

# Graphes

Prof d'info

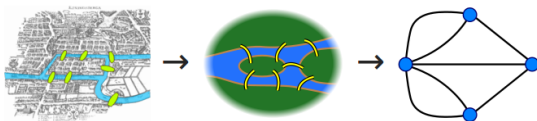
Lycée Thiers

- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles
- 5 Connexité
- 6 Retour à Königsberg

- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles
- 5 Connexité
- 6 Retour à Königsberg

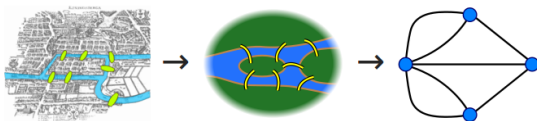
# Les sept ponts de Königsberg

- Au XVIII<sup>e</sup> siècle, on se demande s'il est possible de faire une promenade dans Königsberg qui emprunte chacun des sept ponts exactement une fois et revient au point de départ.



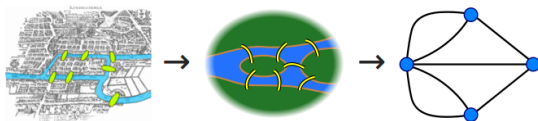
# Les sept ponts de Königsberg

- Au XVIII<sup>e</sup> siècle, on se demande s'il est possible de faire une promenade dans Königsberg qui emprunte chacun des sept ponts exactement une fois et revient au point de départ.
- Euler remplace les différentes zones de terre par des *sommets* et les ponts par des *arêtes*.



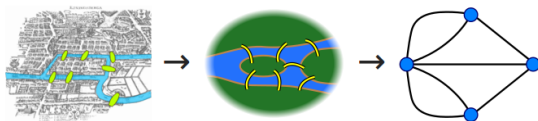
# Les sept ponts de Königsberg

- Au XVIII<sup>e</sup> siècle, on se demande s'il est possible de faire une promenade dans Königsberg qui emprunte chacun des sept ponts exactement une fois et revient au point de départ.
- Euler remplace les différentes zones de terre par des *sommets* et les ponts par des *arêtes*.
- On obtient ainsi une représentation dans laquelle seules les liaisons entre les zones importent.



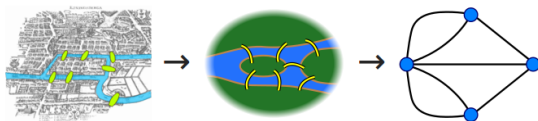
# Les sept ponts de Königsberg

- Au XVIII<sup>e</sup> siècle, on se demande s'il est possible de faire une promenade dans Königsberg qui emprunte chacun des sept ponts exactement une fois et revient au point de départ.
- Euler remplace les différentes zones de terre par des *sommets* et les ponts par des *arêtes*.
- On obtient ainsi une représentation dans laquelle seules les liaisons entre les zones importent.



# Les sept ponts de Königsberg

- Au XVIII<sup>e</sup> siècle, on se demande s'il est possible de faire une promenade dans Königsberg qui emprunte chacun des sept ponts exactement une fois et revient au point de départ.
- Euler remplace les différentes zones de terre par des *sommets* et les ponts par des *arêtes*.
- On obtient ainsi une représentation dans laquelle seules les liaisons entre les zones importent.



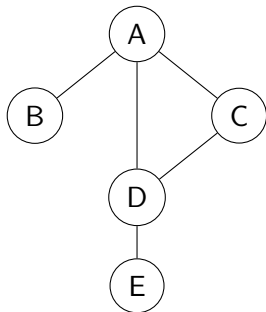
Nous reviendrons sur ce problème après avoir introduit un peu de vocabulaire.

- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles
- 5 Connexité
- 6 Retour à Königsberg

# Graphes orientés et non orientés

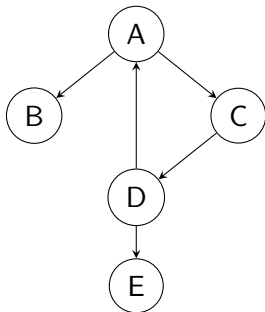
Un graphe est constitué de *sommets* reliés entre eux.

## Graphe non orienté



Les liaisons sont des *arêtes*.

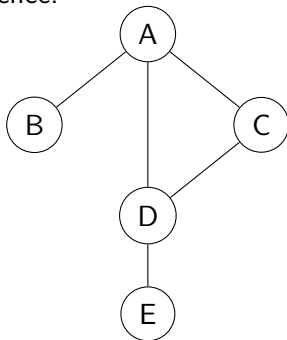
## Graphe orienté



Les liaisons sont des *arcs* : leur sens compte.

