	**Langages et Programmation**	**NSI T^{ale}**
	Applications des parcours de Graphes	Cours/TD

1. Existence de chemin et connexité

1. Nous avons vu que les deux parcours d'un graphe à partir du sommet x permettent de visiter tous les sommets y accessibles à partir de x. Il est donc facile d'adapter ces parcours pour déterminer la **liste de tous les sommets accessibles** ou **pour tester l'existence d'un chemin** entre deux sommets x et y.
2. Dans le cas d'un **graphe non orienté**, le graphe est **connexe** si et seulement si le parcours à partir du sommet 1 visite les n sommets du graphe.

2. Plus courts chemins

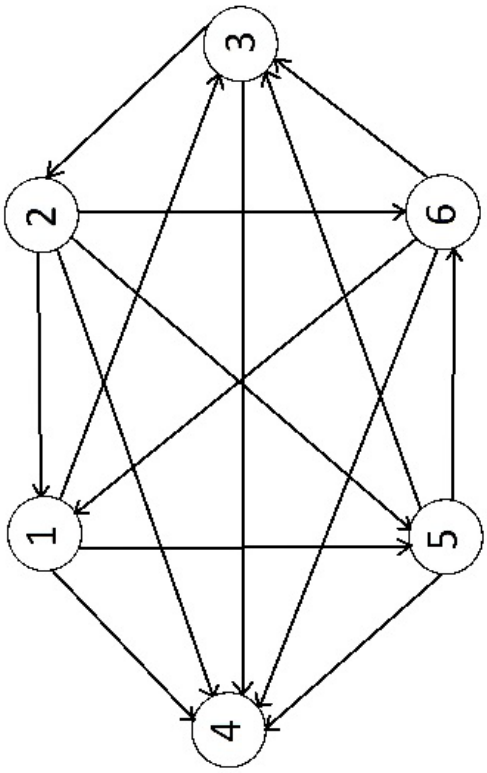
Il s'agit ici de déterminer le **plus court chemin en nombre de sauts** entre deux sommets x et y.
L'avantage d'un parcours en largeur est que, du fait de l'ordre de visite des sommets (les sommets voisins de x, les sommets à une distance 2, etc.) la première fois que l'on rencontrera y ce sera par un chemin de longueur minimale.

Nous allons, en un seul parcours, calculer le plus court chemin de x vers tous les sommets y accessibles à partir de x. Pour cela, nous utilisons `Dist` tel que `Dist[y]` est la longueur du plus court chemin de x vers y (1 si y n'est pas accessible). Afin de retrouver le chemin, nous utilisons un tableau `Pere` tel que `Pere[y]` est le prédécesseur de y dans le plus court chemin de x vers y.

```
trouvé = Faux
initialiser un tableau Visite à Faux
initialiser un tableau Dist à l'infini
F = [x]
Visite[x] = Vrai
Dist[x] = 0
Pere[x] = x
Tant que F != [] :
   y = enleverTete(F)
   Pour chaque successeur z de y
      Si Visite[z] = Faux
         Visite[z] = Vrai
         ajouterFin(z, F)
         Dist[z] = Dist[y]+1
         Pere[z] = y
```

Exemple :

Recherche des plus courts chemins à partir du sommet 2 dans le graphe.



Evolution de la File	Tableau Distance						Tableau Père					
	Sommet 1	Sommet 2	Sommet 3	Sommet 4	Sommet 5	Sommet 6	Sommet 1	Sommet 2	Sommet 3	Sommet 4	Sommet 5	Sommet 6

3. Existence de cycles

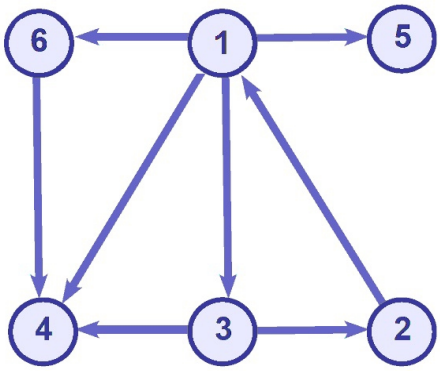
Définition :

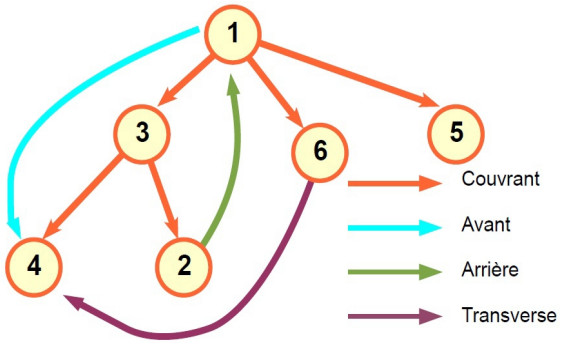
À l'issue d'un parcours, un arc (x, y) (ou une arête $\{x, y\}$) de G est dit **arc** (arête)

- **couvrant** si c'est un arc de l'arbre de parcours
- **avant** s'il existe un chemin de x à y dans l'arbre de parcours
- **arrière** s'il existe un chemin de y à x dans l'arbre de parcours.
- Tous les autres arcs, sont dits **transverses.**

Exemple :

Par exemple, le parcours en profondeur à partir du sommet 1 dans le graphe $G2$ ci-contre, nous donne la classification suivante :





REMARQUE : Selon le parcours (largeur ou profondeur) et les propriétés du graphe (orienté ou non), tous les types d'arcs ne sont pas nécessairement présents.

