Banque MP inter-ENS – Session 2022

Rapport relatif à l'épreuve orale d'informatique fondamentale spécifique Ulm

— **Écoles partageant cette épreuve :** ENS Ulm

— **Coefficients** (en pourcentage du total des points de chaque concours) :
  — Concours MP option MPI : 23,1%
  — Concours INFO : 13,3%

— **Membres du jury :** Laurent Bienvenu, Brice Minaud, Charles Paperman, Pierre Senellart

Modalités de l'épreuve. L'épreuve orale d'informatique fondamentale décrite dans ce rapport est spécifique au concours d'entrée de l'École normale supérieure de Paris, et est entièrement indépendante de l'épreuve analogue qui figure au concours des autres Écoles normales supérieures. L'épreuve dure une heure, sans préparation, et vise à interroger les candidates et candidats sur des questions d'informatique fondamentale au tableau. Elle couvre des notions d'informatique principalement théoriques, mais diffère d'une épreuve de mathématiques en cela que les sujets conduisent en l'étude de notions propres à l'informatique, telles que des algorithmes ou programmes, des langages formels, des graphes, etc. Cette épreuve ne mesure pas la compétence des candidates et candidats en informatique pratique, même s'il leur est parfois demandé de présenter certains points en pseudocode.

Les sujets sont présentés aux candidates et candidats sous forme imprimée, et quelques minutes leur sont laissées pour prendre connaissance des définitions préliminaires et des premières questions. Les examinateurs ont généralement laissé les candidates et candidats traiter le début des sujets en autonomie, en progressant naturellement vers un dialogue interactif pour des questions plus délicates, ou quand il s'avérait nécessaire de préciser certains points des réponses proposées. Les sujets étaient toujours composés d'un unique problème formé de plusieurs questions successives ; pour celles et ceux qui parvenaient à la fin des questions imprimées, les questions suivantes étaient posées directement à l'oral au tableau.

Résultats. Cette année, le jury a examiné 79 candidates et candidats admissibles aux concours MP et INFO. Le jury n'a pas connaissance de quel concours est présenté par les candidates et candidats qu'il auditionne, et les évalue donc de la même manière. Conformément aux instructions fournies pour l'harmonisation, les notes se sont réparties de 5 à 19,4 avec une moyenne de $12,02$ et un écart type de 3,46. Malgré ces écarts inhérents à une épreuve de concours, le jury tient à féliciter l'ensemble des candidats dont le niveau global était élevé.

Programme. L'épreuve porte sur le programme de l'option informatique des *deux* années de classes préparatoires (MPSI et MP), ainsi que sur le programme informatique commun, et peut également faire appel à des compétences mathématiques exigibles suivant les programmes de cette discipline. Il est fortement recommandé aux candidates et candidats de *prendre connaissance de ces programmes* et de s'assurer qu'ils et elles maîtrisent effectivement les points qui y figurent. Les examinateurs ont parfois, au fil de l'oral, demandé à la candidate ou au candidat de préciser un point sous la forme d'une question de cours. Une réponse correcte et immédiate est alors attendue, le cas contraire étant fortement sanctionné.

Bien entendu, les sujets proposés aux candidates et aux candidats demandaient d'explorer des notions nouvelles allant au-delà du programme, et qui étaient donc définies rigoureusement au préalable. Dans l'ensemble, les candidates et candidats se sont bien appropriés ces notions.

Sujets Comme les années précédentes, par souci de transparence, et pour permettre à toutes les candidates et à tous les candidats de préparer cette épreuve de manière équitable, ce rapport de concours inclut en annexe l'intégralité des sujets posés cette année. [1] Soulignons que, présentés tels quels, les sujets sont de difficulté inégale. Pour certains on attendait une résolution complète ou presque quand pour d'autres arriver à la moitié constituait une bonne performance. Par ailleurs, certaines questions s'accompagnaient d'indications orales sans lesquelles la résolution aurait été particulièrement ardue dans le temps imparti.

Critères d'évaluation et recommendations aux candidats Le but des épreuves orales est d'évaluer un ensemble de compétences que l'on espère révélateur d'une aptitude future à faire de la recherche : s'approprier des notions nouvelles, réfléchir en autonomie, expliquer ses intuitions, identifier les difficultés et proposer une ou des approches pertinentes à un problème non trivial, exposer sa solution de façon claire et synthétique, etc.

A ce titre, nous encourageons les candidates et candidats à expliquer à voix haute les approches qu'ils ont tentées lorsqu'ils butent sur une question, afin d'engager un dialogue avec l'examinateur. Une fois la solution à une question trouvée, on attend de la candidate ou candidat qu'il ou elle sache l'exposer clairement, de préférence oralement mais en s'aidant du tableau si nécessaire (par exemple pour écrire un morceau de pseudocode). Certains candidates ou candidats présentent leur solution de façon trop vague ou incomplète quand d'autres tombent dans l'excès inverse en écrivant l'intégralité de la solution au tableau comme s'il s'agissait d'une épreuve écrite (ce qui n'est pas sanctionné en tant que tel mais fait perdre beaucoup de temps). En revanche, si l'examinateur demande une clarification écrite sur un point précis, la candidate ou candidat ne doit pas y rechigner. Un point important de l'évaluation porte sur la capacité de la candidate ou du candidat à réagir aux indications de l'examinateur pour continuer à avancer. Rester bloqué dans une voie sans issue en ignorant les indications est fortement pénalisant.

Un autre point délicat est le niveau de formalisme attendu. Parfois, la définition de l'objet d'étude nécessitait un niveau élevé de formalisme initial pour plus de précision, mais on attendait de la candidate ou candidat qu'il ou elle sache s'affranchir de celui-ci une fois la définition comprise. Dans d'autres cas, le sujet revêtait un caractère intrinsèquement formel, et l'on attendait à l'inverse que la candidate ou candidat sache jongler avec ce formalisme durant l'oral, ce qui n'a hélas pas toujours été le cas. Dans un cas comme dans l'autre, une réponse informelle n'est bien sûr pas pénalisée dès l'instant où le candidat sait préciser sa réponse sur demande de l'examinateur.

Le jury a pu constater que le programme est dans l'ensemble bien maîtrisé. Des lacunes ont cependant été constatées sur le thème des langages formels et automates finis. Alors que la clôture par intersection des langages réguliers est au programme, certains candidats ne connaissent pas la construction de l'automate produit ni ne savent la retrouver avec indications. Par ailleurs, le jury estime que les candidates et candidats doivent être capables d'exhiber un langage non-régulier et de prouver sa non-régularité.

Annexe. La suite de ce document présente l'intégralité des sujets qui ont été posés, sous la forme des feuilles distribuées aux candidates et candidats.

1. Des exemples de corrigés de ces mêmes exercices, donnés à titre indicatif, sont disponibles depuis `https://diplome.di.ens.fr/informatique-ens/`.

B1 – Graphe coucou

Soit X un ensemble de cardinalité n. On appelle « objets » les éléments de X. Soit V un ensemble de cardinalité $m \geq n$. On appelle « cases » les éléments de V. On suppose qu'on dispose de deux fonctions $H_1 : X \to V$ et $H_2 : X \to V$.

Une *assignation* des objets X dans les cases V est une fonction $f : X \to V$. On dit alors que x est assigné dans la case $f(x)$. L'assignation f est *valide* si elle respecte les contraintes suivantes :

1. l'objet $x \in X$ ne peut être assigné que dans la case $H_1(x)$ ou dans la case $H_2(x)$;
2. chaque case reçoit au plus un objet.

On dit qu'une configuration (X, V, H_1, H_2) est *acceptable* s'il existe une assignation valide.

Question 0 (Préliminaires).

a. Donner un exemple de configuration acceptable, et de configuration non acceptable.

b. Proposer un algorithme naïf pour tester si une configuration est acceptable. Quelle est sa complexité ?

Question 1. Le *graphe coucou* (où « coucou » fait référence à l'oiseau) $G(X, V, H_1, H_2)$ est le graphe non-orienté dont l'ensemble des sommets est V, et dont les arêtes sont $\{H_1(x), H_2(x)\}$ pour $x \in X$. On note que ce graphe peut contenir plusieurs arêtes entre deux sommets fixés (lorsqu'il existe $x \neq x'$ tels que $\{H_1(x), H_2(x)\} = \{H_1(x'), H_2(x')\}$), et qu'il peut contenir des boucles (lorsqu'il existe x tel que $H_1(x) = H_2(x)$). On continuera néanmoins d'utiliser le terme « graphe » pour ces objets, et c'est systématiquement dans ce sens que le terme graphe est utilisé dans la suite.

Montrer que l'énoncé « (X, V, H_1, H_2) est acceptable » peut être reformulé de manière équivalente sous la forme : « il existe une manière d'orienter les arêtes de $G(X, V, H_1, H_2)$ de telle sorte que le graphe orienté qu'on obtient satisfait une condition simple ».

Question 2. On dit qu'un graphe connexe $G = (V, E)$ est *complexe* si $|E| > |V|$ (où $|E|$ désigne les arêtes comptées avec leur multiplicité).

a. Montrer qu'un graphe connexe $G = (V, E)$ est un arbre (c'est-à-dire : connexe et acyclique) si et seulement si $|E| = |V| - 1$.

b. Montrer qu'un graphe connexe est non complexe si et seulement si il contient au plus un cycle.

c. Montrer que si $G(X, V, H_1, H_2)$ ne contient pas de composante connexe complexe, la configuration associée est acceptable.

d. Montrer la réciproque : (X, V, H_1, H_2) est acceptable si et seulement si $G(X, V, H_1, H_2)$ ne contient pas de composante connexe complexe.

Question 3. Proposer un algorithme en temps linéaire qui prend en entrée un graphe coucou G, et détermine si G est acceptable.

Question 4. Si une configuration est acceptable, quel est le nombre d'assignations valides ?

