Scientific tools for decision making

Sonia Toubaline

Contact: sonia.toubaline@dauphine.fr
Lecture 1-1

General introduction & Graph theory
**Presentation**

Management Science & Operational Research

**Definition**

*Scientific* approach aiming at supporting *decisions* within organizations (firms, companies and more generally any social organization)

**Objective**

*Model* certain types of problems using mathematical tools and *solve* them with known and automated solution methods
A discipline at the crossroad of well-established scientific fields.
**FIELDS OF APPLICATIONS**

- Production planning (e.g. optimal use of resources to produce goods)
- Logistics and distribution (e.g. how to construct delivery tours?)
- Portfolio management
- Inventory management
- Selection of candidates for a position
- ...

More generally, any decision problem which is both difficult AND with a high stake.
Difficulties

Three main types

- Decisions with a combinatorial structure
- Decisions under uncertainty
- Decisions with multiple criteria
PROBLEMS WITH THE THREE DIFFICULTIES

Example: Portfolio management

- A portfolio is a combination of securities
- Their evaluation is uncertain
- At least two conflicting criteria: profitability vs risk
**Example: Assignment problem**

Assign $n$ workers (resources, machines) to perform $n$ tasks. $a_{ij}$: ability of worker $i$ to perform task $j$ evaluated on a 0-10 scale, $i, j = 1, \ldots, n$.

**Objective:** determine an optimal assignment that maximizes the global performance
## Combinatorial Problems

**Example: Assignment problem**

Assign $n$ workers (resources, machines) to perform $n$ tasks. $a_{ij}$: ability of worker $i$ to perform task $j$ evaluated on a 0-10 scale, $i, j = 1, \ldots, n$.

**Objective:** determine an optimal assignment that maximizes the global performance

**First idea:** Enumerate all the possibilities

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1$</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>$W_2$</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Two possible assignments:

- $W_1 \ T_1$ and $W_2 \ T_2 : 6+8=14$
- $W_1 \ T_2$ and $W_2 \ T_1 : 9+7=16$
**Example: Assignment problem**

Assign $n$ workers (resources, machines) to perform $n$ tasks. 

$a_{ij}$: ability of worker $i$ to perform task $j$ evaluated on a 0-10 scale, $i, j = 1, \ldots, n$.

**Objective:** determine an optimal assignment that maximizes the global performance

**First idea:** Enumerate all the possibilities

<table>
<thead>
<tr>
<th></th>
<th>$T_1$</th>
<th>$T_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1$</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>$W_2$</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Two possible assignments:

- $W_1$ $T_1$ and $W_2$ $T_2$: $6+8=14$
- $W_1$ $T_2$ and $W_2$ $T_1$: $9+7=16$

More generally, the number of all possible assignment is $n!$

$n = 20 \rightarrow 20! \approx 2.4329 \times 10^{18}$

Computer: 1 000 (new) solutions/second $\rightarrow$ 770 000 centuries!
**Problems with uncertainties**

**Example:** In a group of persons, what is the probability that at least two have the same birthday date (not necessarily the same year).

\[ p(x) = 1 - p'(x) \]

\[ p'(x): \text{no person has the same birthday date among} \ n \ \text{persons} \]

\[ p'(x) = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \ldots \times \frac{365-(n-1)}{365} \]

(Hypo.: uniform repartition of the birthday during the year)

<table>
<thead>
<tr>
<th>( n )</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(x) )</td>
<td>0.117</td>
<td>0.411</td>
<td>0.706</td>
<td>0.891</td>
<td>0.97</td>
<td>0.994</td>
<td>0.999</td>
</tr>
</tbody>
</table>
PROBLEMS WITH MULTIPLE CRITERIA

Example: Portfolio management

Two conflicting criteria: profitability and risk

→ Many possible candidates (non-dominated ones)

→ To select among these it is necessary to take account of the DM’s preferences.
THE DECISION SUPPORT PROCESS

