Devoir n°2 Graphes 1h15

Exercice 1 – parcours en largeur d'un graphe non orienté

Soit le graphe de la figure 1 ci-contre :

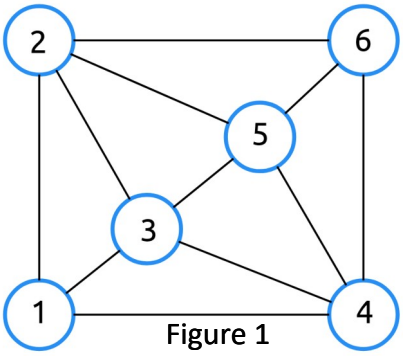


Figure 1

On souhaite créer une implémentation de ce graphe par listes d'adjacences. On décide d'ordonner les sommets adjacents à un sommet donné par ordre croissant.

1. Donner la liste d'adjacence du sommet 5.

On rappelle l'algorithme de principe du parcours en largeur :

```
ParcoursLargeur(Graphe G, Sommet s):
 F ← []
 F.enfiler(s)
 marquer s;
 tant F n'est pas vide :
 | u ← F.defiler()
 | pour tout voisin v de u :
 | | si v non marqué
 | | | F.enfiler(v);
 | | | marquer v;
```

On se propose d'étudier l'application de cet algorithme sur le graphe de la figure 1, à partir du sommet 5.

2. Pourquoi est-il nécessaire de « marquer » les sommets visités ?

3.a. Donner l'ordre dans lequels seront visités les sommets du graphe.

3.b. Donner l'arbre du parcours en largeur ainsi suivi.

3.c. Quel est la diamètre de ce graphe ?

On propose ci-dessous une implémentation du parcours en largeur d'un graphe en Python.

```python
def creer_parcours_largeur(graphe, sommet_depart):
    parcours = {}
    parcours[sommet_depart] = {"couleur": "gris", "parent": None, "distance": 0}
    for sommet in graphe :
        if sommet != sommet_depart :
            parcours[sommet] = {"couleur": "blanc", "parent": None, "distance": None}

    file = [sommet_depart]
    while file != []:
        sommet_visite = file.pop(0)
        for voisin in graphe[sommet_visite] :
            if parcours[voisin]["couleur"] == "blanc" :
                parcours[voisin]["couleur"] = "gris"
                parcours[voisin]["parent"] = sommet_visite
                parcours[voisin]["distance"] = parcours[sommet_visite]["distance"] + 1
                file.append(voisin)
        parcours[sommet_visite]["couleur"] = "noir"
    return parcours
```

4. Décrire la structure de la variable `parcours` : que seront les clés de ses éléments ? Que seront les valeurs associées à chaque clé (on ne demande pas le contenu de la variable `parcours` à la fin de l'exécution de la fonction, mais sa structure générale).

5. À quoi servent les lignes 2 à 6 de la fonction ? Une réponse brève est attendue.

6. Qu'est-ce qui caractérise les trois couleurs possibles des sommets (blanc, gris et noir) ?

7. Si, après exécution de la fonction, la variable parcours renvoyée contient encore des sommets dans la couleur est « blanc », que peut-on en déduire sur le graphe ?

Ex.2 Parcours en profondeur d'un graphe orienté

Soit le graphe de la figure 2 ci-contre.

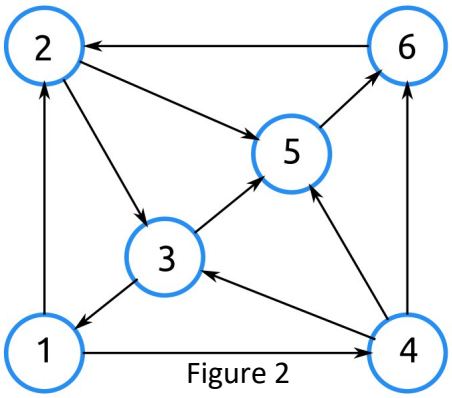


Figure 2

On implémente ce graphe par listes d'adjacences en ordonnant les sommets adjacents à un sommet donné par ordre croissant.

On rappelle l'algorithme de principe du parcours en profondeur :

```
1  explorer(graphe G, sommet s)
2    marquer s
3    pour tout sommet v voisin de s
4    |    si v n'est pas marqué alors
5    |    | explorer(G, v)
```

On se propose d'étudier l'application de cet algorithme sur le graphe de la figure 2, à partir du sommet 1.

1. Donner l'ordre dans lequels seront visités les sommets du graphe.

On donne ci-dessous l'algorithme formel de détection de cycle dans un graphe orienté :

```
1  explorer_cycle_oriente(graphe g, sommet s):
2    marquer s comme "en cours"
3    pour tout sommet v successeur de s :
4    |   si v est marqué "en cours" :
5    |   |   renvoyer True
6    |   sinon si v n'est pas marqué "exploré" :
7    |   |   si explorer_cycle_oriente(g, v) : renvoyer True
8    marquer s comme "exploré"
9    renvoyer False
```

On se propose d'étudier le déroulement de cet algorithme appliqué au graphe de la figure 2, en partant du sommet 5.

2. Quelle va être la succession des sommets visités jusqu'à la détection d'un cycle ? Comment le cycle est-il détecté ?

Correction

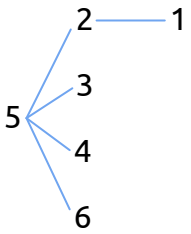
Ex.1

1. [2,3,4,6] **[1]**

2. Pour éviter de revisiter des sommets déjà visités (boucles infinies) **[1]**

3.a 5→2→3→4→6→1 **[1]**

3.b Arbre de parcours en largeur **[1]**



C si le sommet 1 est mal placé

3.c Diamètre = 2 (longueur de chemin maximale) **[1]**

A si le raisonnement est juste sur l'arbre

4. clé = sommet, valeur = dictionnaire avec les clés "couleur", "parent" et "distance" **[1]**

0 si confusion parcours et dictionnaire "couleur", "parent" et "distance"

5. Initialisation du graphe (sommet départ en gris) et tous les autres en blanc **[1]**

6. blanc = non découvert, gris = découvert mais non exploré, noir = exploré **[1]**

D si seulement « noir » correct

7. Tous les sommets du graphe n'ont pas été découverts lors du parcours. Le graphe n'est donc pas connexe. **[1]**

Ex. 2

1. 1→2→3→5→6→4 **[1]**

2. 5→6→2→3→1→2. Lorsqu'il arrive à 2, il détecte un cycle. **[1]**

D si parcours pas correct mais raisonnement juste pour détection d'un cycle

C si faute mineure dans le parcours