DS3 (3 heures)

Lisez tout le texte avant de commencer. La plus grande importance sera attachée à la clarté, à la précision et à la concision de la rédaction. Toute réponse non justifiée ne sera pas prise en compte. Si vous repérez ce qui vous semble être une erreur d'énoncé, signalez la sur votre copie et poursuivez votre composition en expliquant les raisons de vos éventuelles initiatives. L'usage de tout dispositif électronique est interdit.
Le sujet comporte quatre exercices. Vous devez traiter les exercices 1 et 2 et, au choix, soit l'exercice 3, soit l'exercice 4. À titre indicatif, une estimation des durées de traitement et des niveaux de difficultés de chaque exercice est donnée ci-dessous.

- *Exercice 1 : 0,5 h (*)*
- *Exercice 2 : 0,5 h (*)*
- *Exercice 3 : 2,0 h (**)*
- *Exercice 4 : 2,0 h (**)*

Dans tout ce sujet, le seul langage de programmation est OCaml. Seules les fonctions incluses dans la bibliothèque standard du langage sont autorisées.

Exercice 1

Sur l'alphabet $\Sigma = \{a, b\}$, on considère le langage L des mots dans le complémentaire de $\{m \in \Sigma^* \mid \exists u \in \Sigma^*, m = uu\}$.

Question 1. Donner deux exemples de mots dans L, l'un de longueur 4, l'autre de longueur 5.

Question 2. Montrer que tout mot de longueur impaire est dans L et que tout mot $m_1 \dots m_{2n}$ de longueur paire appartient à L si et seulement s'il existe un indice $i \in [\![1, n]\!]$ tel que $m_i \neq m_{n+i}$.

On considère la grammaire algébrique $\mathcal{G}$ de symbole initial S dont les règles sont :

$$S \to A \mid B \mid AB \mid BA$$
$$A \to a \mid aAa \mid aAb \mid bAa \mid bAb$$
$$B \to b \mid aBa \mid aBb \mid bBa \mid bBb$$

Les notations $\mathcal{L}(\mathcal{G}, A)$, $\mathcal{L}(\mathcal{G}, B)$ et $\mathcal{L}(\mathcal{G})$ désignent respectivement les langages dérivés à partir de A, B et S.

Question 3. Décrire $\mathcal{L}(\mathcal{G}, A)$ et $\mathcal{L}(\mathcal{G}, B)$.

Question 4. Montrer que tout mot $m \in L$ de longueur impaire est dans $\mathcal{L}(\mathcal{G})$.

Question 5. Montrer que tout mot $m \in L$ de longueur paire est dans $\mathcal{L}(\mathcal{G})$.

Question 6. Montrer que le langage L est algébrique.

Exercice 2

Pour chacun des problèmes suivants, en justifiant votre réponse, déterminer s'il est *décidable* puis s'il est *semi-décidable* ?

Question 1. Problème A

- *Instance* : une fonction f et son argument x.
- *Question* : le calcul de $f(x)$ termine-t-il ?

Question 2. Problème F

- *Instance* : une fonction f.
- *Question* : le nombre d'arguments x pour lesquels $f(x)$ termine est-il fini ?

Question 3. Problème D

- *Instance* : une fonction f.
- *Question* : le nombre d'arguments x pour lesquels $f(x)$ termine est-il supérieur ou égal à 10 ?

Exercice 3

Un *arbre binaire non raciné* est un graphe connexe sans cycle dont les nœuds sont de degrés 1 ou 3 . Un *arbre binaire raciné* est un graphe connexe sans cycle dont les nœuds sont de degrés 1 ou 3 sauf exactement un nœud qui est de degré 2. Les nœuds de degré 1 sont appelés des *feuilles*, les nœuds de degré 2 ou 3 des *nœuds internes* et l'unique nœud de degré 2 d'un arbre binaire raciné est appelé sa *racine*. Un graphe réduit à un unique nœud est considéré comme un arbre raciné dont le nœud est à la fois feuille et racine. Les arêtes d'un arbre sont appelées des *branches*, les branches reliant des sommets de degré 2 ou 3 étant appelées des *branches internes*. Les feuilles d'un arbre à n feuilles sont étiquetées par des entiers distincts entre 1 et n. On s'intéresse à des opérations d'édition qui transforment un arbre à n feuilles en un second arbre à n feuilles.

