DS5

Durée : 4 heures

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

Le sujet comporte deux exercices et un problème.

Exercice 1

Dans toute cette partie, la lettre ε désigne le mot vide, Σ désigne un alphabet et $\Sigma^\star$ l'ensemble des mots finis sur Σ.

Définition 1. Un automate déterministe A est un quintuplet $A = (Q, \Sigma, q_0, F, \delta)$, avec :
- Q un ensemble d'états ;
- Σ un alphabet
- q_0 l'état initial
- $F \subseteq Q$ un ensemble d'états finaux ;
- $\delta : Q \times \Sigma \to Q$ une application de transition.

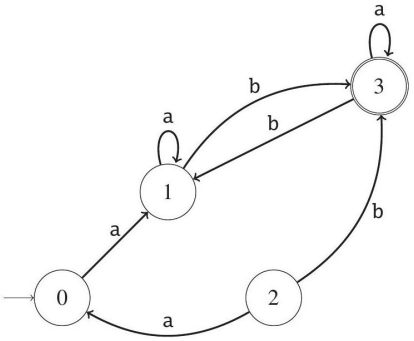
Définition 2. Soit L un langage sur Σ. Le carré du langage L est l'ensemble $\{uu, u \in L\}$. Il est noté $L \odot L$.

Définition 3. Soit L un langage sur Σ. La racine carrée du langage L est l'ensemble $\{u \in \Sigma^\star \mid uu \in L\}$. Elle est notée $\sqrt{L}$.

1. Décrire $\sqrt{L}$ lorsque $\Sigma = \{a, b\}$ et L est décrit par l'expression rationnelle $a^\star b^\star$.
2. Décrire $\sqrt{L}$ lorsque $\Sigma = \{a, b\}$ et L est décrit par l'expression rationnelle $b^\star a^\star b^\star$.

   Définition 4. Soient $A = (Q, \Sigma, q_0, F, \delta)$ un automate fini déterministe, q' un élément de Q et F' une partie de Q. L'automate $(Q, \Sigma, q', F', \delta)$ est noté $A_{q',F'}$. Si on note L le langage reconnu par A, $L_{q',F'}$ désigne le langage reconnu par $A_{q',F'}$.

   Ici, L désigne le langage reconnu par l'automate A suivant :



3. Construire un automate reconnaissant $L_{3,\{1\}}$ en modifiant légèrement l'automate A.
4. On veut construire l'automate de Glushkov de L décrit par $a\,(a + ba^\star b)^\star\, ba^\star$.
   (a) Décrire L', le linéarisé de L.
   (b) Déterminer les préfixes de L' de longueur 1 , les suffixes de L' de longueur 1 et les facteurs de L' de longueur 2.
   (c) En déduire l'automate de Glushkov G de L.
5. Déterminiser l'automate G.

   Ici, on fixe L un langage rationnel sur un alphabet Σ et $A = (Q, \Sigma, q_0, F, \delta)$ un automate fini reconnaissant celui-ci.

6. Soit u un mot de $\Sigma^\star$. Montrer que u est un élément de $\sqrt{L}$ si et seulement s'il existe un $q \in Q$ tel que $u \in L_{q0,\{q\}}$ et $u \in L_{q,F}$.
7. En déduire que $\sqrt{L}$ est un langage rationnel.
8. Montrer que l'on a $(\sqrt{L} \odot \sqrt{L}) \subset L$.

Exercice 2

On considère la grammaire algébrique G sur l'alphabet $\Sigma = \{a, b\}$ et d'axiome S dont les règles sont :

$$S \to SaS \,|\, b$$

1. Cette grammaire est-elle ambiguë ? Justifier.
2. Déterminer (sans preuve pour cette question) le langage L engendré par G. Quelle est la plus petite classe de langages à laquelle L appartient ?
3. Prouver que $L = L(G)$.
4. Décrire une grammaire qui engendre L de manière non ambiguë en justifiant de cette non ambiguité.
5. Montrer que tout langage dans la même classe de langages que L peut être engendré par une grammaire algébrique non ambiguë.

Problème

La correspondance de Burge permet d'exprimer une bijection entre des graphes simples et des tableaux de Young semi-standards. Ces objets combinatoires ont de nombreuses applications, notamment dans l'étude de groupes symétriques et la géométrique algébrique. En théorie des graphes, ils permettent également de trouver, s'ils existent, les graphes simples dont les sommets sont contraints à avoir des degrés donnés.

