

DATA STRUCTURE AND ALGORITHM

CLASS 8

Seongjin Lee

Updated: 2021-02-24
DSA_2017_08

insight@gnu.ac.kr
<http://resourceful.github.io>
Systems Research Lab.
GNU



Table of contents

1. Graph

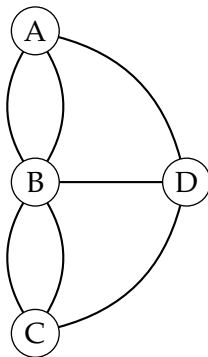
2. Graph Representation

GRAPH



First use of Graph

- Euler used graph to solve the Königsberg bridge problem in 1736
- “Starting at some land area, is it possible to return to the starting point after walking across each of the bridges exactly once?”

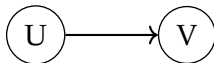


Definitions and Notations

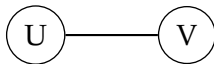
- **Notation:** $G = (V, E)$
- V is the vertex set. Ex. $V = \{0, 1, 2, 3\}$
 - Vertices are also called nodes and points
- E is the edge set. Ex. $E = \{(0, 1), (1, 3), (2, 3), (0, 2)\}$
 - Each edge connects two different vertices
 - Edges are also called arcs and lines

Directed and Undirected Graphs

- Directed Edge has an *Orientation* $\langle u, v \rangle$
 - Directed graph (Digraph) = Every Edge has an orientation

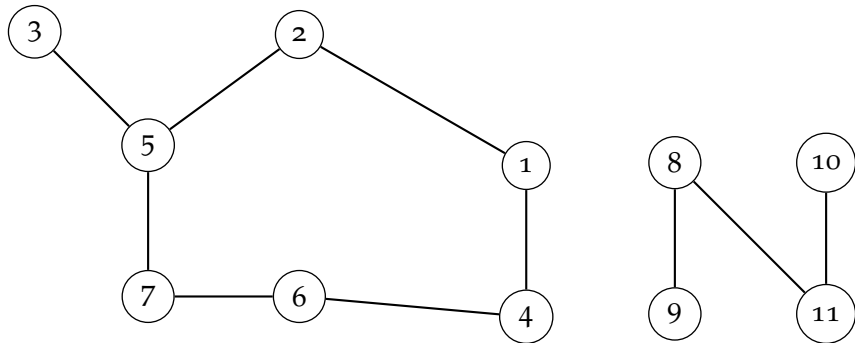


- Undirected edge has no *Orientation* (u, v)
 - Undirected graph = No oriented edge



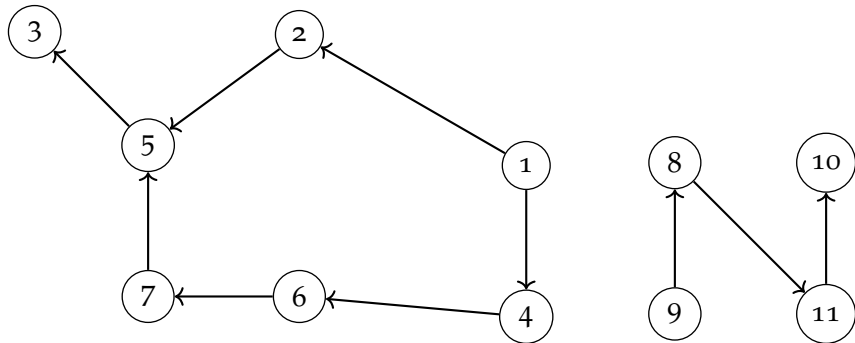
- $(u, v) = (v, u)$ **but** $\langle u, v \rangle \neq \langle v, u \rangle$

Undirected Graph

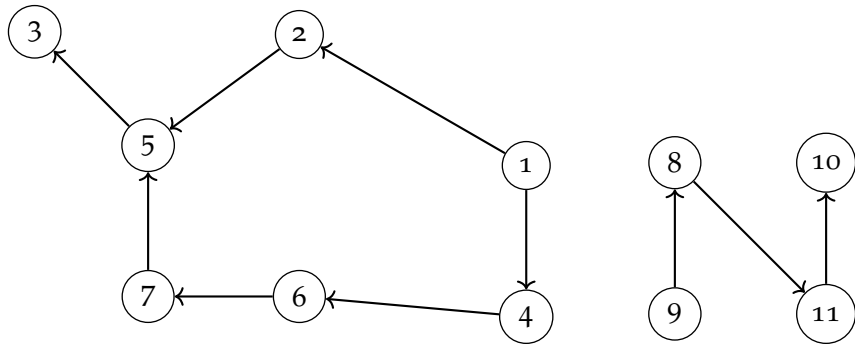


- $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $E = \{(1, 2), (1, 4), (2, 5), (4, 6), (3, 5), (5, 7), (6, 7), (8, 9), (8, 11), (10, 11)\}$

