Feuille d'exercices n°15 - Révisions de MPI

Notions abordées

- programmation dynamique
- automates : déterminisation, algorithme de Berry-Sethi, lemme de l'étoile
- grammaires non contextuelles : arbre de dérivation, ambiguïté
- arbres couvrants, algorithmes glouton et argument d'échange
- problème indécidables, réduction
- classes de complexité P et NP, réduction
- algorithme de branch-and-bound
- algorithmes d'IA

Exercice 1 : L'étoile d'un langage dans P est aussi dans P

Le but de cet exercice est de montrer que pour L un langage de P, L^* est aussi dans P.

Q. 1 Rappeler ce que signifie $L \in \mathsf{P}$ et ce que signifie $L^* \in \mathsf{P}$.

Q. 2 Combien y a-t-il de découpages en facteurs non-vides d'un mot de taille $n > 0$? En déduire la complexité de l'algorithme naïf qui teste l'appartenance d'un mot à L^* à partir d'un test d'appartenance à L en envisageant un à un tous les découpages possibles ?

Q. 3 Combien y a-t-il de facteurs non-vides d'un mot de taille n ? En déduire quel schéma algorithmique il pourrait être judicieux de mettre en place pour tester l'appartenance à L^* ?

Q. 4 À quelle condidtion un mot de taille 0 est-il dans L^* ?
À quelle condidtion un mot de taille 1 est-il dans L^* ?

Q. 5 Soit $w \in \Sigma^\star$ de taille $n \in \mathbb{N}^*$. Pour $\{(i,j) \in [\![1,n]\!]^2 \mid i \leqslant j\}$, on définit $A_{i,j} = \begin{cases} \mathsf{V} & \text{si } w_i \dots w_j \in L^* \\ \mathsf{F} & \text{sinon} \end{cases}$
Donner une caractérisation récursive de A.

Q. 6 Déduire des questions précédentes un algorithme qui teste en temps polynomial l'appartenance à L^*, en supposant que le test d'appartenance à L est en temps polynomial.

Q. 7 Conclure.

Exercice 2 : Énumération des mots reconnus par un automate

Dans cet exercice on se propose d'énumérer les mots reconnus par un automate (non nécessairement déterministe). Un tel ensemble pouvant bien sûr être infini, on cherche à fournir un algorithme qui, étant donné un automate $\mathcal{A}$ sur un alphabet Σ et un entier $n \in \mathbb{N}$ produit l'ensemble des mots de Σ^n reconnus par $\mathcal{A}$, ensemble qu'on notera $\mathcal{L}_n(A)$ dans la suite de cet exercice :

$$\mathcal{L}_n(\mathcal{A}) \overset{\text{déf}}{=} \mathcal{L}(\mathcal{A}) \cap \Sigma^n$$

Étant donné un automate $\mathcal{A} = (\Sigma, Q, I, F, \delta)$ et un état $q \in Q$, on note $\mathcal{A}_q$ l'automate $(\Sigma, Q, \{q\}, F, \delta)$. La seule différence entre $\mathcal{A}_q$ et $\mathcal{A}$ se situe au niveau des états initiaux : ceux de $\mathcal{A}$ sont remplacés par le seul état q dans $\mathcal{A}_q$.

Q. 1 Étant donné un automate $\mathcal{A} = (\Sigma, Q, I, F, \delta)$, exprimer $\mathcal{L}_0(\mathcal{A}_q)$ en fonction de $q \in Q$?

Q. 2 Étant donné un automate $\mathcal{A} = (\Sigma, Q, I, F, \delta)$, donner et prouver, pour tout $(n, q) \in \mathbb{N}^\star \times Q$, une relation entre $\mathcal{L}_n(\mathcal{A}_q)$ et les $\mathcal{L}_p(\mathcal{A}_{q'})$ pour $p < n$ et $q' \in Q$.

Q. 3 Donner, pour tout $n \in \mathbb{N}$, une relation entre $\mathcal{L}_n(\mathcal{A})$ et les $\mathcal{L}_n(\mathcal{A}_q)$ pour $q \in Q$.

