Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université de Carthage
Ecole Nationale d'Ingénieurs de Carthage

ENICARTHAGE
المدرسة الوطنية للمهندسين بقرطاج
Ecole Nationale d'Ingénieurs de Carthage

وزارة التعليم العالي و البحث العلمي
جامعة قرطاج
المدرسة الوطنية للمهندسين بقرطاج

Correction Devoir Surveillé
Matière : Intelligence Artificielle

Enseignant(es)	: M. Fourati, H. Ghazouani & A. Jlassi	**Date**	: 14/03/2019
Filière / Classe	: 2ère Ingénieur Informatique	**Durée**	: 1h30
Barème indicatif	: **2.5-1.5-11-5**	**Documents**	: ~~aut.~~ / non aut.
Nbre. de pages	: **6 pages**	**Calculatrice**	: ~~aut.~~ / nonaut.

Nom :…….. **Prénom :**…… **Salle :**……..

Exercice 1 Vrai ou Faux (2.5 points)
Répondre par vrai ou par faux. Vous avez 0.25 par bonne réponse et -0.25 par mauvaise réponse.

+0.25/bonne réponse **-0.25/mauvaise réponse**	**VRAI**	**FAUX**
L'exploration par Recuit Simulé choisit le meilleur voisin au hasard. ***Même pour ceux qui ont choisi vrai c'est pris en considération.***	☐	x
Une heuristique $h(n)=0$ est toujours admissible quel que soit le problème. ***Ce h ne surestime jamais le coût réel. Mais si on considère un but négatif, on doit prendre les 2 réponses.***	x	☐
Une exploration à coût uniforme ne générera jamais plus de nœuds qu'une exploration avec A*. ***UCS est un cas particulier d'A∗ : h(n) = 0***	☐	x
Lorsque *b* est fini, la recherche en largeur d'abord est toujours complète même si l'espace d'états est infini.	x	☐
Une recherche itérative en profondeur retourne toujours la même solution qu'une recherche en largeur d'abord si *b* est fini et les successeurs d'un nœud sont ordonnés de la même façon.	x	☐
Ce qui compte pour une recherche en largeur d'abord est la profondeur de la solution et non le coût de cette dernière.	x	☐
Les algorithmes RBFS est moins couteux que A* en terme de temps.	☐	x
Une exploration en profondeur d'abord est toujours complète.	☐	x
L'exploration à coût uniforme procède au test du but une fois que le nœud est choisi pour exploration afin de garantir la complétude.	☐	x
Dans la recherche dans un espace d'états, l'exécution de la solution retenue se fait au fur et à mesure de l'exploration.	☐	x

Exercice 2 Question à choix multiples (1.5 points) **+0.25/bonne réponse/-0.25/mauvaise réponse**Pour chacune des questions suivantes, choisir la ou les bonnes réponses. Vous avez 0.25 pt par bonne réponse si toutes les réponses sont correctes, mais vous avez -0.25 pt par mauvaise réponse.

1. Si h_1 et h_2 sont admissibles, alors :
   a. h_1+h_2 est admissible ☐
   b. $h_1 x h_2$ est admissible ☐
   c. $\max(h_1,h_2)$ est admissible x
   d. $\min(h_1,h_2)$ est admissible x

2. L'algorithme trouve une solution optimale :
   a. A* [x]
   b. Meilleurs d'abord
   c. Largeur d'abord [x] Si coût unitaire
   d. Profondeur d'abord
   e. RBFS [x]
   f. Coût uniforme [x]
   g. Hill climbing

Exercice 3 : Recherche dans un espace d'états 11.5 points)

La figure 1 suivante représente l'espace d'état d'un problème. Le but est l'état S_6. Le tableau 1 donne les valeurs heuristiques h_1 et h_2.

Etat	h_1	h_2
S_0	9	10
S_1	8	9
S_2	5	6
S_3	3	10
S_4	7	10
S_5	7	6
S_6	0	0

Tableau 1 : Heuristiques h_1 et h_2

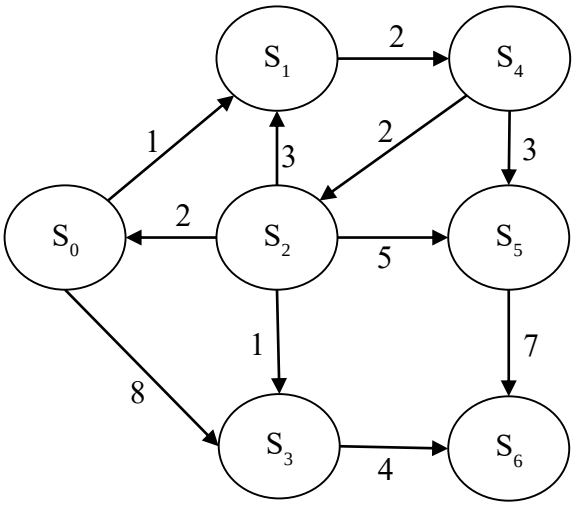


Figure 1 : Espace d'états

1. Dire si h_1 et h_2 sont admissibles. Expliquer. ***(1pt pour le tableau et 0.5 pt pour l'explication admissible non admissible (0.25 pt par fonction))***

