INF8215 : Examen final (Automne 2022) - Quentin Cappart			Note finale ↓
Date : 14/12/2021	Nombre de questions : **6**	Total des points : **40**	
Matricule :	Nom :		Temps : 2h30 heures

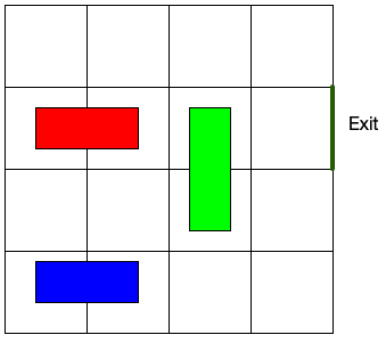
Question:	1	2	3	4	5	6	Total
Points:	0	10	5	10	5	10	40
Score:							

Instructions

1. La documentation papier et électronique sur clé USB est autorisée.
2. L'examen doit se réaliser de manière individuelle. Il est strictement interdit de collaborer.

Question 2 (10 points)

Noël approchant à grand pas, vous souhaitez acheter un jeu de plateau pour votre nièce. Au magasin, vous découvrez le jeu *Rush hour*. Brièvement, le jeu consiste à faire sortir la voiture rouge du plateau en la faisant coulisser à gauche ou à droite suivant le sens de conduite (les mouvements sur les côtés latéraux de la voiture ne sont pas autorisés). La difficulté est que d'autres véhicules sont sur le plateau et bloquent les mouvements. Chaque véhicule est sujet au même règles de déplacement. Une solution valide est une séquence d'actions permettant de faire sortir la voiture rouge, et une solution optimale est la séquence comportant le moins d'actions. Une illustration d'une situation initiale est proposée dans la figure suivante. Suite à ce que vous appris lors du cours, vous vous dites que la résolution de ce jeu peut se faire à l'aide d'une stratégie de recherche. Une illustration du jeu réel et montrée à gauche, et une situation simplifiée est proposée à droite.

