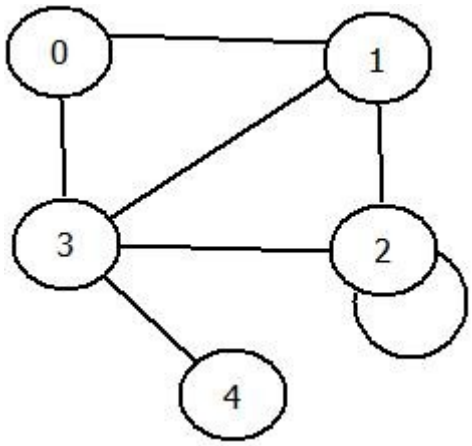
Michaël Guedj

TP d'Algorithmique Avancée

Algorithme de parcours en largeur de graphe
Breadth First Search (BFS)

Soit le graphe *G* (implémenté dans un TP précédant).



On rappelle l'algorithme de parcours en largeur de graphe.

```
FONCTION BFS ( G=(S, succ), s0 ) :
done = [ s0 ]
todo = File_Vide
todo.enfiler( s0 )
TANT QUE todo n'est pas vide :
___ s = todo.defiler()
___ POUR s' ∈ succ(s) :
___ ___ SI s' ∉ done :
___ ___ ___ todo.enfiler(s')
___ ___ ___ done = done + [ s' ]
___ ___ FIN SI
___ FIN POUR
FIN TANT QUE
RETOURNER done
```

Exercice 1

Implémenter le type abstrait File.

En informatique, une **file** dite aussi *file d'attente* (en anglais *queue*) est une structure de données basée sur le principe du premier entré, premier sorti ou PEPS, désigné en anglais par l'acronyme *FIFO* (*first in, first out*) : les premiers éléments ajoutés à la file seront les premiers à en être retirés.

Les primitives communément utilisées pour manipuler des files sont :

- « Enfiler » : ajoute un élément dans la file. Le terme anglais correspondant est *enqueue*.
- « Défiler » : renvoie le prochain élément de la file, et le retire de la file. Le terme anglais correspondant est *dequeue*.
- « La file est-elle vide ? » : renvoie *Vrai* si la file est vide, *Faux* sinon.
- « Nombre d'éléments dans la file » : renvoie le nombre d'éléments dans la file.

Exercice 2

Implémenter l'algorithme de parcours en largeur de graphe ; puis l'exécuter sur le graphe *G*.