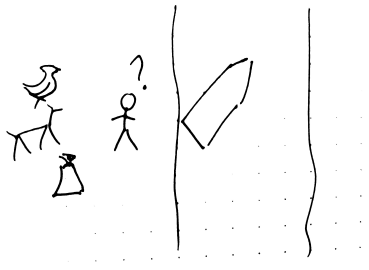
Algorithmes pour l'IA – TD 1
Algorithmes de recherche

Considérant le problème suivant : Charlie doit traverser une rivière avec son renard, son poulet et ses grains. Pour cela il a un bateau mais qui ne peut transporter qu'une seule de ses possessions.

- si le renard reste seul avec le poulet, il le mange
- si le poulet reste seul avec les grains, il les mange



T1: Décrivez l'état initial, l'état final et les transitions entre états pour ce problème

On peut décrire un état en indiquant ce qui se trouve de chaque coté de la rivière à cet état de la résolution. Avec cette représentation, en utilisant les initiales et '‖' pour représenter la rivière, on a :
État initial : (CPRG‖)
État final : (‖CPRG)
Transition : Charlie traverse la rivière, avec une de ses possessions ou rien, en ne laissant jamais le renard seul avec le poulet, ou le poulet seul avec le grain.

T2: Déroulez les algorithmes de recherche en largeur et en profondeur pour le problème du renard, du poulet et des grains ci-dessus. Vous pouvez par exemple dessiner le graphe en montrant à chaque étape qui se trouve de quel côté de la rivière.

Parcours en largeur d'abord :

(CPRG‖) (RG‖CP) (CRG‖P) (G‖CRP) (CPG‖R) (R‖CPG) (CPR‖G) (P‖CRG) (CP‖RG) (‖CPRG)

Parcours en profondeur d'abord :

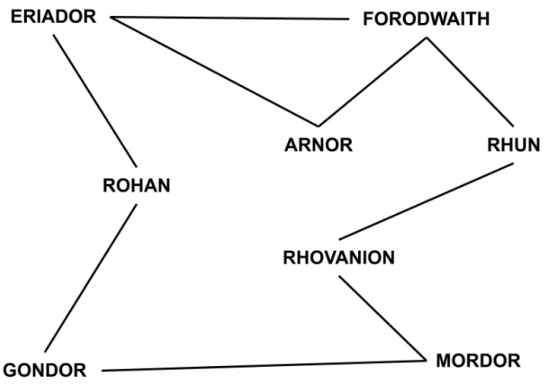
(CPRG‖) (RG‖CP) (CRG‖P) (G‖CRP) (CPG‖R) (P‖CRG) (CP‖RG) (‖CPRG)

Frodon veut se rendre au Mordor à partir d'Eriador. Le graphe ci-dessous montre comment les villes sont connectées et la distance à vol d'oiseau entre chacune de ces villes. La distance "réelle" d'une ville à l'autre est toujours 1.

T3: Déroulez les algorithmes de parcours en largeur et en profondeur d'abord sur ce graphe. Vous pouvez marquer chaque noeud avec un numéro correspondant à l'ordre dans lequel il sont exploré, et indiquer (en les soulignant par exemple) ceux qui sont sur le chemin.

Distance à vol d'oiseau jusqu'au MORDOR

ERIADOR	0.9
GONDOR	0.8
FORODWAITH	0.7
ROHAN	0.6
ARNOR	0.5
RHUN	0.4
RHOVANION	0.3



Parcours en largeur d'abord :

Element défilé	**file**		
	E		
E	RO	A	F
RO	A	F	G
A	F	G	F
F	G	F	RH
F	G	F	RHU
G	F	RHU	**M**

Chemin : E, RO, G, M

On note que d'après l'algorithme vu, on ne test pas les éléments de la liste avant d'en mettre des nouveaux. Du coup, F apparaît 2 fois. Aussi, si on suivait l'algorithme à la lettre, il faudrait continuer jusqu'a sortir M de la file, vu qu'on ne fait le test que sur l'élément en dehors de la liste.

Parcours en profondeur d'abord :

Element dépilé	**pile**		
	E		
E	RO	A	F
F	RO	A	RHU
RHU	RO	A	RHO
RHO	RO	A	**M**

Chemin : E, F, RHU, RHO, M

T4: D'après le tableau fournit, l'heuristique "à vol d'oiseau" est il admissible ?