Proposition :

L'existence de **cycles** est caractérisée par la **présence d'arcs arrières**.

Par ailleurs,

1. **Dans un graphe non orienté**, on détecte un arc arrière (et donc un cycle) en cas de **revisite** d'un sommet déjà visité (et différent de son père).
2. **Dans un graphe orienté**, les arcs en avant et les arcs transverses provoque également des revisites. Nous détectons un arc arrière (et dont un cycle) en cas de **revisite d'un sommet dont le parcours en profondeur n'est pas encore terminé** (on a fait la première visite mais pas la dernière). Nous pouvons pour cela, utiliser le vecteur `Visite` et considérer que `Visite[x]` vaut `0` s'il n'a pas été visité, `1` si on a effectué la première visite et `2` quand on a effectué la dernière visite.

Exemple :

Observons l'évolution du vecteur `Visite` sur le graphe `G2`.

4. Test d'arbre

Remarquons qu'un **arbre** peut être défini comme **un graphe non orienté, connexe et sans cycle**.
Les deux méthodes vues précédemment pour tester l'existence d'un cycle et la connexité d'un graphe non orienté permettent donc de vérifier si un graphe est bien un arbre.

5. TRAVAUX DIRIGÉS : Applications des parcours et Modélisation

→ Modélisation - Dominos

Un jeu de domino dont les pièces sont numérotées de 1 à n est constitué

- des doubles $(1, 1), (2, 2), \ldots, (n, n)$
- des paires (i, j) avec $1 \leq i \leq j \leq n$.

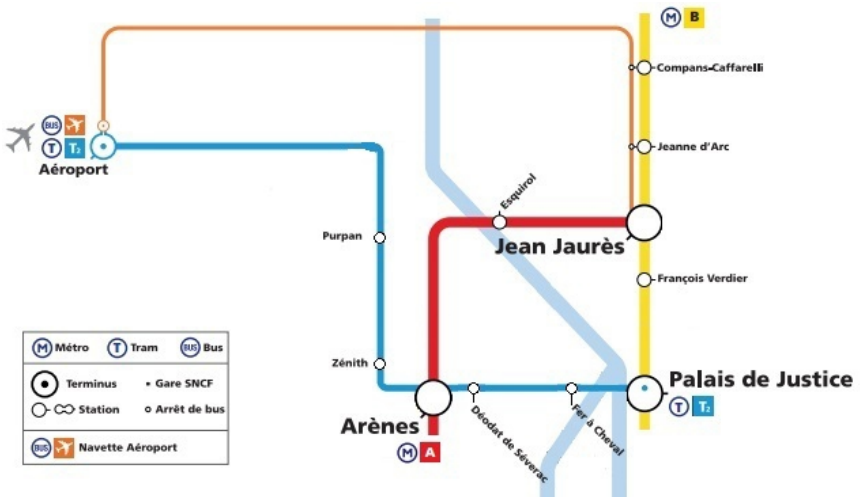
Le jeu consiste à aligner des dominos de telle sorte que les parties voisines de deux dominos consécutifs aient le même numéro.

Pour simplifier, nous pouvons supposer que $n = 3$.

1. Nous souhaitons modéliser notre jeu de domino par un graphe, comment pourrait-on procéder ?
2. Est-il possible de faire un cycle avec tous les dominos ?

→ Modélisation - Se déplacer en métro

Voici le plan simplifié des principales lignes du réseau de transports en commun toulousain. On y voit, les lignes de métro A et B, la ligne de tramway $T1$ et la navette aéroport.

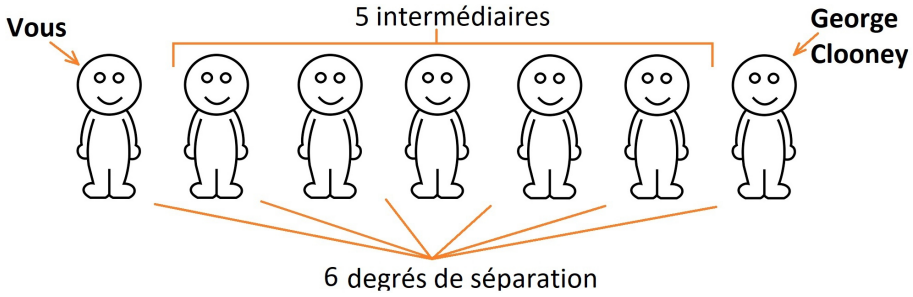


Nicolas souhaite se rendre de l'aéroport à la station **Palais de Justice** en empruntant le trajet comportant le moins d'arrêts.