Question 5. Supposons qu'on connaît une assignation valide f pour la configuration (X, V, H_1, H_2). On souhaite ajouter un nouvel élément $x \notin X$ dans X ; on suppose que $H_1(x)$ et $H_2(x)$ sont déjà définis. Pour cela, on utilise l'algorithme suivant. Soit $x_0 = x$. On insère x_0 dans la case $H_1(x_0)$. Si la case est libre, on a fini. Sinon, soit x_1 l'élément tel que $f(x_1) = H_1(x_0)$. Supposons $f(x_1) = H_1(x_1)$ (le cas $f(x_1) = H_2(x_1)$ est similaire). On insère toujours x_0 dans $H_1(x_0)$, mais on déplace l'élément x_1 de la case $H_1(x_1)$ vers la case $H_2(x_1)$. Si cette case est libre, on a fini. Sinon, on continue d'itérer le même processus. (Cette manière de « voler » les cases déjà occupées est ce qui donne le nom « coucou » à l'algorithme.)

Montrer que si $(X \cup \{x\}, V, H_1, H_2)$ est acceptable, l'algorithme précédent termine. Borner sa complexité.

Question 6. On s'intéresse maintenant au cas où chaque case peut recevoir jusqu'à k objets. On dit qu'une configuration est k-acceptable s'il existe une assignation telle que chaque case reçoit au plus k objets.

a. Traduire la propriété d'être k-acceptable en termes d'orientabilité des arêtes du graphe coucou associé, dans l'esprit de la question 1.

b. Pour $G = (V, E)$ un graphe et $V' \subseteq V$, le sous-graphe *induit* par V' est le graphe $(V', \{\{x, y\} : \{x, y\} \in E \text{ et } x, y \in V'\})$. Montrer que G est k-acceptable si et seulement si tout sous-graphe induit (V', E') de G vérifie $|E'| \leq k|V'|$.
Indication pour le sens difficile : Pour une assignation donnée, on peut définir le *surplus* d'une case qui reçoit t objets comme étant $\max(0, t - k)$. Le *surplus* de l'assignation est la somme des surplus de toutes les cases. Si la configuration n'est pas k-acceptable, on peut considérer une assignation dont le surplus est minimal (nécessairement ≥ 1). Soit Δ l'ensemble des sommets de surplus strictement positif pour cette assignation. Raisonner sur $\Delta' \supseteq \Delta$ l'ensemble des sommets depuis lesquels il existe un chemin orienté vers un sommet de Δ (ou dans l'autre sens, suivant le sens dans lequel le candidat a choisi d'orienter les arêtes à la question 1).

c. Montrer que dans le cas $k = 1$, cette condition est bien équivalente à celle de la question 2c.

Question 7. On revient au cas $k = 1$. Cette fois, on autorise à stocker un objet x en partie dans $H_1(x)$, et en partie dans $H_2(x)$. Plus précisément, une $\mathbb{Q}$-*assignation* des objets X dans les cases V est une fonction $f : X \to ([0, 1] \cap \mathbb{Q})^m$. On note $f(x)[i]$ la i-ième composante de $f(x)$. La $\mathbb{Q}$-assignation est *valide* si elle respecte les contraintes suivantes :

1. l'objet $x \in X$ est assigné entièrement dans $H_1(x)$ et $H_2(x)$, formellement : $\sum_{v \in V} f(x)[v] = 1$, mais $\forall v \notin \{H_1(x), H_2(x)\}, f(x)[v] = 0$;
2. chaque case contient au plus l'équivalent d'un objet, formellement : $\forall v, \sum_{x \in X} f(x)[v] \leq 1$.

Une configuration est dite $\mathbb{Q}$-acceptable s'il existe une $\mathbb{Q}$-assignation valide.

a. Soit une configuraiton (X, V, H_1, H_2) et le graphe coucou associé G. Soit G^k le graphe obtenu en multipliant la multiplicité de toutes les arêtes de G par k. Montrer que G est $\mathbb{Q}$-acceptable si et seulement si il existe un certain k tel que G^k est k-acceptable.

b. Montrer qu'une configuration (X, V, H_1, H_2) est $\mathbb{Q}$-acceptable si et seulement si elle est acceptable.

B2 – Théorème de Gale-Ryser

On note $[\![a, b]\!]$ l'ensemble d'entiers $\{a, a+1, \ldots, b\}$. Soit $S_n = [\![0, n]\!]^n$ les séquences de longueur n dans $[\![0, n]\!]$. Pour $a \in S_n$, on note $a = (a_1, \ldots, a_n)$ les coordonnées de a.

Étant donnés $a, b \in S_n$ tels que $\sum a_i = \sum b_i$, on dit que b *majorise* a, noté $a \preceq b$, si et seulement si :

$$\forall k \in [\![1, n]\!], \quad \sum_{i=1}^{k} a_i \leq \sum_{i=1}^{k} b_i.$$

On rappelle qu'un ordre est une relation réflexive, transitive et anti-symétrique.

On note $\mathcal{M}_n = \{0, 1\}^{n \times n}$ l'ensemble des matrices $n \times n$ à valeur dans $\{0, 1\}$.

Question 0.

a. Montrer que $\preceq$ est un ordre sur S_n.

b. Donner un exemple de deux séquences $a, b \in S_n$ avec $\sum a_i = \sum b_i$, telles que a et b sont incomparables pour $\preceq$.

Question 1. Soit $a \in S_n$. On appelle *conjugué* de a la séquence $a^* \in S_n$ définie par $a^*_i = |\{k \in [\![1, n]\!] : a_k \geq i\}|$.

a. Montrer que a^* est triée par ordre décroissant, et que $\sum_{i \in [\![1,n]\!]} a_i = \sum_{i \in [\![1,n]\!]} a^*_i$.

b. On suppose a triée par ordre décroissant. On définit la matrice $M(a) = (m(a)_{i,j})_{i,j \in [\![1,n]\!]} \in \mathcal{M}_n$ associée à a de la manière suivante : la i-ième ligne de $M(a)$ contient des 1 sur les a_i premières colonnes, et des 0 ensuite. Quelle relation existe entre la matrice associée à a, et la matrice associée à a^* ?

c. Montrer que pour tout $a, b \in S_n$, $a^* = b^*$ si et seulement si il existe une permutation π de $[\![1, n]\!]$ telle que $a_i = b_{\pi(i)}$.

d. Montrer que a est trié par ordre décroissant si et seulement si $a^{**} = a$.

Question 2. Pour $M = (m_{i,j})_{i,j \in [\![1,n]\!]} \in \mathcal{M}_n$, on note $r(M) \in S_n$ la séquence des sommes des lignes de M, *i.e.* $r(M)_i = \sum_{j \in [\![1,n]\!]} m_{i,j}$. On note de même $c(M) \in S_n$ la séquences des sommes des colonnes de M.

Soient $r, c \in S_n$ avec $\sum r_i = \sum c_i$. On dit que (r, c) est *compatible* si et seulement si il existe une matrice $M \in \mathcal{M}_n$ telle que $r = r(M)$ et $c = c(M)$.

a. Donner un exemple de deux séquences $r, c \in S_n$ avec $\sum r_i = \sum c_i$, telles que (r, c) est incompatible.

b. Soient r', c' les séquences définies par $r'_i = r_{\pi(i)}$ et $c'_i = c_{\rho(i)}$, pour des permutations π, ρ quelconques. Montrer que (r, c) est compatible si et seulement si (r', c') est compatible.

c. Suite à la question précédente, on peut donc supposer sans perte de généralité que r et c sont triées par ordre décroissant. Montrer que si (r, c) sont compatibles, alors $r \preceq c^*$.

Question 3. On souhaite montrer la réciproque. Soit $r, c \in S_n$ avec $\sum r_i = \sum c_i$ et $r \preceq c^*$. On considère l'algorithme suivant, qui tente de créer une matrices d'adjacence $n \times n$ dont les sommes des lignes et sommes des colonnes correspondent à (r, c). L'algorithme part de la matrice nulle, et parcourt les lignes dans l'ordre $1, \ldots, n$, et dans chaque ligne, parcourt les entrées dans l'ordre $1, \ldots, n$. À chaque entrée aux coordonnées (i, j), l'algorithme remplace 0 par 1 si et seulement si la somme de la ligne i reste inférieure à r_i, et la somme de la colonne j reste inférieure à c_j.

Montrer que la matrice produite par cet algorithme témoigne du fait que (r, c) est compatible.

Indication : on peut faire une récurrence sur le nombre de coordonnées non nulles dans r.

Question 4. Soient $a, b \in S_n$ avec $\sum a_i = \sum b_i$. Montrer $a^* \preceq b^* \Leftrightarrow b^{**} \preceq a^{**}$.
Indication : il est possible de le déduire des questions précédentes.

Question 5. Soit G un graphe (non-orienté) biparti : les sommets de G sont partitionnés en deux ensembles V, W, et toutes les arêtes sont de la forme $\{v, w\}$ avec $(v, w) \in V \times W$. On associe à G la séquence $d(v)_{v \in V}$ des degrés des sommets de V, et la séquence $d'(w)_{w \in W}$ des degrés des sommets de W.

Étant donné deux séquences finies d'entiers d, d' de longueur respectives n et m, proposer un algorithme qui détermine en temps polynomial s'il existe un graphe biparti avec $|V| = n$ et $|W| = m$, tel que d et d' sont les séquences des degrés de V et W.

Question 6. Soit Ω un ensemble, et $\mathcal{A}$ une partition de Ω. On dit que $S \subseteq \Omega$ est *transverse* si pour tout $A \in \mathcal{A}$, $S \cap A$ contient au plus un élément. On pose $\mathcal{A} = \{A_1, \ldots, A_n\}$ avec $a_i = |A_i|$ tel que $a_1 \geq \cdots \geq a_n$. Soit $r_1 \geq \cdots \geq r_m$ une séquence d'entiers dans $[\![0, n]\!]$.

Montrer qu'il existe une famille de transverses deux à deux disjoints de cardinalités $r_1, \ldots, r_m$ si et seulement si $(r_n^*, \ldots r_1^*) \preceq (a_n, \ldots, a_1)$. (La condition ne requiert pas $\sum_{i \in [\![1,n]\!]} a_i = \sum_{i \in [\![1,n]\!]} r_i^*$.)

