Northeastern University
CS5100 – Foundations of Artificial Intelligence
Fall 2022, Richard Hoshino

Course Synthesis #1

Name: _____

Problem	Points
Short Answer Questions	/10
Problem #1 - Northerners & Easterners	/8
Problem #2 - Mastermind Solver	/8
Problem #3 - The 8-Puzzle	/8
Problem #4 - United Way Volunteering	/8
Problem #5 - The Odd Cycle Game	/8
Problem #6 - The Coin-Taking Game	/8
Problem #7 - Graph Colouring	/8
Total	/50

Instructions

- This Synthesis will be marked out of 50. You are to answer all of the Short Answer Questions, worth up to 10 marks. For the seven Full Solution Problems, you are welcome to attempt as many of them as you wish, but only your *top five* will be counted. For example, if you get 7 points on the Short Answer Questions and your marks on the seven Problems are 8, 0, 6, 6, 8, 3, 5 then your grade will be 7+8+6+6+8+5 = 40 out of 50, since your lowest two scores will be dropped.
- Think of this Course Synthesis as a week-long individual take-home exam where you may consult your notes, the course textbook, anything on the Canvas Page, and any websites linked from the Canvas Page. However, you may NOT consult your classmates or look at **any** other online resources unless explicitly approved by me beforehand. Please post your questions on the Canvas Discussion Forum if you would like any clarifications or hints.
- Submit one .pdf file with your responses to the Short Answer Questions, and then individual files (e.g. in .pdf/.py/.java/.c) for each of the full-solution Problems you attempt.

(10 pts.) Short Answer Questions

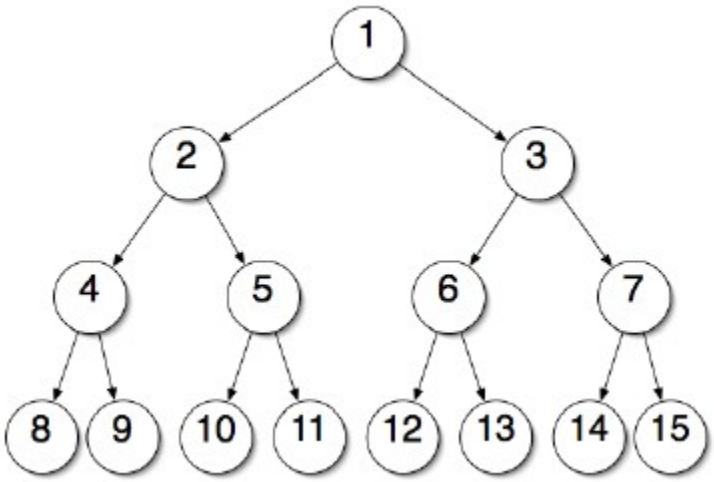
Hand in a single .pdf file (ideally one page long) on which you will provide your answers to the 10 questions below.

All you need to do is submit your final ANSWERS to these 10 questions. No additional work is required or requested. No justification is required – all I want is your final answer.

Each correct answer will be worth 1 points. For each incorrect answer, the TA will decide whether your response will be awarded 0.5 points (for an answer that is almost correct) or 0 points.

(1) Consider a state space, where each state is a positive integer. The start state is 1. For all positive integers $k \geq 1$, state k has two successors: $2k$ (Left) and $2k+1$ (Right).

This state space can be represented as a graph.



Determine the sequence of Left and Right moves that reaches the goal state of 1365.

(2) Let $n \geq 3$ be a positive integer. Consider a network of n cities, with each pair connected by a road. The Travelling Salesperson Problem (TSP) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"

One can use a Local Search algorithm (e.g. simulated annealing, local beam search, genetic programming) to find a solution to the TSP.

Richard makes this bold claim: no matter what Local Search algorithm we use, our solution is guaranteed to be within 1% of the optimal solution. (In other words, if the optimal solution is x miles, then our local search solution is at most $1.01x$ miles).

Determine whether this claim is TRUE or FALSE.

(3) Consider the following two-player game: Jinhao and Luyi take turns placing a rook on an empty square of an n by n chessboard. Jinhao moves first.