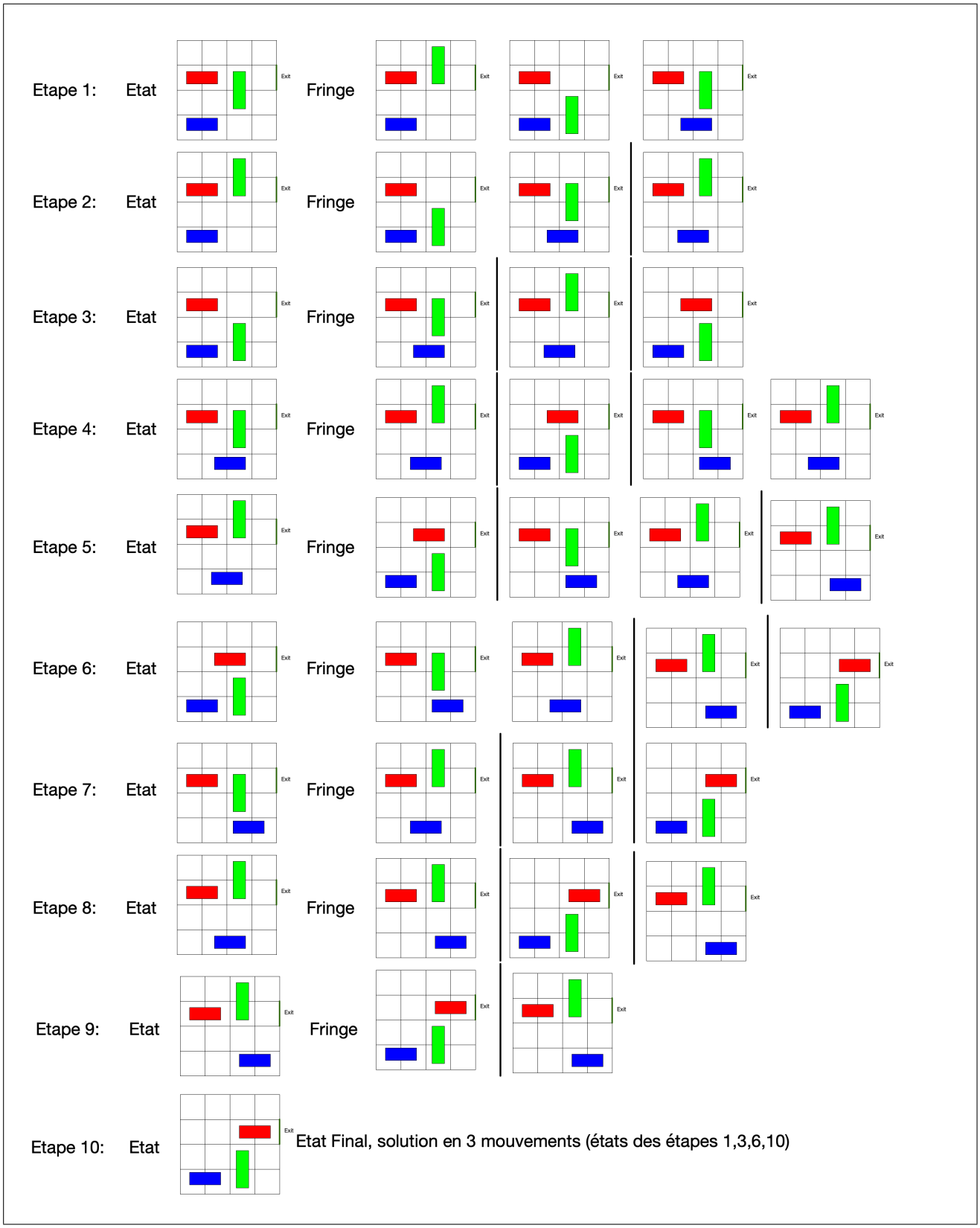


Formellement, une situation du jeu est représentée par un plateau de taille $n \times n$ et contient m véhicules occupant 2 cases (dont la voiture rouge), et p véhicules occupant 3 cases.

1. Formalisez ce problème comme un *problème de recherche* en identifiant bien quels sont les états, les actions, la fonction de transition, et la fonction de coût de votre modèle. Il n'est pas nécessaire de donner une formalisation mathématique stricte. L'important est que je comprenne clairement vos différentes entités.
2. Donnez une borne supérieure raisonnable sur le nombre d'états de votre formulation, ainsi qu'une borne supérieure raisonnable sur le nombre d'actions qui sont possibles à un état (*branching factor*). Veillez à bien préciser quelles approximations vous faites. Exprimez votre solution sur base de la taille du plateau ($n \times n$) ainsi que sur base du nombre de véhicules de chaque catégorie (m et p).
3. Déterminez la séquence d'actions qui sera obtenue en exécutant une *recherche en largeur* jusqu'à la résolution du problème pour la situation de l'image de droite. Indiquez chaque étape de l'algorithme. A chaque étape, veillez à bien indiquer votre état actuel, ainsi que les nœuds présents dans la liste des états candidats (*fringe*). Vous pouvez considérer le problème comme résolu lorsque la case à gauche d'*exit* est touchée par la voiture rouge dans un état. De plus, considérez une *recherche en graphe* (et non en arbre). Précisément, un état ne sera pas ajouté dans la liste des états candidats (*fringe*) si et seulement s'il a déjà été visité. Afin d'éviter de vous faire dessiner un grand nombre de schémas, un patron d'exécution vous est mis disponible aux pages suivantes. Vous pouvez représenter un état seulement par une représentation visuelle. Barrez les cases qui ne sont pas nécessaires.
4. Dans une situation générale de plateau $n \times n$, avez -vous la garantie que votre algorithme trouve la solution optimale ? Expliquez brièvement votre raisonnement.
5. Proposez une heuristique *admissible*, *non-triviale*, et de meilleure qualité que la distance de Manhattan (*voiture rouge* $\rightarrow$ *sortie*) pour ce problème. Justifiez brièvement l'admissibilité de votre heuristique.
6. Supposez qu'on exécute une recherche $A^\star$ en utilisant une heuristique admissible, selon le même scénario que précédemment. Avez-vous la garantie que la solution optimale sera obtenue ?

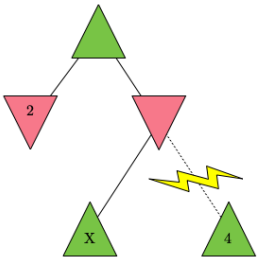
Solution:

1. Proposition d'une formulation (des variations sont possibles) :
   — **Etat** : la position actuelle de chaque voiture et une indication sur leur orientation (verticale ou horizontale). Cela donne une une coordonnée $(x, y) \in \{1, \ldots, n\}^2$ pour chaque voiture, une valeur binaire sur l'orientation $o \in \{0, 1\}$. Pour la coordonnée, on peut prendre la case la plus à gauche, ensuite avec la taille de chaque voiture, on sait calculer les coordonnées de chaque case occupée. Un état est final lorsque la voiture rouge (coordonnée de la case $x + 2$) touche la case *exit*.
   — **Action** : il s'agit simplement de faire coulisser une voiture sur une case adjacente vide.
   — **Fonction de transition** : il s'agit de mettre à jour la coordonnée d'une voiture ($x \pm 1$ ou $y \pm 1$) en tenant compte des obstacles.
   — **Fonction de coût** : incrément de 1 pour chaque mouvement.
2. Comme hypothèse (relaxation), je calcule le nombre d'états comme si chaque voiture n'avait aucun obstacle. Ainsi chaque voiture de taille 2 a $n - 1$ positions possibles, et chaque voiture de taille 3 a $n - 2$ positions possibles. Cela donne $(n - 1)^m(n - 2)^p$ états. Concernant le nombre d'actions, chaque voiture a deux déplacements (sans tenir compte des obstacles et des bords). On a $m + p$ véhicules, et donc $2(m + p)$ actions.
3. La question ne précisait pas comment gérer les égalités en cas d'une même priorité (même ancienneté du noeud), dès lors vous étiez libres de faire le choix que vous souhaitiez. De plus, notez bien qu'on exécute une recherche en graphe, et que dès lors, on retient les états déjà vus. L'exécution est présentée plus bas. Les lignes verticales regroupent les noeuds à priorité égale.
4. Oui, car on exécute une recherche en largeur et que les coûts sont unitaires.
5. Le plus simple est de prendre la distance de Manhattan entre la voiture rouge et la sortie, en faisant +1 pour chaque véhicule qui est sur le chemin. L'heuristique est admissible car la distance de Manhattan l'est, et que pour chaque obstacle, on doit faire obligatoirement au moins une action pour dégager le véhicule.
6. Non,car on exécute une recherche en graphe. Dans cette situation, l'heuristique doit être également consistante pour garantir l'optimalité.

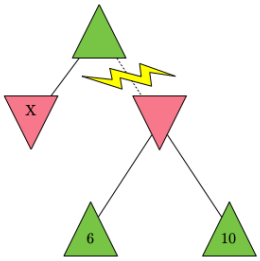

Etape 1: Etat
Fringe
Etape 2: Etat
Fringe
Etape 3: Etat
Fringe
Etape 4: Etat
Fringe
Etape 5: Etat
Fringe
Etape 6: Etat
Fringe
Etape 7: Etat
Fringe
Etape 8: Etat
Fringe
Etape 9: Etat
Fringe
Etape 10: Etat
Etat Final, solution en 3 mouvements (états des étapes 1,3,6,10)

Question 3 (5 points)

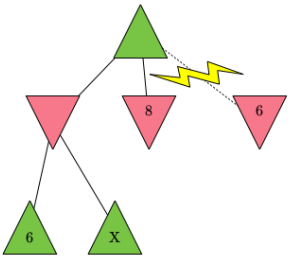
Considérons un jeu déterministe, à deux joueurs (MAX et MIN), tour par tour, à information parfaite, et à somme nulle. La résolution se fait via un algorithme *minimax*, avec le noeud de gauche qui est exploré en priorité (même convention que celle utilisée tout au long du cours). Pour cette question, différents arbres de recherche vous seront présentés. Seulement la valeur des noeuds terminaux est indiquée. Pour chaque arbre, une branche est identifiée pour être éventuellement élaguée via un *alpha-beta pruning* (en pointillé) et la valeur d'un noeud de l'arbre est indéterminée (notée x). Il vous est demandé de dire pour quelles valeurs de x, l'élagage sera effectué. A titre d'exemple, pour la situation présentée ci-dessous (Cas 0), l'élagage ne sera effectué que si $x \leq 2$. Si aucun élagage n'est possible quelque soit la valeur de x, indiquez *jamais*, si l'élagage sera toujours effectué quelque soit la valeur de x, indiquez *toujours*. Vous avez 5 situations à résoudre. Il n'est pas nécessaire de justifier votre réponse.