B3 – Tri aveugle

Un serveur dispose d'un nombre non borné de blocs mémoire, indexés par $\mathbb{N}^*$. Le serveur n'effectue aucun calcul. D'un autre côté, un client effectue des calculs arbitraires, mais dispose seulement de deux blocs mémoire, plus des registres qui permettent de stocker des variables auxiliaires : compteurs, etc.

Les seules interactions possibles entre le client et le serveur sont de la forme suivante : le client envoie au serveur une requête $\mathsf{query}(t, a, b)$. Ensuite :

— si $t = 0$, le serveur renvoie le contenu du a-ième bloc (« accès en lecture »).

— si $t = 1$, le serveur remplace le contenu du a-ième bloc par b (« accès en écriture »).

Tri aveugle. Initialement, la mémoire du serveur contient une liste d'objets $X = (x_1, x_2, \dots, x_n)$. L'objet x_i est stocké dans le bloc i. Le client souhaite trier ces objets suivant un ordre $<$.

Pour cela, le client exécute un algorithme $\mathcal{A}$, qui inclut des instructions query. Soit (t_i, a_i, b_i) les arguments successivement pris par query au cours d'une exécution donnée de l'algorithme. On appelle $(a_1, a_2, \dots)$ la *séquence d'accès* de cette exécution.

Pour X et $<$ fixés, si $\mathcal{A}$ est probabiliste, il induit une distribution $\mathcal{D}_{\mathcal{A}}(X, <)$ sur les séquences d'accès : la probabilité de chaque séquence dépend des choix aléatoires effectués par l'algorithme. Si $\mathcal{A}$ n'est pas probabiliste, $\mathcal{D}_{\mathcal{A}}(X, <)$ est concentré en un point.

Un algorithme de tri est dit « aveugle » si $\mathcal{D}_{\mathcal{A}}(X, <) = \mathcal{D}'_{\mathcal{A}}(n)$ dépend uniquement de n. Informellement : un tri est aveugle si les accès mémoire de l'algorithme ne contiennent aucune information sur l'ordre dans lequel les éléments sont triés.

Question 1.

a. Nommer un algorithme de tri en $O(n \log n)$. Cet algorithme est-il aveugle ?

b. Proposer un algorithme de tri aveugle pour trier deux objets. Proposer un algorithme de tri aveugle pour n objets en temps $O(n^2)$.

Question 2. Un algorithme de *tri aléatoire* est un algorithme qui choisit un ordre $<_\$$ uniformément aléatoirement, et trie suivant cet ordre. Un algorithme de tri aléatoire aveugle est un algorithme de tri aléatoire qui est aveugle au même sens que plus haut.

a. Soit G un groupe, et $g \in G$ fixé. Soit u un élément tiré uniformémement dans G. Montrer que la distribution de gu est uniforme.

b. Étant donné un algorithme de tri aléatoire aveugle en temps T, proposer un algorithme de tri aveugle en temps $T + O(n \log n)$.

Question 3. Dans toute la suite, on suppose que n est une puissance de 2. Soit $z = O(\log n)$ un entier qui divise n. Dans tout l'exercice, on suppose qu'on sait tirer une fonction h uniformément aléatoirement parmi les fonctions $X \to \{0,1\}^z$. Tirer un tel h et l'évaluer se fait en temps constant.

Pour réaliser un tri aléatoire, on souhaite trier X suivant l'ordre induit par $h : x <_h y \Leftrightarrow h(x) \prec h(y)$, où $\prec$ est l'ordre lexicographique. Pour cela, on crée sur le disque n/z « paquets », pour l'instant de n blocs chacun. Initialement, le i-ième paquet contient les objets $x_{z(i-1)+1}, x_{z(i-1)+2}, \dots, x_{z(i-1)+z}$. On suppose qu'on dispose d'un algorithme Compaction qui prend en entrée deux paquets et une fonction $f : X \to \{0,1\}$, et place les objets dont l'image par f est 0 dans le premier paquet, et les objets dont l'image par f est 1 dans le second paquet.

En faisant $O(n \log n)$ appels à Compaction, proposer un algorithme qui permet d'obtenir des paquets triés suivant $<_h$, c'est-à-dire : les éléments du paquet i sont tous inférieurs pour $<_h$ aux éléments du paquet i+1. On ne se préoccupe pas d'être aveugle.

Question 4. Montrer que pour tout paquet construit par COMPACTION au cours de l'algorithme précédent, il existe un entier i tel que la probabilité que le paquet reçoive plus de $6z$ objets est :

$$g(z,t) = \sum_{k=6z+1}^{zt} C_{zt}^{k} t^{-k} (1-t^{-1})^{zt-k} \qquad \text{où } t = 2^i.$$

Question 5. Le but de cette question est de montrer que g décroit exponentiellement avec z.

a. Montrer :
$$C_n^k \leq \frac{n^k}{k!} \leq \left(\frac{ne}{k}\right)^k.$$

b. En déduire $g(z,t) \leq 2^{-6z}$.
Indication : montrer $g(z,t) \leq \sum_{k=6z+1}^{\infty} 2^{-k}$.

Question 6. Suite à la question précédente, on suppose dorénavant que tous le paquets traités par COMPACTION contiennent au plus $6z$ objets. On fixe la taille des paquets à $6z$ exactement, quitte à remplir avec des objets factices.

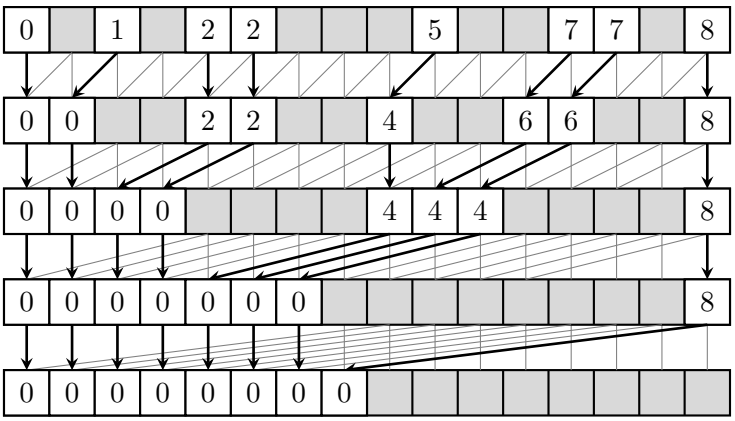
a. Au terme de l'algorithme de la question 3, on obtient n/z paquets triés entre eux contenant des objets réels (ceux de X) et des objets factices. On souhaite créer un algorithme qui extrait la liste triée X de ces paquets, pour réaliser un tri aléatoire aveugle. Cet algorithme d'extraction peut-il se permettre de révéler (via ses accès mémoire) combien d'objets réels sont contenus dans chaque paquet ?
Indication : montrer que la distribution des nombres d'objets par paquet ne dépend pas de l'ordre $<_h$.

b. Proposer un algorithme d'extraction au sens de la question précédente.

Question 7. Proposer un algorithme de tri aveugle en temps $O(n \log^c n)$, pour c une constante.

Question 8 (bonus). On considère des algorithmes de compaction en un sens plus général. Un algorithme de compaction en ce nouveau sens prend en entrée une séquence d'objet X et $f : X \to \{0,1\}$. Il renvoie la liste triée suivant f (noter qu'il n'y a plus de notion de paquets).

Proposer un algorithme de compaction aveugle en temps $O(n \log n)$, en s'inspirant du dessin ci-dessous.



B4 – Circuit universel

Pour un graphe orienté G, on note $\Delta^+(G)$ le maximum des degrés sortants des sommets de G, $\Delta^-(G)$ le maximum des degrés entrants, et $\Delta^*(G) = \max(\Delta^+(G), \Delta^-(G))$. On note $\mathsf{GOA}_d(n)$ l'ensemble des graphes orientés acycliques (GOA) à n sommets tels que $\Delta^* \leq d$. (Pour rappel, un graphe acyclique est un graphe tel qu'il n'existe pas de chemin non trivial d'un sommet à lui-même.)

Ordre topologique. On dit qu'une séquence ordonnée $(v_1, \ldots, v_n)$ des sommets d'un GOA est un *ordre topologique* si pour tous v_i, v_j tels qu'il existe un chemin orienté $v_i \to v_j$, on a $i \leq j$.

Plongement. Un *plongement* d'un GOA $G = (V, E)$ dans un GOA $G' = (V', E')$ est une paire (f, g), où $f : V \to V'$ est une injection, et g envoie $(u, v) \in E$ sur un chemin orienté de $f(u)$ vers $f(v)$, de telle sorte que pour $e \neq e'$, les chemins $g(e)$ et $g(e')$ sont disjoints (pas d'arête commune).

Graphe universel. Soit $G_{\mathrm{u}} = (V_{\mathrm{u}}, E_{\mathrm{u}})$ un GOA muni d'une séquence de sommets distingués $P = (p_1, \ldots, p_n) \in V_{\mathrm{u}}^n$ deux à deux distincts, appelés « pôles ». On dit que G_{u} est *universel* pour $\mathsf{GOA}_d(n)$ si pour tout $G \in \mathsf{GOA}_d(n)$ muni d'un ordre topologique $(v_1, v_2, \ldots, v_n)$, il existe un plongement de G dans G_{u} qui envoie v_i sur p_i.

Question 1.

a. Soit $k < n$. Donner un GOA universel pour $\mathsf{GOA}_k(n)$.

b. Donner un GOA G_{u} universel pour $\mathsf{GOA}_1(4)$ tel que $\Delta^*(G_{\mathrm{u}}) = 2$.

Question 2.

a. Soit G un graphe non-orienté dont les sommets sont de degré au plus 2. On suppose que G est *biparti*, c'est-à-dire qu'il existe une partition des arêtes en V_1, V_2 telle que toutes les arêtes sont de la forme $\{v_1, v_2\}$ pour $v_i \in V_i$. Montrer qu'on peut colorier les arêtes de G en deux couleurs, de sorte que toute paire d'arêtes avec un sommet commun a des couleurs distinctes.

b. Soit $G = (V, E) \in \mathsf{GOA}_2(n)$. Montrer qu'il existe une partition des arêtes $E = E_1 \cup E_2$ telle que $(V, E_i) \in \mathsf{GOA}_1(n)$ pour $i \in \{1, 2\}$.