Directed Graph



Directed Graph

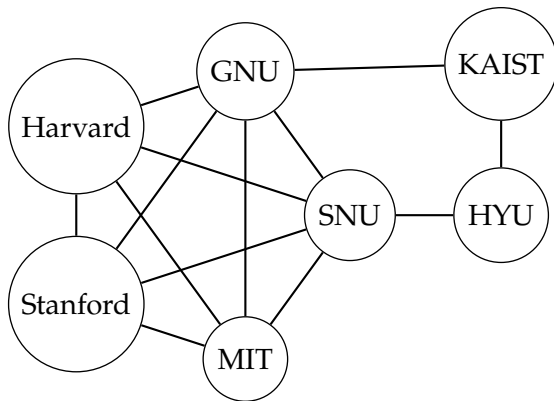


- $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$
- $E = \{ \langle 1, 2 \rangle, \langle 1, 4 \rangle, \langle 2, 5 \rangle, \langle 4, 6 \rangle, \langle 5, 3 \rangle, \langle 6, 7 \rangle, \langle 7, 5 \rangle, \langle 8, 9 \rangle, \langle 8, 11 \rangle, \langle 11, 10 \rangle \}$

Applications

Communication Network

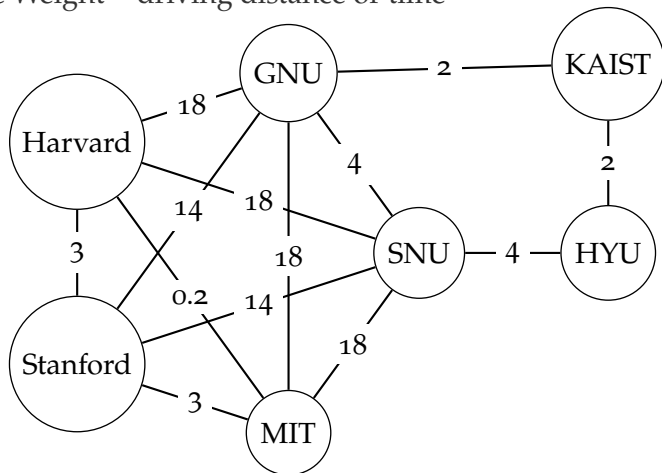
- Vertex = University
- Edge = Communication link



Applications

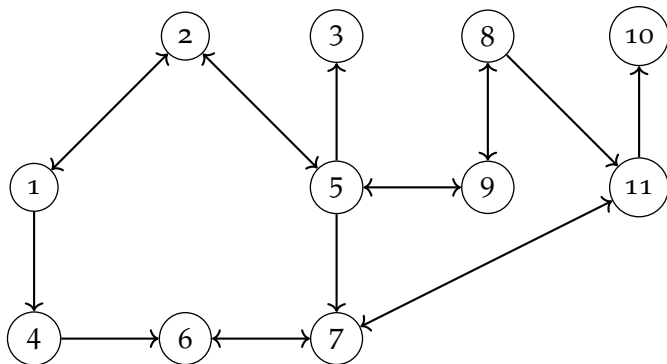
Driving Distance/Time Map

- Vertex = City
- Edge Weight = driving distance or time



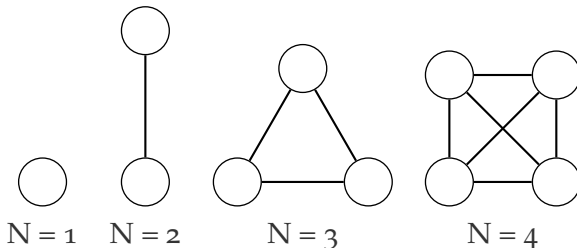
Applications

Some streets are one way



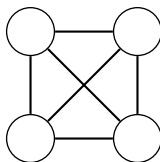
Complete Undirected Graph

- A graph that has the maximum number of edges
- A graph that has all possible edges



Number of Edges in Undirected Graph

- What is number of edges in this graph ?