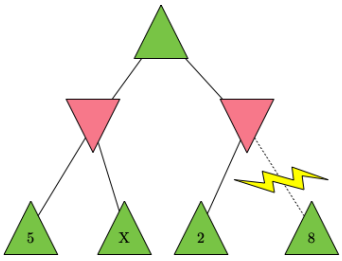
Cas 0: arbre d'exemple



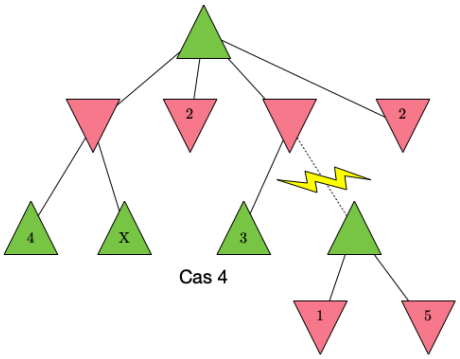
Cas 1



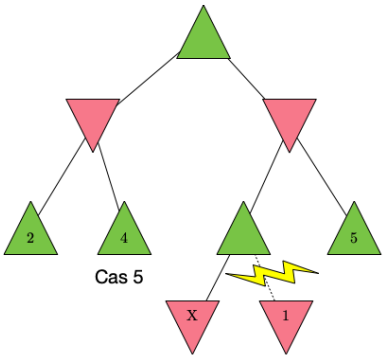
Cas 2



Cas 3



Cas 4



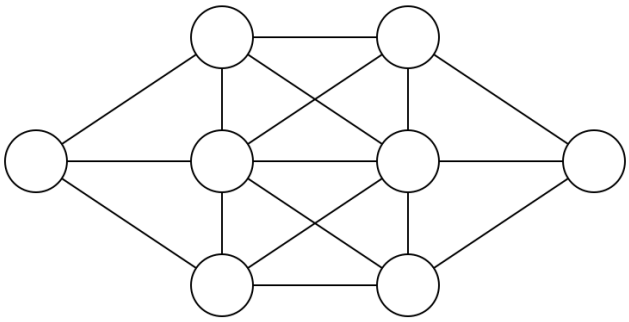
Cas 5

Solution: Note : on accepte également les inégalités strictes.

- Cas 1 : jamais
- Cas 2 : jamais
- Cas 3 : $x \geq 2$
- Cas 4 : $x \geq 3$
- Cas 5 : jamais

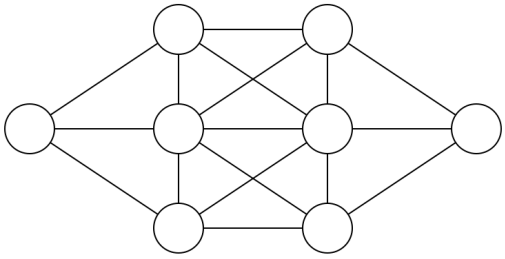
Question 4 (10 points)

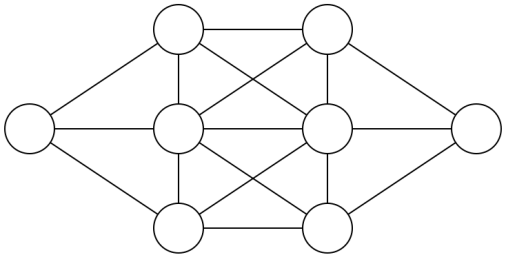
Soit le *problème de satisfaction combinatoire* suivant : étant donné la figure suivante, on souhaite placer les chiffres de 1 à 8 sur les noeuds du graphe de sorte qu'aucun nombre ne soit connecté à un nombre qui lui est consécutif (supérieurement ou inférieurement). Ainsi, le chiffre 5 ne pourra pas être adjacent au chiffre 4 ou au chiffre 6.

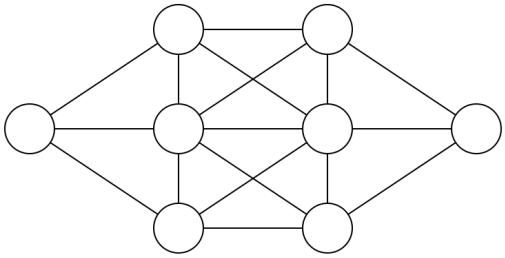


Pour cette question, il ne vous est pas demandé de résoudre complètement ce problème, mais simplement de proposer une résolution basée sur la recherche locale, et d'exécuter quelques étapes de l'algorithme.

1. Proposez un modèle de recherche locale en vue de résoudre ce problème. En particulier, veillez à bien définir les éléments suivants :
   (a) Votre espace de recherche. Indiquez le plus précisément possible quelle est sa taille.
   (b) Les contraintes dures et molles relatives à l'espace de recherche que vous avez considéré.
   (c) Une méthode pour obtenir une solution initiale.
   (d) Votre fonction d'évaluation pour caractériser la qualité d'une solution.
   (e) Votre fonction de voisinage. Indiquez le plus précisément possible quelle est la taille du voisinage.
2. Pour la résolution, vous décidez d'utiliser un algorithme de type *hill climbing* qui prend n'importe quel voisin améliorant. Exécutez deux itérations d'une recherche locale sur base de votre formulation. Dans les figures ci-dessous, illustrez (a) votre solution initiale, (b) la solution obtenue après la première itération, (c) la solution obtenue après la deuxième itération. Pour ces trois solutions, indiquez la valeur de votre fonction d'évaluation. Pour vos deux itérations, veillez à bien indiquer les voisins générés, ainsi que celui qui a été sélectionné.



