

Lecture 10

Kinds of graphs. Data structures for graph representation.
Connectivity: The naïve algorithm and Warshall algorithm

Isabela Drămnesc UVT

Computer Science Department,
West University of Timișoara,
Romania

Remember that:

Graph = mathematical structure $G = (V, E)$ where

- V : set of **nodes** (or **vertices**)
- E : set of edges incident to 2 nodes, or 1 node

Depending on the kind of edges $e \in E$, graphs are of two kinds:

- ▶ **Undirected**: Every edge has one or two endpoints

Graphical representation: $a \xrightarrow{e} b$ or $\overset{e}{\curvearrowright} a$ (loop)

- ▶ **Directed** (or **digraphs**): Every edge $e \in E$ has a **source** (or **start**) end a **destination** (or **end**)

Graphical representation: $a \xrightarrow{e} b$ or $\overset{e}{\downarrow} a$ (loop)

The directed edges are called **arcs**.

Simple graph: graph (directed or not) with at most one arc between any pair of nodes, and no loop.

$$G = (V, E)$$

$e_1, e_2 \in E$ are **parallel** edges if they are incident to the same nodes, and

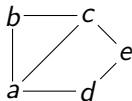
- If G is directed, then $start(e_1) = start(e_2)$ and $end(e_1) = end(e_2)$
- ▶ **Multigraph** (directed or not): no loops, and if the graph is
 undirected: it can have parallel edges
 directed: it can have parallel arcs
- ▶ **Pseudograph**: undirected graph that can have loops and parallel edges.
- ▶ **Weighted graph**: every edge $e \in E$ has a **weight** $w(e)$; usually $w(e) \in \mathbb{R}$.

Graphical representations of graphs

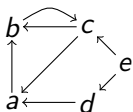
Illustrative examples

- ▶ **Simple graphs:** draw lines or arcs between the connected nodes

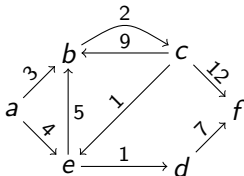
Simple **undirected** graph:



Simple **directed** graph:



- ▶ **Simple weighted graphs:** we indicate the weights along the corresponding connections

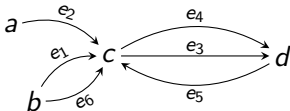


Graphical representations of graphs

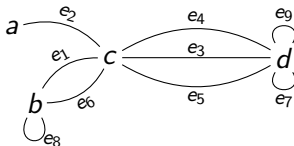
Example (continued)

Multigraphs or **pseudographs**: if we want to distinguish parallel edges, we can label them:

multigraph:



pseudograph:



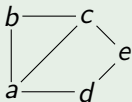
Concrete representations of graphs

- 1 List of nodes + list of edges
- 2 Adjacency lists
- 3 Adjacency matrix
- 4 Incidence matrix
- 5 Weight matrix

Simple graphs

The representation by list of nodes + list of edges

Example

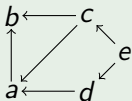


List of nodes $V = [a, b, c, d, e]$

List of edges $E = [\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{c, e\}, \{d, e\}]$

Remarks: $\{a, b\} = \{b, a\}$, $\{a, c\} = \{c, a\}$, a.s.o.

edge \leftrightarrow set of nodes adjacent to the edge



List of nodes $V = [a, b, c, d, e]$

List of arcs $E = [(a, b), (c, a), (c, b), (d, a), (e, c), (e, d)]$

Remarks: $(a, b) \neq (b, a)$, $(a, c) \neq (c, a)$, a.s.o.

edge \leftrightarrow pair (start, end)

Remark

If the graph has no isolated nodes (with 0 neighbors), there is no need to store the list of nodes V :

- ▶ V can be computed from E

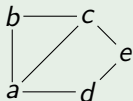
Simple graphs

The representation with adjacency lists

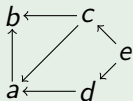
For every node $u \in V$ we keep the **list of nodes that are adjacent to u**

- ▶ If G is **undirected**, v is adjacent to u if there is an edge with endpoints u and v .
 - In undirected graphs, the relation of adjacency is symmetric.
- ▶ If G is **directed**, v is adjacent to u if there is an arc $e \in E$ from u to v , i.e. $start(e) = u$ and $end(e) = v$.

