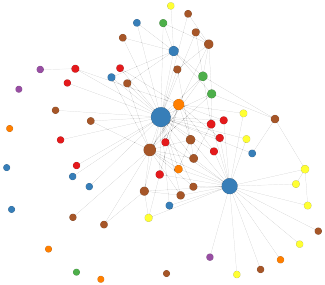
Title of Presentation

Topics



March, 2026

1 Components

2 Examples

1 Components
- Lists
- Maths
- Tables
- Figures
- Others

2 Examples

1 Components
Lists
Maths
Tables
Figures
Others

2 Examples

Simple list

- Google
- Microsoft
- Apple
- Amazon
- Meta

Enumerate list

1. Python
2. Java
3. C++
4. C
5. Go

List of list (1)

- Install requirements
  - `Python`
  - `Numpy`
  - `Pandas`
  - `NetworkX`
  - `Matplotlib`
- Topics
  - Create graphs
  - Graphs visualization
- Links
  - Open in GitHub
  - Open in Google Colab

List of list (2)

- Word embeddings
  - Word2vec
    - **Skip-Gram**
    - Continuous Bag of Words (CBOW)
- Graph embeddings
  - Shallow embeddings
    - DeepWalk
    - **Node2vec**
    - Struc2vec
    - Structural Deep Network Embedding (SDNE)
    - Metapath2vec
  - Graph Neural Networks (GNN)
    - Graph Convolutional Networks (GCN)
    - Graph Attention Networks (GAT)
    - Graph Isomorphism Networks (GIN)
    - GraphSAGE
    - Graph Autoencoders (GAE)

Multicols list

- Computer vision
- Natural language processing
- Recommender systems
- Program analysis
- Knowledge graphs
- Bioinformatics
- Chemistry
- Biology
- Physic
- Anomaly detection
- Urban intelligence
- Financial transaction

Description list

Input: Graph data

Output: Community labels

Method: Leiden algorithm

Sequential list display

- Many people use LaTeX, and many universities have their own Beamer themes

Sequential list display

- Many people use LaTeX, and many universities have their own Beamer themes
- For Chinese support, use XeLaTeX as the compiler

List (dynamic) I

- This template is modified from a Tsinghua University project
- Original template: `https://www.overleaf.com/latex/templates/thu-beamer-theme/vwnqmzndvwyb`
- Minor modifications were made
- Style adjusted to personal preference
- Added small Beamer usage examples
- This is *emphasized text*
- Replaced default fonts with more readable ones

List (dynamic) II

- Fixed inconsistent Chinese font issues
- Changed math fonts to serif for better appearance
- **Use `\alert` to emphasize text**
- This template is modified from a Tsinghua University project
- Original template: `https://www.overleaf.com/latex/templates/thu-beamer-theme/vwnqmzndvwyb`

1 Components
- Lists
- Maths
- Tables
- Figures
- Others

2 Examples

Simple equations

$V = \frac{4}{3}\pi r^3$

$$V = \frac{4}{3}\pi r^3$$

Equations

$$E = mc^2$$

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t] \tag{1}$$

$$Q_{\text{target}} = r + \gamma Q^\pi(s', \pi_\theta(s') + \epsilon) \tag{2}$$

$$\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) \tag{3}$$

Equation with numbers

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(a^2 + \Big(a^2 + 2a\Delta x + (\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 2a\Delta x + 2^2\,(\Delta x)^2\Big) \\
+ \Big(a^2 + 2\cdot 3a\Delta x + 3^2\,(\Delta x)^2\Big) \\
+ \dots \\
+ \Big(a^2 + 2\cdot (n-1)a\Delta x + (n-1)^2\,(\Delta x)^2\Big)\Big) \\
= \frac{1}{3}\Big(b^3 - a^3\Big) \qquad (4)
\end{aligned}
$$

Algorithms

Algorithm 1: My algorithm.