	S_0	S_1	S_2	S_3	S_4	S_5	S_6
h^*	10	9	5	4	7	7	0
h_1	9	8	5	3	7	7	0
h_2	10	9	6	10	10	6	0

$\forall i\ h^*(S_i) \geq h_1$ donc h_1 est admissible.
$h^*(S_3) < h_2(S_3)$ donc h_2 est non admissible.

2. Pour chacune des méthodes de recherche suivantes indiquer quel but est atteint. Donner à chaque étape les nœuds sur la frontière ainsi que le nœud choisi pour développement. En cas d'égalité entre les nœuds, choisir les nœuds en priorité dans l'ordre numérique : S_0, S_1, S_2, ...

a. Coût uniforme avec une exploration en graphe. ***(1.75pt pour le tableau : 0.25pt/ligne et 0.25pt pour la séquence)***

Nœuds sur la frontière	Nœud choisi pour être développé
(S_0,0)	(S_0,0)
(S_1,1), (S_3,8)	(S_1,1)
(S_4,3), (S_3,8)	(S_4,3)
(S_2,5), (S_5,6), (S_3,8)	(S_2,5)
(S_3,6), (S_5,6)	(S_3,6)
(S_5,6), (S_6,10)	(S_5,6)
(S_6,10)	**(S_6,10) arrêt**

Séquence choisie pour être exécutée :
S_0- S_1- S_4- S_2- S_3- S_6

b. Largeur d'abord en graphe. ***(0.75pt pour le tableau : 0.25pt/ligne et 0.25pt pour la séquence)***

Nœuds sur la frontière	Nœud choisi pour être développé
S_0	S_0
S_1, S_3	S_1
S_3, S_4	S_3 arrêt le fils S_6 de S_3 est but

Séquence choisie pour être exécutée :
S_0- S_3- S_6

c. A* en graphe. ***(1.5 pour le tableau : 0.25pt/ligne et 0.25pt pour la séquence)***

Nœuds sur la frontière	Nœud choisi pour être développé
(S_0,9)	(S_0,9)
(S_1,9), (S_3,11)	(S_1,9)
(S_4,10), (S_3,11)	(S_4,10)
(S_2,10), (S_3,11), (S_5,13)	(S_2,10)
(S_3,9), (S_5,13)	(S_3,9)
(S_6,10), (S_5,13)	**(S_6,10) arrêt**

Séquence exécutée :
S_0- S_1- S_4- S_2- S_3- S_6

d. Gloutonne par le meilleur d'abord avec une exploration en graphe. ***(0.75pt pour le tableau : 0.25pt/ligne et 0.25pt pour la séquence)***

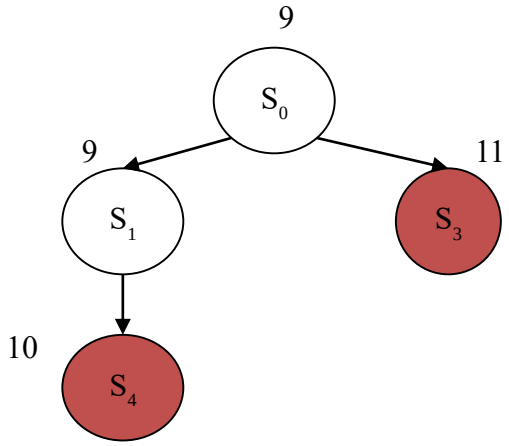
Nœuds sur la frontière	Nœud choisi pour être développé
(S_0,9)	(S_0,9)
(S_3,3), (S_1,8)	(S_3,3)
(S_6,0), (S_1,8)	(S_6,0) **arrêt**

Séquence exécutée :
S_0- S_3- S_6

e. IDA* en graphe. Donner pour chaque itération l'arbre d'exploration (contenant les nœuds développés ainsi que la frontière), la valeur de f limite, ainsi que les nœuds développés.

