CS4725/CS6705 - Fall 2019

Assignment # 2

Due: Friday, October 4, by 12:30 pm (beginning of class)

Submission Instructions:

- Please include your name and student number, the course number and the assignment number on the front page of your assignment.
- Solutions to Questions 2-6 should be handed in on paper, by placing them in the CS4725/CS6705 bin on the E level of Gillin Hall (or by bringing them to class on the due date).
- Your code for Question 1 should be submitted through Desire2Learn, as explained on the next page.

---

1. **(10 marks)** In this exercise, you will write code for a simple PokerSquares player.

   **** Important note **:** Even if you plan to complete the final project in a group, this task on Assignment 2 is to be completed **individually**.

   - Download the partially completed file `FlushPlayer.java` from the `Assignments` folder on Desire2Learn and add it to the directory where you stored the Poker Squares files that you downloaded for exercise #1 on Assignment 1.

   - You should also download the slightly-revised `PokerSquares.java` file that is provided in the `Assignments` folder and use it to replace the file that you downloaded for Assignment 1.

   - In the version of `FlushPlayer.java` that is provided, the `getPlay` method always stores the card in the top-left position of the grid and then returns that location (row 0, column 0). (Note that this will cause an error on the second move if you actually try to run the program as it is because you will be playing the card in an already-occupied position.)

   - Your task is to modify the `getPlay` method so that it implements the simple strategy below. Note that there are certainly ways in which you could improve on the suggested strategy. However, you should implement exactly what is described below, to make it easier for your code to be tested.

     – If a card's suit is clubs, insert it in the first empty square in column 0 (the leftmost column). Similarly, use column 1 for diamonds, column 2 for hearts, and column 3 for spades.

     – If the appropriate column is already full, then insert the card in the first empty square in column 4 (the rightmost column).

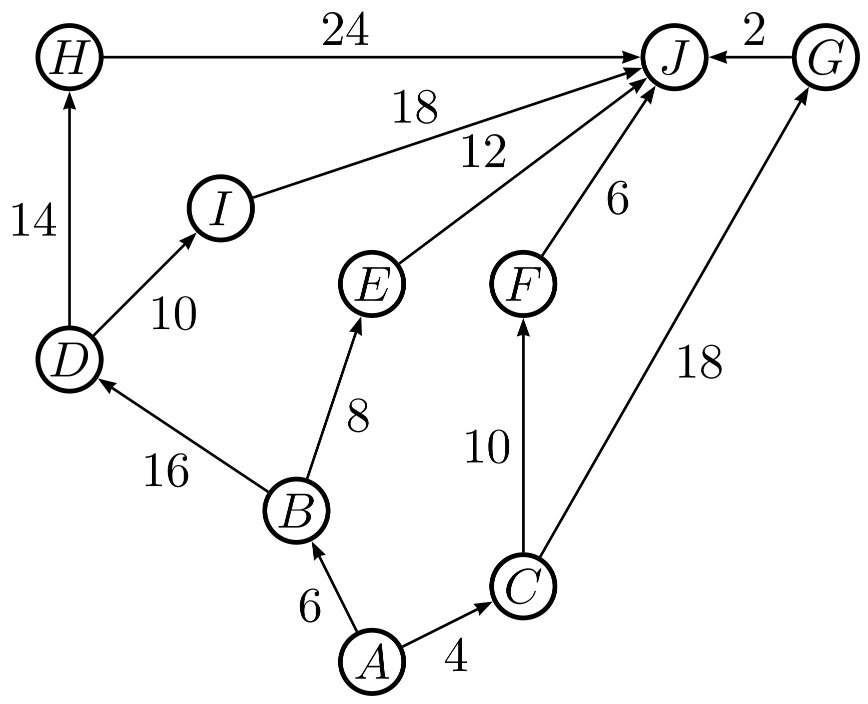
     – If column 4 is also full, then insert the card in the first available square in the grid, trying column 0 first, then column 1, then column 2, and then column 3.

   - Once you have completed your implementation, you can test your player against the random player in a tournament. Recompile the code (`javac *.java`) and run `java PokerSquares` again to see how the `FlushPlayer` performs against the `RandomPlayer` in a series of 10 games, with 3 different scoring systems.

   - Once your code is finished, you can submit it on Desire2Learn. While on the CS4725/CS6705 page on Desire2Learn, from the `Assessments` drop-down menu near the top of the page, choose `Assignments`. You should then go to the dropbox folder called `Assignment 2 - FlushPlayer` to submit your file.

2. **(10 marks)** Consider the search problem involving getting from A to J in the diagram pictured below. Edges are labelled with the cost associated with following that edge.

   In the questions below, break any ties alphabetically. For questions that ask for a path, please give answers in the form $A \rightarrow B \rightarrow D \rightarrow I \rightarrow J$.



   (a) Uniform cost search:

   - In what order would nodes be selected for expansion? Show the contents of the frontier after each step.
   - What solution path would be returned by the algorithm and what is its cost?

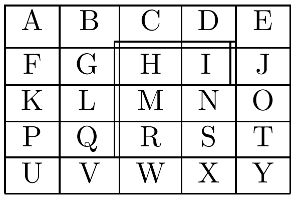
   (b) Greedy search, with the heuristic function $h(A) = 9$, $h(B) = 8$, $h(C) = 7$, $h(D) = 11$, $h(E) = 5$, $h(F) = 2$, $h(G) = 1$, $h(H) = 10$, $h(I) = 8$, $h(J) = 0$:

   - In what order would nodes be selected for expansion? Show the contents of the frontier after each step.
   - What solution path would be returned by the algorithm and what is its cost?

