Structures de données

Listes chaînées - Piles et Files - Arbres - Graphes

M. Tellene

Structures de données

Nous avons vu, lors de séquence sur les protocoles de routage, qu'un réseau pouvait être représenté par un graphe. Mais qu'est-ce qu'un graphe ?

Un graphe est une structure de données constituée de sommets et de liens entre ses sommets

Structures de données

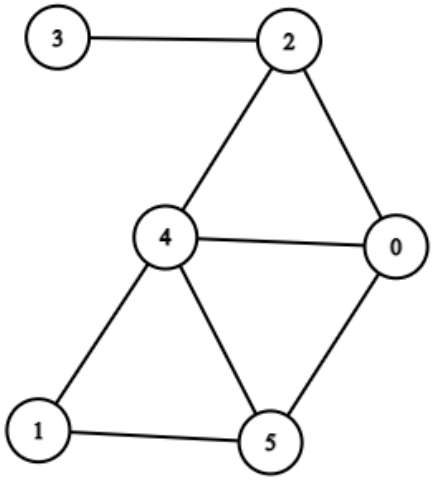
Nous avons vu, lors de séquence sur les protocoles de routage, qu'un réseau pouvait être représenté par un graphe. Mais qu'est-ce qu'un graphe ?

Un graphe est une structure de données constituée de sommets et de liens entre ses sommets

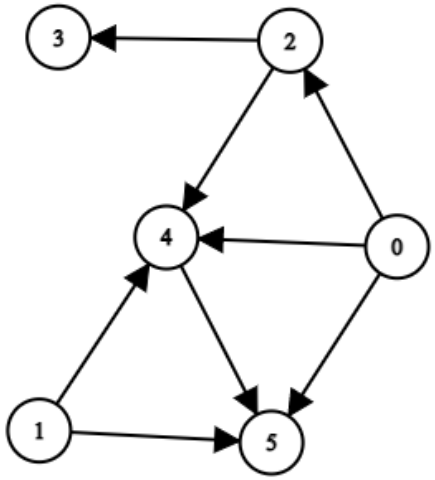
Il existe 2 types de graphes :

- les graphes orientés
- les graphes non orientés

Structures de données - Graphe



Graphe non orienté : les sommets sont reliés entre eux par des arêtes



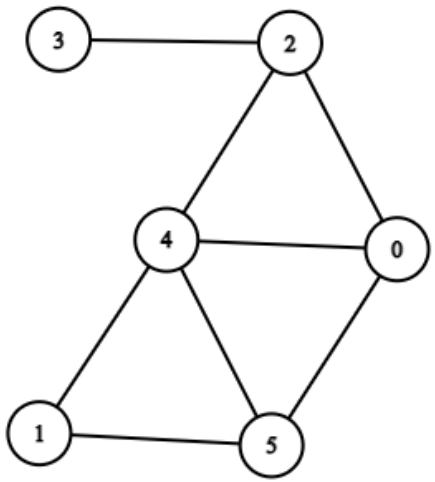
Graphe orienté : les sommets sont reliés entre eux par des arcs

Structures de données - Graphe

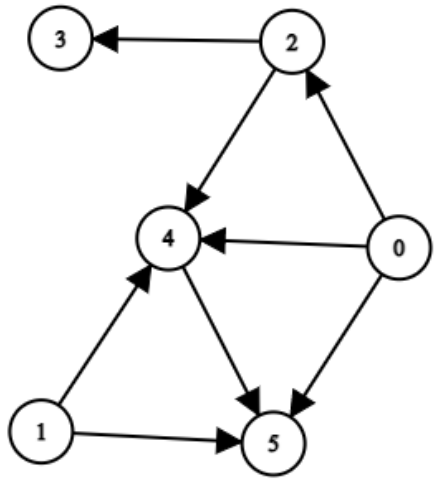
Les arcs impliquent un sens de circulation dans le graphe

Structures de données - Graphe

Les arcs impliquent un sens de circulation dans le graphe



Graphe non orienté : on peut aller du sommet 2 au sommet 3 et inversement



Graphe orienté : on peut aller du sommet 2 au sommet 3, mais l'inverse n'est pas possible

