MARO 037 - Competitive Programming

Fabian Lecron - Arnaud Vandaele

Faculté Polytechnique - Services MIT et MARO

UMONS

Année académique 2024-2025

Séance 06 :
Graphes

Séance 06 – Graphes

Tables des matières

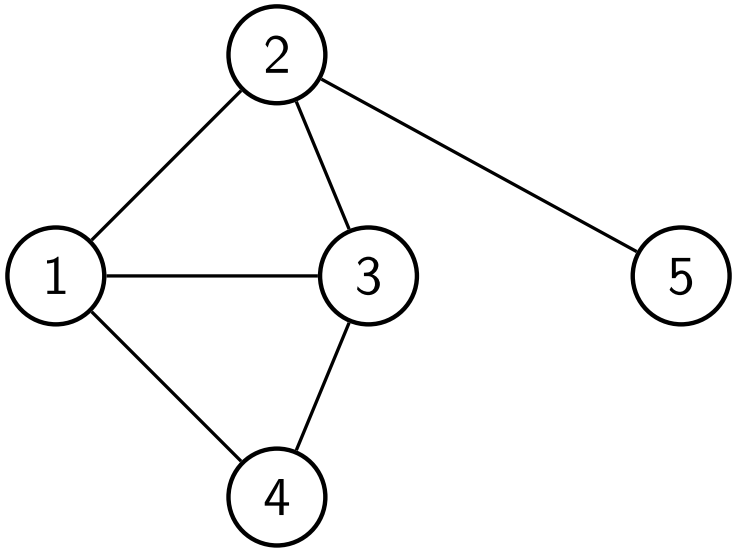
Plan

Graphes

Un graphe G est un couple ordonné de deux ensembles $G = (V, E)$, où

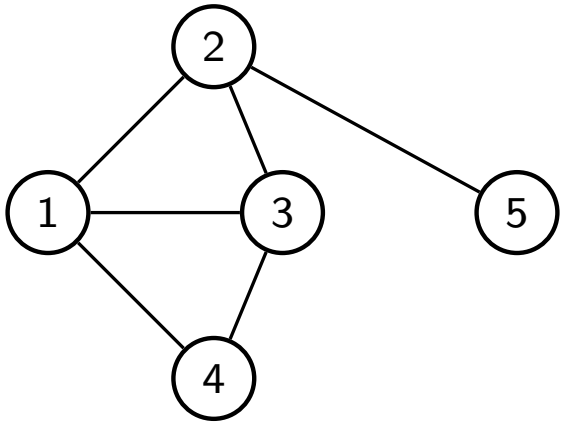
- V est l'ensemble des sommets (ou noeuds)
- E est l'ensemble des arêtes, défini comme $E \subseteq \{(x, y) : x, y \in V\}$

Exemple d'un graphe non-orienté :



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

Représentations d'un graphe



- Matrice d'adjacence (sommets/sommets), forcément symétrique pour un graphe non-orienté :

$$A = \begin{array}{c} \begin{matrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{matrix}$$

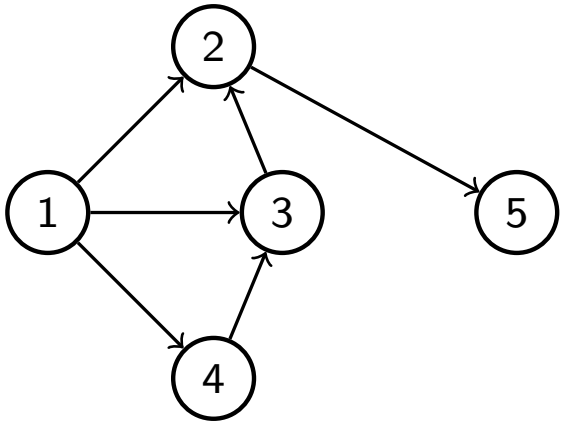
- Matrice sommets/arêtes (deux entrées à 1 par colonne) :

$$A = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{matrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{matrix}$$

