COMP3411/9414/9814 Artificial Intelligence
Session 1, 2018

Assignment 2 – Heuristics and Search

Due: Sunday 29 April, 11:59pm
Marks: 10% of final assessment

Question 1: Search Algorithms for the 15-Puzzle (2 marks)

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 four different searches:

(i) Uniform Cost Search (with Dijkstra's Algorithm)
(ii) Iterative Deepening Search
(iii) A*Search (using the Manhattan Distance heuristic)
(iv) Iterative Deepening A*Search

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 four rows and five columns. Label the rows as UCS, IDS, A* and IDA*, 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. Note that you will need to re-start prolog each time you switch to a different search.

(b) Briefly discuss the efficiency of these four algorithms (including both time and memory usage).

Question 2: Heuristic Path Search for 15-Puzzle (2 marks)

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	14642512	60	321252368	64	1209086782
1.2						
1.4						
1.6						
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 (using the Manhattan Distance heuristic defined in `puzzle15.pl`).

(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 $w = 1.2$.

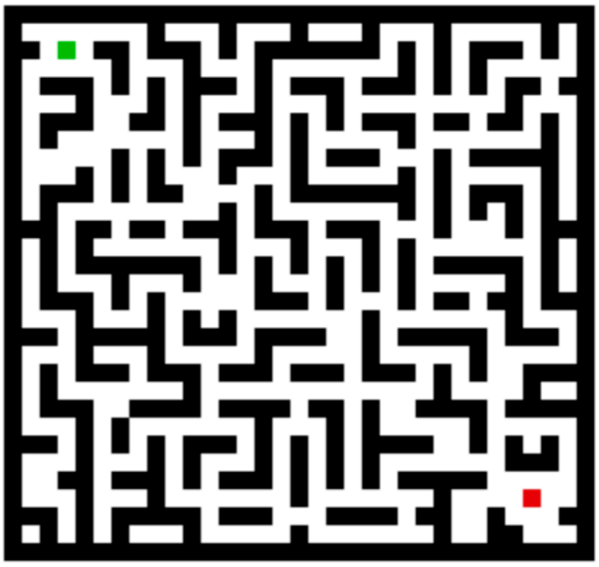
In your submitted document, briefly show the section of code that was changed, and the replacement code.

(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; in each case, run the algorithm on the same three start states and record the values of `G` and `N` in your table.

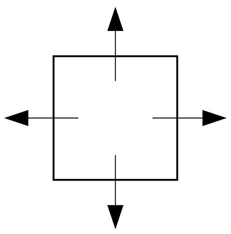
(d) Briefly discuss the tradeoff between speed and quality of solution for these five algorithms.

Question 3: Maze Search Heuristics (2 marks)

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:



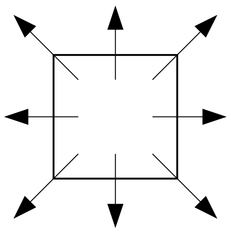
One admissible heuristic for this problem is the Straight-Line-Distance heuristic:

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

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:

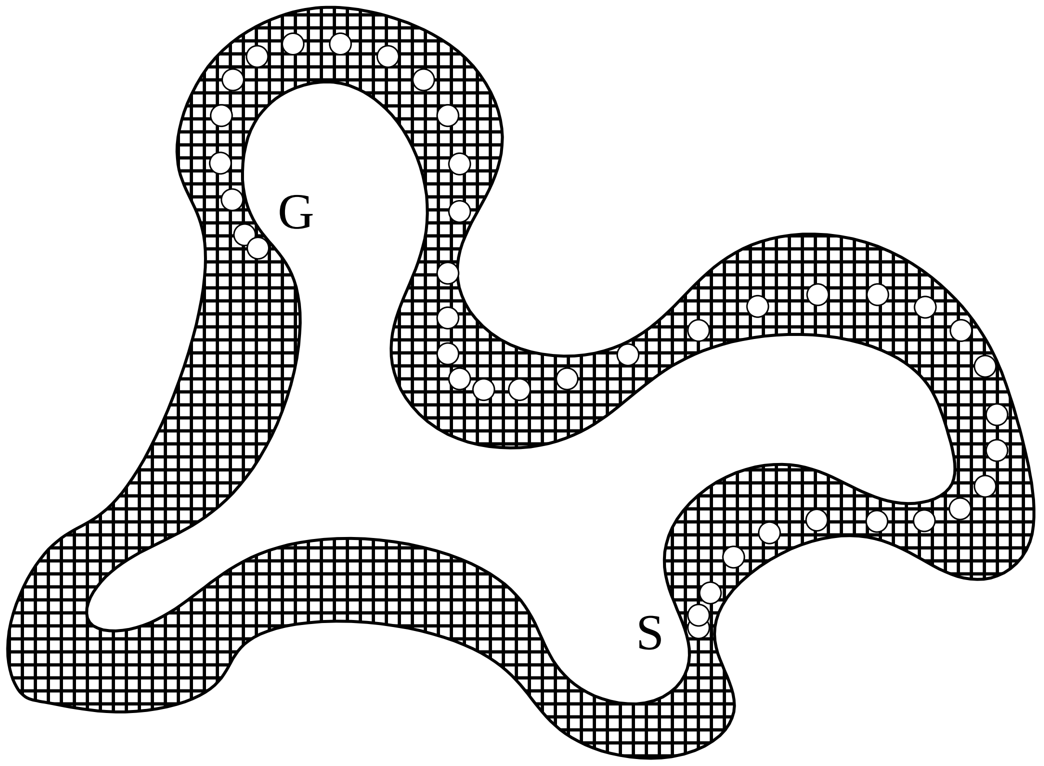
$$h(x, y, x_G, y_G) = \quad \ldots$$

Question 4: Graph Paper Grand Prix (4 marks)

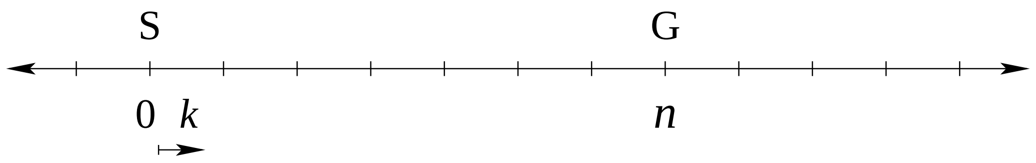
We saw in lectures how the Straight-Line-Distance heuristic can be used to find the shortest-distance path between two points. However, in many robotic applications we wish to minimize not the distance but rather the **time** taken to traverse the path, by speeding up on the straight bits and avoiding sharp corners.

In this question you will be exploring a simplified version of this problem in the form of a game known as Graph Paper Grand Prix (GPGP). [I was first introduced to this game in Year 9 of high school, as a distraction for the last day of class before the summer holiday].

To play GPGP, you first need to draw the outline of a racing track on a sheet of graph paper, and choose a start location S = (r_S, c_S) as well as a Goal location G = (r_G, c_G) where r and c are the row and column. The agent begins at location S, with velocity (0,0). A "state" for the agent consists of a position (r, c) and a velocity (u, v), where r, c, u, v are (positive or negative) integers.



At each time step, the agent has the opportunity to increase or decrease each component of its velocity by one unit, or to keep it the same. In other words, the agent must choose an acceleration vector (a, b) with $a, b \in \{-1, 0, +1\}$. It then updates its velocity from (u, v) to $(u', v') = (u + a, v + b)$, and updates its position – using the *new* velocity – from (r, c) to $(r + u', c + v')$. The aim of the game is to travel as fast as possible, but without crashing into any obstacles or running off the edge of the track, and eventually stop at the Goal with velocity $(0, 0)$.



We first consider a 1-dimensional version of GPGP where the vehicle moves through integer locations on a number line, with no obstacles. Assume the Goal is at location n, and that the agent starts at location 0, with velocity k. We will use $M(n, k)$ to denote the minimum number of time steps required to arrive and stop at the Goal. Clearly $M(-n, -k) = M(n, k)$ so we only need to compute $M(n, k)$ for $k \geq 0$.

(a) Starting with the special case $k = 0$, compute $M(n, 0)$ for $1 \leq n \leq 21$ by writing down the optimal sequence of actions for all n between 1 and 21. For example, if $n = 7$ then the optimal sequence is $[+ + \circ - \circ -]$ so $M(7, 0) = 6$. (When multiple solutions exist, you should pick the one which goes "fast early" i.e. with all the +'s at the beginning.)

(b) Assume $n \geq 0$. By extrapolating patterns in the sequences from part (a), explain why the general formula for $M(n, 0)$ is

$$M(n, 0) = \left\lceil 2\sqrt{n} \right\rceil,$$

where $\lceil z \rceil$ denotes z rounded up to the nearest integer.
Hint: Do not try to use recurrence relations. You should instead use this identity:

$$\left\lceil 2\sqrt{n} \right\rceil = \begin{cases} 2s+1, & \text{if} \quad s^2 \quad < n \leq s(s+1) \\ 2s+2, & \text{if } s(s+1) < n \leq \ (s+1)^2 \end{cases}$$

(c) Assuming the result from part (b), show that if $k \geq 0$ and $n \geq \frac{1}{2}k(k-1)$ then

$$M(n, k) = \left\lceil 2\sqrt{n + \tfrac{1}{2}k(k+1)} \right\rceil - k$$

Hint: Consider the path of the agent as part of a larger path.

(d) Derive a formula for $M(n, k)$ in the case where $k \geq 0$ and $n < \frac{1}{2}k(k-1)$.

(e) Write down an admissible heuristic (that approximates the true number of moves as closely as possible) for the original 2-dimensional GPGP game in terms of the function $M()$ derived above.
Hint: Your heuristic should be of this form:

$$h\,(r, c, u, v, r_G, c_G) = \max(M(..,..), M(..,..))$$

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!