Algorithmique et Programmation II

Structure de données : Listes chaînées, Files, Piles

Une ***liste chaînée*** est une suite d'éléments, appelés *nœuds*, chacun composé :

- ❒ de l'information que l'on veut traiter,
- ❒ d'un lien de chaînage qui est un pointeur vers l'élément suivant de la liste.

Chaque nœud pointe vers son successeur, sauf le dernier, qui contient le pointeur `NULL`. L'exploitation d'une liste nécessite un pointeur spécial, appelé **tête**, qui pointe vers le premier élément de la liste (cf. Figure 1).

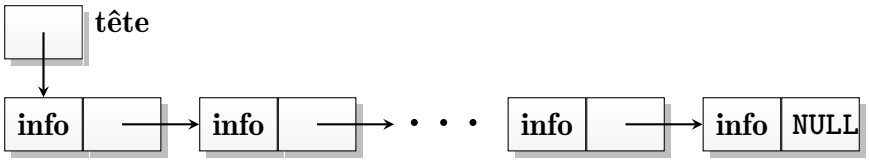

Figure 1 – Liste chaînée

Les ***piles*** et les ***files*** sont des cas particuliers de liste chaînée :

- ❒ **pile** (*stack*) : structure de données dans laquelle on peut ajouter (i.e. *empiler/push*) et supprimer (i.e. *dépiler/pop*) des éléments suivant la règle du dernier arrivé premier sorti LIFO (Last In First Out).
- ❒ **file** (*queue*) : structures de données dans laquelle on peut ajouter (i.e. *enfiler/enqueue*) et supprimer (i.e. *défiler/dequeue*) des éléments suivant la règle du premier arrivé premier sorti FIFO (First In First Out).

Exercice 1 : (**Listes chaînées/File**) Considérons le graphe orienté G illustré dans la figure ci-dessous :

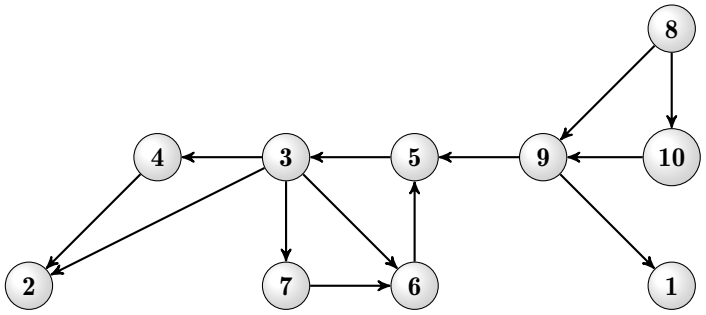

Figure 2 – Graphe $G = (V, E)$

1. Proposer et implémenter une représentation du graphe G, sachant que cet exercice s'intéresse au parcours en largeur du graphe G. Identifier les avantages et les inconvénients de la représentation choisie.
2. Implémenter et appliquer le parcours en largeur (cf. Algorithme 1) au graphe G en prenant pour sommet de départ le sommet 3, ensuite en prenant pour sommet de départ le sommet 8. Afficher la liste des sommets visités dans l'ordre dans lequel ils sont visités.

3. Quelle est la complexité de l'algorithme 1 dans le pire cas ? Argumenter votre réponse.

Algorithm 1 : Parcours en largeur d'un graphe G à partir d'un sommet source s

```
1: f = creerFile();
2: f.enfiler(s);
3: marquer(s);
4: while f.vide() = FAUX  do
5:   s = f.defiler();
6:   for all voisins t de s dans G do
7:     if t non marqué then
8:       f.enfiler(t);
9:       marquer(t);
10:    end if
11:  end for
12: end while
```

Exercice 2 : (**Récursivité**) En vous inspirant du pseudo-code de l'Algorithme 2, implémenter et appliquer le parcours en profondeur au graphe de la Figure 2 en prenant pour sommet de départ le sommet 5, et respectivement, le sommet 7. Afficher la liste des sommets visités dans l'ordre dans lequel ils sont visités.

Algorithm 2 : Parcours en profondeur d'un graphe G à partir d'un sommet source s

```
explorer(graphe G, sommet s)
1: marquer(s);
2: for all voisins t de s dans G do
3:   if t non marqué then
4:     explorer(G, t);
5:   end if
6: end for
```