Presentation

Real problem

Fuzzy, uncertain context
THE DECISION SUPPORT PROCESS

Presentation

Abstract representation
Ex: LP, graph Theory

Model

Modelling

Real problem

Fuzzy, uncertain context
THE DECISION SUPPORT PROCESS

Presentation

Abstract representation
Ex: LP, graph Theory

Model

Solving

Solution

Modelling

Real problem

Fuzzy, uncertain context
THE DECISION SUPPORT PROCESS

Presentation

Abstract representation
Ex: LP, graph Theory

Model

- Solving

Solution

- Interpretation

Decision

- Fuzzy, uncertain context

Real problem

Modelling
THE DECISION SUPPORT PROCESS

Presentation

Abstract representation
Ex: LP, graph Theory

Model

Solving

Solution

Modelling

Real problem

Fuzzy, uncertain context

Implementation
Application

Interpretation

Decision
Definition: A **model** is a more or less abstract **representation** of a situation (decision situation in our context) or an object.

Two types:

- **Concrete model**
  physical device which mimics the original object (e.g. a plane model)

- **Abstract model**
  mathematical model (e.g. set of equations and inequations, a graph)

A good model emphasizes the aspects to be studied and eliminates the irrelevant aspects.
COMPONENTS OF A MODEL

Inputs

- External: "parameters" → uncontrollable inputs
- Internal: "decision variables" → controllable inputs

Outputs

- Measure of quality of the solution: Objective function(s)

Example: An inventory problem

A manager must decide how to manage the stock of a given product $P$ considering a demand each month over a time horizon $T$
COMPONENTS OF A MODEL

Decision variables

- $x_t$: quantity of product $P$ to be ordered at the beginning of the period $t$, $t = 1, \ldots, T$
- $s_t$: level of the stock at the end of period $t$, $t = 1, \ldots, T$

Parameters

- $d_t$: demand of product $P$ for period $t$, $t = 1, \ldots, T$
- $c_t$: purchase cost of product $P$ at period $t$, $t = 1, \ldots, T$

When parameter are known with certainty $\rightarrow$ deterministic model
otherwise $\rightarrow$ stochastic model

The consistency of the system and restrictions are expressed by constraints.
**Mathematical formulation - Inventory problem**

$$\begin{align*}
\text{min} & \quad \sum_{t=1}^{T} c_t x_t \\
\text{s.t.} & \quad s_{t-1} + x_t = d_t + s_t \quad t = 1, \ldots, T \\
& \quad s_0 = 0 \\
& \quad s_T = 0 \\
& \quad s_t \leq C \\
& \quad x_t \geq 0, s_t \geq 0 \quad t = 1, \ldots, T
\end{align*}$$

\(C\): capacity of the warehouse

**Objective:** Min. total purchase cost
**Objectives of this course**

- To acquire the fundamental concepts for modelling and solving decision problems arising in socio-economic organizations.
- Eventually, to become a specialist in the field (OR researcher, consultant, . . .)
- Or at least, to be able to discuss with these specialists.

**Content**
Presentation of three classical frameworks for modelling and solving decision problems

- Graphs
- Linear programming
- Decision theory

**Assessment**
Participation 20% – Midterm exam 30% – Final Exam 50%
Graphs
Definitions

First definition

A graph $G$ is a schema constituted of:

- A set $X$ of points called nodes (or vertices)
- A set $U$ of arrows (or lines) called arcs (edges) linking these nodes

We note $G = (X, U)$ with $|X| = n$ and $|U| = m$
Example: Oriented graph

We say $a$ and $b$ are **adjacent**, and $a$ (and $b$) is **incident** to $(a, b)$.
**DEFINITIONS**

**Example Non oriented graph**

We say $a$ and $b$ are **adjacent**, and $a$ (and $b$) is **incident** to $(a, b)$.