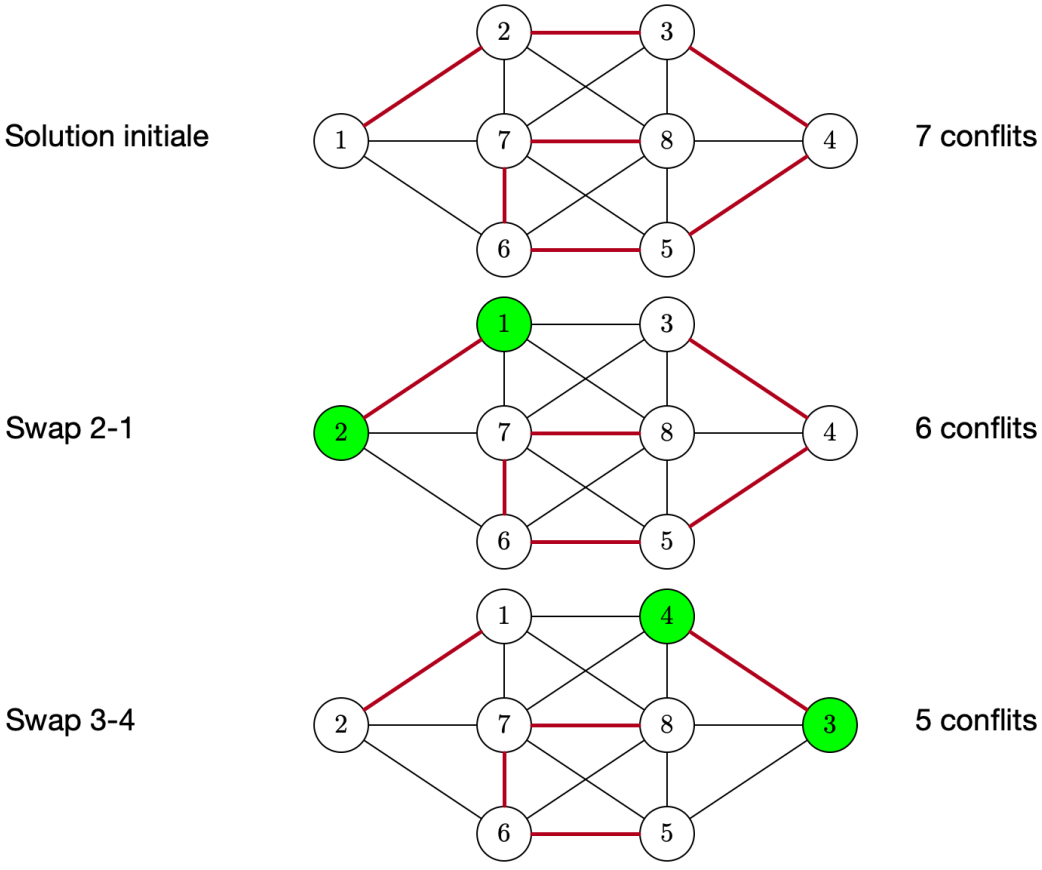




3. Proposez une méthode pour éviter que votre algorithme se retrouve bloqué dans un minimum local.

Solution: Plusieurs solutions sont possibles. Le plus important est que le modèle soit cohérent dans sa globalité. Par exemple, vous ne pouvez pas définir un voisinage qui viole une contrainte dure que vous avez définie. Voici un exemple de modèle et de solution.

1. (a) Les nombres de 1 à 8 sont répartis dans le graphe sans répétition. On a alors un espace de 8! solutions.
   (b) La contrainte de non-répétition est dure, et celles des arêtes (pas de nombre consécutif) est molle.
   (c) Simplement générer une permutation aléatoire.
   (d) Le nombre de contraintes molles non-satisfaites.
   (e) Faire une permutation de 2 nombres. Cela donne $(8 \times 7)/2$ voisins ($\mathcal{O}(n^2)$ avec n le nombre d'éléments à placer, en supposant un graphe quelconque). Notez que la permutation $x \rightarrow y$ est la même que $y \rightarrow x$.
2. On va choisir un voisin améliorant à chaque fois. Les arêtes rouges indiquent les conflits, et les noeuds en vert indiquent les valeurs permutées. Les itérations sont illustrées plus bas.
3. On peut par exemple exécuter des redémarrages aléatoires.



Question 5 (5 points)

Pour l'entièreté des questions, on se situe dans la logique des propositions, sauf si le contraire est explicitement mentionné. Indiquez si les assertions suivantes sont vraies ou fausses. Si elles sont fausses, justifiez succinctement (deux phrases au maximum) pourquoi ou donnez un contre-exemple. Il est nécessaire de fournir cette justification pour avoir les points.

