# COMP9414 Assignment 2 - Heuristics and Search

Shu Hao Ivan Teong (z3386180)

April 30, 2015

**Abstract**

Artificial intelligence assignment investigating heuristics, different types of pathfinding searches, game trees and alpha-beta pruning.

## Question 1 - Maze Search Heuristics

### Q1(a) Admissible Heuristic dominating Straight-Line Distance Heuristic - Manhattan Distance Heuristic:

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

Also known as the taxicab distance or L1 distance, which calculates the total travel distance in either direction. Notice that if there are no obstacles, this heuristic is in fact exact. Clearly, introducing obstacles can only increase the length of the optimal path to get to the goal. Thus, this heuristic is admissible.

This heuristic dominates the straight line distance heuristic because it will always be bigger or equal to the straight line distance heuristic, since the straight line distance heuristic is always the shortest distance between 2 points. However, it is a better heuristic especially for maze search. This is because the typical search costs which is the average number of nodes expanded, will always be less than that of the straight line distance heuristic.

### Q1(b)(i)

Even when the agent can move diagonally, the straight line distance will still be considered as the shortest possible distance between 2 points, so it is guaranteed to never overestimate the actual cost to reach its goal position. Hence, it is still admissible in this case.

### Q1(b)(ii)

For the heuristic that is used in Q1(a) which is the Manhattan Distance Heuristic, if there are no obstacles between the current and target positions, the estimated cost needed to get to the target position will be exact. However, in this case where there are obstacles in the way, the number of steps (actual cost) will increase while the heuristic value will remain unchanged.

Hence, it will still be admissible as the estimated heuristic cost will always be less than than the actual cost which increases with the obstacles.

### Q1(b)(iii) Admissible Heuristic allowing movements up, down, left, right or diagonal - Chebyshev Distance Heuristic:

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

Maximum distance (also known as the L$\infty$ distance) of the current position to the goal position, where — · — is the absolute value operator. That is, the L$\infty$ distance is the maximum travel distance in either direction x or direction y.

Hence, the steps required to get to the goal is at least the maximum of travel in either direction. Thus, this heuristic is admissible.

## Question 2 - Comparing BFS and A* for the 8-Puzzle

Let S be Specified Start Position and N be Total Number of States Expanded.

### Q2(a) Breadth First Search, excluding Repeated States

| S | N |
|---|---|
| start1 | 41 |
| start2 | 173 |
| start3 | 2409 |
| start4 | 9508 |
| start5 | 58419 |

### Q2(b) Breadth First Search, including Repeated States

| S | N |
|---|---|
| start1 | 139 |
| start2 | 1857 |

### Q2(c) A*Search, with Total Manhattan Distance Heuristic

| S | N |
|---|---|
| start1 | 6 |
| start2 | 11 |
| start3 | 19 |
| start4 | 61 |
| start5 | 213 |

## Q2(d) A*Search, with Count Misplaced Tiles Heuristic

| S | N |
|---|---|
| start1 | 9 |
| start2 | 15 |
| start3 | 79 |
| start4 | 393 |
| start5 | 1971 |

**Code for A*Search, with Count Misplaced Tiles Heuristic:**

```
% H is the Hamming Distance, the number of misplaced tiles.
% If X and Y for initial and final positions the same, add
% nothing to H counter and stop at cut.
hamdist(X/Y, X/Y, H) :-
    H is 0, !.

% If X and Y for initial and final positions differ, add
% 1 to H counter and stop at cut (use _ to prevent singleton
% variable error).
hamdist(_/_, _/_, H) :-
    H is 1, !.

totdist([Tile|Tiles], [Position|Positions], D) :-
%     mandist(Tile, Position, D1), % comment out for hamdist
    hamdist(Tiles, Positions, D1),
    totdist(Tiles, Positions, D2),
    D is D1 + D2.
```

## Q2(e) Ranking of Searches from Slowest to Fastest:

1. Breadth First Search, including Repeated States (Slowest)
2. Breadth First Search, excluding Repeated States
3. A*Search, with Count Misplaced Tiles Heuristic
4. A*Search, with Total Manhattan Distance Heuristic (Fastest)

## Q2(f) Invent 3 new starting positions, requiring 20, 23 and 26 moves respectively, running fastest of 4 searches on these states:

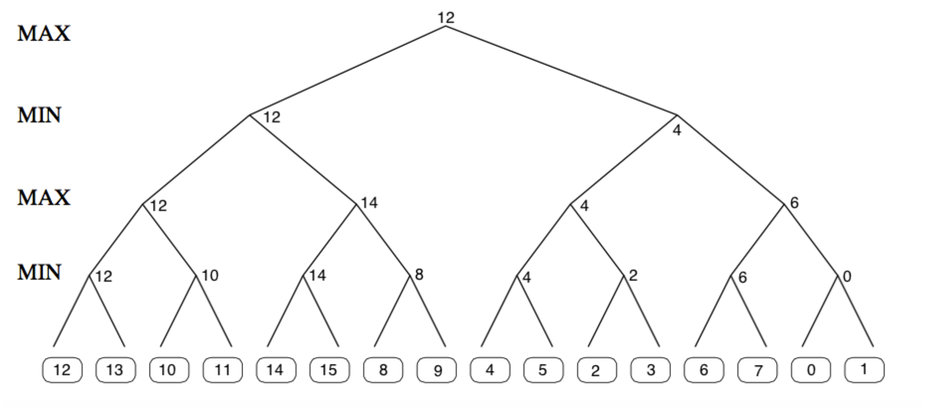
**A*Search, with Total Manhattan Distance Heuristic and the 3 new starting positions, start6, start7 and start8**

