CSI - 3105 Design & Analysis of Algorithms
Course 5

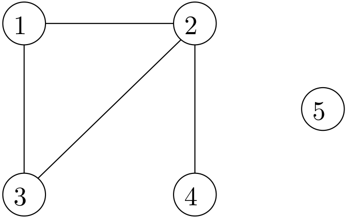
Jean-Lou De Carufel

Fall 2020

Chapter 3: Graph Algorithms

A *graph* G is made of a set V of *vertices* (or *nodes*) together with a set E of edges. We write $G = (V, E)$.

A graph is *undirected* if each edge in E is a pair $\{u, v\}$, where $u, v \in V$ and $u \neq v$.



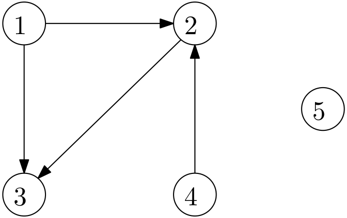
$$V = \{1, 2, 3, 4, 5\}$$

$$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}\}$$

Chapter 3: Graph Algorithms

A *graph* G is made of a set V of *vertices* (or *nodes*) together with a set E of edges. We write $G = (V, E)$.

A graph is *directed* is each edge in E is an **ordered** pair (u, v), where $u, v \in V$ and $u \neq v$.



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

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

Examples:

- A road map
- Facebook. Vertices are users. There is an edge $\{A, B\}$ if and only if A and B are “friends”.
- WWW. Vertices are web pages. There is a **directed** edge (A, B) if and only if A has a link to B.

Examples:

- Scheduling exams!

$$V = \text{set of all courses taught this term}$$

There is an edge $\{u, v\}$ if and only if there is at least one student taking both courses u and v.

Let C be the minimum number of colors needed such that

- each vertex gets one color.
- For each edge $\{u, v\}$, u and v have different colors.

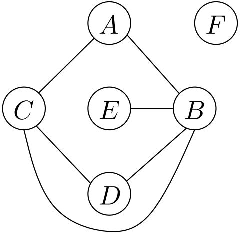
Then we can make an exam schedule with C time slots $1, 2, ..., C$: all vertices (i.e., courses) with color i have their exam in time slot i. In this way, there are no conflicts!

But computing C is very difficult...

In a graph $G = (V, E)$, two vertices $u, v \in V$ are *adjacent* if there is an edge between u and v.

A vertex $u \in V$ is *incident* to an edge $e \in E$ if one of the two vertices of e is u.

The *degree* of a vertex $u \in V$ is equal to the number of edges incident to u.



$V = \{A, B, C, D, E, F\}$

$E = \{\{A, B\}, \{A, C\}, \{B, C\}, \{B, D\}, \{B, E\}, \{C, D\}\}$

$\deg(B) = 4$

$\deg(E) = 1$

$\deg(F) = 0$

When the graph is oriented, the *outdegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the starting point of e. The *indegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the endpoint of e.

When the graph is oriented, the *outdegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the starting point of e. The *indegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the endpoint of e.

Theorem (Handshaking Lemma)

Let $G = (V, E)$ be a graph. then

$$\sum_{u \in V} \deg(u) = 2|E|.$$

PROOF: $\square$

When the graph is oriented, the *outdegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the starting point of e. The *indegree* of a vertex $u \in V$ is equal to the number of edges $e \in E$ such that u is the endpoint of e.

Theorem (Handshaking Lemma)

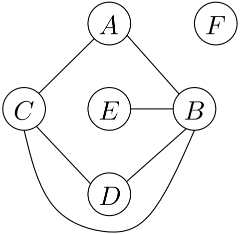
Let $G = (V, E)$ be a graph. then

$$\sum_{u \in V} \deg(u) = 2|E|.$$

PROOF: Each edge is counted twice! □

How to store a graph?

