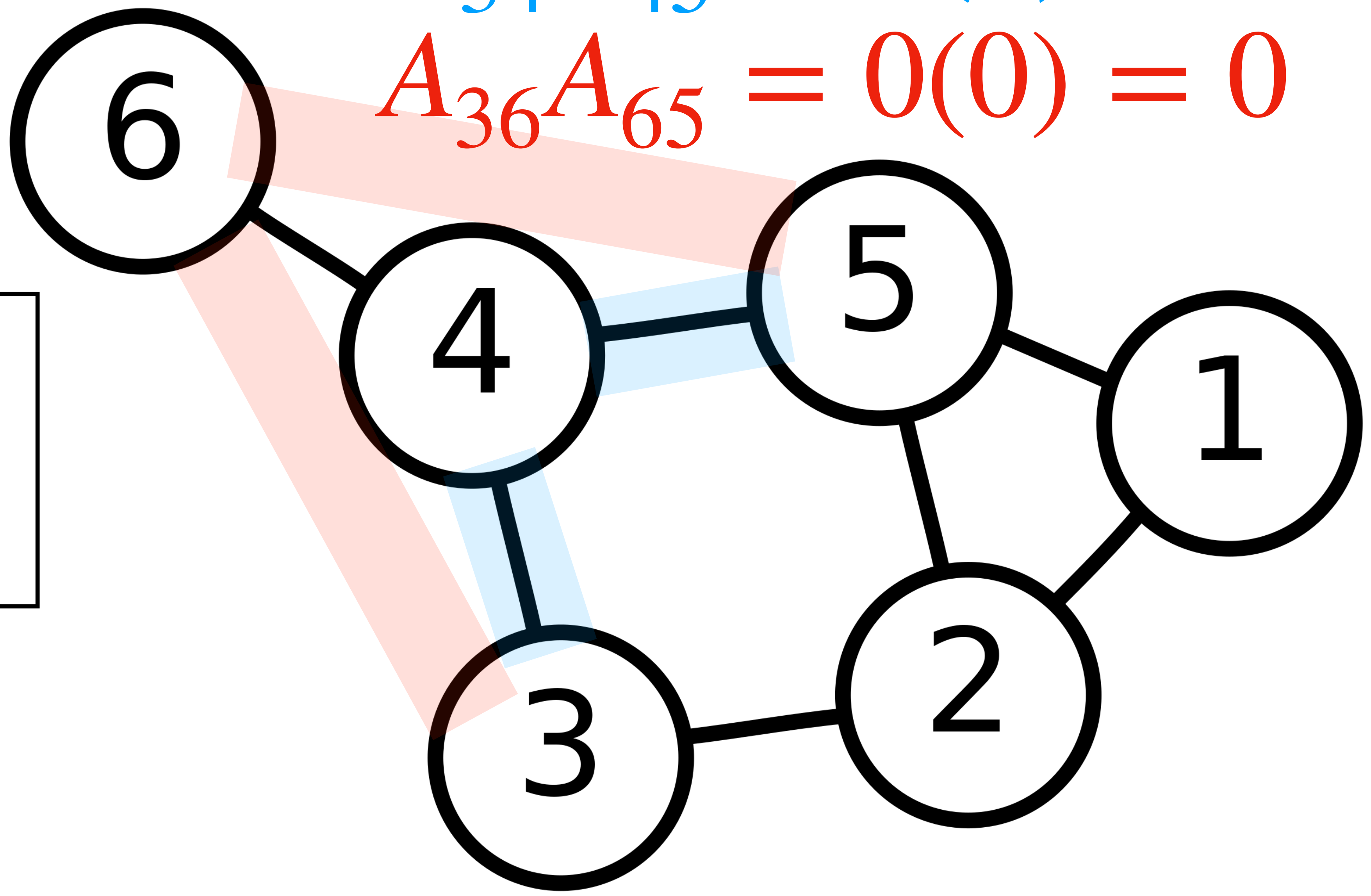
$$(A^2)_{ij} = A_{i1}A_{1j} + A_{i2}A_{2j} + \ldots + A_{in}A_{nj}$$

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

$$A_{34}A_{45} = 1(1) = 1$$

$$A_{36}A_{65} = 0(0) = 0$$

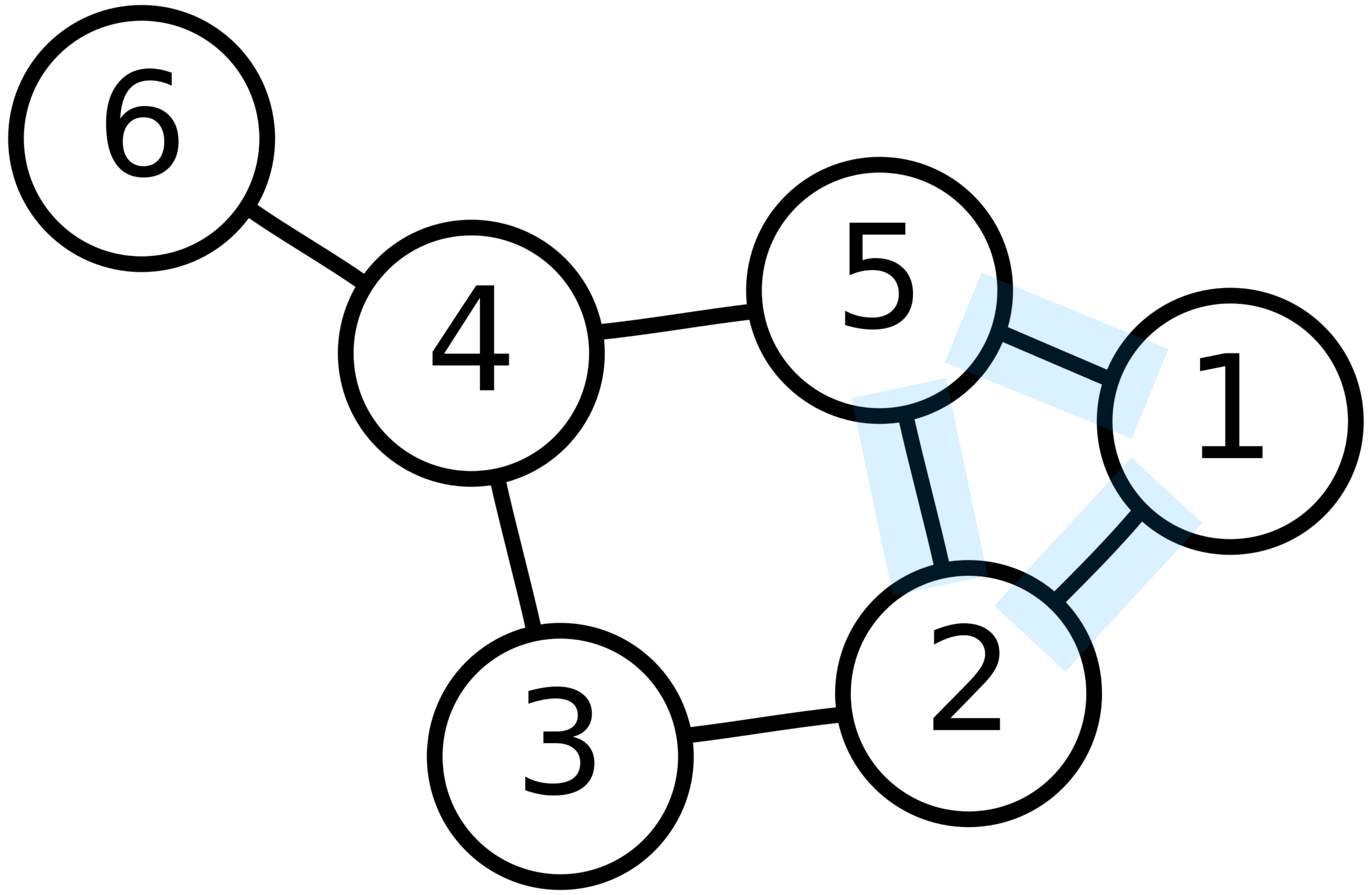
