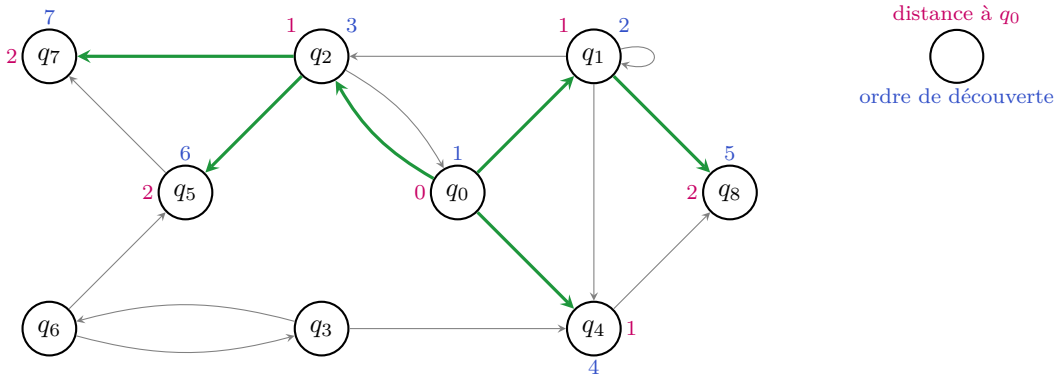
# AL5 – Algorithmique
# Interrogation n° 1 – 9 octobre 2020 (correction)

### Exercice 1 : parcours en largeur

Appliquer l'algorithme du parcours en largeur à partir du sommet $q_0$ sur le graphe ci-dessous. On supposera que les sommets seront toujours examinés par ordre croissant d'étiquette lorsqu'un choix est à faire. Indiquer sur la figure l'ordre de découverte des sommets et l'arborescence du parcours. Donner le contenu final des tables Π et Dist.




états successifs de la file : `[0]`, `[1,2,4]`, `[2,4,8]`, `[4,8,5,7]`, `[8,5,7]`, `[5,7]`, `[7]`, `[]`,
ordre de parcours : $q_0$, $q_1$, $q_2$, $q_4$, $q_8$, $q_5$, $q_7$,
tableau des pères : Π = `[None,0,0,None,0,2,None,2,1]`,
tableau des distances à $q_0$ : `Dist = [0,1,1,None,1,2,None,2,2]`.

### Exercice 2 : parcours en profondeur

**1.** Appliquer maintenant l'algorithme du parcours en profondeur sur le graphe ci-dessous, en supposant toujours que les sommets seront examinés par ordre croissant d'étiquette lorsqu'un choix est à faire. Indiquer sur la figure les dates `d` et `f` obtenues pour chaque sommet et le type des arcs non retenus dans la forêt de parcours (arc avant, arc retour, arc transverse). Indiquer également le contenu final de la table Π.

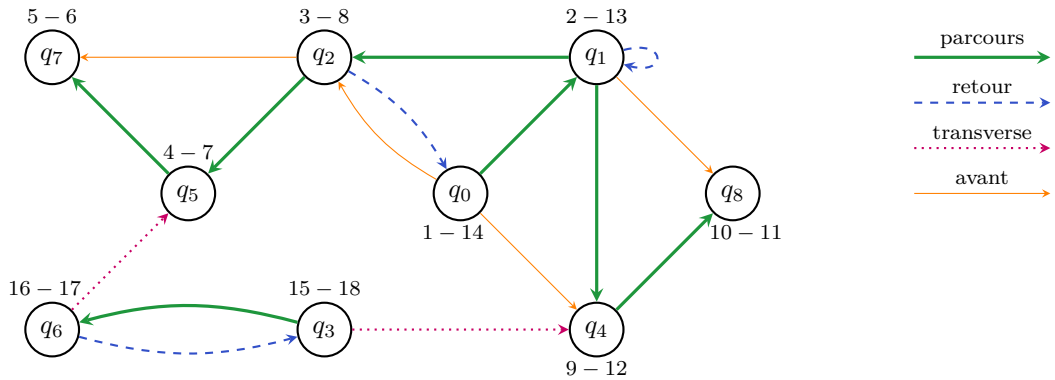



tableau des pères : Π = `[None,0,1,None,1,2,3,5,4]`

**2.** Dans cet exercice, on veut dessiner un graphe orienté pour lequel un parcours en profondeur peut renvoyer les dates `d` et `f` données ci-dessous (il manque des dates, à vous de les compléter !). On impose les deux contraintes suivantes : au moins un sommet peut accéder à tous les autres sommets du graphe, et le nombre de composantes fortement connexes est minimal.