Oui, car les distances réelles entre villes sont au minimum de 1, et aucune des valeurs heuristiques ne sont plus grandes que 1, donc l'heuristique va forcement sous-estimer la distance réelle. Par contre, l'heuristique n'est pas terrible même si corrélé avec la valeur réelle.

T5: Déroulez les algorithmes glouton et A* en utilisant l'heuristique 'à vol d'oiseau".

Parcours glouton :

Element dépilé	**pile ($h(n)$)**		
	E(0.9)		
E	F(0.7)	RO(0.6)	A(0.5)
A	F(0.7)	RO(0.6)	F(0.7)
F	F(0.7)	RO(0.6)	RHU(0.4)
RHU	F(0.7)	RO(0.6)	RHO(0.3)
RHO	F(0.7)	RO(0.6)	**M(0)**

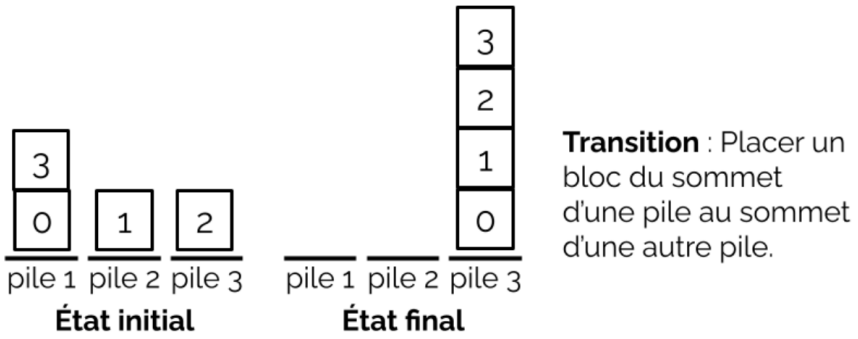
Chemin : E, A, F, RHU, RHO, M

A* :

Element dépilé	**pile ($f(n) = g(n) + h(n)$)**		
	E(0+0.9)		
E	F(1+0.7)	RO(1+0.6)	A(1+0.5)
A	F(1+0.7)	RO(1+0.6)	
RO	G(2+0.8)	F(1+0.7)	
F	G(2+0.8)	RHU(2+0.4)	
RHU	RHO(3+0.3)	G(2+0.8)	
G	RHO(3+0.3)	**M(3+0)**	

Chemin : E, RO, G, M

Considérant le problème dans le "monde des blocs" suivant:



T6: Donnez deux heuristiques admissibles pour résoudre ce problème. Montrez quelles sont admissibles.

Heuristique 1: nombre de blocs pas dans la pile 3. Admissible parce qu'il faudra au moins autant de mouvement pour les mettre dans la pile 3.

Heuristique 2: Heuristique 1 + nombre de blocs dans la pile trois dans le désordre et hors de la pile trois dans l'ordre. Même raisonnement.

T7: Montrez pourquoi, si c'est le cas, une d'entre elles est meilleure que l'autre.

Heuristique 2 donnera des valeurs plus élevé que heuristique 1 tout en étant admissible. Elle est donc plus une meilleur approximation de la distance réelle à la solution et trouvera le résultat plus rapidement.

T8: Déroulez une branche de l'arbre A* sur au moins 4 niveaux avec votre meilleure heuristique. Indiquez à chaque noeud la valeur de f(n). Indiquez avec une ligne pointillée si une branche devrait normalement être explorée, mais vous choisissez d'en dérouler une autre.

... prend du temps et beaucoup de retour arrière...

T9: Êtes vous toujours satisfait.e de votre heuristique ?
Non, l'heuristique n'est clairement pas monotone et requiert l'exploration de beaucoup de branches une après l'autre.

T10: On a vu que le parcours bi-directionel pouvait avoir des avantages dans le cas du parcours en largeur d'abord.

- Pourrait-on l'appliquer à A* ? oui, c'est possible
- Décrivez comment ça pourrait marcher. comme pour le parcours en largeur, en alternant entre les deux directions jusqu'à ce qu'elles se rencontrent.
- Aurait-on le même avantage, dans le pire cas, que dans le cas du parcours en largeur ? Il y aurait un avantage, mais pas nécessairement aussi conséquent que pour le parcours en largeur. Aussi, il est possible que les résultats ne soient plus optimaux.
- En pratique, peut-on espérer réduire significativement le nombre d'opérations à réaliser ? Est-ce mieux ou pire en ce qui concerne l'espace mémoire ? Cela requiert plus d'espace mémoire.