# Voisins et degré

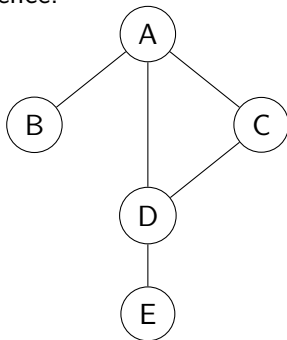
On utilise désormais le graphe non orienté suivant comme graphe de référence.



- Deux sommets reliés par une arête sont *voisins*.

# Voisins et degré

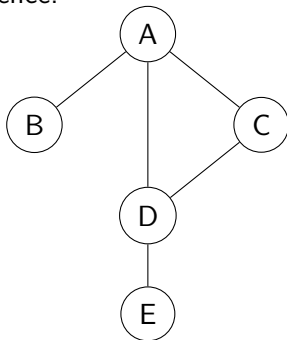
On utilise désormais le graphe non orienté suivant comme graphe de référence.



- Deux sommets reliés par une arête sont *voisins*.
- Les voisins de *A* sont *B*, *C* et *D*.

# Voisins et degré

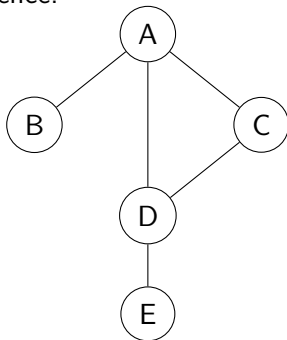
On utilise désormais le graphe non orienté suivant comme graphe de référence.



- Deux sommets reliés par une arête sont *voisins*.
- Les voisins de *A* sont *B*, *C* et *D*.
- Le *degré* d'un sommet est le nombre d'arêtes qui arrivent à ce sommet.

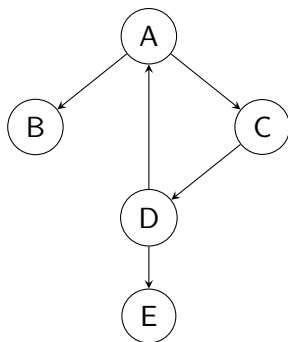
# Voisins et degré

On utilise désormais le graphe non orienté suivant comme graphe de référence.



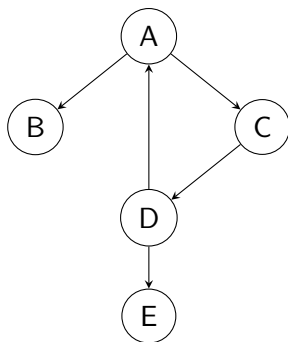
- Deux sommets reliés par une arête sont *voisins*.
- Les voisins de  $A$  sont  $B$ ,  $C$  et  $D$ .
- Le *degré* d'un sommet est le nombre d'arêtes qui arrivent à ce sommet.
- Ainsi  $d(A) = 3$ ,  $d(B) = 1$  et  $d(D) = 3$ .

# Dans un graphe orienté



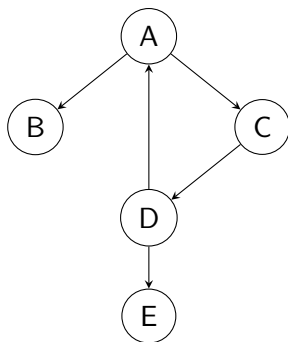
- Si un arc va de  $x$  vers  $y$ , on considère  $y$  comme un voisin de  $x$ .

# Dans un graphe orienté



- Si un arc va de  $x$  vers  $y$ , on considère  $y$  comme un voisin de  $x$ .
- On distingue le *degré sortant*, nombre d'arcs qui partent du sommet, et le *degré entrant*, nombre d'arcs qui arrivent au sommet.

# Dans un graphe orienté

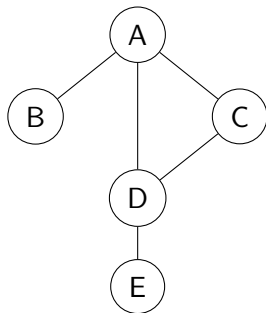


- Si un arc va de  $x$  vers  $y$ , on considère  $y$  comme un voisin de  $x$ .
- On distingue le *degré sortant*, nombre d'arcs qui partent du sommet, et le *degré entrant*, nombre d'arcs qui arrivent au sommet.
- Pour  $D$ , le degré sortant vaut 2 et le degré entrant vaut 1.

- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python**
- 4 Chemins et cycles
- 5 Connexité
- 6 Retour à Königsberg

# Matrice d'adjacences

On fixe l'ordre des sommets :  $A, B, C, D, E$ .