Les questions précédentes devraient donner une idée d'algorithme récursif permettant d'énumérer les mots d'une longueur donnée reconnus par un automate donné. Toutefois, avant de se jeter dans l'implémentation, on s'interroge sur la complexité de l'algorithme envisagé.

Q. 4 Donner, pour chaque $n \in \mathbb{N}$ un automate $\mathcal{A}_n$ à n états et $\Theta(n)$ transitions♣ tel que l'arbre des appels récursifs pour le calcul de $\mathcal{L}_n(\mathcal{A}_n)$ est de taille exponentielle en n, tandis que le cardinal de $\mathcal{L}_n(\mathcal{A}_n)$ est en $\mathcal{O}(n)$.

Q. 5 Donner, en fonction du nombre d'états M de l'automate $\mathcal{A}$ et de l'entier n une majoration du nombre d'ensembles $\mathcal{L}_p(\mathcal{A}_q)$ qui entrent en jeu dans le calcul récursif de $\mathcal{L}_n(\mathcal{A})$.

Q. 6 Proposer un schéma algorithmique permettant la réconciliation des observations des deux dernières questions.

Q. 7 Proposer une implémentation en OCaml de cet algorithme. On pourra supposer que l'ensemble Q des états de l'automate est de la forme $[\![0, n-1]\!]$, avec $n \in \mathbb{N}$. On accordera une attention particulière au fait que :

- il n'est pas nécessaire d'avoir un espace mémoire en $\Theta(Mn)$, un $\Theta(n)$ suffit.
- si l'on itère sur chaque état, et que pendant chacune de ces itérations, on itère sur ses prédécesseurs, il est peut être envisageable de ne parcourir qu'une fois la liste des transitions.

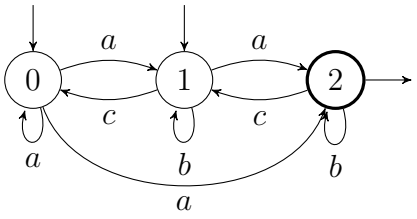
♣. on entend par là que pour tout $n \in \mathbb{N}$, $\mathcal{A}_n$ a exactement t_n transitions, et que $(t_n)_{n\in\mathbb{N}} \in \Theta(n)$.

Exercice 3 : Automates (révisions)

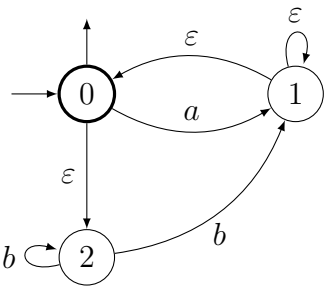
Q. 1 Donner un automate complet non déterministe à 5 états qui reconnaît les mots de $\{a, b\}^\star$ ayant $baba$ pour facteur.

Q. 2 Donner un automate déterministe qui reconnaît les mots de $\{a, b\}^\star$ ayant un nombre de b multiple de 3 et n'ayant pas deux a consécutifs.

Q. 3 **En appliquant exactement l'algorithme de déterminisation du cours,** donner un automate déterministe et complet équivalent à l'automate ci-dessous, défini sur $\Sigma = \{a, b, c\}$.
On attend uniquement la table de l'automate, on pensera à y faire figurer les éventuels états initiaux et finaux.

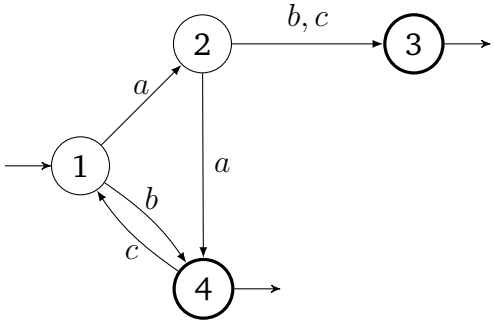


Q. 4 Donner l'automate obtenu, **en appliquant exactement l'algorithme de suppression des ε-transitions du cours,** sur l'automate suivant, en traitant les états dans l'ordre $0, 1, 2$.
On attend une représentation de l'automate obtenu après le traitement de chaque état.