La première opération, notée SPR, pour *Subtree Prune and Regraft* (ou découpe et greffe d'un sous-arbre), prend en entrée 2 branches e et f de l'arbre. La branche e est supprimée, créant deux arbres racinés par les sommets à chaque extrémité de e ; notons ces arbres T_1 (qui contient la branche f) et T_2 (qui ne la contient pas). On divise ensuite la branche f pour créer un nouveau nœud v de degré 2. Le sous-arbre raciné T_2 est rattaché par sa racine à v par une nouvelle branche ; v devient de degré 3. On obtient alors un arbre raciné, avec une racine de degré 2, que l'on supprime en reliant directement ses deux voisins pour obtenir un arbre non raciné.

Une deuxième autre opération, notée NNI, pour *Nearest Neighbor Interchange* (ou échange entre plus proches voisins), prend une branche interne de l'arbre e et échange entre eux deux des quatre sous-arbres incidents à cette branche. C'est-à-dire, en notant a et b les extrémités de la branche interne e, a a deux sommets voisins qui ne sont pas b, a_1 et a_2, et de même pour b, b_1 et b_2. On choisit alors un des a_i et un des b_j à échanger, par exemple a_1 et b_1, on supprime l'arête reliant a à a_1 et b à b_1 et on relie a à b_1 et b à a_1.

La figure 1 présente quelques exemples d'arbres non racinés.

- L'arbre 1 est un arbre binaire non raciné avec ses feuilles étiquetées.
- L'arbre 2 est obtenu à partir de l'arbre 1 par un SPR en coupant la branche e et en la greffant sur la branche f.
- Dans l'arbre 3, les triangles représentent des sous-arbres racinés et la branche interne e sépare 4 sous-arbres racinés A, B, C et D.
- L'arbre 4 est obtenu à partir de l'arbre 3 par un NNI sur la branche e, en échangeant les sous-arbres B et D.
- L'arbre 5 est obtenu à partir de l'arbre 1 par un NNI sur la branche e, en échangeant le sous-arbre contenant la feuille 5 et le sous-arbre contenant les feuilles 1 et 2.

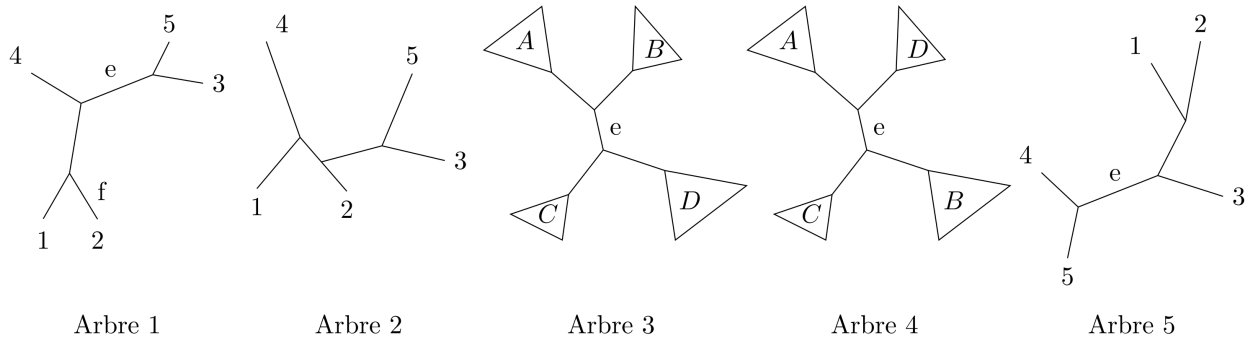


Figure 1 – Exemple d'arbres non racinés

Dans un graphe non orienté, un *chemin* est une suite de sommets reliés par des arêtes. On dit qu'un chemin *passe* par un sommet si ce sommet appartient au chemin. Un *circuit* est un chemin qui commence et se termine au même sommet. Un *chemin hamiltonien* est un chemin qui passe une et une seule fois par chaque sommet du graphe. Un *circuit hamiltonien* est un circuit qui passe par chaque sommet une et une seule fois.

