Graphes

Thibault PENNING

Introduction

C'est une structure de données permettant de relier les données entre elles.

Cela permet par exemple de relier différentes données et y voir les liens existants. Cela permet par exemple aussi de modéliser des situations de la vie réelle. Par exemple, en géographie, un graphe peut être utilisé afin de remarquer quels pays sont frontaliers d'un autre. On relierait ainsi chaque sommet entre eux lorsque les pays sont frontaliers. Ou par exemple, on pourrait représenter un réseau routier à travers une liste d'intersection, où chaque arc indiquerait qu'une intersection possède une route menant à une autre. Dans les réseaux sociaux, on pourrait représenter, par exemple, les connexions amicales entre les personnes à travers un graphe ou chaque arc indique l'amitié existant entre 2 personnes. Enfin, on peut aussi utiliser les graphes comme étant des machines à état, par exemple dans les jeux vidéo. Ce graphe permet ainsi de décrire les différents états et leur transition possible (attaque, neutre, défense, course, …).

Définition 0.1 (Graphe orienté)**.** Un graphe est un ensemble de *sommets* et d'*arcs*. Chaque arc se compose d'un couple de sommets permettant d'indiquer que le premier sommet est relié au $2^{\text{ème}}$.

> **Note**
>
> Un arbre est un cas particulier d'un graphe.
> On appelle un sommet un nœud.

> **Mise en garde**
>
> Si un sommet a possède un arc vers le sommet b, l'inverse n'est pas forcément vrai. Ainsi, dans ce cas, seul le sommet a est relié b et le sommet b ne voient pas qu'il est relié à a.

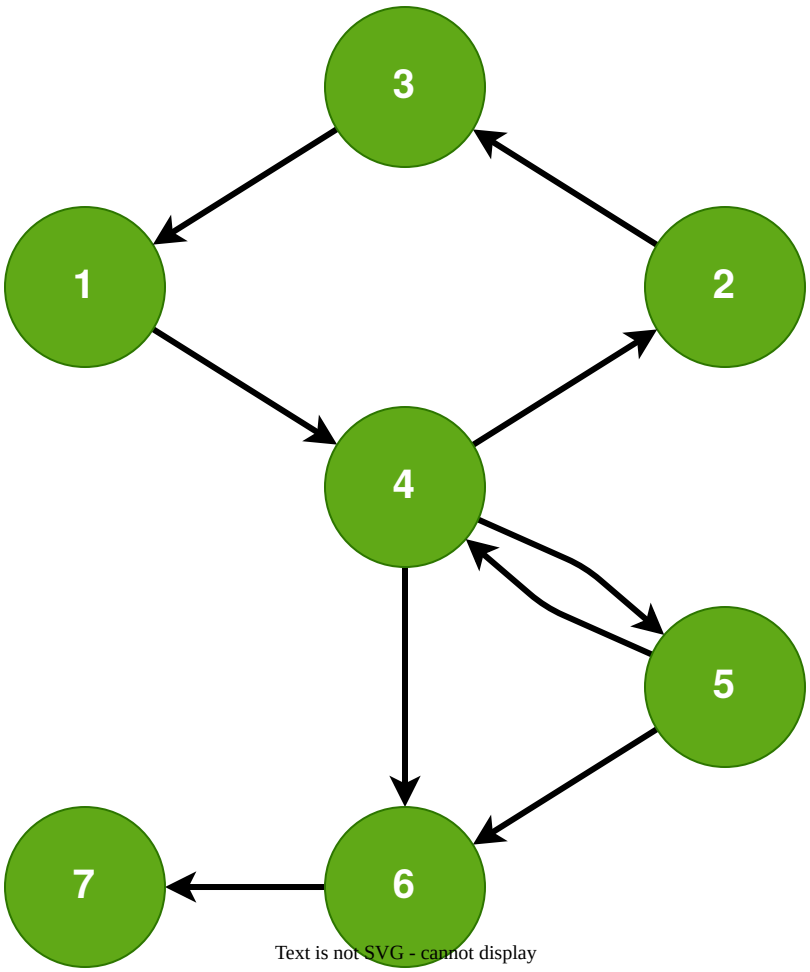

Figure 1: Un exemple de Graphe

Définition 0.2 (Adjacence)**.** Un sommet b est dit *adjacent* à a s'il existe un arc tel que le sommet a pointe vers le sommet b.