Example



$a \mapsto [b, c, d]$	$d \mapsto [a, e]$
$b \mapsto [a, c]$	$e \mapsto [c, d]$
$c \mapsto [a, b, e]$	



$a \mapsto [b]$	$d \mapsto [a]$
$b \mapsto []$	$e \mapsto [c, d]$
$c \mapsto [a, b]$	

Graphs

The representation with adjacency matrix A_G

If G has n nodes, $A_G = (m_{ij})$ has dimension $n \times n$ and
 $m_{ij} :=$ number of edges from the i -th node to the j -th node.

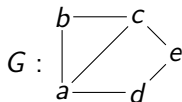
Remarks

- 1 Before computing M_G from G , be must fix an enumeration of its nodes: $[v_1, v_2, \dots, v_n]$
- 2 If G is undirected, A_G is a symmetric matrix
- 3 If G is a simple graph, A_G contains just 0s and 1s

Undirected graphs

The adjacency matrix A_G of an undirected graph G

The adjacency matrix the undirected graph



for the node enumeration $[a, b, c, d, e]$ is

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

Remark: the matrix A_G is symmetric.

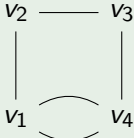
Graphs

The digraph corresponding to a symmetric adjacency matrix

If A is a symmetric matrix of size $n \times n$ with $a_{ij} \in \mathbb{N}$ for all i, j , an undirected graph G whose adjacency matrix is A can be built as follows:

- 1 Draw n points v_1, \dots, v_n in plane
- 2 For every $i, j \in \{1, \dots, n\}$, draw a_{ij} distinct edges between v_i and v_j

Example

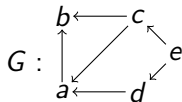
$$A = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{pmatrix} \Rightarrow G :$$


The diagram shows four vertices labeled v_1, v_2, v_3, v_4 arranged in a square. There are edges between v_1 and v_2 , v_1 and v_4 , v_2 and v_3 , and v_3 and v_4 . There are two curved lines between v_1 and v_4 , representing two distinct edges.

Digraphs

The adjacency matrix A_G of a digraph G

The adjacency matrix of the digraph



for the enumeration $[a, b, c, d, e]$ of nodes is

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

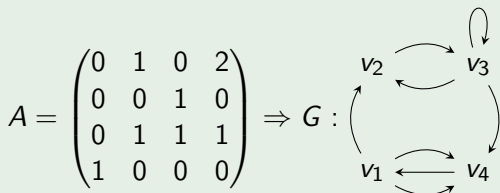
Digraphs

The digraph corresponding to an adjacency matrix

If A is an $n \times n$ matrix with $a_{ij} \in \mathbb{N}$ for all i, j , a digraph G whose adjacency matrix is A can be constructed as follows:

- 1 Draw n points v_1, \dots, v_n in the plane
- 2 For every $i, j \in \{1, \dots, n\}$, draw a_{ij} distinct arcs from v_i to v_j

Example



Digraphs with labeled edges

The representation with incidence matrices

We assume given two lists (or enumerations):

- $V = [v_1, \dots, v_n]$ of the nodes of G
- $L = [e_1, \dots, e_p]$ of the labels of edges of G

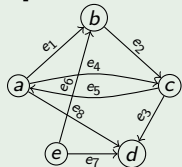
The **incidence matrix** $M_G = (m_{ij})$ has dimension $n \times p$ and

$$m_{ij} = \begin{cases} -1 & \text{if } \text{start}(e_j) = v_i \\ 1 & \text{if } \text{end}(e_j) = v_i \\ 0 & \text{otherwise.} \end{cases}$$

Example

If $V = [a, b, c, d, e]$, $L = [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8]$ and

$$M_G = \begin{pmatrix} -1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\ 1 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \end{pmatrix} \Rightarrow$$



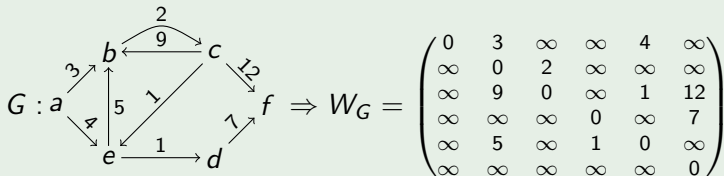
Simple weighted graphs

The representation with weight matrix

The **weight matrix** $W_G = (w_{ij})$ of a simple weighted graph G with n nodes $[v_1, \dots, v_n]$ has dimension $n \times n$ and

- ▷ $w_{ij} = 0$ for all $1 \leq i \leq n$.
- ▷ $w_{ij} = w(\{v_i, v_j\})$ for every edge $\{v_i, v_j\} \in E$, if G is undirected.
- ▷ $w_{ij} = w((v_i, v_j))$ for every arc $(v_i, v_j) \in E$, if G is directed.
- ▷ $w_{ij} = \infty$ otherwise.