On étudie l'espace $B(n)$ des arbres binaires non racinés étiquetés de n feuilles suivant ces deux processus d'édition. On définit $G_{\text{NNI}}(n)$ le graphe dont les sommets sont les arbres de $B(n)$ et dans lequel une arête relie deux arbres si l'on peut passer de l'un à l'autre par un mouvement NNI. On définit de même $G_{\text{SPR}}(n)$ avec les mouvements SPR.

Prise en main

Question 1. Montrer que l'on peut passer de l'arbre 1 à l'arbre 2 par une opération NNI. Dessiner tous les voisins de l'arbre 1 dans $G_{\text{NNI}}(5)$.

Question 2. Pour $n \geqslant 2$, donner le nombre de branches et de nœuds d'un arbre de $B(n)$.

Question 3. On note $RB(n)$ l'ensemble des arbres binaires racinés à n feuilles étiquetées. Montrer les deux résultats suivants.

$$|RB(n)| = (2n-3) \times |B(n)| \qquad |B(n)| = |RB(n-1)|$$

En déduire le nombre d'arbres binaires $B(n)$ en fonction de n.

Question 4. Tracer $G_{\text{NNI}}(4)$.

Question 5. Combien de voisins un arbre de $B(n)$ a-t-il dans $G_{\text{NNI}}(n)$?

On appelle *arbre chenille* un arbre binaire non raciné dans lequel il existe un chemin passant une et une seule fois par chaque nœud interne.

Question 6. Montrer que l'on peut passer de tout arbre de $B(n)$ à un arbre chenille en effectuant seulement des mouvements NNI. En déduire que $G_{\text{NNI}}(n)$ est connexe.

Question 7. Montrer que tout mouvement SPR peut se décomposer en mouvements NNI et que tous les mouvements NNI sont des mouvements SPR. En déduire que $G_{\text{SPR}}(n)$ est connexe.

Étude de $G_{\text{NNI}}(5)$

Question 8. Combien de sommets possède $G_{\text{NNI}}(5)$ et quel est le degré de ces sommets ?

Question 9. Montrer que tout arbre de $G_{\text{NNI}}(5)$ fait partie de cycles de longueur 5, de deux cycles de longueurs 3, et que pour tout arbre deux arbres sont à distance 3 de cet arbre.

Question 10. Tracer $G_{\text{NNI}}(5)$.

Construction de $G_{\text{NNI}}(n)$

Afin d'écrire un programme qui construit $G_{\text{NNI}}(n)$, on définit le type suivant pour représenter les arbres binaires (racinés ou non).

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

Un arbre raciné est ainsi représenté par sa racine, qui peut être soit une feuille soit un nœud interne ayant un sous-arbre gauche et un sous-arbre droit. Un arbre non raciné à n feuilles est représenté à partir d'une racine fictive ajoutée entre la feuille d'étiquette n et le nœud auquel elle est reliée. Enfin, pour assurer l'unicité de la représentation informatique d'un arbre, et ainsi simplifier les programmes, on adopte la convention, pour chaque nœud interne, que la plus grande des étiquettes du sous-arbre droit est supérieure aux étiquettes du sous-arbre gauche.
Par exemple, l'arbre 1 de la figure 3 est représenté par :

```
Noeud (Noeud (Feuille 3,
       Noeud (Noeud (Feuille 1,
                     Feuille 2),
              Feuille 4)),
       Feuille 5);;
```

On représente les graphes par liste d'adjacence, plus précisément par une liste de couples dont le premier élément est le nom d'un sommet et le second la liste d'adjacence de ce sommet.

```
type graphe = (arbre * arbre list) list;;
```

Question 11. Écrire une fonction `feuilles` qui prend en argument un arbre et renvoie la liste des étiquettes de ses feuilles.

