Interrogation 4 du Lundi 8 Décembre

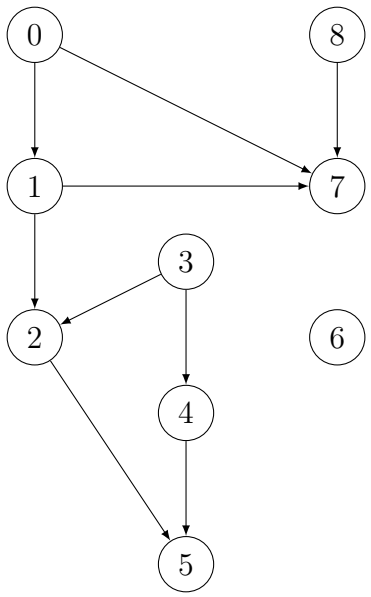
Dans l'ensemble du devoir, toutes les fonctions seront :

- *écrites en langage OCaml,*
- *précédées d'une explication des variables utilisées,*
- *précédées d'une explication de l'algorithme.*

Exercice 1 (Tri topologique d'un graphe orienté)

Un **tri topologique** d'un graphe orienté acyclique $G = (S, A)$ est un ordre linéaire des sommets de G tel que, pour tous sommets u et v appartenant à S, si l'arc (u, v) appartient à A, alors u apparait avant v.

1. Proposer plusieurs tris topologiques du graphe orienté acyclique G suivant :



2. Montrer que, si $G = (S, A)$ est un graphe orienté qui admet un tri topologique, alors G est sans cycle.
3. Soit $G = (S, A)$ est un graphe orienté acyclique.
   (a) Montrer que G admet un sommet de degré entrant 0. On pourra considérer un chemin dans G de longueur maximal.
   (b) En déduire que G admet un tri topologique.

On représente les graphes en OCaml par liste d'adjacence et on les implémente par le type :

```
type graphe = int list array ;;
```

4. Définir le graphe G de la question 1 sur OCaml.
5. (a) Écrire en OCaml une fonction `suppression x l` de type `int -> int list -> int list` qui permet de supprimer un élément x d'une liste l.
   (b) Écrire en OCaml une fonction `supprime_arc g a b` de type `graphe -> int -> int -> unit` qui permet de supprimer un arc (a, b) d'un graphe G.

(c) Écrire en OCaml une fonction `degre g a` de type `graphe -> int -> int` qui permet de calculer le degré entrant d'un sommet a d'un graphe G.

(d) Écrire en OCaml une fonction `tri_topologique g` de type `graphe -> int list` qui renvoie un tri topologique des sommets du graphe G basé sur le fait qu'un sommet de degré entrant nul peut être placé en tête d'un tri topologique.

Exercice 2 (Diamètre d'un graphe)

Dans cet exercice, on considère des graphes non orientés connexes. Les sommets d'un graphe à n sommets ($n \in \mathbb{N}^*$) sont numérotés de 0 à $n-1$. On suppose qu'aucune arête ne boucle sur un même sommet.

Un **chemin de longueur** $p \in \mathbb{N}$ d'un sommet a vers un sommet b dans un graphe est la donnée de $p+1$ sommets $s_0, s_1, \ldots, s_p$ tels que $s_0 = a$, $s_p = b$ et, pour tout $1 \leqslant k \leqslant p$, les sommets s_{k-1} et s_k sont reliés par une arête.

Un **plus court chemin** d'un sommet a vers un sommet b dans un graphe G est un chemin de longueur minimale parmi tous les chemins de a vers b. Sa longueur est notée $d_G(a,b)$.

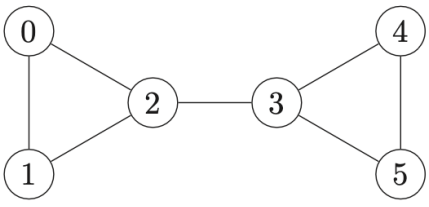
Le **diamètre** d'un graphe G, noté $\mathrm{diam}(G)$, vaut le maximum des longueurs des plus courts chemins entre deux sommets du graphe G. Autrement dit,

$$\mathrm{diam}(G) = \max_{a,b \text{ sommets de } G} d_G(a,b).$$

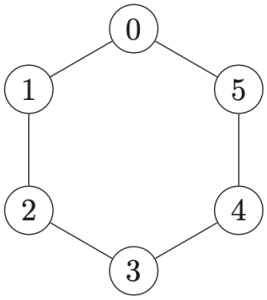
Un **chemin maximal** d'un graphe G est un plus court chemin de G de longueur $\mathrm{diam}(G)$.

Partie 1 : Exemples de graphes

1. Donner sans justification le diamètre et les chemins maximaux pour chacun des deux graphes G_1 et G_2 ci-dessous.



Graphe G_1



Graphe G_2

En OCaml, les graphes sont représentés par liste d'adjacence et implémentés par le type

```
type graphe = int list array ;;
```

2. Graphes de diamètre maximal.

(a) Dessiner sans justification un graphe à 5 sommets ayant un diamètre le plus grand possible.