$G = (V, E) \qquad V = \{v_1, v_2, ..., v_n\}$

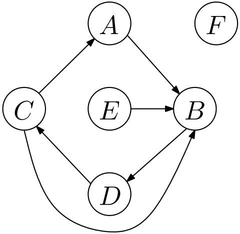


$$\begin{array}{c c} & \begin{matrix} A & B & C & D & E & F \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency matrix

How to store a graph?

$$G = (V, E) \qquad V = \{v_1, v_2, \dots, v_n\}$$

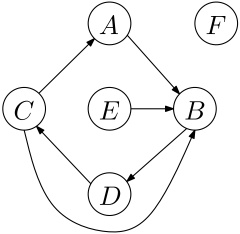


$$\begin{array}{c c} & \begin{matrix} A & B & C & D & E & F \end{matrix} \\ \begin{matrix} A \\ B \\ C \\ D \\ E \\ F \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency matrix

How to store a graph?

$$G = (V, E) \qquad\qquad V = \{v_1, v_2, \dots, v_n\}$$



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

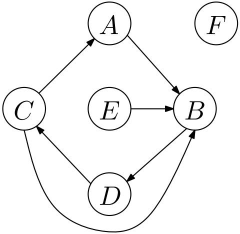
Adjacency matrix

Advantage:

- In $O(1)$ time, we can test if there is an edge between two given vertices.

How to store a graph?

$$G = (V, E) \qquad V = \{v_1, v_2, ..., v_n\}$$



$$\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 1 & 0 & 0 \\ C & 1 & 1 & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 1 & 0 & 0 & 0 \\ E & 0 & 1 & 0 & 0 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Adjacency matrix

Advantage:

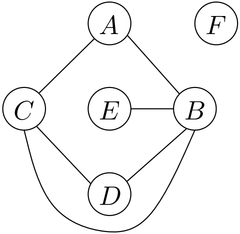
- In $O(1)$ time, we can test if there is an edge between two given vertices.

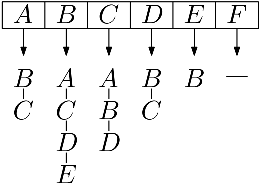
Disadvantage:

- Uses $\Theta(n^2)$ space for any graph.
- Finding all neighbours of a given vertex takes $O(n)$ time.

How to store a graph?

$$G = (V, E) \qquad V = \{v_1, v_2, \ldots, v_n\}$$

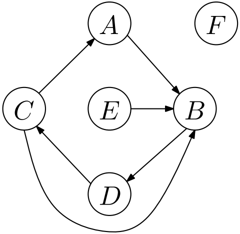


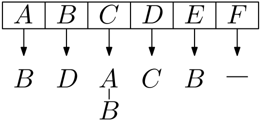


Adjacency list

How to store a graph?

$$G = (V, E) \qquad V = \{v_1, v_2, \dots, v_n\}$$

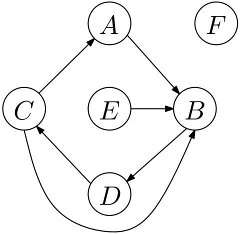


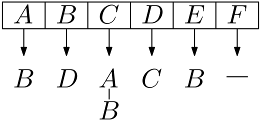


Adjacency list

How to store a graph?

$G = (V, E)$ $V = \{v_1, v_2, \dots, v_n\}$





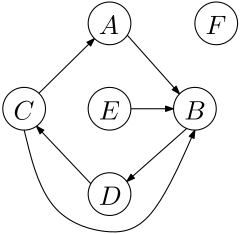
Adjacency list

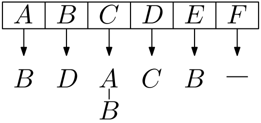
Advantage:

- Uses $\Theta(|V| + |E|)$ space.
- Finding all neighbours of a vertex $u \in V$ takes $O(1 + \deg(u))$ time.

How to store a graph?

$G = (V, E)$ $\qquad$ $V = \{v_1, v_2, \ldots, v_n\}$





Adjacency list

Advantage:

- Uses $\Theta(|V| + |E|)$ space.
- Finding all neighbours of a vertex $u \in V$ takes $O(1 + \deg(u))$ time.

Disadvantage:

- Testing if $\{u, v\}$ (or (u, v)) is an edge takes $O(1 + \deg(u))$ time.