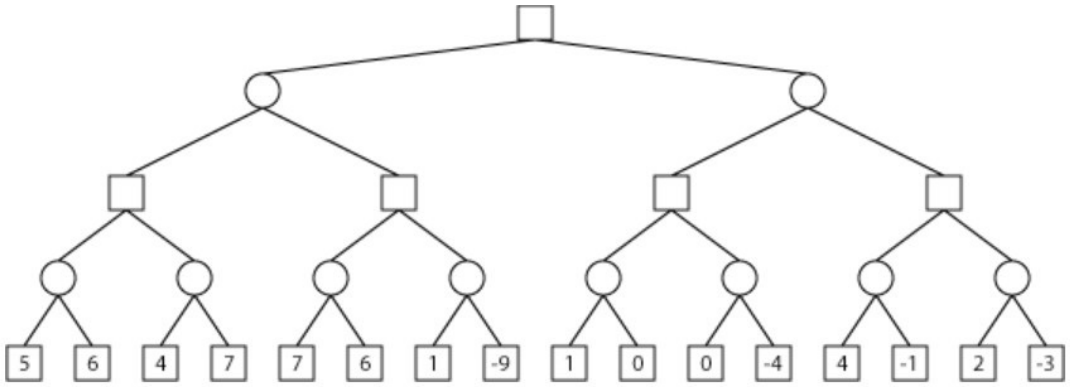
The game ends as soon as a player has made a move that makes two rooks attack each other, i.e., they lie on the same row or column. And that player LOSES the game. For example, if Jinhao's first move is b1 and Luyi's move is b3, then Luyi loses the game because b1 and b3 lie on the same column. Thus, Jinhao wins the game.

Assume both players are rational agents, i.e., both play optimally. If $n = 5$, determine who will win the game. Answer either JINHAO or LUYI.

(4) Consider the same "attacking rooks" game above, except this time $n = 2022$. If $n = 2022$, determine who will win the game. Answer either JINHAO or LUYI.

(5) Consider an adversarial search scenario, i.e., a two-player game. Suppose the two players are Maximus and Minnie, both rational agents playing their optimal strategies.

In the game tree below, the square nodes represent Maximizer nodes, and the circular nodes represent Minimizer nodes.



Determine the value of this game, i.e., the value of the node at the top of this tree.

(6) In the game tree above, there are 16 leaf nodes (i.e., nodes at the bottom of the tree).

Alpha–Beta pruning is a powerful technique that decreases the number of nodes examined by the Minimax algorithm. By pruning branches that cannot be part of the optimal path, many of these leaf nodes are left unexamined.

Determine how many of these 16 leaf nodes are unexamined because their branches get pruned.

Assume that the Alpha-Beta search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.

(7) In a Cryptarithmetic problem, we are given an equation where each letter represents a different digit. For example, the equation SEND + MORE = MONEY has a unique solution:

$$\text{O} = 0, \quad \text{M} = 1, \quad \text{Y} = 2, \quad \text{E} = 5, \quad \text{N} = 6, \quad \text{D} = 7, \quad \text{R} = 8, \quad \text{S} = 9.$$

You can double-check that the solution is correct, since 9567 + 1085 = 10652.

In a Cryptarithmetic problem, the first digit of a multi-digit number cannot be zero, and there must be a unique solution.

There is a unique solution to the following Cryptarithmetic problem:

$$\text{FORTY} + \text{TEN} + \text{TEN} = \text{SIXTY}.$$

Determine the digit Y.

(8) In the game of Minesweeper, your goal is to identify all of the land mines by putting a flag on each one. For each square, if there is a number on that square, this number tells us how many mines are adjacent to that square.

The diagram below shows a game that is nearly complete. All mines have been identified, except for three. These unidentified mines are in three of the following squares: A, B, C, D, E, F.

For example, in the diagram above, the number 6 shows that there are six mines that touch this square. Since five of the six mines have already been identified, this means that either D or E is a mine, but not both. As you can see, Minesweeper is an example of a Constraint Satisfaction Problem.

Determine the squares containing the three unidentified mines.

(9) Here is a map of the United States.

You are given four colours: Apple (A), Brown (B), Crimson (C), Denim (D). Your task is to colour the 50 states, with each state being assigned one of these four colours. There is only one rule: if two states share a common border, they cannot receive the same colour.

You are given one penalty point for every state whose colour is D. Richard makes the following claim: 1 is the minimum number of penalty points in a valid colouring of the 50 states.

Determine whether this claim is TRUE or FALSE.

(10) One hundred employees from Microsoft have agreed to volunteer their time for United Way, to support an exciting new initiative that will last for exactly one hundred days. Each employee will volunteer on exactly one day, and each day will be represented by exactly one employee.

Each employee marks a preference for volunteering on a given day, ranking their preferences on a scale from 1 (least desired) to 100 (most desired). The higher the score, the better.

These preference scores are given in the Excel sheet found in the "Notes from Class" page on Canvas, titled "Input Excel File for Course Synthesis 1".

Your job is to assign employees to days, subject to the given constraints, while maximizing the sum of the preference ratings. The *score* of your solution is the sum of the 100 preference ratings corresponding to your assignment.

