Écrivez ici votre numéro de table : ☐

Université de Caen Normandie — Année 2022-2023
U.F.R. des Sciences — Examen terminal – Session 1

L3 Informatique

SMINFL6E — Théorie des graphes

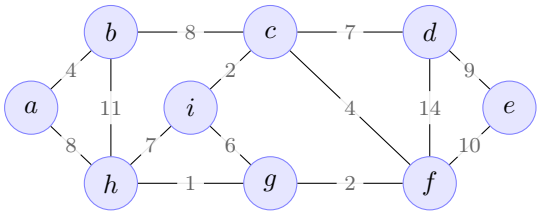
jeudi 6 avril 2022 – 9h30

Durée : 2h — Tous documents autorisés — Calculatrices autorisées — Le barème est indicatif

Les réponses sont à écrire sur ce sujet (inutile de rendre une copie vide avec). Indiquez votre numéro de table dans le cadre en haut à droite.

1 Arbres couvrants de poids minimal (3 points)

Question 1.1. Appliquer l'algorithme de Kruskal au graphe ci-dessous, en coloriant ou entourant les arêtes sélectionnées et en faisant une croix sur les arêtes rejetées.



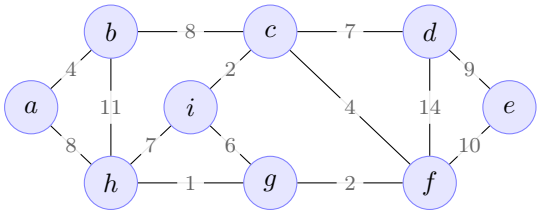
Indiquer ci-dessous l'ordre de sélection des arêtes ainsi que le poids de l'arbre couvrant obtenu.

On parcourt les arêtes dans l'ordre croissant de leurs poids. On rejette si l'arête crée un cycle avec les arêtes déjà choisies.

1. (g, h), poids 1
2. (c, i) ou (f, g), poids 2
3. (f, g) ou (c, i), poids 2
4. (a, b) ou (c, f), poids 4
5. (c, f) ou (a, b), poids 4
   [rejet de (g, i)]
6. choix de (c, d), poids 7
   [rejet de (h, i)]
7. choix de (a, h) ou (b, c), poids 8
   [rejet de (b, c) ou (a, h), selon le choix précédent]
8. (d, e), poids 9
   [rejet de (e, f), (b, h) et (d, f)]

Poids : 37.

Question 1.2. Appliquer l'algorithme de Prim au graphe ci-dessous, en coloriant ou entourant les arêtes sélectionnées.



Indiquer ci-dessous l'ordre de sélection des arêtes ainsi que le poids de l'arbre couvrant obtenu.

Départ du sommet e (précisé pendant l'exam). On choisit l'arête de poids minimal parmi les arêtes non choisies qui sont incidentes aux sommets « découverts ». J'ai accepté les réponses qui donnaient juste l'ordre de découverte des sommets, ce qui est plus proche de l'algo tel qu'on l'a vu.

1. (d, e), poids 9 ; découverte du sommet d
2. (c, d), poids 7 ; découverte du sommet c
3. (c, i), poids 2 ; découverte du sommet i
4. (c, f), poids 4 ; découverte du sommet f
5. (f, g), poids 2 ; découverte du sommet g
6. (g, h), poids 1 ; découverte du sommet h
7. (a, h), poids 8 ; découverte du sommet a
   OU (b, c), poids 8 ; découverte du sommet b
8. (a, b), poids 4 ; découverte du sommet b ou a, selon le choix précédent

Poids : 37.

