UNIVERSITÉ DE SHERBROOKE
Département d'informatique

IFT 615
Intelligence artificielle

Examen périodique
Été 2009

Le samedi 13 juin, 9 h à 10 h 50, au D3-2031

CHARGÉ DE COURS SUPPLÉANT
Éric Beaudry
http://planiart.usherbrooke.ca/~eric/

INSTRUCTIONS

L'examen dure une heure et cinquante minutes.

Les notes du cours (copie des présentations), le manuel (livres de référence) et les calculatrices sont autorisées. **Tout autre appareil électronique est strictement interdit, en particulier tout appareil muni d'un moyen de communication.**

L'examen comporte quatre questions pour un total de vingt points. Le questionnaire contient 7 pages incluant celle-ci.

Répondez directement sur le questionnaire aux endroits encadrés.

Des feuilles de brouillon vous sont fournies.

Ne détachez aucune feuille de ce questionnaire.

Écrivez votre nom, prénom et matricule ci-dessous.

NOM : _____ **PRÉNOM :** _____

MATRICULE : _____

SIGNATURE : _____

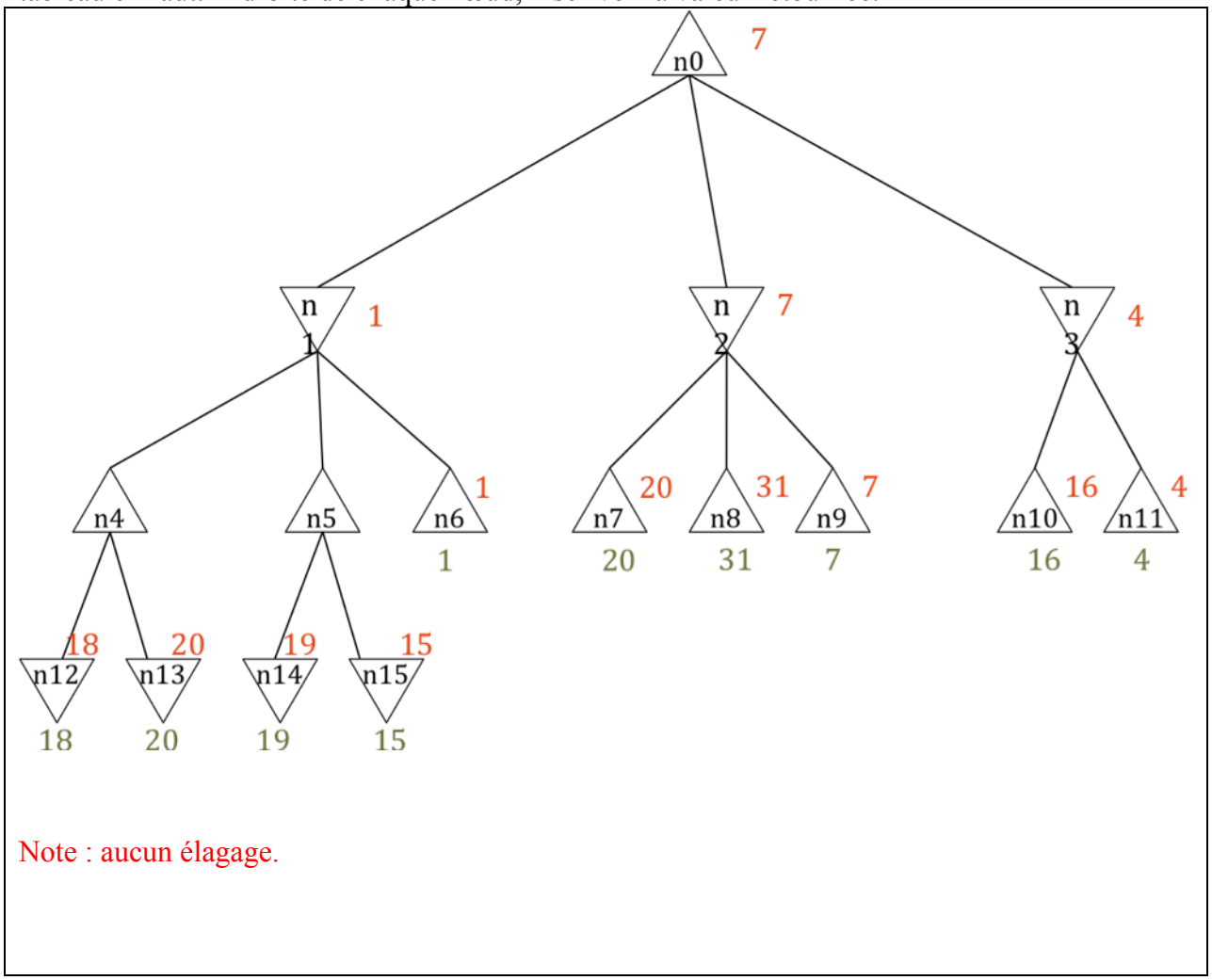
Q1 /5	Q2 /5	Q3 /5	Q4 /5	TOTAL /20

Question 1 (5 points) – *Alpha-Beta Pruning*

Soit un jeu à somme nulle où deux joueurs, Min et Max, s'affrontent. Le tour est rendu à Max. La configuration actuelle du jeu est donnée par le nœud n_0. La fonction *Successeurs*() donne les nœuds enfants d'un nœud. La fonction *Utilité*() retourne la valeur d'utilité d'un nœud feuille.

Nœud	Successeurs()	Utilité()	h()	Nœud	Successeurs()	Utilité()	h()
n_0	$\{n_1, n_2, n_3\}$		1	n_8	{}	31	31
n_1	$\{n_4, n_5, n_6\}$		0	n_9	{}	7	7
n_2	$\{n_7, n_8, n_9\}$		6	n_{10}	{}	16	16
n_3	$\{n_{10}, n_{11}\}$		13	n_{11}	{}	4	4
n_4	$\{n_{12}, n_{13}\}$		16	n_{12}	{}	18	18
n_5	$\{n_{14}, n_{15}\}$		14	n_{13}	{}	20	20
n_6	{}	1	1	n_{14}	{}	19	19
n_7	{}	20	20	n_{15}	{}	15	15

a) (3 points) Dessinez l'arbre visité par l'algorithme *Alpha-Beta prunning*. Les nœuds enfants d'un nœud doivent être visités dans le même ordre que ceux de la fonction *Successeurs*() du tableau ci-haut. À droite de chaque nœud, inscrivez la valeur retournée.

