COMP 3981: Project Description (Part #2)

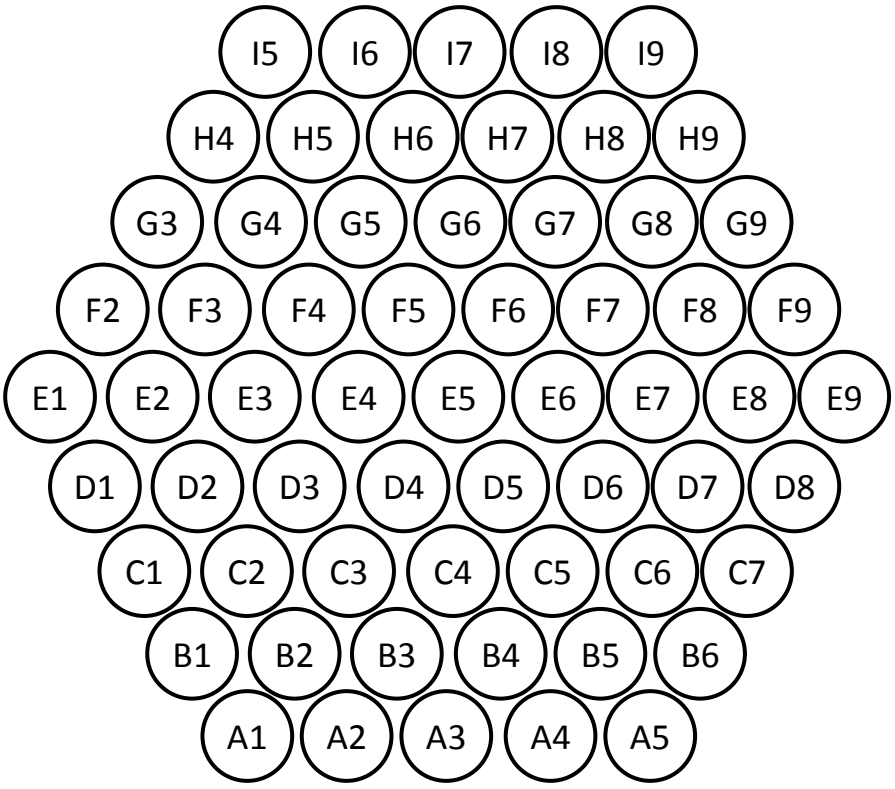
State Space Generator

General Instruction:

For this part of the project, you will work (with your team members) on developing the state space generator for the abalone game-playing agent. Your task is, given any legal configuration of the game board, to generate all the possible legal next-ply moves (in-line and side step) for either the black or white player as well as all the resulting game board configurations. Note that the state space generator needs to take the legality of the resulting move into consideration and that it is possible for different next-ply moves to end up with the same resulting game board configuration.

I Board Configuration

You may use any game board representation (coordinate system) within your program, however, the input and output interfaces must conform to the game board coordinate system shown below.



II Input

For each input game board configuration, a plain text file, Test<#>.input, will be provided to indicate the positions of the marbles. The file contains two (2) lines:

- 1st line:
  A single character, b or w, denoting the color of the marbles to be moved in the next ply.

- 2nd line:
  A string containing the coordinates of the marbles on the game board, as specified in (I), delimited by commas:

  $<Row_1><Col_1><color_1>,<Row_2><Col_2><color_2>,\ldots,<Row_n><Col_n><color_n>$

  where

Row_i	= {A, B, …, I}
Col_i	= {1, 2, …, 9}
$color_i$	= b for black marble or w for white marble
i	= {1, 2, …, n}, $n \leq 28$

  The marbles are ordered first by color (black before white), then by rows (alphabetically) and then by columns (numerically in an ascending order).

III Output

For each Test<#>.input, two (2) plain text files, Test<#>.move and Test<#>.board, are to be generated to indicate all the possible legal next-ply moves as well as all the resulting game board configurations, as follows:

- Test<#>.move:
  A file containing all possible legal next-ply moves, listed one move per row, for each input game board configuration. Note you can use any team-defined move representation that you specify in the submitted documentation.

- Test<#>.board:
  A file containing all resulting game board configurations, listed one resulting game board configuration per row, corresponding row-by-row to the next-ply moves listed in Test<#>.move. The positions of the marbles for each resulting game board configuration are to be indicated as with Test<#>.input:

  $<Row_1><Col_1><color_1>,<Row_2><Col_2><color_2>,\ldots,<Row_n><Col_n><color_n>$

where

Row_i = {A, B, …, I}
Col_i = {1, 2, …, 9}
$color_i$ = b for black marble or w for white marble
i = {1, 2, …, n}, n ≤ 28

The marbles are ordered first by color (black before white), then by rows (alphabetically) and then by columns (numerically in an ascending order).

IV Sample Input/Output Files

You are provided with two (2) sets of sample input/output files, Test1.input, Test1.board, Test2.input and Test2.board, which you can download from D2L for debugging and verification purposes. You are not provided with Test<#>.move, of which you can deduce from the provided files.

Note that your state space generator must work for any legal input game board configuration, as may be encountered during actual gameplay. You are strongly encouraged to comprehensively test your state space generator by creating additional input game board configurations and verifying the resulting game board configurations.

V Grading

Your state space generator will be tested with various input game board configurations to evaluate the correctness of your solution. For each input game board configuration, a correct match of the resulting game board configuration will earn 1 point and an incorrect (including missing or extra) resulting game board configuration will result in a 1 point deduction.

You will receive a mark of zero for any of the following:

1) your code does not compile or execute on the (freshly-imaged) instructor terminal in the option lab (SE12-306)
2) your state space generator does not take the input file as specified in (II)
3) your state space generator does not output the files as specified in (III)

You are also required to indent and comment your code based on best practices - poorly indented and commented code will receive deductions, as appropriate.

Deliverables:

1) [dropbox] State Space Generator Report – the submitted report must include, at a minimum, the following:
   a. state representation, actions and transition model used
   b. design and architecture of your state space generator, including how the legality of a resulting game board configuration is determined
   c. team-defined move representation used in Test<#>.move
   d. per team member contribution
   e. list of references used in the development of your state space generator
   f. any additional documentation (as determined by each team) pertinent to the assessment of your deliverable
2) [dropbox] Source code and executable including detailed instructions on how to build and run the executable

Due Date:

The deliverables for Project Part #2 are due on March 11, 2018 by 2359 (dropbox submissions) at the latest. Late submissions will not be accepted.

Note that there is no in-class demonstration for this part of the project.

Upcoming Project Due Dates:

April 3, 2018: Project Part #3 - Search Strategy and Optimization
April 10, 2018: Project Part #4 - Tournament and Final Deliverable