Insat Année universitaire : 2017-2018

NOM :	N°CIN :
PRENOM :	N° inscription :
N° salle :	N° Place :

EXAMEN SESSION PRINCIPALE

Matière : INTELLIGENCE ARTIFICIELLE	*Semestre*: Matière semestre 1
Enseignante : Mme M. Chater	*Date* JANVIER 2018
Filière : GL4	*Durée*: 1h30 heure
Nombre de pages : 7	*Documents*: NON autorisés

XXX--XXX-

Exercice 1. Formalisation d'un problème (5 pts)

Nous considérons le problème de pièces à échanger sur une plaquette avec la configuration initiale suivante :

N	N	N	B	B	B	V

Il y a 3 pièces noires (N), 3 pièces blanches (B) et une case vide (V).
Le problème permet les coups suivants :

a) Une pièce peut se déplacer dans la case vide adjacente avec un coût unitaire.

b) Une pièce peut sauter dans la case vide par-dessus deux pièces au maximum avec un coût égal au nombre de pièces par-dessus lesquelles elle a sauté.

L'état solution à obtenir est le suivant :

B	B	B	N	N	N	V

1. Proposer une représentation compacte du problème

………

………

2. Donner l'état initial

………

3. Donner l'état but

………

4. Avec un choix de représentation judicieux, quatre règles peuvent suffire à modéliser l'ensemble des opérations nécessaires à la modélisation du problème (*la notation de cet exercice en tiendra compte*). En utilisant des variables, donner 4 règles d'inférence, permettant de passer d'un état valide à l'autre. Ces règles ne peuvent pas comporter : de OU dans la partie condition, de OU dans la partie conclusion. Donner un titre explicatif à chacune des règles.

a) ……………………………………………………………………………………………………

………

b) ……………………………………………………………………………………………………

………

c) ……………………………………………………………………………………………………

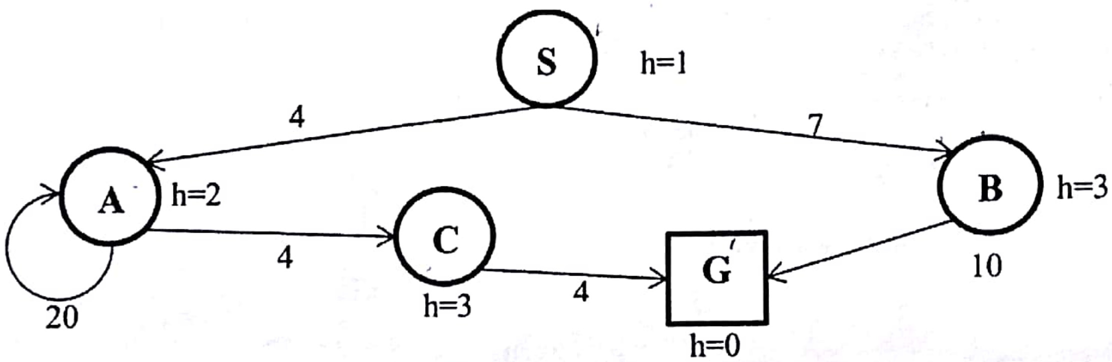
………

d) ……………………………………………………………………………………………………

………

Exercice 2. Recherche dans espace d'état (3pts)

Dans l'espace de recherche suivant, l'état S est l'état de départ et G le but. Le nombre au dessus d'un arc représente le coût pour le parcourir. La valeur de la fonction heuristique h est inscrite à côté de chaque noeud.



Appliquer l'algorithme de **rechercheEnArbre** vu en cours sur ce graphe. Les successeurs de chaque nœud sont retournés de gauche à droite (successeur de S=A,B). Dans les tableaux de simulation, noter les nœuds avec les chemins associés (exemple D_{SBD}, c'est-à-dire D en passant par SB) et ce pour plus de lisibilité. Pour chacune des stratégies de recherche, donner l'ordre dans lesquels les nœuds sont visités jusqu'àce que le but soit rencontré. Donner le chemin depuis S jusqu'au but ou mentionner AUCUN si le chemin n'est pas trouvé. Indiquer le coût du chemin s'il est trouvé.

1. *Recherche en profondeur d'abord*

Nœud examiné	*FRINGE*

Ordre dans lequel les nœuds sont visités : ....

Chemin de la solution : ....**Coût de la solution :** ....

2. ***Recherche en profondeur limitée itérative***

Nœud examiné	***FRINGE***	*Nœud examiné*	***FRINGE***

Ordre dans lequel les nœuds sont visités : ....

Chemin de la solution : ....Coût de la solution : ....

3. ***A****

	Liste des nœuds ouverts	***Liste des nœuds fermés***

Ordre dans lequel les nœuds sont visités : ....

Chemin de la solution : ....Coût de la solution : ....

La solution retournée par A* est-elle optimale ? Pourquoi ?

....

....