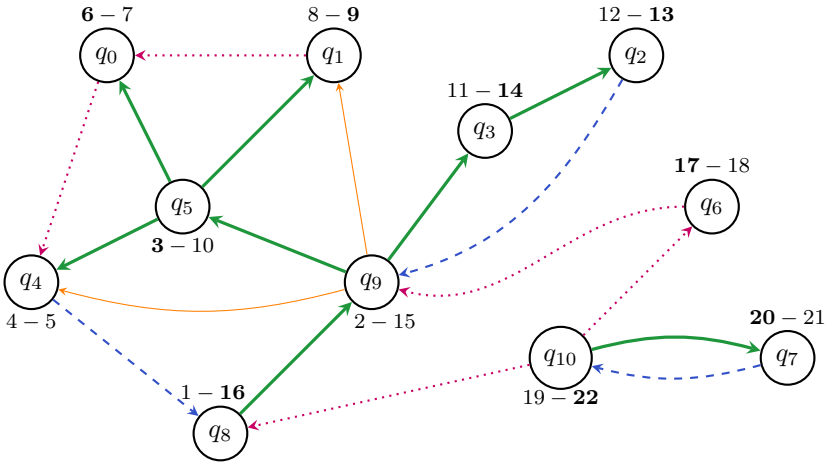
*Il y a essentiellement 2 solutions pour la forêt du parcours, avec respectivement 1 et 3 arbres. À ce stade, le nombre de CFC est 11 (chaque sommet est une CFC), et la contrainte d'accessibilité est satisfaite seulement pour la forêt qui n'est formée que d'un arbre. Il faut donc rajouter des arcs pour satisfaire les deux conditions supplémentaires, en faisant bien attention de ne pas contredire la manière dont le parcours est censé se dérouler : aucun arc ne doit aller « plus profond » que le parcours.*

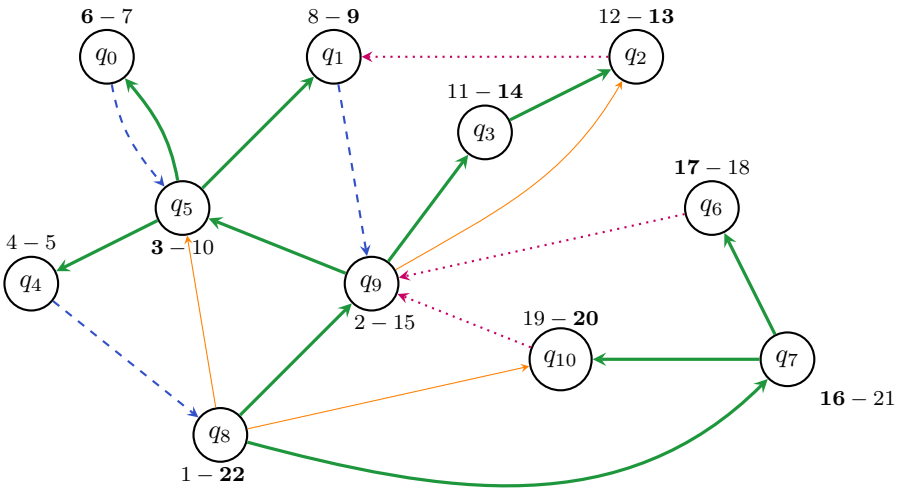
*Pour satisfaire la contrainte d'accessibilité, dans le cas où la forêt de parcours est un arbre, il n'y a rien à faire, tous les sommets sont accessibles depuis $q_8$. Pour l'autre forêt, le sommet qui peut accéder à tous les autres doit être dans le dernier arbre visité, sinon on a une contradiction. Il faut donc ajouter un arc transverse depuis l'arbre $\{q_7, q_{10}\}$ vers $\{q_6\}$, et de l'un de ces trois sommets vers le « gros » arbre. À noter que l'extrémité de ce 2e arc transverse n'a pas d'importance, nous allons transformer le « gros » arbre en CFC.*