(b) Écrire en OCaml une fonction `diam_max` de type `int -> graphe` qui prend en argument un entier naturel n non nul et qui renvoie un graphe à n sommets de diamètre maximal.

3. Graphes de diamètre minimal.
   (a) Dessiner sans justification un graphe à 5 sommets ayant un diamètre le plus petit possible.
   (b) Écrire en OCaml une fonction `diam_min` de type `int -> graphe` qui prend en argument un entier naturel n non nul et qui renvoie un graphe à n sommets de diamètre minimal.

Partie 2 : Algorithmes de calcul du diamètre

Dans cette partie, on suppose que les graphes sont représentés par listes d'adjacence.

4. Donner l'entrée et la sortie de l'algorithme de Dijkstra. Comment cet algorithme permet-il de calculer le diamètre d'un graphe ?
5. Quel parcours de graphe peut être utilisé pour le calcul du diamètre ?
6. Laquelle des deux méthodes précédentes est la mieux adaptée pour calculer le diamètre d'un graphe ?

Partie 3 : Diamètre d'un arbre binaire

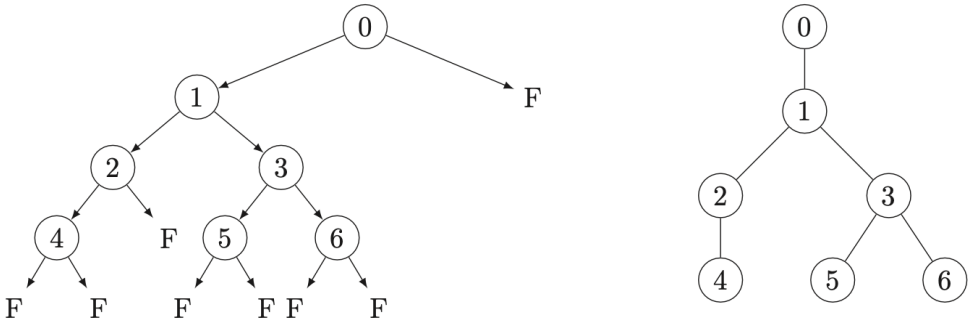
Dans cette partie, on s'intéresse aux arbres binaires, qui sont des cas particuliers de graphes. On travaille avec une représentation spécifique de ces graphes particuliers, implémentée en OCaml par le type suivant :

```
type arbre = Feuille | Noeud of int * arbre * arbre ;;
```

Le **graphe sous-jacent** G_A à un arbre binaire A est défini comme le graphe orienté dont :
- les sommets correspondent aux noeuds de l'arbre (et pas aux feuilles) ;
- les arêtes correspondent aux branches de l'arbre reliant deux noeuds (et non pas celles reliant un noeud à une feuille).

Le diamètre d'un arbre binaire est alors défini comme le diamètre de son graphe sous-jacent.

Voici un exemple d'arbre binaire A (à gauche) et de son graphe sous-jacent G_A (à droite) :



7. Donner l'expression OCaml représentant l'arbre A de l'exemple. Donner le diamètre de A et les chemins maximaux du graphe sous-jacent G_A.
8. Quel est le nombre r d'arêtes du graphe sous-jacent à un arbre binaire possédant n noeuds ?

Une première approche pour calculer le diamètre d'un arbre consiste à le transformer en un graphe et à employer un algorithme général sur les graphes de la partie 2.

9. Écrire en OCaml une fonction `nb_noeuds` de type `arbre -> int` qui renvoie le nombre de nœuds d'un arbre binaire donné en argument.
10. Écrire en OCaml une fonction `numerotation` de type `arbre -> arbre` qui prend en argument un arbre binaire A à n nœuds et qui renvoie un arbre binaire A' de même graphe sous-jacent que A et dont les nœuds sont étiquetés de 0 à $n-1$.
11. Écrire en OCaml une fonction `arbre_vers_graphe` de type `arbre -> graphe` qui prend en argument un arbre binaire A à n nœuds étiquetés de 0 à $n-1$ et qui renvoie le graphe G_A sous-jacent à A (le type `graphe` est défini dans la partie 1).
12. Décrire un algorithme qui calcule le diamètre d'un arbre de type `arbre` en se ramenant à un graphe. Quelle est sa complexité ?

Une seconde approche pour calculer le diamètre d'un arbre consiste à employer une technique diviser-pour-régner. Pour tout arbre A non réduit à une feuille, de la forme `Noeud (x, arbre_g, arbre_d)`, on note :

— A_g le fils gauche de A, représenté par `arbre_g` ;
— A_d le fils droit de A, représenté par `arbre_d`.

La hauteur de l'arbre A, notée $h(A)$, est la longueur du plus long chemin descendant de la racine vers une feuille. Dans l'arbre exemple de la partie 3, l'arbre A est de hauteur 4.

13. Quelle est la longueur d'un chemin maximal passant par la racine ?
14. Écrire en OCaml une fonction `diam_arbre` de type `arbre -> int` qui calcule le diamètre d'un arbre donné en argument. Cette fonction devra être de complexité linéaire en le nombre de nœuds de l'arbre.
