Une structure de données est une organisation logique des données permettant de simplifier ou d'accélérer leur traitement.

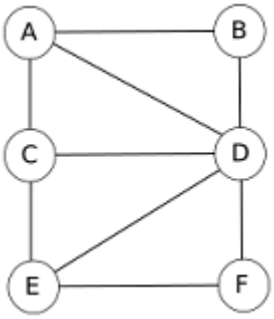
Sources : Laurent Godefroy, Stéphane Grandcolas, Benjamin Monmège, Didier Müller, David Roche, Dominique Laporte, wikipedia

1. Graphes

Un graphe fini G = (V, E) est défini par l'ensemble fini V = {v_1, v_2, ..., v_n } dont les éléments sont appelés sommets, et par l'ensemble fini E = {e_1, e_2, ..., e_m } dont les éléments sont appelés arêtes. Une arête e de l'ensemble E est définie par une paire non-ordonnée de sommets, appelés les extrémités de e. Si l'arête e relie les sommets a et b, on dira que ces sommets sont adjacents.

> Plus formellement on dira qu'un graphe G est un couple G = (V,E) avec V un ensemble de sommets et E un ensemble d'arêtes

1.1. Représentation des graphes



Imaginons un réseau social ayant 6 abonnés (A, B, C, D, E et F) où :

- A est ami avec B, C et D
- B est ami avec A et D
- C est ami avec A, E et D
- D est ami avec tous les autres abonnés
- E est ami avec C, D et F
- F est ami avec E et D

On peut représenter chaque sommet par un cercle (avec le nom de l'abonné situé dans le cercle) et chaque relation "X est ami avec Y" par un segment de droite reliant X et Y ("X est ami avec Y" et "Y est ami avec X" étant représenté par la même arête).

Exemple : Construire un graphe de réseau social à partir des informations suivantes :

- A est ami avec B et E
- B est ami avec A et C
- C est ami avec B,F et D
- D est ami avec C,F et E
- E est ami avec A,D et F
- F est ami avec C, D et E

1.2. Graphe orienté

Un graphe orienté est la donnée :

- d'un ensemble S de sommets.
- et d'un ensemble A d'arcs (arête orientée), chaque arc étant un couple (u,v) de deux sommets (possiblement les deux mêmes), parfois noté u → v : u est la source de l'arc et v sa destination.

Un chemin est une suite de sommets u_0, u_1, ..., u_k (avec k ≥ 0) reliés par des arcs, c'est-à-dire telle que u_i → u_{i+1} pour tout i ∈ {0, ..., k − 1}.

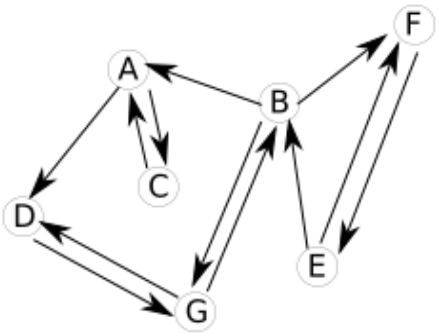
> On dira qu'un graphe orienté G est un couple G = (V,A) avec V un ensemble de sommets et A un ensemble d'arcs.

Prenons le cas des logiciels de cartographie (ces logiciels sont souvent utilisés couplés à des récepteurs GPS). Ces logiciels de cartographie permettant, connaissant votre position grâce à un récepteur GPS, d'indiquer la route à suivre pour se rendre à endroit précis.

Soit les lieux suivants : A, B, C, D, E, F et G.

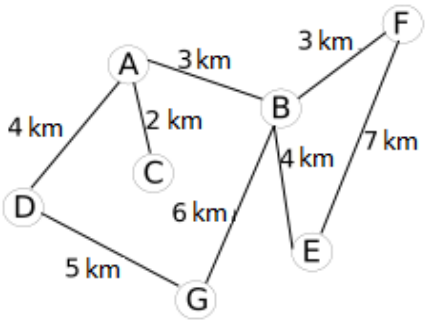
Les différents lieux sont reliés par les routes suivantes :

- il existe une route entre A et C (double sens)
- il existe une route entre A et B (sens unique B->A)
- il existe une route entre A et D (sens unique A->D)
- il existe une route entre B et F (sens unique B->F)
- il existe une route entre B et E (sens unique E->B)
- il existe une route entre B et G (double sens)
- il existe une route entre D et G (double sens)
- il existe une route entre E et F (double)



On obtient alors la représentation sous forme de graphe ci-contre.

Parfois il est intéressant d'associer aux arrêtes ou aux arcs des valeurs, on parle alors de graphes pondérés. Si nous revenons à notre "graphe cartographie", il est possible d'associer à chaque arête la distance en km entre les 2 lieux.



1.3. Caractérisation des arbres

7.3.1. Connexité

Une composante connexe d'un graphe non orienté est un ensemble maximal de sommets tous reliés les uns aux autres par un chemin. Un graphe peut se décomposer de manière unique en un ensemble de composantes connexes.

Un graphe est dit connexe s'il ne possède qu'une unique composante connexe.

Une notion similaire existe pour les graphes orientés : la forte connexité.

7.3.2. Acyclicité

Un circuit dans un graphe non orienté est une chaîne partant et arrivant dans le même sommet. On parle de cycle dans un graphe orienté.

Un graphe est dit acyclique s'il ne possède aucun circuit/cycle.

> Tout graphe connexe et acyclique est un arbre...
>
> ... au sens où on peut choisir n'importe quel sommet comme racine et obtenir un arbre d'arité non borné en considérant ses voisins, puis les voisins de ses voisins, etc.