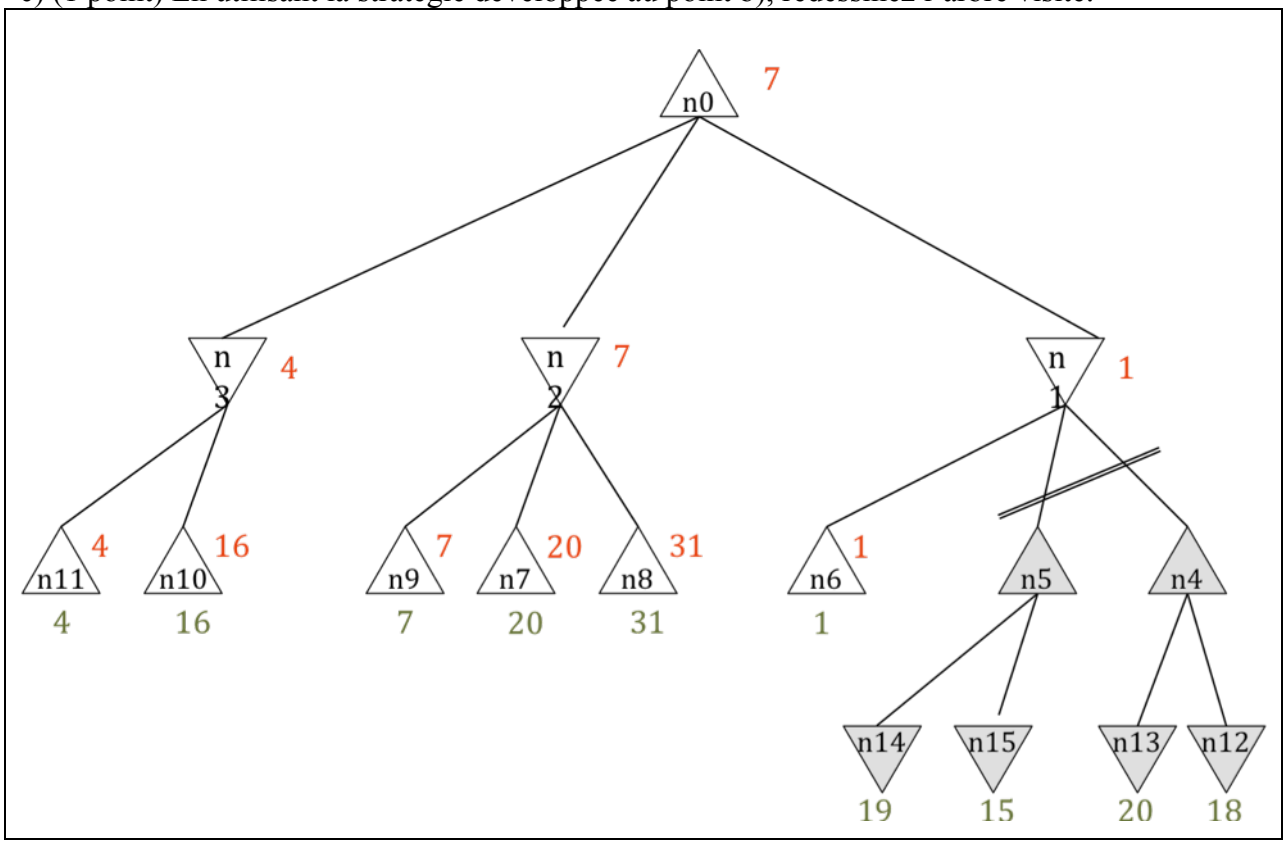


Note : aucun élagage.

b) (1 point). L'ordre de visite des nœuds peut avoir un impact considérable sur le nombre de nœuds visités et élagués, et ainsi affecter la performance de l'algorithme. La fonction *h*() retourne une estimation de la valeur d'utilité pour chaque nœud. Comment utiliseriez-vous la fonction *h*() afin d'élaguer le plus de nœuds possibles?

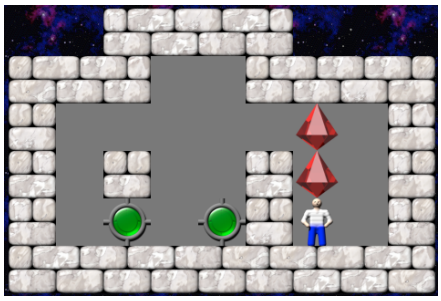
L'idée est de visiter les nœuds qui vont provoquer des élagages en premier.
Donc, on trie les successeurs des nœuds Max en ordre décroissant de h() et les successeurs des nœuds Min en ordre croissant

c) (1 point) En utilisant la stratégie développée au point b), redessinez l'arbre visité.



Question 2 (5 points) – *Algorithme A* appliqué au jeu de Sokoban*

Vous devez programmer un résolveur pour le jeu Sokoban. Dans la grille de jeu, le joueur peut se déplacer dans quatre directions (haut, bas, gauche et droite). Le but du jeu consiste à pousser des boîtes dans des cases cibles. Il est à noter que le joueur ne peut pas tirer les boîtes. Un plateau de jeu est donné par une grille de n cases, la position initiale du joueur (x, y), m boîtes placées à des positions (bx_1, by_1) … (bx_m, by_m) et m cases cibles (cx_1, cy_1) … (cx_m, cy_m). Il est à noter que les boîtes et les cases cibles ne peuvent pas être distinguées. L'image ci-contre montre une capture d'écran d'un jeu Sokoban. Dans ce plateau, le joueur est représenté par