Input : G, s ▹ Graph G and a source node s
Output: All nodes reachable from s

```
Q ← ∅                                  ▹ Create an empty queue
V ← ∅                                  ▹ Create an empty set of visited
enqueue(Q, s)
append(V, s)
while Q ≠ ∅ do
    v ← dequeue(Q)
    print(v)
    foreach u ∈ N(v) do
        if u ∉ V then
            append(V, u)
            enqueue(Q, u)
```

1 Components
Lists
Maths
Tables
Figures
Others

2 Examples

Table

Table 1: My table.

Representation	Complexity
Adjacency matrix	$O(\|V\|^2)$
Adjacency list	$O(\|V\| + \|E\|)$
Edge list	$O(\|E\|)$
Incidence matrix	$O(\|V\|\|E\|)$

Table (custom)

Table 2: My table.

Representation	Advantages	Disadvantages
Adjacency matrix	Simple structure, constant-time edge lookup.	High memory usage for large sparse graphs.
Adjacency list	Memory efficient for sparse graphs, easy to iterate over neighbors.	Edge existence check may be slower.
Edge list	Very simple representation, useful for algorithms that process edges sequentially.	Inefficient for neighbor queries.
Incidence matrix	Useful in some theoretical formulations and algebraic graph analysis.	High memory usage and rarely used in practical implementations.

Table (custom)

Table 3: My table.

IA	Tasks
AI learns patterns	AI automates tasks
AI processes data	AI supports decisions
AI powers assistants	AI transforms industries
AI evolves	AI predicts

1 Components
Lists
Maths
Tables
Figures
Others

2 Examples

Figure

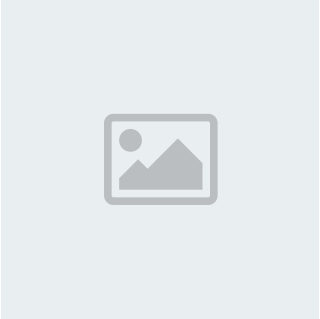


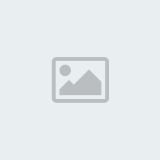
Figure 1: My image.
Source: Author, 2024.

Multiple figures 1

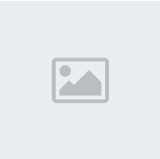
(a) My image. (b) My image. (c) My image. (d) My image.

Figure 2: My image.
Source: Author, 2024.

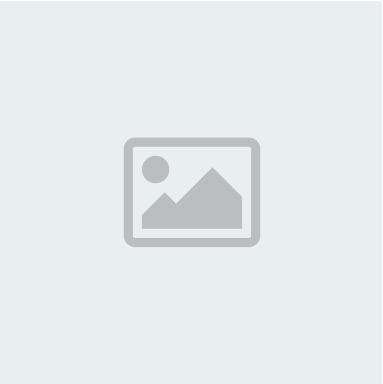
Multiple figures 2



(a) My image.
Source: Author, 2024.



(b) My image.
Source: Author, 2024.



(c) My image.
Source: Author, 2024.

Figure 3: My image.
Source: Author, 2024.

Figure (dynamic)

Figure 4: My image.
Source: Author, 2024.

1 Components
- Lists
- Maths
- Tables
- Figures
- Others

2 Examples

Blocks 1

Info

General content

Warning

Important message

Example

Example content

Definition

Example definition

Example

Example content

Blocks 2

Theorem (Pythagoras)

$a^2 + b^2 = c^2$

Corollary

$x + y = y + x$

Proof.

$\omega + \phi = \epsilon$ □

Frame (two columns)

Se enfoca solo en:

- Números (notas)
- Operaciones matemáticas

Se ignora:

- Nombres de cursos
- Detalles del estudiante

Frame (two columns (list of list))

