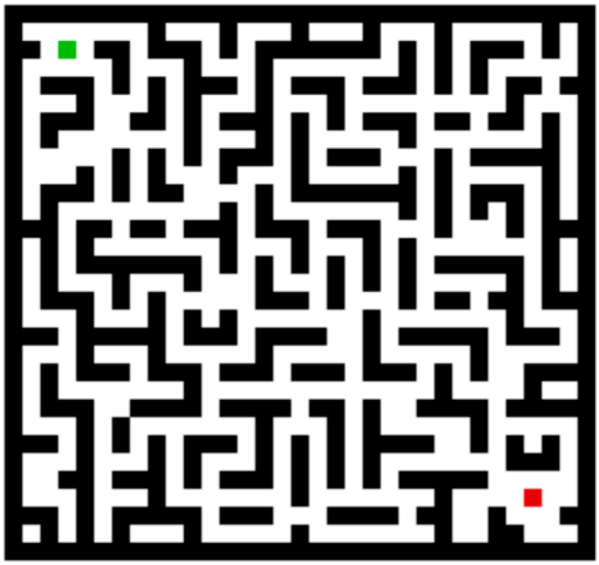
COMP3411/9414/9814 Artificial Intelligence
Session 1, 2017

Assignment 2 – Heuristics and Search

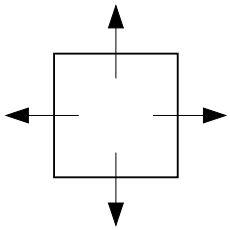
Due: Sunday 30 April, 11:59pm
Marks: 10% of final assessment

## Question 1 – Maze Search Heuristics

Consider the problem of an agent moving around in a 2-dimensional maze, trying to get from its current position $(x, y)$ to the Goal position $(x_G, y_G)$ in as few moves as possible, avoiding obstacles along the way.



(a) Assume that at each time step, the agent can move one unit either up, down, left or right, to the centre of an adjacent grid square:



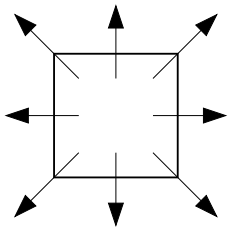
One admissible heuristic for this problem is the Straight-Line-Distance heuristic:

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

However, this is not the best heuristic. Name another admissible heuristic which dominates the Straight-Line-Distance heuristic, and write the formula for it in the format:

$$h(x, y, x_G, y_G) = \quad \ldots$$

(b) Now assume that at each time step, the agent can take one step either up, down, left, right or **diagonally**. When it moves diagonally, it travels to the centre of a diagonally neighboring grid square, but a diagonal step is still considered to have the same "cost" (i.e. one "move") as a horizontal or vertical step (like a King move in Chess).



(i) Assuming that the **cost** of a path is the total number of **moves** to reach the goal, is the Straight-Line-Distance heuristic still admissible? Explain why.

(ii) Is your heuristic from part (a) still admissible? Explain why.

(iii) Try to devise the best admissible heuristic you can for this problem, and write a formula for it in the format:

$$h(x, y, x_G, y_G) = \quad \ldots$$

## Question 2 – Search Algorithms for the 15-Puzzle

In this question you will construct a table showing the number of states expanded when the 15-puzzle is solved, from various starting positions, using five different searches:

(i) Uniform Cost Search (with Dijkstra's Algorithm)
(ii) Iterative Deepening Search
(iii) A*Search
(iv) Iterative Deepening A*Search with Manhattan distance heuristic
(v) Iterative Deepening A*Search with Misplaced Tiles heuristic

Go to the Course Web Site, Week 3 Prolog Code: Path Search, scroll to the Activity at the bottom of the page and click on "`prolog_search.zip`". Unzip the file and change directory to prolog_search, e.g.

```
unzip prolog_search.zip
cd prolog_search
```

Start `prolog` and load `puzzle15.pl` and `ucsdijkstra.pl` by typing

```
[puzzle15].
[ucsdijkstra].
```

Then invoke the search for the specified `start10` position by typing

```
start10(Pos),solve(Pos,Sol,G,N),showsol(Sol).
```

When the answer comes back, just hit Enter/Return. This version of UCS uses Dijkstra's algorithm which is memory efficient, but is designed to return only one answer. Note that the length of the path is returned as `G`, and the total number of states expanded during the search is returned as `N`.

(a) Draw up a table with five rows and five columns. Label the rows as UCS, IDS, A*, IDA*(Man) and IDA*(Mis) and the columns as `start10`, `start12`, `start20`, `start30` and `start40`. Run each of the following algorithms on each of the 5 start states:

(i) `[ucsdijkstra]`
(ii) `[ideepsearch]`
(iii) `[astar]`
(iv) `[idastar]`

In each case, record in your table the number of nodes generated during the search. If the algorithm runs out of memory, just write "Mem" in your table. If the code runs for five minutes without producing output, terminate the process by typing Control-C and then "a", and write "Time" in your table.

(b) Now copy `puzzle15.pl` to a new file `puzzle15mis.pl` and modify the code for this new file so that it uses the Count Misplaced Tiles heuristic instead of the Total Manhattan Distance heuristic (you are free to use "cut" if you wish).

