Informatique – TP22

Vésale Nicolas - Henrik Thys

Exercice 1: Manipulations des files.

Dans chacun des cas suivants, donner la valeur de `f` après exécution et donnez la complexité des opérations effectuées.

1.
```
#utilisation d'une liste
f=[1,2,3,4]
f.append(5)
f=[0]+f
f.pop()
f=f[1:]
```

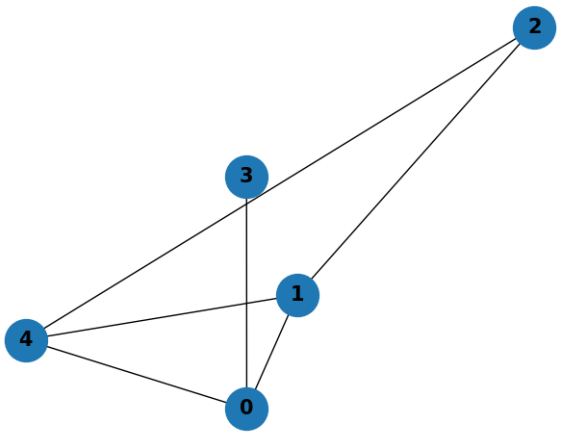
2.
```
from collections import deque
f=deque([1,2,3,4])
f.append(5)
f.appendleft(0)
f.pop()
f.popleft()
```

Exercice 2: Distance entre deux sommets.

On se donne un graphe non orienté `g` et deux sommets `Si` et `Sb`.
Donner une fonction `distance(g,Si,Sb)` qui rend la distance entre `Si` et `Sb` c'est-à-dire le plus petit nombre d'arêtes qui compose un chemin de `Si` à `Sb` si il en existe un et `False` sinon.
On pourra utiliser la liste des « pères » vue au TP précédent. Vous pouvez utiliser votre cours si vous en avez besoin, mais c'est mieux si vous réussissez à vous en passer.

Exercice 3: Un exemple d'utilité du parcours en profondeur.

Étant donné un graphe non orienté `g`, un cycle de `g` est une suite d'arêtes deux à deux distinctes qui relient un sommet à lui-même. Par exemple dans le graphe:



il existe un cycle partant de tous les sommets sauf 3.
On peut utiliser un parcours en profondeur pour détecter efficacement la présence d'un cycle dans `g`. En effet, si à un moment de son exécution, il pointe vers un sommet déjà visité qui n'est pas le sommet père, cela signifie que nous sommes en présence d'un cycle.
En déduire une fonction `cycle(g)` qui rend `True` si la graphe `g` possède un cycle et `False` sinon. On supposera pour simplifier que `g` n'a qu'une seule composante connexe.