On dit aussi que a est **voisin** de b. L'ensemble des voisins d'un nœud est appelé **voisinage** du nœud.

Définition 0.3 (Successeur)**.** Un sommet b est dit *successeur* de a s'il existe un arc tel que le sommet a pointe vers le sommet b.

Définition 0.4 (Prédécesseur)**.** Un sommet a est dit *prédécesseur* de b s'il existe un arc tel que le sommet a pointe vers le sommet b.

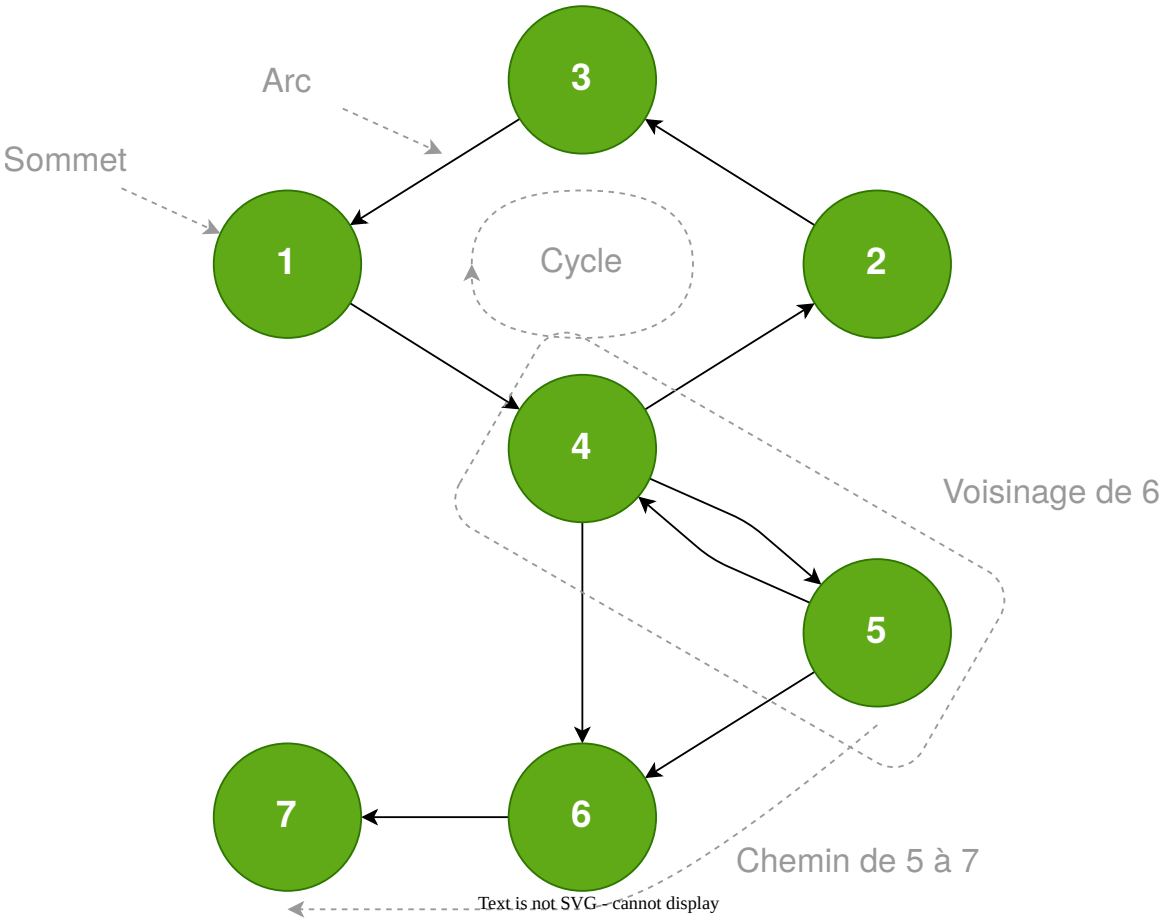

Figure 2: Vocabulaire d'un graphe

Définition 0.5 (Graphe non orienté)**.** Un graphe est dit ***non orienté*** si pour chaque arc (a, b), il existe aussi l'arc (b, a).

