INF8175 : Examen final (Automne 2023) - Quentin Cappart	Note finale ↓
Date : 21/12/2023 Nombre de questions : **7** Total des points : **40**	
Matricule : Nom :	Temps : 2h30 heures

Question:	1	2	3	4	5	6	7	Total
Points:	0	10	6	6	6	6	6	40
Score:								

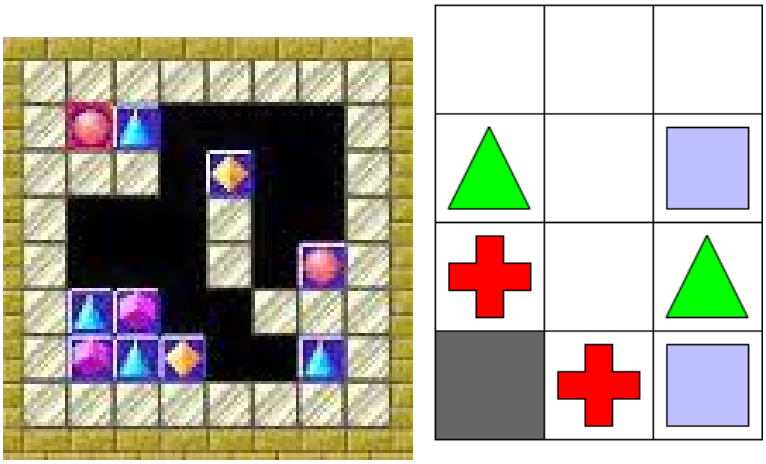
Instructions

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)

Après une session d'examen bien remplie, vous allez fêter le réveillon chez vos grands-parents En fouillant dans le grenier afin de trouver des vieilles affaires, vous tombez sur la console Atari de votre grand-père, ainsi que sur le jeu *Puzznic*. Une illustration de ce jeu est proposée sur la figure ci-dessous (à gauche). Brièvement, le jeu consiste à faire bouger un ensemble de pièces afin de les placer à côté (au dessus, en dessous, à gauche, ou à droite) d'une autre pièce ayant le même motif. Chaque motif est toujours présent sur un nombre pair de pièces, et les mouvements autorisés sont de déplacer une pièce à gauche ou à droite. Un point important est que le jeu est sujet à la gravité : une pièce va tomber si elle ne se trouve sur aucun support. A titre d'exemple sur la figure de gauche, le triangle bleu du dessus va tomber sur le losange jaune s'il est déplacé à droite. Une autre conséquence de la gravité est qu'il ne sera jamais possible de déplacer une pièce plus haut que sa position initiale. Finalement, lorsque deux pièces identiques sont adjacentes, elles disparaissent de la grille. Le jeu est gagné lorsque toutes les pièces sont enlevées du plateau.

Une solution valide à ce problème est une séquence d'actions permettant de gagner le jeu, et une solution optimale est la séquence comportant le moins d'actions. Suite à ce que vous avez 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 d'une situation simplifiée est proposée à droite. La case grise indique un mur infranchissable.



Formellement, une situation du jeu est représentée par une grille de largeur n, de hauteur m et contient p pièces, commençant chacune à une position (x_p, y_p). Etant donné que chaque pièce a une et une seule autre pièce identique, p est pair. Chaque mur est infranchissable.

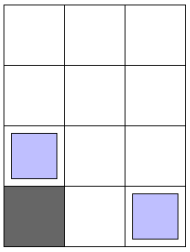
1. *(1 point)* Formalisez ce problème comme un *problème de recherche* en identifiant bien quels sont les états, les actions, la fonction de transition, et la fonction de coût de votre modèle. Il n'est pas nécessaire de donner une formalisation mathématique stricte. L'important est que je comprenne clairement vos différentes entités. Concernant la transition, considérez que les pièces au même motif disparaissent directement lorsqu'elles sont adjacentes. Ainsi, vous n'aurez jamais un état ayant deux pièces identiques adjacentes (elles auront disparues).
2. *(1 point)* 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 de la grille ($n \times m$) ainsi que sur base du nombre de pièces (p) et de leur position initiale (x_p, y_p).
3. *(3 points)* Déterminez la séquence d'actions qui sera obtenue en exécutant une *recherche en profondeur* 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*). Dans le cas d'une priorité égale lors de l'ajout d'un successeur dans la *fringe*, ajoutez d'abord ceux impliquant la croix rouge, puis le triangle vert, puis le rectangle bleu. En cas d'une égalité au sein d'un motif, déplacez la pièce la plus à droite. Vous pouvez considérer le problème comme résolu lorsque toutes les pièces ont disparu de la grille. De plus, considérez une *recherche en graphe* (et non en arbre). Précisément, un état ne sera pas