- Liste des arêtes : $\{(1,2),(1,3),(3,4),(2,3),(2,5)\}$

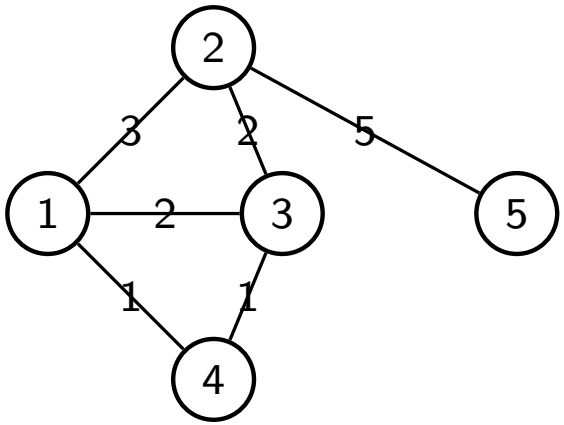
Graphes orientés - Graphes pondérés

Un graphe peut être orienté :



$$A = \begin{array}{c} \begin{matrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{matrix}$$

Un graphe peut être pondéré :

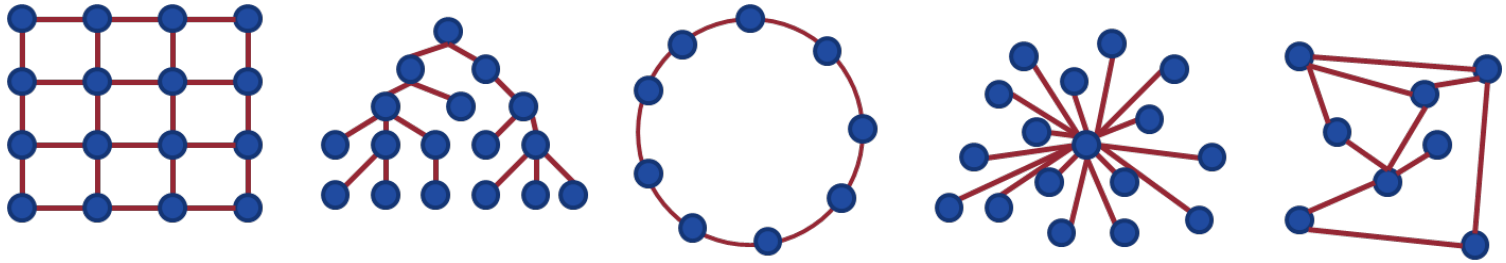


$$W = \begin{array}{c} \begin{matrix} s_1 & s_2 & s_3 & s_4 & s_5 \end{matrix} \\ \begin{pmatrix} 0 & 3 & 2 & 1 & 0 \\ 3 & 0 & 2 & 0 & 5 \\ 2 & 2 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \end{pmatrix} \end{array} \begin{matrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{matrix}$$

Un graphe peut être orienté et pondéré.

Structures et algorithmes

Les graphes peuvent posséder des structures, et donc des propriétés, particulières