$$(A^2)_{ij} = \boxed{\begin{array}{c}\textbf{number of 2-step paths} \\ \textbf{from i to j}\end{array}}$$



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 ← 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 ← 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
```

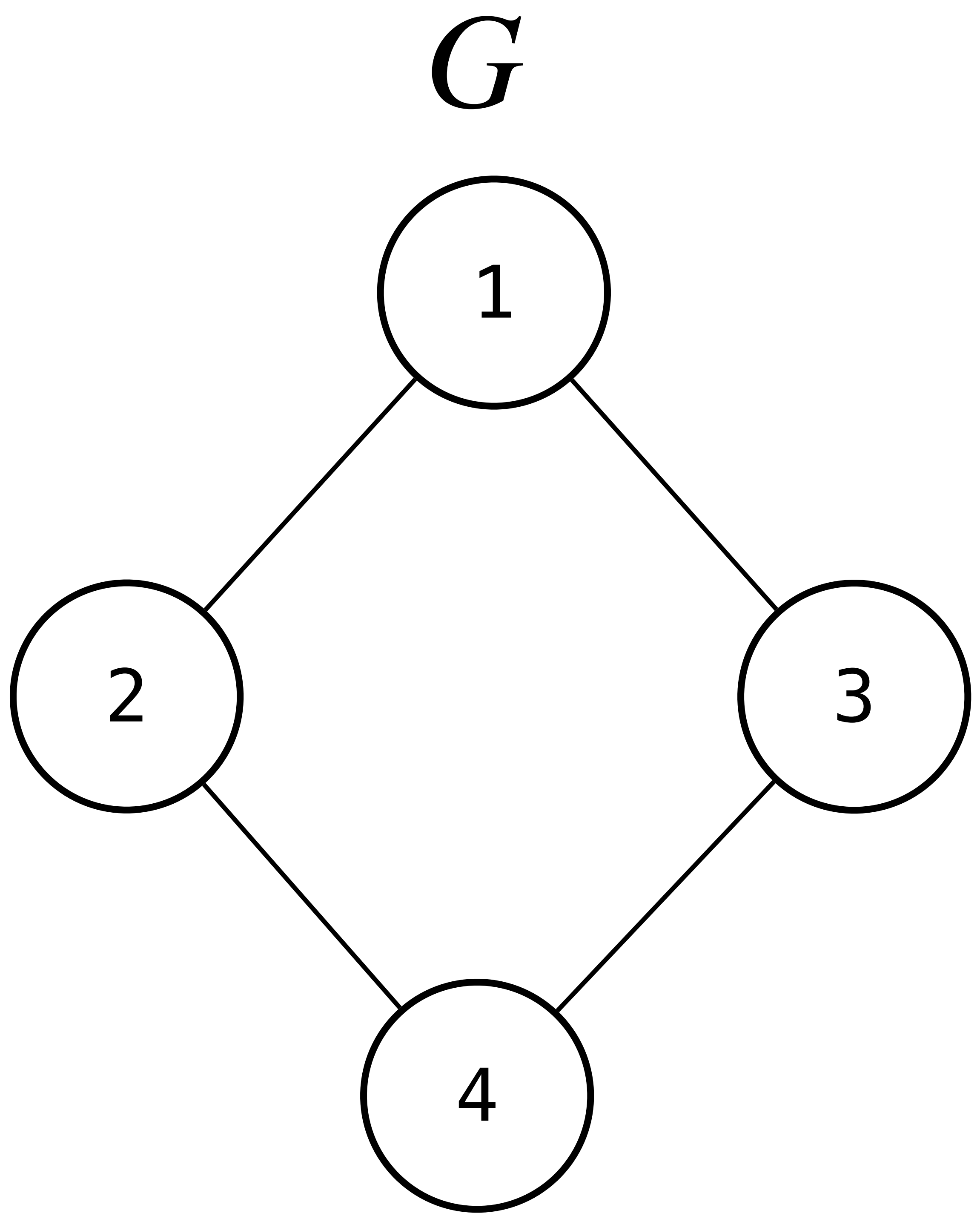
demo

Another Application: Reachability

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

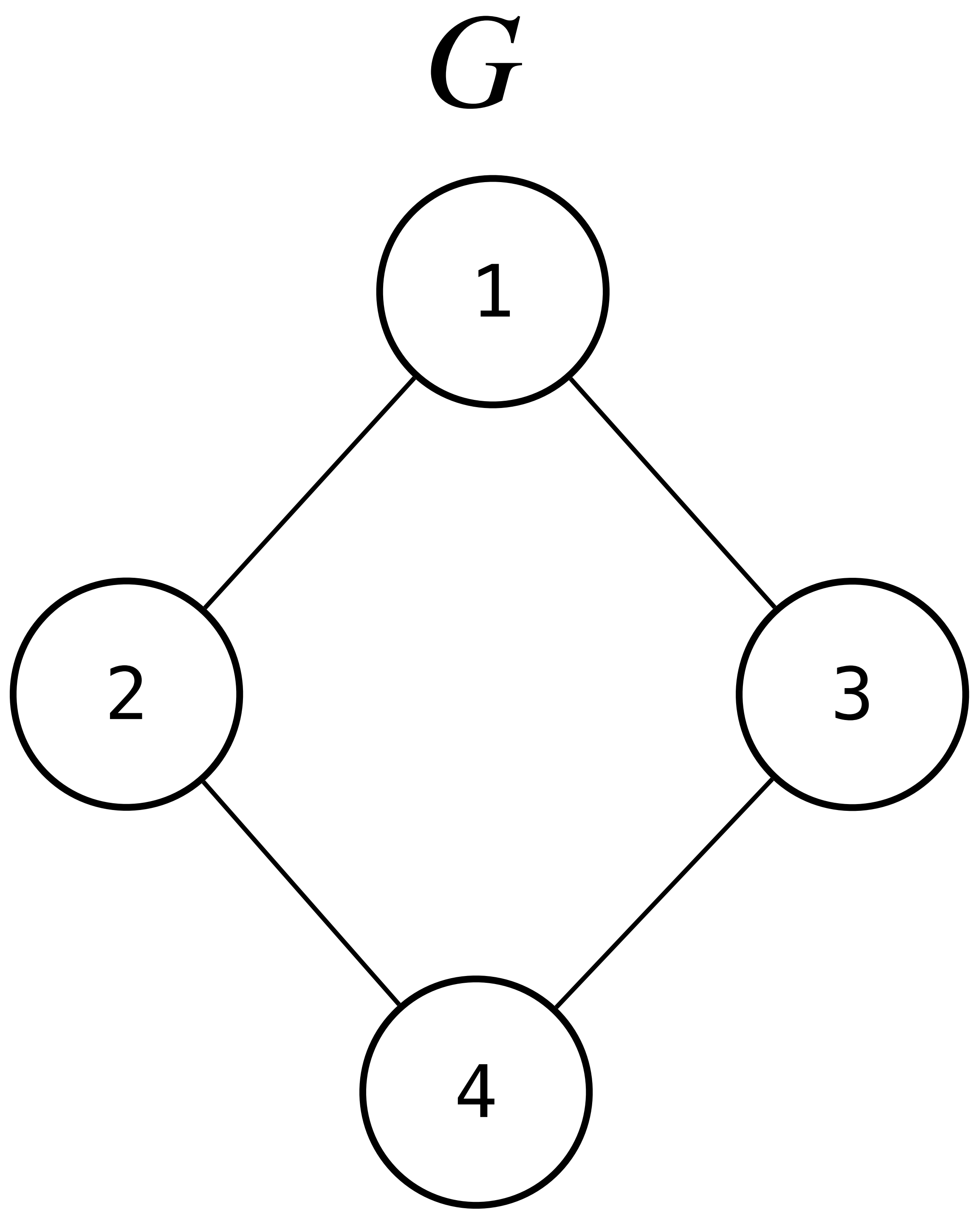
A^k gives us information about k–length paths.