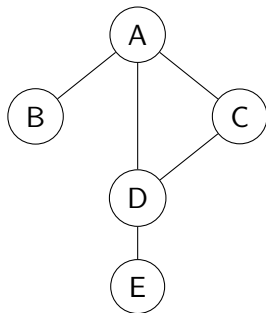


$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- $M[i][j]$  vaut 1 lorsque les sommets correspondants sont voisins, 0 sinon.

# Matrice d'adjacences

On fixe l'ordre des sommets :  $A, B, C, D, E$ .



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

- $M[i][j]$  vaut 1 lorsque les sommets correspondants sont voisins, 0 sinon.
- Pour un graphe non orienté, la matrice est symétrique.

# Listes d'adjacences

À chaque sommet, on associe la liste de ses voisins.

## Liste de listes

Les sommets sont implicitement numérotés 0, 1, 2, 3, 4 correspondant à  $A, B, C, D, E$ .

```
1 G = [[1, 2, 3],  
2      [0],  
3      [0, 3],  
4      [0, 2, 4],  
5      [3]]
```

# Listes d'adjacences

À chaque sommet, on associe la liste de ses voisins.

## Liste de listes

Les sommets sont implicitement numérotés 0, 1, 2, 3, 4 correspondant à A, B, C, D, E.

```
1 G = [[1, 2, 3],
2      [0],
3      [0, 3],
4      [0, 2, 4],
5      [3]]
```

## Dictionnaire de listes

```
1 G = {'A': ['B', 'C', 'D'], 'B': ['A'],
2      'C': ['A', 'D'], 'D': ['A', 'C', 'E'],
3      'E': ['D']}
```

# Deux représentations à connaître

- Une *matrice d'adjacences* permet de savoir directement si deux sommets sont voisins.

# Deux représentations à connaître

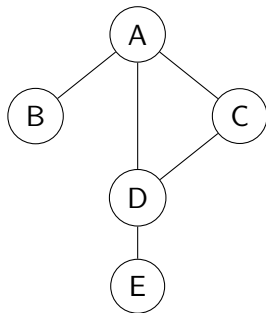
- Une *matrice d'adjacences* permet de savoir directement si deux sommets sont voisins.
- Une *liste d'adjacences* donne directement les voisins d'un sommet.

# Deux représentations à connaître

- Une *matrice d'adjacences* permet de savoir directement si deux sommets sont voisins.
- Une *liste d'adjacences* donne directement les voisins d'un sommet.
- En Python, une liste d'adjacences sera généralement représentée par une *liste de listes* ou un *dictionnaire de listes*.

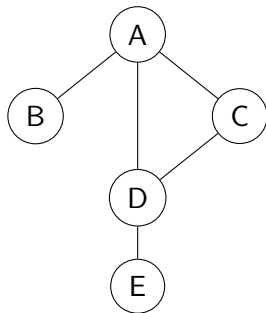
- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles**
- 5 Connexité
- 6 Retour à Königsberg

# Chemins



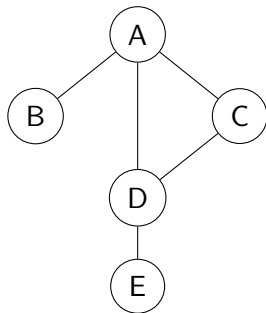
- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.

# Chemins



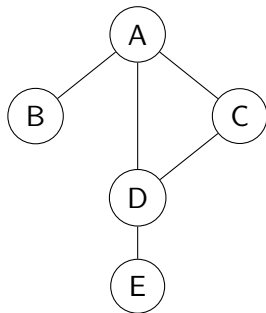
- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.

# Chemins



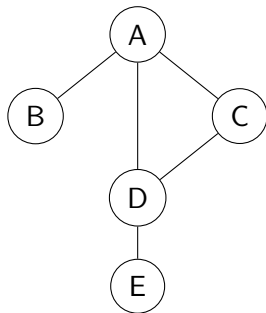
- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.
- $B, A, C, D, E$  est également un chemin.

# Chemins



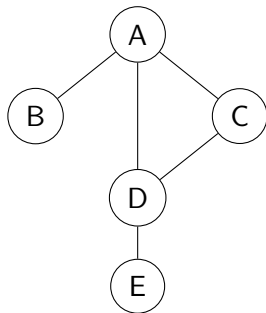
- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.
- $B, A, C, D, E$  est également un chemin.
- $B, C, D$  n'est pas un chemin.