$$N = 4$$

Number of edges is 6

Number of Edges in Undirected Graph

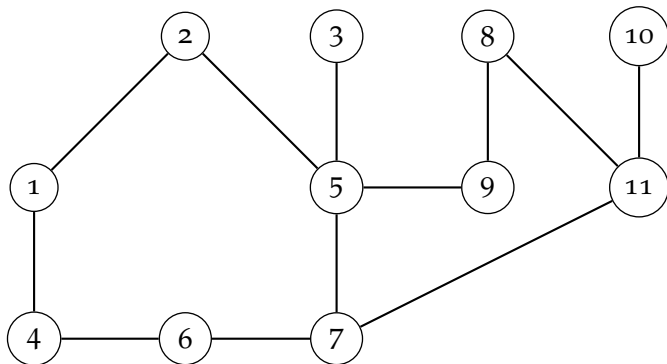
- Each edge is of the form (u, v) , and $u \neq v$
- Number of such pairs in an n vertex graph is $n(n - 1)$.
- Since edge (u, v) is the same as edge (v, u) the number of edges in a complete undirected graph is $\frac{n(n-1)}{2}$.
- Number of edges in an undirected graph is $\leq \frac{n(n-1)}{2}$.

Number of Edges in Directed Graph

- Each edge is of the form $\langle u, v \rangle, u \neq v$.
- Number of such pairs in an n vertex graph is $n(n - 1)$.
- Since edge $\langle u, v \rangle$ is not the same as edge $\langle v, u \rangle$, the number of edges in a complete directed graph is $n(n - 1)$.
- Number of edges in a directed graph is $\leq n(n - 1)$.

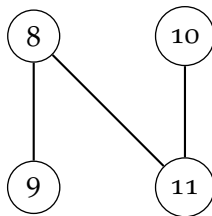
Vertex Degree

- **Vertex Degree** is the number of edges incident to that vertex
- $\text{Degree}(2) = 2$, $\text{Degree}(5) = 4$, $\text{Degree}(3) = 1$,



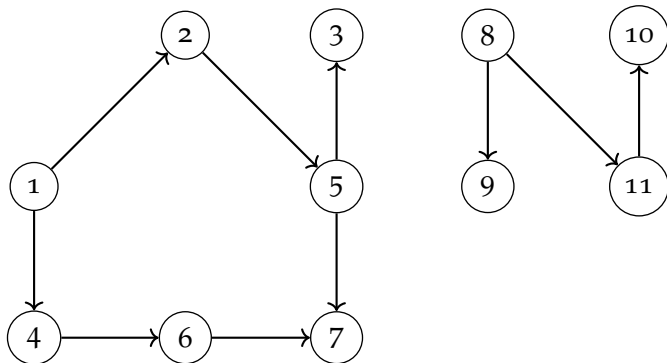
Sum of Vertex Degrees

- **Sum of Vertex Degrees** is $2 \times e$, where e is number of edges



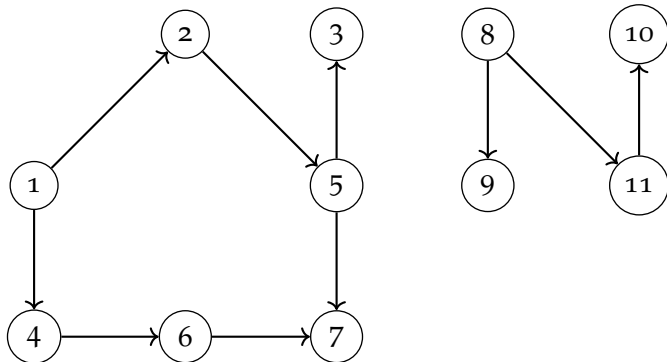
In-Degree of a Vertex

- In-degree of a vertex is the number of incoming edges
- In-degree(2) = 1, in-degree(8) = 0



Out-Degree of a Vertex

- Out-degree of a vertex is the number of outbound edges
- $\text{Out-degree}(2) = 1$, $\text{out-degree}(8) = 2$



Sum of In- and Out-Degrees

- Each edge contributes 1 to the in-degree of some vertex and 1 to the out-degree of some other vertex
- Sum of in-degree = sum of out-degree = e , where e is the number of edges in the directed graph

Other Definitions

- **Graph:** $G = (V, E)$
- If G is undirected graph and (u, v) is an edge of G , then
 - u and v are **adjacent**
 - (u, v) is **incident**(부속/교차) on vertices u and v
- If G is directed graph and $\langle u, v \rangle$ is an edge of G , then
 - u is **adjacent** to v
 - v is **adjacent** from u
 - the edge $\langle u, v \rangle$ is incident to u and v
- Path from u to v
- **Simple path** : a path in which all vertices, except possibly the first and the last, are distinct (repeats no vertices)
- **Cycle** : a simple path in which the first and the last vertices are the same