| S | N |
|---|---|
| start1 | 6 |
| start2 | 11 |
| start3 | 19 |
| start4 | 61 |
| start5 | 213 |
| start6 | 284 |
| start7 | 1151 |
| start8 | 4743 |

```
start6([1/3,3/1,1/2,2/1,3/2,2/3,2/2,3/3,1/1]).    %  20  moves

             State
        +-----------+
1       | 8   2     |
2       | 3   6   5 |
3       | 1   4   7 |
        +-----------+
          1   2   3

start7([2/1,3/2,1/3,3/1,2/2,2/3,1/2,3/3,1/1]).    %  23  moves

             State
        +-----------+
1       | 8   6   2 |
2       |     4   5 |
3       | 3   1   7 |
        +-----------+
          1   2   3

start8([2/2,3/3,1/2,3/1,3/2,1/3,1/1,2/3,2/1]).    %  26  moves

             State
        +-----------+
1       | 6   2   5 |
2       | 8       7 |
3       | 3   4   1 |
        +-----------+
          1   2   3
```

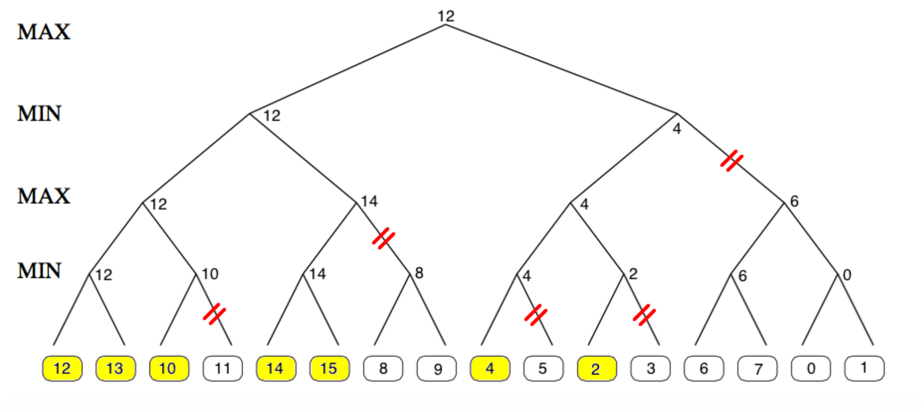
## Question 3 - Game Trees and Pruning

### Q3(a) My game tree of depth 4, where each internal node has exactly 2 children, with values from 0 to 15 for the leaves and in such a way that the alpha-beta algorithm prunes as many nodes as possible:

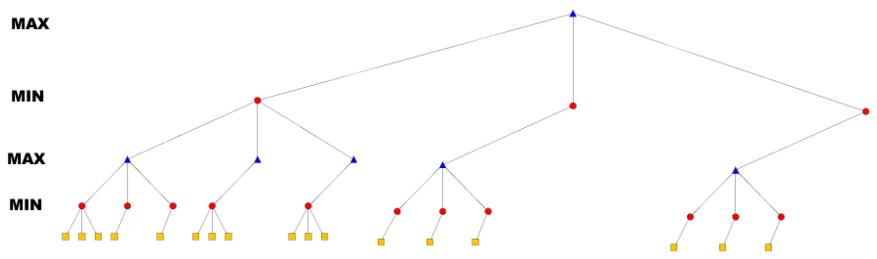


## Q3(b)

7 of the original 16 game tree leaves are evaluated after tracing through the alpha-beta algorithm on this tree.



## Q3(c) Shape of pruned game tree of depth 4, where each internal node has exactly 3 children:



17 of the original 81 game tree leaves are evaluated after tracing through the alpha-beta algorithm on this tree. The above diagram shows the shape of the pruned game tree.

## Q3(d) Time complexity of alpha-beta search:

If the best move is always examined first at every branch of the tree, the move ordering for the search is optimal (perfect ordering) and hence, increases the effectiveness of pruning.

With "perfect ordering", time complexity =

$$O(b^{d/2})$$

where b = branching factor, d = depth of the tree, O = the limiting behavior of a function when the argument tends towards a particular value or infinity and the base case is when all the preferred nodes are expanded first.

The number of leaf node positions evaluated is about

$$O(b * 1 * b * 1 * ... * b)$$

for odd depth and

$$O(b * 1 * b * 1 * ... * 1)$$

for even depth, or

$$O(b^{d/2})$$

In the latter case, where the ply of a search is even, the effective branching factor is reduced to its square root, or, equivalently, the search can go twice as deep as plain minimax with the same amount of computation. This is because to prove that a "bad" move is bad, we only need to consider one (good) reply, although we still need to consider all replies to prove that a "good" move is good.

The explanation of b*1*b*1*... is that all the first player's moves must be studied to find the best one, but for each, only the best second player's move is needed to refute all but the first (and best) first player move – alpha–beta ensures that no other second player moves need to be considered. Put simply, you "skip" every two levels, increasing the efficiency of pruning.