Cette partie comporte des questions nécessitant un *code OCaml*. Pour ces questions, *les réponses ne feront pas appel aux fonctionnalités impératives du langage* (références, champs mutables, exceptions).

Notations

Dans toute la suite on notera :
— $[l] = [\![1, l]\!]$ l'ensemble des entiers naturels de 1 à l,
— $|E|$ le cardinal de l'ensemble E,
— $G = ([l], A)$ un *graphe simple*, c'est-à-dire un graphe non orienté à l sommets et m arêtes, sans boucles ni arêtes multiples,
— $d_i = |\{j \in [l], (i, j) \in A\}|$ le degré du sommet $i \in [l]$ dans le graphe $G = ([l], A)$.

De plus, les sommets de G seront ordonnés par degrés décroissants, de sorte que $d_1 \geq d_2 \geq \cdots \geq d_l \geq 0$.

Définition 1 (Suite de degrés d'un graphe)**.** Soit $G = ([l], A)$ un graphe simple. Si d_i, $i \in [l]$ est le degré du sommet i, avec $d_1 \geq d_2 \ldots \geq d_l \geq 0$, la *suite de degrés* de G est le l-uplet $(d_1, d_2, \ldots, d_l)$.

Partitions d'un entier

Définition 2 (Partition)**.** Une *partition* d'un entier $n \in \mathbb{N}^*$, notée $\lambda \vdash n$, est une suite décroissante $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l)$ de nombres entiers strictement positifs de somme n.

Par exemple, $n = 4$ a cinq partitions : (4), $(3, 1)$, $(2, 2)$, $(2, 1, 1)$ et $(1, 1, 1, 1)$.
Une partition $\lambda = (\lambda_1, \ldots, \lambda_l) \vdash n$ peut être vue comme une suite de degrés sous certaines conditions.

Question 1. Montrer que si $\sum_{i=1}^{l} \lambda_i$ est impaire, alors la partition $\lambda \vdash n$ ne peut pas générer de graphe simple ayant la suite de degrés $(\lambda_1, \lambda_2, \ldots, \lambda_l)$.

Dans la suite, on considérera que la somme des termes de la suite d'entiers $(\lambda_1, \lambda_2, \ldots, \lambda_l)$ est paire. Même avec cette contrainte, cette suite peut ne pas pouvoir générer de graphe simple.

Définition 3 (Suite graphique)**.** Une suite d'entiers $d_1 \geq d_2 \ldots \geq d_l \geq 0$ est dite *graphique* s'il existe un graphe simple dont la suite des degrés est $(d_1, d_2, \ldots, d_l)$.

Notons qu'une suite graphique vérifie par définition $d_1 \geq d_2 \geq \cdots \geq d_l$.

Pour déterminer si une suite d'entiers donnée est graphique, on peut utiliser l'algorithme d'Havel-Hakimi, basé sur le théorème suivant :

Théorème 1 (Havel-Hakimi)**.**

(i) Pour tout $(d_1, \ldots, d_l) \in \mathbb{N}^l$, si $(d_1, \ldots, d_l)$ est graphique, alors $d_{d_1+1} > 0$ et toute permutation décroissante de $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_l)$ est graphique.

(ii) Réciproquement, pour tout $(d_2, \ldots, d_l) \in \mathbb{N}^{l-1}$, si $(d_2, \ldots, d_l)$ est graphique, alors pour $d_1 \in [\![d_2 + 1, l - 1]\!]$, la suite $(d_1, d_2 + 1, \ldots, d_{d_1+1} + 1, d_{d_1+2}, \ldots, d_l)$ est graphique.

Question 2. Montrer que si $(d_2, \ldots, d_l)$ est graphique, alors il existe un graphe simple $G = (S, A)$ à l sommets, de sommet 1 de degré $d_1 \in [\![d_2 + 1, l - 1]\!]$ tel que $(d_1, d_2 + 1, \ldots, d_{d_1+1} + 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_l)$ soit la suite des degrés des sommets de G.

