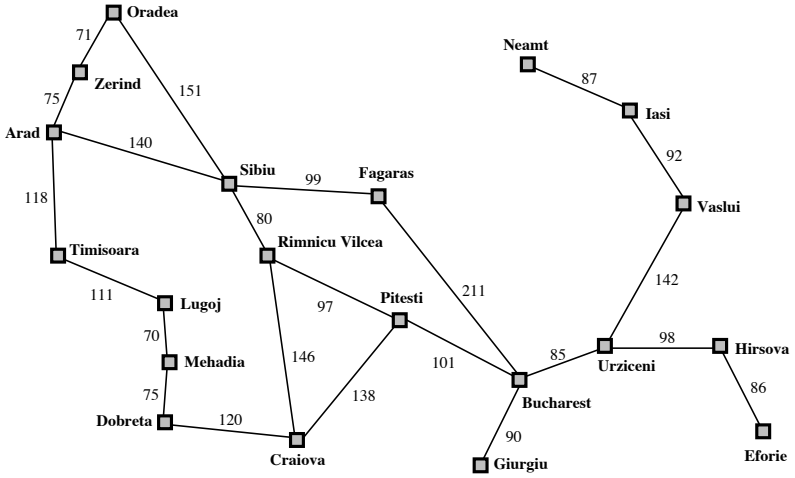
Université d'Orléans **Master Informatique**
Intelligence Artificielle Année 2012-2013

Feuille n° 3 - Algorithmes de recherche heuristique

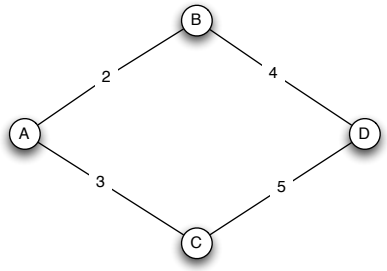
Exercice 1 Nous avons la carte suivantes avec des villes de la Roumanie, avec les distances en accès direct :



Nous avons également la distance en vol d'oiseau des villes par rapport à Bucarest :

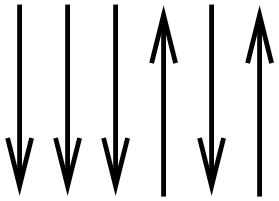
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Dobreta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

1. Simuler l'algorithme de recherche gloutonne pour trouver un chemin allant de Lugoj à Bucarest en utilisant l'heuristique de distance en vol d'oiseau.
2. Pour le même problème simuler l'algorithme A*.
3. Simuler les deux algorithmes pour trouver un chemin de A à D avec l'heuristique A(5), B(4), C(2), D(0).

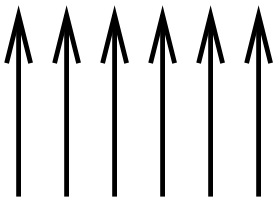


Exercice 2 On dispose de six flèches disposées comme dans la figure à gauche. On cherche à les ammener dans la position de la figure à droite. On ne peut que retourner ensemble d'un seul coup

deux flèches adjacentes.



Etat initial



Etat final

1. Donner la formulation d'état, de l'état initial et de l'état final.
2. Imaginer plusieurs heuristiques pour ce problème et donner leur résultat. Écrire les prédicats permettant de calculer ces heuristiques.
3. Écrire les actions permettant de passer d'un état à un état suivant.
4. Proposer une implantation des nœuds du graphe de recherche de l'algorithme A*. Entre autres, pour chaque nœud, il faut qu'on puisse connaître le chemin qui mène de l'état initial à l'état courant présenté par le nœud, l'action qui mène de l'état précédent à l'état courant, le coût réel pour arriver à l'état courant et le coût estimé f.
5. Écrire les prédicats permettant d'extraire les informations d'un nœud.
6. Écrire un prédicat qui effectue la recherche avec A*. Il doit indiquer à la fin le coût total de la solution, le nombre de nœuds générés et le nombre de nœuds visités pour arriver à la solution.

Exercice 3 Un algorithme de recherche heuristique utilise la fonction d'évaluation $f(n) = (2 - w)g(n) + wh(n)$. Avec quelles valeurs de w l'algorithme est optimal ? Quel genre de recherche l'algorithme effectue lorsque $w = 0$? $w = 1$? $w = 2$?

Exercice 4 On se propose un algorithme de recherche où les nœuds dans la liste d'attente sont ordonnés en ordre croissant d'une fonction f. Le nœud n suivant à visiter est celui qui possède la valeur $f(n)$ minimale. Quelle est la fonction f pour que l'algorithme devienne :

1. l'algorithme de recherche en largeur ?
2. l'algorithme de recherche en profondeur ?
3. l'algorithme de recherche en coût uniforme ?
4. l'algorithme de recherche gloutonne ?
5. l'algorithme de recherche A* ?

Exercice 5 Problème du taquin (puzzle-n, **à programmer à la maison**)

1. Donner deux heuristiques possibles h_1 et h_2.
2. Créer un générateur aléatoire de problème solubles qui place m problèmes dans un fichier texte. Prener par exemple $m = 1000$. Un problème correspond à une ligne du fichier. Il se compose de n^2 nombre de 0 à $n^2 - 1$ séparés par un virgule, sans espaces. Le fichier comporte un en-tête en première ligne indiquant la dimension du taquin.
3. Ecrire les procédures de recherche suivantes et les tester sur la base de problèmes : BFS, IDS, A$^*(h_1)$ et A$^*(h_2)$. Pour chaque problème, la recherche doit retourner la profondeur à laquelle la solution a été trouvée et le nombre de nœuds dévéloppés.