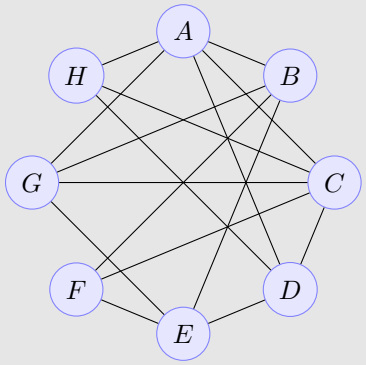
2 Coloration des sommets d'un graphe (3 points)

Alice souhaite acheter huit poissons, de huit espèces différentes, qu'on nommera A, B, C, D, E, F, G et H. Certaines espèces ne peuvent cohabiter ensemble dans un même aquarium, en raison de la température de l'eau, du degré de salinité, de la présence de certaines algues, etc. Le tableau ci-dessous indique les incompatibilités entre espèces. Alice se demande maintenant le nombre minimum d'aquariums (noté *nbopt*) qu'elle va devoir acheter.

	A	B	C	D	E	F	G	H
A		×	×	×			×	×
B	×				×	×	×	
C	×			×		×	×	×
D	×		×		×			×
E		×		×		×	×	
F		×	×		×			
G	×	×	×		×			
H	×		×	×				

Question 2.1. Modéliser cette question sous la forme d'un problème de coloration des sommets d'un graphe. Dessiner le graphe en question et indiquer à quoi correspondent les sommets, les arêtes, les couleurs, et *nbopt*.

La question revient à chercher le nombre minimal de couleurs permettant de colorier le graphe ci-dessous, dont les sommets sont les espèces et les arêtes sont les **incompatibilités** entre espèces. Les couleurs correspondent à des aquariums : on a la garantie que l'ensemble des espèces ayant la même couleur ne partagent pas d'arête, et sont donc toutes compatibles. Le nombre minimal d'aquariums nécessaires *nbopt* est donc le nombre chromatique du graphe.



NB : on ne peut pas modéliser la question comme un problème de coloration sur le graphe des compatibilités : les couleurs ne représentent rien d'utile sur ce graphe !

Question 2.2. Appliquer l'algorithme de Welsh-Powell au graphe, en indiquant les différentes étapes.

On classe les sommets dans l'ordre décroissant de leurs degrés, puis on choisit une couleur C et on parcourt les sommets. Pour chacun, s'il n'est pas encore coloré et n'est pas adjacent à un sommet de couleur C, on lui attribue la couleur C. On recommence le parcours avec une autre couleur, tant qu'il reste un sommet non coloré.

sommet	1er parcours	2e parcours	3e parcours	4e parcours
A (5)	bleu	-	-	-
C (5)	*voisin de A*	rouge	-	-
B (4)	*voisin de A*	rouge	-	-
D (4)	*voisin de A*	*voisin de C*	jaune	-
E (4)	bleu	-	-	-
G (4)	*voisin de A*	*voisin de C*	jaune	-
F (3)	*voisin de E*	*voisin de C*	jaune	-
H (3)	*voisin de A*	*voisin de C*	*voisin de D*	vert

NB : pour les sommets de même degré, l'ordre à choisir n'était pas précisé. Ci-dessus j'ai pris l'ordre alphabétique, mais j'ai bien sûr validé les réponses qui avaient fait un autre choix (à condition qu'elles soient correctes).

Question 2.3. En analysant la structure du graphe, proposer un minorant de *nbopt*. (Justifier.)

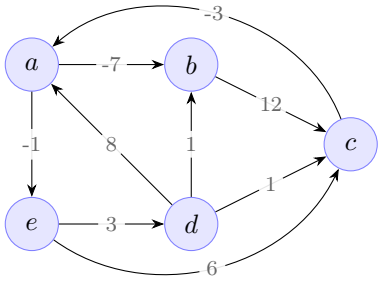
Le sous-graphe induit par $\{A, C, D, H\}$ est complet, il nécessite donc 4 couleurs. Le graphe entier ne peut donc pas être coloré avec moins de 4 couleurs : $nbopt \geq 4$.

Question 2.4. Déduire des deux questions précédentes la valeur de *nbopt*. (Justifier.)

L'algorithme de Welsh-Powell ne donne pas forcément une coloration optimale. Cependant, comme il a trouvé une 4-coloration, on en déduit que $nbopt \leq 4$ (si *nbopt* était plus grand, l'algo n'aurait pas pu trouver une 4-coloration). Avec la réponse à la question précédente, on a $4 \leq nbopt \leq 4$, donc $nbopt = 4$.