Q. 5 Donner l'automate obtenu en appliquant **l'algorithme de Berry-Sethi** à l'expression régulière $e = a(a|(b{\cdot}(a^\star{\cdot}b)))^\star{\cdot}b{\cdot}a^\star$.

Q. 6 En appliquant l'algorithme reposant sur des automates généralisés vu en cours, donner une expression régulière décrivant le langage reconnu par l'automate ci-dessous. *On pourra simplifier légèrement les expressions régulières au fur et à mesure. On prendra soin de supprimer les états par numéros croissants, et on demande de représenter l'automate après chaque suppression.*



Q. 7 Le langage $\{a^n b^m c^p | (n, m, p) \in \mathbb{N}^3\}$ est-il régulier ? Justifier.

Q. 8 Le langage $\{a^n b^m c^n | (n, m) \in \mathbb{N}^3\}$ est-il régulier ? Justifier.

Exercice 4 : Grammaires non contextuelles (révisions)

Soit $\Sigma = \{a, b, c\}$. Soit $\mathcal{V} = \{S, X\}$. Considérons alors la grammaire $\mathcal{G} = (\Sigma, \mathcal{V}, S, P)$ où P est l'ensemble de règles de production ci-contre.

$$\begin{array}{l} S \to aXc \\ X \to S \mid XX \mid b \mid \varepsilon \end{array}$$

Q. 1 Justifier à l'aide d'une suite de dérivations immédiates que $aabcc \in \mathcal{L}(\mathcal{G})$.

Q. 2 Montrer que $\mathcal{G}$ est ambigüe.

Q. 3 Décrire explicitement le langage $\mathcal{L}(\mathcal{G})$.

Q. 4 Donner une grammaire $\mathcal{G}'$ non ambigüe faiblement équivalente à $\mathcal{G}$.

Q. 5 Donner en OCaml une fonction qui réalise l'analyse syntaxique d'un mot pour $\mathcal{G}'$.

Q. 6 Donner une grammaire qui engendre le langage $\{w \in \Sigma^* \mid |w|_c \geqslant 2\}$.

Q. 7 Donner une grammaire qui engendre le langage $\{w \in \Sigma^* \mid |w|_c = 2 \text{ et } |w|_a = 1\}$.

Exercice 5 : Arguments d'échange et ACPM

On considére dans cet exercice la variante du problème ACPM suivante.

$$\text{ECPM} : \begin{cases} \textbf{Entrée :} \text{ Un graphe connexe non orienté pondéré positivement } G = (S, A, c) \\ \textbf{Sortie :} \text{ Un ensemble d'arêtes } A' \subseteq A \text{ tel que } \sim_A = \sim_{A'} \text{ minimisant } \sum_{a \in A'} c(a) \end{cases}.$$

Soit $G = (S, A, c)$ un graphe connexe non orienté pondéré positivement.

Q. 1
$a)$ Montrer qu'il existe une solution optimale de ECPM pour G qui est acyclique.
$b)$ Montrer que si $\forall a \in A, c(a) > 0$ alors toute solution optimale de ECPM pour G est acyclique.

Q. 2 On suppose que G admet un cycle γ. Soit a^+ une arête γ de poids maximal.
$a)$ Montrer qu'il existe une solution optimale de ECPM pour G ne contenant pas a^+.
$b)$ Montrer que si $\forall a \in \gamma, c(a) < c(a^+)$ alors aucune solution optimale de ECPM pour G ne contient a^+.

Q. 3 Soit a^- une arête γ de poids minimal dans G.
$a)$ Montrer qu'il existe une solution optimale de ECPM pour G contenant pas a^-.
$b)$ Montrer que si $\forall a \in A, c(a^-) < c(a)$ alors toute solution optimale de ECPM pour G contient a^-.