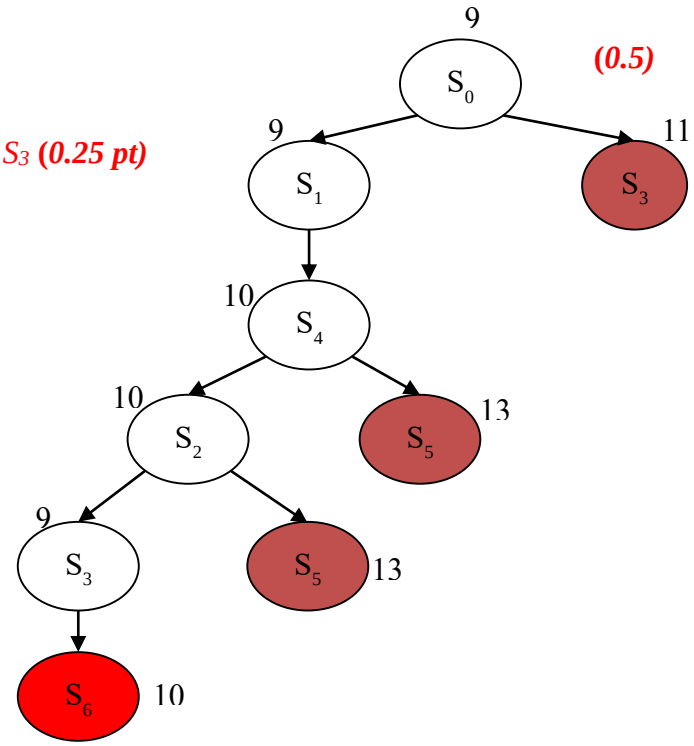
Itération 1
f = 9 ***(0.25 pt)***
Nœuds développés : S_0 et S_1 ***(0.25 pt)***

(0.25)



Nom :	Prénom :

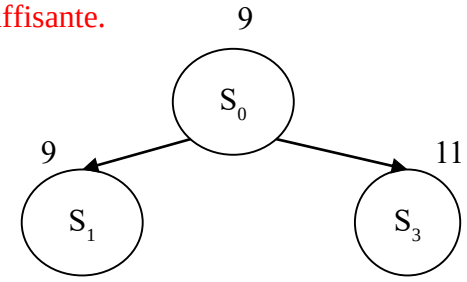
Itération 2
f = 10 ***(0.25 pt)***
Nœuds développés : S_0, S_1, S_4, S_2 et S_3 ***(0.25 pt)***



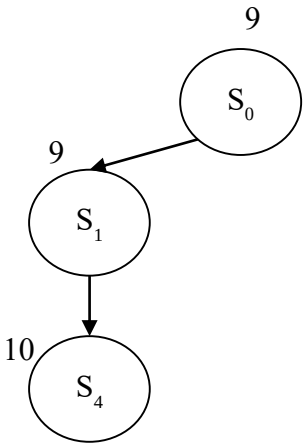
3. En supposant que la mémoire est limitée à 3 nœuds, l'algorithme SMA* parviendrait-t-il à trouver une solution optimale à ce problème ? Expliquez brièvement. ***(0.75pt)***

   Non puisque la solution optimale suit le chemin *S_0 S_1 S_4 S_2 S_3 S_6 donc il doit aller à une profondeur 5 et la mémoire est limitée à 3 nœuds.*

   L'étudiant n'est pas obligé de faire le développement ici-bas, la réponse ci-haut est suffisante.

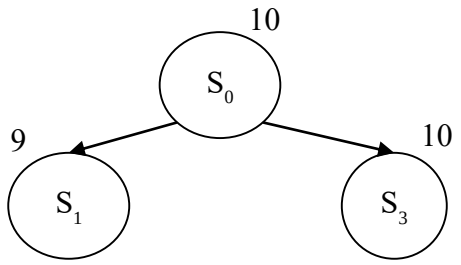


Ici, l'algorithme doit abandonner un nœud, S_3

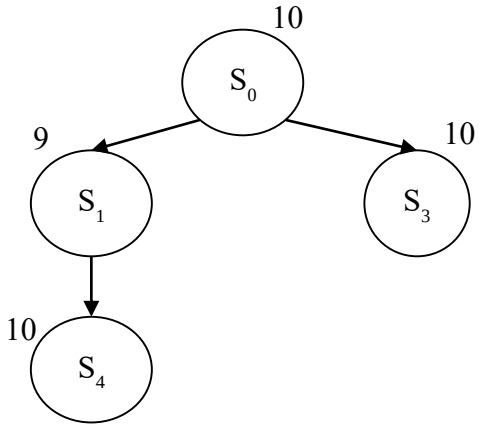


L'algorithme ne peut plus développer le fils de S4 et effacera ce dernier pour générer le fils de S3, soit S6 et il va le prendre comme solution. Mais ce chemin n'est optimal.