Question 3. Dans cette question, on suppose qu'on connaît un graphe G universel pour $\mathsf{GOA}_1(n-1)$ avec $\Delta^*(G) \leq 2$, tel que les pôles sont de degré entrant et sortant au plus 1.

a. Donner une construction d'un graphe universel G_{u} pour $\mathsf{GOA}_2(n)$ avec $\Delta^*(G_{\mathrm{u}}) \leq 2$, qui contient deux copies de G comme sous-graphes.

b. En déduire une construction d'un graphe universel G'_{u} pour $\mathsf{GOA}_1(2n)$ avec $\Delta^*(G'_{\mathrm{u}}) \leq 2$, qui contient encore deux copies de G comme sous-graphes.

Question 4. Montrer qu'il existe un graphe universel G_{u} pour $\mathsf{GOA}_2(n)$ avec $\Delta^*(G_{\mathrm{u}}) \leq 2$. Borner son nombre de sommets.

Circuit. On appelle *circuit* à n entrées et m portes un GOA $G = (V, E)$ dont les sommets sont partitionnés comme suit :

- n sommets « entrée » $(s_1, \ldots, s_n)$, qui ont degré entrant 0 et degré sortant au plus 1 ;
- un sommet « sortie » t, de degré entrant 1 et degré sortant 0 ;
- m autres sommets appelés « portes », qui sont de deux types : NAND, de degré entrant 2 et sortant 1, et COPIE, de degré entrant 1 et sortant 2.

Soit $\mathcal{C}_{n,m}$ l'ensemble des circuits avec n entrées et m portes. Étant donné $C \in \mathcal{C}_{n,m}$, et $x = (x_1, \ldots, x_n) \in \{0,1\}^n$, on associe par induction une valeur booléenne à chaque arête de C : les arêtes provenant des entrées s_i prennent la valeur x_i, les arêtes sortant des portes COPIE prennent la valeur de l'arête en entrée, et les arêtes en sortie des portes NAND prennent la valeur $\neg(a \wedge b)$, où a, b sont les valeurs des arêtes en entrée. On définit $C(x)$ comme la valeur de l'arête menant au sommet de sortie t.

Question 5. Montrer que pour tout n, il existe m tel que pour toute fonction booléenne $f : \{0,1\}^n \to \{0,1\}$, il existe un circuit $C \in \mathcal{C}_{n,m}$ satisfaisant $C(x) = f(x)$ pour tout x.

Circuit universel. On dit qu'un circuit $C_{\mathrm{u}} \in \mathcal{C}_{n_{\mathrm{u}},m_{\mathrm{u}}}$ est *universel* pour $\mathcal{C}_{n,m}$ avec $n \leq n_{\mathrm{u}}$ si pour tout $C \in \mathcal{C}_{n,m}$, il existe $c \in \{0,1\}^{n_{\mathrm{u}}-n}$ tel que pour tout x, $C(x) = C_{\mathrm{u}}(x, c)$.

Question 6. Soit $m' = m + n$. Montrer qu'il existe un circuit universel pour $\mathcal{C}_{n,m}$ avec un nombre de portes $O(m' \log m')$.

Question 7. On souhaite généraliser le résultat de la question 2b. Soit $G = (V, E) \in \mathsf{GOA}_d(n)$. Montrer qu'il existe une partition des arêtes $E = E_1 \cup \cdots \cup E_d$ telle que $(V, E_i) \in \mathsf{GOA}_1(n)$ pour tout i.

Pour cela, le théorème de Kőnig peut être utile. Soit $G = (V, E)$ un graphe non-orienté biparti. Une *couverture* de G est un sous-ensemble R de sommets tel que toute arête est adjacente à au moins un sommet de R. Un *couplage* de G est un sous-ensemble C d'arêtes tel qu'aucune paire d'arêtes de C n'a de sommet en commun. Le théorème de Kőnig dit que la taille minimale d'une couverture, et la taille maximale d'un couplage, sont égales.

B5 – Graphes triangulés

Pour S un ensemble, on note $\mathcal{P}_2(S)$ l'ensemble des sous-ensembles de cardinalité 2 de S. Un *graphe* G est une paire (V, E) avec $E \subseteq \mathcal{P}_2(V)$. Étant donné $S \subseteq V$, on note $G[S] = (S, E \cap \mathcal{P}_2(S))$ le *sous-graphe de G induit par S*. Un graphe est *cyclique* de taille $n \geq 3$ s'il est isomorphe à $(\mathbb{Z}/n\mathbb{Z}, \{\{i, i+1\} : i \in \mathbb{Z}/n\mathbb{Z}\})$. Un graphe est *triangulé* si tous ses sous-graphes induits cycliques (s'il en existe) sont de taille 3.

Question 0. Donner un example de graphe triangulé, et de graphe non triangulé.

Question 1. Montrer qu'un graphe est triangulé si et seulement si pour toute suite de sommets formant un cycle de longueur au moins 4, il existe une *corde*, c'est-à-dire une arête qui lie deux sommets non consécutifs du cycle.

Question 2. Étant donné deux sommets a, b non adjacents dans un graphe $G = (V, E)$, un *séparateur* de a et b est un ensemble de sommets $S \subseteq V \setminus \{a, b\}$ tel que lorsqu'on enlève les sommets S de G, a et b se retrouvent dans des composantes connexes distinctes (formellement, « enlever » les sommets S signifie qu'on se place dans le graphe $G[V \setminus S]$). Un séparateur est dit *minimal* s'il est minimal pour l'ordre d'inclusion.

a. Soit G un graphe tel que tout séparateur minimal induit un graphe complet, c'est-à-dire : $G[S] = (S, \mathcal{P}_2(S))$. Montrer que G est triangulé.

b. Montrer la réciproque : si G est triangulé, alors tout séparateur minimal induit un graphe complet.

Question 3. Soit v un sommet d'un graphe $G = (V, E)$. On note $\mathsf{adj}_G(v)$ l'ensemble des sommets adjacents à v dans G. On dit que v est *simpliciel* si $\mathsf{adj}_G(v)$ induit un graphe complet.

Montrer que tout graphe triangulé qui n'est pas un graphe complet contient au moins deux sommets simpliciels non adjacents.

Indication : considérer un séparateur S pour deux sommets a, b non adjacents, et chercher un sommet simpliciel dans A (resp. B), la composante connexe de a (resp. b) dans $G[V \setminus S]$.

Question 4. Soit $G = (V, E)$ un graphe à n sommets. Soit $\sigma = (v_1, \ldots, v_n)$ un ordre sur les sommets. On dit que σ est un *schéma d'élimination* si pour tout i, v_i est simpliciel dans $G[\{v_i, \ldots, v_n\}]$.

a. Montrer qu'un graphe est triangulé si et seulement si il admet un schéma d'élimination.

b. Proposer un algorithme en temps polynomial pour tester si un graphe est triangulé.

Question 5. Le *nombre chromatique* $\chi(G)$ d'un graphe $G = (V, E)$ est le plus petit entier k tel que G admet un k-coloriage, c'est-à-dire un coloriage des sommets en k couleurs tel que toute paire de sommets adjacents a des couleurs distinctes. Le *nombre de clique* $\omega(G)$ de G est le nombre de sommets de la plus grande clique de R, c'est-à-dire le plus grand sous-graphe induit complet.

a. Montrer $\chi(G) \geq \omega(G)$.

b. Soit G un graphe triangulé, et S un séparateur. Soient $A_1, \ldots A_k$ les composantes connexes de $G[V \setminus S]$. Montrer :

$$\chi(G) = \max_i \chi(G[S \cup A_i]).$$

c. Montrer que pour un graphe triangulé, $\chi(G) = \omega(G)$.

Question 6. Soit $C = (V, E)$ un graphe cyclique de taille n. Une *triangulation* de C est un graphe triangulé (V, E') avec $E' \supseteq E$.

a. Montrer qu'une triangulation de C est minimale, au sens de l'ordre induit par l'inclusion des arêtes, si et seulement si elle est minimale pour l'ordre induit par le nombre d'arêtes.

b. Donner une formule explicite pour le nombre de triangulations minimales de C.

C1 – Automates et monoïdes

Un *semigroupe* est un couple $(E, \cdot)$ où E est un ensemble et $\cdot$ une *loi de multiplication interne associative*. On rappelle :

— Une (loi de) multiplication (interne) est dite *associative* si $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.

— Un élément neutre est un élément e tel que pour tout $x \in E$, on a $x \cdot e = e \cdot x = x$.

Un *monoïde* est un semigroupe qui dispose d'un élément neutre. Dans la suite, on identifiera le semigroupe (monoïde) et son ensemble. Un semigroupe (monoïde) est dit fini s'il dispose d'un ensemble fini d'éléments.

Question 0. Préliminaires

a Montrez qu'un monoïde possède un unique élément neutre.

b Pour A un alphabet fini et $\cdot$ la concaténation, montrez que $(A^*, \cdot)$ est un monoïde.

Question 1. Monoïdes des transformations Soit Q un ensemble fini. Montrez que l'ensemble des fonctions de $Q \to Q$ est un monoïde. On appelle ce monoïde le *monoïde des transformations* de Q.

Un monoïde N est un *sous-monoïde* d'un monoïde M s'il existe un morphisme injectif de N vers M. Montrez que tout monoïde fini $M = (E, \cdot)$ est un sous-monoïde d'un monoïde des transformations.

Question 2. Monoïdes des transitions Soit $\mathcal{A} = (Q, A, \delta, i, F)$ un automate déterministe fini avec Q les états, A l'alphabet, $\delta \colon Q \times A \to Q$ les transitions, $i \in Q$ l'état initial et $F \subseteq Q$ les états finaux. Dans la suite on note $q \cdot a = \delta(q, a)$ et pour $u_1 \cdots u_k \in A^*$ on note $q \cdot u := (\cdots (q \cdot u_1) \cdot u_2 \cdots) \cdot u_k$.