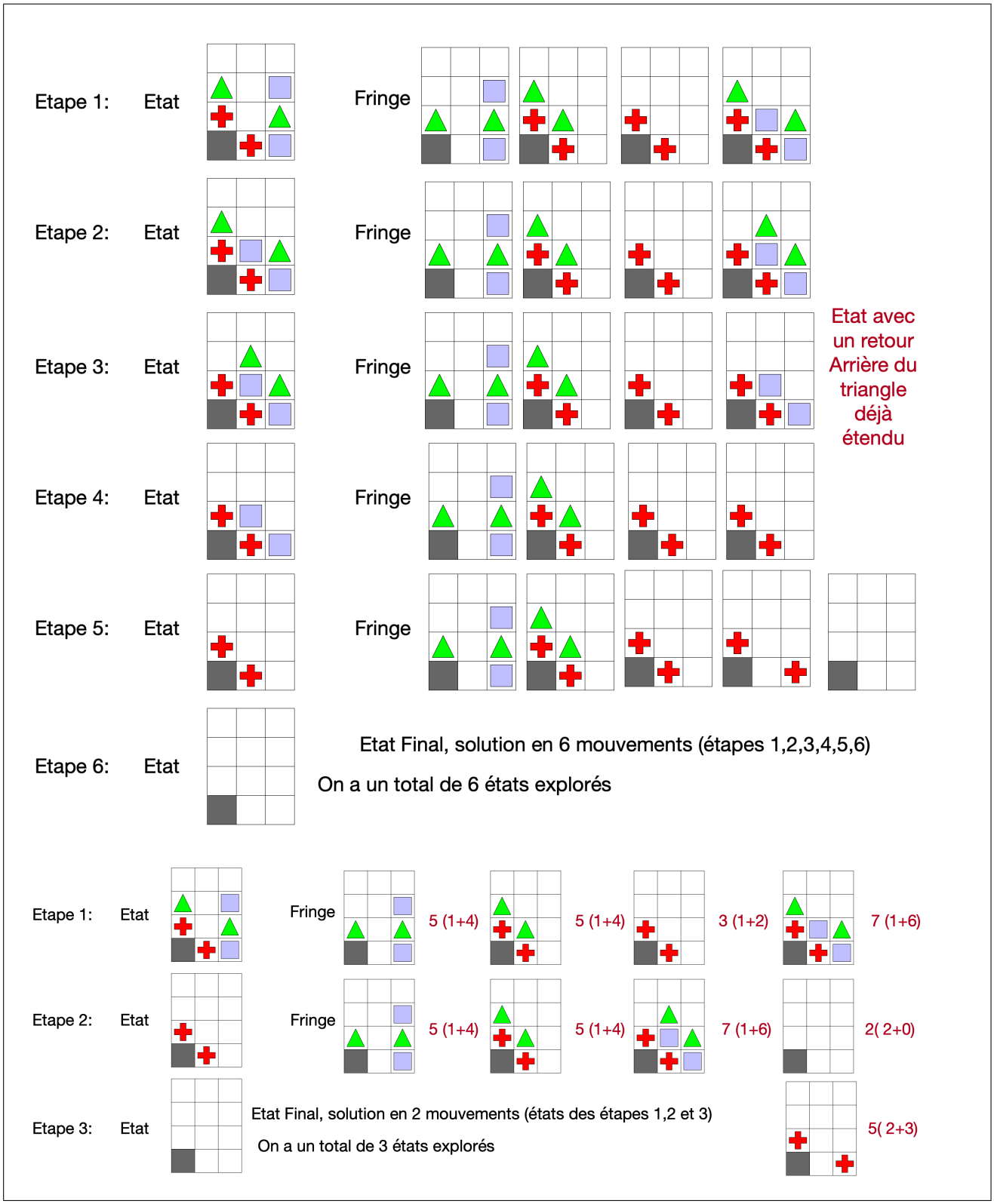
ajouté dans la liste des états candidats (*fringe*) si et seulement s'il a déjà été visité. Afin d'éviter de vous faire dessiner un grand nombre de schémas, un patron d'exécution vous est mis disponible aux pages suivantes. Vous pouvez représenter un état seulement par une représentation visuelle. Barrez les cases qui ne sont pas nécessaires.