Exercice 3. QCM (4pts) entourer toutes les réponses correctes

1) Lorsqu'on utilise l'algorithme de rechercheEnArbre pour la résolution de problèmes généraux vu en cours, laquelle (ou lesquelles) de ces stratégies de recherche est complète :
   a. La recherche en Profondeur Limitée Iterative (Iterative deepening search)
   b. La recherche en Profondeur
   c. La recherche en Largeur d'abord
   d. La recherche en coût uniforme
   e. Aucune de ces réponses

2) Lorsqu'on utilise l'algorithme de rechercheEnArbre pour la résolution de problèmes généraux vu en cours, laquelle (ou lesquelles) de ces stratégies de recherche est optimale :
   a. La recherche en Profondeur Limitée Iterative (Iterative deepening search) avec des arcs de même coût
   b. La recherche en Profondeur avec des arcs de même coût
   c. La recherche en coût uniforme
   d. La recherche en Largeur d'abord
   e. Aucune de ces réponses

3) Soit deux fonctions heuristiques h1 et h2 tel que $h1(x) \leq h2(x)$ pour tout nœud x.
   a. L'utilisation de h1 par A* permet de visiter un nombre de nœuds inférieur ou égal par rapport à h2.
   b. L'utilisation de h2 par A* permet de visiter un nombre de nœuds inférieur ou égal par rapport à h1.
   c. Aucune de ces affirmations n'est vraie

4) Soient deux fonctions heuristiques h1 et h2 admissibles.
   a. h3= max(h1,h2) est admissible
   b. h3=min(h1,h2) est admissible
   c. h3=(h1+h2)/2 est admissible
   d. Aucune de ces affirmations n'est vraie

5) Lorsqu'on utilise l'algorithme A* pour la résolution de problèmes généraux vu en cours, on obtient un chemin optimal si l'heuristique est :
   a. concise
   b. admissible
   c. parfaite
   d. consistante
   e. toutes ces réponses

6) Lorsqu'on utilise l'algorithme RechercheEnGraphe vu en cours et que l'on choisit f=g+h minimal pour sélectionner le prochain nœud à examiner :
   a. Si h est admissible alors l'algorithme retourne le chemin optimal
   b. Si h est consistante alors l'algorithme retourne le chemin optimal
   c. L'algorithme retourne toujours le chemin optimal

7) L'algorithme de recherche locale hill-climbing est :
   a. Complet
   b. Optimal
   c. Aucune de ces réponses

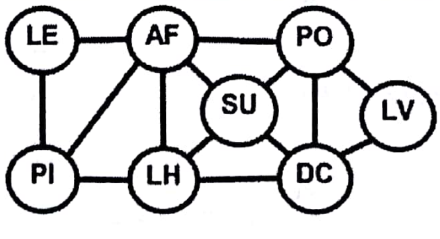
8) Soit l'algorithme de rechercheEnGraphe pour la résolution de problèmes généraux suivant :

```
function GRAPH-SEARCH(problem, fringe)
    closed ← an empty set,
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop
        if fringe is empty then
            return failure
        end if
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem,STATE[node]) then
            return node
        end if
        ADD STATE[node] TO closed
        fringe ← INSERTALL(EXPAND(node,problem), fringe)
    end loop
end function
```

<u>Entourer les problèmes de cet algorithme :</u>
   a. Les nœuds peuvent être développés plus de 2 fois
   b. Cet algorithme n'est pas complet
   c. Cet algorithme peut retourner une solution incorrecte

Exercice 4. Problèmes de satisfaction de contraintes (5pts)

Il s'agit de colorier une carte de telle sorte que deux pays voisins n'aient pas la même couleur. Soit le graphe de contraintes suivant correspondant à cette carte. Les couleurs à utiliser sont le rouge (R), le vert (V) et le bleu (B).



1. On affecte la valeur V à la variable **LH**. **<u>BARRER</u> <u>lisiblement</u>** toutes les valeurs qui seront éliminées par le **FORWARD CHECKING** dans le tableau suivant

AF			DC			LE			LH	LV			PI			PO			SU		
R	V	B	R	V	B	R	V	B	<u>V</u>	R	V	B	R	V	B	R	V	B	R	V	B

2. Soit le tableau suivant. On affecte PO=R. La propagation de contraintes a été réalisée par forward checking. **<u>Donner</u>** pour chaque variable non affectée, leur valeur calculée par l'heuristique **MRV** (Minimum Remaining Value) et **<u>indiquer</u>** la ou les variables à choisir en priorité. ....

AF	DC	LE	LH	LV	PI	PO	SU
V B	V B	R V B	R V B	V B	R V B	R	V B

3. Soit l'affectation PO=R. La propagation de contraintes a été réalisée par forward checking. Donner pour chaque variable non affectée, sa valeur calculée par l'heuristique **Degree Heuristic**. Indiquer la -ou les variables- à choisir en priorité d'après cette heuristique ....