On appelle *fonctions de transitions* de l'automate $\mathcal{A}$, les fonctions $f \colon Q \to Q$ tel qu'il existe un mot $u \in A^*$ avec $f(q) = q \cdot u$ pour tout $q \in Q$.

a. Montrez que l'ensemble des fonctions de transition de $\mathcal{A}$ forme un monoïde fini. On l'appelle le *monoïde des transitions* de $\mathcal{A}$.

b. Quelle est la complexité de vérifier qu'une fonction $f \colon Q \to Q$ est dans le monoïde des transitions de $\mathcal{A}$.

c. En déduire un algorithme qui énumère les fonctions de transition de l'automate $\mathcal{A}$. Donnez sa complexité en fonction de $|Q|$, le nombre d'états de $\mathcal{A}$.

d. Proposez un algorithme polynomial en $|Q|$ et N où N est le nombre de fonctions de transition de $\mathcal{A}$.

e. Un langage $L \subseteq A^*$ est *reconnu* par un monoïde M s'il existe un morphisme $\varphi \colon A^* \to M$ tel que $L = \varphi^{-1}(\varphi(L))$. Montrez que le monoïde des transitions reconnait le langage calculé par l'automate $\mathcal{A}$.

Question 3. Reconnaissabilité et régularité Un langage est *reconnaissable* s'il est reconnu par un monoïde fini. Montrez qu'un langage est reconnaissable si et seulement s'il est calculable par un automate fini.

Question 4. Monoïdes des relations Soit Q un ensemble fini. On rappelle qu'une *relation* sur Q est un sous-ensemble de $Q \times Q$. Pour R_1 et R_2 deux relations sur Q, on note $R_1 \circ R_2$ leur composition, c'est-à-dire l'ensemble $\{(x, y) \in Q \times Q \mid \exists z, (x, z) \in R_1 \text{ et } (z, y) \in R_2\}$.

a. Montrez que l'ensemble des relations sur un ensemble fini, muni de la composition, est un monoïde.

b. Donnez la taille du monoïde en fonction de la taille de l'ensemble Q.

Question 5. Automates non déterministe Soit $\mathcal{A} = (Q, A, \delta, I, F)$ un automate non déterministe fini avec Q les états, A l'alphabet, $\delta \colon Q \times A \times Q$ les transitions, $I \in Q$ les états initiaux et $F \subseteq Q$ les états finaux. Dans la suite on note $q \cdot a = \delta(q, a) \subseteq Q$ et pour $u_1 \cdots u_k \in A^*$ on note $q \cdot u := (\cdots (q \cdot u_1) \cdot u_2 \cdots) \cdot u_k$.

On appelle *relations de transition* de l'automate $\mathcal{A}$, les relations R de Q tel qu'il existe un mot $u \in A^*$ avec $(q, q \cdot u) \in R$ pour tout $q \in Q$.

a. Montrez que l'ensemble des relations d'un automate forme un monoïde fini. On l'appelle le *monoïde des relations* de l'automate.

b. Montrez que le monoïde des relations reconnait le langage calculé par l'automate $\mathcal{A}$.

c. Donnez rapidement un algorithme pour calculer le monoïde des relations d'un automate et indiquez sa complexité.

Question 6. Monoïde syntaxique et borne inférieur de complexité Le *monoïde syntaxique* est le plus petit monoïde qui reconnaît le langage régulier L. Soit L un langage régulier dont l'automate et dont le monoïde syntaxique est de taille n. Montrer que tout automate non déterministe calculant L est de taille au moins $\sqrt{\log n}$.

C2 – Automates et ordres partiels

Soit $\mathcal{A} = (Q, A, \delta, I, F)$ un automate fini (non nécessairement déterministe). Pour $q \in Q$ et $u \in A^*$ on note $q \cdot u$ l'ensemble des états accessibles depuis q en lisant u. Si $\mathcal{A}$ est déterministe, alors on note également $q \cdot u$ (l'unique) état accessible depuis q en lisant u.

L'automate $\mathcal{A}$ est *partiellement ordonné* si tous ses cycles sont de taille au plus un. Formellement, si pour tout $q, q' \in Q$ et $u, v \in A^*$, tel que $q' \in q \cdot u$ et $q \in q' \cdot v$, alors $q = q'$.

Question 0.

a. Justifiez le nom *partiellement ordonné* pour ces automates.

b. Montrez que le langage $\{abaa, baaa, abb\}$ est calculable par un automate déterministe partiellement ordonné.

c. Proposez un exemple de langage non fini calculable par un automate déterministe partiellement ordonné.

d. Montrez que le langage sur l'alphabet $A = \{a, b, c\}$, $K = A^*ac^*aA^*$ est calculable par un automate non déterministe partiellement ordonné.

Question 1. Clôture par les opérations booléennes. Montrez l'intersection et l'union de deux langages calculables par des automates partiellement ordonnés sont également calculables par des automates partiellement ordonnés.

Est-ce que les opérations d'union et d'intersection sont toujours vérifiées quand on considère des automates partiellement ordonnés déterministes ?

Question 2. Les langages finis. Un automate $\mathcal{A} = (Q, A, \delta, I, F)$ est *strictement partiellement ordonné* s'il est partiellement ordonné et si pour tout etat q et lettre $a \in A$, on a $q \notin \delta(q, a)$.

a. Montrer qu'un langage est calculable par un automate strictement partiellement ordonné si et seulement s'il est fini.

b. Illustrer par un exemple que la taille de l'automate d'un langage fini peut être beaucoup plus petit que la taille du langage.

Question 3. Langages clos par sur-mots. Soit $u = a_1 \cdots a_n \in A^*$.
On note $\uparrow u$ le langage $A^*a_1A^*a_2A^* \cdots a_nA^*$. Un langage $L \subseteq A^*$ est dit *clos par sur-mots* si pour tout mot $u \in L$ on a $\uparrow u \subseteq L$. On souhaite montrer qu'un langage clos par sur-mots est régulier et calculable par un automate déterministe partiellement ordonné. On admettra dans la suite le résultat suivant.

Lemme (Higman).
Soit $(w_i)_{i\in\mathbb{N}}$ une suite infinie de mots sur l'alphabet A. Il existe deux entiers i et j tels que w_i est un sur-mot de w_j.

a. Montrez que l'intersection de deux langages clos par sur-mots est clos par sur-mots.

b. Soit $F \subset A^*$ un ensemble fini de mots. Montrer que $\bigcup_{u\in F} \uparrow u$ est calculable par un automate déterministe partiellement ordonné.

c. Montrez qu'un langage L est clos par sur-mots si et seulement s'il existe un ensemble fini F tel que $L = \bigcup_{u\in F} \uparrow u$ et conclure.

Question 4. Monômes et automates non-déterministe partiellement ordonnés. On appelle *monôme* un langage régulier de la forme $B_0^* a_0 B_1^* a_1 \cdots B_n^*$ avec $B_i \subseteq A$ et $a_i \in A$ pour tout i. On utilise la convention $\varnothing^* = \{\epsilon\}$ avec ϵ le mot vide.

a. Montrez qu'un langage régulier est calculable par un automate non-déterministe partiellement ordonné si et seulement s'il est égal à une union finie de monômes.

b. Montrez que l'intersection de deux monômes est une union finie de monômes.

c. Montrez que le langage $T = (ab)^*$ n'est pas calculable par un automate non-déterministe partiellement ordonné.

d. Conclure qu'il existe un langage L tel que L est calculable par un automate non-déterministe partiellement ordonné mais pas L^c.

C3 – Langages réguliers de Delphes

Soit A et B deux alphabets et $u = u_1 \cdots u_n \in A^n$ et $v = v_1 \cdots v_n \in B^n$. On note $u * v$ le mot $(u_1, v_1) \cdots (u_n, v_n)$ de $(A \times B)^n$. Soit $\mathcal{A} = (Q, A \times B, \delta, I, F)$ un automate fini (non nécessairement déterministe) on note $L(\mathcal{A})$ le langage reconnu par $\mathcal{A}$.

Un *oracle* pour $\mathcal{A}$ est un ensemble $O \subseteq B^*$ tel $|O \cap B^n| = 1$ pour tout entier $n \in \mathbb{N}$. On note $\mathcal{A}(O)$ le langage des mots reconnus par $\mathcal{A}$ avec l'oracle O, c'est-à-dire,

$$\mathcal{A}(O) = \{u \in A^* | \exists v \in O, u * v \in L(\mathcal{A})\}$$

On appelle *langages réguliers de Delphes* les langages reconnus par des automates avec oracle.

Question 0. Montrez que le langage $\{a^n b^n \mid n \in \mathbb{N}\}$ est un langage régulier de Delphes.

Question 1. Montrez que l'intersection et l'union de deux langages réguliers de Delphes est un langage régulier de Delphes.

Question 2.

a. Montrez que la classe des langages réguliers est dénombrable.

b. Montrez que la classe des langages réguliers de Delphes n'est pas dénombrable.

Question 3. Montrez que le langage sur $\{a, b\}$ des mots avec autant de a que de b n'est pas un langage régulier de Delphes.

Question 4. Soit L un langage régulier de Delphes avec $L = \mathcal{A}(O)$. Soit $v \in B^*$, On note $E_v = \{u \mid u * v \in L(\mathcal{A})\}$.

Montrez que si L est un langage régulier alors le langage $K = \{v \mid E_v \subseteq L\}$ est un langage régulier.

Question 5. Dans la suite, on note $<_{\text{lex}}$ l'ordre lexicographique sur les mots pour un ordre abitraire sur l'alphabet A.

Soit L un langage régulier sur l'alphabet A, montrez que le langage $L_{\text{lex}} = \{u \mid u \in L \wedge \exists v \in L \cap A^{|u|}, v <_{\text{lex}} u\}$ est également un langage régulier.

Question 6. Soit L un langage régulier reconnu par un automate avec oracle $\mathcal{A}(O)$. Montrez qu'il existe un oracle O' régulier tel que $\mathcal{A}(O) = \mathcal{A}(O') = L$.