![Graph with nodes a, b, and c connected by edges](image-url)
DEFINITIONS

- A graph is entirely defined by giving $X$ and $U$

Previous example: Oriented graph $G = (X, U)$ with $X = \{a, b, c\}$ and $U = \{(a, a), (a, b), (b, c), (c, b)\}$

- A graphical representation of a graph does not have a significance
**DEFINITIONS**

**Other definitions**

A graph $G$ is defined by:

- A set of nodes $X$
- A "multivoque" set $\Gamma : X \to X$

(A multivoque application from $E$ to $E'$ is a process that associates to any element of $E$ none, one or several elements of $E'$)

$\Gamma$ associates to each element $x \in X$ the set of its successors.

**Previous example:** Oriented graph

$X = \{a, b, c\}$ and $\Gamma(a) = \{a, b\}$, $\Gamma(b) = \{c\}$, $\Gamma(c) = \{b\}$

We note $G = (X, \Gamma)$
Definitions

Graph $G$

$\Gamma(a)$

$\Gamma(c)$

$\Gamma(b)$
Other definitions

A graph $G$ is defined by:

- A set of nodes $X$
- A "multivoque" set $\Gamma^{-1}: X \to X$

$\Gamma^{-1}$ associates to each element $x \in X$ the set of its predecessors.

**Previous example:** Oriented graph

$X = \{a, b, c\}$ and $\Gamma^{-1}(a) = \{a\}, \Gamma^{-1}(b) = \{a, c\}, \Gamma^{-1}(c) = \{b\}$

We note $G = (X, \Gamma^{-1})$
DEFINITIONS

Graph $G$

$\Gamma^{-1}(a)$

$\Gamma^{-1}(b)$

$\Gamma^{-1}(c)$
SOME GRAPH REPRESENTATIONS

Graphical representation

- The one considered so far
- Good visualisation but not implementable on computer

Example:
Some graph representations

Matrix representation

**Adjacency matrix** (nodes nodes):

$n \times n$ matrix whose general term is

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

$$M = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$
**Some Graph Representations**

**Matrix representation**

**Incidence matrix (nodes arcs):**

An \( n \times m \) matrix whose general term is

\[
d'_{ij} = \begin{cases} 
1 & \text{if } i \text{ is the initial extremity node of } u_j \\
-1 & \text{if } i \text{ is the terminal extremity node of } u_j \\
0 & \text{otherwise.}
\end{cases}
\]

\[
\begin{array}{l}
u_1 \\
u_2 \\
u_3 \\
u_4
\end{array} = \begin{bmatrix}
1 & -1 & 1 & 0 \\
0 & 0 & -1 & 1 \\
-1 & 1 & 0 & -1
\end{bmatrix}
\]

\[
M' = \begin{bmatrix}
1 & -1 & 1 & 0 \\
0 & 0 & -1 & 1 \\
-1 & 1 & 0 & -1
\end{bmatrix}
\]
Some graph representations

Matrix representation

Incidence matrix (nodes edges):

An $n \times m$ matrix whose general term is

$$\forall i \in X, u_j \in U \quad a_{ij}' = \begin{cases} 
1 & \text{if } i \text{ is an extremity node of } u_j \\
0 & \text{otherwise.}
\end{cases}$$

$$M' = \begin{bmatrix}
1 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1
\end{bmatrix}$$
Some graph representations

List representation

Forward adjacency list: consists on the description of $\Gamma$

Example:

\[
\begin{array}{c|c}
 x & \Gamma(x) \\
\hline
 a & b, c \\
 b & c \\
 c & a \\
\end{array}
\]
Some graph representations

List representation

**Backward adjacency list**: consists on the description of $\Gamma^{-1}$

Example:

\[
\begin{array}{c|c}
 x & \Gamma^{-1}(x) \\
 a & c \\
 b & a \\
 c & a,b \\
\end{array}
\]
GRAPH MODELLING: SOME EXAMPLES
Problem: route all the messages, task assignment and process
COMMUNICATION NETWORKS