4. Nous désirons appliquer l'algorithme par escalade (Hill Climbing) en considérant l'heuristique h_2 comme valeur de la fonction *objectif*. Dans ce cas, cet algorithme permet-il d'arriver à une solution ? Expliquer le résultat obtenu.



L'algorithme commence par développer les fils de S_0 et il va choisir le nœud S_1. ***(0.5 pt)***



En choisissant S1, il sera piégé dans un minimum local plat. Donc il va retourner S_1 comme solution. ***(0.5 pt)***

5. Expliquer comment l'algorithme de recuit simulé procède-t-il pour éviter le problème rencontré dans la question 4 et dites s'il parvient à tous les coups à l'éviter. ***(0.75 pt)***

   Il utilise un schéma de température, qui, au début de l'algorithme est très élevé, et donc l'algorithme choisit n'importe quel nœud généré s'il ne dégrade pas sa fonction *objectif* ou même s'il dégrade sa fonction, il peut quand même le choisir avec une bonne probabilité (car la température est élevée). Ici, il peut choisir S_1, ensuite S_4 et avec un peu de chance, il peut arriver à S_6.

Exercice 4 Recherche dans un espace d'états (4.5 points)

Supposons que deux amis vivent dans différentes villes d'une carte, telle que celle de la Roumanie vue en cours. A chaque tour, nous déplaçons chacun des amis simultanément (en même temps) vers une ville voisine sur la carte (il faut bien sûr qu'il y ait une route entre les deux villes). Le temps nécessaire pour se déplacer d'une ville *i* à une ville *j* voisine est égal à la distance *d*(*i*,*j*) entre les villes. Cependant, à chaque tour, l'ami qui arrive en premier doit attendre que l'autre arrive (il doit donc appeler le premier sur son téléphone portable) avant que le tour suivant ne commence. Nous désirons que les deux amis se rencontrent dans une même ville le plus tôt possible. Nous allons donc formuler ce problème en tant qu'un problème de recherche dans un espace d'état. Il faut répondre aux questions suivantes par des notations formelles (en fonction de de variable telles que i, j et des fonctions que vous pouvez définir).

1. Quel est l'espace d'état de ce problème (on ne demande pas de le dessiner mais de le formuler) ? **(0.5)**

   L'espace d'états du problème est constitué de paires (i,j) représentant les positions des deux amis. *Il est à noter que la carte n'est pas l'espace d'état*.

2. Quel est la fonction successeur de ce problème (elle doit définir les actions admissibles pour passer d'un état à un autre) ? **(0.5)**

   Les successeurs de (i,j) sont toutes les paires (x,y) tel que *adjacent*(x,i) et *adjacent*(y,j).

3. Est-ce qu'il y a un seul état initial pour ce problème ? Le définir ou les définir. **(0.5)**

   N'importe quelle configuration de i et j : (i,j).

4. Quel est l'état but de ce problème ? **(0.5)**

   N'importe quel état : (i,i).

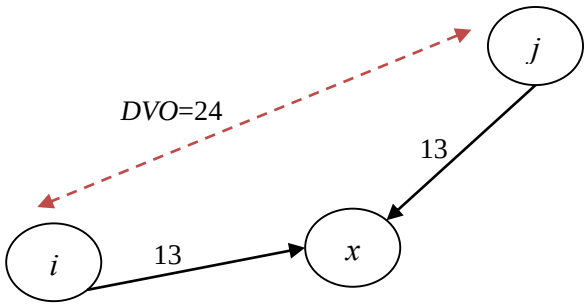
5. Quel est le coût de chaque action de ce problème ? **(1)**

   Le coût pour aller de (i,j) à (x,y) est le plus long déplacement entre ceux des deux amis. Donc $\max(d(i,x),d(j,y))$ **(1)**

6. Trouver une fonction heuristique admissible pour ce problème. **(1)**

   Supposons que les deux amis se retrouvent respectivement dans les villes *i* et j. Si on considère que *DVO*(i,j) est la distance à vol d'oiseau entre deux villes *i* et *j*, une heuristique admissible serait *DVO*$(i,j)/2$. En effet, dans le meilleur des cas, les deux amis se déplacent en même temps et peuvent se retrouver en allant directement de l'un vers l'autre en passant par des étapes de même cout.

   Exemple, ici h=12 alors que le coût réel est 13.



7. Est-ce qu'il existe une carte pour laquelle aucune solution n'existe ? Expliquer. **(0.5)**
   Oui. Imaginez une carte qui contient uniquement 2 villes adjacentes connectées par un seul arc.