On parle en général d'**arrêtes** entre a et b (et on représente rarement les deux arcs, mais une seule arrête), et de nœud pour ses sommets.

> **Note**
>
> Dans un graphe non orienté, le sens des arcs n'a donc pas d'importance.

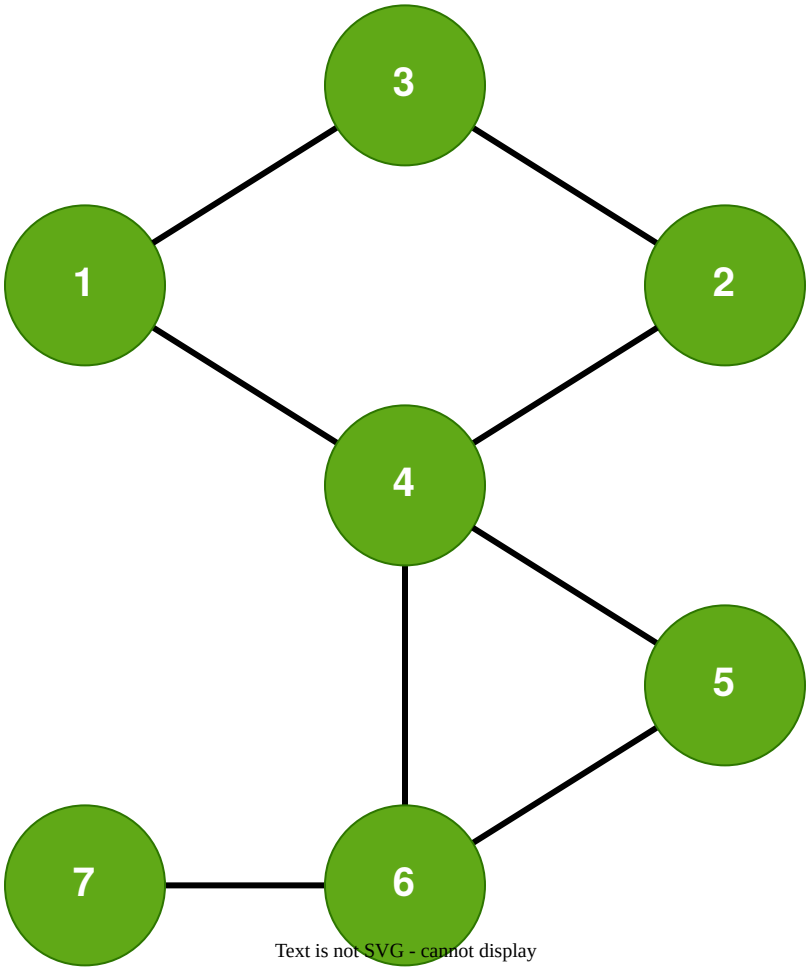

Figure 3: Exemple de graphe non orienté

Définition 0.6 (Chemin)**.** Un chemin de u à v dans un graphe est une suite finis d'arc $(a_1, a_2), (a_2, a_3), \dots, (a_{n-1}, a_n)$ tel que pour tout i, a_i pointe vers a_{i+1} et $a_1 = u, a_n = v$.

On peut aussi représenter un chemin par la suite de sommet parcouru

> **Note**
>
> Un chemin est donc une suite d'arc (ou de sommet) relié entre eux, reliant au final u à v.

Définition 0.7 (Longueur)**.** La longueur d'un chemin est le nombre d'arcs de ce dernier.

Définition 0.8 (Distance)**.** La distance entre deux sommets a et b dans un graphe est la longueur du plus court chemin entre eux.

Définition 0.9 (Chemin simple)**.** Un chemin simple est un chemin qui ne contient pas deux fois le même arc.

Définition 0.10 (Cycle)**.** Un cycle dans un graphe est un chemin simple (d'au moins 1 arc) dont le sommet d'arrivée est le même que celui de départ.