Exercice 6 : Réductions entre problèmes indécidables (révisions)

Q. 1 Montrer que le problème suivant est indécidable.

$$Q_1 : \begin{cases} \textbf{Entrée} & : \text{La sérialisation d'une machine } \mathcal{M}\text{, deux mots } w_1 \in \Sigma^\star \text{ et } w_2 \in \Sigma^\star \\ \textbf{Sortie} & : \text{La machine } \mathcal{M} \text{ s'arrête-t-elle } w_1 \text{ mais pas sur } w_2 \text{ ?} \end{cases}$$

Q. 2 On dit qu'un langage L est stable par extraction si et seulement si tout sous-mot d'un mot de L est lui-même dans L. Montrer que le problème suivant est indécidable.

$$Q_2 : \begin{cases} \textbf{Entrée :} \text{ La sérialisation d'une machine } M. \\ \textbf{Sortie :} \mathcal{L}(M) \text{ est-il stable par extraction ?} \end{cases}$$

Exercice 7 : Partition Bis

On rappelle que le problème SubsetSum ci dessous est NP complet.

$$\begin{cases} \textbf{Entrée :} \text{ Un entier } n \in \mathbb{N}\text{, une suite finie } (w_i)_{i \in [\![1,n]\!]} \in \mathbb{N}^n\text{, un entier } W \\ \textbf{Sortie :} \text{ Existe-t-il } I \subseteq [\![1,n]\!] \text{ tel que } \sum_{i \in I} w_i = W \text{ ?} \end{cases}$$

On définit le problème PartitionBis comme prenant en entrées : un entier n, une famille de n nombres entiers, deux entiers W_1 et W_2 et demandant s'il est possible de partitionner cette famille d'entiers de sorte que la somme des éléments de la première partition soit inférieur à W_1, et la somme des éléments de la seconde partition soit inférieur à W_2.

Q. 1 Formaliser le problème PartitionBis.

Q. 2 Montrer que PartitionBis $\in$ NP.

Q. 3 Montrer que PartitionBis est NP-difficile.

Exercice 8 : Coloration et couverture de graphe

Dans cet exercice on considère deux problèmes de décision associés à des problèmes de minimisation sur les graphes non orientés. Le problème de coloration consiste à colorier les sommets d'un graphe avec le moins de couleur possible de sorte que deux sommets reliés par une arête ne soient pas de la même couleur. Le problème de couverture par des cliques consiste à répartir les sommets d'un graphe en le moins de groupe possible de sorte que chacun de ces groupes forme une clique♣.

Q. 1 Formaliser les problèmes de décision ColorationMin et Couv.Clique respectivement associés au problème de coloration et au problème de couverture par des cliques.

Q. 2 Montrer que Couv.Clique $\preccurlyeq_P$ ColorationMin et ColorationMin $\preccurlyeq_P$ Couv.Clique.

♣. On rappelle qu'une **clique** est un ensemble de sommets qui induit un sous-graphe complet.

Exercice 9 : Problèmes NP-difficiles de dominants

Dans cet exercice on s'intéresse à deux problèmes de décision sur des graphes non orientés. Afin de définir ces problèmes, on donne d'abord quelques définitions.

Définition 0.1

Soit $G = (S, A)$ un graphe non orienté.

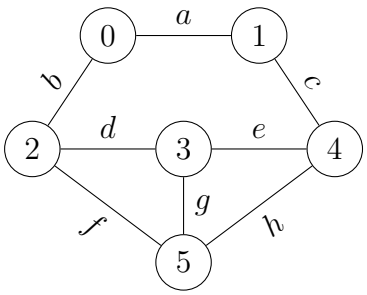
- *Deux sommets u et v sont **voisins** dans G ssi $\{u, v\} \in A$.*
- *Deux arêtes e et f sont **adjacentes** dans G ssi $e \cap f \neq \emptyset$.*
- *Soit $R \subseteq S$ un sous-ensemble de sommets.*
  *R est un **ensemble de sommets dominant** de G ssi $\forall u \in S \setminus R, \exists v \in R, \{u, v\} \in A$, $e \cap R \neq \emptyset$, autrement dit ssi chaque sommet de G qui n'est pas dans R est voisin d'un sommet de R.*
- *Soit $D \subseteq A$ un sous-ensemble d'arêtes.*
  *D est un **ensemble d'arêtes dominant** de G ssi $\forall e \in A \setminus D$, $\exists d \in D, e \cap d \neq \emptyset$, autrement dit ssi chaque arête de G qui n'est pas dans D est adjacente à une arête de D.*

