	**Langages et Programmation**	**NSI T^{ale}**
	Parcours de Graphes	Cours/TD

1. Introduction

Nous le verrons dans la suite, beaucoup de problèmes sur les graphes nécessitent un examen exhaustif des sommets ou des arêtes (arcs) du graphes. Nous distinguerons deux types de parcours : en profondeur et en largeur.

2. Parcours en profondeur

Ce parcours est aussi appelé **DFS** pour `Deep First Search` en anglais.

Le **parcours en profondeur** consiste, à partir d'un sommet donné, à **suivre un chemin le plus loin possible**, **sans passer deux fois par le même sommet**. Quand on rencontre un sommet déjà visité, on fait marche arrière pour revenir au sommet précédent et explorer les chemins ignorés précédemment.

→ Algorithme

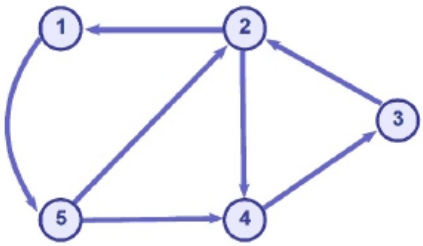
Cet algorithme se conçoit naturellement de manière récursive. Pour savoir si un sommet a déjà été visité, nous utilisons un tableau de booléens *`Visite`* tel que *`Visite[x] = Vrai`* si et seulement si le sommet y a déjà été visité.

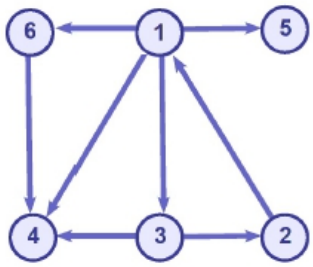
Nous obtenons alors l'algorithme suivant :

```
profond(G,x) : {parcours en profondeur du graphe G à partir du sommet x}
   initialiser un tableau Visite à Faux
   Visite[x] = Vrai
   {première visite de x}
   Pour chaque voisin y de x faire
      Si Visite[y] = faux alors
         profond(G,y)
      {Sinon on détecte une revisite de y}
   {traiter x en dernière visite}
```

Exemple :

Exemple sur les graphes G2 et G3 suivants.



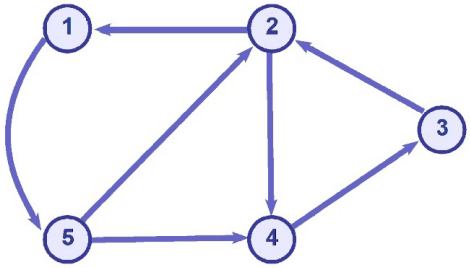


→ Ordre de parcours en première et dernière visite

Dans les applications, nous traiterons les sommets x rencontrés, soit en première visite (quand on commence le parcours en profondeur de x), soit en dernière visite (quand ce parcours est terminé et qu'on remonte au sommet qui a appelé récursivement `profond(G,x)`).

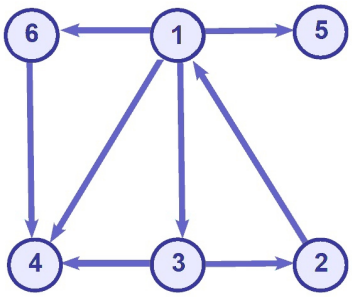
L'ordre dans lesquels les sommets sont traités est alors respectivement appelé **ordre de parcours en première ou en dernière visite**.

Exemple :



Pour le parcours de ce graphe `G2`, à partir du sommet `1`,

- l'ordre de première visite est `[1,5,2,4,3]`
- l'ordre de dernière visite est `[3,4,2,5,1]`



Pour le parcours de ce graphe `G3`, à partir du sommet `1`,

- l'ordre de première visite est `[1,3,2,4,5,6]`
- l'ordre de dernière visite est `[2,4,3,5,6,1]`

Observons sur cet exemple que l'ordre de parcours en dernière visite n'est pas toujours l'ordre inverse de l'ordre de parcours en première visite.

→ Arborescence de parcours en profondeur

Définition :

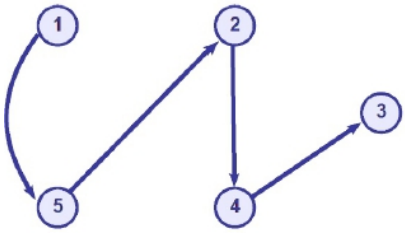
L'**arborescence de parcours en profondeur** de G à partir du sommet x est un **arbre enraciné en x, orienté** et tel qu'il existe un arc (x, y) dans A si et seulement si l'appel à `profond(G,x)` a engendré récursivement un appel à `profond(G,y)`.

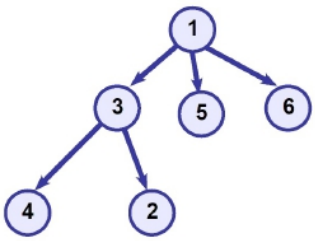
Conséquence :

Les sommets de l'arborescence de parcours sont tous les sommets de G accessibles à partir de x.