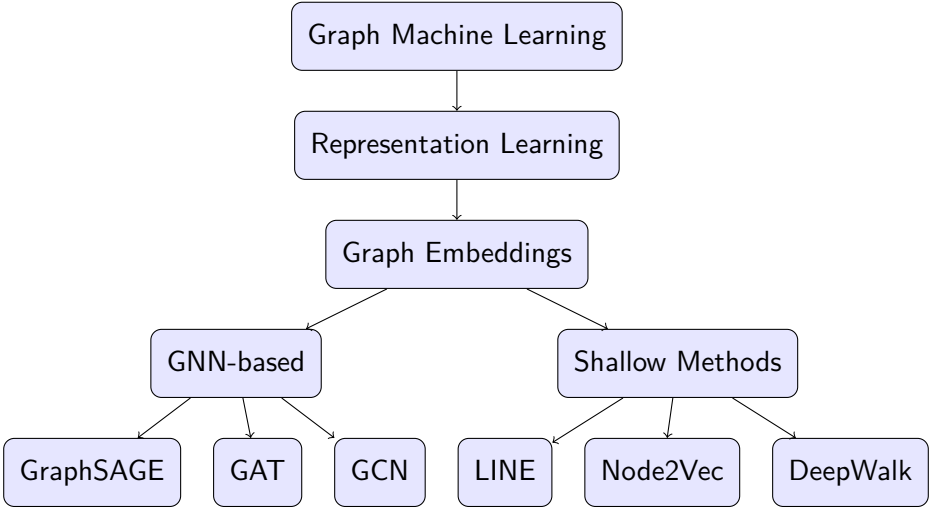
- **Graph traversal**
  - Breadth-First Search
  - Depth-First Search

- **Shortest path algorithms**
  - Dijkstra
  - Floyd–Warshall
  - Bellman-Ford

Frame (three columns)

- **Node-level**
  - Degree
  - Degree centrality
  - Closeness centrality
  - Betweenness centrality
  - Eccentricity
- **Edge-level**
  - Shortest path
  - Betweenness centrality
  - Distance
- **Graph-level**
  - Order
  - Size
  - Density
  - Diameter
  - Radius
  - Clustering coefficient
  - Modularity

Tree


Graph Machine Learning
Representation Learning
Graph Embeddings
GNN-based
Shallow Methods
GraphSAGE
GAT
GCN
LINE
Node2Vec
DeepWalk

1 Components

2 Examples

Example 1

In mathematics and computer science, a graph (network) is defined as $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the nodes (or *vertices*, or *points*) set and $E = \{e_1, e_2, ..., e_m\}, e_i \in V \times V$ a set of edges (or *lines*, or *arcs*) formed by unordered pairs of nodes.

Number of nodes is denoted $|V|$.
Number of edges is denoted $|E|$.

Figure 5: My image.
Source: Author, 2024.

Example 2

Furthermore, each edge $e \in E$ is said to join two vertices. If e joins $u, v \in V$, then $e = (u, v)$. Vertices u and v are said to be adjacent.

Neighborhood

$$N(v) = \{w \in V \mid v \neq w \wedge (v, w) \in E\} \tag{5}$$

Backward propagation computes the gradients of the loss $\mathcal{L}$ or $\mathscr{L}$ and updates the parameters using gradient descent.

$$W^{(l)} = W^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial W^{(l)}} \tag{6}$$

$$b^{(l)} = b^{(l)} - \eta \frac{\partial \mathcal{L}}{\partial b^{(l)}} \tag{7}$$

Sum operator: $z_{uv} = z_u + z_v \in \mathbb{R}^d$.

Example 3

Adjacency matrix

Adjacency matrix A, is an $n \times n$ matrix where:

$$A_{i,j} = \begin{cases} 1 & \text{if there is an edge from vertex } i \text{ to vertex } j \\ 0 & \text{otherwise} \end{cases}$$

(a) My image. (b) My image.

Figure 6: My image.
Source: Author, 2024.

Example 4

Idea

Does the shortest path from a to c go through b?

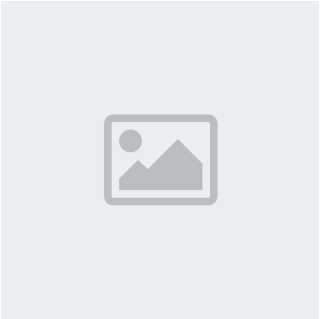


Figure 7: My image.
Source: Author, 2024.