*Pour minimiser le nombre de composantes fortement connexes, il faut transformer chaque arbre en une CFC. C'est toujours possible en ajoutant des arcs, par exemple un arc retour de chaque feuille vers la racine, ou un arc retour à rebours de chaque arc de parcours, mais il y a plein d'autres possibilités ; il faut au minimum un arc retour vers la racine de l'arbre, un arc retour depuis la première feuille, et un arc (retour ou transverse) depuis chaque autre feuille.*

*On peut ensuite ajouter d'autres arcs « pour le plaisir », en particulier des arcs avant.*

*Deux exemples avec respectivement 3 et 1 CFC :*



### Exercice 3 : autour des parcours

Étant donné un graphe $G = (S, A)$, et un sommet $x \in S$, on appelle *excentricité* de $x$ dans $G$ la distance maximale entre $x$ et un (autre) sommet de $G$ : `exc`$(x) = \max\{\mathsf{dist}(x, y) \mid y \in S\}$. Décrire un algorithme `excentricite(x, G)` qui calcule l'excentricité de $x$ dans $G$. Justifier la réponse et donner la complexité de l'algorithme.

*Je rappelle que la distance entre deux sommets est définie comme la plus petite longueur d'un chemin entre ces deux sommets (s'il en existe), et infinie sinon. Cette notion est toujours bien définie : même s'il existe une infinité de chemins entre les deux sommets considérés, l'ensemble des chemins simples (qui ne passent pas deux fois par un même sommet) est fini, et le chemin le plus court en fait partie.*

*Pour calculer l'excentricité de `x`, donc le max de l'ensemble (fini) de toutes les distances depuis `x`, il suffit d'adapter un parcours en largeur de `G` à partir de `x`, puisque ce parcours calcule toutes ces distances. Par ailleurs, les sommets sont visités par distance croissante depuis `x`, donc si tous les sommets sont visités, le dernier sommet visité `last` est le (ou plutôt l'un des) plus éloigné(s) de `x` et l'excentrictié de `x` est `Dist[last]`. Sinon l'excentricité de `x` est infinie.*
*La complexité de cet algorithme est celle du parcours en largeur, $\Theta(|A| + |S|)$.*

*Attention, certains ont voulu utiliser un parcours en profondeur. Cela ne peut pas fonctionner, ce parcours ne donnant aucun indication sur les distances – bien au contraire, puisqu'il essaie toujours d'aller « plus loin », il a tendance à emprunter de longs chemins. Pensez par exemple aux parcours d'un graphe complet : l'arbre du parcours en largeur sera une étoile (tous les sommets sont à distance 1 de la racine), alors que l'arbre du parcours en profondeur sera filiforme.*

*Plusieurs d'entre vous ont (mal) compris la définition de l'excentricité de `x` et cru qu'il fallait chercher la longueur d'un chemin le plus long possible depuis `x`. Cette notion aurait été mal définie : un tel chemin n'existe que si le graphe n'a pas de cycle (en tout cas pas de cycle atteignable depuis `x`).*

### Exercice 4 : autour des parcours

Décrire un algorithme qui teste si un graphe orienté $G = (S, A)$ est un DAG (*directed acyclic graph*, *i.e.* graphe orienté sans circuit). Justifier la réponse et donner la complexité de l'algorithme.

*La présence de circuit est équivalente à celle d'arcs retour dans un parcours en profondeur quelconque. Il suffit donc d'adapter un parcours en profondeur de `G`. Un arc retour est caractérisé par le fait que son extrémité cible est (déjà) sur le chemin gris, et peut donc être repéré pendant le parcours en profondeur ; si c'est le cas, on sort de manière anticipée du parcours en retournant `False`. Si le parcours arrive à son terme, on retourne `True`.*

*Plusieurs d'entre vous ont voulu utiliser la caractérisation des arcs retour par les dates de début et de fin des deux extrémités. C'est possible, mais attention cependant : lors du (premier) parcours, les dates de fin des deux sommets considérés ne sont pas encore connues au moment où l'arc est découvert ; il faut donc terminer le parcours, puis en faire un 2e– pas forcément en profondeur, mais il faut considérer un à un tous les arcs pour tester la caractérisation des arcs retour. Je dis bien tous les arcs, pas tous les couples de sommets, sinon la complexité en prend un coup.*

*Troisième option, utiliser l'algorithme de calcul des CFC : pour un DAG, les CFC sont chacune réduites à un unique sommet.*

*La complexité de ces trois versions est (en ordre de grandeur) la même que celle du parcours en profondeur classique : $\Theta(|A| + |S|)$.*