1. [20%] General AI Knowledge and Application

For a)-h), give 1% for every correct answer.

Select the best answer by checking the boxes next to the questions (□). If you need to make a correction and want to clear a checked box, please fill it out completely (□).

		True	False	
a.	[1%]	☒	□	A model-based reflex agent selects the next action based on condition-action rules.
b.	[1%]	□	☒	DFS is complete if the graph search version is used. Think of infinite graphs.
c.	[1%]	☒	□	Minimax minimizes the possible loss for a worst-case scenario.
d.	[1%]	☒	□	All consistent heuristics are admissible.
e.	[1%]	☒	□	Determining whether there is a model for a CSP with finite domains is NP-hard.
f.	[1%]	□	☒	If the temperature decreases slowly enough simulated annealing is guaranteed to reach the optimal state. It is still random, we cannot *guarantee* it.
g.	[1%]	☒	□	Most current AI research devotes little attention to passing the Turing test.
h.	[1%]	□	☒	The beta value represents the upper bound on the value of a maximizing node.

Multiple Choice - Select all answers that apply to the question.

For i), j), k), give 1% for every correctly checked box.

i) [4%] If $f(s)$, $g(s)$ and $h(s)$ are all admissible heuristics then which of the following are also guaranteed to be admissible heuristics:

1) ☒ $f(s)6+g(s)3+h(s)2$

2) ☒ $\min\left(f(s),g(s),h(s)\right)$

3) ☒ $\max\left(f(s),g(s),h(s)\right)$

4) ☐ $f(s) \cdot g(s) \cdot h(s)$

The expression must be less than or equal to the max of the heuristics, i.e. sums and products do not satisfy these. The weighted average is less than the max. Min and max are admissible.

j) [4%] Which statements on constraint satisfaction problems (CSP) are TRUE:

1) ☒ The minimum remaining values heuristic chooses the most constrained variable first.

2) ☒ Backtracking is an uninformed search algorithm.

3) ☐ It is not sufficient to only consider assignments to a single variable at every step of backtracking.

4) ☐ The number of complete assignments is in the order of $O(nd)$ for n discrete variables and domain size d.

k) [4%] Which statements on games and game theory are FALSE:

1) ☒ Poker is a deterministic game where some information is hidden to the players. Poker is nondeterministic.

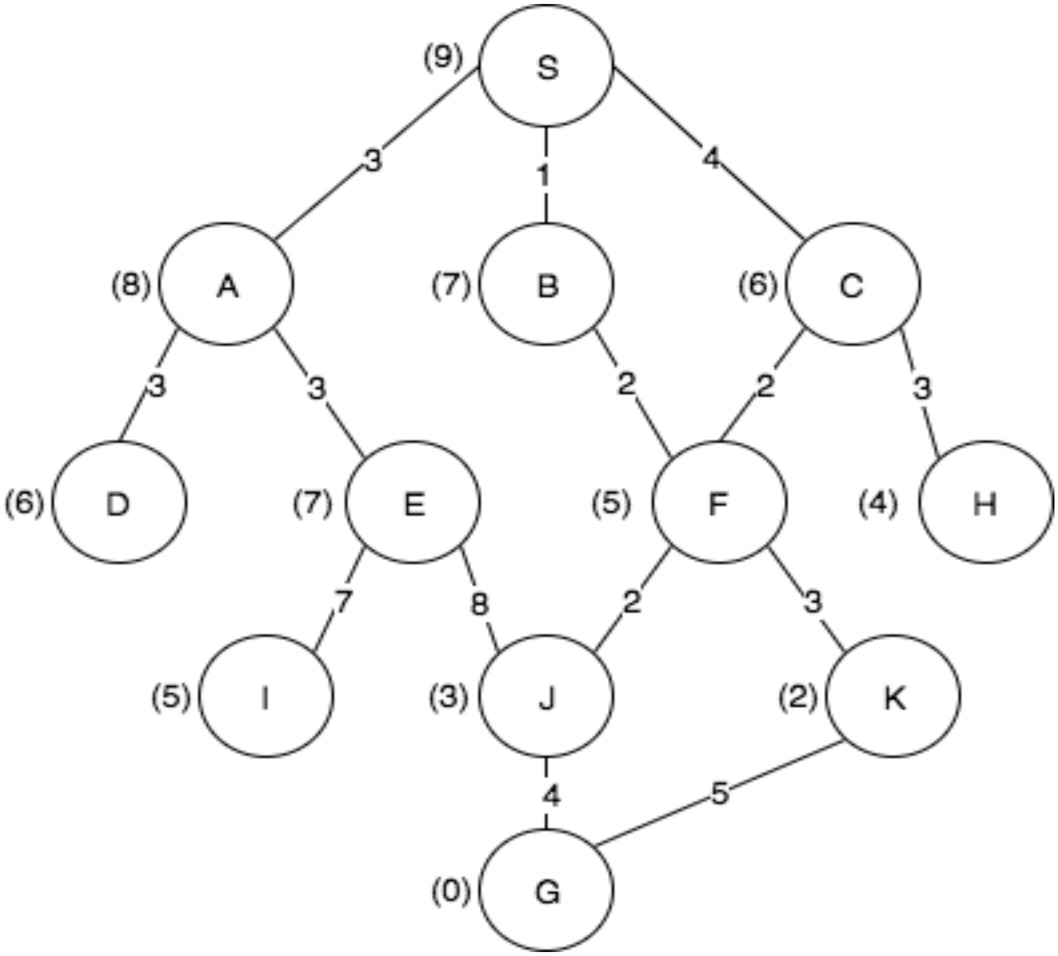
2) ☐ Minimax is an optimal search algorithm.