3 Plus courts chemins (4 points)

On considère le graphe orienté G ci-contre.
On souhaite déterminer les plus courts chemins entre le sommet e et tous les autres sommets.



Question 3.1. Peut-on appliquer l'algorithme de Dijkstra ? (Justifier.)

L'algorithme de Dijkstra ne fonctionne que si tous les poids sont positifs ou nuls, il n'est donc pas applicable sur ce graphe qui comporte des poids négatifs.

Question 3.2. Si on applique l'algorithme de Bellman-Ford, donnera-t-il un résultat correct ? (Justifier.)

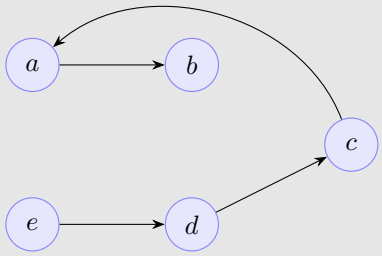
L'algorithme de Bellman-Ford est applicable aux graphes comportant des poids négatifs. S'il existe un cycle absorbant dans le graphe, la notion de plus court chemin n'a pas de sens, et le tableau construit par l'algo n'est pas correct (mais l'algorithme le détecte et le signale). Le graphe qui nous intéresse ne contient pas de cycle absorbant, donc Bellman-Ford donnera un résultat correct.

Question 3.3. On applique l'algorithme de Bellman-Ford au graphe G. Les arcs doivent être parcourus dans l'ordre lexicographique ((a, b) et (a, e), puis (b, c), puis (c, a), puis (d, a), (d, b) et (d, c), et enfin (e, c) et (e, d)).

Le tableau ci-dessous montre l'état de l'algorithme à la fin de la première itération. Terminer l'exécution en complétant le tableau ; dans chaque itération, on fera une ligne pour chaque arc qui modifie l'état de dist ou pred et on n'écrira que les valeurs qui changent (comme c'est fait pour la première itération). Les itérations qui ne change pas l'état doivent apparaître (mettre le numéro de l'itération, mais laisser la ligne vide).

itér.	arc	dist[a]	dist[b]	dist[c]	dist[d]	dist[e]	pred[a]	pred[b]	pred[c]	pred[d]	pred[e]
0		∞	∞	∞	∞	0	nil	nil	nil	nil	nil
1	(e,c)			6					e		
1	(e,d)				3					e	
2	(c,a)	3					c				
2	(d,b)		4					d			
2	(d,c)			4					d		
3	$(a,b$		-4					a			
3	(c,a)	1					c				
4	(a,b)		-6					a			
5											

Question 3.4. Représenter le graphe correspondant au tableau pred. Est-ce un arbre ? (Justifier.)



Il s'agit d'un arbre orienté, puisque son graphe non orienté sous-jacent est connexe et acyclique. (C'est même un arbre orienté enraciné en a, puisqu'il existe un unique chemin orienté de a à n'importe quel sommet.)

Question 3.5. Est-il possible de modifier l'algorithme de Bellman-Ford pour qu'il sorte un tableau de distances correct quel que soit le graphe d'entrée ?

Non, car si le graphe contient un cycle absorbant, la notion de plus court chemin n'a pas de sens, et donc la distance n'est pas définie. (J'ai accepté les réponses qui proposaient de mettre $-\infty$ comme distance de a à s s'il y a un chemin de a à s qui passe par un cycle absorbant, même si formellement c'est une extension de notre définition de distance.)

4 Plan d'évacuation (5 points)

Les parties de cet exercice sont indépendantes.

On s'intéresse à des plans d'évacuation de bâtiments, représentés par ce qu'on appellera des *graphes d'évacuation*. Ce sont des graphes simples non orientés pondérés où les sommets sont des pièces, les arêtes correspondent à des passages entre les pièces (portes, escaliers, couloirs…), et le poids de chaque arête représente le nombre de personnes pouvant emprunter le passage correspondant en une seconde.