Structures de données - Graphe

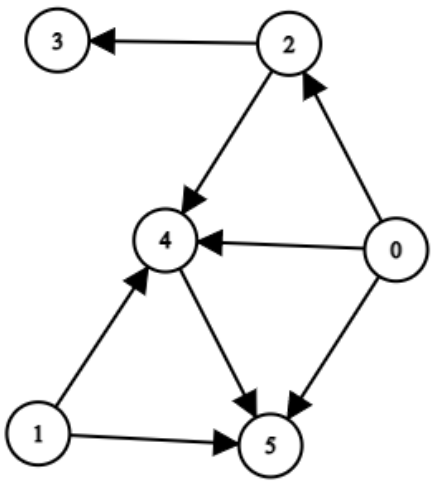
Un graphe est représenté de la manière suivante : `G=(V, E)` où

- `V` est un ensemble contenant tous les sommets du graphe
- `E` est un ensemble contenant tous les arcs ou arêtes

Structures de données - Graphe

Un graphe est représenté de la manière suivante : `G=(V, E)` où

- `V` est un ensemble contenant tous les sommets du graphe
- `E` est un ensemble contenant tous les arcs ou arêtes




```
V = {0,1,2,3,4,5}
E = {(0,2), (0,4), (0,5),
(1,4), (1,5), (2,3), (2,4),
(4,5)}
```

Structures de données - Graphe

Voisin : lorsqu'il existe un lien entre un sommet `s` et un sommet `t`, on dit que `s` et `t` sont **adjacents** ou **voisins**

Structures de données - Graphe

Voisin : lorsqu'il existe un lien entre un sommet `s` et un sommet `t`, on dit que `s` et `t` sont **adjacents** ou **voisins**

Chemin : un chemin reliant un sommet `s` à un sommet `t` est une séquence finie de sommets reliés par des liens

Structures de données - Graphe

Voisin : lorsqu'il existe un lien entre un sommet `s` et un sommet `t`, on dit que `s` et `t` sont **adjacents** ou **voisins**

Chemin : un chemin reliant un sommet `s` à un sommet `t` est une séquence finie de sommets reliés par des liens

Un chemin est dit *simple* s'il n'emprunte pas deux fois le même lien

Un chemin est dit *élémentaire* s'il ne passe pas deux fois par le même sommet

Structures de données - Graphe

Voisin : lorsqu'il existe un lien entre un sommet `s` et un sommet `t`, on dit que `s` et `t` sont **adjacents** ou **voisins**

Chemin : un chemin reliant un sommet `s` à un sommet `t` est une séquence finie de sommets reliés par des liens

Un chemin est dit *simple* s'il n'emprunte pas deux fois le même lien

Un chemin est dit *élémentaire* s'il ne passe pas deux fois par le même sommet

Cycle : un cycle est un chemin simple reliant un sommet à lui-même et contenant au moins un lien

Structures de données - Graphe

Distance : la longueur d'un chemin est définie comme le nombre de liens qui constituent ce chemin. La distance entre 2 sommets est la longueur du plus court chemin reliant ces deux sommets

Il est à noter que la distance entre 2 sommets n'est pas définie s'il n'existe aucun chemin entre ces deux sommets

Structures de données - Graphe

Distance : la longueur d'un chemin est définie comme le nombre de liens qui constituent ce chemin. La distance entre 2 sommets est la longueur du plus court chemin reliant ces deux sommets

Il est à noter que la distance entre 2 sommets n'est pas définie s'il n'existe aucun chemin entre ces deux sommets

Connexité :

- on dit qu'un graphe non orienté est *connexe* si pour toute paire `u`, `v`, il existe un chemin entre `u` et `v`

Structures de données - Graphe

Distance : la longueur d'un chemin est définie comme le nombre de liens qui constituent ce chemin. La distance entre 2 sommets est la longueur du plus court chemin reliant ces deux sommets

Il est à noter que la distance entre 2 sommets n'est pas définie s'il n'existe aucun chemin entre ces deux sommets

Connexité :

- on dit qu'un graphe non orienté est *connexe* si pour toute paire `u`, `v`, il existe un chemin entre `u` et `v`
- on dit qu'un graphe orienté est *connexe* si le graphe obtenu en oubliant le sens des arcs est connexe

Structures de données - Graphe

Distance : la longueur d'un chemin est définie comme le nombre de liens qui constituent ce chemin. La distance entre 2 sommets est la longueur du plus court chemin reliant ces deux sommets

Il est à noter que la distance entre 2 sommets n'est pas définie s'il n'existe aucun chemin entre ces deux sommets

Connexité :

- on dit qu'un graphe non orienté est *connexe* si pour toute paire `u`, `v`, il existe un chemin entre `u` et `v`
- on dit qu'un graphe orienté est *connexe* si le graphe obtenu en oubliant le sens des arcs est connexe
- on dit qu'un graphe orienté est *fortement connexe* lorsqu'il existe un chemin de `u` à `v` **et** un chemin de `v` à `u` pour toute paire `u`, `v` de sommets

Structures de données - Graphe

Qu'est-ce qui peut être représenté par un graphe ?