# Chemins



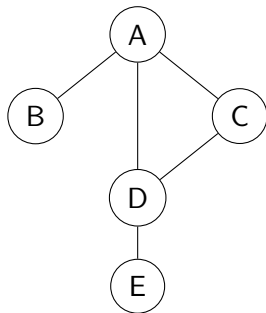
- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.
- $B, A, C, D, E$  est également un chemin.
- $B, C, D$  n'est pas un chemin.
- La *longueur* d'un chemin est le nombre d'arêtes ou d'arcs parcourus : le chemin  $B, A, D, E$  a longueur 3.

# Chemins



- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.
- $B, A, C, D, E$  est également un chemin.
- $B, C, D$  n'est pas un chemin.
- La *longueur* d'un chemin est le nombre d'arêtes ou d'arcs parcourus : le chemin  $B, A, D, E$  a longueur 3.

# Chemins

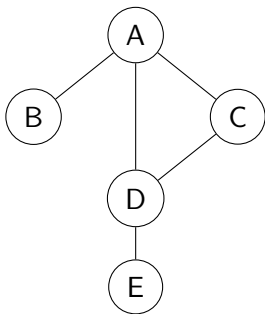


- Un *chemin* est une suite de sommets telle que chaque sommet est voisin du suivant.
- $B, A, D, E$  est un chemin.
- $B, A, C, D, E$  est également un chemin.
- $B, C, D$  n'est pas un chemin.
- La *longueur* d'un chemin est le nombre d'arêtes ou d'arcs parcourus : le chemin  $B, A, D, E$  a longueur 3.

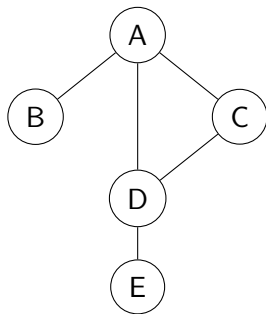
Dans un graphe orienté, il faut bien sûr respecter le sens des arcs.

# Cycles

- Un *cycle* est un chemin qui revient à son sommet de départ.

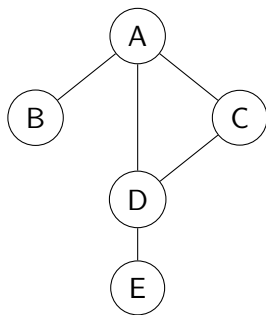


# Cycles



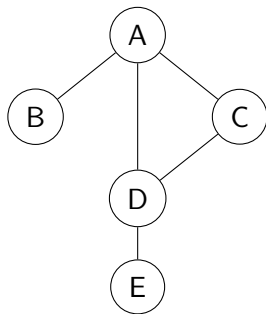
- Un *cycle* est un chemin qui revient à son sommet de départ.
- $A, C, D, A$  est un cycle.

# Cycles



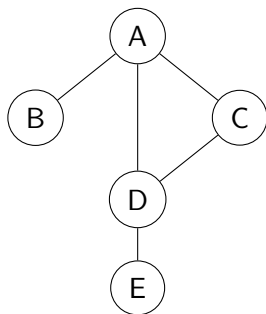
- Un *cycle* est un chemin qui revient à son sommet de départ.
- $A, C, D, A$  est un cycle.
- Un cycle est *simple* s'il ne passe pas deux fois par une même arête ou un même arc.

# Cycles



- Un *cycle* est un chemin qui revient à son sommet de départ.
- $A, C, D, A$  est un cycle.
- Un cycle est *simple* s'il ne passe pas deux fois par une même arête ou un même arc.
- Un cycle est *élémentaire* s'il ne passe pas deux fois par un même sommet, sauf le sommet de départ qui est aussi le sommet d'arrivée.

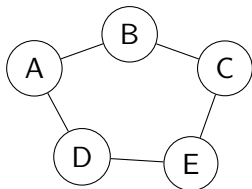
# Cycles



- Un *cycle* est un chemin qui revient à son sommet de départ.
- $A, C, D, A$  est un cycle.
- Un cycle est *simple* s'il ne passe pas deux fois par une même arête ou un même arc.
- Un cycle est *élémentaire* s'il ne passe pas deux fois par un même sommet, sauf le sommet de départ qui est aussi le sommet d'arrivée.
- Dans un graphe orienté, le sens des arcs doit être respecté.

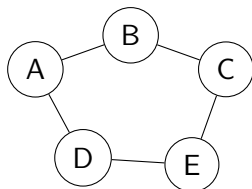
- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles
- 5 Connexité**
- 6 Retour à Königsberg

# Connexité



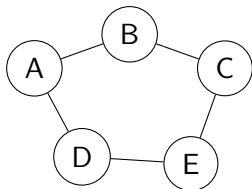
- Un graphe non orienté est *connexe* s'il existe un chemin entre n'importe quels deux sommets.