4.1 Capacité d'évacuation

On définit la *capacité d'évacuation* d'une pièce comme la somme des poids des arêtes incidentes au sommet correspondant. En cas de sinistre, une pièce est d'autant plus dangereuse que sa capacité d'évacuation est basse.

Question 4.1. Écrire le pseudo-code d'une fonction capacité qui prend en paramètre un graphe d'évacuation G et un sommet s, et qui renvoie la capacité d'évacuation de s.

L'algorithme le plus simple et efficace est le suivant, en $O(d(s))$:

```
fonction capacité(G,s):
    cap = 0
    pour chaque arête a incidente à s:
        cap += poids[a]
    renvoyer cap
```

J'ai accepté également des variantes, notamment celle-ci, en $O(|A|)$:

```
fonction capacité(G,s):
    cap = 0
    pour chaque arête a de G:
        si a est incidente à s:
          cap += poids[a]
    renvoyer cap
```

Question 4.2. Écrire le pseudo-code d'une fonction dangereuse qui prend en paramètre un graphe d'évacuation G et qui renvoie la pièce la plus dangereuse (ou n'importe laquelle des plus dangereuses s'il y a égalité).

Attention, il fallait renvoyer la *pièce* la plus dangereuse, pas sa capacité.

```
fonction dangereuse(G):
    mincap = +oo
    res = nil
    pour chaque sommet s de G:
        cap = capacité(G, s)
        si cap < mincap:
            mincap = cap
            res = s
    renvoyer res
```

Question 4.3. Quelle est la complexité au pire cas de votre fonction dangereuse ? Justifiez.

En utilisant la version efficace de capacité, qui est en $O(d(s))$, l'itération sur un sommet s est en $O(1) + O(d(s)) = O(d(s) + 1)$ (et pas seulement $O(d(s))$, car l'itération fait des opérations autres que l'appel de fonction). La boucle entière est donc en $O(\sum_{s\in S}(1 + d(s))) = O(|S| + \sum_{s\in S} d(s))$, or $\sum_{s\in S} d(s) = 2 \cdot |A|$ d'après la formule des poignées de main, donc la complexité est $O(|S| + |A|)$.

En utilisant la version moins efficace de capacité, chaque itération sur un sommet est en $O(|A|)$, donc la complexité est $O(|S| \times |A|)$.

4.2 Confirmation d'évacuation

À la fin d'une évacuation, les pompiers et secouristes font le tour du bâtiment pour vérifier qu'il ne reste personne. Il ne faut pas se contenter de regarder dans chaque pièce, mais aussi dans chaque passage entre pièces. Idéalement, pour que l'opération soit la plus efficace possible, il faut pouvoir parcourir tout le bâtiment en partant de l'entrée, emprunter *une seule fois* chaque passage, et revenir à l'entrée. Un graphe d'évacuation permettant un tel parcours est dit CEEC (pour « confirmation efficiente d'évacuation correcte »).

Question 4.4. Dessiner un graphe d'évacuation qui n'est pas CEEC.

Un graphe d'évacuation est CEEC si et seulement s'il est eulérien. Il suffit donc de donner un graphe non eulérien : un graphe non connexe par exemple (l'énoncé ne précisait pas que les graphes d'évacuation étaient forcément connexes), ou un graphe avec au moins un sommet de degré impair.

Question 4.5. Écrire le pseudo-code d'une fonction ceec qui prend en paramètre un graphe d'évacuation G et qui indique si le graphe est CEEC.