Question 12. Écrire une fonction `degres` qui prend en argument un graphe implémenté par liste d'adjacence et renvoie la liste des degrés de ces nœuds.

Question 13. Écrire une fonction `egaux` qui teste si deux arbres sont égaux, au sens des arbres binaires non racinés étiquetés. Justifier que cette fonction est correcte.

Question 14. Étant donnés une liste d'arbres et un arbre, écrire une fonction `appartient` qui teste si l'arbre fait partie de la liste.

Question 15. En remarquant que parmi les 4 sous-arbres à échanger lors d'une opération NNI autour d'une branche interne, on peut en choisir un qui restera fixe, écrire une fonction `voisinsNNI` qui prend en argument un arbre et renvoie tous les arbres que l'on peut obtenir à partir de celui-ci par un mouvement NNI.

Question 16. Écrire une fonction `chenille` qui prend en argument un entier n et renvoie l'arbre chenille dans lequel les feuilles sont rangées de 1 à n.

Question 17. Écrire une fonction `insere` qui prend en arguments un arbre et un graphe et ajoute l'arbre à la liste des sommets du graphe.

Question 18. Écrire une fonction `relie` qui prend en arguments un graphe et deux arbres et qui rajoute au graphe une arête reliant les deux sommets, si elle n'est pas déjà présente.

Question 19. Écrire une fonction `grapheNNI` qui prend en argument un entier n et renvoie $G_{\text{NNI}}(n)$, en implémentant le graphe avec des listes d'adjacences. Justifier la correction et la terminaison de votre programme.

$G_{\text{SPR}}(n)$ est hamiltonien

On va démontrer par récurrence que $G_{\text{SPR}}(n)$ est hamiltonien, c'est-à-dire qu'il possède un circuit hamiltonien. On note S_i l'ensemble des arbres de taille $n+1$ obtenu à partir d'un arbre t_i de taille n en ajoutant une feuille étiquetée $n+1$ à une des branches de t_i. Pour ajouter une feuille à une branche, on supprime la branche et on crée un nouveau nœud, relié à la nouvelle feuille et aux deux nœuds qui étaient reliés par la branche.

Question 20. Montrer que les sous-graphes $G_{\text{SPR}}(n+1)_{|S_i}$ induits dans $G_{\text{SPR}}(n+1)$ par les sommets de S_i sont complets.

Question 21. Si t_i et t_j sont deux sommets de $G_{\text{SPR}}(n)$ reliés entre eux, montrer qu'il existe des arêtes entre S_i et S_j dans $G_{\text{SPR}}(n+1)$.

Question 22. Démontrer par récurrence que $G_{\text{SPR}}(n)$ est hamiltonien.