3) ☒ When computed by expectimax, a player's behavior is the ~~same~~ under any monotonic transformation of the state evaluation function. Lecture 5, p 67

4) ☐ The value of a chance node is computed by the sum of the children's values, each multiplied by their respective probabilities.

2.[30%] Graph Search

Consider the following graph. The start node for the graph search is S, and the goal node is G. The cost of each edge is shown on the edge. The heuristic value for each node is shown within parentheses next to that node.



Using graph search, for each of the following search algorithms, show the order in which the nodes are expanded. In case of a tie in frontier, pick the node that is leftmost in the graph. Terminate the search once the goal node is reached.

2A. [6%] Breadth-First Search

[4%]Nodes Explored: SABCDEFHIJKG

[2%]Path to goal : SAEJG

Each wrong letter got -1

Ex:

If the student's answer: SBACDEFHIJKG

Then -2

If the student's answer: SBACDHIJKG

Then -4

2B. [6%] Depth-First Search

[4%]Nodes Explored: SADEIJG

[2%]Path to goal: SAEJG

2C. [6%] Uniform Cost Search

[4%]Nodes Explored: SBAFCJDEKHG

[2%]Path to goal:SBFJG

2D. [6%] Greedy best-first Search

[4%]Nodes Explored:SCHFKG

[2%]Path to goal:SCFKG

2E. [6%] A* Search

[4%]Nodes Explored: SBFJKG
[2%]Path to goal:SBFJG

3.[20%] Game Playing

We have a 3×3 game board, and two players. Each player takes turns as in chess or tic-tac-toe. That is, player 1 takes a move, then back to player 2, then back to player 1 and so forth. Each square has a fixed point value as shown below in (a). The objective of the game for each player is to score the most points, i.e. the total point value of all their occupied squares. The goal is to capture the squares worth the most points with the constraint that the square is adjacent to the opponent’s last move.