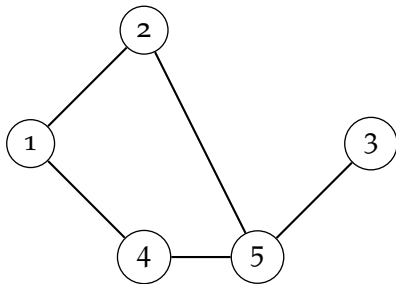
GRAPH REPRESENTATION

Graph Representation

- Adjacency Matrix
- Adjacency Lists
 - Linked Adjacency Lists
 - Array Adjacency Lists

Adjacency Matrix

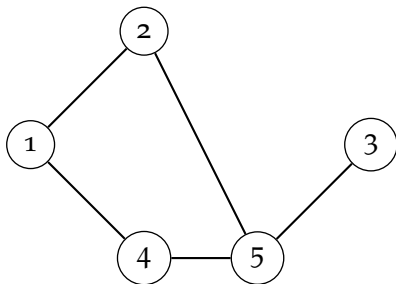
- $n \times n$ matrix with 0 and 1 representing the incidents, where $n = \#$ of vertices
- $A(i, j) = 1$ iff (i, j) is an edge



	1	2	3	4	5
1	0	1	0	1	0
2	1	0	0	0	1
3	0	0	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0

Properties of Adjacency Matrix

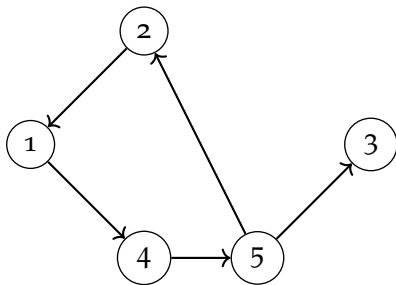
- Diagonal entries are zero
- Adjacency matrix of an undirected graph is symmetric
 - $A(i, j) = A(j, i)$ for all i and j



	1	2	3	4	5
1	0	1	0	1	0
2	1	0	0	0	1
3	0	0	0	0	1
4	1	0	0	0	1
5	0	1	1	1	0

Adjacency Matrix for Directed Graph

- Diagonal entries are zero
- Adjacency matrix of a directed graph need not be symmetric



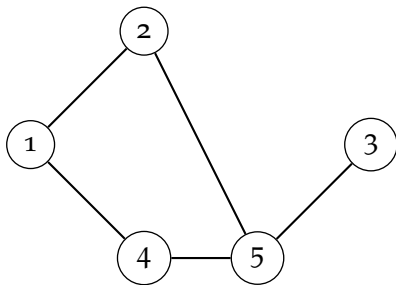
	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1
5	0	1	1	0	0

Properties of Adjacency Matrix

- n^2 bits of space is required
- For an undirected graph, may store only lower or upper triangle (exclude the diagonal)
 - $\frac{n(n-1)}{2}$ bits
- $O(n)$ time to find vertex degree and/or vertices adjacent to a give vertex

Adjacency Lists

- Adjacency list for vertex i is a linear list of vertices adjacent from vertex i .
- An array of n adjacency lists.



$\text{AdjList}[1] = (2, 4)$

$\text{AdjList}[2] = (1, 5)$

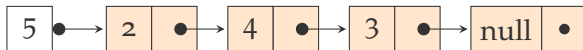
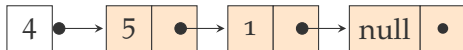
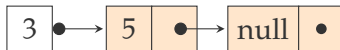
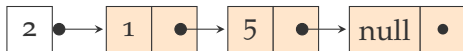
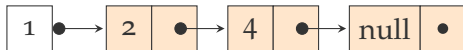
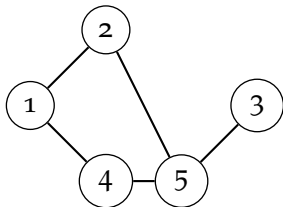
$\text{AdjList}[3] = (5)$

$\text{AdjList}[4] = (5, 1)$

$\text{AdjList}[5] = (2, 4, 3)$

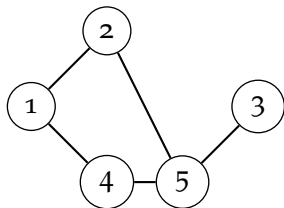
Linked Adjacency Lists

- Each adjacency list is a chain
 - Array length = n
 - # of chain nodes = $2e$ (undirected graph)
 - # of chain nodes = e (directed graph)



Linked Adjacency Lists

- Each adjacency list is an array list
 - Array length = n
 - # of chain nodes = $2e$ (undirected graph)
 - # of chain nodes = e (directed graph)



1	2	4	
2	1	5	
3	5		
4	5	1	
5	2	4	3

Weighted Graphs

- Cost Adjacency Matrix
 - $C(i, j)$ = cost of edge (i, j)
- Each list element of adjacency lists is a pair (Adjacent vertex, Edge weight)