Structures de données - Graphe

Qu'est-ce qui peut être représenté par un graphe ?

- un réseau routier, réseau informatique, réseau social
- une carte du monde (par exemple)
- un labyrinthe
- un jeu comme les dames, les échecs, le morpion, ...
- un arbre

Structures de données - Graphe

Comment représenter un graphe en machine ?

Deux manières possibles :

- en utilisant une matrice d'adjacence
- en utilisant un dictionnaire d'adjacence

Structures de données - Graphe

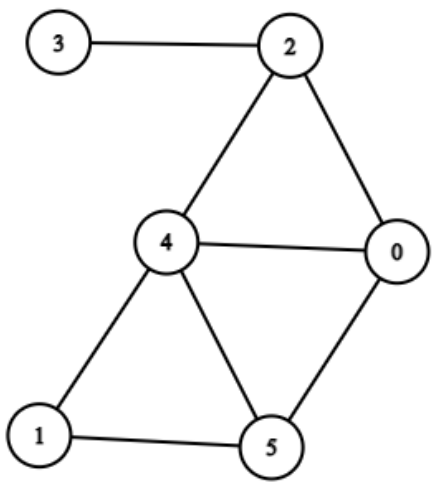
Matrice d'adjacence :

- les sommets du graphe sont supposés être les entiers 0, ..., $N - 1$
- matrice `adj` de booléens de taille $N \times N$
- le booléen `adj[i][j]` indique la présence d'un lien entre `i` et `j`

Structures de données - Graphe

Matrice d'adjacence :

- les sommets du graphe sont supposés être les entiers 0, ..., $N - 1$
- matrice `adj` de booléens de taille $N \times N$
- le booléen `adj[i][j]` indique la présence d'un lien entre `i` et `j`



	0	1	2	3	4	5
0	0	0	1	0	1	1
1	0	0	0	0	1	1
2	1	0	0	1	1	0
3	0	0	1	0	0	0
4	1	1	1	0	0	1
5	1	1	0	0	1	0

Structures de données - Graphe

Efficacité de la matrice d'adjacence :

Structures de données - Graphe

Efficacité de la matrice d'adjacence :

- simple à mettre en place
- occupe un espace mémoire proportionnel à $N \times N$
- pour connaître tous les voisins d'un sommet, il faut parcourir toute une ligne (même s'il n'y a pas beaucoup de voisin)
- limite les sommets à des entiers

Structures de données - Graphe

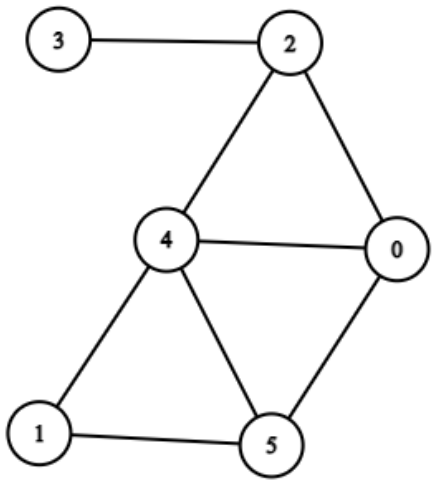
Dictionnaire d'adjacence :

- les sommets du graphe ne sont pas limités aux entiers 0, ..., $N-1$
- dictionnaire `adj` où les clés sont les sommets et les valeurs la liste des voisins du sommet

Structures de données - Graphe

Dictionnaire d'adjacence :

- les sommets du graphe ne sont pas limités aux entiers 0, ..., $N-1$
- dictionnaire `adj` où les clés sont les sommets et les valeurs la liste des voisins du sommet




```
{
    0 : [2,4,5],
    1 : [4,5],
    2 : [0,3,4],
    3 : [2],
    4 : [0,1,2,5],
    5 : [0,1,4]
}
```

Structures de données - Graphe

Efficacité du dictionnaire d'adjacence :

3. $|V|$ est la cardinalité de V, soit le nombre de sommets et $|E|$ est la cardinalité de E, soit le nombre de liens

Structures de données - Graphe

Efficacité du dictionnaire d'adjacence :

- le coût des opérations est optimal (ajout, présence, parcours)
- occupe un espace mémoire proportionnel à $|V| + |E|$ [3]
- parcourir les voisins d'un sommet donné se fait en temps proportionnel au nombre de des voisins
- permet d'utiliser des chaînes de caractères, des objets, ... pour « nommer » les sommets

3. $|V|$ est la cardinalité de V, soit le nombre de sommets et $|E|$ est la cardinalité de E, soit le nombre de liens

Structures de données - Graphe

Algorithmes possibles sur les graphes :

- recherche d'une chaîne
- présence d'un cycle
- vérification de connexité
- coloration de graphe
- ...