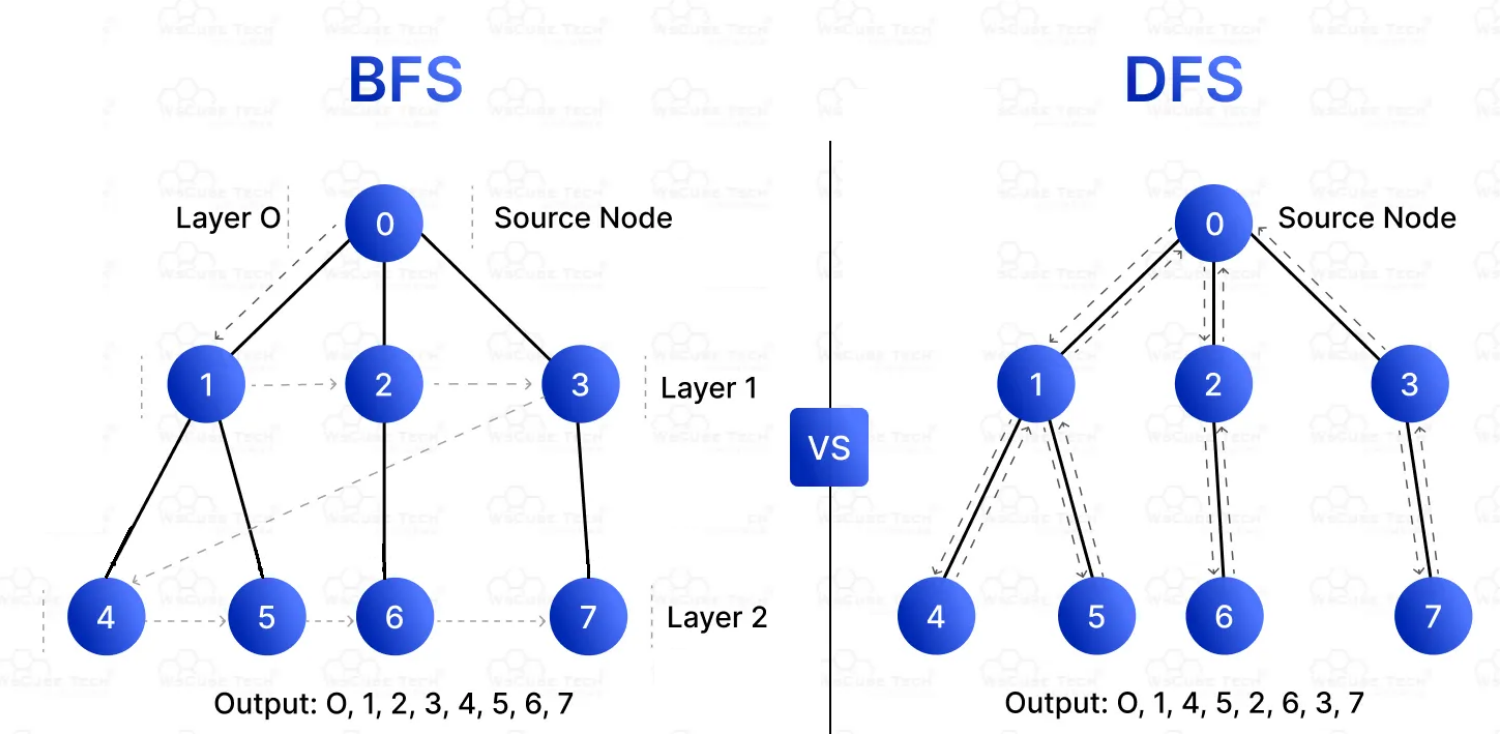
Il existe de multiples algorithmes/utilisations des graphes :

- Exploration de graphes
  - DFS - Depth First Search
  - BFS - Breadth First Search
- Algorithme de Dijkstra, pour la recherche du plus court chemin entre deux noeuds
- Algorithme de Ford-Fulkerson, pour la recherche du chemin de poids maximum, dans un graphe pondéré, entre deux noeuds
- Algorithme de Kruskal et Prim, pour la recherche de l'arbre couvrant de poids minimal
- et tellement d'autres...

Exploration de graphes

Partant d'un noeud de départ donné, explorer un graphe c'est déterminer l'ensemble des descendants de ce noeud, c'est-à-dire l'ensemble des noeuds accessibles via ce noeud de départ. Plusieurs procédés d'exploration de graphe existent. Suivant les noeuds déjà visités, ces algorithmes diffèrent dans le choix du prochain noeud à visiter.