4. *(1 point)* Dans une situation générale de plateau $n \times m$, avez vous la garantie que votre algorithme trouve la solution optimale ? Expliquez brièvement votre raisonnement.
5. *(3 points)* On considère maintenant une heuristique consistant à prendre la somme des distances de Manhattan entre chaque pièce de même motif. A titre d'exemple, l'évaluation de l'heuristique à la figure d'exemple (de droite) est de 7 (2 pour les croix, 3 pour les triangles, et 2 pour les carrés). Déterminez la séquence d'actions qui sera obtenue en exécutant une *recherche A** jusqu'à la résolution du problème pour la situation de l'image de droite.
6. *(1 point)* Est-ce que cette heuristique est admissible ? Justifiez brièvement votre réponse.

Solution:

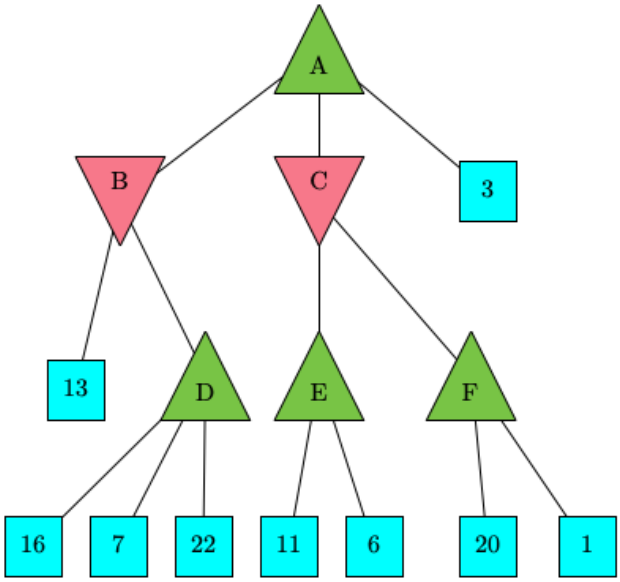
1. Proposition d'une formulation (des variations sont possibles) :
   - **Etat** : la position actuelle de chaque pièce initiale et une indication binaire si elle est encore sur le plateau ou pas. Cela donne une une coordonnée $(x, y) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$ pour chaque pièce, et une valeur binaire sur sa présence $g \in \{0, 1\}$.
   - **Action** : il s'agit simplement de déplacer une pièce à gauche ou à droite
   - **Fonction de transition** : il s'agit de mettre à jour la coordonnée d'une pièce (en tenant compte de la gravité) et de vérifier si elle doit disparaître en cas d'adjacence.
   - **Fonction de coût** : incrément de 1 pour chaque mouvement.
2. On sait qu'une pièce ne sera jamais plus haute que sa hauteur initiale (coordonnée y_p pour une pièce p). Comme hypothèse (relaxation), je calcule le nombre d'états comme si chaque voiture n'avait aucun obstacle. Ainsi, chaque pièce a $y_p \times n$ positions possibles. Vu qu'on a p pièces, on a $(y_1 \times n) \times \cdots \times (y_p \times n)$ états. Concernant le nombre d'actions, chaque pièce a deux déplacements (sans tenir compte des obstacles et des bords). On a donc $2p$ actions.
3. Notez bien qu'on exécute une recherche en graphe, et que dès lors, on retient les états déjà vus. Attention à ne pas confondre le nombre d'états visités et le coût de la solution retournée. L'exécution est montrée plus bas.
4. Non, car on effectue une recherche en profondeur.
5. L'exécution est montrée plus bas. Attention à ne pas oublier le coût passé et le coût heuristique.
6. Non, voici un contre exemple. Le coût réel optimal est de 1 mouvement, et notre heuristique donne une valeur de 3. On a donc surestimé le coût.