Question 3. On suppose que $(d_1, d_2, \ldots, d_l)$ est graphique. Montrer qu'il existe $G = (S, A)$, $S = [l]$, le degré du sommet i étant d_i pour tout $i \in [l]$, tel que $\forall j \in [\![2, d_1 + 1]\!], (1, j) \in A$. Pour cela, on pourra raisonner par l'absurde et supposer que le sommet 1 est adjacent à un maximum de sommets dans $(2, \ldots, d_1 + 1)$, mais pas à tous.

Question 4. En déduire que si $d_{d_1+1} > d_{d_1+2}$, alors $(d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, d_{d_1+3}, \ldots, d_l)$ est graphique.

Question 5. Déterminer si les suites $(4, 3, 3, 3, 3)$ et $(6, 4, 4, 2, 2, 1, 1)$ sont graphiques.

Question 6. Écrire une fonction de signature `compare_entiers : int -> int -> int` qui compare deux entiers et telle que l'appel à `compare_entiers m n` renvoie 1 si `m < n`, -1 si `m > n` et 0 sinon.

Question 7. Écrire une fonction de signature `decr_n : int -> int list -> int list option` telle que pour tout $n \geq 0$ l'appel `decr_n n l` retourne `Some l'`, avec `l'` la liste `l` dont les n premiers éléments sont décrémentés, s'ils étaient tous supérieurs à 1 et `None` sinon.

Question 8. Écrire une fonction récursive de signature `havel_hakimi : int list -> bool` qui retourne `true` si la liste passée en entrée, triée par ordre décroissant, est graphique et `false` sinon. À chaque étape utilisant le théorème 1, il sera nécessaire de réordonner par ordre décroissant la liste `list` construite, ce qui pourra être fait à l'aide de l'appel à `List.sort compare_entiers list`.

Question 9. Donner deux graphes simples à $l = 5$ sommets ayant une même suite de degrés $(3, 2, 2, 2, 1)$.

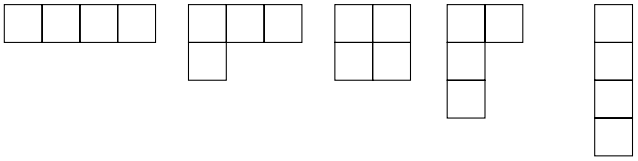
Ainsi, il n'existe pas de correspondance univoque entre suite de degrés et graphe simple.

Diagramme et tableau de Young

Définition 4 (Diagramme de Young d'une partition)**.** Le diagramme de Young de forme λ, noté $Y(\lambda)$, d'une partition $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_l) \vdash n$ est un tableau de cases constitué de l lignes alignées à gauche, chaque ligne $i \in [l]$ ayant λ_i cases.

Par convention, la ligne associée à λ_1 est la première ligne du tableau.

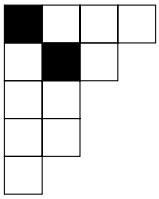
Par exemple, les diagrammes de Young des partitions de l'entier 4 sont



(4) $(3,1)$ $(2,2)$ $(2,1,1)$ $(1,1,1,1)$

Définition 5 (Diagonale d'un diagramme de Young)**.** Soit $Y(\lambda)$ un diagramme de Young. La diagonale de $Y(\lambda)$ est définie par les r cases $(i,i), i \in [r]$.

Dans le tableau suivant, $r = 2$ et les cases de la diagonale sont grisées.



Définition 6 (Partition conjuguée)**.** Soit $Y(\lambda)$ un diagramme de Young associé à $\lambda \vdash n$. La partition conjuguée de λ, notée $\lambda^* = (\lambda_1^*, \ldots, \lambda_k^*)$ est la partition obtenue en énumérant le nombre de cases de $Y(\lambda)$ par colonne, en partant de la colonne de gauche.

On remarque immédiatement que pour tout j, $\lambda_j^* = |\{i, \lambda_i \geq j)|$.

Question 10. Soit $\lambda = (5,4,1,1,1,1) \vdash 13$. Donner λ^*.

Définition 7 (Représentation de Frobenius d'un diagramme de Young)**.** Soient $(\lambda_1, \lambda_2, \ldots \lambda_l) \vdash n$ une partition, $Y(\lambda)$ son diagramme de Young et r la taille de sa diagonale. Pour $i \in [r]$, soient $\alpha_i = \lambda_i - i$ le nombre de cases à droite de la case (i,i) dans la i^{e} ligne de $Y(\lambda)$ et $\beta_i = \lambda_i^* - i$ le nombre de cases en-dessous de la case (i,i) dans la i^{e} colonne de $Y(\lambda)$. Alors $\alpha_1 > \alpha_2 \geq \ldots \alpha_r \geq 0$, $\beta_1 > \beta_2 \geq \ldots \beta_r \geq 0$ et la notation de Frobenius de λ est donnée par $\lambda = \begin{pmatrix} \alpha_1 & \alpha_2 & \ldots \alpha_r \\ \beta_1 & \beta_2 & \ldots \beta_r \end{pmatrix}$.