Exemple :

Voici les arbres de parcours des graphes $G2$ et $G3$, à partir du sommet 1.





3. Parcours en largeur

Ce parcours est aussi appelé **BFS** pour `Breadth First Search` en anglais.

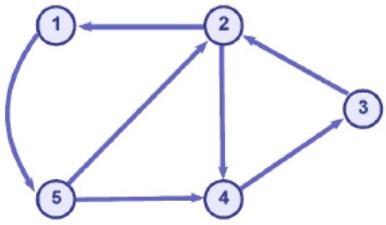
Le **parcours en largeur** est un parcours plus "prudent". Au lieu de s'aventurer le plus loin possible comme dans le parcours en profondeur, nous allons d'abord commencer par visiter tous les voisins de x, puis les voisins des voisins, etc.

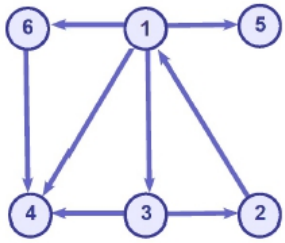
→ Algorithme

Pour cela, au fur et à mesure que l'on rencontre de nouveaux sommets (non encore visités), on mémorise leurs voisins dans une file d'attente F pour une visite prochaine.
`Visite` est toujours un vecteur de booléens (une liste) tel que `Visite[x] = Vrai` si et seulement si le sommet x a déjà été visité. F est la file des sommets qu'on devra prochainement visiter.

```
largeur(G,x) : {parcours en largeur du graphe G à partir du sommet x}
   initialiser un tableau Visite à Faux
   F = [x]
   Visite[x] = vrai
   Tant que F n'est pas vide
      considérer y la tête de F (et l'enlever de F)
      pour chaque successeur z de y
         Si Visite[z] = faux
            Visite[z] = vrai
            ajouter z à la fin de la file F
```

Exemple :
Exemple sur les graphes $G2$ et $G3$ suivants.





→ Ordre de parcours

L'ordre dans lesquels les sommets sont traités est alors appelé **ordre de parcours en largeur**.

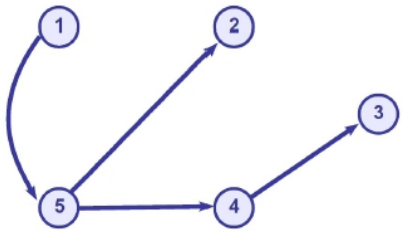
Exemple :
Lors du parcours en largeur du graphe $G2$ à partir du sommet 1, les sommets sont visités dans l'ordre : $[1,5,2,4,3]$.
Pour le graphe $G3$, toujours à partir du sommet 1, l'ordre est : $[1,3,4,5,6,2]$.

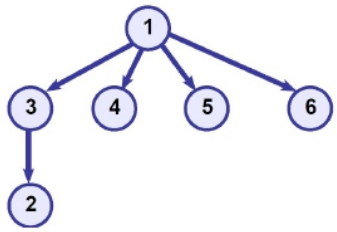
→ Arborescence de parcours en largeur

Définition :
L'**arborescence de parcours en largeur** de G à partir du sommet x est un **arbre enraciné en x, orienté** et tel qu'il existe un arc (y, z) si et seulement si le traitement de y ajoute le sommet z dans la liste d'attente F.

Conséquence :
Les sommets de l'arborescence de parcours sont tous les sommets de G accessibles à partir de x.

Exemple :
Voici les arbres de parcours des graphes `G2` et `G3`, à partir du sommet `1`.





4. Complexité des parcours

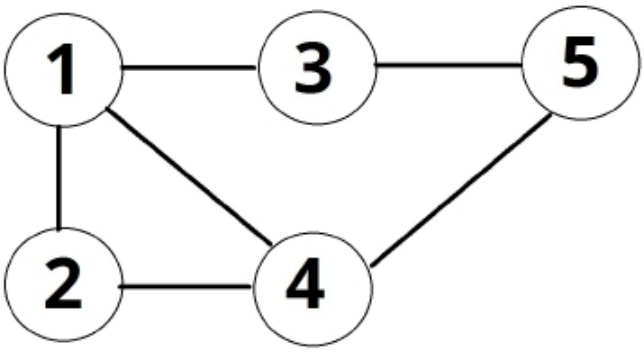
Le tableau `Visite` est initialisé en `O(n)`.
Le parcours, quel qu'il soit, est appelé exactement une fois pour chaque sommet `x` de `G` et a pour complexité `O(1)` pour le traitement de `x` et `O(d(x))` pour l'exploration des successeurs de `x`.
Dans le pire des cas, le parcours à partir d'un sommet entraîne une visite complète du graphe `G` et nous avons donc :