1.4. Implémentation d'un graphe

Il existe deux méthodes permettant d'implémenter un graphe : les matrices d'adjacences et les listes d'adjacences.

7.4.1. Matrice d'adjacence

Il faut savoir qu'à chaque ligne correspond un sommet du graphe et qu'à chaque colonne correspond aussi un sommet du graphe. À chaque intersection ligne i-colonne j (ligne i correspond au sommet i et colonne j correspond au sommet j), on place un 1 s'il existe une arête entre le sommet i et le sommet j, et un zéro s'il n'existe pas d'arête entre le sommet i et le sommet j.

Reprenons l'exemple du "graphe cartographie" :

- Il existe une arête entre le sommet E et le sommet F, nous avons donc placé un 1 à l'intersection de la ligne E et de la colonne F (il en est de même à l'intersection de la ligne F et de la colonne E)
- Il n'existe pas d'arête entre le sommet D et le sommet C, nous avons donc placé un 0 à l'intersection de la ligne D et de la colonne C (il en est de même à l'intersection de la ligne C et de la colonne D)

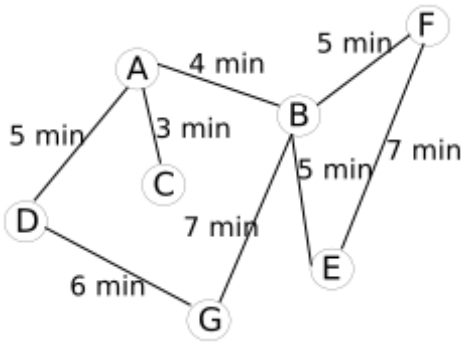
	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	0	0	1	1	1
C	1	0	0	0	0	0	0
D	1	0	0	0	0	0	1
E	0	1	0	0	1	0	0
F	0	1	0	0	1	0	0
G	0	1	0	1	0	0	0

NB : il est aussi possible d'établir une matrice d'adjacence pour un graphe orienté. Le principe reste le même : si le sommet i (ligne) est lié au sommet j (colonne), nous avons un 1 à l'intersection (0 dans le cas contraire).

Il est assez simple d'utiliser les matrices d'adjacence en Python à l'aide des listes

```
#matrice d'adjacence pour le graphe cartographie.
m = [[0, 1, 1, 1, 0, 0, 0],
     [1, 0, 0, 0, 1, 1, 1],
     [1, 0, 0, 0, 0, 0, 0],
     [1, 0, 0, 0, 0, 0, 1],
     [0, 1, 0, 0, 0, 1, 0],
     [0, 1, 0, 0, 1, 0, 0],
     [0, 1, 0, 1, 0, 0, 0]]
```

Pour implémenter un graphe pondéré : on remplace les 1 par les valeurs liées à chaque arc.



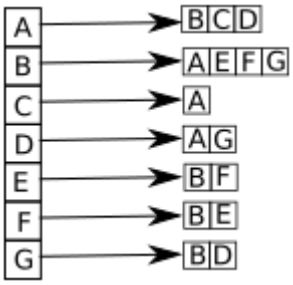

```
0 4 3 5 0 0 0
4 0 0 0 5 5 7
3 0 0 0 0 0 0
5 0 0 0 0 0 6
0 5 0 0 0 7 0
0 5 0 0 7 0 0
0 7 0 6 0 0 0
```

7.4.2. Liste d'adjacence

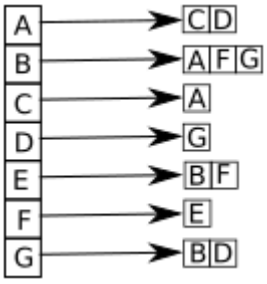
On peut aussi représenter un graphe en donnant pour chacun de ses sommets la liste des sommets auxquels il est adjacent. C'est la méthode qui est utilisée pour implémenter un graphe sur ordinateur.

Exemple du "graphe cartographie" :

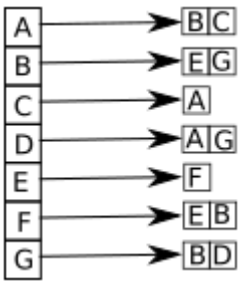
graphe non orienté



graphe orienté



liste d'adjacence successeurs



liste d'adjacence prédécesseurs

Pour les graphes orientés, il est nécessaire de définir 2 listes : la liste des successeurs et la liste des prédécesseurs. Soit un arc allant d'un sommet A vers un sommet B (flèche de A vers B). On dira que B est un successeur de A et que A est un prédécesseur de B.

Il est possible de travailler avec des listes d'adjacences en Python en utilisant les dictionnaires :

```
#liste d'adjacence pour le graphe non orienté cartographie
liste = {
   'A':('B','C','D'),
   'B':('A', 'E', 'F', 'G'),
   'C':('A'),
   'D':('A', 'G'),
   'E':('B', 'F'),
   'F':('B', 'E'),
   'G':('B', 'D')
   }
```

Exercice

Reliez chaque tâche à effectuer à la structure de données la plus adéquate.

stocker les dernières pages web visitées dans un navigateur	**1**	**A**	file
écrire tous les mots dont le début est connu	**2**	**B**	tas
retrouver instantanément un nom dans un annuaire	**3**	**C**	arbre AVL
connaître à tout moment le prochain événement à traiter dans une simulation (p. ex. la collision entre 2 boules de billard)	**4**	**D**	trie
trier des nombres donnés au fur et à mesure	**5**	**E**	table de hachage
traiter les demandes sur une imprimante branchée en réseau	**6**	**F**	pile