Example (Weight digraph with node enumeration $[a, b, c, d, e, f]$)



The representation of graphs

Comparative study

- ▶ The representation with **list of edges**
 - Adequate for the representation of simple graphs without isolated nodes, and with $|E| \ll |V|$
 - Space complexity: $O(|E|)$
- ▶ The representation with **lists of adjacencies**
 - Allows the fast enumeration of the neighbors of a node
 - Space complexity: $O(|V| + |E|)$
- ▶ The representation with **adjacency matrix** $A_G = (a_{ij})$ or with **weight matrix** $W_G = (w_{ij})$
 - Fast test to check direct connection between 2 nodes: $O(1)$
 - $\nexists (v_i, v_j) \in E$ if $a_{ij} = 0$ or if $w_{ij} = \infty$
 - Space complexity: $O(|V|^2)$
 - inadequate representation when $|E| \ll |V|^2$
- The representation with incidence matrix M_G
 - ▶ Time complexity: $O(|V| \cdot |E|)$

Simple digraphs

Properties of the adjacency matrix A_G

We assume that G is a simple (di)graph with n nodes and with the adjacency matrix $A_G = (a_{ij})$

- $a_{ij} = 0$ (or false) if \nexists arc $v_i \rightarrow v_j$
- $a_{ij} = 1$ (or true) if \exists arc $v_i \rightarrow v_j$

We define:

- I_n : the identity matrix of size $n \times n$
- The Boolean operations \odot (conjunction) and \oplus (disjunction):

\odot	0	1
0	0	0
1	0	1

\oplus	0	1
0	0	1
1	1	1

Remarks:

$$a \odot b = \min(a, b)$$

$$a \oplus b = \max(a, b)$$

- If $U = (u_{ij})$, $V = (v_{ij})$ are $n \times n$ matrices with elements 0 or 1, we define
 - $U \oplus V = (c_{ij})$ if $c_{ij} = u_{ij} \oplus v_{ij}$ for all i, j
 - $U \odot V = (d_{ij})$ if $d_{ij} = (u_{i1} \odot v_{1j}) \oplus \dots \oplus (u_{in} \odot v_{nj})$
 - $U^k = \underbrace{U \odot \dots \odot U}_{k \text{ times}}$ for every $k > 0$

k times

Simple digraphs

Properties of the adjacency matrix A_G (contd.)

Properties

- 1 If $A_G^k = (a_{ij}^{(k)})$ for $k \geq 1$ then $a_{ij}^{(k)} = 1$ if and only if there is a path with length k from node v_i to v_j .
- 2 Let $A_G^* = I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^{n-1} = (\bar{a}_{ij})$. Then
 - $\bar{a}_{ij} = 1$ if and only if there is a path with length $j \in \{1, \dots, n-1\}$ from node v_i to v_j .
 - A_G^* can be computed in $O(n^4)$.
 - A_G^* is called **reflexive and transitive closure** of A_G .
- 3 v_i and v_j are connected $\Leftrightarrow \exists$ a simple path $v_i \rightsquigarrow v_j \Leftrightarrow \bar{a}_{ij} = 1$.
- 4 $I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^{k+1} = I_n \oplus (I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^k) \odot A_G$
 $\Rightarrow A^*$ can be computed in $O(n^4)$.

Simple digraphs

Properties of the adjacency matrix A_G (contd.)

Properties

- 1 If $A_G^k = (a_{ij}^{(k)})$ for $k \geq 1$ then $a_{ij}^{(k)} = 1$ if and only if there is a path with length k from node v_i to v_j .
- 2 Let $A_G^* = I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^{n-1} = (\bar{a}_{ij})$. Then
 - $\bar{a}_{ij} = 1$ if and only if there is a path with length $j \in \{1, \dots, n-1\}$ from node v_i to v_j .
 - A_G^* can be computed in $O(n^4)$.
 - A_G^* is called **reflexive and transitive closure** of A_G .
- 3 v_i and v_j are connected $\Leftrightarrow \exists$ a simple path $v_i \rightsquigarrow v_j \Leftrightarrow \bar{a}_{ij} = 1$.
- 4 $I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^{k+1} = I_n \oplus (I_n \oplus A_G \oplus A_G^2 \oplus \dots \oplus A_G^k) \odot A_G \Rightarrow A^*$ can be computed in $O(n^4)$.