- La recherche en profondeur (DFS) consiste à visiter d'abord les noeuds enfants avant de visiter les noeuds *du même niveau*.
- La recherche en largeur (BFS) consiste à visiter d'abord les noeuds *du même niveau* avant de visiter les noeuds enfants.

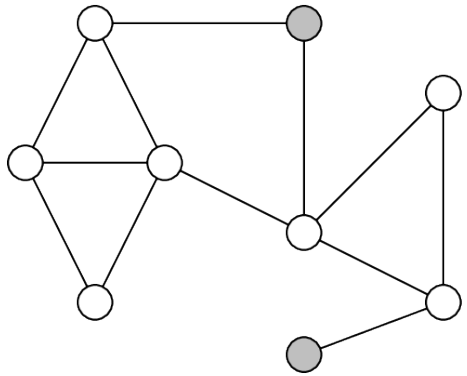


Plan

Exercice 1 : Weak vertices

Un premier exercice pour lire une matrice d'adjacence et manipuler un graphe

Enoncé :
Etant donné un graphe non orienté, un sommet i fait partie d'un *triangle* lorsque i possède au moins deux voisins différents j et k tels que j et k sont voisins l'un de l'autre.



Dans ce problème, on vous demande d'identifier les sommets *faibles* dans un graphe, c'est-à-dire les sommets qui ne font partie d'aucun triangle.

Input : L'input consiste en une série de cas à traiter, chacun constitué d'abord d'une ligne avec un entier n, suivi de n lignes de n entiers, la matrice d'adjacence du graphe. La dernière ligne de l'input est constitué de l'entier -1.

Output : Pour chacun des graphes, écrivez une ligne contenant la liste des sommets faibles, par ordre croissant.

Plan

Exercice 2 : Counting Stars

1er exercice d'exploration de graphe. Attention, le graphe n'est pas donné explicitement !

On vous donne une *image* composée de pixels qui sont soit noirs (caractère #), soit blancs (caractère −). Tous les pixels noirs sont considérés comme faisant partie du ciel, et chaque pixel blanc est considéré comme faisant partie d'une étoile. Sur l'image, les pixels blancs adjacents verticalement ou horizontalement font partie de la même étoile.

Sample Input 1

```
10 20
##################---
##-################--
#---################
##-################
########---#########
#######-----########
########---#########
##################--
##################---
##################-#
3 10
#-########
----------
#-########
```

Sample Output 1

```
Case 1: 4
Case 2: 1
```

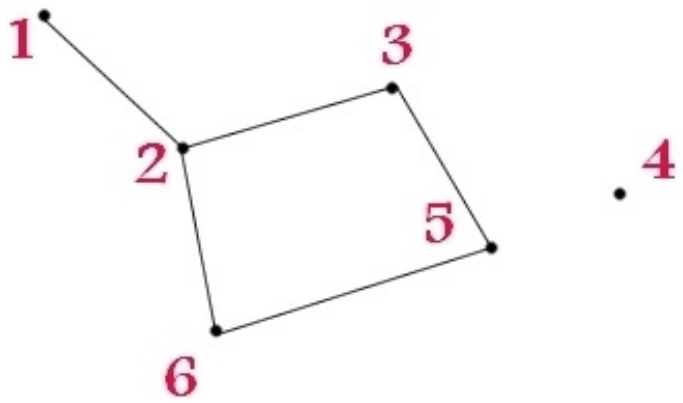
Input : L'input consiste en une série de cas à traiter, chacun constitué d'abord d'une ligne avec deux entiers m et n, suivis de m lignes composées de n caractères.
Output : Pour chaque cas, affichez le numéro du cas suivi du nombre d'étoiles visibles dans l'image correspondante (respectez le format attendu).

Plan

Exercice 3 : Where's My Internet??