Question 7. Un oracle O est dit *uniforme* s'il existe un mot infini $w \in B^\omega$ tel que O est la suite des préfixes de w. Dans ce cas, on note $\mathcal{A}(w)$ pour $\mathcal{A}(O)$.

a. Montrez que $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas reconnaissable par un automate avec oracle uniforme.

b. Pour K un langage, on note $\textsc{Prefix}(K)$ l'ensemble de ces préfixes. C'est-à-dire, l'ensemble $\textsc{Prefix} = \{u | \exists v \in A^*, uv \in K\}$. Un langage K est clos par préfixe si $K = \textsc{Prefix}(K)$. Montrez que si K est langage régulier non fini clos par préfixe alors il existe u et v tel que $\textsc{Prefix}(uv^*) \subseteq K$.

c. Soit L un langage régulier reconnu par un automate avec oracle uniforme $\mathcal{A}(w)$. Montrez qu'il existe $u, v \in B^*$ tel que tel que $\mathcal{A}(O) = \mathcal{A}(uv^\omega)$.

C4 – Circuits booléens linéaires

Soit $\mathcal{X}_n = \{x_1, \ldots, x_n\}$ un ensemble énuméré de variables.

Un circuit booléen sur $\mathcal{X}_n$ est un graphe orienté acyclique (V, E), où les sommets de degré entrant 0, les entrées du circuit, sont étiquetés par des variables de $\mathcal{X}_n$ et les autres noeuds, les portes internes du circuit, sont étiquetés par des fonctions $\{0,1\}^r \to \{0,1\}$ avec r le degré entrant du noeud. Dans la suite pour une porte p du circuit, on note e_p la fonction qui l'étiquète. On ne considère que des fonctions parmi $\wedge, \vee, \neg$ respectivement le ET, OU, NON logique des booléens en entrée. Une *sortie du circuit* est un noeud de degré sortant 0.

Une assignation booléenne de $\mathcal{X}_n$ est une fonction $g\colon \mathcal{X}_n \to \{0,1\}$. Pour une porte p du circuit on définit son évaluation g_p comme suit :

— Pour une entrée du circuit $p = x_i$ est $g_p = g(x_i)$

— Pour un noeud interne p avec $p_1, \ldots p_r$ ses antécédents dans le circuits, $g_p = e_p(g_{p_1}, \ldots, g_{p_k})$

Une fonction $f\colon \{0,1\}^n \to \{0,1\}^k$ est calculée par un circuit si pour toute évaluation g, nous avons $f(g(x_1), \ldots, g(x_n)) = (g_{o_1}, \ldots, g_{o_k})$ avec $o_1, \ldots, o_k$ une séquence de portes de sortie du circuit. Un ensemble de mots $E \subseteq \{0,1\}^n$ est calculé par un circuit si sa fonction caractéristique (c'est-à-dire, la fonction $\mathbf{1}_E$ tel que $\mathbf{1}_E(u) = 1$ ssi $u \in E$) est calculée par le circuit.

FIGURE 1 – Un circuit calculant l'ensemble $\{00, 01, 11\}$

Dans la suite, on appellera *câbles* les arêtes du graphes.

Question 0. Montrez que toute fonction $f\colon \{0,1\}^n \to \{0,1\}$ est calculable par un circuit booléen. Indiquez le nombre de portes nécessaires et le nombre de câbles nécessaires.

Quelle relation y a-t-il entre le nombre de portes et le nombre de câbles dans un circuit ?

Question 1. La profondeur du circuit est le plus long chemin d'une entrée à la sortie.

Soit $k \in \mathbb{N}$. On note T_k^n l'ensemble des mots de $\{0,1\}^n$ qui ont au plus k positions à 1.

1. Proposez un circuit avec un nombre de câbles linéaire en n (mais pas forcément linéaire en k, qui est vue comme une constante fixée) qui calcule l'ensemble T_k^n. Quelle est sa profondeur en fonction de n ?
2. Proposez un circuit de profondeur constante qui calcule l'ensemble T_k^n. Quelle est sa taille (câbles et portes) en fonction de n ?

Question 2. On va construire un circuit de profondeur constante avec un nombre de câbles linéaire en n qui calcule T_k^n. Pour ce faire, on passe par une fonction intermédiaire qu'on nomme PREFIX-$\vee_n\colon \{0,1\}^n \to \{0,1\}^n$ telle que PREFIX- $\vee_n\,(x_1, \ldots, x_n)_i = \bigvee_{j \le i} x_j$.

Montrez que si un circuit de profondeur constante avec un nombre de câbles linéaire en n pour PREFIX-$\vee_n$ existe alors il en existe également un pour T_k^n.

Question 3. Construire un circuit de profondeur constante avec un nombre de câbles linéaire en n pour PREFIX-$\vee_n$.

Indication. On pourra regrouper les entrées par paquets de taille $\sqrt{n}$ afin d'identifier à gros trait où peut être le premier 1 dans le mot et appliquer de manière parcimonieuse un circuit naïf des ensembles portes de taille $\sqrt{n}$.

Question 4. Soit $\text{BIN}_n \colon \{0, \dots, 2^n\} \to 2^n$, la fonction qui associe un nombre à son écriture en binaire.

On considère maintenant la fonction $\text{ADDITION}_n \colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{n+1}$ qui réalise l'addition d'entiers représenté en binaire. Formellement, pour deux entiers $i, j < 2^n$, alors $\text{ADDITION}_n(\text{BIN}_n(i), \text{BIN}_n(j)) = \text{BIN}_{n+1}(i+j)$.

1. Montrez que ADDITION_n est calculable par un circuit avec un nombre de câbles linéaire en n.
2. Montrez que ADDITION_n est calculable par un circuit avec un nombre de câbles polynomial en n. et de profondeur constante.

Question 5. Montrez en appliquant une méthode similaire à la question 3, que si on sait implémenter ADDITION_n à profondeur constante avec un circuit utilisant $n \leq f(n) \leq n^2$ câbles, alors il est possible de construire un circuit avec un nombre de câbles en $O(\sqrt{n} f(\sqrt{n}))$.

En déduire, que pour tout $\epsilon > 0$, il existe un circuit calculant ADDITION_n utilisant un nombre de câbles $O(n^{1+\epsilon})$.

Question 6. Soit $G = (V, E)$ un graphe acyclique avec n sources $s_1, \dots, s_n$ (noeuds de degré entrant 0) et n sorties $o_1, \dots o_n$ (noeuds de degré sortant 0).

On dit que G est un n-super concentrateur si pour un entier $k < n$ et pour toute séquence $i_1 < j_1 < i_2 < \cdots < i_k < j_k$ il existe des k-chemins disjoints qui relient i_1 à j_1, i_2 à j_2, etc.

On admet le théorème difficile suivant.

Théorème.
Pour tout entier d, il existe une fonction $f_d \colon \mathbb{N} \to \mathbb{N}$ croissante et non-bornée telle que tout n-super concentrateur de profondeur d a un nombre d'arêtes au moins égal à $\Omega(n f_d(n))$.

En déduire qu'il n'existe pas de circuit de profondeur constante et avec un nombre de câbles linéaire en n qui calcule ADDITION_n.

On pourra éventuellement s'aider du théorème de *Menger* : étant donné deux ensemble de sommets I et J, le nombre maximal de chemins disjoints reliant I et J est égal à la taille de la plus petite coupe entre I et J.

L1 – Chip firing game

Un CFG ('chip firing game') est un jeu à un joueur qui se joue sur un graphe non-orienté $G = (V, E)$, qu'on suppose de plus connexe. Sur chaque sommet du graphe se trouve un certain nombre de jetons, ce qu'on représente par une fonction $j : V \to \mathbb{N}$, qu'on appelle une *configuration*. Un *coup* du joueur consiste à choisir un sommet $v \in V$ sur lequel se trouvent au moins autant de jetons que v a de voisins (ou encore : $j(v) \geq deg(v)$) et déplacer simultanément un jeton de v vers chacun de ses voisins (on obtient donc une nouvelle configuration j' où $j'(v) = j(v) - deg(v)$, $j'(u) = j(u) + 1$ pour tout $(u, v) \in V$ et $j'(u) = j(u)$ pour les autres sommets de G).

Une partie finie à partir d'une configuration c_0 est une suite de coups

$$c_0 \to c_1 \to \ldots \to c_n$$

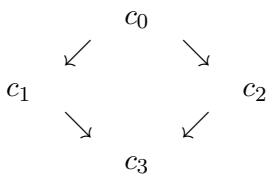
telle qu'aucun coup ne peut être joué à partir de la configuration c_n (on dit que la configuration c_n est *terminale*). De même, une partie infinie à partir d'une configuration c_0 est une suite infinie de coups

$$c_0 \to c_1 \to \ldots \to c_n \to \ldots$$

(dans laquelle aucune configuration c_i n'est donc terminale).

Question 1. Donner un exemple de partie finie sur un graphe de votre choix à partir d'une configuration de votre choix. Même question pour avec une partie infinie.

Question 2. Montrer que si c_0 est une configuration dans laquelle deux coups différents $c_0 \to c_1$ et $c_0 \to c_2$ sont possibles, alors il existe une configuration c_3 telle que $c_1 \to c_3$ et $c_2 \to c_3$ sont des coups. Graphiquement :



Question 3. Déduire de la question précédente qu'à partir d'une configuration c_0, s'il existe une partie finie alors toutes les parties sont finies, ont la même longueur et se terminent toutes sur la même configuration terminale.

Question 4. Montrer que s'il existe une partie infinie, alors chaque sommet est choisi par le joueur infiniment souvent dans la partie.

Question 5. Dans cette question on fixe un graphe $G = (V, E)$ et on s'intéresse maintenant au lien entre nombre total de jetons $\sum_{v \in V} j(v)$ sur le graphe et l'existence d'une partie finie. Montrer par un exemple que l'existence d'une partie finie ne dépend pas uniquement du nombre total de jetons.