1. Soit la base de connaissance KB : $\{\neg\mathsf{EchecProjet}, (\mathsf{EchecProjet} \wedge \neg\mathsf{ReussiteExamen}) \equiv \mathsf{Tristesse}\}$. La formule $f_1 : \neg\mathsf{Tristesse}$ est une conséquence logique de KB.

2. Soit la même base de connaissances KB. La formule $f_2 : \neg\mathsf{EchecProjet} \wedge \mathsf{Tristesse}$ est en contingence avec KB.

3. Soit la même base de connaissances KB. La formule $f_3 : \mathsf{ReussiteExamen} \vee \mathsf{Tristesse}$ est en contradiction avec KB.

4. Soit la règle d'inférence $\frac{f \rightarrow g,\ \neg f \rightarrow \neg g}{f \equiv g}$. Cette règle est cohérente (*sound*).

5. Soit la règle d'inférence $\frac{f,\ f \rightarrow g}{g}$. Cette règle est complète (*complete*).

6. Supposons un agent logique basé sur un algorithme d'inférence complet mais non-cohérent et sur une base de connaissance KB. Vous avez la garantie que toutes les formules qui sont en conséquence logique avec KB peuvent être générées via l'algorithme d'inférence.

7. Supposons un agent logique basé sur un algorithme d'inférence cohérent mais non-complet et sur une base de connaissance KB. Vous effectuez une requête ASK à cet agent, qui vous retourne une formule f. Vous ne pouvez pas être certain que cette formule est une conséquence logique de KB.

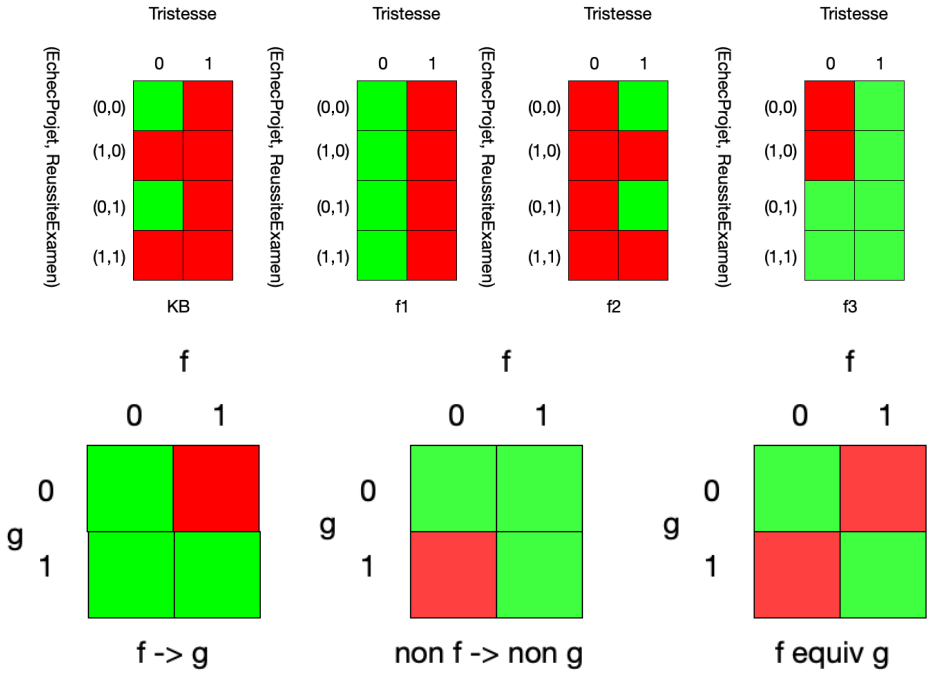
8. La logique des propositions avec les clauses de Horn est utile car elle permet de rendre la règle du Monus Ponens cohérente.

9. La clause $f_4 : \neg P_1 \vee \neg P_2 \vee P_3 \vee P_4$ est une clause de Horn.

10. Afin d'exprimer la connaissance *il fait froid au Québec*, il serait pertinent d'utiliser une logique probabiliste.

Solution:

1. Vrai : les modèles de KB sont inclus dans les modèles de f_1.
2. Faux : f_2 est en contradiction avec KB (aucun modèle en commun).
3. Faux : f_3 est en contingence avec KB (intersection non nulle et non complète).
4. Vrai : les modèles des prémices sont inclus dans ceux de la conclusion.
5. Faux : il s'agit du Modus Ponens (prouvé en cours).
6. Vrai : car complet.
7. Faux : on est certain car cohérent.
8. Faux : elle rend le Podus Ponens complet.
9. Faux : on doit à avoir au plus un littéral positif.
10. Faux : aucun probabilité rentre en jeu. Une logique floue est plus pertinente.