Il suffit de vérifier que le graphe est connexe et que tous les sommets sont de degré pair. Je détaille toutes les fonctions utiles ci-dessous, mais j'ai accepté les réponses qui n'ont pas vérifié la connexité et celles qui ont considéré qu'on disposait d'une fonction degré.

```
fonction parcours(G, s):
    marquer s en gris
    pour chaque voisin s' de s:
        si s' est blanc:
            parcours(G, s')
    marquer s' en noir

fonction connexe(G):
    si G n'a pas de sommet:
        renvoyer vrai
    sinon:
        choisir un sommet s de G
        parcours(G, s)
        pour chaque sommet s de G:
            si s est blanc:
                renvoyer faux
        renvoyer vrai

fonction ceec(G):
    si non connexe(G):
        renvoyer faux
    pour chaque sommet s de G:
        degré_pair = vrai
        pour chaque arête incidente à s:
            degré_pair = non degré_pair
        si non degré_pair:
            renvoyer faux
    renvoyer vrai
```

4.3 Accessibilité lors d'un incendie

Lors d'un incendie, les pompiers ont besoin de savoir si chaque pièce est toujours accessible depuis l'entrée du bâtiment. Un capteur situé dans chacun des passages indique si le passage en question est en feu (et donc non empruntable). On considère que l'ensemble des informations des capteurs nous est donnée dans un dictionnaire feu[ ], associant à chaque arête du graphe d'évacuation un booléen qui est à vrai si le passage correspondant est en feu (et donc non empruntable), et à faux sinon. (On ne sait pas quelles pièces sont en feu et on ne le prend pas en compte pour l'accessibilité.)

Question 4.6. Écrire une fonction inaccessibles qui prend en paramètre un graphe d'évacuation G, un sommet s représentant l'entrée du bâtiment, et un dictionnaire feu, et qui renvoie une liste (ou un ensemble) de toutes les pièces inaccessibles depuis l'entrée du bâtiment. Attention, votre fonction doit être linéaire en le nombre de pièces et de passages du plan.

Pour trouver les pièces accessibles, il suffit de lancer un algo de parcours (n'importe lequel – ci-dessous c'est un DFS) en partant de s et en ignorant les arêtes qui sont dans feu []. Toutes les pièces qui n'ont pas été visitées lors du parcours sont nécessairement inatteignables.

```
fonction parcours(G, s, feu):
    marquer s en gris
    pour chaque voisin s' de s:
        si s' est blanc et feu[s,s'] est faux:
            parcours(G, s', feu)
    marquer s en noir

fonction inaccessibles(G, s, feu):
    parcours(G, s, feu)
    res = ensemble vide
    pour chaque sommet s de G:
        si s est blanc:
            ajouter s à res
    renvoyer res
```

5 Tournois (5 points)

À part dans la partie 5.2, les questions de cet exercice sont toutes indépendantes.

On appelle *tournoi* un graphe orienté sans boucles tel que pour tout couple de sommets (s, s'), il y a soit un arc (s, s') soit un arc (s', s) (mais pas les deux). En d'autres termes, on obtient un tournoi en choisissant une direction pour chaque arête d'un graphe non orienté complet.

Question 5.1. Montrer qu'un tournoi est toujours faiblement connexe.

Il y avait manifestement beaucoup de confusions sur la faible et la forte connexité. Un graphe orienté est faiblement connexe si son graphe non orienté sous-jacent est connexe. Un graphe orienté est fortement connexe si pour tout couple de sommets (s, s'), il existe un chemin orienté de s à s'.

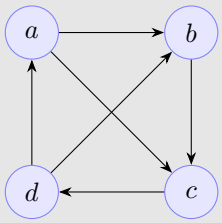
Les deux notions ne sont pas incompatibles, et encore moins l'inverse l'une de l'autre ! Au contraire, un graphe fortement connexe est nécessairement également faiblement connexe. Quand on dit qu'un graphe est faiblement connexe, ça n'exclut pas qu'il puisse aussi être fortement connexe.

Pour revenir à la question : par définition, le graphe non orienté sous-jacent d'un tournoi est complet, donc il est *a fortiori* connexe (tout sommet est accessible depuis n'importe quel autre, qui plus est par un chemin de longueur 1). Tout tournoi est donc faiblement connexe.

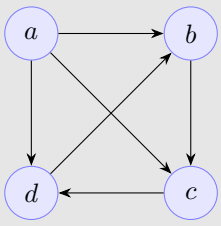
Question 5.2. Dessiner un tournoi d'ordre 4 qui est fortement connexe et un tournoi d'ordre 4 qui ne l'est pas.