Ce que l'on veut déterminer sont les deux valeurs seuil $T^*(G)$ et $T^\infty(G)$ telles que :

— Toute configuration avec un nombre total $t < T^*(G)$ de jetons n'admet que des parties finies.

— Toute configuration avec un nombre total $t > T^\infty(G)$ de jetons n'admet que des parties infinies.

— Pour $T^*(G) \leq t \leq T^\infty(G)$, certaines configurations à t jetons n'admettent que des parties finies, d'autres configurations à t jetons n'admettent que des parties infinies.

Question 6. Quelle est la valeur de $T^\infty(G)$?

Question 7. Montrer que $T^*(G) = |E|$. *Indication : On pourra s'aider de la notion suivante. Supposons que l'on mette un ordre total $\prec$ sur V. On note $deg^+(u, \prec)$ le nombre de voisins v de u tels que $v \prec u$. Etant donnés une configuration j et un ordre $\prec$, on dit que u est* deficient (pour l'ordre $\prec$) *si $j(u) < deg^+(u, \prec)$.*

Corrigé

Question 1. Pour les parties finies on a l'embarras du choix, on peut même prendre n'importe quel graphe avec aucun jeton dessus ! Pour les parties infinies, on prend par exemple un graphe à deux sommets avec un jeton sur l'un des deux qui fait 'ping-pong' indéfiniment.

Question 2. Raffinons la notation, en utilisant $c \xrightarrow{v} c'$ pour dire que c' a été obtenue à partir de c en jouant sur le sommet v.

Maintenant, si on a deux coups possibles $c_0 \xrightarrow{u} c_1$ et $c_0 \xrightarrow{v} c_2$ ($u \neq v$), alors le sommet v peut être joué dans la position c_1 : en effet, il peut être joué dans la position c_0 et comme $u \neq v$, le nombre de jetons du sommet v dans c_1 est au moins égal à celui de ce même sommet dans c_0. On a donc un coup valide $c_1 \xrightarrow{v} c_3$. Par le même raisonnement, on a un coup valide $c_2 \xrightarrow{u} c'_3$. Un coup pouvant être vu comme une addition de vecteur (en terme de nombre de jetons), jouer d'abord sur le sommet u puis sur le sommet v et jouer d'abord sur le sommet v puis le sommet u aboutit à la même configuration, d'où $c_3 = c'_3$.

Question 3. On appelle *hauteur* d'une configuration la plus courte partie finie que l'on peut obtenir à partir de celle-ci. Par convention cette hauteur est ∞ s'il n'existe pas de partie finie. On prouve le résultat voulu par récurrence sur la hauteur h d'une configuration.

Si une configuration c a hauteur 0, c'est qu'elle est terminale. Toutes les propriétés que l'on voulait sont vérifiées trivialement.

Supposons maintenant le résultat vrai jusqu'à hauteur h. Soit une configuration c_0 de hauteur $h+1$ et soit

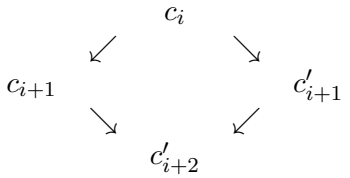
$$c_0 \to c_1 \to \ldots \to c_{h+1}$$

une plus courte partie finie, avec c_{h+1} terminale.

La configuration c_1 a hauteur $\leq h$ (puisqu'on a une partie de longueur h à partir de c_1), donc par récurrence, à partir de c_1, toutes les parties sont finies, ont même longueur (et cette longueur est donc h) et se terminent toutes sur même configuration, qui est donc c_{h+1}.

On distingue deux cas :

- Soit le coup $c_0 \to c_1$ était le seul possible à partir de c_0. Dans ce cas, d'après ce qu'on vient d'établir sur c_1, toutes les parties à partir de c_0 ont longueur $h+1$ et se terminent en c_{h+1}, ce que l'on voulait.
- Sinon, considérons n'importe quel un autre coup possible $c_0 \to c'_1$ avec $c'_1 \neq c_1$. Dans ce cas, en partant de $i = 0$, tant que $c_{i+1} \neq c_i$, on utilise la Question 2 pour obtenir un diamant



L2 – Autour de l'inégalité de Kraft

Dans ce qui suit, on se fixe un alphabet fini Σ de cardinalité $K \geq 2$. On rappelle qu'un mot $u \in \Sigma^*$ est *préfixe* d'un mot $w \in \Sigma^*$ s'il existe $u' \in \Sigma^*$ tel que $uu' = w$. On dit qu'un ensemble de mots $\Gamma \subseteq \Sigma^*$ est *sans préfixe* si aucun mot u de Γ n'est préfixe d'un autre mot v de Γ ($v \neq u$).

Question 1. Donner un exemple d'ensemble Γ sans préfixe sur l'alphabet $\Sigma = \{a, b, c\}$, contenant au moins 6 mots. Donner maintenant un exemple d'ensemble sans préfixe infini.

Question 2. Soit $\Gamma \subseteq \Sigma^*$ un ensemble de mots sans préfixe ne contenant pas le mot vide. Montrer que Γ possède la *propriété de décomposabilité unique* : si un mot w peut s'écrire à la fois $w = u_1 \dots u_n$ et $w = v_1 \dots v_m$ avec tous les u_i et v_j dans Γ, alors $n = m$ et $u_i = v_i$ pour tout i.

Question 3. Etant donné un ensemble fini de mots Γ, proposer un algorithme naïf pour tester si Γ est sans préfixe. Quelle est sa complexité ?

Question 4. Un arbre K-aire est un arbre dont chaque noeud a au plus K fils. Une feuille de l'arbre est un noeud qui n'a aucun fils. On appelle *hauteur* d'un noeud sa distance à la racine (cette dernière a donc hauteur 0). Dit autrement, la racine a hauteur 0 et si un noeud a hauteur h, ses fils ont hauteur $h + 1$.

Soit T un arbre K-aire fini. Montrer que

$$\sum_{\sigma \text{ feuille de } T} K^{-hauteur(\sigma)} \leq 1$$

Dans quel cas a-t-on égalité ?

Question 5. En utilisant la question précédente, montrer que si $\Gamma \subseteq \Sigma^*$ est un ensemble fini sans préfixe, alors

$$\sum_{w \in \Gamma} K^{-|w|} \leq 1$$

On appelle cette inégalité l'*inégalité de Kraft*. Dans quel cas a-t-on égalité ? Est-ce que l'inégalité de Kraft reste vraie pour les ensembles infinis (sans préfixe) ?

Question 6. Proposer maintenant un algorithme en temps linéaire pour tester si un ensemble de mots finis est sans préfixe.

Question 7. On va maintenant donner une preuve algorithmique de la réciproque de l'inégalité de Kraft : si $(l_i)_{i=1}^n$ est une famille d'entiers telle que

$$\sum_{i=1}^{n} K^{-l_i} \leq 1$$

alors il existe une famille de mots distincts $(u_i)_{i=1}^n$ de Σ^* formant un ensemble sans préfixe avec $|u_i| = l_i$ pour tout i.

Donner un algorithme prenant en entrée une telle suite l_i qui retourne une telle suite u_i. On demande de plus que l'algorithme soit 'en ligne', c'est-à-dire que l'algorithme lit les l_i un par un et produit u_i immédiatement après la lecture de l_i, sans attendre la valeurs des l_j pour $j > i$. *Indication : Utiliser un invariant faisant intervenir, à l'étape $t + 1$, l'écriture en base K de $1 - \sum_{i=1}^{t} K^{-l_i}$.*

Question 8. Montrer que l'inégalité de Kraft reste vraie pour tout ensemble de mots Γ (fini ou infini) vérifiant la propriété de décomposabilité unique de la Question 2. *Indication : On pourra considérer la série entière $\sum_n a_n x^n$ où a_n est le nombre de mots de Γ de longueur n – que sait-on de son rayon de convergence ? – et étudier $(\sum_n a_n x^n)^N$ pour un 'grand' entier N.*

Corrigé

Question 1. Trouver des ensembles finis sans préfixe est trivial. Pour un ensemble infini, on peut prendre par exemple

$$\Gamma = \{a^n b \mid n \in \mathbb{N}\}$$

Question 2. Par récurrence sur la longueur de w. Pour $|w| = 0$ aucune décomposition n'est possible car Γ ne contient pas le mot vide, la propriété est donc satisfaite de facto. Supposons maintenant qu'on a deux décompositions

$$w = u_1 \dots u_n = v_1 \dots v_m$$

(avec les u_i et v_j dans Γ, donc en particulier non-vides). Sans perte de généralité on a $0 < |u_1| \leq |v_1|$, et l'égalité ci-dessus implique que u_1 est donc un préfixe de v_1, donc $u_1 = v_1$ car Γ est sans préfixe. Il suit que $u_2 \dots u_n = v_2 \dots v_m$ et on conclut par récurrence, le mot $w' = u_2 \dots u_n$ étant de longueur strictement inférieure à celle de w.

Question 3. On teste pour toute paire (u, v) de mots distincts de Γ si u est un préfixe de v. Chaque comparaison prend un temps $O(|u| + |v|)$ donc on a une complexité totale en $O(n^2)$, n étant la taille mémoire de l'entrée Γ.

Question 4. Par récurrence sur la hauteur h de l'arbre, on montre en même temps l'inégalité et le fait qu'on a égalité si et seulement si l'arbre est localement complet (tout noeud est soit une feuille soit a exactement K fils). On notera $f(T) = \sum_{\sigma \text{ feuille de } T} K^{-hauteur(\sigma)}$.

Pour $h = 0$, on a une seule feuille de hauteur 0 donc $\sum_{w \in \Gamma} K^{-|w|} = 1$ et l'arbre est bien localement complet.

Pour $h + 1$, on se observe que

$$f(T) = \frac{1}{K} \sum_i f(T'_i)$$

la somme étant prise sur les sous-arbres non-vides de T. En effet la hauteur d'un noeud dans un T'_i est un de moins que sa hauteur dans T. Par récurrence, on a $f(T'_i) \leq 1$ pour tout i et on a au plus K sous-arbres, d'où l'inégalité $f(T) \leq 1$. Pour avoir l'égalité, il faut avoir exactement K sous-arbres et $f(T'_i) = 1$ pour chacun d'entre eux, ce qui oblige par récurrence a avoir chaque T'_i localement complet et donc T doit être lui-même localement complet.