un bonhomme. Deux boîtes sont initialement placées au-dessus du joueur et doivent être transportées sur les cases cibles qui sont désignées par les deux cercles. Pour pousser une boîte d'une case, le joueur doit être sur une case voisine et de déplacer vers la boîte. Par exemple, pour déplacer la première boîte d'une case vers la gauche, le joueur doit exécuter les actions suivantes : droite, haut, haut et gauche. Une seule boîte peut être poussée à la fois.



a) (3 points) Comment utiliseriez-vous l'algorithme A* pour trouver une solution optimale à ce problème? Décrivez la représentation des états ainsi que les fonctions successeurs et de but.

État :
Position du bonhomme (x, y) + positions des boîtes (bx1, by1) … (bxm, bym).

Fonction successeurs :
La fonction successeur prend en entré un état et retourne jusqu'à 4 nouveaux états successeurs, chacun étant le résultat d'une action haut, bas, gauche et droite. Dans ces états, on déplace la position du joueur (x, y) si la case dans la direction de l'action est libre. Si une boîte se trouve à l'endroit où le joueur souhaite aller, on déplace la boîte dans la même direction si la case est libre. Sinon l'action n'est pas possible et ne génère pas de successeur.

But :
La fonction prend un état en paramètre et retourne vrai si et seulement si toutes les boîtes se retrouve dans une case cible (cx1, cy1) … (cxm, cym).

b) (2 points) Donnez une heuristique admissible la plus efficace possible pour le jeu de Sokoban.