Attention, il fallait donner des exemples de *tournois*, pas de n'importe quel graphe.

Le tournoi d'ordre 4 suivant est fortement connexe (grâce au cycle orienté (a, b, c, d, a)) :



Le tournoi d'ordre 4 suivant n'est pas fortement connexe (a n'a pas d'arc entrant, donc on ne peut l'atteindre depuis aucun autre sommet) :



5.1 Transitivité d'un tournoi

Un tournoi est dit *transitif* si pour tout couple d'arcs (a, b) et (b, c), il contient l'arc (a, c) (autrement dit, pour tout ensemble de trois sommets $\{a, b, c\}$, si le tournoi contient les arcs (a, b) et (b, c), alors il contient aussi l'arc (a, c)).

Question 5.3. Montrer que tout tournoi acyclique est transitif.

Soit T un tournoi acyclique. Considérons trois sommets distincts a, b et c tels que le tournoi contient les arcs (a, b) et (b, c). Puisque c'est un tournoi, il contient forcément soit l'arc (a, c) soit l'arc (c, a). Il ne peut pas contenir l'arc (c, a), puisqu'alors il contiendrait un cycle (a, b, c, a), ce qui contredit l'hypothèse d'acyclicité. Il contient donc nécessairement l'arc (a, c).

Cela est vrai pour tout ensemble de trois sommets vérifiant les conditions, le tournoi est donc transitif.

NB: si le tournoi ne contient aucun ensemble de trois sommets vérifiant les conditions (en particulier, s'il est d'ordre strictement inférieur à 3), alors il est transitif par vacuité. C'est implicite dans la preuve ci-dessus. Je n'ai pas exigé que ce soit précisé dans la réponse.

Question 5.4. Montrer que tout tournoi transitif est acyclique.

Soit T un tournoi transitif. Par l'absurde, supposons qu'il contient un cycle orienté et montrons qu'on arrive à une contradiction. Considérons un cycle orienté de T qui est de longueur minimale, et notons $k \in \mathbb{N}^*$ sa longueur.

- Si $k = 1$, alors cela signifie que T contient une boucle, ce qui est impossible par définition.
- Si $k = 2$, le cycle a la forme (s, s', s), ce qui signifie que T contient à la fois l'arc (s, s') et l'arc (s', s), ce qui est impossible par définition.
- Supposons que $k \geq 3$. Notons s_1, s_2 et s_3 les trois premiers sommets du cycle (qui sont forcément distincts puisqu'il n'y a pas de cycle plus court). Par définition d'un chemin orienté, T contient les arcs (s_1, s_2) et (s_2, s_3), or comme le graphe est transitif cela implique qu'il contient l'arc (s_1, s_3). En remplaçant les deux arcs précédents par celui-ci, on obtient un cycle de longueur $k - 1$, ce qui est impossible puisqu'on a supposé qu'il n'existait pas de cycle plus court.

Le tournoi T ne peut donc pas contenir de cycle.

NB: raisonner sur le cycle de longueur minimale est une astuce évitant d'écrire une démonstration par récurrence, mais ça marcherait aussi très bien. Cela dit, j'ai également accepté des démonstrations moins pointilleuses (et en particulier, je n'exigeais pas que vous explicitiez les cas $k < 3$).

5.2 Rois d'un tournoi

Dans un graphe orienté, un *roi* est un sommet r tel que la distance (orientée) de r à tous les autres sommets est au plus 2 (autrement dit, $\forall s \in S : \vec{\mathrm{d}}(r, s) \leq 2$). On va montrer qu'un tournoi, transitif ou non, a forcément (au moins) un roi.

Question 5.5. Soit u et v deux sommets d'un tournoi. Montrer que si $\vec{\mathrm{d}}(u, v) > 1$, alors u est un successeur de v.