Graph modelling

Entities: computers, cables, switches, …
COMMUNICATION NETWORKS

Graph modelling

Entities: computers, cables, switches, ...
Nodes: computers, switches ⇒ \( X \) set of all the nodes
COMMUNICATION NETWORKS

Graph modelling
Entities: computers, cables, switches, ...
Nodes: computers, switches $\Rightarrow X$ set of all the nodes
Edges: cables
  2 extremity nodes and connection between them $\Rightarrow U$ set of all the edges
**COMMUNICATION NETWORKS**

Graph modelling

Entities: computers, cables, switches, ...
Nodes: computers, switches \( \Rightarrow X \) set of all the nodes
Edges: cables

2 extremity nodes and connection between them \( \Rightarrow U \) set of all the edges

\( \Rightarrow \) non oriented graph \( G = (X, U) \)
Problems: commodity delivery, task scheduling, shortest path, minimise the cost, minimise the ending time of the delivery, ...
Graph modelling

Entities: trucks, houses, factories, depot centres, warehouses, roads, ...
Graph modelling

Entities: trucks, houses, factories, depot centres, warehouses, roads, ... 
Nodes: different locations ⇒ X set of all the nodes
Graph modelling

Entities: trucks, houses, factories, depot centres, warehouses, roads, ...  
Nodes: different locations ⇒ $X$ set of all the nodes  
Arcs: roads  
  - 2 extremity nodes and connection between them  
  - Initial extremity node, terminal extremity node  
⇒ $U$ set of all the arcs
Transport Networks

Graph modelling

Entities: trucks, houses, factories, depot centres, warehouses, roads, ...
Nodes: different locations ⇒ $X$ set of all the nodes
Arcs: roads
  - 2 extremity nodes and connection between them
  - Initial extremity node, terminal extremity node
⇒ $U$ set of all the arcs
⇒ an oriented graph $G = (X, U)$
Parisian Underground
London Underground
Graph theory

–

Basic notions
**Partial graphs and subgraphs**

Graph $G=(X,U)$

Partial graph $G^p=(X,U^p)$

Diagram showing overlapping and distinct nodes and edges.
**Partial graphs and subgraphs**

Graph $G=(X,U)$

Partial graph $G^p=(X,U^p)$

Subgraph $G'=(X',U')$
Partial graphs and subgraphs

Graph $G=(X,U)$

Partial graph $G^p=(X, U^p)$

Subgraph $G'=(X', U')$

Partial subgraph $G^{ps}=(X^{ps}, U^{ps})$
**Chain - Cycle**

A **chain** of length $q$ is a sequence of $q$ arcs $\mu = [u_1 u_2 \ldots u_q]$ s.t. each arc $u_i$ ($i=2, \ldots, q-1$) has a common extremity with $u_{i-1}$ and the other with $u_{i+1}$

**Example:** $\mu = [u_1 u_2 u_6]$ is a chain of length 3

A **cycle** is a chain whose extremities (initial and end) coincide.

**Example:** $\mu = [u_3 u_6 u_5 u_4]$ ($x_3, x_4, x_6, x_5, (x_3)$)

The notions of chain & cycle are transportable to the non oriented case.
A path is a chain $\mu = [u_1 u_2 \ldots u_q]$ s.t. each arc $u_i$ ($i = 1, \ldots, q - 1$) has its terminal extremity that coincides with the initial extremity of $u_{i+1}$. It is a chain where all arcs are oriented in the same direction.

Example: $\mu = [u_2 u_3 u_7 u_6] (x_1, x_4, x_3, x_6, x_4)$

A circuit is a cycle $\mu = [u_1 u_2 \ldots u_q]$ s.t. $\forall u_i$, $i = 1, \ldots, q - 1$, the terminal extremity of $u_i$ coincides with the initial extremity of $u_{i+1}$. It is a path whose extremities coincide.

