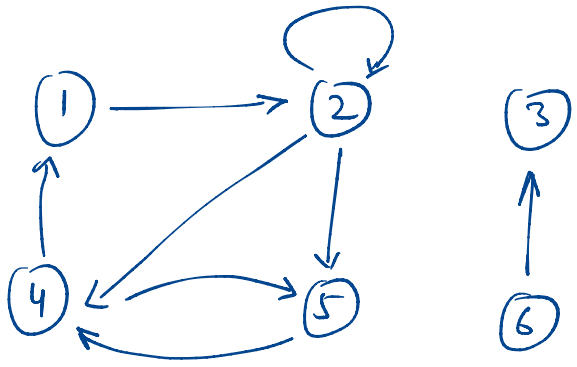
Graphs - I

- A directed graph (or digraph) G is a pair (V, E) where V is a finite set and E is a binary relation on V

  ↳ V is called the vertex set of G (and its elements are called vertices / nodes)

  ↳ E is called the Edge set of G (and its elements are called edges)

Example:



$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (2,2), (2,5), (2,4), (4,1), (4,5), (5,4), (6,3)\}$

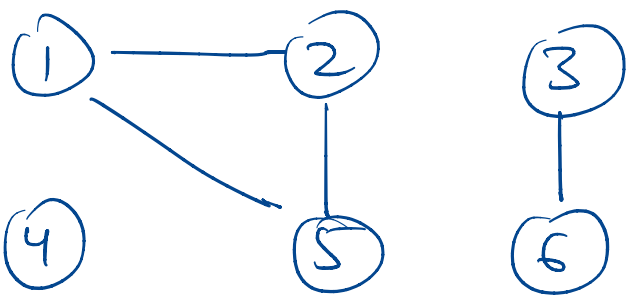
self loops are allowed.

A graph may contain multiple connected components.

→ In an Undirected graph $G = (V, E)$, the edge set E consists of unordered pairs of vertices, rather than ordered

↳ That is, an edge is a set $\{u, v\}$ where $u, v \in V$, and $u \neq v$. ⇒ (u, v) and (v, u) are the same

→ self loops are not allowed.



Isolated vertex ← 4

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{(1,2), (1,5), (2,5), (3,6), (2,1), (5,1), (5,2), (6,3)\}$

duplicates!

- In a digraph if (u,v) is an edge then (u,v) is incident from or leaves vertex u, and is incident to or enters vertex v

- If (u,v) is an edge of the graph $G=(V,E)$ then we say that vertex v is adjacent to vertex u.
  - ↳ For an undirected graph, the adjacency relationship is symmetric
  - ↳ In a digraph if v is adjacent to u, then it is written as $u \rightarrow v$

→ The degree of a vertex in an undirected graph is the number of edges incident on it.
  - ↳ A vertex whose degree is 0, is called an isolated vertex.

- In a directed graph, we define:
  - in-degree: # of incoming edges
  - out-degree: # of outgoing edges

  degree = in-degree + out-degree

Representation of graphs

Two standard ways → Adjacency list
→ Adjacency matrix

↳ Adjacency list provides a compact representation for sparse graphs — for which $|E|$ is much less than $|V|^2$

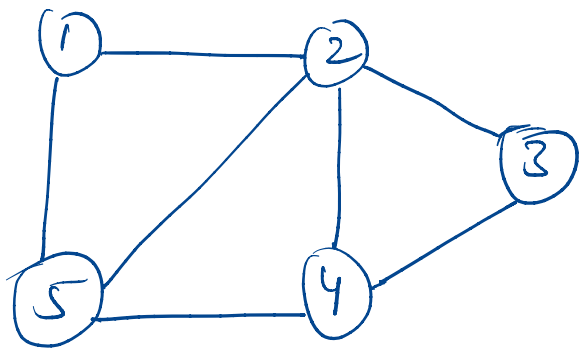
↳ Adjacency matrix : good for dense graphs, i.e, for which $|E|$ is close to $|V|^2$.

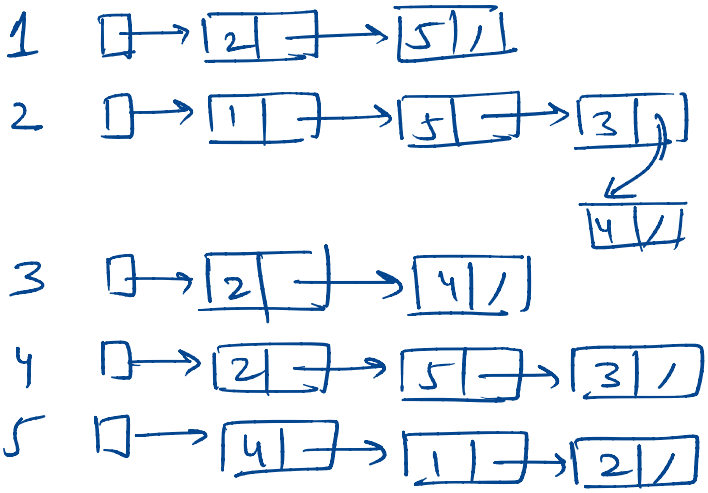
Adjacency list : Representation of a graph $G=(V,E)$

↳ Consists of an array Adj of $|V|$ lists, one for each vertex in V. For each $u \in V$, the adjacency list Adj[u] contains all vertices v such that an edge $(u,v) \in E$.