> **Avertissement**
>
> Ainsi dans un graphe non orienté, il n'est pas possible de créer un cycle avec seulement 2 sommets. En revanche, cela est possible sur un graphe orienté.

Définition 0.11 (Connexe)**.** Un graphe est connexe si tous les sommets sont accessibles entre eux par un chemin simple, peu importe le sens des arcs.

Définition 0.12 (Fortement connexe)**.** Un graphe orienté est fortement connexe si tous les sommets sont accessibles entre eux par un chemin simple, en tenant compte du sens des arcs.

> **Note**
>
> Ainsi le graphe présenté Figure 1 est connexe mais pas fortement connexe.

Représentation d'un graphe

Il existe plusieurs représentations d'un graphe. Chaque représentation a ses avantages et ses inconvénients, dont un élève de terminal est censé maitriser.

> **Important**
>
> Dans toute la suite on représentera chaque sommet par un entier positif, entre 0 et le nombre de sommets.
> Au besoin nous pouvons utiliser un dictionnaire pour convertir depuis le type voulu vers un entier.

Liste d'adjacence

Définition 0.1 (Liste d'adjacence)**.** La liste d'adjacence est une représentation de graphe où pour chaque sommet est associé à une liste des sommets adjacents au sommet en question.

On stocke par exemple dans un tableau un ensemble d'entier, ou l'indice du tableau correspond au sommet dont on veut connaitre les successeurs.

```
graphe:List[Set[int]] = []
```

> **Note**
>
> On parle ici de liste d'adjacence, puisque le graphe souhaité est mutable. On peut très bien utiliser un tableau dynamique.

Cette implémentation permet de très facilement itéré de successeur en successeur. En revanche, la complexité pour itérer de prédécesseur en prédécesseur est plus haute.

Avantages et incovéniens

La liste d'adjacence est un moyen compacte de représenter les graphes peu denses, c'est-à-dire que le nombre d'arcs est très faible comparé au nombre de sommets au carré ($A \ll B^2$), soit le nombre d'arcs possible. En effet, dans ce cas la taille mémoire est proportionnelle à $S + A$

Il n'est cependant pas recommandé dans le cas où il est soit nécessaire d'itérer sur les prédécesseurs, soit lorsqu'il est nécessaire de savoir si un arc (u, v) existe. Dans ce cas, cette implémentation rend ces opérations peu efficaces.

Il n'est pas nécessaire d'utiliser des entiers pour représenter les sommets, si nous utilisons un dictionnaire.

Matrice d'adjacence

Définition 0.2 (Matrice d'adjacence)**.** Une matrice d'ajacence est un tableau de tableau de booléen, telle qu'aun arc (u, v) existe si et seulement si $m[u][v] = \text{Vrai}$

On représente souvent une matrice d'adjacence ainsi :

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} \begin{array}{c} \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

> **Note**
>
> Dans le cas d'un graphe non orienté, la matrice est symétrique (selon la diagonale). Ainsi $m[u][v] = m[v][u]$.

Avantage et inconvéniens

Cette représentation se trouve plus lourde que la précédente, et par la même occasion, moins utilisé. En revanche contrairment à sa cousine, il est possible d'itérer sur les prédécésseur de manière optimal.

La taille mémoire occupé par cette représentation est de S^2, soit le nombre total d'arcs possibles. Cette représentation est donc peu conseillé pour des gaphe peu dense.

De même si nous revenons aux itération des succésseur, avec une matrice d'adjacence, cela reviens a tester toutes les possibilité. Pour un graphe peu dense, peu de ces possibilité d'arc snt présente, et donc cette itération, si elle est équivalente en complxité, est peu efficace dans une situation réel.

Algorithme sur les graphes

Parcours d'un graphe

Comme pour les arbres, il existe 2 types de parcours, en largeur et en profondeurs.

Définition 0.1 (Parcours en profondeur)**.** Le parcours en profondeurs est un parcours qui privilégie le parcours immédiat de ses succésseurs, les sommets les plus distant se font en premier.

Définition 0.2 (Parcours en largeur)**.** Le parcours en largeur est un parcours qui privilégie le parcours des succésseurs des noeuds de ses prédécésseurs. Il privilégie les noeuds le plus proche de l'origine (du parcours).