**COMP 3710 Applied Artificial Intelligence**

**Seminar/Lab 3.**

**A\* Algorithm, and *n*-Puzzle Game**

1. **Objectives**
* Understand how A\* algorithm works
* Use of A\* algorithm for solving 8-puzzle game
1. **How to implement A\* algorithm for 8-puzzle game**
	* An *n* x *n* board is a 1-dimensional array of *n* x *n* elements.
	* The goal board is [1, 2, 3, 4, 5, 6, 7, 8, 0].

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 0 |

0 means empty tile.

* The following functions are included in this library – <http://cs.tru.ca/~mlee/comp3710/Software/board_game.min.js>
	+ make\_initial\_board(n) Return an 1 x *n* x *n* array. *n* could be 3, not 8.

E.g., [1, 3, 7, 4, 2, 8, 0, 5, 6].

* + index\_of\_empty\_cell(board) Return the index of the empty cell, i.e., 0, in board.

E.g., the return value for [1, 3, 7, 4, 2, 8, 0, 5, 6] is 6.

* + Queue MIN priority queue

E.g., var expandedQ = new Queue();

* + - push(id, priority, obj) Push an object into the queue.
		- pop(id) Return the object of id.

The object will be removed from the queue.

* + - get(id) Return the object of id.

The object will NOT be removed from the queue.

* + - popTheHighestPriorityOne() Return the object of the highest priority. (Note that Queue is a MIN priority queue.)

The object will be removed from the queue.

* + - getTheHighestPriorityOne() Return the object of the highest priority.

The object will NOT be removed from the queue.

* + - isIn(id) Return true if an object of id is in the queue, otherwise false.
		- update(id, priority, obj) Update the object of id with different priority and data.
* **board\_game\_student.html**
	+ You can use the f-value as the priority in the queue.
	+ The id of a board (e.g., 1 × 9 array) can be the string converted from the array. E.g., [1, 3, 7, 4, 2, 8, 0, 5, 6] -> “1,3,7,4,2,8,0,5,6” can be the id of the board.
	+ var expandedQ = new Queue();
	+ var node = **create\_node**(initial\_board, 0, null);

A **node** is an object that contains a **board**, its **g-value**, and **parent node**.

* + expandedQ.push(get\_id\_of\_node(node), get\_**fvalue**\_of\_node(node), **node**); // NOT board
* For h-values, you can use the hamming distance heuristic (i.e., the number of missed tiles) or the Manhattan distance heuristic. But it is recommended to use the Manhattan distance heuristic.
1. **What to do**
* Utility functions used in exercise\_student.html and board\_game\_student.html – You do not have to change anything. They will be used in other parts of the programs.
	+ print\_message(message)
	+ get\_id\_of\_board(board)
	+ get\_id\_of\_node(node)
	+ get\_board\_of\_node(node)
	+ get\_gvalue\_of\_node(node)
	+ get\_hvalue\_of\_node(node)
	+ get\_fvalue\_of\_node(node)
	+ get\_parent\_node\_of\_node(node)
	+ get\_next\_boards(board, board0, board1, board2, board3)
	+ …
* exercise\_student.html – You need to try the 4 exercises in the class.
	+ 4 exercises to implement
		- is\_the\_goal\_board(board)
		- get\_heuristic\_value(board) – compute the heuristic value for board
		- create\_node(**board**, **gvalue**, **parent\_node**)
		- push\_or\_update\_node\_in\_expandedQ(node)
* board\_game\_student.html – You need to complete the followings
	+ the main part of A\* algorithm
	+ the above 4 functions
* Note. You may turn on ‘Developer tools’ to see any syntax errors in your programs.
1. **Assignment**
	1. Submission
* Submit board\_game\_student.html by email. Please include your name and ‘COMP 3710’ in the title.
	+ - Due
			* 11:59 pm, January 21, 2018 – with bonus 10%
			* 6:00 pm, January 23, 2018 – with the full marks
			* 6:00 pm, January 24, 2018 – with penalty 10%
			* 6:00 pm, January 25, 2018 – with penalty 20%
		- Total marks: 10
		- Any late submission will not be accepted.
		- No partial marks for any code that has syntax errors.