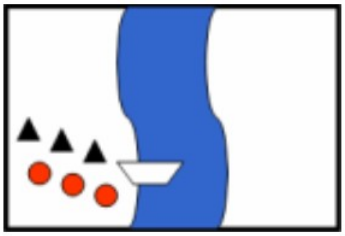
Determine the maximum possible score.

(8 pts.) Problem #1 - Northerners & Easterners

In this problem, you will solve a famous AI puzzle.

There are six human beings. Each is a "Northerner" or an "Easterner".

Three Northerners and three Easterners are on the left bank of a river, along with a boat that can hold one or two people.



Your goal is to get all six people to the right bank of the river without ever having a group of Northerners on one side outnumbered by the Easterners on that side. (If this scenario ever occurs, the Easterners would kill the Northerners.)

For example, if one side has 1 Northerner and 2 Easterners, then that is not acceptable. However, it is okay for one side to have 0 Northerners and 2 Easterners.

Forbidden positions are those where one side has N Northerners and E Easterners, where $E > N$ and $N \geq 1$. Legal positions have the following property: on both sides of the river, if $N \geq 1$ then $N \geq E$.

Note that the boat cannot cross the river by itself with no people on board.

Make sure you clearly describe each "state" in this problem. One idea is to draw a picture for each state, showing the location of the boat and the number of Northerners (N) and Easterners (E) on each side of the river.

Another option is to represent each state using the three-tuple (N, E, B), where N is the number of Northerners on the left bank, E is the number of Easterners on the left bank, and B is the number of boats on the left bank. For example, the above diagram would correspond to $(3, 3, 1)$.

(a) Clearly explain why this is an AI search problem. To do this, identify the initial state, actions, successor function, goal test, and path cost, writing one sentence for each response.

(b) Draw a diagram of the complete state space of legal positions. Ignore repeated states, i.e., no state should appear in your diagram more than once.

(c) Using your diagram in part (b), solve the Northerners and Easterners problem! Describe how to get all six individuals to the right bank of the river.

(8 pts.) Problem #2 - Mastermind Solver

In this question, you will create a computer program that will play the AI Search game Mastermind. You may use the programming language of your choice.

This time, the roles are reversed. You will input a hidden code and your Python program will attempt to guess the code in as few moves as possible.

Your program's input is a hidden code, where the code consists of four *distinct* colours chosen from the set {A,B,C,D,E,F}. For example, EFBC is a valid code, but EFBF and AAAA are not. There are $6 \times 5 \times 4 \times 3 = 360$ possible codewords.

In each turn, your computer program guesses the hidden code. Note that each of the guesses must be a possible code: thus EFBC is a valid guess, but EFBF and AAAA are not.

After your program makes a guess, it will compare the guess to the hidden code (your input), outputting a red peg for each colour in the correct position, and a white peg for each colour in the wrong position. For example, suppose your input is EFBC. Then here is one possible output for your game.

```
Guess is ABCD                0 Red, 2 White
Guess is ABDC                1 Red, 1 White
Guess is FEBC                2 Red, 2 White
Guess is EFBC                4 Red, 0 White
Solved in 4 guesses
```

Your computer program needs to contain the following.

(a) The function Mastermind(Code) that inputs a hidden code (e.g. EFBC) and returns an output such as the one above, listing the guesses, the number of red/white pegs for each guess, and the total number of guesses needed to identify the code. You are welcome to format the output in any way that you wish.

(b) A clear explanation of HOW your Mastermind program chooses each of its guesses. Include this information somewhere in your program.

(c) The overall results of your Mastermind algorithm. Run your algorithm on each of the 360 possible codewords and output the minimum, maximum, and average number of guesses needed.

```
The minimum number of guesses is 1
The maximum number of guesses is 10
The average number of guesses is 8.73
```

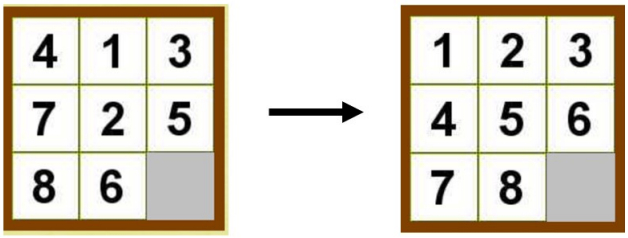
You will get full marks for any well-documented program for which the maximum number of guesses is at most 10.