Question 5. Il y a une interprétation assez évidente d'ensemble préfixe (non-vide) en terme d'arbre. Soit h la longueur du plus long mot de Γ. On peut identifier chaque mot de Γ comme un noeud de l'arbre K-aire parfait T de hauteur h, où la racine correspond au mot vide et inductivement si un noeud correspond au mot w, ses fils correspondent aux mots $\{wa \mid a \in \Sigma\}$. Le fait d'être sans préfixe est équivalent à ce que parmi les noeuds de Γ on n'en ait pas un qui soit préfixe d'un autre. On peut alors considérer l'arbre $T_\Gamma = \{u \mid u \text{ est préfixe d'un mot } w \in \Gamma\}$. Si Γ est sans préfixe, chaque $w \in \Gamma$ est une feuille de T_Γ, avec $hauteur(w) = |w|$. L'inégalité de Kraft suit.

De plus, on a égalité quand Γ est *maximal*, c'est-à-dire contenu dans aucun ensemble de mots sans préfixe strictement plus grand. Par la Question 4, il suffit de montrer que Γ est maximal si et seulement si son arbre T_Γ est localement parfait. En effet, si T_Γ n'est pas localement parfait, un noeud non

L3 – Ordres et intervalles

On rappelle que sur un ensemble E donné, un *ordre strict* est une relation $\prec$ irréflexive et transitive (et donc aussi anti-symétrique), pas nécessairement totale (pour $x, y \in E$, on peut n'avoir ni $x \prec y$, ni $y \prec x$).

Question 1. Soit $\mathcal{I}$ l'ensemble des intervalles $[a, b]$ (avec $a \leq b$) de $\mathbb{R}$. Montrer que la relation $\prec_{\mathcal{I}}$ definie sur $\mathcal{I}$ par

$$[a, b] \prec_{\mathcal{I}} [c, d] \Leftrightarrow b < c$$

est bien un ordre strict.

Soit E un ensemble et $\prec$ un ordre strict sur cet ensemble. On dit que $\prec$ est un *ordre d'intervalles* si on peut associer à chaque $x \in E$ un intervalle $[a_x, b_x]$ de $\mathbb{R}$ de telle sorte que pour tout $x, y \in E$:

$$x \prec y \Leftrightarrow [a_x, b_x] \prec_{\mathcal{I}} [a_y, b_y]$$

Dans ce qui suit, E sera un ensemble fini à n éléments qu'on identifie à $\{1, \ldots, n\}$ et $\prec$ une relation d'ordre strict sur E représenté par une matrice M de taille $n \times n$ où $M[i, j] = 1$ si $i \prec j$, et $M[i, j] = 0$ sinon.

Question 2. Montrer que si $\prec$ est un ordre d'intervalles, alors $(E, \prec)$ a la propriété suivante, qu'on note $(\star)$: il n'existe pas $x, x', y, y' \in E$ tels que : $x \prec x'$, $y \prec y'$, $x \not\prec y'$ et $y \not\prec x'$.

On admet pour le moment que la réciproque de la question précédente est vraie (c'est le théorème de Fishburn, qu'on démontrera dans les questions suivantes) : si $(E, \prec)$ a la propriété $(\star)$, alors $\prec$ est un ordre d'intervalles. En déduire un algorithme polynomial pour tester si la matrice M représente un ordre d'intervalles. Quelle est en fonction de n la complexité de cet algorithme ?

Question 3. Pour tout $x \in E$, on pose

$$D(x) = \{y \in E \mid y \prec x\}$$

Montrer que si $(E, \prec)$ possède la propriété $(\star)$, alors il possède la propriété $(\star\star)$ suivante : pour tout $x, x' \in E$, on a soit $D(x) \subseteq D(x')$, soit $D(x') \subseteq D(x)$.

Question 4. Montrer maintenant que si $(E, \prec)$ possède la propriété $(\star\star)$, alors $\prec$ est un ordre d'intervalles (ce qui conclut la preuve du théorème de Fishburn). *Indication : en raisonnant pas récurrence, on pourra isoler un élément x qui soit le 'plus maximal' possible en utilisant la question précédente.*

Question 5. A partir des questions précédentes, donner maintenant un algorithme en $O(n^2)$ qui décide si M représente un ordre d'intervalles.

Corrigé

Question 1. La vérification est triviale.

L4 – Mariage en treillis

Soient A et B deux ensembles finis disjoints, qu'on suppose de même cardinalité n. On cherche à « marier » les éléments de A aux éléments de B, chaque élément devant être marié à exactement un élément de l'autre ensemble (un mariage $\mathcal{M}$ peut donc être représenté par une bijection de A dans B). De plus, chaque élément de A (respectivement de B) a un ordre de préférence strict et total entre les éléments de B (respectivement de A). On notera $<^x$ l'ordre associé à l'élément x (le x pouvant être omis quand le contexte est clair). On appellera *système de préférences* la liste des ordres $<^x$ pour $x \in A \cup B$.

Un mariage $\mathcal{M}$ est *instable* s'il existe un quadruplet $(a_1, a_2, b_1, b_2) \in A \times A \times B \times B$ tel que

— a_1 est marié à b_2 via $\mathcal{M}$;

— b_1 est marié à a_2 via $\mathcal{M}$;

— a_1 préfère b_1 à b_2 (autrement dit : $b_2 <^{a_1} b_1$) ;

— b_1 préfère a_1 à a_2 (autrement dit : $a_2 <^{b_1} a_1$).

Un mariage qui n'est pas instable est dit *stable*.

Question 1. Donner un exemple d'ensembles A, B à deux éléments chacun avec des ordres de préférence tel qu'il existe plus d'un mariage stable.

L'algorithme de Gale–Shapley permet de montrer qu'il existe toujours au moins un mariage stable et d'en trouver un de façon efficace. L'idée de l'algorithme est que les éléments de A font des propositions aux éléments de B. Quand un élément de A fait une proposition à un élément de B, celui-ci peut soit la refuser définitivement, soit l'accepter provisoirement (en attendant une éventuelle meilleure offre). Toute nouvelle acceptation vaut rejet définitif des propositions précédemment acceptées. Cela donne l'algorithme suivant.

```
Initialiser M = []
Tant qu'un élément x de A n'est pas marié
        y = élément de B que x préfère parmi ceux à qui x n'a pas encore proposé
        Si y n'est pas marié dans M
                Ajouter (x,y) à M
        Sinon si y est marié dans M à x' et y préfère x à x'
                Retirer (x',y) de M
                Ajouter (x,y) à M
        Sinon
                 Ne rien faire

Retourner M
```

Question 2. Montrer que l'algorithme de Gale–Shapley termine. Proposer des structures de données permettant d'implémenter l'algorithme le plus efficacement possible ; quelle est la complexité qui en résulte ?

Question 3. Montrer la correction de cet algorithme, à savoir qu'on obtient bien un mariage stable à la fin de l'exécution. *Indication : Fixer un $a \in A$ et regarder ce qui lui arrive au cours de l'algorithme. Même chose pour un $b \in B$.*

Question 4. Soit $\mathcal{E}$ un ensemble non vide et $\preceq$ une relation d'ordre (pas nécessairement totale) sur $\mathcal{E}$. On dit qu'un élément $e \in \mathcal{E}$ est *maximal* s'il n'existe pas $e' \neq e$ tel que $e \preceq e'$. On dit que e est un élément *maximum* (ou *le* maximum car il est alors unique) si pour tout e' on a $e' \preceq e$.

Un ensemble quelconque non vide $\mathcal{E}$ muni d'une relation d'ordre a-t-il toujours un élément maximal ? Un élément maximum ? Et si $\mathcal{E}$ est fini ?

Question 5. On suppose maintenant fixés les ensembles A, B et le système de préférences. On définit un ordre partiel $\preceq$ sur les mariages stables, où $\mathcal{M}_1 \preceq^A \mathcal{M}_2$ veut dire que les éléments de A sont tous au moins aussi satisfaits dans le mariage $\mathcal{M}_2$ que dans le mariage $\mathcal{M}_1$ (c'est-à-dire, pour tout $a \in A$, si a est marié avec b_1 dans $\mathcal{M}_1$ et b_2 dans $\mathcal{M}_2$, alors $b_1 \preceq^a b_2$). $\mathcal{M}_1 \preceq^B \mathcal{M}_2$ est défini de façon similaire, quand tous les éléments de B sont au moins satisfaits dans $\mathcal{M}_2$ que dans $\mathcal{M}_1$.

Montrer que si $\mathcal{M}_1$ et $\mathcal{M}_2$ sont deux mariages stables et que $\mathcal{M}_1 \preceq^A \mathcal{M}_2$, alors $\mathcal{M}_2 \preceq^B \mathcal{M}_1$.

Question 6. Toujours en supposant fixés les ensembles A, B et le système de préférences, on définit l'opération $\vee$ sur les mariages stables de la façon suivante : pour tout $a \in A$, si a est marié à b_1 dans $\mathcal{M}_1$ et à b_2 dans $\mathcal{M}_2$, on marie a à son élément préféré de $\{b_1, b_2\}$. Montrer qu'on obtient bien ainsi un mariage stable, qu'on note donc $\mathcal{M}_1 \vee \mathcal{M}_2$.

Question 7. Montrer que dans l'ensemble des mariages stables, il existe un élément maximum et un élément minimum pour l'ordre $\preceq^A$.

Question 8. Montrer que le mariage produit par l'algorithme de Gale–Shapley est en fait l'élément maximum pour $\preceq^A$. *Indication : pour $a \in A$, on dit que $b \in B$ est un* partenaire valide *de a s'il existe un mariage stable où a est marié à b. Dans l'exécution de Gale–Shapley, considérer le premier moment où un élément de A est rejeté par un de ses partenaires valides.*