COMP3411/9414/9814
Artificial Intelligence
Session 1, 2017

Assignment 2 – Heuristics and Search

Student Name:	Boshen Hu
ZID:	z5034054
Lecturer:	Alan Blair

Boshen Hu, z5034054

Question 1 – Maze Search Heuristics

(a)

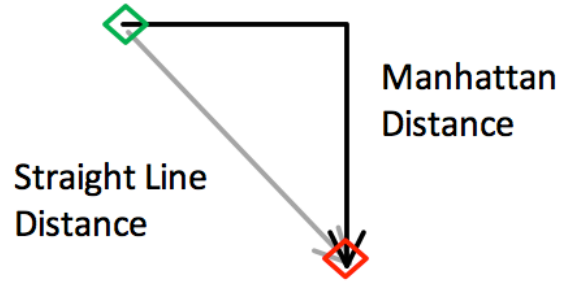


Figure 1.1

When the agent can move only up, down, left or right, the Manhattan-Distance heuristic would be admissible and used, and its value is always equal or larger than the value of the Straight-Line-Distance heuristic (see the picture Figure 1.1). Therefore, a better $h(x, y, x_G, y_G)$ would be:

$$h(x, y, x_G, y_G) = |x - x_G| + |y - y_G|$$

(b)

(i) Yes, the Straight-Line-Distance heuristic is still admissible. If it is admissible, it should $h_{SLD}(x, y, x_G, y_G) <= h^*(x, y, x_G, y_G)$. The "worst" case is $h_{SLD}(x, y, x_G, y_G) = h^*(x, y, x_G, y_G)$ when the shortest path is to move diagonally by all steps. Therefore, the Straight-Line-Distance heuristic is still admissible.

(ii)

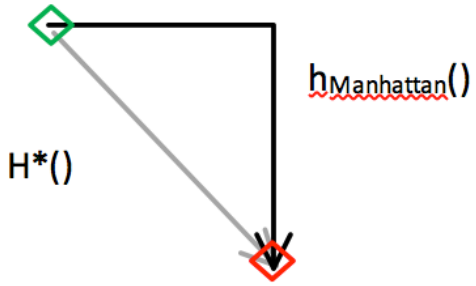


Figure 1.2

No, it is now admissible now, for the reason that in the case shown by Figure 1.2, $h_{Manhattan}(x, y, x_G, y_G) > h^*(x, y, x_G, y_G)$ makes the Manhattan-Distance heuristic would not be admissible.

(iii)

$$h_{SLD}(x, y, x_G, y_G) = \sqrt{(x - x_G)^2 + (y - y_G)^2}$$

Boshen Hu, z5034054

Question 2 – Search Algorithms for the 15-Puzzle

(a)

	Start10	Start12	Start20	Start30	Start40
UCS	2565	Mem	Mem	Mem	Mem
IDS	2407	13182	5297410	Time	Time
A*	33	26	915	Mem	Mem
IDA*(Man)	29	21	952	17297	112571
IDA*(Mis)	Time	Time	Time	Time	Time

(b)

The Count Misplaced Tiles heuristic is to compute the sum of the misplaced tiles. For simplifying the operation, I do not change the heads of the modified rules (though they look a little bit weird…). This is what I have changed (from the first picture to the second):

```
37 mandist(X/Y, X1/Y1, D) :-
38     dif(X, X1, Dx),
39     dif(Y, Y1, Dy),
40     D1 is Dx + Dy,
41     D1 > 0,
42     D is 1, !.
43
44 mandist(X/Y, X1/Y1, D) :-
45     dif(X, X1, Dx),
46     dif(Y, Y1, Dy),
47     D1 is Dx + Dy,
48     D1 = 0,
49     D is 0, !.
50
51
52 dif(A, B, D) :-                % D is
53     D is A-B, D = 0, D is 1, !.
54
55 dif(A, B, D) :-                % D is
56     D is A-B, not(D = 0), D is 0.
```

```
37 mandist(X/Y, X1/Y1, D) :-      % D is Ma
38     dif(X, X1, Dx),
39     dif(Y, Y1, Dy),
40     D is Dx + Dy.
41
42 dif(A, B, D) :-                % D is |A
43     D is A-B, D >= 0, !.
44
45 dif(A, B, D) :-                % D is |A
46     D is B-A.
47
48 goal(Pos) :-
49     length(Pos,9),
50     goal3(Pos).
```

This is what I got from the searches for [idastar] using [puzzle15mis].

IDA*(Mis)	Time	Time	Time	Time	Time

(c) From the form, it can be observed that UCS is easy to run out of memory, although IDS could help to save memory but still not efficient with start30 and above. The Informed search A* is a more efficient way, but still cause a giant cost and run out of the memory in complicated situations. Therefore, an Iterative Deepening version of the Heuristic Path Search algorithm would contribute to reduce the cost. However, it should be noticed that the efficiency of an IDA search is highly dependent on the heuristic it uses. IDA*(Mis) performs no good in every situation of the searches, while IDA* by using the Manhattan-Distance heuristic is the most efficient algorithm among these five algorithms.