Example: $\mu = [u_4 u_7 u_5] (x_5, x_3, x_6, (x_5))$
Properties

A chain (path) is said

- **Elementary**: if it passes only once through each of its own nodes
- **Simple**: if it passes only once through each of its own edges (arcs)
  
  Elementary $\Rightarrow$ Simple
- **Hamiltonian**: if it passes once and only once through every node of the graph
- **Pre-Hamiltonian**: if it passes at least once through every node of the graph
- **Eulerian**: if it passes once and only once through every edge (arc) of the graph
- **Pre-Eulerian**: if it passes at least once through every edge (arc) of the graph
Ascendant, descendant, root, antiroot

- A node $x$ is **ascendant** of a node $y$ if there exists a path linking $x$ to $y$.
- A node $x$ is **descendant** of a node $y$ if there exists a path linking $y$ to $x$.
- $x$ is a **root** of $G = (X, U)$ if $\forall y \neq x$, $x$ is ascendant of $y$.
  There exists a path from $x$ to every other node of $G$.
- $x$ is an **anti-root** of $G = (X, U)$ if $\forall y \neq x$, $x$ is descendant of $y$.
  There exists a path from every node $y$ of $G$ to $x$, $y \neq x$.
**Connectivity**

- $G$ is **connected** if for any couple of nodes $x$ and $y$, there exists a chain linking $x$ and $y$.

**Examples**

Connected

Not connected
**Strong connectivity**

- \( G \) is **strongly connected** if for any couple of nodes \( x \) and \( y \), there exists a path from \( x \) to \( y \) (and then from \( y \) to \( x \)).

**Examples**

![Graph 1](image1.png)

**Strongly connected**

![Graph 2](image2.png)

**Not strongly connected**
Simple algorithms
**Algorithm to detect circuits of a graph**

**Remark:** A node without successors or predecessors cannot be part of a circuit

**Algorithm**

(0) Represent $G$ by its forward (backward) adjacency list

while (possible) do

(1) Find a node with an empty list

(2) Delete this node wherever it appears in the table (list)

--- At the END ---

if all the nodes are deleted then no circuit exists in $G$

otherwise There exists at least one circuit in $G$
Algorithm to detect circuits of a graph

Example

Graph $G$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Algorithm to detect circuits of a graph

Example

Graph $G$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Algorithm to detect circuits of a graph

Example

Graph $G$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Algorithm to detect circuits of a graph

Example

Graph $G$

\[
\begin{array}{c|c}
 x_i & \Gamma(x_i) \\
\hline
 x_1 & x_2, x_3 \\
 x_2 & x_3, x_5 \\
 x_3 & x_5, x_6 \\
 x_4 & x_2, x_7 \\
 x_5 & x_4, x_6, x_7 \\
 x_6 & x_7, x_8 \\
 x_7 & x_8 \\
 x_8 & - \\
\end{array}
\]
Algorithm to detect circuits of a graph

Example

Graph $G$

$\begin{array}{c|c}
  x_i & \Gamma(x_i) \\
  \hline
  x_1 & x_2, x_3 \\
  x_2 & x_3, x_5 \\
  x_3 & x_5, x_6 \\
  x_4 & x_2, x_7 \\
  x_5 & x_4, x_6, x_7 \\
  x_6 & x_7, x_8 \\
  x_7 & x_8 \\
  x_8 & - \\
\end{array}$

END $\Rightarrow \exists$ at least one circuit
Algorithm to detect circuits of a graph

Example

Graph $G$

$\begin{align*}
\text{Graph } G \quad & \quad \text{Table: } \quad \begin{array}{c|c}
  x_i & \Gamma(x_i) \\
  \hline
  x_1 & x_2, x_3 \\
  x_2 & x_3, x_5 \\
  x_3 & x_5, x_6 \\
  x_4 & x_2, x_7 \\
  x_5 & x_4, x_6, x_7 \\
  x_6 & x_7, x_8 \\
  x_7 & x_8 \\
  x_8 & - \\
\end{array} \\
\end{align*}$