Question 11. Soit $(4,3,2,2,1) \vdash 12$. Donner la représentation de Frobenius de $Y(\lambda)$.

Définition 8 (Tableau de Young)**.** Soit $\lambda \vdash n$ une partition de $n > 0$. Un λ-tableau de Young est un tableau obtenu en remplissant les cases d'un diagramme $Y(\lambda)$ par des entiers dans $[n]$.

Par exemple, pour $\lambda = (2,1) \vdash 3$, les tableaux suivants sont des λ-tableaux

1 2	2 1	1 3	3 1	2 3	3 2	1 1	1 3
3	3	2	2	1	1	2	1

Dans la suite, on notera $T(i,j)$ l'entier en position (i,j) dans le tableau T.

Définition 9 (Tableau de Young (semi)standard)**.** Un tableau de Young est dit *semi standard* si les éléments de chaque ligne (respectivement colonne) forment une suite croissante de gauche à droite (respectivement strictement croissante de haut en bas). Il est dit *standard* s'il est semi-standard et si les entiers de 1 à n n'apparaissent qu'une et une seule fois.

Dans les exemples précédents, les premier, troisième et septième tableaux sont semi standards. Les premier et troisième tableaux sont standards.

Insertion de Schensted

Pour construire un tableau de Young semi-standard à partir d'une suite d'entiers $d_1, \ldots, d_l$, on utilise une méthode d'insertion proposée par Schensted.

On cherche à insérer dans un tableau de Young semi standard T un entier k, de sorte à ce que le tableau créé (contenant une case de plus) soit toujours un tableau de Young semi standard. On note cette opération d'insertion $T \leftarrow k$.

L'entier k est inséré dans T en le comparant aux valeurs de la première ligne et en déplaçant le premier entier plus grand que k en lisant la ligne de gauche à droite. S'il n'y a pas de plus grand élément, k est ajouté à la fin de la ligne. Si un élément est déplacé, il est inséré dans la deuxième ligne et les lignes suivantes du tableau sont traitées de la même manière. L'algorithme 1 décrit l'insertion de k dans la i^{e} ligne de T.

Algorithme 1 Algorithme d'insertion d'un entier k dans la ligne i d'un tableau de Young semi standard

fonction INSERTION(T, k, i)
  Entrées : un tableau de Young semi standard T, un entier k à insérer, i la ligne traitée
  Sorties : un tableau de Young semi standard contenant k
  si (la ligne i est vide) ou (k plus grand que l'élément le plus à droite de la ligne i) **alors**
    Ajouter k en bout de la ligne i de T
  sinon
    Soit j le plus petit indice tel que $k < T(i, j)$
    $p \leftarrow T(i, j)$
    $T(i, j) \leftarrow k$
    INSERTION(T, p, $i + 1$)
  fin si
fin fonction

Soit T_0 le tableau de Young semi standard

1	1	2	2	4
2	3	3		
3				
4				

Question 12. Insérer la valeur 5 dans T_0 et donner le tableau de Young semi standard résultant. Même question pour l'insertion de la valeur 1 dans T_0.

Question 13. Énoncer une précondition de l'algorithme 1 permettant de prouver sa correction, c'est-à-dire qu'il retourne bien un tableau de Young semi standard contenant k.

Pour coder les tableaux de Young, on choisit d'utiliser un type OCaml `type tableau = int list list` et on encode la structure par liste de lignes.

Question 14. Écrire une fonction récursive de signature `insereligne : int -> int list -> int * int list` qui, à partir d'un entier k et d'une ligne i d'un tableau de Young semi standard, retourne un couple (m, r) où

- $m = 0$ et r est la ligne i où k a été ajouté en queue, si k est plus grand que tous les éléments de la ligne i,
- m, qui est dans la ligne i et r est la ligne i où m a été remplacé par k sinon.