Etape 1: Etat
Fringe
Etape 2: Etat
Fringe
Etape 3: Etat
Fringe
Etat avec un retour Arrière du triangle déjà étendu
Etape 4: Etat
Fringe
Etape 5: Etat
Fringe
Etape 6: Etat
Etat Final, solution en 6 mouvements (étapes 1,2,3,4,5,6)
On a un total de 6 états explorés
Etape 1: Etat
Fringe
5 (1+4)
5 (1+4)
3 (1+2)
7 (1+6)
Etape 2: Etat
Fringe
5 (1+4)
5 (1+4)
7 (1+6)
2(2+0)
Etape 3: Etat
Etat Final, solution en 2 mouvements (états des étapes 1,2 et 3)
On a un total de 3 états explorés
5(2+3)

Question 3 (6 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). Voici un arbre représentant les différentes actions possibles. Les carrés bleus indiquent un score terminal, atteint par le joueur MAX.



1. *(1 point)* Donnez la valeur minimax du noeud A.
2. *(3 points)* Sur le schéma, barrez toutes les branches qui seront élaguées lors de l'exécution d'un *alpha-beta pruning*.
3. *(1 point)* Au lieu d'explorer en priorité le noeud de gauche, vous construisez une heuristique h pour déterminer l'ordre de considération des noeuds avant d'exécuter l'*alpha-beta pruning*. Est-ce que la qualité de votre heuristique va impacter l'action finale qui sera choisie. Dans le cas positif, fournissez un exemple illustratif. Dans le cas contraire, justifiez brièvement votre raisonnement.
4. *(1 point)* On repend l'heuristique h. Est-ce que la qualité de votre heuristique va impacter le nombre de noeuds élagués par l'*alpha-beta pruning* ? Dans le cas positif, justifiez brièvement votre raisonnement, dans le cas contraire, fournissez un arbre de recherche minimaliste servant de contre-exemple.

Solution:

1. La valeur du noeud A est de 13.
2. les noeuds 7, 22, et F (et ses descendants) seront élagués.
3. Non, l'*alpha-beta pruning* ne fait aucune approximation et va donner la valeur minimax optimale, quelque soit l'ordre de considération des noeuds. La différente se fait sur l'efficacité de l'élagage.
4. Faux, un contre-exemple simple est donné dans les diapositives du cours.

Question 4 (6 points)

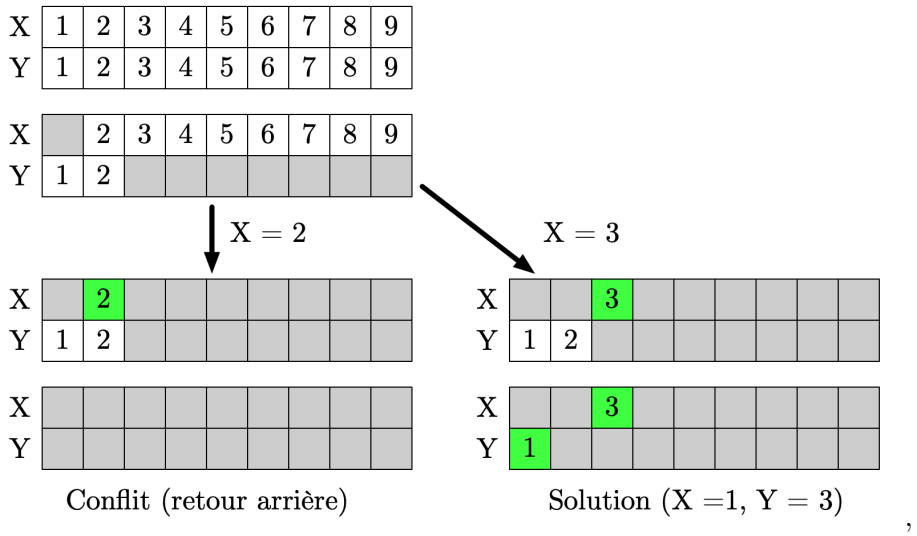
Soit le problème de satisfaction combinatoire ci-dessous.