(8 pts.) Problem #3 - The 8-Puzzle

A* Search is a powerful informed search algorithm. For each node n, we define the evaluation function $f(n)$ to equal $g(n) + h(n)$, where $g(n)$ is the actual cost from the initial state to state n, and $h(n)$ is the estimated cost from state n to the goal state. In other words, $f(n)$ is the *estimated cost* of the cheapest solution through state n, and at each step, we expand the node that has the lowest value of $f(n)$.

(a) Show that any heuristic function $h(n)$ that is *consistent* must also be *admissible*. Either provide a formal mathematical proof or provide a clear explanation of why consistency implies admissibility.

(b) Recall the 8-puzzle that we explored earlier in this course. The goal is to place the tiles in order by making sliding moves that use the empty space. The desired goal state is indicated on the right.



For each state n, let $h(n)$ be the heuristic function corresponding to the sum of the "Manhattan distance" of the tiles from their goal positions. In the initial state, $h(n) = 1+1+0+1+1+2+1+1 = 8$, since six of the seven misplaced tiles are one move away from their destination, while Tile #6 is two moves away. And in the goal state, $h(n) = 0+0+0+0+0+0+0+0 = 0$.

Clearly justify why $h(n)$ is an admissible heuristic function.

(c) For each game state, list the eight numbers in order, from top left to bottom right. (In the above problem, the initial state is 41372586). For any given state, we define the following "score": count one point for each pair (i, j), with $1 \leq i < j \leq 8$, for which j appears before i in the initial state.

In the above problem, the score of the initial state is 8, as the following pairs appear in the wrong order: $\{(1,4),(3,4),(2,4),(2,3),(2,7),(5,7),(6,7),(6,8)\}$.

For each game state n, let $h(n)$ be the "score", based on the above definition.

Is $h(n)$ an admissible heuristic function? Clearly justify your answer.

(8 pts.) Problem #4 - United Way Volunteering

Four employees from Microsoft have agreed to volunteer their time for United Way, to support an exciting new initiative that will last for exactly four days. Each employee will volunteer on exactly one day, and each day will be represented by exactly one employee.

Each employee marks a preference for volunteering on a given day, ranking their preferences on a scale from 1 (least desired) to 100 (most desired). The higher the score, the better. Here were the results.

	Day 1	Day 2	Day 3	Day 4
Bill	60	80	90	70
Paul	80	90	100	80
Satya	60	50	90	50
Xiaofeng	70	60	50	80

Your goal is to assign one employee to each day, to maximize the total preference score. You must ensure that each day is covered by exactly one employee.

For example, suppose you decide on the following assignment: (Bill, Day 1), (Xiaofeng, Day 2), (Paul, Day 3), (Satya, Day 4). Then the total preference score is $60 + 60 + 100 + 50 = 270$.

(a) Using any method of your choice (e.g. logical reasoning, analysis of cases, constraint optimization), determine the optimal assignment that maximizes the total preference score. Clearly show your steps and justify that your solution is optimal.

(b) Suppose you apply one of many possible **local search** algorithms to improve this initial assignment of (Bill, Day 1), (Xiaofeng, Day 2), (Paul, Day 3), (Satya, Day 4). Pick any local search algorithm of your choice, explain how it works, and apply your algorithm to iteratively improve this initial assignment of players to positions until no further improvements are possible. Does your local search algorithm output the optimal assignment?

(c) Suppose we have $n = 100$ employees and $n = 100$ days. Download the Excel sheet found in the "Notes from Class" page on Canvas, titled "Input Excel File for Course Synthesis 1". This is the same file from Question #10 of the Short Answer questions.

Using this data set, apply your local search algorithm from part (b) to make the total preference score as large as possible. Write a computer program to solve this problem, in the programming language of your choice.

For your solution to part (c), all you need to do is output your total preference score. Submit your computer program as a separate file (.py, .java, .c, etc.). You will receive full credit for any solution that is within 2% of the optimal solution, and partial credit otherwise.

NOTE: your algorithm **must** be local search and **not** constraint optimization!

(8 pts.) Problem #5 - The Odd Cycle Game

Let n be a positive integer, with $n \geq 3$. On a sheet of paper are n points, arranged in a circle.

Jinming and Szeka play a game called the Odd Cycle Game. They take turns playing, with Jinming always moving first.

On their turn, the player draws a line between any pair of points that have not yet been connected. The first player to complete an *odd cycle* loses the game.

NOTE: an odd cycle refers to any cycle of length $3, 5, 7, 9, \ldots$, where each vertex in the cycle is among the n given points.

It is easy to see that this game cannot end in a tie, since $n \geq 3$. Suppose, on the contrary, that the game has finished in a tie, with no odd cycles formed. Consider points P, Q, R. If the game has ended, then lines PQ, QR, and RP have all been connected, and these three lines form the odd cycle $P - Q - R - P$. So whoever completed that odd cycle must be the loser of this game.