Example 5

BFS is a graph traversal algorithm that explores nodes level by level, visiting all nodes at a given distance from the starting node before moving to the next level.

Description:

- Uses a queue (FIFO) to manage the nodes to be explored.
- Marks visited nodes to avoid cycles or repeated visits.
- Ideal for finding the shortest path in unweighted graphs.

Example 6

In this lab, students will apply basic programming concepts to solve problems, develop coding skills, and practice writing and debugging efficient solutions.

- Requirements
  - `Python`
  - `Numpy`
  - `Pandas`
  - `NetworkX`
  - `Matplotlib`
- Topics
  - Create graphs
  - Graphs visualization
- Lab
  - Open in Google Colab

Example 7

This section introduces key equations to model and analyze the problem.

Main equation

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t]$$

Supplementary equations

$$Q_{\text{target}} = r + \gamma Q^{\pi}(s', \pi_\theta(s') + \epsilon) \tag{8}$$

$$\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma), -c, c) \tag{9}$$

Solution

$$
\begin{aligned}
A = \lim_{n\to\infty} \Delta x \Big(a^2 + \Big(& a^2 + 2a\Delta x + (\Delta x)^2 \Big) \\
& + \Big(a^2 + 2 \cdot 2a\Delta x + 2^2 (\Delta x)^2 \Big) \\
& + \Big(a^2 + 2 \cdot 3a\Delta x + 3^2 (\Delta x)^2 \Big) \\
& + \dots \\
& + \Big(a^2 + 2 \cdot (n-1)a\Delta x + (n-1)^2 (\Delta x)^2 \Big) \Big) \\
& = \frac{1}{3} \left(b^3 - a^3 \right) \quad (10)
\end{aligned}
$$

Example 8

1. Visit the adjacent unvisited node. Mark as visited, display, and insert (queue) in a queue.
2. In case no adjacent node is found, remove the first node from the queue.
3. Repeat steps 1, and 2 until the queue is empty.

Figure 8: My image.
Source: Author, 2024.

Example 9

The following table summarizes the key information and results, providing a clear overview of the main variables and their relationships.

Table 4: My table.

Representation	Complexity
Adjacency matrix	$O(\|V\|^2)$
Adjacency list	$O(\|V\| + \|E\|)$
Edge list	$O(\|E\|)$
Incidence matrix	$O(\|V\|\|E\|)$

Example 10

The following table summarizes the key information and results, providing a clear overview of the main variables and their relationships.

Table 5: My table.

Representation	Complexity
Adjacency matrix	$O(\|V\|^2)$
Adjacency list	$O(\|V\| + \|E\|)$
Edge list	$O(\|E\|)$
Incidence matrix	$O(\|V\|\|E\|)$

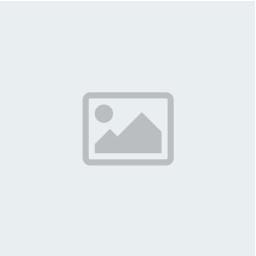


Figure 9: My image.
Source: Author, 2024.

Example 11

The following table summarizes the key information and results, providing a clear overview of the main variables and their relationships.

1. **Análisis:** Definir requisitos y comprender el problema.
2. **Diseño:** Planificar la estructura del algoritmo.
3. **Implementación:** Escribir el código en un lenguaje de programación.
4. **Prueba:** Verificar el funcionamiento y corregir errores.

Figure 10: My image.
Source: Author, 2024.

Example 12

Degree

Given a graph $G = (V, E)$.
The degree of a node u, denoted as $deg(u)$, is defined as the number of edges that are adjacent to this node u.

For undirected graphs:

$$deg(u) = |\{v \in V : (u, v) \in E\}| \quad (11)$$

For directed graphs:

$$deg(u) = deg^{in}(u) + deg^{out}(u) \quad (12)$$

Where, $deg^{in}(u)$: in-degree and $deg^{out}(u)$: out-degree.

