NSI · TERMINALE / EXERCICES CORRIGÉS

Parcourir un graphe

Astrolab Cours · astrolabcours.fr

Ex. 1 Un parcours en largeur ●○○ Application

Avec `g = {"A": ["B", "C"], "B": ["A", "D"], "C": ["A", "D", "E"], "D": ["B", "C", "F"], "E": ["C"], "F": ["D"]}`, donner l'ordre de visite du parcours en largeur depuis A.

Indice

Défiler un sommet, enfiler ses voisins non visités dans l'ordre de la liste.

La file

A → enfile B, C. B → enfile D. C → enfile E (D déjà vu). D → enfile F. E, F : rien de nouveau.

L'ordre

A, B, C, D, E, F.

Ex. 2 Un parcours en profondeur ●●○ Méthode

Avec le même graphe, donner l'ordre de visite du parcours en profondeur récursif du cours depuis A.

Indice

On descend dans le premier voisin non visité avant de regarder les autres.

La descente

A, puis B (premier voisin), puis D (premier voisin non visité de B), puis C (premier voisin non visité de D), puis E. On remonte jusqu'à D, qui a encore F.

L'ordre

A, B, D, C, E, F.

Ex. 3 Le parcours qui ne finit pas ●●○ Piège

Un élève écrit :

© Astrolab Cours · astrolabcours.fr/nsi/terminale/parcours-de-graphes/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

```
def explore(g, s):
    print(s)
    for v in g[s]:
        explore(g, v)
```

Sur le graphe précédent, le programme plante. Pourquoi ?

Indice

Que se passe-t-il entre A et B ?

L'erreur

A appelle B, qui appelle A, qui appelle B... Rien n'empêche de revisiter un sommet : la récursion ne s'arrête jamais.

La correction

Il faut mémoriser les sommets visités et n'explorer que les voisins pas encore vus, comme dans la fonction du cours.

Ex. 4 Distance minimale ●●● Type bac

Modifier le parcours en largeur pour qu'il renvoie un dictionnaire donnant, pour chaque sommet atteint, sa distance (en nombre d'arêtes) au départ.

Indice

Un voisin découvert depuis `s` est à la distance de `s` plus 1.

L'idée

Le dictionnaire des distances sert aussi de liste des visités : un sommet est visité dès qu'il a une distance.

La fonction

```
from collections import deque

def distances(g, depart):
    dist = {depart: 0}
    file = deque([depart])
    while file:
        s = file.popleft()
        for v in g[s]:
            if v not in dist:
                dist[v] = dist[s] + 1
                file.append(v)
    return dist
```

© Astrolab Cours · astrolabcours.fr/nsi/terminale/parcours-de-graphes/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits

© Astrolab Cours · astrolabcours.fr/nsi/terminale/parcours-de-graphes/ · Tous droits réservés : usage personnel uniquement, republication et usage commercial interdits