Since this is a perfect information game that cannot end in a tie, one of the two players *must* have a winning strategy. This is because the Minimax value at the top of the game tree must be $+1$ or -1, which implies that one of the players can guarantee victory if they play perfectly.

In each of the questions below, you will determine which player has a winning strategy, using any method of your choice (e.g. logical reasoning, minimax tree analysis, etc.) In your solution you need to clearly explain why your strategy guarantees victory for that player regardless of how well their opponent plays.

(a) Determine which player has a winning strategy for the game with $n = 4$ points.

(b) Determine which player has a winning strategy for the game with $n = 5$ points.

(c) Determine which player has a winning strategy for the game with $n = 6$ points.

(d) **OPTIONAL BONUS:** (+1 mark) Determine which player has a winning strategy for the game with $n = 2022$ points.

(8 pts.) Problem #6 - The Coin-Taking Game

Maximus and Minnie play the Coin-Taking Game, where the game starts with n coins on the table, for some positive integer n.

They take turns playing, with Maximus always moving first.

On their turn, each player removes 1 coin or 2 coins from the board. Whoever removes the last coin wins the game.

The value of the game is +1 to the winner, and -1 to the loser.

Let $v(n)$ be the value of the game to Maximus, when the game starts with n coins.

(a) Suppose that both players play optimally. Using any method of your choice (e.g. logical reasoning, minimax tree analysis, etc.), explain why $v(n) = -1$ when n is a multiple of 3, and why $v(n) = 1$ when n is not a multiple of 3.

(b) Suppose that Maximus plays optimally, but Minnie plays randomly – i.e., on each move, she removes one coin with probability $\frac{1}{2}$, and removes two coins with probability $\frac{1}{2}$. The only exception is if there is only 1 coin left on the board, in which case Minnie will always take the 1 remaining coin and win the game.

Using any method of your choice (e.g. logical reasoning, expectimax tree analysis, etc.), determine the values of $v(n)$ for each of $n = 1, 2, 3, 4, 5, 6$.

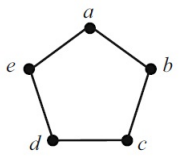
(c) Suppose that Maximus plays optimally, but Minnie plays randomly, as in part (b).

Using any method of your choice, determine the smallest integer n for which $v(n)$ is more than 0.999 and less than 1.

NOTE: I encourage you to solve part (c) without writing a computer program. However, if you are unable to do so, you may write a computer program to solve part (c), in the programming language of your choice. Make sure your code is well-documented so that the TA can follow your solution.

(8 pts.) Problem #7 - Graph Colouring

Given a graph G, we say that G is k-colourable if every vertex of G can be assigned one of k colours so that for every pair u, v of adjacent vertices, u and v are assigned different colours.

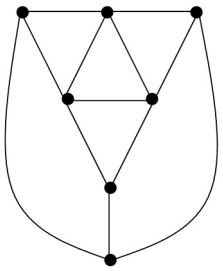


For example, the above graph G is 3-colourable, since we can assign colour 1 to vertices a and c, colour 2 to vertices b and d, and colour 3 to vertex e. It is easy to show that this graph is not 2-colourable.

The *chromatic number* of a graph G, denoted by $\chi(G)$, is the smallest integer k for which graph G is k-colorable. In our example above, $\chi(G) = 3$.

To show that $\chi(G) = k$, you must show that the graph is k-colourable and that the graph is not $(k-1)$-colourable.

(a) Let G be the graph below, with 7 vertices and 12 edges. Clearly explain why $\chi(G) = 4$.



(b) Clearly explain the difference between a Constraint Satisfaction Problem (CSP) and a Constraint Optimization Problem (COP). Explain why determining $\chi(G)$ is a COP and not a CSP.

(c) Write a computer program that determines $\chi(G)$ for the graph in part (a). You may use the programming language of your choice.

To do this, define $X_{i,j}$ to be the binary variable that equals 1 if vertex i is assigned to colour j, and is equal to 0 otherwise. Define Y_j to be the binary variable that equals 1 if colour j is assigned to at least one of the vertices, and is equal to 0 otherwise.

In your program, you need to create the *objective function* as well as list out all of the *constraints* for the 7-vertex 12-edge graph in part (a). Then you can call the Google OR-Tools command that will optimally solve your Integer Linear Program.

All you need to do in part (c) is submit your completed program (e.g. in .py, .java, .c), where your program correctly determines that the minimum number of colours needed is 4. Make sure you clearly document your code so that the TA can understand your solution.