In your submitted document, briefly show the section of code that was changed, and the replacement code.

(v) Repeat the searches for `[idastar]` using `[puzzle15mis]` instead of `[puzzle15]` and add the results to the last row of your table in part (a).

(c) Briefly discuss the efficiency of these five algorithms.

## Question 3 – Heuristic Path Search for the 15-Puzzle

In this question you will be exploring an Iterative Deepening version of the Heuristic Path Search algorithm discussed in the Week 4 Tutorial. Draw up a table in the following format:

| | start50 | | start60 | | start64 | |
|---|---|---|---|---|---|---|
| IDA* | 50 | 1462512 | 60 | 321252368 | 64 | 1209086782 |
| 1.2<br>1.4<br>1.6<br>1.8 | | | | | | |
| Greedy | | | | | | |

The top row of the table has been filled in for you (to save you from running some rather long computations).

(a) Run `[greedy]` for `start50`, `start60` and `start64`, and record the values returned for `G` and `N` in the last row of your table. Remember to use the Manhattan Distance heuristic defined in `puzzle15.pl` (not the Misplaced Tile heuristic from the previous question).

(b) Now copy `idastar.pl` to a new file `heuristic.pl` and modify the code of this new file so that it uses an Iterative Deepening version of the Heuristic Path Search algorithm discussed in the Week 4 Tutorial Exercise, with $w = 1.2$ .

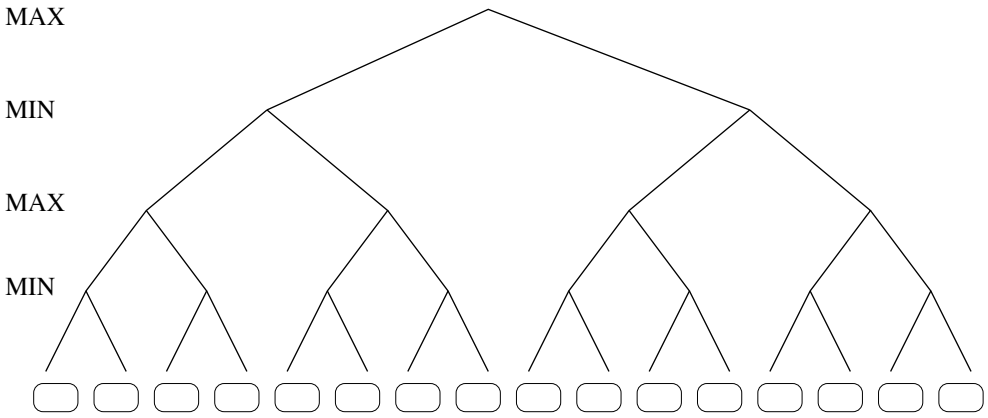
In your submitted document, briefly show the section of code that was changed, and the replacement code.

(c) Run `[heuristic]` on `start50`, `start60` and `start64` and record the values of `G` and `N` in your table. Now modify your code so that the value of $w$ is 1.4, 1.6, 1.8; in each case, run the algorithm on the same three start states and record the values of `G` and `N` in your table.

(d) Briefly discuss the tradeoff between speed and quality of solution for these six algorithms.

## Question 4 - Game Trees and Pruning

(a) Consider a game tree of depth 4, where each internal node has exactly **two** children (shown below). Fill in the leaves of this game tree with all of the values from 0 to 15, in such a way that the alpha-beta algorithm prunes as many nodes as possible. Hint: make sure that, at each branch of the tree, all the leaves in the left subtree are preferable to all the leaves in the right subtree (for the player whose turn it is to move).




(b) Trace through the alpha-beta search algorithm on your tree. How many of the original 16 leaves are evaluated?

(c) Now consider another game tree of depth 4, but where each internal node has exactly **three** children. Assume that the leaves have been assigned in such a way that the alpha-beta algorithm prunes as many nodes as possible. Draw the shape of the pruned tree. How many of the original 81 leaves will be evaluated?

Hint: If you look closely at the pruned tree from part (b) you will see a pattern. Some nodes explore all of their children; other nodes explore only their leftmost child and prune the other children. The path down the extreme left side of the tree is called the line of best play or Principal Variation (PV). Nodes along this path are called PV-nodes. PV-nodes explore all of their children. If we follow a path starting from a PV-node but proceeding through non-PV nodes, we see an alternation between nodes which explore all of their children, and those which explore only one child. By reproducing this pattern for the tree in part (c), you should be able to draw the shape of the pruned tree (without actually assigning values to the leaves or tracing through the alpha-beta algorithm).

(d) What is the time complexity of alpha-beta search, if the best move is always examined first (at every branch of the tree)? Explain why.

## Submission

This assignment must be submitted electronically.
COMP9414/9814 students should submit by typing

```
give cs9414 hw2 ...
```

COMP3411 students should submit by typing

```
give cs3411 hw2 ...
```

The give script will accept `*.pdf *.txt *.doc *.rtf`
If you prefer some other format, let me know.
Late submissions will incur a penalty of 15% per day, applied to the maximum mark.

Group submissions will not be allowed. By all means, discuss the assignment with your fellow students. But you must write (or type) your answers individually. Do NOT copy anyone else's assignment, or send your assignment to any other student.

Good luck!