Corollary

The connectivity between all pairs of nodes in a simple digraph can be checked in $O(n^4)$.

Simple digraphs

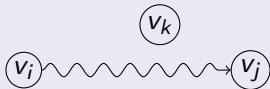
Better ways to compute A_G^*

Warshall algorithm computes A_G^* in $O(n^3)$.

Warshall's main idea

If $V = [v_1, \dots, v_n]$ is an enumeration of the nodes of G and $v_k \in V$, then every simple path $\pi : v_i \rightsquigarrow v_j$ is of one of the following two kinds:

- 1 v_k does not appear along π between v_i and v_j



- 2 v_k appears exactly once in π , between v_i and v_j



Simple digraphs

Warshall algorithm to compute A_G^*

Assumption: $A_G = (a_{ij})$ has size $n \times n$

- ▶ Compute recursively $C^{[n]} = (c_{ij}^{[n]})$ where

$$c_{ij}^{[k]} := \begin{cases} a_{ij} & \text{if } k = 0 \\ c_{ij}^{[k-1]} \oplus (c_{ik}^{[k-1]} \odot c_{kj}^{[k-1]}) & \text{if } k \geq 1 \end{cases}$$

Properties

- 1 $C^{[0]} = A_G$
- 2 $c_{ij}^{[k]} = 1$ if and only if there is a path $\pi : v_i \rightsquigarrow v_j$ where all intermediary nodes are from the subset $\{v_1, \dots, v_k\}$
- 3 $C^{[n]} = A_G^*$
- 4 $C^{[n]}$ can be computed in $O(n^3)$.

Simple weighted digraphs

The lightest paths

Consider the simple digraph $G = (V, E)$ with $V = \{1, \dots, n\}$ and the weight function $w : E \rightarrow \mathbb{R}^+$

- ▶ In G , the **weight** of a path $\pi : v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_p$ is
 - $w(\pi) = \sum_{i=1}^{p-1} w((v_i, v_{i+1}))$
 - (we add up the weights of all arcs of π)
- ▶ For every pair of nodes (i, j) from V , we wish to find
 - a lightest path from node i to node j
(there can be more than one lightest path)
 - the weight of a lightest path

Remember that the **weight matrix** of G is $W_G = (w_{ij})$ where

$$w_{ij} = \begin{cases} 0 & \text{if } j = i, \\ w((i, j)) & \text{if } (i, j) \in E, \\ \infty & \text{if } (i, j) \notin E \end{cases}$$

Finding the lightest paths in a simple weighted digraph

Warshall algorithm: main idea

Let $k \in V = \{1, \dots, n\}$, and $\pi : i \rightsquigarrow j$ be a lightest path from i to j . We distinguish two cases:

- 1 k is not an intermediary node of π
- 2 k is an intermediary node of π . Then $\pi = i \xrightarrow{\pi_1} k \xrightarrow{\pi_2} j$ and $w(\pi) = w(\pi_1) + w(\pi_2)$.

Finding the lightest paths in a simple weighted digraph

Warshall algorithm: main idea

Let $k \in V = \{1, \dots, n\}$, and $\pi : i \rightsquigarrow j$ be a lightest path from i to j . We distinguish two cases:

- 1 k is not an intermediary node of π
- 2 k is an intermediary node of π . Then $\pi = i \xrightarrow{\pi_1} k \xrightarrow{\pi_2} j$ and $w(\pi) = w(\pi_1) + w(\pi_2)$.

An useful auxiliary data structure:

- A matrix of paths $P^{[k]} = (p_{ij}^{[k]})$ where

$$p_{ij}^{[k]} := \begin{cases} \bullet & \text{special value: } \nexists \text{ path } i \rightsquigarrow j \text{ through} \\ & \text{intermediary nodes from } \{1, \dots, k\} \\ \pi & \text{a lightest path from } i \text{ to } j \text{ through} \\ & \text{intermediary nodes from } \{1, \dots, k\} \end{cases}$$

Finding the lightest paths in a simple weighted digraph

Warshall algorithm: the recursive formula of computation

We assume that $W_G = (w_{ij})$

$$p_{ij}^{[k]} := \begin{cases} \bullet & \text{if } k = 0 \text{ and } w_{ij} = \infty \\ [i, j] & \text{if } k = 0 \text{ and } w_{ij} < \infty \\ p\text{-min}(p_{ij}^{[k-1]}, p_{ik}^{[k-1]} \succcurlyeq p_{kj}^{[k-1]}) & \text{if } k \geq 1 \end{cases}$$

where $p_{ik}^{[k-1]} \succcurlyeq p_{kj}^{[k-1]}$ denotes the concatenation of the paths $p_{ik}^{[k-1]}$ and $p_{kj}^{[k-1]}$. If both exist (i.e., they are not \bullet), and \bullet otherwise.