AF	DC	LE	LH	LV	PI	PO	SU
V B	V B	R V B	R V B	V B	R V B	R	V B

4. On considère l'affectation complète mais inconsistante présentée dans le tableau ci-dessous. La variable AF a été sélectionnée pour une nouvelle affectation de valeur dans le but de parvenir à une affectation complète et consistante. Quelle nouvelle valeur pour AF sera choisie par **min-conflicts** heuristic ? ....
Pourquoi ? ....

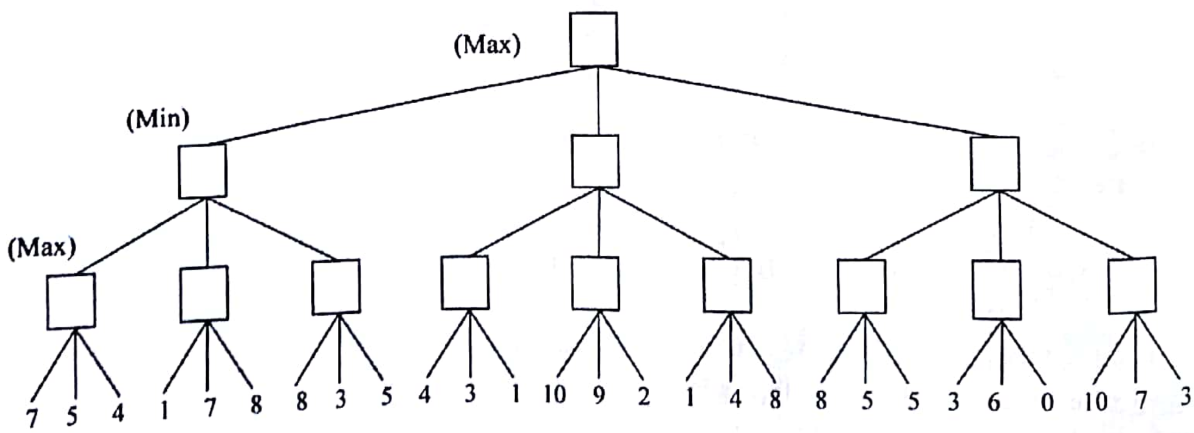
AF	DC	LE	LH	LV	PI	PO	SU
?	V	V	R	V	B	B	V

5. On affecte les valeurs AL=B et LE=R. Aucune propagation de contraintes n'a été réalisée. **BARRER lisiblement** toutes les valeurs qui seront éliminées par l'algorithme **d'ARC CONSISTENCE AC3** dans le tableau suivant

AF	DC	LE	LH	LV	PI	PO	SU
B	R V B	R	R V B	R V B	R V B	R V B	R V B

Exercice 5. Jeux : Algorithme minimax et élagage alpha-beta (3 pts)

1. Soit un jeu à deux joueurs opposant les joueurs MAX et MIN. Indiquer dans l'arbre de jeu la valeur des nœuds en simulant l'algorithme minimax.



2. *Quel coup doit jouer MAX pour maximiser son gain ? L'indiquer sur l'arbre de jeu ci-dessus.*

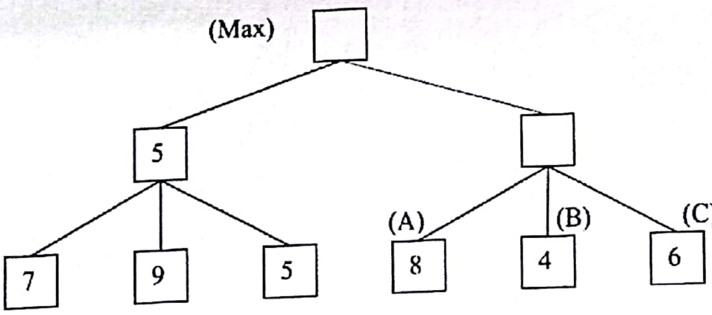
3. *Est-ce qu'on obtiendrait le même résultat en utilisant minimax avec élagage alpha-beta ? Pourquoi ?*

………

………

………

4. *L'efficacité de l'élagage alpha-beta dépend de l'ordre d'évaluation des valeurs des nœuds. En fonction de cet ordre, on peut être amené soit à ne faire aucun élagage soit à en faire de nombreux. Dans l'arbre de jeux ci-dessous, nous avons déterminé par minimax que la branche de gauche vaut 5. Il s'agit à présent d'étudier la branche de droite. Indiquer comment l'ordre de gauche à droite des nœuds A, B, C de la branche de droite peut avoir un effet sur l'élagage alpha beta :*



Donner l'ordre de présentation des nœuds qui permettrait d'élaguer 2 nœuds : ………
Donner l'ordre de présentation des nœuds qui permettrait d'élaguer 1 nœud : ………
Donner l'ordre de présentation des nœuds qui ne permettrait d'élaguer aucun nœud : ………