Example



Example

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

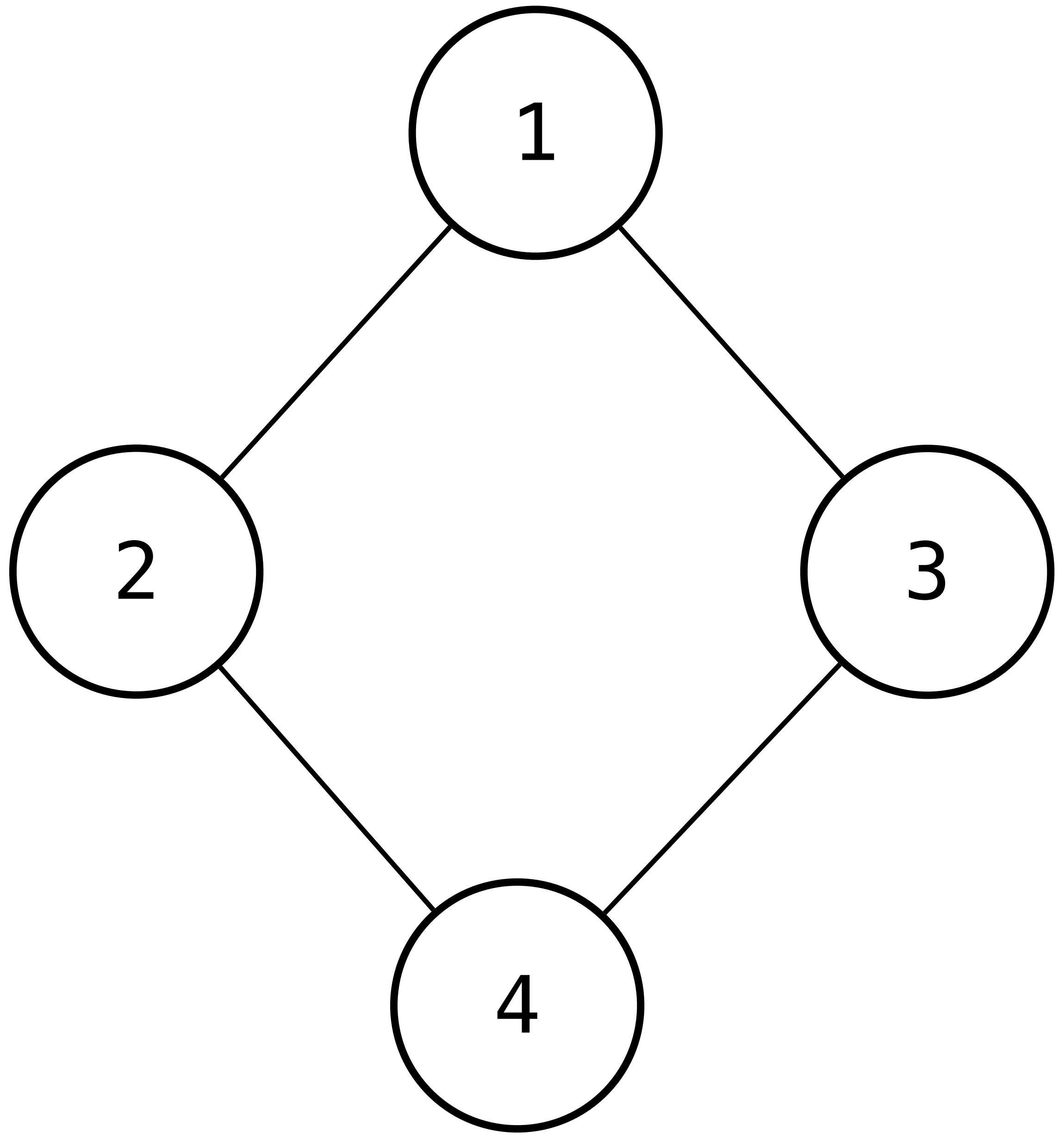


Example

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

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

G



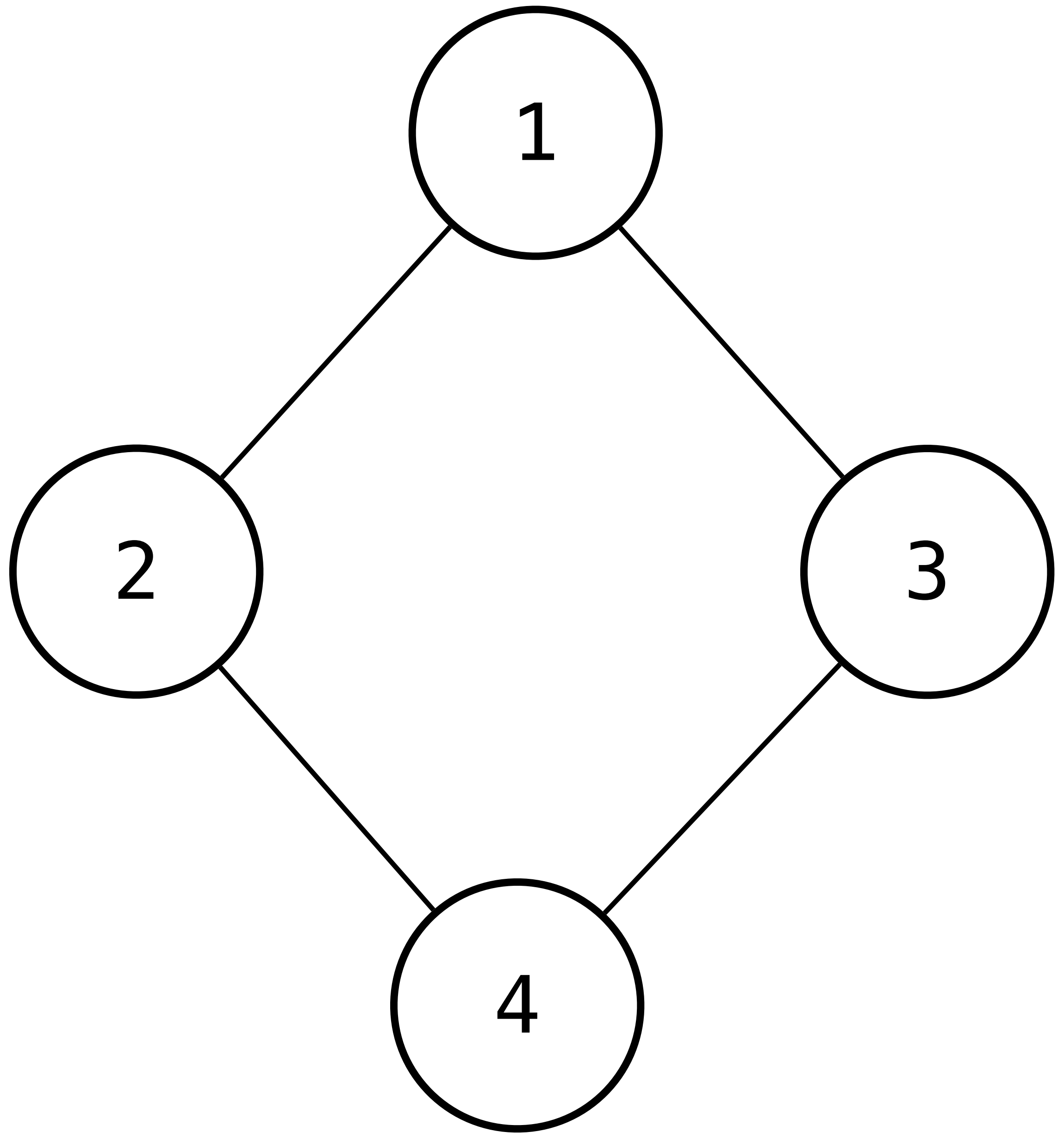
Example

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

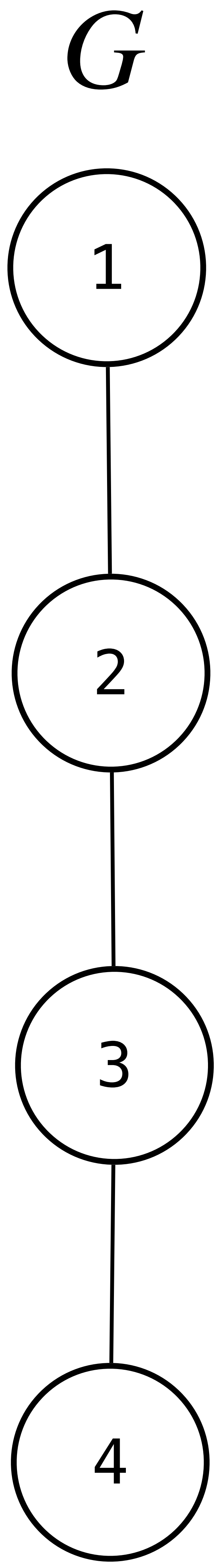