$$
\begin{array}{lll}
\text{satisfy} & & \\
\text{subject to} & x \neq y & (c_1) \\
& x > y^2 & (c_2) \\
& x + y \neq 3 & (c_3) \\
& x, y \in \{1, \ldots, 9\} &
\end{array}
$$

1. *(1 point)* Indiquez la taille de l'espace de recherche si on souhaite réaliser une recherche exhaustive.
2. *(4 points)* Il vous est demandé de résoudre ce problème en utilisant une résolution basée sur la programmation par contraintes. Spécifiquement, il vous est demandé d'appliquer une recherche à retours-arrières (*backtracking search*) et d'appliquer l'algorithme du point fixe à chaque noeud de l'arbre de recherche, jusqu'à ce qu'une solution faisable soit trouvée. Utilisez la cohérence d'arc (*arc consistency*) pour la propagation de chaque contrainte. Représentez visuellement l'arbre de recherche généré, et à chaque noeud de recherche, veillez à bien indiquez : (1) l'état des domaines avant la propagation, et l'état des domaines après la propagation. Vous n'êtes pas obligés d'écrire toutes les étapes du point de fixe. Pour le branchement, considérez d'abord la variable x puis y, et prenez en priorité la valeur la plus petite du domaine.
3. *(1 point)* Combien de noeuds avez-vous du explorer avant de trouver une solution faisable ?

Solution:

1. Il s'agit du produit cartésien des domaines : $|D(x)| \times |D(y)| = 9 \times 9 = 81$.
2. L'illustration de l'arbre de recherche est ci-dessous.
3. L'arbre de recherche comporte 3 noeuds.



,

Question 5 (6 points)

Pour cette question, on se situe dans la logique des propositions. Considérons un agent logique ayant les formules suivantes dans sa base de connaissances :

— *S'il pleut, Quentin prend son parapluie.*
— *Si Quentin prend son parapluie, il n'est pas mouillé.*
— *S'il ne pleut pas, Quentin n'est pas mouillé.*

On pose la requête ASK suivante : *Quentin n'est pas mouillé*. On veut savoir si cette formule est une conséquence logique de la base de connaissances. Pour cela, il vous est demandé d'appliquer l'algorithme de résolution.

1. *(1 point)* Exprimez ces 4 formules en formules valides dans la logique des propositions.
2. *(1 point)* Exprimez ces formules en une forme normale conjonctive (CNF).
3. *(4 points)* Appliquez l'algorithme de résolution pour prouver la conséquence logique. Veillez bien à indiquer chaque étape de votre algorithme. Vous êtes libres de choisir votre propre ordre de considération pour les inférences.

Solution:

1. J'introduis trois symboles : R, W, U pour *raining*, *Quentin is wet*, and *Quentin with umbrella*. On peut encoder les formules comme suit :
   — $f_1 : R \rightarrow U$.
   — $f_2 : U \rightarrow \neg W$.
   — $f_3 : \neg R \rightarrow \neg W$.
   — $o : \neg W$.
2. On utilise la simplification d'une implication $(u \rightarrow v) \equiv (\neg u \vee v)$. Ce qui donne :
   — $f_1 : \neg R \vee U$.
   — $f_2 : \neg U \vee \neg W$.
   — $f_3 : R \vee \neg W$ (la double négation s'annule).
   — $o : \neg W$.
3. On doit prouver que la base de connaissances $(f_1 \wedge f_2 \wedge f_3)$ est en contradiction avec $\neg o$ (c-à-d W). On applique ensuite la règle de résolution jusqu'à générer la clause vide, indiquant une contradiction. Voici une exécution possible.
   — $\frac{W \qquad R \vee \neg W}{R}$. On génère la clause R.
   — $\frac{R \qquad \neg R \vee U}{U}$. On génère la clause U.
   — $\frac{U \qquad \neg U \vee \neg W}{\neg W}$. On génère la clause $\neg W$.
   — $\frac{W \qquad \neg W}{\emptyset}$. On génère la clause vide, on a une contradiction. *Quentin n'est pas mouillé* est donc une conséquence logique de la base de connaissances.