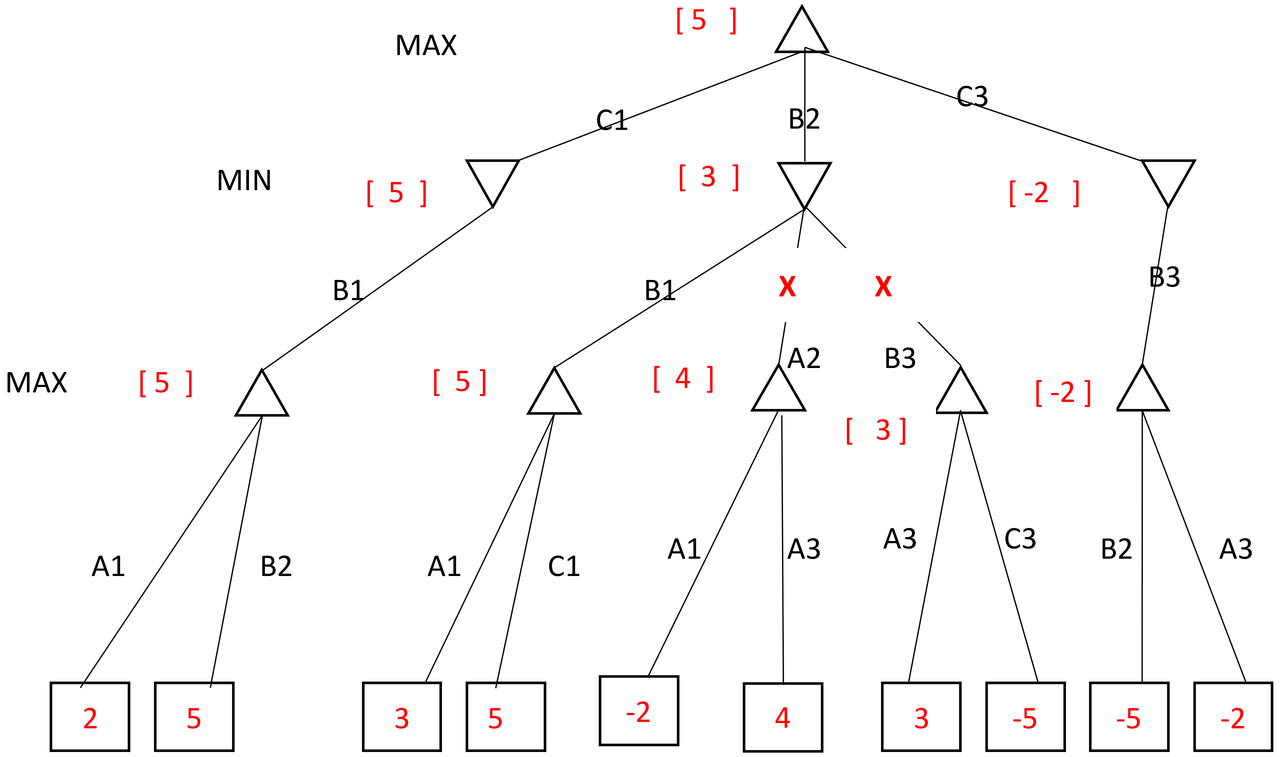
For example, we have player 1 (represented with x) as the MAX player, and player 2 (represented with o) as the MIN player. Initially, we have the board below, as shown in (b). As you can see, square C2 is occupied by player 2. Player 1 now can only take one of the squares next to square C2 (up, down, left, right, but not diagonally) if it is unoccupied. For example, in (b), player 1 could only take C1, B2, C3 in the next move. **The evaluation is defined as (total_points_of_MAX-player) – (total_points_of_MIN-player)**

	1	2	3
A	3	7	9
B	2	6	8
C	5	4	1

(a)

	1	2	3
A			
B			
C		o	

(b)



Note that, either the above or (all values above+4, i.e. if do not include the initial move's score, still give full credit.)

3A. [8%] Fill the evaluation value in the leaf node of the game tree above. Note that the actions taken for each player are denoted on the edge of the game tree.

Each incorrect -1, until 0

3B. [8%] Perform the minimax algorithm on the game tree, without pruning. Show the final min/max values in the bracket [ ] next to each node in the tree.

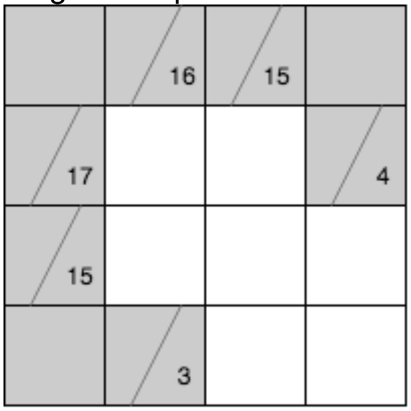
Each incorrect -1, until 0

3C. [4%] Now with αβ pruning, please show which edges are pruned by marking X on the game tree above.

Each correct gets 2 pt. And each incorrect -1, until 0. (if edge to -2,4,3, -5 are marked, no points deducted.)

5. [20%] Kakuro and Constraint Satisfaction

Kakuro is a logic puzzle and is sometimes referred to as mathematical crossword. Just like a crossword, it consists of a grid of squares. For example, take the puzzle below:



A set of connected horizontal or vertical blank squares is called a block. The goal is to fill out all the squares with digits from 1 to 9 in a way that the some of the squares in each block equals the number at the end of the block. For instance, in the above puzzle squares at positions ([row number, column number]) [4, 3] and [4, 4] form a horizontal block and their sum should equal 3. Also, you cannot use a digit more than once in a block. For example, you cannot use digit 2 twice in the vertical block above leading to 4, to get sum 4.

