Python: module graph

Étienne Miquey
emiquey@fing.edu.uy

1 Prise en main rapide

Avant toute choses, copier le fichier `graph.py` dans votre répertoire de travail. Enregistrer votre feuille de code ouverte avec IDLE dans ce même répertoire. Votre feuille de code devra commencer par la commande suivante, afin de charger le module `graph` :

```
from graph import *
```

Voici un extrait du fichier `test.py`, qui contient des exemples d'utilisation de certaines méthodes du module `graph`. Avec ceci, vous devriez être capable de programmer les algorithmes de recherche de chemin. La section suivante contient une documentation exhaustive de toutes les méthodes dont vous pourrez avoir besoin.

```
from graph import *
gr = graph()
gr.add_nodes(['X','Y','Z'])
gr.add_edge(('X','Z'))
gr.add_edges([('A','B'),('A','C'),('Y','B')])
gr.has_node('A'); gr.nodes();gr.has_edge(('A','X'));gr.edges()
gr.neighbors('A')
g1=generate(10,20);g2=generate(10,20);g1==g2
go=digraph(); go.DIRECTED; isinstance(go,digraph)
goo=generate(10,20,True)
```

2 Liste des méthodes

Création/suppresion dans un graphe

- `graph()` : Renvoie un graphe non-orienté vide
- `g.add_node(node)` : Ajoute le sommet `node` au graphe `g`, où `node` peut être de type `string` ou `int`.
- `g.add_nodes(nodelist)` : Ajoute tous les sommets de la liste `nodelist` au graphe `g`.
- `g.add_edge(edge)` : Ajoute l'arête `edge` au graphe `g`, où `edge` est un couple de sommets.
- `g.add_edges(edgelist)` : Ajoute toutes les arêtes de la liste `edgelist` au graphe `g`.
- `g.add_graph(other)` : Ajoute le graphe `other` au grapheg.
- `g.del_node(node)` : Supprime le sommet `node` du graphe `g`.
- `g.del_edge(edge)` : Supprime l'arête `edge` du graphe `g`.

Composantes d'un graphe

- `g.nodes()` : Renvoie la liste de tous les sommets de `g`.
- `g.edges()` : Renvoie la liste de toutes les arêtes de `g`.
- `g.has_node(node)` : Teste si `node` est un sommet de `g`.
- `g.has_edge(edge)` : Teste si `edge` est une arête de `g`.
- `g.neighbors(node)` : Renvoie la liste des sommets accessibles depuis le sommet `node` dans le graphe `g`.

Graphes orientés

- `digraph()` : Renvoie un graphe orienté vide
- `g.DIRECTED` : Renvoie un booléen indiquant si le graphe est orienté.
- `g.incidents(node)` : Renvoie la liste des sommets accessibles depuis le sommet `node` dans le graphe `g`.
- `g.reverse()` : Renvoie le graphe avec les mêmes sommets que `g` et les arêtes inversées.

Divers

- `g1==g2` : Teste si `g1` et `g2` sont égaux.
- `print(g)` : Imprime la liste des sommets et la liste des arêtes de `g`.
- `generate(num_nodes,num_edges)` : Renvoie un graphe aléatoire à `num_nodes` sommets et `num_edges` arêtes
- `generate(num_nodes,num_edges,True)` : Renvoie un graphe orienté aléatoire avec `num_nodes` sommets et `num_edges` arêtes
- `isinstance(g,classe)` : Teste si l'objet `g` est un élément de la classe `classe`.