Question 6 (10 points)

Une compagnie de gestion de produits financiers décide de créer un outil de prédiction de la valeur future d'une action. On suppose que cette estimation prend une valeur réelle. Une action est représentée par deux caractéristiques : sa valeur actuelle, et la variance de l'action sur la dernière semaine. Pour réaliser cet objectif, la compagnie décide de construire un réseau de neurones assez minimaliste. L'architecture est la suivante :

— $L^{[0]}$: à déterminer par vous même étant donné la description du problème.
— $L^{[1]}$: 1 neurone, avec une activation `ReLU`.
— $L^{[2]}$: à déterminer par vous même étant donné la description du problème.

Sur base de cette architecture, répondez aux questions suivantes :

1. Donnez le nombre de neurones qui composent les couches $L^{[0]}$ et $L^{[2]}$.
2. Donnez les équations de ce réseau pour obtenir une prédiction à partir d'un ensemble de données en entrée. Utilisez la notation matricielle. Veillez à bien définir la fonction d'activation pour la dernière couche.
3. Donnez les dimensions des différentes matrices et vecteurs X, Z, A, W, b, Y de votre réseau. Considérez 100 données en entrée.
4. Donnez le nombre de paramètres devant être appris dans ce réseau.
5. Donnez une fonction d'écart adapté au problème. Formalisez la mathématiquement.
6. Représentez les opérations effectuées dans ce réseau pour calculer la valeur de la fonction d'écart par un *graphe de dépendance*. Au niveau de la granularité des opérations, vous pouvez considérer une *combinaison linéaire*, une *application d'une fonction d'activation*, et le *calcul de la fonction d'écart* comme opérations élémentaires.
7. A l'initialisation, on suppose que tous les paramètres de la matrice $W^{[1]}$ ont été initialisés à une valeur de 2, tous les paramètres du vecteur $b^{[1]}$ ont été initialisés à 0, tous les paramètres de la matrice $W^{[2]}$ ont été initialisés à 1, et que tous les paramètres du vecteur $b^{[2]}$ ont été initialisés à 5. Considérez une action dont la valeur actuelle est de 12, et dont la variance est de 20. La valeur future connue (donnée historique, y) est de 75. Effectuez une passe en avant afin d'obtenir : (1) la prédiction pour cette donnée, (2) l'évaluation de la fonction d'écart pour cette donnée.
8. Étant donné cette évaluation de la fonction d'écart, effectuez une passe en arrière pour obtenir le gradient (c-à-d, une valeur pour chaque dérivée partielle des paramètres). Vous êtes libres d'utiliser directement n'importe quel résultat qui a été démontré dans les diapositives du cours pour le calcul de vos dérivées partielles. **Indice :** procédez avec la règle du chaînage.
9. Finalement, utilisez l'information de votre gradient pour effectuer une étape de descente de gradient. Utilisez un taux d'apprentissage de 0.01.

Solution:

1. $L^{[0]} = 2$ (2 caractéristiques réelles) et $L^{[2]} = 1$ (une valeur réelle).
2. Pour la dernière couche, on considère la fonction d'activation identité, car on souhaite obtenir une valeur réelle.

$$A^{[0]} = X$$
$$Z^{[1]} = W^{[1]}A^{[0]} + b^{[1]}$$
$$A^{[1]} = \mathsf{ReLU}\big(Z^{[1]}\big)$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = Z^{[2]}$$
$$\hat{Y} = A^{[2]}$$

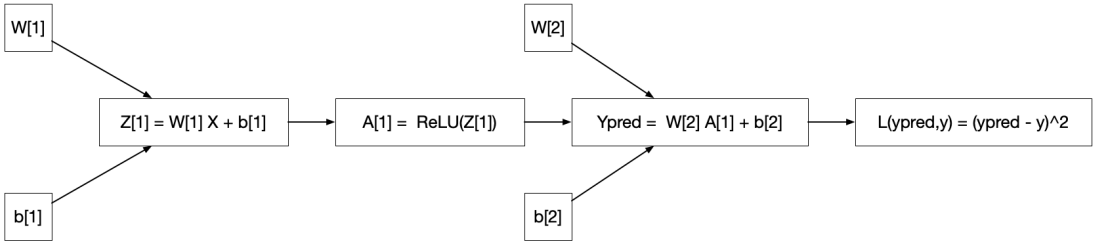
3. Les matrices sons les suivantes :

$X = A^{[0]} = n^{[0]} \times m = 2 \times 100$
$Z^{[1]} = A^{[1]} = n^{[1]} \times m = 1 \times 100$
$\hat{Y} = Z^{[2]} = A^{[2]} = n^{[2]} \times m = 1 \times 100$
$W^{[1]} = n^{[1]} \times n^{[0]} = 1 \times 2$
$W^{[2]} = n^{[2]} \times n^{[1]} = 1 \times 1$
$b^{[1]} = n^{[1]} \times 1 = 1 \times 1$
$b^{[2]} = n^{[2]} \times 1 = 1 \times 1$