Question 15. En déduire une fonction récursive de signature `insertion : int -> tableau -> tableau` réalisant l'algorithme 1.

Question 16. Écrire une fonction récursive de signature `construit_tableau : int list -> tableau` qui construit un tableau semi standard à partir d'une suite d'entiers.

L'algorithme 1 permet de définir une position (s, t), coordonnées de la case où k a été ajouté à T.

Correspondance de Burge

Définition 10 (Tableau de Burge)**.** Soit $G = ([l], A)$ un graphe simple, $|A| = m$. Le tableau de Burge associé à G est défini par

$$\mathcal{B}_G = \begin{pmatrix} u_1 & u_2 & \dots & u_m \\ v_1 & v_2 & \dots & v_m \end{pmatrix}$$

où pour tout $i \in [m]$ (u_i, v_i) est une arête de G, avec $u_i > v_i$, le tableau étant formé de sorte que pour $i \in [m-1]$, $u_i \leq u_{i+1}$ et si $u_i = u_{i+1}$ alors $v_i > v_{i+1}$.

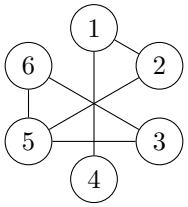


FIGURE 1 – Graphe exemple

Soit le graphe G de la figure 1.

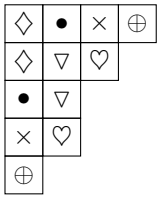
Question 17. Donner le tableau $\mathcal{B}_G$ correspondant au graphe de la figure 1.

On utilise alors $\mathcal{B}_G$ pour associer à G un diagramme de Young $Y(\lambda)$ représenté par une forme de Frobenius particulière, notée $\mathcal{F}_Y = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_r \\ \alpha_1+1 & \alpha_2+1 & \dots & \alpha_r+1 \end{pmatrix}$, où r est le nombre de cases de la diagonale principale du diagramme de Young $Y(\lambda)$ associé.

Question 18. Montrer que, dans ce cas, pour tout $i \in [r]$, $\lambda_i^* = \lambda_i + 1$.

Ainsi, $Y(\lambda)$ est divisé en deux parties symétriques. La partie inférieure est constituée de toutes les cases qui se trouvent strictement sous la diagonale et la partie supérieure est constituée du reste. Chaque position dans la partie supérieure (inférieure) du diagramme de Young correspond à une position unique, appelée *position opposée*, dans la partie inférieure (supérieure) de $Y(\lambda)$. Par exemple, dans le diagramme de Young suivant, ayant comme représentation de Frobenius $\mathcal{F}_Y$, les positions opposées sont codées par le même symbole.

$$\mathcal{F}_Y = \begin{pmatrix} 3 & 1 \\ 4 & 2 \end{pmatrix}$$



Question 19. Soit (s,t) une case. Donner la position opposée en fonction de s et t.

L'algorithme 2 utilise alors la représentation $\mathcal{B}_G$ d'un graphe G pour construire un tableau de Young semi standard dont le diagramme correspondant $Y(\lambda)$ admet une représentation de Frobenius du type $\mathcal{F}_Y$. Dans cet algorithme, $T \leftarrow v_i$ insère la valeur v_i dans le tableau de Young semi standard T.

Algorithme 2 Algorithme de Burge

Entrées : $\mathcal{B}_G$ de taille $m \times 2$
Sorties : tableau de Young semi standard T dont la représentation de Frobenius du diagramme correspondant est du type $\mathcal{F}_Y$
$T \leftarrow$ tableau vide `// Initialisation`
pour i de 1 à m **faire**
  $(s,t) = T \leftarrow v_i$
  Placer u_i dans la position opposée à (s,t)
fin pour

Cet algorithme produit, à partir du graphe de la figure 1, le tableau :

1	1	2	3
2	3	5	
4	5		
5	6		
6			

Question 20. Donner les tableaux de Young semi standard intermédiaires produits à chaque itération.

Question 21. À quoi correspond le nombre d'apparitions de chaque entier contenu dans les cases du tableau de Young résultat de l'algorithme 2 ?

Notons pour terminer qu'il est à l'inverse possible de produire un tableau bidimensionnel $\mathcal{B}_G$ (et donc un graphe G) à partir d'un tableau de Young semi standard $Y(\lambda)$ de type $\mathcal{F}_Y$.

$\star\star\star$