Question 6 (6 points)

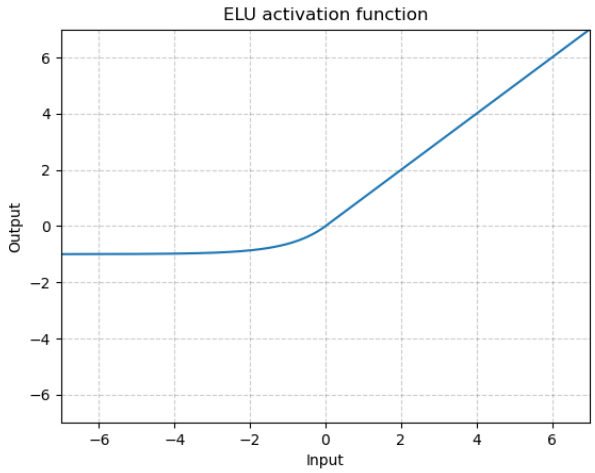
Une association œuvrant dans le soutien des pays en voie de développement décide de mettre un outil de prédiction afin de déceler si une eau est potable ou non. Leur objectif est d'obtenir une probabilité sur le fait qu'un échantillon d'eau soit potable. Un échantillon d'eau est représenté par les 4 *features* suivantes : quantité (mg/L) de calcium, de magnésium, de sodium, et de sulfate. Pour réaliser cet objectif, la compagnie souhaite utiliser un modèle basé sur un réseau de neurones ayant l'architecture suivante :

— $L^{[0]}$: À trouver par vous-même étant donné la description du problème.
— $L^{[1]}$: 3 neurones.
— $L^{[2]}$: 5 neurones.
— $L^{[3]}$: 2 neurones.
— $L^{[4]}$: À trouver par vous-même étant donné la description du problème.

1. *(1 point)* Donnez les dimensions des différentes matrices et vecteurs $X, Z, A, W, b, \hat{Y}$ de votre réseau. Considérez 50 données en entrée.
2. *(1 point)* Donnez le nombre exact de paramètres devant être appris dans ce réseau.
3. *(1 point)* 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 toutes les fonctions d'activation que vous avez considérées pour chacune des couches.
4. *(1 point)* Proposez une fonction de coût adaptée à cette situation. Indiquez l'expression mathématique de cette fonction.
5. *(2 points)* En vous renseignant dans la littérature scientifique, vous découvrez la fonction d'activation `ELU` (définie plus bas), avec α un hyperparamètre qui doit être défini par l'utilisateur. Donnez un avantage et un inconvénient de la fonction d'activation `ELU` par rapport à `ReLU` (*Rectified Linear Unit*). Pour rappel, $\texttt{ReLU}(z) = \max(0, z)$.

$$\texttt{ELU}(z, \kappa) = \begin{cases} z \texttt{ if } z > 0 \\ \alpha(e^z - 1) \texttt{ otherwise} \end{cases}$$

Pour vous aider, cette fonction est illustrée ci-dessous.