Remark

If $p_{ij}^{[n]} = \bullet$, there is no path from node i to j . Otherwise, $p_{ij}^{[n]}$ is a lightest path from node i to j .

Finding the lightest paths in a simple weighted digraph

Warshall algorithm: pseudocode

Compute recursively and simultaneously two matrices:

$$W^{[k]} = \left(w_{ij}^{[k]} \right) \quad \text{and} \quad P^{[k]} = \left(p_{ij}^{[k]} \right) \quad \text{for } 0 \leq k \leq n$$

$$w_{ij}^{[k]} = \begin{cases} w_{ij} & \text{if } k = 0, \\ \max \left(w_{ij}^{[k-1]}, \min \{ w_{ik}^{[k-1]}, w_{kj}^{[k-1]} \} \right) & \text{otherwise.} \end{cases}$$

$$p_{ij}^{[k]} = \begin{cases} \bullet & \text{if } k = 0 \text{ and } w_{ij}^{[0]} = \infty, \\ [i, j] & \text{if } k = 0 \text{ and } w_{ij}^{[0]} \neq \infty, \\ w_{ij}^{[k-1]} & \text{if } k > 0 \text{ and } w_{ij}^{[k-1]} \leq w_{ik}^{[k-1]} + w_{kj}^{[k-1]}, \\ w_{ik}^{[k-1]} \asymp w_{kj}^{[k-1]} & \text{otherwise.} \end{cases}$$

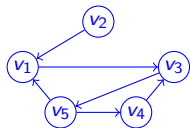
Remark

$W^{[n]}$ and $P^{[n]}$ can be computed in $O(n^3)$.

A special case of Warshall algorithm

Finding the shortest paths in a simple digraph

All arcs are assumed to have weight 1



$$P_G^{[0]} = \begin{pmatrix} \bullet & \bullet & [v_1, v_3] & \bullet & \bullet \\ [v_2, v_1] & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & [v_3, v_5] \\ \bullet & \bullet & [v_4, v_3] & \bullet & \bullet \\ [v_5, v_1] & \bullet & \bullet & [v_5, v_4] & \bullet \end{pmatrix},$$

$$P_G^{[1]} = \begin{pmatrix} \bullet & \bullet & [v_1, v_3] & \bullet & \bullet \\ [v_2, v_1] & \bullet & [v_2, v_1, v_3] & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & [v_3, v_5] \\ \bullet & \bullet & [v_4, v_3] & \bullet & \bullet \\ [v_5, v_1] & \bullet & [v_5, v_1, v_3] & [v_5, v_4] & \bullet \end{pmatrix},$$

$$P_G^{[2]} = P_G^{[1]}, \quad P_G^{[3]} = \begin{pmatrix} \bullet & \bullet & [v_1, v_3] & \bullet & [v_1, v_3, v_5] \\ [v_2, v_1] & \bullet & [v_2, v_1, v_3] & \bullet & [v_2, v_1, v_3, v_5] \\ \bullet & \bullet & \bullet & \bullet & [v_3, v_5] \\ \bullet & \bullet & [v_4, v_3] & \bullet & [v_4, v_3, v_5] \\ [v_5, v_1] & \bullet & [v_5, v_1, v_3] & [v_5, v_4] & \bullet \end{pmatrix}, \quad P_G^{[4]} = P_G^{[3]},$$

$$P_G^{[5]} = \begin{pmatrix} [v_1, v_3, v_5, v_1] & \bullet & [v_1, v_3] & [v_1, v_3, v_5, v_4] & [v_1, v_3, v_5] \\ [v_2, v_1] & \bullet & [v_2, v_1, v_3] & [v_2, v_1, v_3, v_5, v_4] & [v_2, v_1, v_3, v_5] \\ [v_3, v_5, v_1] & [v_3, v_5, v_1, v_3] & [v_3, v_5, v_1, v_3] & \bullet & [v_3, v_5] \\ [v_4, v_3, v_5, v_1] & \bullet & [v_4, v_3] & [v_4, v_3, v_5, v_4] & [v_4, v_3, v_5] \\ [v_5, v_1] & \bullet & [v_5, v_1, v_3] & [v_5, v_4] & [v_5, v_1, v_3, v_5] \end{pmatrix}$$