5A. [10%] Formulate the puzzle above as a CSP with variables, domains, and constraints. Draw the constraint graph.

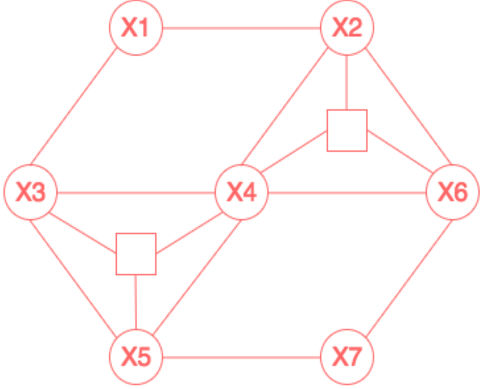
We consider a variable for each of the blank squares: X1, X2, X3, X4, X5, X6, X7. [1%, No Partial Credit]

The domain for all the variables is {1, 2, 3, …, 9}. [1%, No Partial Credit]

Constraints: [6%] [2% Inequality Constraints] [4% Sum Constraints]

X1 + X3 = 16, X1 != X3
X1 + X2 = 17, X1 != X2
X2 + X4 + X6 = 15 , X2 != X4, X4 != X6, X2 != X6
X3 + X4 + X5 = 15 , X3 != X4, X4 != X5, X3 != X5
X5 + X7 = 4, X5 != X7
X6 + X7 = 3, X6 != X7

Constraint Graph: [2%] [1% binary constraints] [1% 3-ary constraints]



Most common mistakes:

- Not paying attention to 3-ary constraints involving X4 and therefore not including square boxes in the graph
- Including Alldiff constraint for all of the variables, while only variables in the same block should be unequal

5B. [7%] Show the domains of the variables after running arc-consistency.

Pay attention that arc-consistency only involves binary constraints.

X1 = {9}, X2 = {8}, X3 = {7}, X4 = {1, 4, 5, 6, 9}, X5 = {3}, X6 = {2}, X7 = {1}

[Each Worth 1%]

Most common mistakes:

- Not applying arc-consistency thoroughly and as a result ending up with domains such as X1 = {9, 8}
- Not considering the fact that arc-consistency only deals with binary constraints and as a result ending up with the domain X4 = {5}

5C. [1%] Give the solution to the CSP.

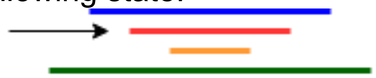
X1 = 9, X2 = 8, X3 = 7, X4 = 5, X5 = 3, X6 = 2, X7 = 1 [1%, No Partial Credit]

5D. [2%] Assume one wants to solve Kakuro using search. Would you suggest using classical search or local search? Explain the reasoning behind your suggestion.

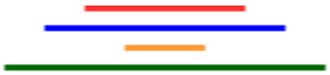
Local search [1%] is the correct suggestion, because the path to the goal is not relevant and only the final state is important [1%]. However, classical search returns a path.

6. [10%] Pancakes and Heuristics

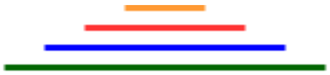
We have a stack of pancakes and want to sort them in order (largest to smallest from bottom to top) by doing a number of flips. For each flip, we choose a point in the stack and flip. Each flip reverses the order of all the pancakes above the point we chose. For example if the stack is in the following state:



And we choose to flip the above two pancakes (where the arrow is placed), we get to the following state:



The goal state is:



6A. [4%] What's the size of the state Space? What's the number of the successors to each state?

Size of the state space: 4! [2%, No Partial Credit], Number of the successors: 4 [2%, No Partial Credit]

6B. [3%] Assume we assign cost 1 to each flip, so the path cost to the goal state is the number of the total number of the flips we've done. Now consider the following heuristic function:

h(s) = number of pancakes that are not in the right place.

Is h admissible? Why or why not?

No, h is not admissible [1%]. For example, the heuristic value for the following state is 4; however, the path cost to the goal is 1 [2%, Any Correct Reasoning].

6C. [3%] Now assume the cost of each flip is equal to the number of the pancakes that get flipped. So the cost of the flip in the example at the beginning is 2. With this modification, is the heuristic function in part 6B admissible?

Yes, it is admissible. [3%, No Partial Credit]