Définition 0.2

*Soit $G = (S, A)$ un graphe non orienté. Le **graphe adjoint** de G, noté $L(G)$♣, est le graphe non orienté qui représente la relation d'adjacence des arêtes de G.*

$$L(G) \overset{\text{déf}}{=} (A, B) \text{ où } B = \{\{e, f\} \mid e \in A, f \in A, e \cap f \neq \emptyset\}$$

Q. 1 Représenter $L(G)$ pour le graphe G ci-dessous.



On remarque que, peu importe le graphe non orienté $G = (S, A)$, S est toujours un ensemble de sommets dominant, et A est toujours un ensemble d'arêtes dominant. On considère alors les deux problèmes de décision suivants.

DOM.SOMMETS
- **Entrée :** Un graphe non orienté $G = (S, A)$, un seuil $K \in \mathbb{N}$.
- **Sortie :** G admet-il un ensemble de sommets dominant R tel que $|R| \leqslant K$?

DOM.ARÊTES
- **Entrée :** Un graphe non orienté $G = (S, A)$, un seuil $K \in \mathbb{N}$.
- **Sortie :** G admet-il un ensemble d'arêtes dominant D tel que $|D| \leqslant K$?

Q. 2 Justifier que le problème DOM.SOMMETS est dans la classe NP.

Q. 3 Proposer une relation de réduction polynomiale entre ces deux problèmes et la démontrer.

Q. 4 Que peut-on en déduire quant à la difficulté de ces problèmes de décision ?

♣. Ce graphe est appelé **line graph** de G en anglais.

Exercice 10 : Problèmes Dom.Sommets et Couv.Ens

1. Introduction des problèmes

Étant donné une famille $\mathcal{F}$ de sous-ensembles d'un ensemble X, on appelle **couverture** de X une sous-famille de $\mathcal{F}$ dont l'union est X. Le problème de la couverture par ensembles minimum consiste à trouver une couverture de cardinal minimum. Le problème de décision associé est le suivant.

$$\text{Couv.Ens} \begin{cases} \textbf{Entrée :} \text{ Un ensemble fini } X \neq \emptyset, n \in \mathbb{N}^*, (X_i)_{i\in[\![1,n]\!]} \in \mathcal{P}(X)^n, \text{ un seuil } K \in \mathbb{N}. \\ \textbf{Sortie :} \text{ Existe-t-il } J \subseteq [\![1,n]\!] \text{ tel que } X = \bigcup_{j\in J} X_j \text{ et } |J| \leqslant K \text{ ?} \end{cases}$$

Q. 1 Soit $(X, n, (X_i)_{i\in[\![1,n]\!]}, K)$ une instance de Couv.Ens. Donner une condition suffisante pour que X admette une couverture (de cardinal quelconque) et justifier que l'on peut tester cette condition en temps polynomial.

Étant donné un graphe non orienté $G = (S, A)$, on appelle **ensemble de sommets dominant** de G un sous-ensemble $R \subseteq S$ tel que chaque sommet de G qui n'est pas dans R est voisin d'un sommet de R, *i.e.* tel que $\forall u \in S \setminus R, \exists v \in R, \{u, v\} \in A,\ e \cap R \neq \emptyset$. On remarque que S est toujours un ensemble de sommets dominant. Le problème du dominant minimum consiste alors à trouver l'ensemble de sommets de plus petit cardinal qui est un dominant. Le problème de décision associé est le suivant.

$$\text{Dom.Sommets} \begin{cases} \textbf{Entrée :} \text{ Un graphe non orienté } G = (S, A) \text{ avec } S \neq \emptyset, \text{ un seuil } K \in \mathbb{N}. \\ \textbf{Sortie :} \text{ } G \text{ admet-il un ensemble de sommets dominant } R \text{ tel que } |R| \leqslant K \text{ ?} \end{cases}$$

