CSC242: Introduction to Artificial Intelligence
Homework 1: AIMA Chapter 1

1. What are key differences between Type I and Type II intelligence, and examples of the operation of each?

2. Identify the five components of a well-defined state-space search problem.

3. Consider the task of solving Rubic's Cube.
   (a) Formulate solving Rubic's Cube as a state-space search problem.

4. Your goal is to navigate a robot out of a maze built on a $n \times n$ grid with walls separating the corridors. You are given a map of the maze at the outset. The robot starts in the center of the maze facing north. You can turn the robot to face north, south, east, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.
   (a) Formulate this as a state-space search problem. How large is the state space?
   (b) In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate the problem using this observation. How large is the state space?
   (c) From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation? Does this affect how the plan we find gets executed?
   (d) Suppose that instead of a grid, the maze is defined on a square from $[-1, -1]$ to $[1, 1]$ on the Cartesian plane. The positions of the walls and of the robot are arbitrary real coordinates. Reformulate the problem. How big is the state space?

5. Consider a state space where the start state is number 1 and each state k has two successors numbered $2k$ and $2k + 1$ respectively.
   (a) Draw the portion of the state space for states 1 to 15.
   (b) Suppose the goal state is 11. List the order the nodes will be visited for breadth-first search, depth-first search to a depth limit of 3, and iterative deepening depth-first search.

6. For a search tree, let b be the branching factor, d the depth of the shallowest ("best") solution, and m the maximum depth of the tree. Complete the following table:

	Breadth-first search	Depth-first graph search	Depth-first tree search	Iterative-deepening depth-first tree search
Time Complexity				
Space Complexity				
Complete?				
Optimal?				

7. Trace the operation of A* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. Show the sequence of nodes and the g, h, and f scores for each node.

8. (This is harder than the previous problems. Think hard and discuss with fellow students / friends / enemies / frenimies until you solve it!) Prove that if a heuristic h never overestimates by more than c, then A* using h returns a solution whose cost does not exceed that of the optimal solution by more than c. This is a useful thing to know in practice.

9. True or false: Local search is a form of systematic search.

10. True or false: Local search is guaranteed to find the optimal solution to a problem.

11. What is the minimum number of states that a local search algorithm must keep track of? What is the maximum?

12. As you probably already know, a Traveling Salesperson Problem (TSP) for a set of cities involves finding the shortest tour (path) that visits each city exactly once. In the classic version of the problem, you are given a symmetric matrix of distances between cities, and all cities are connected (*i.e.*, they form a complete graph). Variations involve having actual road networks (limited connectivity) and asymmetric distances.

    Formulate the classic TSP as a local search problem.

13. Suppose you have a set of Boolean (true/false) variables, and a set of rules of the form "if x_1 is true, then x_2 must be false," or "if both x_1 and x_2 are true, then either x_3 or x_4 are true," and so on. Your goal is to find values (true or false) for each of the Boolean variables (x_i) such that all the rules are satisfied.

    Formulate this so-called Boolean Satisfiability Problem as a local search problem.