# Introduction to Algorithms and Data Structures

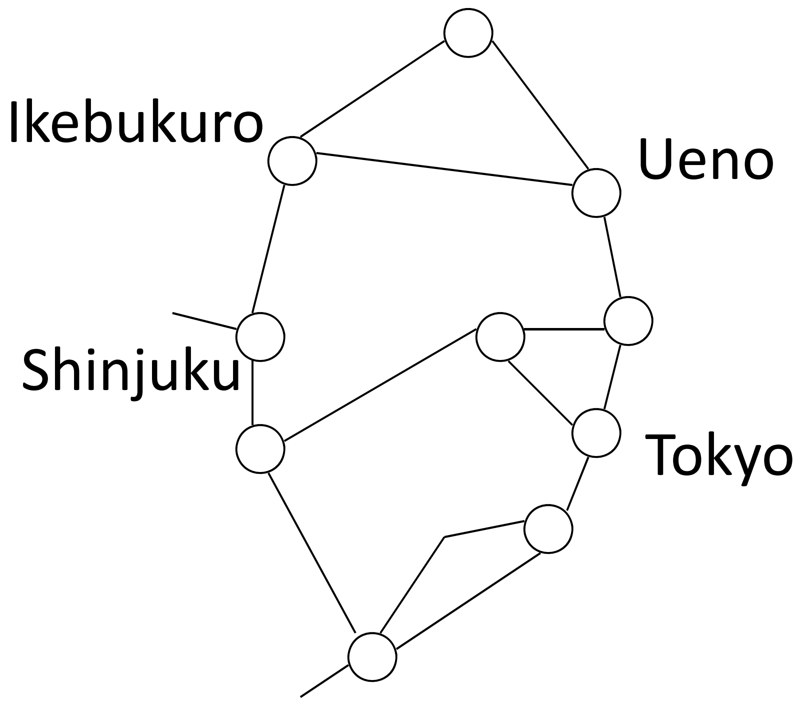
# 10. Data Structure (3) Data structures for graphs

Professor Ryuhei Uehara,
School of Information Science, JAIST, Japan.
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara
http://www.jaist.ac.jp/~uehara/course/2020/myanmar/

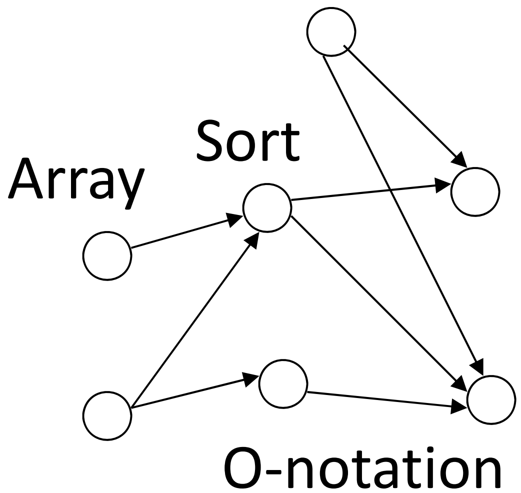
# Graph

- "<u>Vertices</u>" (nodes) are joined by <u>edges</u> (arcs)
  - Directed graph: each edge has direction
  - Undirected graph: each edge has no direction