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 2 & 0 & 0 & 2 \\ 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}$$

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

G

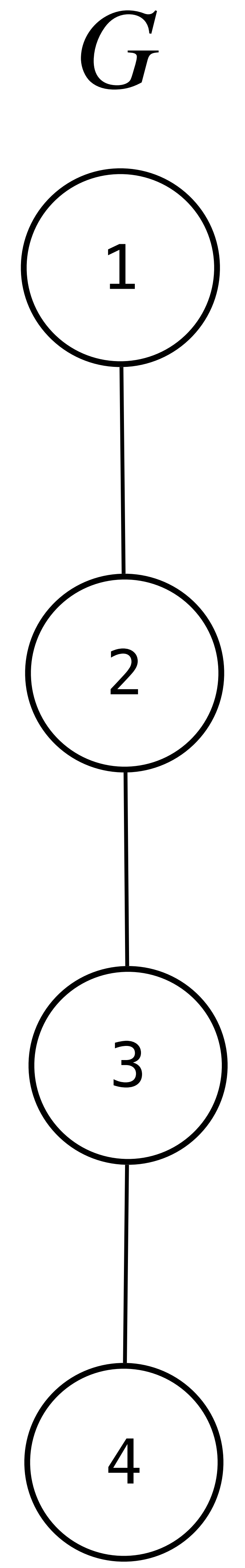


Example



Example

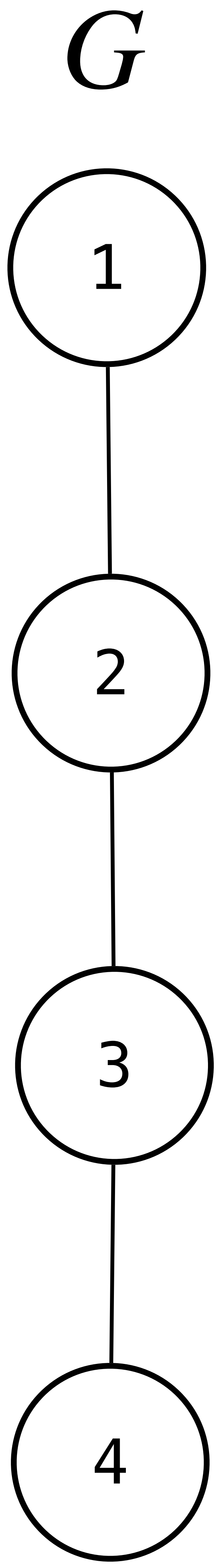
$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \texttt{adjacency matrix for } G$$



Example

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

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