Exercice 4

Préliminaires

Ce sujet traite des graphes connexes non orientés et non pondérés $G = (S, A)$ tels que l'ensemble des sommets est défini par $S = \{0, 1, \dots, n-1\}$ avec $n = |S|$. Si a et a' sont deux arêtes de A, on dit que a est *incidente* à a' si et seulement si $a \neq a'$ et $a \cap a' \neq \varnothing$, c'est-à-dire si a et a' sont distinctes et partagent un sommet en commun. On dit que a est *incidente* à $s \in S$ si et seulement si $s \in a$. On appelle *degré* de s, noté $\deg(s)$, le nombre d'arêtes incidentes à s. On note $V(s)$ l'ensemble des voisins de s, c'est-à-dire l'ensemble des sommets t tels que $\{s, t\} \in A$, et $I(s)$ l'ensemble des arêtes incidentes à s. On remarque que $\deg(s) = |I(s)|$.
On appelle *tas de sable* un couple $G_q = (G, q)$ où $G = (S, A)$ est un graphe et $q \in S$ est un sommet appelé *puits*. Les sommets de $S \setminus \{q\}$ sont appelés sommets *réguliers*. Une *configuration* de G_q est un n-uplet $c = (c_0, \dots, c_{n-1}) \in \mathbb{Z}^n$ tel que si s est régulier, $c_s \geqslant 0$. Informellement, une valeur c_s (ou $c(s)$ si la notation est plus pertinente) correspond au nombre de grains de sable sur le sommet s. Seul le puits peut avoir un nombre de grains de sable négatif. Le *poids* d'une configuration c est la somme $|c| = \sum_{s=0}^{n-1} c_s$.
Un sommet régulier s est dit *stable* pour une configuration c si $c_s < \deg(s)$. Il est dit *instable* sinon. L'*éboulement* d'un sommet instable consiste à distribuer l'un de ses grains de sable à chacun de ses voisins. Plus formellement, si c et c' sont deux configurations, on dit que c' est obtenue par éboulement de s dans c, noté $c \to_s c'$, si et seulement si, pour tout sommet $t \in S$:

- si $t = s$, alors $c'_t = c_t - \deg(t)$;
- si $t \in V(s)$, alors $c'_t = c_t + 1$;
- sinon, $c'_t = c_t$.

On dit que c s'éboule en c', noté $c \to c'$, s'il existe un sommet régulier $s \in S$ tel que $c \to_s c'$. La figure 2 illustre un éboulement. Le sommet grisé est le puits. Les valeurs des configurations sont notées à côté des sommets.
Pour $s \in S$, on pose Δ_s la configuration telle que $\Delta_s(s) = -\deg(s), \Delta_s(t) = 1$ si $t \in V(s)$ et $\Delta_s(t) = 0$ sinon. On remarque que $c \to_s c + \Delta_s$.
On note $\to^*$ la clôture réflexive et transitive de $\to$, c'est-à-dire que $c \to^* c'$ si et seulement s'il existe $k \in \mathbb{N}$ et des configurations $c = c_0, c_1, \dots, c_k = c'$ telles que pour tout $i \in [\![0, k-1]\!], c_i \to c_{i+1}$.
Une configuration dont tous les sommets réguliers sont stables est elle-même dite *stable*. On appelle *avalanche* une suite d'éboulements qui termine par une configuration stable.

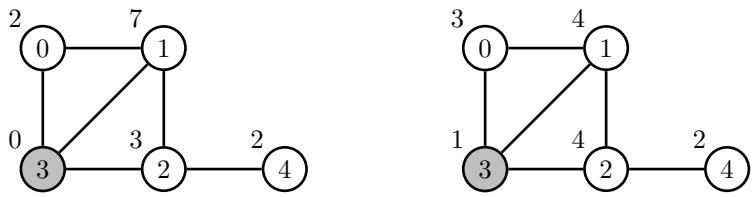


Figure 2 – Un tas de sable et deux configurations c et c' telles que $c \to_1 c'$.

Question 1. Montrer que si c s'éboule en c' alors c et c' ont le même poids.

Question 2. Montrer que les éboulements sont commutatifs : si $s, t \in S^2, c \to_s c_1 \to_t c_2$ et $c \to_t c'_1 \to_s c'_2$ alors $c_2 = c'_2$.

Question 3. Sans justification, représenter graphiquement une configuration obtenue par avalanche depuis c, la configuration de gauche dans la figure 2. Quelle est la longueur de l'avalanche (en nombre d'éboulements) ?

Pour les trois questions suivantes, on considère c une configuration d'un tas de sable G_q.

Question 4. Montrer que toute avalanche depuis c est finie.
Indication : on pourra raisonner par l'absurde et considérer l'ensemble des sommets qui s'éboulent une infinité de fois.