Example: railway in Tokyo



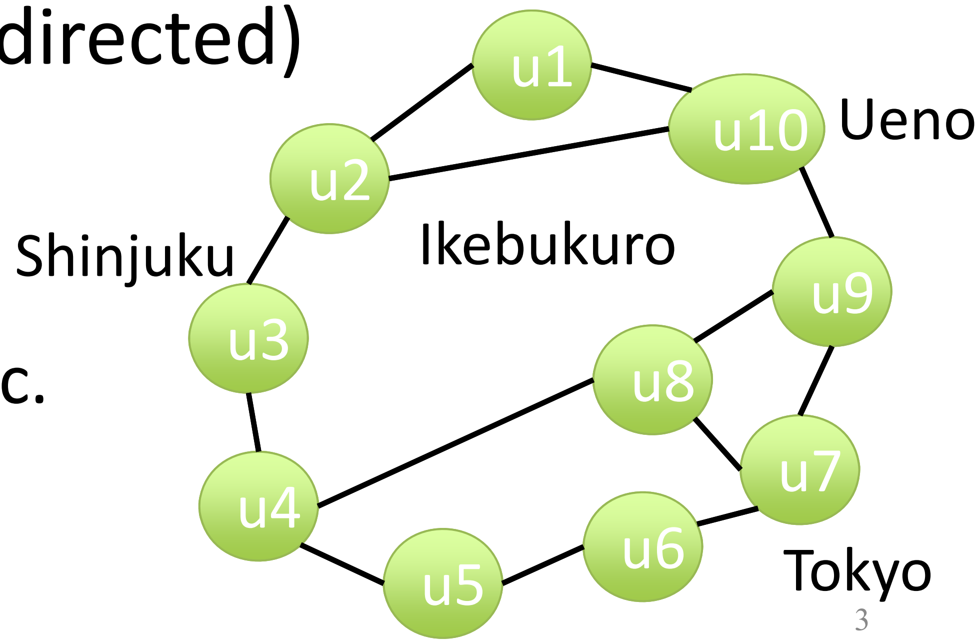

Example:
relationship between topics



# Graph: Notation

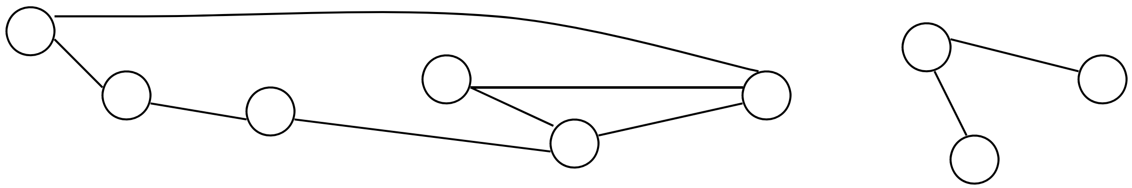
- Graph $G = (V, E)$
  - $V$: vertex set, $E$: edge set
- Vertices: $u, v, \ldots \in V$
- Edges: $e = \{u, v\} \in E$ (undirected)

  $a = (u, v) \in E$ (directed)
- Weighted variants;
  - $w(u)$, $w(e)$
  - Distance, cost, time, etc.

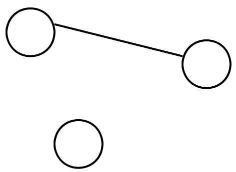


# Graph: basic notions/notations (1/2)

- Path: sequence of vertices joined by edges
  - Simple path: it never visit the same vertex again

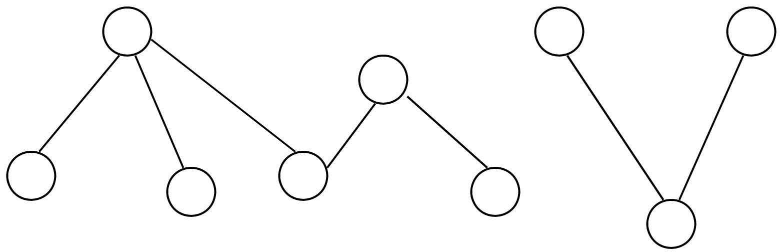


- Cycle, closed path: path from *v* to *v*
- Connected graph: Every pair of vertices is joined by path

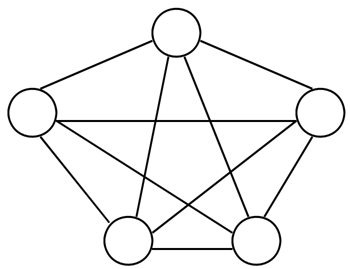


# Graph: basic notions/notations (2/2)

- Forest: Graph with no cycle
- Tree: Connected, and no cycle



- Complete graph: Every pair of vertices is connected by an edge
  - Example: $K_5$



# Computational complexity of graph problems

- The number $n$ of vertices, the number $m$ of edges;
  - Undirected graph: $m \leqq n(n-1)/2$
  - Directed graph: $m \leqq n(n-1)$
    - $m \in O(n^2)$

- Every tree has $m=n-1$ edges, so $m \in O(n)$.

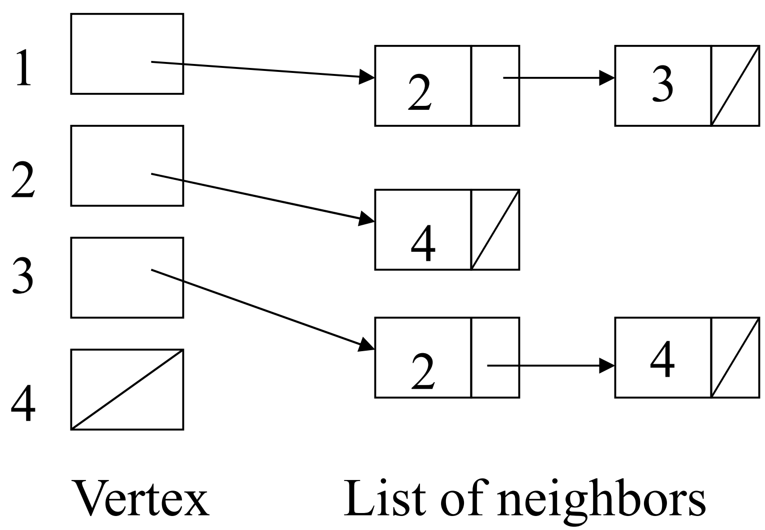
- Computational complexity of graph algorithm is described by equations of $n$ and $m$.