1. Modéliser le problème de Nicolas à l'aide d'un graphe en précisant les propriétés de ce graphe. Expliquer pourquoi l'algorithme de plus court chemin dans un graphe permet d'y apporter une solution.
2. En donnant les différentes étapes de l'algorithme, apporter une solution à Nicolas.

→ Modélisation - Degré de séparation

La théorie des 6 degrés de séparation a été établie par le Hongrois F. Karinthy en 1929. Cette théorie affirme que toute personne sur Terre peut être reliée à n'importe quelle autre, au travers d'une chaîne de relations individuelles comprenant 5 intermédiaires.



Avec le développement des technologies de l'information et de la communication, le degré de séparation entre deux individus quelconques a été mesuré à une moyenne de 4,74 sur Facebook en 2011, et de 3,5 en 2016.

Nous souhaitons vérifier cette théorie dans le monde du cinéma, et nous considérons que deux acteurs sont reliés s'ils ont tourné un film ensemble. En particulier nous souhaitons déterminer les degrés de séparation entre George Clooney (GC) et les acteurs suivants :

Omar Sy (OS), André Dussolier (AD), Jean Dujardin (JD), Vincent Cassel (VC), Audrey Tautou (AT), François Cluzet (FC), Meryl Streep (MS), Tom Hanks (TH), Anna Kendrick (AK).

Voici les films qu'ils ont tourné ensemble :

	George Clooney	Omar Sy	André Dussolier	Jean Dujardin	Vincent Cassel	Audrey Tautou	François Cluzet	Meryl Streep	Tom Hanks	Anna Kendrick
GC				Monuments Men	Ocean's 12					In the Air
OS							Intouchables		Inferno	
AD					La belle et la bête	Le fabuleux destin d'Amélie Poulin				
JD	Monuments Men						Les petits mouchoirs			
VC	Ocean's 12		La belle et la bête				Un moment d'égarement			
AT			Le fabuleux destin d'Amélie Poulin						Da Vinci Code	
FC		Intouchables		Les petits mouchoirs	Un moment d'égarement					
MS									Pentagon Papers	Into the Woods
TH		Inferno				Da Vinci Code		Pentagon Papers		
AK	In the Air							Into the Woods		

1. Montrer que les relations entre ces acteurs peuvent être modélisées par un graphe (en précisant les propriétés de celui-ci).
2. Expliquer alors pourquoi l'algorithme de plus court chemin dans un graphe permet le calcul du degré de séparation entre deux acteurs.
3. Appliquer l'algorithme.
4. Quel est le plus haut degré de séparation entre George Clooney et les autres acteurs ? Quelle est la distance moyenne de George Clooney aux autres acteurs ?

6. *TRAVAUX PRATIQUES : Applications des parcours*

Avertissement : Là encore, nous choisissons de nous limiter au cas des graphes non orientés. Les plus rapides d'entre vous pourront essayer d'adapter leur fonction aux graphes orientés - la difficulté principale réside dans la recherche d'existence d'un cycle (et donc d'un arc arrière) qui s'effectue de préférence dans le cadre d'un parcours en profondeur.

→ Graphes non orientés

Tester la connexité d'un graphe à l'aide d'un parcours en largeur :

Rappelons qu'un graphe non orienté est connexe si et seulement si un parcours à partir du sommet `1` visite tous les sommets.

1. Copier et modifier votre parcours en largeur en une fonction booléenne `estConnexe(G)` déterminant si un graphe `G` est connexe.

Tester l'existence d'un cycle par un parcours en largeur :

Un graphe non orienté `G` ne contient pas de cycle si et seulement si le parcours ne provoque pas de revisite (autre que celle du père). On ajoute donc au parcours en largeur un tableau `Pere`, que l'on met à jour lorsqu'on ajoute un tableau dans la `File`.
Si, au cours de l'étude des successeurs d'un sommet `y`, on trouve un sommet `z`, déjà visité, tel que `Pere[y]!=z`, alors c'est que l'on vient de trouver un cycle.

2. Écrire une fonction `estCyclique(G)`, adaptée de votre parcours en largeur, qui renvoie `True` si `G` contient un cycle, `False` sinon.

Test d'arbre :

Rappelons qu'un graphe non orienté est un arbre si et seulement si il est connexe et sans cycle.

3. Déduire des fonctions précédentes, une fonction booléenne `estArbre(G)` qui renvoie `True` si et seulement si `G` est un arbre, `False` sinon.

Recherche des plus courts chemins à l'aide d'un parcours en largeur :

Comme vu précédemment, on utilise un vecteur `Dist` tel que `Dist[y]` est la longueur du plus court chemin de `x` vers `y` (ou la valeur `-1` si `y` n'est pas accessible à partir de `x`).
Afin de retrouver le chemin, on utilise également un tableau des pères tel que `Pere[y]` est le prédécesseur de `y` dans le plus court chemin de `x` vers `y`.

4. Écrire une fonction `plusCourtChemin(G,i)` qui retourne les vecteurs `Dist` et `Pere` donnant les plus courts chemins issus du sommet `i`.