END $\Rightarrow \exists$ at least one circuit

$x_1, x_2, x_3, x_5, x_4, x_2 \Rightarrow \text{Circuit } C$
Algorithm to check if a node $x_i$ is a root in $G$

**Idea:** Test whether we can reach any node from $x_i$

**Algorithm**

(0) Represent $G$ by its forward adjacency list

(1) Mark $x_i$

Repeat (while possible)

(2) Mark any unmarked node which is in the list of (i.e. successor of) a marked node

--- At the END ---

if all the nodes are marked then $x_i$ is a root of $G$

Otherwise $x_i$ is not a root of $G
Algorithm to check if a node $x_i$ is a root in $G$.

Example

Graph $G$ and $x_i = x_1$

![Graph G with nodes $x_1$ to $x_8$ connected by edges.]

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td></td>
</tr>
</tbody>
</table>
Algorithm to check if a node $x_i$ is a root in $G$

Example

Graph $G$ and $x_i = x_1$

$\Gamma(x_i) = \{x_2, x_3\}$

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
**Algorithm to check if a node \( x_i \) is a root in \( G \)**

**Example**

Graph \( G \) and \( x_i = x_1 \)

\[
\begin{array}{c|cc}
\hline
x_i & \Gamma(x_i) \\
\hline
(1) \star & x_1 & x_2, x_3 \\
(2) \star & x_2 & x_3, x_5 \\
(2) \star & x_3 & x_5, x_6 \\
& x_4 & x_2, x_7 \\
& x_5 & x_4, x_6, x_7 \\
x_6 & x_7, x_8 \\
x_7 & x_8 \\
x_8 & - \\
\end{array}
\]
Algorithm to check if a node $x_i$ is a root in $G$

Example

Graph $G$ and $x_i = x_1$

\[ (1) \ast x_1 \rightarrow x_2, x_3 \]
\[ (2) \ast x_2 \rightarrow x_3, x_5 \]
\[ (2) \ast x_3 \rightarrow x_5, x_6 \]
\[ (3) \ast x_5 \rightarrow x_4, x_6, x_7 \]
\[ (3) \ast x_6 \rightarrow x_7, x_8 \]
\[ x_7 \rightarrow x_8 \]

\[ x_8 \rightarrow \]
**Algorithm to Check If a Node $x_i$ is a Root in $G$**

**Example**

Graph $G$ and $x_i = x_1$

![Graph Diagram]

<table>
<thead>
<tr>
<th>$x_i$</th>
<th>$\Gamma(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2, x_3$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_3, x_5$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_5, x_6$</td>
</tr>
<tr>
<td>$x_4$</td>
<td>$x_2, x_7$</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$x_4, x_6, x_7$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7, x_8$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_8$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$-$</td>
</tr>
</tbody>
</table>
Algorithm to check if a node $x_i$ is a root in $G$

Example

Graph $G$ and $x_i = x_1$

$\Rightarrow x_1$ is a root in $G$
**Algorithm to check if a node \( x_i \) is a root in \( G \)**

**Example**

Graph \( G \) and \( x_i = x_1 \)

\[
\begin{array}{c|cc}
\text{\( x_i \)} & \text{\( \Gamma(x_i) \)} \\
\hline
(1) & x_1 & x_2, x_3 \\
(2) & x_2 & x_3, x_5 \\
(2) & x_3 & x_5, x_6 \\
(4) & x_4 & x_2, x_7 \\
(3) & x_5 & x_4, x_6, x_7 \\
(3) & x_6 & x_7, x_8 \\
(4) & x_7 & x_8 \\
(4) & x_8 & -
\end{array}
\]

\( \Rightarrow x_1 \) is a root in \( G \)

**Remark:** for anti-root we use backward adjacency list.