Autre exercice d'exploration où il faut détecter si un graphe connexe

Enoncé :
Les maisons d'une ville sont numérotées de 1 à N. La maison numéro 1 a déjà été connectée à internet via un câble vers une ville voisine. Il est prévu de fournir internet à d'autres maisons en connectant des paires de maisons à l'aide de câbles réseau distincts. Une maison est connectée à internet si elle est reliée par un câble à une autre maison déjà connectée à internet.
Étant donné la liste des paires de maisons déjà connectées par un câble réseau, déterminez les maisons qui ne sont pas encore connectées à internet.



Input : La première ligne contient deux entiers N (nombre de maisons) et M (nombre de cables), suivis de M lignes composées chacune de deux entier a et b, signifiant que la ville a est reliée à la ville b.
Output : Si toutes les maisons sont déjà connectées, écrivez la ligne `Connected`. Dans le cas contraire, écrivez les numéros de maison par ordre croissant, un par ligne, représentant les maisons qui ne sont pas encore connectées.

Plan

Exercice 4 : Prime path

Application de BFS

Etant donnés deux nombres premiers de quatre chiffres, on vous demande de déterminer s'il existe un chemin de nombres premiers entre les deux, où deux nombres premiers consécutifs ne diffèrent que par un seul chiffre.

Exemple avec 1033 et 8179, où le chemin est de longueur 6 :

$$1033 \to 1733 \to 3733 \to 3739 \to 3779 \to 8779 \to 8179.$$

Input : La première ligne contient le nombre de cas à tester. Ensuite, chacun des lignes suivantes contient deux nombres premiers.

Output : Une ligne pour chaque cas, soit avec un chiffre indiquant la longueur du plus petit chemin, soit le mot `Impossible`.

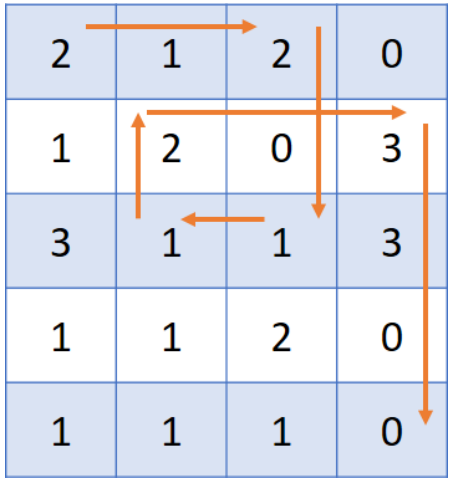
Plan

Exercice 5 : Grid

Un BFS plus difficile

Vous possédez une grille de taille $n \times m$ où chaque case de la grille contient un chiffre. A partir d'une case donnée sur laquelle figure un chiffre k, un déplacement consiste à sauter exactement k cases dans l'une des quatre directions cardinales (un déplacement ne peut évidemment pas dépasser les bords de la grille).

Quel est le nombre minimum de déplacements nécessaires pour aller du coin supérieur gauche au coin inférieur droit ?



Input : La première ligne contient deux entiers n et m, les dimensions de la grille Ensuite, chacune des n lignes contient m chiffres entre 0 et 9.
Output : Ecrivez un seul entier sur une ligne représentant le nombre minimum de déplacements nécessaires pour aller du coin supérieur gauche de la grille au coin inférieur droit. Si ce n'est pas possible, écrivez -1.

Plan

Exercice 6 : Island Hopping

L'algorithme de Prim/Kruskal.

Etant donné un ensemble de coordonnées représentant la position d'îles, il est demandé de déterminer la longueur de ponts à construire entre les îles afin qu'il existe un chemin entre toute paire d'îles.

Input : La première ligne contient un entier, n, le nombre de cas à tester. Ensuite, pour chaque cas, on trouve d'abord une ligne avec un seul entier, m, reprenant le nombre d'îles. Après, m lignes suivent, chacune composée des coordonnées (deux floats, avec au plus 3 décimales).

Output : Une ligne pour chaque cas, avec la longueur minimal de ponts qu'il est nécessaire de construire.