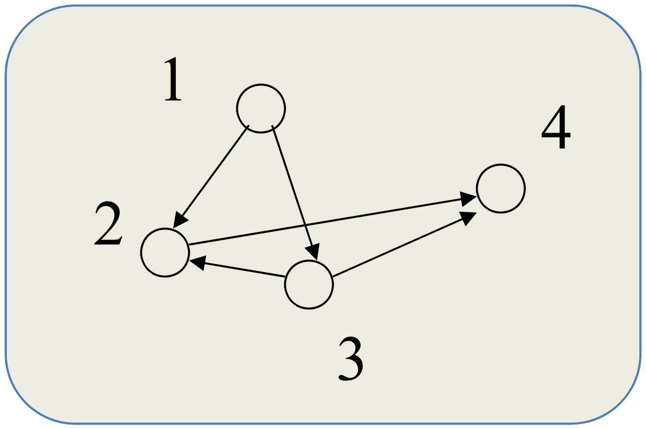
# Representations of a graph in computer

- Adjacency matrix

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- Adjacency list

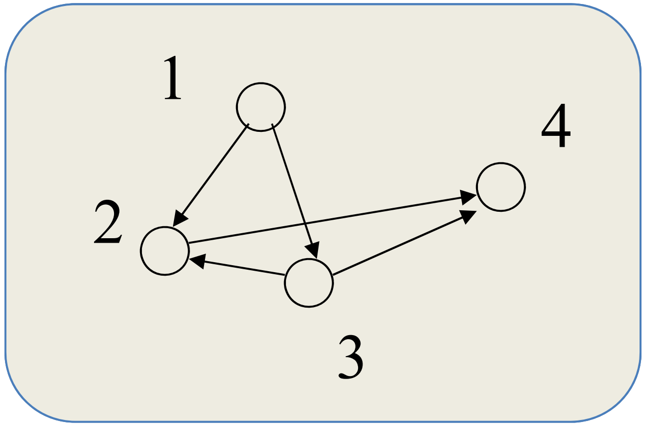


# Representation of a graph:
## matrix representation (adjacency matrix)

- $(u, v) \in E \Rightarrow M[u, v] = 1$
- $(u, v) \notin E \Rightarrow M[u, v] = 0$

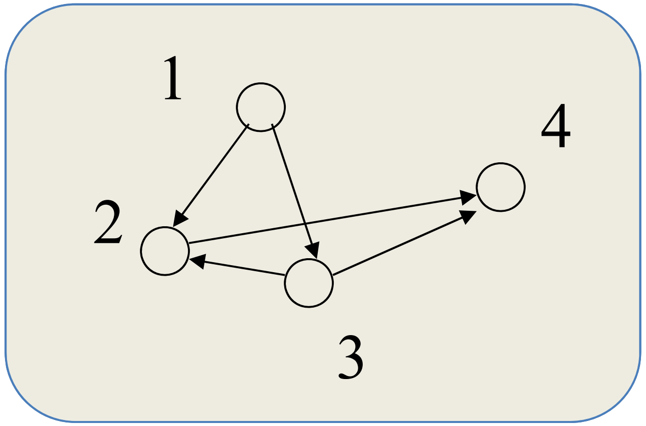
It is easy to extend edge-weighted graph.



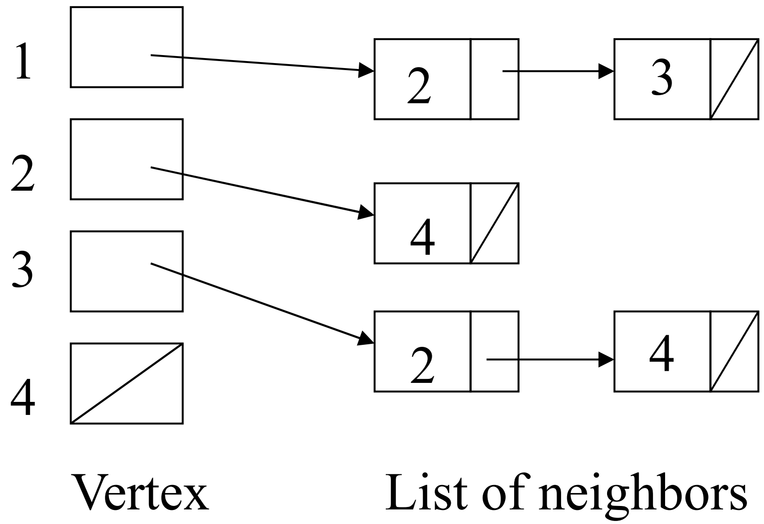

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

# Representation of a graph: list representation (adjacency list)

- $(u, v) \in E \Leftrightarrow v \in L(u)$
  - $L(u)$ is the list of neighbors of $u$




It is easy to extend vertex-weighted graph.



# Adj. matrix v.s. Adj. list

- Space complexity
  - Adjacency matrix: $\Theta(n^2)$
  - Adjacency list: $\Theta(m \log n)$
- Time complexity of checking if $(u, v) \in E$ ?
  - Adjacency matrix: $\Theta(1)$
  - Adjacency list : $\Theta(n)$

**Q. How about update graph?**
**(e.g., add/remove vertex/edge)**