Solution:

1. 4 neurones pour la couche d'entrée (car 4 features), et 1 neurone pour la couche de sortie (car une seule prédiction)

2. Les matrices sons les suivantes :

   $X = A^{[0]} = n^{[0]} \times m = 4 \times 50$
   $Z^{[1]} = A^{[1]} = n^{[1]} \times m = 3 \times 50$
   $Z^{[2]} = A^{[2]} = n^{[2]} \times m = 5 \times 50$
   $Z^{[3]} = A^{[3]} = n^{[3]} \times m = 2 \times 50$
   $\hat{Y} = Z^{[4]} = A^{[4]} = n^{[4]} \times m = 1 \times 50$
   $W^{[1]} = n^{[1]} \times n^{[0]} = 3 \times 4$
   $W^{[2]} = n^{[2]} \times n^{[1]} = 5 \times 3$
   $W^{[3]} = n^{[3]} \times n^{[2]} = 2 \times 5$
   $W^{[4]} = n^{[4]} \times n^{[3]} = 1 \times 2$
   $b^{[1]} = n^{[1]} \times 1 = 3 \times 1$
   $b^{[2]} = n^{[2]} \times 1 = 5 \times 1$
   $b^{[3]} = n^{[3]} \times 1 = 2 \times 1$
   $b^{[4]} = n^{[4]} \times 1 = 1 \times 1$

3. On doit compter le nombre de valeurs dans les matrices W et b : 50 paramètres.
4. Pour la dernière couche, on considère la fonction d'activation sigmoide, car on veut obtenir une valeur entre 0 et 1. Pour les couches cachées, n'importe quelle activation non-linéaire (p.e., ReLU) fait l'affaire.

   $A^{[0]} = X^{[l]}$
   `for` l `in` 1 `to` 4 :
   $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$
   $A^{[l]} = g^{[l]}(Z^{[l]})$
   $\hat{Y} = A^{[4]}$

5. Vu qu'on a un problème de classification binaire, on peut utiliser la *binary cross entropy loss*

$$J(W, b) = -\frac{1}{100} \sum_{i=1}^{50} \left(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right) \tag{1}$$

6. Un avantage est que le gradient n'est pas nul dans la zone négative, ce qui permet d'éviter l'effet de neurones mort. Un inconvénient est qu'on doit calculer une exponentielle à chaque activation, ce qui est plus couteux. Un autre inconvénient est qu'on a un hyperparamètre de plus à calibrer.

Question 7 (6 points)

Soit l'ensemble de données suivantes :

— $x^{(1)} = (1, 1)$
— $x^{(2)} = (1, 2)$
— $x^{(3)} = (2, 3)$
— $x^{(4)} = (4, 3)$
— $x^{(5)} = (5, 4)$

Les données sont ainsi non labellisées et sont caractérisées par deux *features*. On souhaite regrouper ces données en deux *clusters*. Pour ce faire, on compte utiliser l'algorithme *k-means*, en considérant la fonction de coût suivante : $J\big(c^{(1)}, \ldots, c^{(5)}, \mu_1, \mu_2\big) = \frac{1}{m} \sum_{i=1}^{m} \big\|x^{(i)} - \mu_{c^{(i)}}\big\|_2$. Pour rappel, $\|z\|_2 = \sqrt{z_1^2 + z_2^2}$, pour un vecteur $z \in \mathbb{R}^2$. À l'initialisation, on pose le centre du premier cluster sur la donnée $x^{(1)}$ et le centre du deuxième sur la donnée $x^{(3)}$.

1. *(1 point)* Représentez graphiquement la situation initiale.
2. *(4 points)* Faites deux itérations complètes de l'algorithme *k-means* sur l'ensemble des données.
3. *(1 point)* Représentez graphiquement la situation après chaque itération, donnez le coût de la solution, et indiquez les valeurs $c^{(1)}, \ldots, c^{(5)}, \mu_1, \mu_2$.

Solution:

Situation initiale :

— $\mu_1, \mu_2 = \big((1, 1), (2, 3)\big)$
— $c = (-1, -1, -1, -1, -1)$ (non assigné). J'accepte aussi $(1, 1, 2, 2, 2)$ si l'étudiant assigne directement au cluster le plus proche.
— $J = 1.23$ (au besoin)

Après la première itération :

— $\mu_1, \mu_2 = \big((1, 1.5), (3.6, 3.3)\big)$
— $c = (1, 1, 2, 2, 2)$
— $J = 0.93$

Après la deuxième itération :

— $\mu_1, \mu_2 = \big((1, 1.5), (3.6, 3.3)\big)$
— $c = (1, 1, 2, 2, 2)$
— $J = 0.93$