Structures de données - Graphe

Algorithmes possibles sur les graphes :

- recherche d'une chaîne
- présence d'un cycle
- vérification de connexité
- coloration de graphe
- ...

Tous ces algorithmes se basent sur le parcours de graphe

Structures de données - Graphe

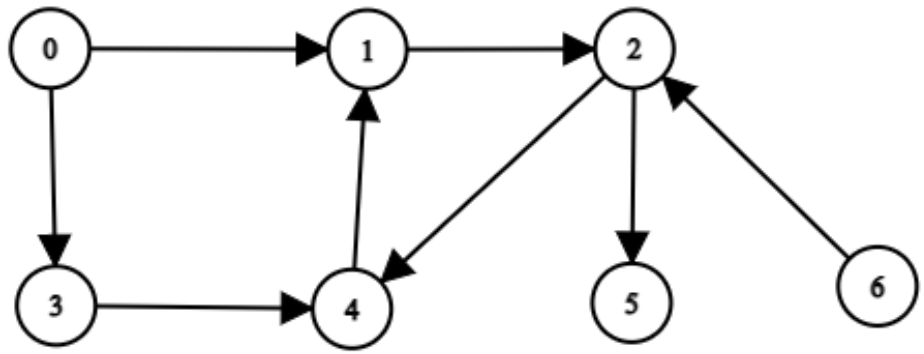
Comme pour les arbres, il existe plusieurs parcours :

- parcours en profondeur
- parcours en largeur

Structures de données - Graphe

Le parcours en profondeur est un parcours s'appliquant à n'importe quel graphe et permet de déterminer tous les sommets atteignables depuis un sommet donné

Soit le graphe suivant :



Lançons le parcours en profondeur sur 0

Structures de données - Graphe

sommet visité	action
0	marquer 0, emprunter arc $0 \rightarrow 1$

Structures de données - Graphe

sommet visité	action
0	marquer 0, emprunter arc $0 \rightarrow 1$
.1	marquer 1, emprunter arc $1 \rightarrow 2$

Structures de données - Graphe

sommet visité	action
0	marquer 0, emprunter arc $0 \rightarrow 1$
.1	marquer 1, emprunter arc $1 \rightarrow 2$
..2	marquer 2, emprunter arc $2 \rightarrow 4$
...4	marquer 4, emprunter arc $4 \rightarrow 1$
....1	déjà vu

Structures de données - Graphe

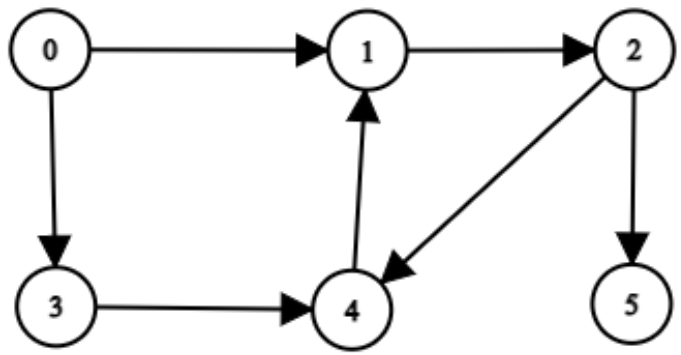
sommet visité	action
0	marquer 0, emprunter arc $0 \rightarrow 1$
.1	marquer 1, emprunter arc $1 \rightarrow 2$
..2	marquer 2, emprunter arc $2 \rightarrow 4$
...4	marquer 4, emprunter arc $4 \rightarrow 1$
....1	déjà vu
...4	pas d'autre arc, terminé