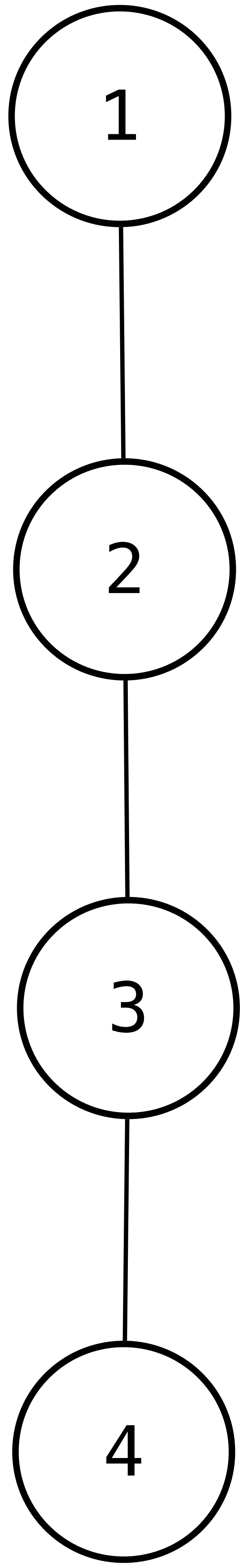
Example

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

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

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

G

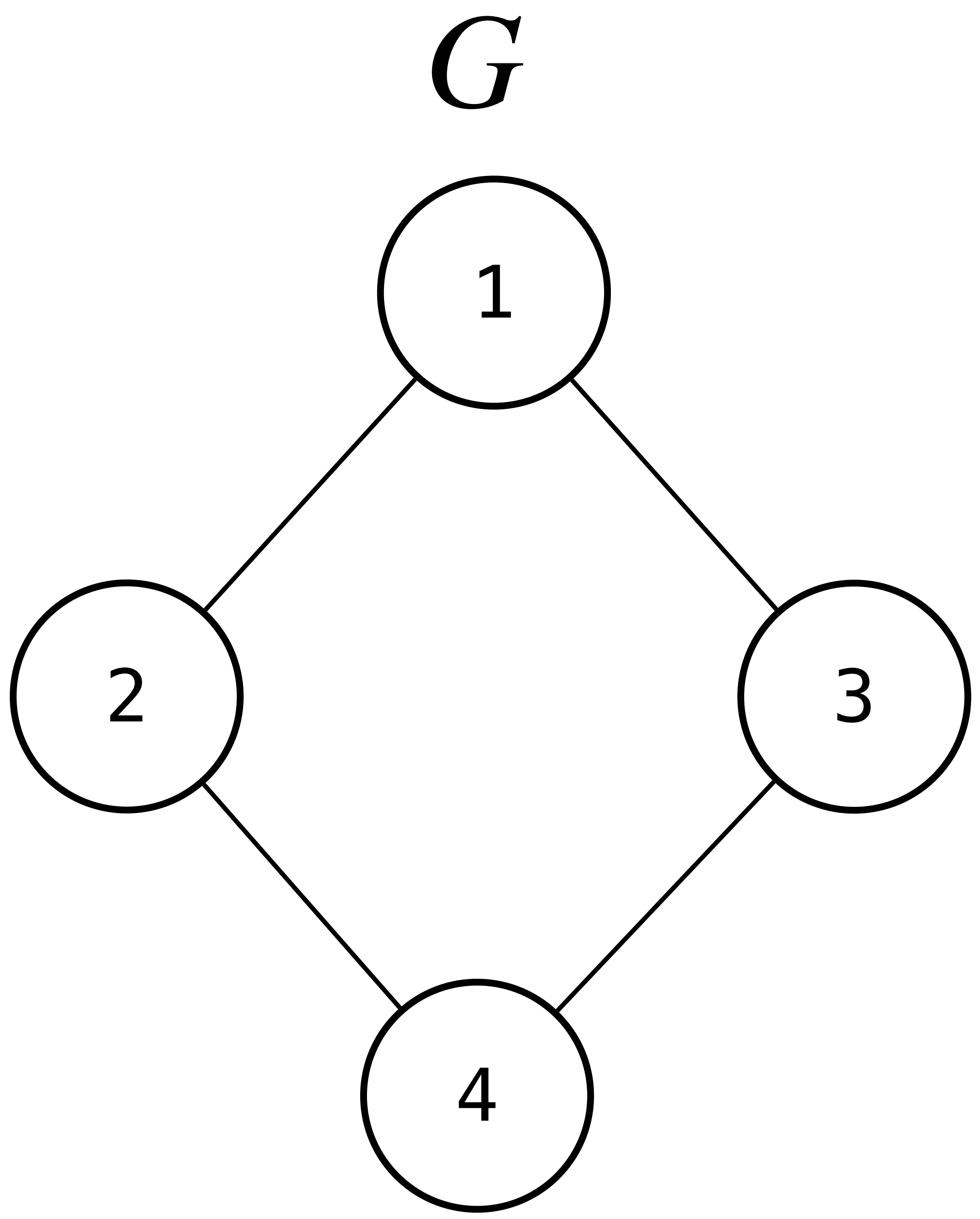


Another Application: Reachability

Theorem: Let G be a simple graph.

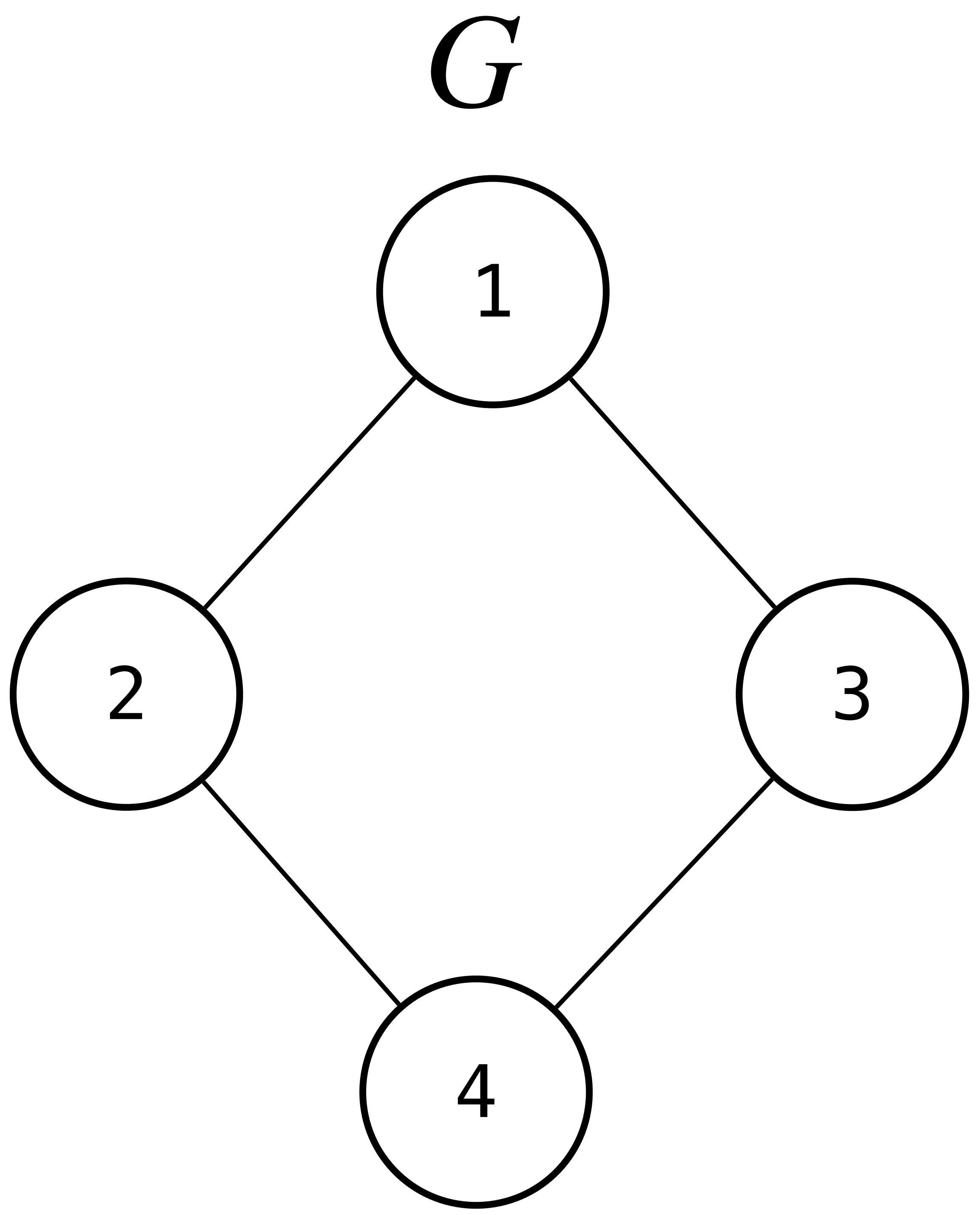
- $(A_G^k)_{ij}$ is the number of paths of length exactly k from v_i to v_j.