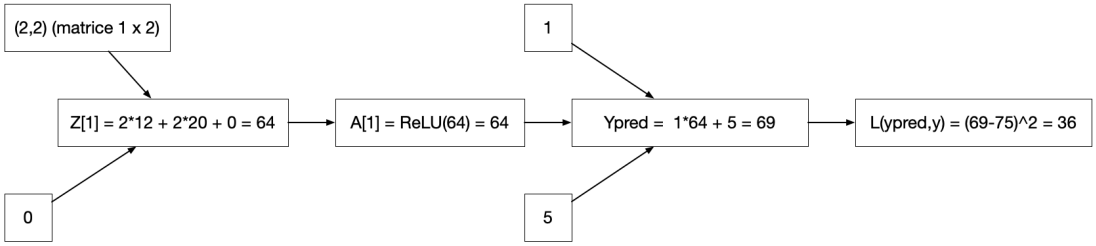
4. On a $2 + 1 + 1 + 1 = 5$ paramètres.
5. On a une régression continue, dès lors, l'erreur quadratique moyenne est adaptée.

$$L(\hat{y}^{(i)}, y^{(i)}) = (\hat{y}^{(i)} - y^{(i)})^2$$

6. Voici le graphe de dépendance.



7. La prédiction est $\hat{y} = 69$ et l'erreur est de $L(\hat{y}, y) = 36$.



8. Par la règle du chaînage, on a

$$\frac{\partial L(\hat{y}, y)}{\partial W[2]} = \frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial W[2]}$$

$$\frac{\partial L(\hat{y}, y)}{\partial b[2]} = \frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial b[2]}$$

$$\frac{\partial L(\hat{y}, y)}{\partial W[1]} = \frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial A[1]} \times \frac{\partial A[1]}{\partial Z[1]} \times \frac{\partial Z[1]}{\partial W[1]}$$

$$\frac{\partial L(\hat{y}, y)}{\partial b[1]} = \frac{\partial L(\hat{y}, y)}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial A[1]} \times \frac{\partial A[1]}{\partial Z[1]} \times \frac{\partial Z[1]}{\partial b[1]}$$

9. En appliquant les règles de base d'analyse, on peut calculer chaque dérivée partielle.

$$\frac{\partial L(\hat{y}, y)}{\partial \hat{y}} = 2(\hat{y} - y)$$

$$\frac{\partial \hat{y}}{\partial W[2]} = A[1]$$

$$\frac{\partial \hat{y}}{\partial b[2]} = 1$$

$$\frac{\partial \hat{y}}{\partial A[1]} = W[2]$$

$$\frac{\partial A[1]}{\partial Z[1]} = 1 \text{ ou } 0 \text{ suivant la valeur de } Z[1]$$

$$\frac{\partial Z[1]}{\partial W[1]} = X$$

$$\frac{\partial Z[1]}{\partial b[1]} = 1$$

10. Ensuite, on remplace juste avec les valeurs numériques. Notez bien que $W^[1]$ est une matrice de dimension 1×2.

$$\frac{\partial L(\hat{y}, y)}{\partial W[2]} = 2(69 - 75) \times 64 = -768$$

$$\frac{\partial L(\hat{y}, y)}{\partial b[2]} = 2(69 - 75) \times 1 = -12$$

$$\frac{\partial L(\hat{y}, y)}{\partial W[1]} = 2(69 - 75) \times 1 \times 1 \times (12, 20) = (-144, -240)$$

$$\frac{\partial L(\hat{y}, y)}{\partial b[1]} = 2(69 - 75) \times 1 \times 1 \times 1 = -12$$

11. La descente de gradient se fait sur chaque dérivée partielle.

$$W^{[2]} = W^{[2]} - \alpha \frac{\partial L(\hat{y}, y)}{\partial W[2]} = 1 - 0.01 \times -768 = 8.68$$

$$b^{[2]} = b^{[2]} - \alpha \frac{\partial L(\hat{y}, y)}{\partial b[2]} = 5 - 0.01 \times -12 = 5.12$$

$$W^{[1]} = W^{[1]} - \alpha \frac{\partial L(\hat{y}, y)}{\partial W[1]} = (2, 2) - 0.01 \times (-144, -240) = (3, 44, 4, 4)$$

$$b^{[1]} = b^{[1]} - \alpha \frac{\partial L(\hat{y}, y)}{\partial b[1]} = 0 - 0.01 \times -12 = 0.12$$