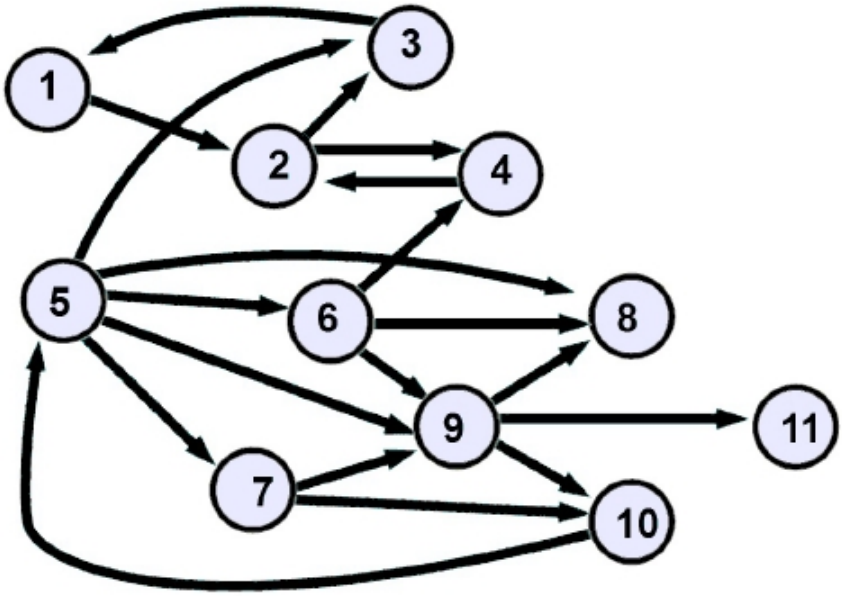
$$\begin{aligned} C(n) &= O(n) + \sum_{x=1}^{n} (d(x) + O(1)) \\ &= O(n) + \sum_{x=1}^{n} d(x) \\ &= O(n) + O(m) \\ &= O(n+m) \end{aligned}$$

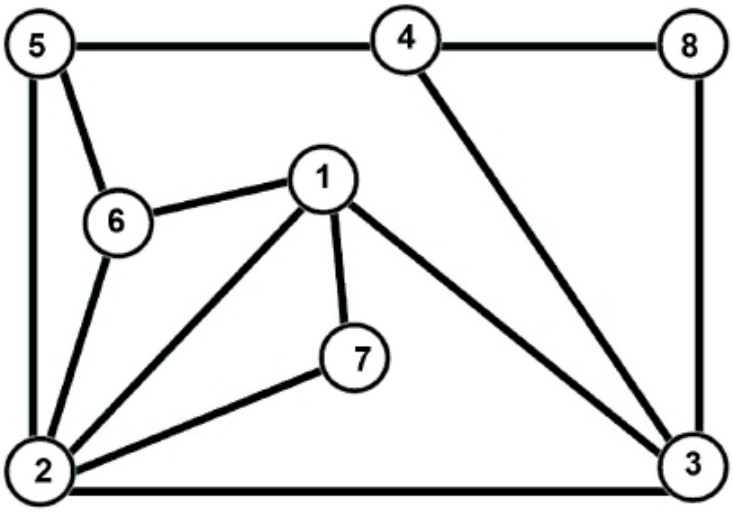
5. TRAVAUX DIRIGÉS

→ Parcours d'un graphe

Exercez-vous aux parcours de graphes sur les graphes suivants. Variez les plaisirs ! Ne commencez pas toujours du sommet *1*.







6. TRAVAUX PRATIQUES : Parcours d'un graphe

→ Introduction

Nous poursuivons le TP précédent sur les graphes. Dans toute la suite, les graphes sont représentés par leur liste d'adjacence.

→ Parcours en largeur

Écrire une fonction `largeur(G,i)` : effectue le parcours en largeur du graphe `G` à partir du sommet `i`.

Indications :

- On utilise un vecteur `Visite` tel que `Visite[i]=False` si le sommet `i` n'a pas encore été visité et `Visite[i]=True` sinon.
- On utilise une liste `File` pour gérer la file d'attente des sommets à visiter prochainement.
- Le résultat de l'appel de la fonction sera la liste des sommets visités dans l'ordre de ce parcours représenté par une liste `ordreVisite`.

→ Parcours en profondeur

Comme souvent lorsqu'on écrit un algorithme récursif, on distinguera une fonction auxiliaire (récursive) et une fonction d'appel dont le rôle est d'initialiser les variables et d'enclencher le premier appel récursif.

Écrire les fonctions suivantes :

1. `profRec(G,i,Visite,ordreVisite)` : fonction auxiliaire récursive qui provoque un parcours en profondeur du graphe à partir du sommet i. Cette fonction ne retourne aucun résultat et se contente d'initialiser les paramètres `Visite` et `ordreVisite.`
2. `profond(G,i)` : effectue le parcours en profondeur du graphe `G` à partir du sommet `i`

Indications :

- Le résultat de l'appel de la fonction sera la liste des sommets visités dans l'ordre de ce parcours représenté par une liste `ordreVisite.`
- En fonction de l'endroit où on positionne la mise à jour de la liste `ordreVisite` la fonction donnera l'ordre de première visite ou l'ordre de dernière visite. On pourra écrira donc deux versions des fonctions pour permettre de réaliser les 2 ordres de visite.