Structures de données - Graphe

sommet visité	action
0	marquer 0, emprunter arc 0 $\rightarrow$ 1
.1	marquer 1, emprunter arc 1 $\rightarrow$ 2
..2	marquer 2, emprunter arc 2 $\rightarrow$ 4
...4	marquer 4, emprunter arc 4 $\rightarrow$ 1
....1	déjà vu
...4	pas d'autre arc, terminé
..2	emprunter arc 2 $\rightarrow$ 5
...5	marquer 5, pas d'autre arc
..2	pas d'autre arc
.1	pas d'autre arc
0	emprunter arc 0 $\rightarrow$ 3
.3	marquer 3, emprunter arc 3 $\rightarrow$ 4
..4	déjà vu
.3	pas d'autre arc
0	pas d'autre arc

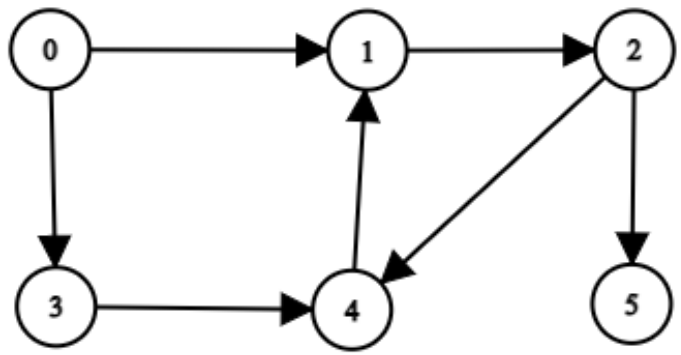
Structures de données - Graphe

Il est possible de construire un graphe résultant de ce parcours :



Structures de données - Graphe

Il est possible de construire un graphe résultant de ce parcours :



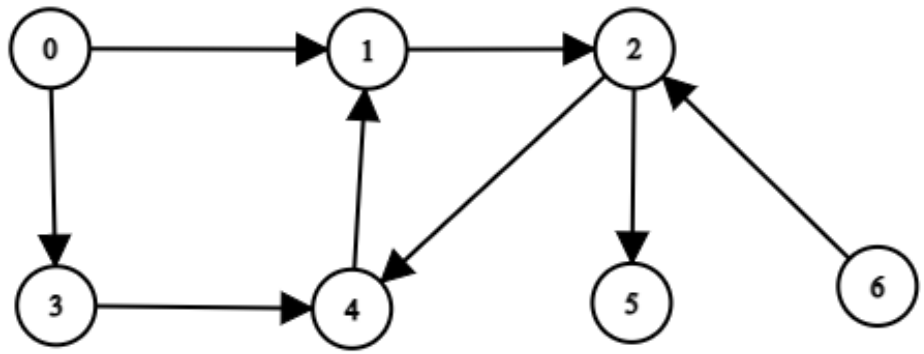
Complexité : O(nombre d'arcs examinés)

- pire des cas : tous les sommets sont atteignables $\rightarrow N$

Structures de données - Graphe

Le parcours en largeur nous permet de déterminer l'existence d'un chemin entre deux sommets, et même d'en construire un, mais pas de déterminer la distance entre deux sommets

Soit le graphe suivant :



Lançons le parcours en profondeur sur 0

Structures de données - Graphe

courant	suivant	action
0	1, 3	marquer 0

Structures de données - Graphe

courant	suivant	action
0	1, 3	marquer 0
1, 3	2	marquer 1

Structures de données - Graphe

courant	suivant	action
0	1, 3	marquer 0
1, 3	2	marquer 1
3, 2	4	marquer 3
2, 4	4, 5	marquer 2
4, 4, 5	1	marquer 4

Structures de données - Graphe

courant	suivant	action
0	1, 3	marquer 0
1, 3	2	marquer 1
3, 2	4	marquer 3
2, 4	4, 5	marquer 2
4, 4, 5	1	marquer 4
4, 5, 1		4 déjà vu
5, 1		marquer 5
1		1 déjà vu

Structures de données - Graphe

Le parcours en profondeur et le parcours en largeur doivent donner les mêmes sommets parcourus mais dans un ordre possiblement différent

Structures de données - Graphe

Le parcours en profondeur et le parcours en largeur doivent donner les mêmes sommets parcourus mais dans un ordre possiblement différent

Mais comment nous aider à écrire ces algorithmes de parcours ?

Structures de données - Graphe

Le parcours en profondeur et le parcours en largeur doivent donner les mêmes sommets parcourus mais dans un ordre possiblement différent

Mais comment nous aider à écrire ces algorithmes de parcours ?

En utilisant des structures de données !

- Une pile pour le parcours en profondeur
- Une file pour le parcours en largeur