Une fonction heuristique admissible ne doit pas surestimer le coût restant. Chaque boîte doit suivre un chemin vers un but. Donc, la somme des distances entre chaque boîte et le but le plus prêt est une borne inférieure du nombre de coups à faire. À Cela on peut ajouter le nombre de coups requis pour déplacer le joueur vers une boîte.

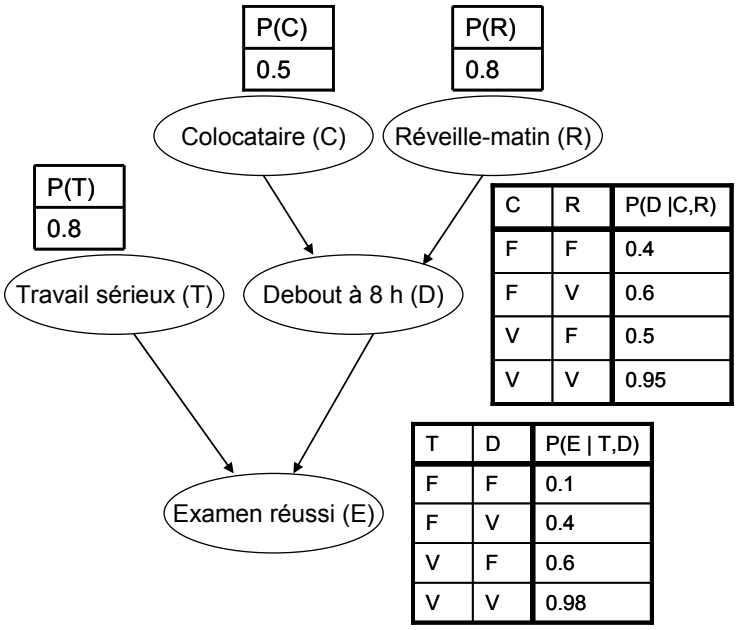
Donc :

$$h(n) = \min_{i=1..m}(d((x,y),(bx_i,by_i)) - 1) + \sum_{i=1..m} \min_{j=1..m} d((bx_i,by_i),(cx_j,cy_j))$$

où d(a,b) est la distance entre les cases a et b. Une distance de Manhattan peut être utilisée, ou mieux on peut précalculer les chemins optimaux entre toutes les paires de cases possibles avec A*, Djkastra ou .

Question 3 (5 points) – *Réseaux bayésiens*

Soit le réseau bayésien suivant dans lequel les nœuds représentent des variables booléennes. Ce réseau bayésien détaille des relations causales dans le contexte de la réussite d'un examen. La réussite de l'examen dépend de l'effort de l'étudiant (travail sérieux) et du fait qu'il puisse se lever assez tôt le jour de l'examen. Pour se lever tôt, l'étudiant a demandé à son colocataire de le réveiller. Comme ce dernier a tendance à faire souvent la fête la vielle, l'étudiant lui fait peu confiance et compte aussi sur son réveille-matin.



C	R	P(D |C,R)
F	F	0.4
F	V	0.6
V	F	0.5
V	V	0.95

T	D	P(E | T,D)
F	F	0.1
F	V	0.4
V	F	0.6
V	V	0.98

a) (1 point) Calculez P(d), c'est-à-dire, P(D=V).

P(¬ c)*P(¬ r) * P(d | ¬ c, ¬ r) + P(¬ c)*P(r) * P(d | ¬ c,r)
+ P(c)*P(¬ r) * P(d | c, ¬ r) + P(c)*P(r) * P(d | c,r)
= 0.71

b) (1 point) Vrai ou faux : P(T|D) = P(T)? Justifiez.

VRAI. T et D sont indépendants.

c) (1 point) Vrai ou faux : P(E|D,C,R) = P(E|D)? Justifiez.

VRAI. E est conditionnellement indépendant de C et R étant donné que D est connu.

d) (1 point) Vrai ou faux : P(T | D) = P(T|E, D)? Justifiez.

Faux. T et ~~D~~ E sont conditionnellement dépendants sachant ~~E~~ D.

e) (1 point) Calculez P(e | ¬ c, ¬ r), c'est-à-dire, P(E=V | C=F, R=F).