↳ i.e, Adj[u] contains all vertices adjacent to u in G.

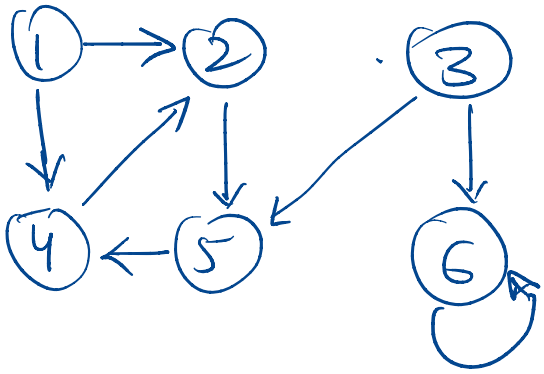
Example:

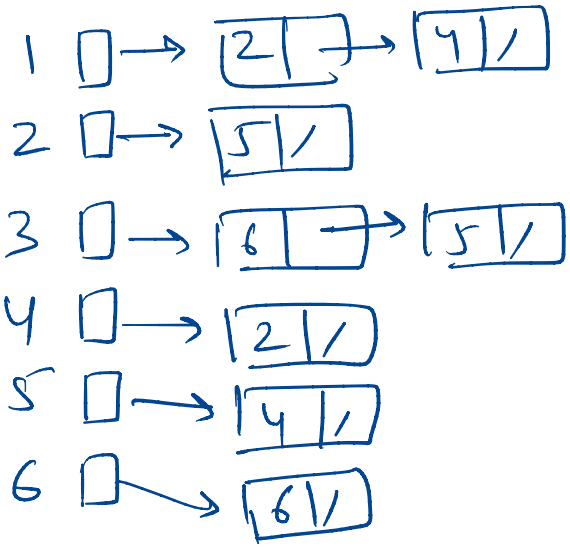




$\sum = 2|E|$

Ex. Directed graph





→ Memory requirement : $\Theta(|V|+|E|)$

→ Finding each edge takes $\Theta(V+E)$ time

→ Can be modified to represent a weighted graph, with weight $w: E \to \mathbb{R}$ associated with each edge, by storing $w(u,v)$ with every node of lists.

→ Disadvantage: No quick way to check if an edge (u,v) is present, other than to search for it.

Adjacency matrix representation

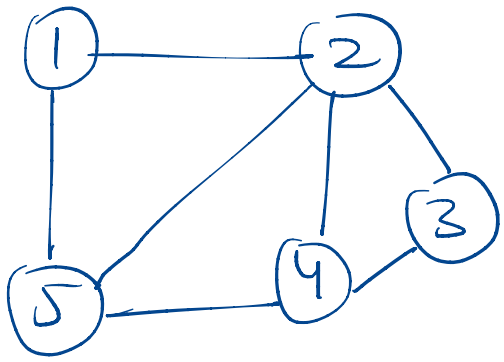
Assumes vertices are numbered $1, 2, \ldots, |V|$,

the adjacency matrix is a $|V| \times |V|$ matrix

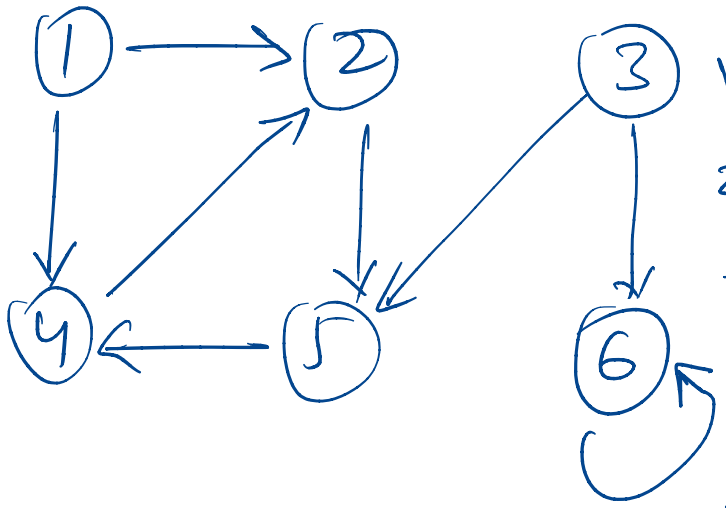
$$A = \{a_{ij}\}$$

$$a_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Examples:



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



	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

→ Requires $O(|V|^2)$ memory, independent of # of edges

→ for undirected graphs, A is symmetric

i.e, $A = A^T$

→ Can be modified to store a weighted graph in which case

$$a_{ij} = \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$