- $\left((A_G + I)^k\right)_{ij}$ is nonzero if and only if there is a path of length at at most k from v_i to v_j.

Example



Example

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

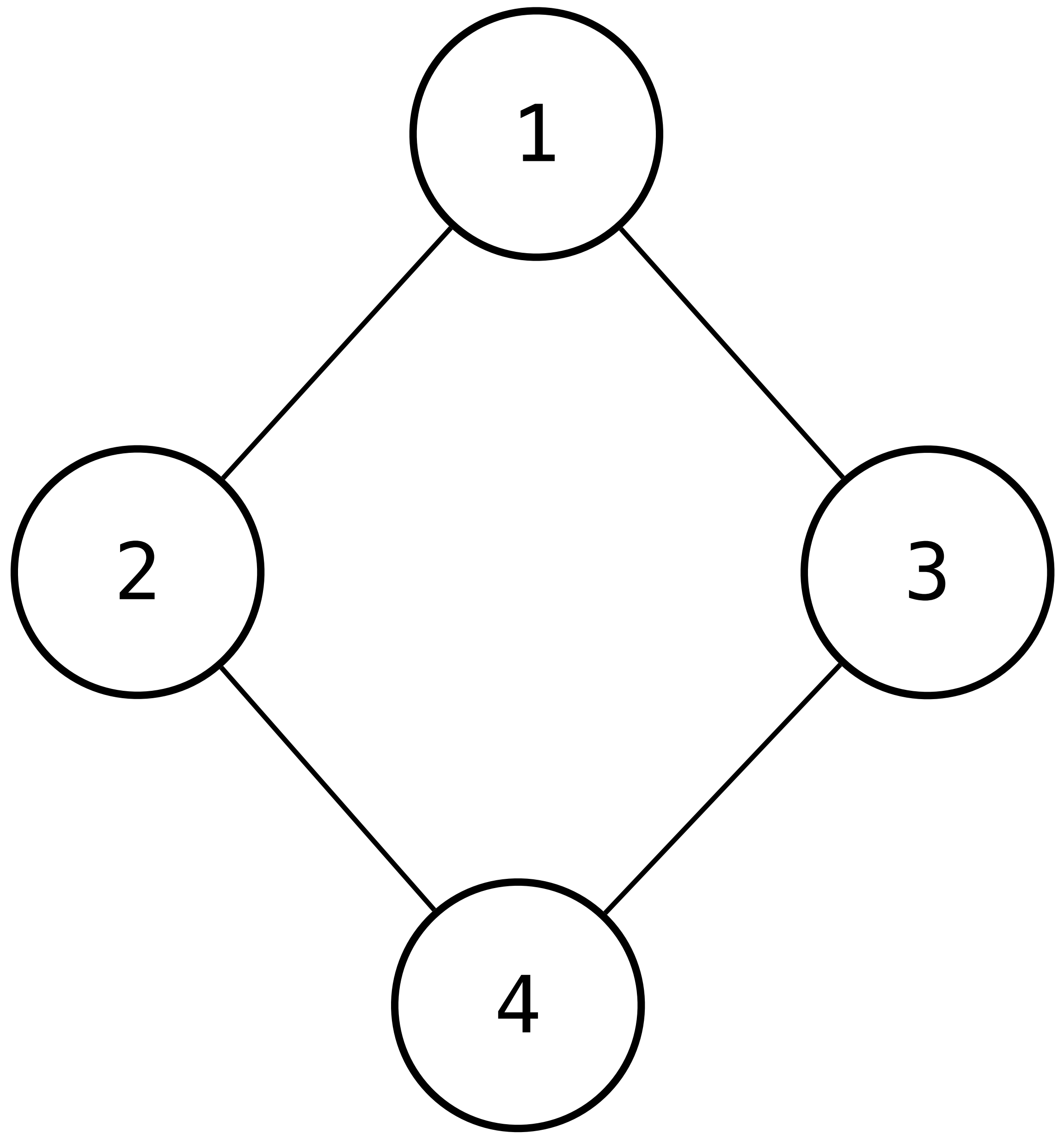


Example

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

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

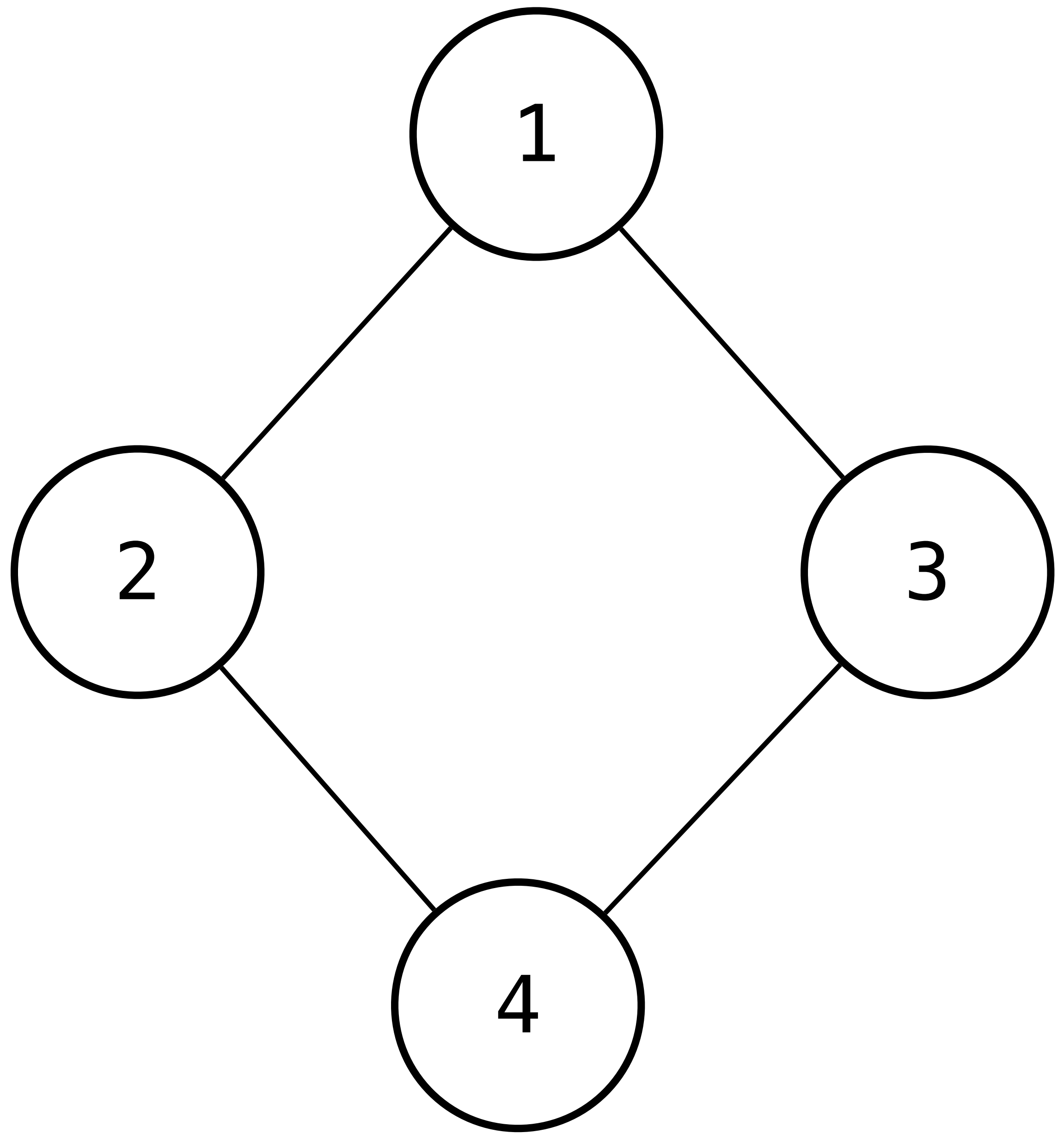
G



Example