Q. 2 Donner une instance négative et une instance positive de Dom.Sommets.

2. Réduction de Dom.Sommets à Couv.Ens

Q. 3 Montrer que Dom.Sommets se réduit en temps polynomial à Couv.Ens.

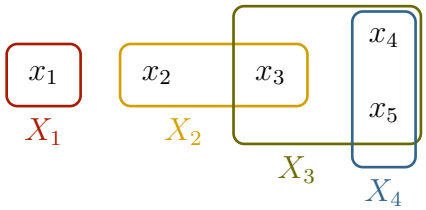
3. Réduction de Couv.Ens à Dom.Sommets

Soit $(X, n, (X_i)_{i\in[\![1,n]\!]}, K)$ une instance de Couv.Ens. On suppose que X admet au moins une couverture. De plus, quitte à renommer les éléments de X, on suppose que $X \cap [\![1, n]\!] = \emptyset$. On pose alors $I = [\![1, n]\!]$, $S = X \sqcup I$, $A = A_1 \sqcup A_2$ où

$$A_1 = \{\{i, j\} \mid (i, j) \in I^2, i \neq j\} \text{ et } A_2 = \{\{i, x\} \mid i \in I, x \in X, x \in X_i\}.$$

On considère enfin le graphe $G = (S, A)$, ainsi (G, K) est une instance de Dom.Sommets.

Q. 4 Dessiner le graphe G pour l'instance représentée ci-dessous où $X = \{x_1, x_2, x_3, x_4, x_5\}$, $n = 4$, $X_1 = \{x_1\}$, $X_2 = \{x_2, x_3\}$, $X_3 = \{x_3, x_4, x_5\}$ et $X_4 = \{x_4, x_5\}$.



Q. 5 Supposons qu'il existe $J \subseteq [\![1, n]\!]$ une couverture de X. Montrer qu'il existe un dominant de G de même cardinal.

Q. 6 Supposons que $R \subseteq S$ est un dominant de cardinal minimum pour G. Posons $J = R \cap I$.

$a)$ Montrer que J n'est pas nécessairement une couverture pour X.
On pourra s'appuyer sur l'exemple ci-dessus.

$b)$ Montrer que si $R \subseteq I$, alors J (et donc R) est une couverture pour X.

$c)$ Montrer que si $|R \cap X| > 0$, alors il existe $R' \subseteq S$ un autre dominant minimum pour G tel que $|R' \cap X| < |R \cap X|$.

Q. 7 Montrer que si R est un dominant de G cardinal minimum, alors il existe une couverture de X de cardinal $|R|$.

Q. 8 Montrer finalement que COUV.ENS se réduit en temps polynomial à DOM.SOMMETS.

Exercice 11 : Séparation et évaluation

On s'intéresse dans cet exercice à l'utilisation du schéma algorithmique de séparation et évaluation dans le cadre la résolution d'un problème de minimisation.
Afin de rester générique, on note instance le type des instances du problème de minimisation, solution le type des solutions et sol_partielle le type des solutions partielles. On entend par là qu'une solution partielle contient l'ensemble des décisions déjà prises à un nœud, ainsi que les éventuelles informations complémetaires utiles pour générer les solutions partielles correspondant aux enfants d'un tel nœud selon le schéma de branchement choisi.
On suppose qu'on dispose des fonctions suivantes :

- heuristique_init qui prend en entrée une instance et qui renvoie un couple de solution $\times \mathbb{R}$ donnant une solution et sa valeur ;
- heuristique_compatible qui prend en entrée une instance ainsi qu'une sol_partielle et qui renvoie un couple de solution $\times \mathbb{R}$ donnant une solution compatible avec la solution partielle et sa valeur ;
- calcule_minorant qui prend en entrée une instance ainsi qu'une sol_partielle et qui renvoie un minorant de la valeur des solutions compatibles avec la solution partielle ;
- genere_enfants qui prend en entrée une instance ainsi qu'une sol_partielle et qui renvoie la liste des solutions partielles enfant de cette solution partielle selon la statégie de branchement choisie♣.

On suppose que lorsqu'une solution partielle est en fait une solution complète, les valeurs obtenues par les fonctions heuristique_compatible et calcule_minorant coïncident, et même coïncident toutes deux avec la valeur de ladite solution.

Q. 1 Écrire le pseudo-code de l'algorithme de séparation et évaluation dont la stratégie d'exploration est un parcours en profondeur.

