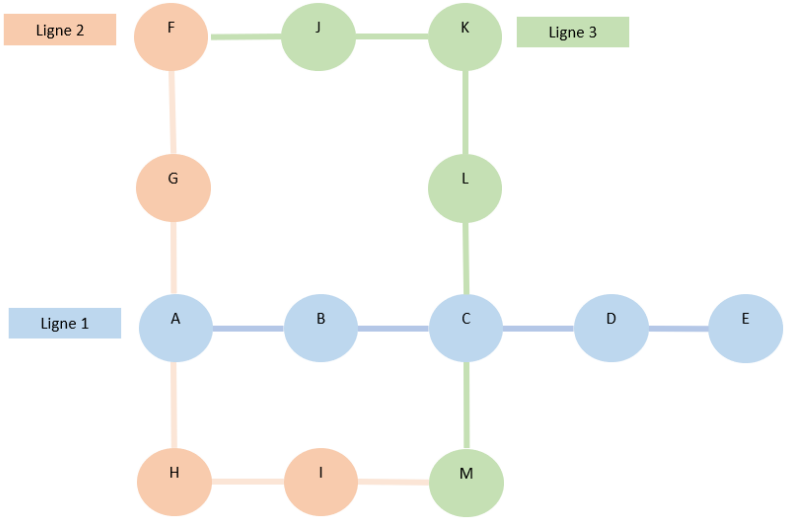
Informatique

11
Bases des graphes

TD 11-1
Parcours

Exercice 1: Parcours

Voici un plan de métro :



Le réseau est composé de trois lignes. Les stations sont nommées par des lettres afin de faciliter l'étude. Nous allons programmer les algorithmes de parcours en largeur et profondeur.

On définit le réseau ainsi :

```
Ligne_1 = ['A','B','C','D','E']
Ligne_2 = ['F','G','A','H','I','M']
Ligne_3 = ['F','J','K','L','C','M']
Lignes = [Ligne_1,Ligne_2,Ligne_3]

Segments_1 = [[1,1],[1,1],[1,1],[1,1]]
Segments_2 = [[1,1],[1,1],[1,1],[1,1],[1,1]]
Segments_3 = [[1,1],[1,1],[1,1],[1,1],[1,1]]
Segments = [Segments_1,Segments_2,Segments_3]
```

On notera que les segments définis ci-dessus permettent de définir les passages possibles d'une station à l'autre dans les deux sens. Ainsi, bloquer le segment de B à C et de B à A revient à écrire :

```
Segments_1 = [[1,0],[0,1],[1,1],[1,1]]
```

A partir de cela, on propose les fonctions permettant de créer automatiquement

- La liste **Stations** des différentes stations du réseau
- Le nombre de stations **Nb**
- La matrice **Graphe** (array de numpy) d'adjacence du réseau

Téléchargez le fichier en lien ici afin d'avoir le graphe à disposition. Vous le compléterez à la suite dans la partie réservée à l'algorithme de Dijkstra.

Préliminaires

Question 1: Mettre en place une fonction f_Voisins(S) qui renvoie le liste des voisins accessibles de S

Vérifier :

```
>>> f_Voisins('F')
['G', 'J']

>>> f_Voisins('C')
['B', 'D', 'M', 'L']
```

Remarques :

- Vous partirez évidemment de la matrice d'adjacence créée par mon code élèves
- Pour obtenir les mêmes résultats que dans le cours et l'animation proposée, on cherchera les voisins de S dans l'ordre la liste Stations.
- On pourra utiliser la commande « `iS = Stations.index(S)` » pour obtenir l'indice `iS` dans `Stations` de la station `S`

Parcours en largeur

L'algorithme se déroule ainsi :

- Mettre le nœud de départ dans une file et le marquer comme visité
- Tant que la file n'est pas vide :
  - Retirer le premier sommet de la file pour le traiter
  - Mettre tous ses voisins non visités à la fin de la file et les marquer visités

On utilisera dans cette partie une collection (deque) pour réaliser la file des stations à traiter.

Question 2: Mettre en place le code permettant de réaliser le parcours du graphe des stations depuis la station F par un parcours en largeur

Aide :

- Une boucle infinie ? Avez-vous bien exclu les voisins visités ?
- Pensez à ajouter les voisins un par un dans Visite et dans la file

Question 3: Afficher la liste des stations explorées dans l'ordre dans lequel elles l'ont été

Question 4: Améliorer cet algorithme afin qu'il renvoie un dictionnaire Distances tel que Distance[S] est le nombre de stations pour atteindre S

Exemple :

```
>>> Distances['A']
2

>>> Distances['E']
6
```

Vous pourrez vérifier votre algorithme en vous aidant de l'exemple d'exécution suivant : LIEN PDF (à télécharger et ouvrir en mode présentation 😊).

Remarque : on peut croire que le simple fait de sortir le dernier sommet de la file (le dernier entré) en traitant une pile permet de réaliser un parcours en profondeur. Amusez vous à modifier légèrement votre code pour vous convaincre que cela ne fonctionne pas.

Parcours en profondeur

L'algorithme se déroule ainsi :

- Fonction récursive d'exploration Explorer(s) :
    - Marquer le sommet s comme visité
        - Pour tout voisin v de s non marqué :
            - Explorer(v)
- Le parcours total est alors réalisé ainsi :
    - Pour tout sommet s non marqué du graphe :
        - Explorer(s)

On utilisera une liste globale Visite contenant les stations marquées.

Question 5: Mettre en place la fonction Explorer(S) réalisant le travail demandé
Question 6: Mettre en place le code permettant de réaliser le parcours du graphe des stations par un parcours en profondeur
Question 7: Afficher la liste des stations visitées dans l'ordre dans lequel elles l'ont été

Vous pourrez vérifier votre algorithme en vous aidant de l'exemple d'exécution suivant : LIEN PDF (à télécharger et ouvrir en mode présentation 😉).

Question 8: Commenter le résultat de l'appel Explorer(S) quelle que soit la station S
Remarque : pensez à réinitialiser les stations marquées de l'exécution précédente

Ajouter à votre code les 3 lignes suivantes :

```
Segments_1 = [[1,1],[1,1],[0,1],[1,1]]
Segments = [Segments_1,Segments_2,Segments_3]
Graphe = f_Graphe()
```

Question 9: Préciser le changement réalisé
Question 10: Commenter alors le résultat de l'appel Explorer('F')