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

G



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

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

How To: Reachability

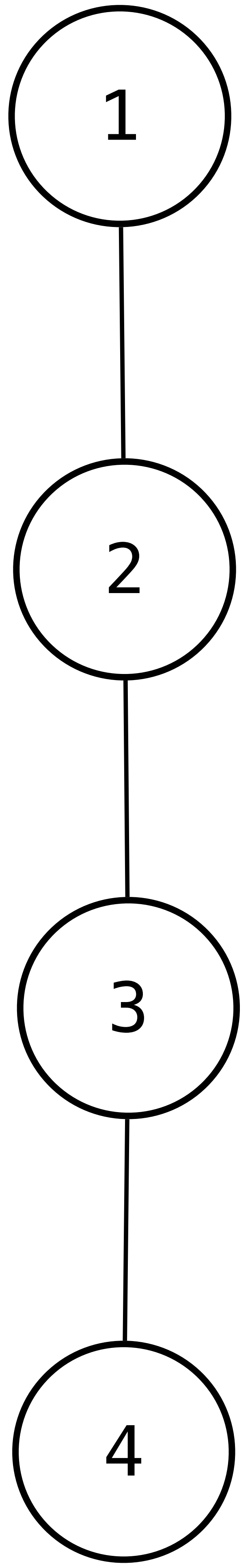
Question: Given a simple graph G determine how many nodes v_i can reach in at least k steps.

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.*

G



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