Supposons que $\vec{\mathrm{d}}(u, v) > 1$. Par définition d'un tournoi, il y a forcément soit un arc (u, v) soit un arc (v, u). S'il y avait un arc (u, v), la distance de u à v serait de 1, ce qui est contraire à l'hypothèse. Le tournoi contient donc nécessairement l'arc (v, u), et u est bien successeur de v.

(Attention, les successeurs d'un sommet s sont les sommets s' tels qu'il existe un arc (s, s'). Ce ne sont pas les sommets atteignables depuis s par un chemin orienté.)

Question 5.6. Soit u et v deux sommets d'un tournoi. Montrer que si $\vec{\mathrm{d}}(u, v) > 2$, alors tout successeur de u est un successeur de v.

Supposons que $\vec{\mathrm{d}}(u, v) > 2$. Soit w un successeur de u. Le tournoi ne peut pas contenir l'arc (w, v), car dans ce cas (u, w, v) serait un chemin orienté de longueur 2 de u à v, ce qui contredit l'hypothèse. Comme le tournoi contient nécessairement un arc entre w et v, c'est donc (v, w) qui est présent, et w est bien successeur de v.

Question 5.7. Soit u et v deux sommets d'un tournoi tels que $\vec{\mathrm{d}}(u, v) > 2$. En utilisant les deux questions précédentes, montrer que $\mathrm{d}^+(u) < \mathrm{d}^+(v)$. (Rappel : $\mathrm{d}^+(s)$, le degré sortant de s, est son nombre d'arcs sortants).

On note $\mathrm{Succ}(s)$ l'ensemble des successeurs d'un sommet s. Notons que $\mathrm{d}^+(s) = |\mathrm{Succ}(s)|$.

Supposons que $\vec{\mathrm{d}}(u, v) > 2$. Notons $\mathrm{Succ}(u)$ les successeurs de u et $\mathrm{Succ}(v)$ les successeurs de v. Par la question précédente, tous les successeurs de u sont également successeurs de v, donc $\mathrm{Succ}(u) \subseteq \mathrm{Succ}(v)$.

Mais par la question d'avant, on sait aussi que u est un successeur de v (puisque $\vec{\mathrm{d}}(u, v) > 2 > 1$), or u n'est pas successeur de lui-même (car un tournoi n'a pas de boucle), donc $\mathrm{Succ}(u) \neq \mathrm{Succ}(v)$.

On a donc $\mathrm{Succ}(u) \subsetneq \mathrm{Succ}(v)$ (inclusion stricte), donc $\mathrm{d}^+(u) < \mathrm{d}^+(v)$.

Je n'attendais pas une démonstration aussi formelle. Si vous me disiez que les successeurs de v sont au moins tous les successeurs de u plus u lui-même, ça suffisait.

Question 5.8. Soit u un sommet d'un tournoi. En utilisant la question précédente, montrer que si u n'est pas un roi, alors u n'est pas de degré sortant maximal.

Si u n'est pas un roi, alors il existe un sommet s tel que $\vec{\mathrm{d}}(u, s) > 2$ (sinon u serait un roi). Par la question précédente, cela implique que $\mathrm{d}^+(u) < \mathrm{d}^+(s)$, donc u n'est pas de degré sortant maximal.

Question 5.9. Utiliser la question précédente pour montrer que tout tournoi a forcément (au moins) un roi.

Les sommets de degré sortant maximal sont tous des rois. En effet, soit u un sommet de degré sortant maximal ; s'il n'était pas un roi, alors il y aurait un autre sommet s de degré sortant supérieur, ce qui est absurde. (On peut aussi dire tout simplement que c'est la contraposée du résultat de la question précédente.)

Un tournoi ayant forcément au moins un sommet de degré sortant maximal, il a donc forcément au moins un roi.

Remarque : comme cela a été relevé dans une copie, la dernière phrase n'est vraie que parce qu'il y a un nombre fini de sommets. On peut aussi considérer des graphes infinis, auquel cas ce n'est pas aussi simple : par exemple, quel est le sommet de degré sortant maximal dans le graphe de la relation $>$ sur $\mathbb{N}$? Mais ça sort du cadre du cours.