Question 5. Un *multi-ensemble* est un ensemble où on autorise les répétitions d'un même élément. Soit c une configuration d'un tas de sable G_q et soit $(s_1, s_2, \dots, s_k)$ et $(t_1, t_2, \dots, t_\ell)$ deux suites de sommets telles que $c \to_{s_1} c_1 \to_{s_2} \dots \to_{s_k} c_k$ et $c \to_{t_1} c'_1 \to_{t_2} \dots \to_{t_\ell} c'_\ell$. On suppose que c_k et c'_ℓ sont stables. Montrer que $k = \ell$ et que les multi-ensembles $\{s_1, \dots, s_k\}$ et $\{t_1, \dots, t_\ell\}$ sont égaux.

Question 6. En déduire qu'il existe une unique configuration stable obtenue par avalanche depuis c.

Par la suite, on notera $\text{av}(c)$ l'unique configuration stable telle que $c \to^* \text{av}(c)$.

Implémentation

On représente un graphe par un tableau de listes d'adjacence du type suivant.

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

Si $G = (S, A)$ est représenté par `g` de type `graphe`, alors `g` est un tableau de taille $n = |S|$, et pour $s \in S$, `g.(s)` est une liste de taille $\deg(s)$ contenant les éléments de $V(s)$ dans un ordre quelconque.
Une configuration c dans un tas de sable $G_q = ((S, A), q)$ est un tableau `c` de taille n tel que pour $s \in S$, `c.(s)` est égal à c_s.

Question 7. Écrire une fonction `eboulement (g: graphe)(c: int array)(s: int): unit` qui calcule l'éboulement de s dans c, une configuration du graphe G. La fonction devra modifier directement le tableau c pour le transformer en la configuration obtenue.

Question 8. En déduire une fonction `avalanche (g: graphe)(q: int)(c: int array): unit` qui calcule la configuration obtenue par avalanche depuis c dans le tas de sable G_q. La fonction devra modifier directement le tableau `c` pour le transformer en la configuration obtenue.

Question 9. Déterminer la complexité temporelle et spatiale de la fonction `avalanche` en fonction de n et de k, la longueur d'avalanche depuis c.

Configurations récurrentes

Question 10. Montrer qu'il existe $s \in S$ tel que le graphe induit $G[S \setminus \{s\}]$ est connexe.

Dans la suite de l'exercice, on suppose que le puits q est un tel sommet de sorte que le sous-graphe induit par les sommets réguliers est connexe.
Une configuration c est dite *nulle* si $c_s = 0$ pour tout sommet régulier s. Elle est dite *récurrente* s'il existe une configuration c' non nulle telle que $\mathrm{av}(c + c') = c$, où l'addition des configurations correspond à l'addition terme à terme.

Question 11. Montrer que si $c \to^* c'$ et $d \to^* d'$, alors $c + d \to^* c' + d'$.

Question 12. Une configuration nulle peut-elle être récurrente ? Justifier.

Question 13. Montrer que pour une configuration c non nulle, il existe un entier k et une configuration c' telle que $kc \to^* c'$ et $c'_s > 0$ pour tout s sommet régulier.

Question 14. On pose δ la configuration telle que $\delta_s = \deg(s)$ pour tout sommet s. Montrer que c est une configuration récurrente si et seulement s'il existe une configuration c' telle que $\mathrm{av}\,(c' + \delta) = c$.

Question 15. On pose $\beta = 2\delta - \mathrm{av}(2\delta)$. Calculer β pour le tas de sable de la figure 2.

Question 16. Montrer que $\delta + \beta \to^* \delta$.

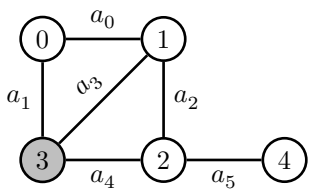
Question 17. Montrer que c est une configuration récurrente si et seulement si $\mathrm{av}(c + \beta) = c$.