Applications:

- Social networks: users with many connections.
- Biology networks: metabolites involved in many reactions.
- Web: most linked pages.

Example 13

Degree centrality

Given a graph $G = (V, E)$.
Degree centrality considers the node with the highest degree as the most central node in the network G. Degree centrality focuses on individual nodes; it simply counts the number of edges that a node has.

$$DC(v) = \frac{deg(v)}{|V| - 1} \tag{13}$$

Applications:

- Social networks: who has the most friends?
- Internet: which routers or servers handle the most connections?
- Citation networks: which papers are most cited?

Example 14

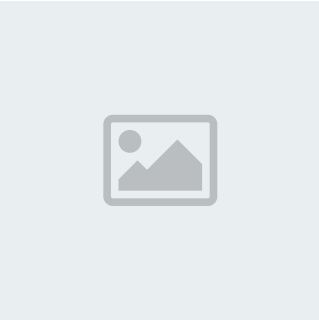


Figure 11: My image.
Source: Author, 2024.

Example 15

Descomposición

Consiste en **dividir** el problema en sub problemas o en partes más pequeñas (unidades lógicas de menor complejidad); luego, solucionar cada sub problema independientemente. Finalmente, las soluciones a los subproblemas se integran para resolver el problema original completo.

Ejemplo: Sistema para el cálculo de promedio de notas.

Dividir el problema en:

1. Ingresar notas
2. Calcular suma
3. Calcular promedio
4. Mostrar resultado

Example 16

Abstracción

Consiste en la capacidad de identificar los aspectos más importantes de un problema o situación y reducir la complejidad al enfocarse en las partes más relevantes, ignorando los detalles innecesarios. En otras palabras, abstraer implica crear una **representación simplificada** de un problema, modelo o sistema, lo cual facilita su comprensión, análisis y resolución.

Ejemplo: Sistema para el cálculo de promedio de notas.

Se enfoca solo en:

- Números (notas)
- Operaciones matemáticas

Se ignora:

- Nombres de cursos
- Detalles del estudiante

Example 17

Diseño de algoritmo

Es la capacidad de descomponer un problema en una secuencia de pasos lógicos o instrucciones claras que pueden ser ejecutadas para solucionar un problema. En otras palabras, es el proceso de **crear**, **diseñar** y **evaluar** algoritmos que resuelven problemas de manera eficiente.

Ejemplo: Sistema para el cálculo de promedio de notas.

Algorithm 2: Promedio de notas.

Input : $nota_1$, $nota_2$ y $nota_3$ ▹ `Notas del estudiante`

Output: $promedio$

1 $suma \leftarrow nota_1 + nota_2 + nota_3$

2 $promedio \leftarrow \frac{suma}{3}$ ▹ `Cálculo del promedio`

3 Mostrar $promedio$

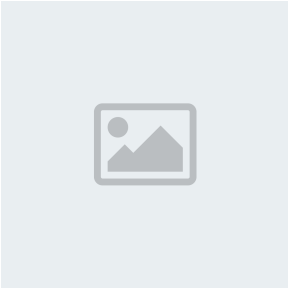
Example 18

Secuenciales

En este tipo de estructura las instrucciones se realizan o se ejecutan una después de la otra.

(a) My image.
Source: Author, 2024.

(b) My image.
Source: Author, 2024.



(c) My image.
Source: Author, 2024.

Figure 12: My image.
Source: Author, 2024.

Thanks!

https://github.com/win7

- Erciyes, K. (2018). *Guide to graph algorithms*
- Yao Ma, Jiliang Tang. (2021). *Deep Learning on Graphs*.
- Stamile C., Marzullo A., Deusebio E. (2021). *Graph Machine Learning*.
- Wu, L., Cui, P., Pei, J., Zhao, L., & Guo, X. (2022). *Graph neural networks: foundation, frontiers and applications*.
- Knickerbocker, D. (2023). *Network Science with Python*.
- Labonne, M. (2023). *Hands-On Graph Neural Networks Using Python*.
- Broadwater, K., & Stillman, N. (2025). *Graph neural networks in action*.