On peut simplifier P(D| ¬ c, ¬ r) = [0.6, 0.4]
P(e| ¬ c, ¬ r)) = P(¬ t) P(¬ d) * 0.1
+ P(¬ t) P(d) * 0.4
+ P(t) P(¬ d) * 0.6
+ P(t) P(d) * 0.98 = 0.6456

Question 4 (5 points) – *CSP*

Lors de votre party de fin de session, on vous mandate d'être le D.J. Votre mission est de sélectionner les cinq pièces musicales à jouer lors de la soirée. Votre répertoire musical est composé d'un ensemble de dix pièces musicales $\{M_1, \ldots, M_{10}\}$. De ce nombre, six sont en anglais ($M_1, \ldots, M_6$) et quatre en français ($M_7, \ldots, M_{10}$). Les pièces sont classées en styles musicaux : *rock*=$\{M_1, M_2, M_7\}$, *jazz*=$\{M_3, M_8\}$, *techno*=$\{M_4, M_5, M_9\}$ et *alternatif*=$\{M_6, M_{10}\}$. Le comité organisateur vous impose certaines contraintes que vous devez respecter :

1) vous ne pouvez pas jouer deux pièces consécutives dans la même langue;
2) vous ne pouvez pas jouer deux pièces consécutives du même style de musique;
3) vous devez faire jouer au moins une pièce de chaque style;
4) vous devez placer une demande spéciale du président de votre association qui veut la pièce M_{10}.
5) vous devez terminer la soirée avec une pièce de *jazz*.

a) (3 points) Indiquez comment modéliser ce problème dans un cadre CSP. Donnez les variables et les contraintes nécessaires.

Variables :
Il faut une variables pour chaque entrée de la sélection : S1, S2, S3, S4 et S5.
Le domaine est l'ensemble des 10 pièces musicales W = {M1 à M10}.

Fonctions :
Langue(M) : retourne la langue d'une pièce
Style(M) : retourne le style d'une pièce

Contraintes :
Contraintes unaires : S5 □ ـبM3, M8} // Jazz (5)

Contraintes binaires :
Langue(S1) ≠ Langue(S2) ≠Langue(S3) ≠Langue(S4) ≠Langue(S5) // #1
Style(S1) ≠ Style (S2) ≠ Style (S3) ≠ Style (S4) ≠ Style (S5) // #2

Containres n-aires
U Style(Si) = {rock, jazz, techno, alternatif} // #3

{M10} □ U Si // #4

b) (2 points) Quel algorithme utiliseriez-vous pour résoudre votre problème? Simulez les étapes de l'algorithme et donnez la solution obtenue.

Algorithme : « Backtracking search avec forward checking »
1) utiliser les contraintes unaires pour restreindre les domaines des variables.
2) utiliser les contraintes binaires pour le « forward checking ».
3) lors d'une assignation complète, tester les contraintes n-aires.

Stratégies (~ heuristique) pour choisir la prochaine variable + valeurs à assigner:
1) MRV (Most Restrictive VALUE) : commencer par la variable la plus contrainte.
2) allouer M10 aussitôt qu'elle apparaît dans le domaine d'une variable.
3) dans le choix des valeurs, prioriser les styles qui n'ont pas été encore alloués.

Étapes de l'algorithme

Étape \ Var	**S1**	**S2**	**S3**	**S4**	**S5**
Init Domaines	{M1,…, M10}	{M1,…, M10}	{M1,…, M10}	{M1,…, M10}	M3, M8}
Choix S5 / Heuristique 1	{M1,…, M10}	{M1,…, M10}	{M1,…, M10}	{M1,…, M10}	M3
Forward Checking	M1, M2, M4,M5, M6	M7, M8,M9,M10	M1, M2, M4,M5, M6	M7, M8,M9,M10	M3
Choix S5 / Heuristique 2	M1, M2, M4,M5, M6	M10	M1, M2, M4,M5, M6	M7, M8,M9,M10	M3
Forward Checking	M1, M2, M4,M5, M6	M10	M1, M2, M4,M5, M6	M7, M8,M9	M3
Choix S4 / Heuristique 3	M1, M2, M4,M5, M6	M10	M1, M2, M4,M5, M6	M7	M3
Choix S1 / Heuristique 3	M4	M10	M1, M2, M4,M5, M6	M7	M3
Forward Checking	M4	M10	M1, M2,M5, M6	M7	M3
Choix S3 / (arbitraire)	M4	M10	M1	M7	M3
Tests contraintes n-aires réussis. L'algorithme retourne la solution obtenue.					

Solution finale : S1=M4, S2=M10, S3=M1, S4=M7 et S5=M3.

Aucun « backtracking » n'a été requis.