Identifier dans le pseudo-code les trois cas d'arrêts de l'exploration locale :

- le nœud est "résolu" ;
- le nœud est stérile ;
- le nœud n'est pas intéressant (c'est l'élagage).

Q. 2 Même question lorsque la stratégie d'exploration est un parcours en largeur.

Q. 3 Même question lorsque la stratégie d'exploration est de développer prioritairement les nœuds les plus prometteurs (c'est-à-dire ayant le plus petit minorant).

♣. La signature de genere_enfants est un peu restrictive et ne couvre pas le cas où l'on souhaite brancher sur la variable la plus fractionnaire du relâché continu par exemple. Pour cela, il aurait fallut que calcule_minorant renvoie un couple solution relachée/valeur et que la fonction genere_enfants prenne en argument cette solution relachée.

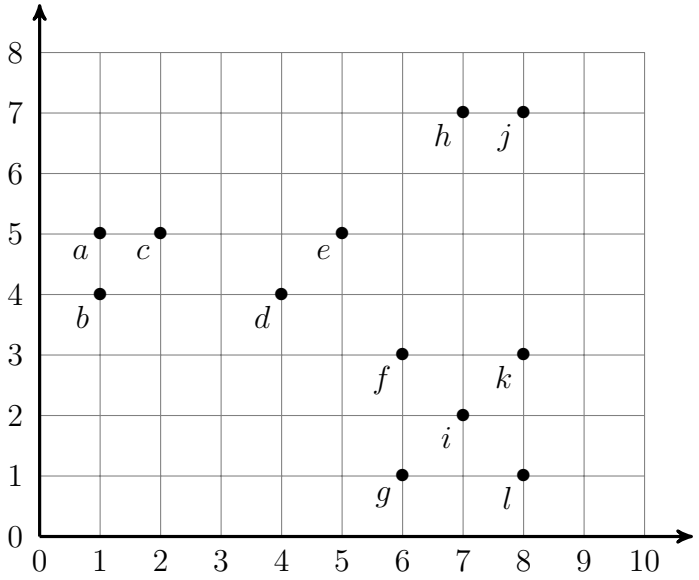
Exercice 12 : Algorithmes d'intelligence artificielle (révisions)

Q. 1 Construire un arbre 3-dimmensionnel permettant de représenter le jeu de données suivant.
On rangera à droite d'un nœud des valeurs strictement inférieures, et à gauche des valeurs supérieures ou égales. Dans le cas d'un nombre paire de valeurs, on choisira comme médiane la plus grande des deux valeurs possibles.

- $(0, 1, 4)$
- $(0, 2, 3)$
- $(1, 3, 10)$
- $(1, 5, 9)$
- $(2, 10, 10)$
- $(2, 10, 11)$
- $(3, 3, 5)$
- $(3, 4, 6)$
- $(5, 0, 8)$
- $(5, 0, 2)$
- $(5, 3, 1)$
- $(5, 9, 10)$
- $(7, 10, 1)$
- $(7, 10, 3)$

Q. 2 Rappeler quel est l'intérêt d'un arbre k-dimensionnel et dans le cadre de quel algorithme d'intelligence artificielle il est notamment utilisé.

Q. 3 Appliquer l'algorithme de classification hiérarchique ascendante (HAC) sur le jeu de données réprésenté ci-dessous, en utilisant le diamètre comme mesure de dissimilarité, et en rassemblant les classes tant que le diamètre ne dépasse pas 2.
On traitera les points de gauche à droite.



Q. 4 Appliquer l'algorithme ID3 sur le jeu de données suivant.

i	x_i	y_i	z_i	classe
0	F	F	F	F
1	F	F	V	F
2	F	V	F	V
3	F	V	V	V

i	x_i	y_i	z_i	classe
4	V	F	F	V
5	V	F	V	V
6	V	V	F	V
7	V	V	V	V

Q. 5 Parmi les algorithmes appliqués dans cet exercice, dire lesquels relèvent de l'apprentissage supervisé et lesquels relèvent de l'apprentissage non supervisé. Justifier brièvement.