Boshen Hu, z5034054

Question 3 – Heuristic Path Search for the 15-Puzzle

(a)

	Start50		Start60		Start64	
IDA*	50	1462512	60	321252368	64	1209086782
1.2	52	191438	62	230861	66	431033
1.4	66	116174	82	3673	94	188917
1.6	100	34647	148	55626	162	235852
1.8	236	6942	314	8816	344	2529
Greedy	164	5447	166	1617	184	2174

(b)

```
34 depthlim(Path, Node, G, F_limit, Sol, G2)  :-
35     nb_getval(counter, N),
36     N1 is N + 1,
37     nb_setval(counter, N1),
38     % write(Node),nl,    % print nodes as they are expand
39     s(Node, Node1, C),
40     not(member(Node1, Path)),      % Prevent a cycle
41     G1 is G + C,
42     h(Node1, H1),
43     F1 is G1 + H1,
44     F1 =< F_limit,
45     depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

```
34 depthlim(Path, Node, G, F_limit, Sol, G2)  :-
35     nb_getval(counter, N),
36     N1 is N + 1,
37     nb_setval(counter, N1),
38     % write(Node),nl,    % print nodes as they are expanded
39     s(Node, Node1, C),
40     not(member(Node1, Path)),      % Prevent a cycle
41     G1 is G + C,
42     h(Node1, H1),
43     W is 2,
44     F1 is (2-W)*G1 + W*H1,
45     F1 =< F_limit,
46     depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
```

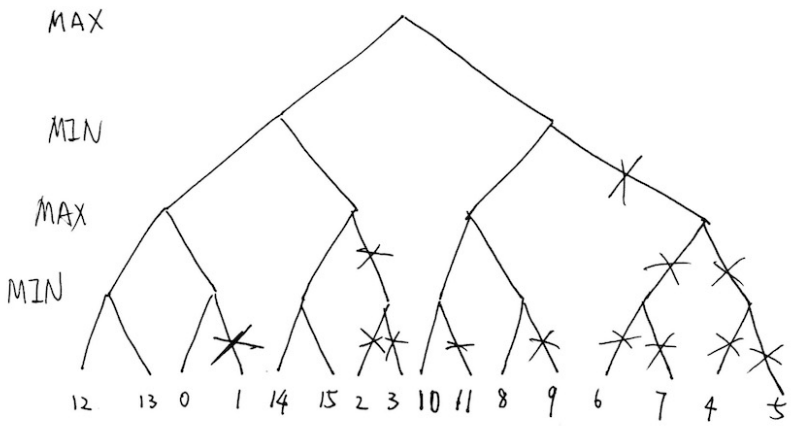
(c)

The filled table has shown in (a). We can see that IDA* always get the optimal result, however, it spends the longest time among the six algorithms. The speed of the algorithm is generally directly proportional to the value of ω (ω = 1.2, 1.4, 1.6, 1.8)despite the exception at ω=1.4, Start60), while quality of the solution and the value of ω is in inverse proportion. Although greedy algorithm could be equivalent to A* with ω = 2, i.e. f(n) = 2h(n), the quality of the solution is not the worst. In general, a conclusion could be given that if we need to balance the speed and quality, we need to carefully choose the ω, and give chance to greedy algorithm; if we need only the best solution, we could only consider IDA* algorithm

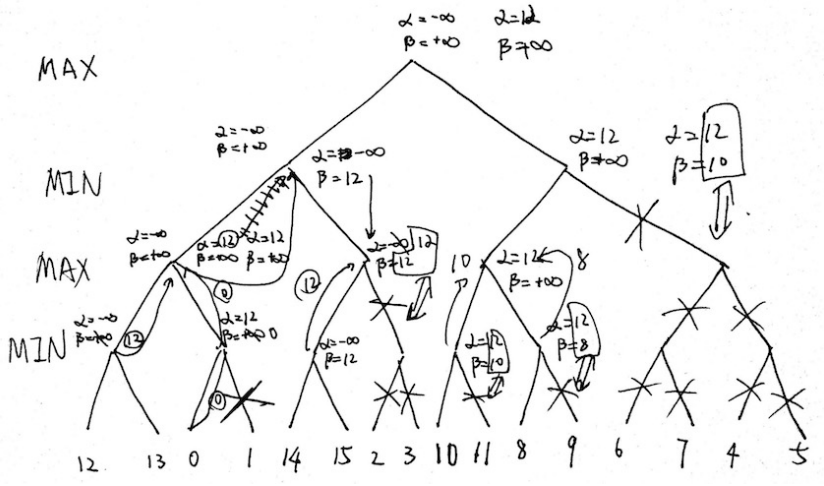
Boshen Hu, z5034054

Question 4 – Game Trees and Pruning

(a)

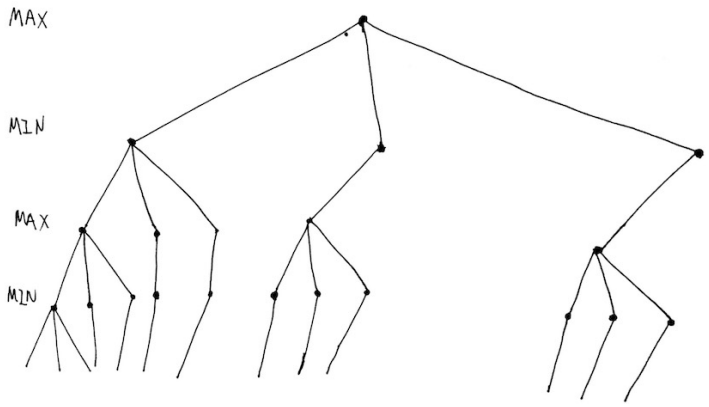


(b)



In this tree, only 12, 13, 0, 14, 15, 10, 8 are evaluated, therefore there are 7 leaves.

(c)



(Pruned nodes and branches have not drawn in the picture.)

(d)

It would be $O(b^{(d/2)})$ where b is the number of branches of a node and d is the depth of the tree. The maximum number of the evaluated leaves would be $b*b*\ldots*b = b^d$. In the optimal situation of pruning in an alpha-beta tree, the best move is always examined first. That is to say, in every one of two layers, we only need to examine one node. Therefore, the complexity of the best situation is $O(b*1*\ldots*b)$ or $O(b*1*\ldots*1)$ which is about $O(b^{(d/2)})$.