# Connexité



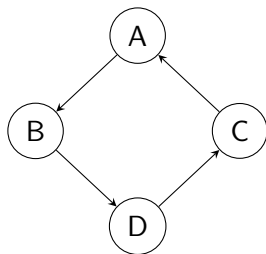
- Un graphe non orienté est *connexe* s'il existe un chemin entre n'importe quels deux sommets.
- Dans le graphe ci-contre, on peut aller de tout sommet à tout autre : il est connexe.

# Connexité



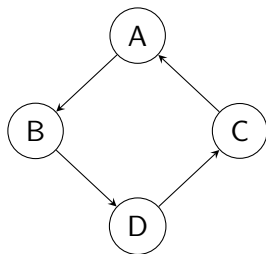
- Un graphe non orienté est *connexe* s'il existe un chemin entre n'importe quels deux sommets.
- Dans le graphe ci-contre, on peut aller de tout sommet à tout autre : il est connexe.
- Si ce n'est pas le cas, le graphe se décompose en *composantes connexes*.

# Forte connexité



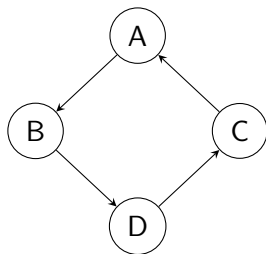
- Un graphe orienté est *fortement connexe* si, pour tous sommets  $x$  et  $y$ , il existe un chemin de  $x$  vers  $y$ .

# Forte connexité



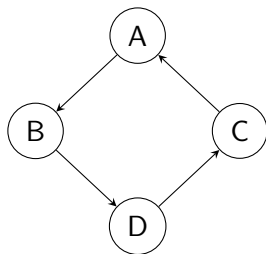
- Un graphe orienté est *fortement connexe* si, pour tous sommets  $x$  et  $y$ , il existe un chemin de  $x$  vers  $y$ .
- Il faut donc pouvoir circuler dans les deux sens entre n'importe quels deux sommets, éventuellement par des chemins différents.

# Forte connexité



- Un graphe orienté est *fortement connexe* si, pour tous sommets  $x$  et  $y$ , il existe un chemin de  $x$  vers  $y$ .
- Il faut donc pouvoir circuler dans les deux sens entre n'importe quels deux sommets, éventuellement par des chemins différents.
- Le graphe ci-contre est fortement connexe.

# Forte connexité



- Un graphe orienté est *fortement connexe* si, pour tous sommets  $x$  et  $y$ , il existe un chemin de  $x$  vers  $y$ .
- Il faut donc pouvoir circuler dans les deux sens entre n'importe quels deux sommets, éventuellement par des chemins différents.
- Le graphe ci-contre est fortement connexe.
- Sinon, on peut considérer ses *composantes fortement connexes*.

- 1 Introduction
- 2 Graphes et vocabulaire
- 3 Représentation en Python
- 4 Chemins et cycles
- 5 Connexité
- 6 Retour à Königsberg**

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.
- Le degré de chaque sommet doit donc être pair.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.
- Le degré de chaque sommet doit donc être pair.
- Dans le graphe de Königsberg, les quatre sommets ont des degrés impairs : 3, 3, 3 et 5.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.
- Le degré de chaque sommet doit donc être pair.
- Dans le graphe de Königsberg, les quatre sommets ont des degrés impairs : 3, 3, 3 et 5.
- Il n'existe donc pas de parcours qui emprunte chacun des sept ponts exactement une fois et revienne au point de départ.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.
- Le degré de chaque sommet doit donc être pair.
- Dans le graphe de Königsberg, les quatre sommets ont des degrés impairs : 3, 3, 3 et 5.
- Il n'existe donc pas de parcours qui emprunte chacun des sept ponts exactement une fois et revienne au point de départ.

# Retour aux sept ponts de Königsberg

- Pour emprunter toutes les arêtes et revenir au point de départ, chaque fois que l'on arrive à un sommet, il faut pouvoir en repartir.
- Les arêtes incidentes à chaque sommet doivent donc pouvoir être regroupées par paires.
- Le degré de chaque sommet doit donc être pair.
- Dans le graphe de Königsberg, les quatre sommets ont des degrés impairs : 3, 3, 3 et 5.
- Il n'existe donc pas de parcours qui emprunte chacun des sept ponts exactement une fois et revienne au point de départ.

## Théorème d'Euler

Un graphe connexe possède un circuit eulérien si et seulement si tous ses sommets sont de degré pair.