Parcours décroissants

Pour $(G = (S, A), q)$ un tas de sable, avec $n = |S|$ et $m = |A|$, on se donne une numérotation arbitraire des arêtes $A = \{a_0, \dots, a_{m-1}\}$. On considère la relation $<_A$ définie sur A^2 par $a <_A a'$ si et seulement s'il existe $0 \leqslant i < j < m$ tels que $a = a_i$ et $a' = a_j$.
On appelle *parcours arêtes-sommets* une suite $\sigma = (\sigma_i)_{i \in [\![0, n+m-1]\!]}$ où chaque sommet et chaque arête de G apparaît exactement une fois. Pour σ une telle séquence et $i \in [\![0, n + m]\!]$, on note $\sigma^{<i} = \{\sigma_j \mid j < i\}$. Un parcours arêtes-sommets est dit *décroissant* si :

- si σ_i est un sommet régulier, alors σ_{i-1} est une arête incidente à σ_i ;
- si σ_i est une arête, alors cette arête est maximale selon $<_A$ parmi les arêtes qui ne sont pas dans $\sigma^{<i}$ et qui sont incidentes à un sommet de $\sigma^{<i}$.

On remarque en particulier qu'un parcours décroissant commence par le puits du tas de sable. La figure 3 donne un exemple de parcours décroissant. On note $D\left(G_q\right)$ l'ensemble des parcours décroissants de G_q.



$$\sigma = \{3, a_4, a_3, 1, a_2, 2, a_5, 4, a_1, 0, a_0\}$$

Figure 3 – Un tas de sable et un parcours décroissant.

Question 18. Soient $\sigma \neq \tau$ deux parcours décroissants d'un même tas de sable. On pose k le plus petit indice tel que $\sigma_k \neq \tau_k$. Montrer que $\{\sigma_k, \tau_k\}$ est un ensemble constitué d'un sommet et d'une arête.

Question 19. Pour σ un parcours décroissant, on pose $\Phi(\sigma) = \{\sigma_{k-1} \mid \sigma_k \in S \setminus \{q\}\}$. Montrer que $(S, \Phi(\sigma))$ est un arbre couvrant de G.

Question 20. On note $C(G)$ l'ensemble des parties de A qui forment un arbre couvrant de G. Montrer que Φ est une bijection de $D\left(G_q\right)$ dans $C(G)$.

Pour σ un parcours décroissant, on dit qu'une arête $\sigma_i = \left\{\sigma_j, \sigma_k\right\}$ est *forte* si elle apparaît après ses extrémités, c'est-à-dire si $j < i$ et $k < i$. Dans le parcours de la figure 3, l'arête a_0 est une arête forte.
Pour $T = (S, B)$ un arbre couvrant de G, on dit qu'une arête $a \in A \setminus B$ est *active extérieurement* si c'est l'arête minimale selon $<_A$ dans l'unique cycle de $(S, B \cup \{a\})$.

Question 21. Soit σ un parcours décroissant et $a \in A$. Montrer que a est forte pour σ si et seulement si elle est active extérieurement pour $\Phi(\sigma)$.

Parcours décroissants et configurations récurrentes

Pour σ un parcours décroissant, on pose $\Psi(\sigma)$ la configuration c telle que $c_{\sigma_i} = \left|I\left(\sigma_i\right) \setminus \sigma^{<i}\right|$.

Question 22. Déterminer la configuration $\Psi(\sigma)$, pour σ le parcours décroissant de la figure 3.

Question 23. Montrer que si σ est un parcours décroissant, alors $\Psi(\sigma)$ est une configuration récurrente.
Indication : on pourra considérer $\Psi(\sigma) + \Delta_q$.

Question 24. On note $\mathrm{Rec}\left(G_q\right)$ l'ensemble des configurations récurrentes c de G_q telles que $c_q = \deg(q)$. Montrer que Ψ est une bijection de $D\left(G_q\right)$ dans $\mathrm{Rec}\left(G_q\right)$.

Question 25. Montrer que si σ est un parcours décroissant, alors la somme entre le nombre d'arêtes et le nombre d'arêtes fortes pour σ est égale au poids de $\Psi(\sigma)$.