3. **(8 marks)** Consider the maze shown below, where the successors of a cell are the cells directly to the east, west, north and south of the cell, except that you are not allowed to pass through the thick wall indicated by the double line. For example, the successors of cell I are H and N (and not D or J).

   Trace the operation of **A* search** applied to the problem of getting from cell R to cell G. The heuristic function is just the Manhattan distance, ignoring the existence of the wall. The heuristic values for each node are summarized for you in the table to the right.

   - Draw the search tree, starting from cell R. Beside each node in your tree, indicate the f, g and h scores for the node (in the format $g(n) + h(n) = f(n)$). Assume that the cost of each move is 1.
   - To the side of your search tree, provide a list of the order in which the nodes are expanded. Show the contents of the priority queue after each node is expanded. To make it easier to keep track of what you are doing, include both the node's f value and h value as superscripts when you write a node in your priority queue. For example, $Z^{11,7}$ would indicate that Z is a node such that $f(Z) = 11$ and $h(Z) = 7$.
   - If two or more nodes are tied for the lowest f value, give priority to the one with minimum h value. Use alphabetical order if there is still a tie.
   - You can avoid generating states that have already been expanded. For example, you should not consider going west from cell I to cell H if you had already expanded cell H in an earlier step.
   - Also, when inserting a state onto the priority queue, if a copy of the state is already on the queue, just keep the copy with the lower f value.
   - You can stop as soon as you have found a path to G that you are certain is an optimal path.



cell	h(cell)	cell	h(cell)
A	2	N	3
B	1	O	4
C	2	P	3
D	3	Q	2
E	4	R	3
F	1	S	4
G	0	T	5
H	1	U	4
I	2	V	3
J	3	W	4
K	2	X	5
L	1	Y	6
M	2		

4. **(6 marks)** Consider a best-first search technique that uses the following evaluation function, where w is an adjustable weight value:

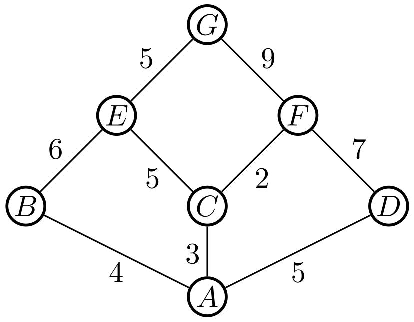
$$f(n) = w \cdot g(n) \; + \; (2 - w) \cdot h(n)$$

   (a) When $w = 0$, this search technique will behave exactly like which search algorithm we have studied in class? Explain.

   (b) When $w = 1$, this search technique will behave exactly like which search algorithm we have studied in class? Explain.

   (c) When $w = 2$, this search technique will behave exactly like which search algorithm we have studied in class? Explain.

5. **(6 marks)**



Consider the states shown above. A is the initial state and G is the goal state. Each edge is labelled by its cost. An important point for this question is that these edges can be traversed in both directions; for example, you can move from A to B, and you can also move from B to A.

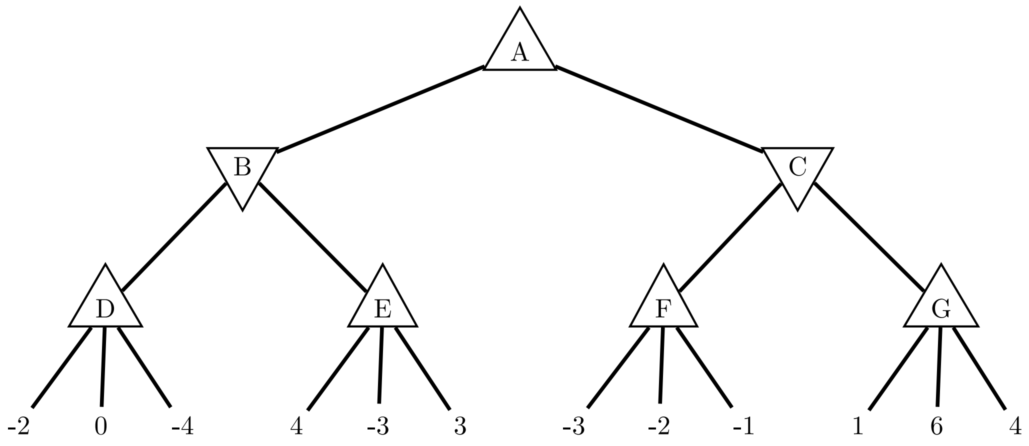
You have been given the heuristic function h below, but you were unable to read the value of $h(C)$. You can assume that it is a non-negative value.

Node n	A	B	C	D	E	F	G
$h(n)$	10	9	?	13	5	7	0

   (a) What range of values for $h(C)$ would make h an admissible heuristic? Explain your answer.

   (b) What range of values for $h(C)$ would make h a consistent heuristic? Think carefully about what the definition of *consistent* says. Explain your answer.

6. **(10 marks)** Consider the following game tree, where A, D, E, F, G are MAX nodes, and where B and C are MIN nodes.



(a) Use the minimax algorithm and the alpha-beta pruning algorithm to explore this game tree. For each node A-G, show the sequence of values that are assigned to the node's minimax value, to its α value, and to its β value, in a table such as this:

$\vdots$

X:	value:	~~$-\infty$~~ 2
	α:	~~$-\infty$~~ 2
	β:	~~∞~~ ~~8~~ 7
Y:	value:	~~∞~~ -3
	α:	7
	β:	~~∞~~ -3

$\vdots$

(b) Which nodes A-G and which children of nodes D-G are never visited because of alpha-beta pruning?

(c) What is the best move for MAX at node A?